

Microolap EtherSensor

Administrator's Guide

Contents

1. Microolap EtherSensor.....	1
1.1. Brief description of features.....	3
1.2. EtherSensor scope of application.....	5
1.3. System requirements.....	7
1.4. Description of operation.....	8
1.5. Administrator skills.....	10
1.6. List of operational documents.....	10
1.7. About EtherSensor PCAP Edition.....	11
1.8. Delivery of traffic analysis results.....	11
2. Microolap EtherSensor installation and configuring.....	11
2.1. Content of the software distribution kit.....	12
2.2. Connection Microolap EtherSensor to Ethernet.....	13
2.2.1. Management interface	13
2.2.2. Listening Interface	14
2.2.3. Switch configuration	14
2.3. Using Third-Party Information Security Tools.....	15
2.4. Sensor settings.....	16
3. Sources of data and metadata.....	16
3.1. EtherSensor PCAP service.....	18
3.1.1. EtherSensor PCAP settings	20
3.1.2. EtherSensor PCAP configuration file	21
3.1.3. EtherSensor PCAP packet filters	27
3.2. EtherSensor EtherCAP service.....	27
3.2.1. EtherSensor EtherCAP settings	31
3.2.2. EtherSensor EtherCAP configuration file	33
3.2.3. EtherSensor EtherCAP packet filters	39
3.3. EtherSensor ICAP service.....	43
3.3.1. EtherSensor ICAP settings	44
3.4. EtherSensor LotusTXN service.....	46
3.4.1. EtherSensor LotusTXN settings	48
3.5. EtherSensor Identity service.....	49
3.5.1. EtherSensor Identity settings	52
3.5.1.1. Domain Controllers.....	52
3.5.1.1.1. Alternative: logon script.....	54
3.5.1.1.2. DNS servers.....	55
3.5.1.1.3. DNSBL servers.....	56
3.5.1.1.4. Authentication Logs.....	58
3.5.1.1.4.1. Lua scripts.....	62
3.5.1.1.5. EtherSensor Agens server.....	62
3.6. EtherSensor Agent.....	63

3.6.1. System requirements for the Agent	63
3.6.2. Agent Installation	64
3.6.3. Agent files	65
3.6.4. Logic modules of the Agent	66
3.6.5. Data transferred to EtherStat	67
3.6.6. Data transferred to EtherSensor	70
3.6.7. Working with the Agent	71
3.6.7.1. Possible options for the Agent's work.....	73
3.6.7.2. Configuration of the EtherSensor Agent service.....	74
3.6.7.3. Agent work logging.....	77
3.6.7.4. Problems and solutions.....	80
4. Analysis of events and objects.....	81
4.1. EtherSensor Analyser settings.....	83
4.1.1. Pre-filtration	85
4.1.2. Detection and normalization of events	87
4.1.2.1. INCLUDE: Lua scripts functions.....	89
4.1.3. Final filtration	90
4.2. Generated events/messages.....	92
4.3. Filtering interception results.....	99
4.3.1. Filtration basics	99
4.3.1.1. Filer configuration.....	101
4.3.1.2. Table.....	102
4.3.1.3. Rules.....	102
4.3.1.3.1. Criteria and conditions.....	103
4.3.1.3.1.1. Condition ALL, *	106
4.3.1.3.1.2. Condition DETECTOR.....	107
4.3.1.3.1.3. Condition PROTOCOL.....	108
4.3.1.3.1.4. Condition MSG-SIZE, TOTAL-SIZE.....	109
4.3.1.3.1.5. Condition CHECK-MD5.....	110
4.3.1.3.1.6. Condition CHECK-MESSAGE-ID.....	111
4.3.1.3.1.7. Condition HOSTNAME.....	112
4.3.1.3.1.8. Condition IP.....	113
4.3.1.3.1.9. Condition HEADER.....	115
4.3.1.3.1.10. Condition ATTACH-NAME.....	117
4.3.1.3.1.11. Condition ATTACH-EXIST.....	119
4.3.1.3.1.12. Condition TAG.....	119
4.3.1.3.1.13. Condition FROM, TO, CC, BCC, ADDRESS, SUBJECT.....	121
4.3.1.3.1.14. Condition TEXT.....	124
4.3.1.3.2. Actions.....	126
4.3.1.3.2.1. Action ACCEPT.....	127
4.3.1.3.2.2. Action DROP.....	128

4.3.1.3.2.3. Action JUMP	129
4.3.1.3.2.4. Action RETURN.....	130
4.3.1.3.2.5. Action LABEL.....	131
4.3.1.3.2.6. Action TAG.....	133
4.3.1.3.2.7. Action DATETIME.....	135
4.3.1.3.2.8. Action DNS.....	136
4.3.1.3.2.9. Action DNSBL-LH, DNSBL-RH.....	137
4.3.1.3.2.10. Action SAVE RAW DATA.....	140
4.3.1.3.2.11. Action TRANSPORT.....	141
4.3.1.3.2.12. Action HEADER.....	142
4.3.1.3.2.13. Action HEADER_EX.....	143
4.3.1.3.2.14. Action LOG.....	144
4.3.1.4. Brief rules for writing filters	148
4.3.1.5. Tips.....	149
4.3.2. Pre-filtering HTTP requests	150
4.3.2.1. Conditions.....	151
4.3.2.1.1. Condition ALL, *.....	151
4.3.2.1.2. Condition METHOD.....	152
4.3.2.1.3. Condition IP.....	153
4.3.2.1.4. Condition REQ-SIZE, RESP-SIZE, SIZE.....	155
4.3.2.1.5. Condition REQ-HEADER, RESP-HEADER.....	157
4.3.2.1.6. Condition URL.....	160
4.3.2.1.7. Condition TAG.....	161
4.3.2.2. Actions	163
4.3.2.2.1. Action ACCEPT.....	163
4.3.2.2.2. Action DROP.....	164
4.3.2.2.3. Action JUMP.....	165
4.3.2.2.4. Action RETURN.....	166
4.3.2.2.5. Action COPY.....	167
4.3.2.2.6. Action ACCESS-LOG.....	168
4.3.2.2.7. Action TAG.....	169
4.3.2.2.8. Action LABEL.....	172
4.3.3. Examples of applying filters	174
4.3.3.1. Adding hostname.....	174
4.3.3.2. Filtering by hosts.....	177
4.3.3.3. Filtering by URL.....	179
4.3.3.4. Filtering by HTTP + DNSBL.....	181
4.3.3.5. Filtering large HTTP objects.....	183
5. Delivery of results to systems-consumers	184
5.1. ARCHIVING profiles.....	188
5.1.1. FILEDROP profiles	188
5.1.2. FTP profiles	191

5.1.3. IMAP profiles	193
5.1.4. SFTP profiles	196
5.1.5. SMB profiles	198
5.1.6. SMTP profiles	201
5.2. DLP profiles.....	203
5.2.1. DEVICELOCK profiles	204
5.2.2. FALCONGAZE profiles	206
5.2.3. INFOWATCH profiles	208
5.3. SIEM profiles.....	211
5.3.1. SYSLOG profiles	211
5.3.1.1. Lua scripts.....	213
5.4. SANDBOX profiles.....	213
5.4.1. VIRUSTOTAL profiles	213
5.4.2. ATHENA profiles	215
5.5. STATS profiles.....	217
5.5.1. NETFLOW profiles	218
5.6. GROUP profiles.....	220
5.7. General delivery settings.....	222
6. EtherSensor Watcher logging.....	223
6.1. Configure EtherSensor Watcher logging.....	223
6.2. EtherSensor Watcher stats settings.....	226
7. Remote management and monitoring of EtherSensor.....	230
7.1. EtherSensor RAPI profiles settings.....	231
8. Microolap EtherSensor update service.....	234
8.1. Microolap EtherSensor update service settings.....	236
9. Sensor routine maintenance.....	241
9.1. Sensor maintenance issues.....	242
10. Emergency response.....	243
11. What's new.....	244
12. GUI localization.....	245
13. Licensing Microolap EtherSensor.....	249
13.1. License file.....	250
13.2. Uhid (HardwareID) of the runtime environment.....	253
13.3. Working with the licensing system.....	254
13.4. After purchasing the license.....	255
13.5. License agreement.....	256

1. Microolap EtherSensor

Annotation

Microolap EtherSensor, v. 6.1.

Software Microolap EtherSensor is registered in the "Unified register of Russian programs for electronic computers and databases" of the Ministry of Communications of the Russian Federation under No. 5034.

Administrator's Guide.

2001 - 2020, Microolap Technologies

This document describes administrative tasks required to manage the operation of Microolap EtherSensor on your system. This document is intended for a technical audience, specifically the system administrators and information security officers in charge of Microolap EtherSensor.

Reproduction, adaptation or translation without written permission of Microolap Technologies is prohibited. The information contained in this document is subject to change without notice.

About Microolap EtherSensor

EtherSensor is an infrastructure platform for automating network traffic analysis (Network Traffic Analysis, NTA). EtherSensor extracts from traffic and analyzes objects from the data link layer to the application level layer inclusive: packets, sessions, files, messages, events and their metadata. Upon completion of the analysis, EtherSensor delivers the results to one or more systems-consumers.

EtherSensor is used as a provider of data and metadata extracted from traffic objects for NDR, DLP, eDiscovery, Enterprise Archiving systems, as well as for various SOC subsystems (SIEM systems, U(E) BA, high-load DLP systems, systems Network Detection and Response, Threat Intelligence/Management, Asset Management, Application Management, etc.).

Distinctive features of Microolap EtherSensor are:

- High performance of processing network traffic for this class of solutions (20Gbps "out of the box" on commodity hardware, scalable to 50Gbps)
- No restrictions on the number of supported Internet/Intranet services due to the openness of the system for detecting and capturing network traffic objects.

Microolap EtherSensor has been supplied since 2008. It is mainly used in information security systems.

How EtherSensor works:

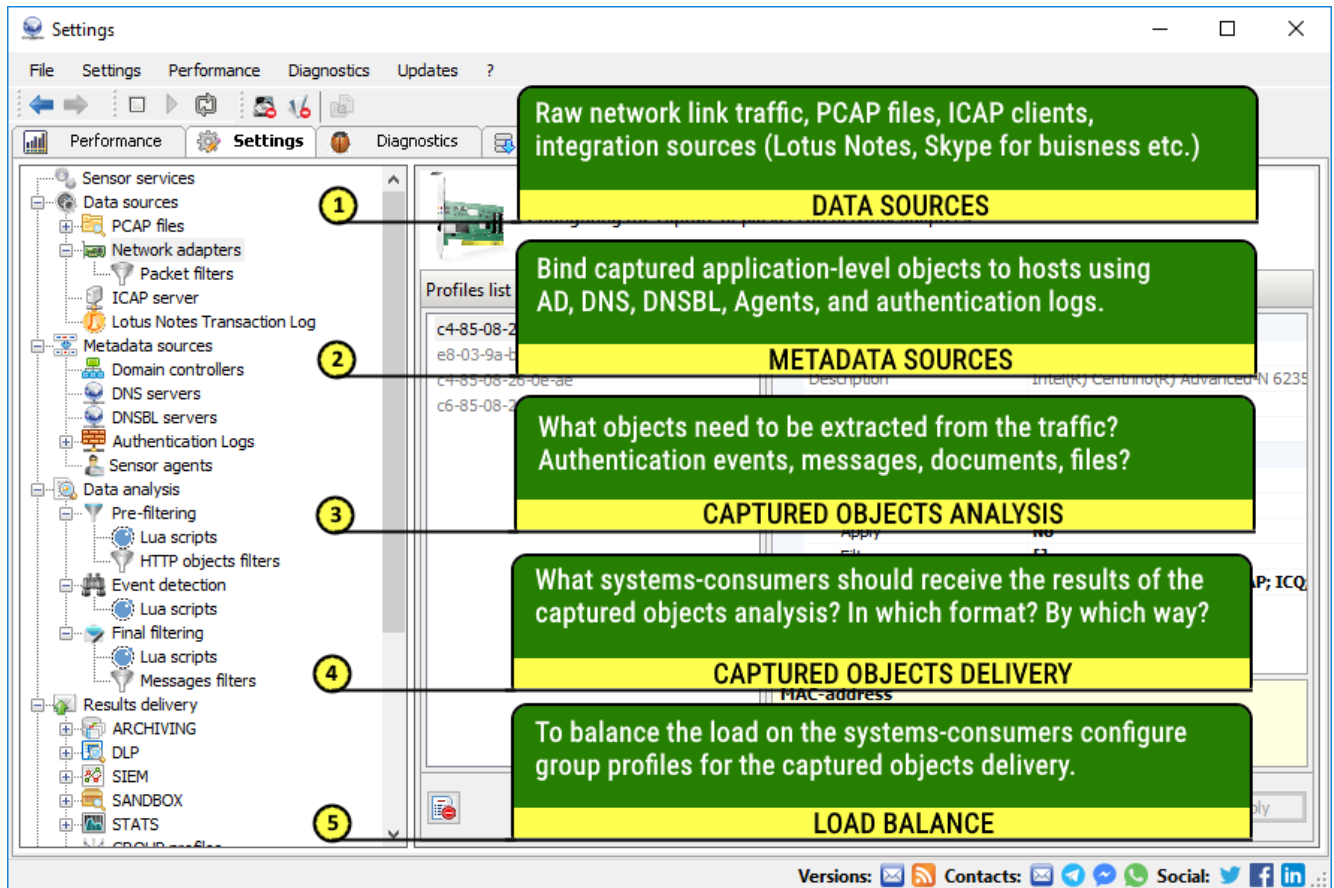


Fig.1. Scheme of EtherSensor work.

Maintenance of the working instance of EtherSensor

To maintain and support your working instance of EtherSensor use EtherSensor PCAP Edition⁽¹¹⁾.

EtherSensor PCAP Edition does not require time-consuming preparatory measures for deployment in the network infrastructure of the organization and has fully identical functionality with the full version of EtherSensor in terms of network traffic analysis and delivery of results to systems-consumers.

With this edition you can debug and test filters, detectors and delivery profiles on pre-prepared PCAP files. After debugging and testing, you can put them on your EtherSensor production instance.

If you can't find the PCAP file for the case you need (Google: "pcap files collection"), use TCPDUMP or Wireshark to record your own traffic. EtherSensor PCAP Edition supports pcap and pcapng formats.

Tip: Use TCPDUMP for Windows to write PCAP files on Windows.

1.1. Brief description of features

Microolap EtherSensor is a high-performance platform for extracting events and messages from network traffic in real time with the following properties:

- Extract from network traffic events and messages at levels from L2 to L7 according to the OSI model
- Significant number (several thousand) of Internet services recognized by EtherSensor software.
- High performance: 20Gbps "out of the box" on commodity hardware, scalable to 50Gbps
- Delivery of events, messages and metadata to any systems-consumers (archives, SOC, DLP, SIEM, eDiscovery subsystems, etc.).
- High level of continuous operation without maintenance
- Operation on commercially available off-the shelf (COTS) hardware with low consumption of resources.

EtherSensor consists of several Windows services that together capture and analyze application level messages and their metadata (usually network user messages). The messages, their metadata or the data extracted from the messages EtherSensor then delivers to the systems-consumers ⁽⁵⁾.

Distinctive feature and the main principle of EtherSensor is its non-participation in the process of transferring network traffic of the controlled network and, as a consequence, the independence of network reliability from its work. At the same time, EtherSensor is guaranteed to provide full traffic control in networks with a load of 20Gbps and above, detecting messages from several thousand Internet services.

Methods of filtering network traffic at all levels - both at the level of IP packets and at the level of already reconstructed objects of the application level - allow you to achieve a minimum consumption of resources for controlling network communications.

The extensibility of EtherSensor allows both receiving data from external sources (SPAN/TAP traffic, ICAP clients, Lotus Notes transaction log, PCAP files) and delivering reconstructed messages to any number of external systems-consumers.

Below is a description of functions EtherSensor grouped by functional modules.

Web Mail (WM)

Extracting outgoing messages from the traffic of web mail services: Mail.RU, Yandex.RU, Pochta.RU, GMail, etc. (more than 40 domains), as well as EtherSensors all services based on the "SquirrelMail" engine. To receive messages sent over an encrypted channel, use SSLSplitter, or third-party SSL decryption solutions, or EtherSensor ICAP.

Social networks (SN)

Extracting various types of user data from traffic (authentication data, text messages, comments, etc.) of the following web services:

- Social networks Vk.com, Facebook, LinkedIn, Mamba.ru and others.
- Forums powered by the phpbb, ipb, vbulletin, mybb engines
- SMS/MMS messaging services.

To retrieve messages sent over an encrypted channel, use SSLSplitter, either a third-party SSL decryption solution or EtherSensor ICAP.

Email (EM)

Extract e-mail messages from traffic via SMTP, POP3, IMAP and Lotus Notes.

To retrieve messages sent over an encrypted channel, use SSLSplitter, or third-party SSL decryption solutions, or EtherSensor ICAP.

ICAP Server (IS)

Allows you to use as a source of traffic for retrieving messages HTTP and FTP traffic transmitted to EtherSensor via ICAP protocol by external systems such as: SQUID, Blue Coat Proxy SG, Cisco WSA, Webwasher, Websense, McAfee Web Gateway, FortiGate, Entensys UserGate, etc.

Instant messages (IM)

Retrieval of instant message traffic sent and received via Skype, XMPP/Jabber, IRC, MSN, YAHOO and OSCAR instant messaging services.

To retrieve messages sent over an encrypted channel, use SSLSplitter, or third-party SSL decryption solutions, or EtherSensor ICAP.

Lotus Notes (LN)

Extract Lotus Notes events such as messages, calendar events, and others from the traffic. If Lotus Notes traffic is encrypted, EtherSensor extracts messages from Lotus Notes Transaction Log. Both methods do not affect Lotus Notes in any way.

File Transfer (FT)

Extraction of files transferred via SMB, HTTP, FTP and WebDAV protocols from traffic.

To retrieve messages sent over an encrypted channel, use SSLSplitter, or third-party SSL decryption solutions, or EtherSensor ICAP.

Job Search (CV)

Extract from the traffic events (registration, authentication, response to vacancies, CV update) posted on job and job search services, such as HH.ru, SuperJob.ru, Job.ru, etc. (over 150 domains).

To retrieve messages sent over an encrypted channel, use SSLSplitter, or third-party SSL decryption solutions, or EtherSensor ICAP.

EtherSensor Agent (AG)

EtherSensor Agent is installed on workstations in case the installation of a complete end-point DLP solution is not possible or desirable for some reason.

EtherSensor Agent is designed to bind network connections created by local processes on workstations to user and workstation names when using NAT, terminal servers, etc. in the organization's network.

In addition, EtherSensor Agent solves the tasks of tracking changes (hardware, processes, etc.) on the workstation and transmitting data about such events to the servers EtherSensor and EtherStat.

1.2. EtherSensor scope of application

EtherSensor is used to analyze traffic from L2 to L7 according to the OSI model. The result of the analysis is the content and metadata of the messages and their corresponding events.

During the development of EtherSensor requirements were set:

Adaptability to the system-consumer:

EtherSensor must be able to transmit to any system-consumer both data and metadata of the captured object in any required format and any transport protocol. The type and purpose of the consumer can be any: SIEM, DLP, eDiscovery, UEBA, Enterprise Search, file system, cloud, DBMS, etc.

Simultaneous work with many sources and consumers:

The number of systems-consumers with which EtherSensor works simultaneously should not be limited.

The number of data sources with which EtherSensor works simultaneously should also be unlimited.

Full control:

EtherSensor must detect any application layer object transmitted over the network regardless of its type. A specialized system-consumer must get the result of processing such an object.

Real time:

EtherSensor should run in real time on the fastest data links available to organizations. In the current version, EtherSensor confidently handles 20Gbps data flows out of the box on serial hardware, scalable to 50Gbps.

Work "out of the box":

EtherSensor must be deployed "out of the box" and must not require constant attention from either the developer or the Customer. Ideally, it should be a completely unattended element of the information infrastructure.

Microolap EtherSensor is most often used for solving the following groups of tasks:

Message archiving (eDiscovery and Compliance Archiving):

Extracting documents and other objects of e-mail, instant messengers, social networks, blogs, forums and other communication tools from traffic.

Examples of systems-consumers:

- Bloomberg Vault
- Global Relay
- Google Vault (help)
- IBM Content Collector
- Mailarchiva
- Smarsh
- Somansa Mail-i
- Veritas eDiscovery Platform (former Clearwell)
- Veritas Enterprise Vault.

SIEM systems, U(E)BA, log analyzers:

Extracting from traffic the metadata of captured application-level objects and related events (including in real time from the object's content) for registration in SIEM systems.

Examples of systems-consumers:

- AlienVault
- ArcSight Enterprise Security Manager (ESM) (former HP ArcSight)
- Elastic Stack (former ELK Stack: Elasticsearch, Logstash, Kibana)
- EventTracker
- FortiSIEM (former AccelOps)
- Graylog
- IBM QRadar
- LogRhythm
- ManageEngine EventLog Analyzer
- McAfee Enterprise Security Manager (ESM)
- Micro Focus Sentinel (former NetIQ)
- RuSIEM
- SolarWinds Log & Event Management (LEM)
- Splunk
- SQLstream
- Trustwave SIEM
- Any software that receives data via SYSLOG, NETFLOW or other TCP/UDP protocol.

Prevention of confidential data leaks (DLP systems)

Extracting message content from application layer traffic (L7 according to the OSI model) of external and internal communication services with their subsequent sending to the consumer system: DLP system, mail archiving system, document flow system, local search system and any other system that has the function of document archiving.

Examples of systems-consumers:

- DeviceLock DLP
- Forcepoint DLP
- Proofpoint Email DLP
- Symantec DLP
- McAfee DLP
- Trustwave DLP
- Falcongaze SecureTower
- InfoWatch Traffic Monitor
- Any other DLP solution.

1.3. System requirements

Minimum system requirements for the server to function Microolap EtherSensor:

Data flow	50 Mbps	150 Mbps	250 Mbps	500 Mbps	750 Mbps	1.5 Gbps	2.5 Gbps	5 Gbps	10 Gbps	15 Gbps	20 Gbps
Users	100	300	500	1000	1500	3000	5000	10000	20000	30000	40000
CPU	1CPU*2 Cores	1CPU*4 Cores	1CPU*4 Cores	1CPU*4 Cores	1CPU*4 Cores	1CPU*6 Cores	1CPU*8 Cores	2CPU*6 Cores	2CPU*10 Cores	4CPU*10 Cores	4CPU*12 Cores
RAM	1 GB	4 GB	4 GB	8 GB	8 GB	32 GB	64 GB	128 GB	128 GB	128 GB	256 GB
HDD	75 GB SATA	2x75 GB SATA RAID1	2x75 GB SATA RAID1	2x150 GB SATA RAID1	2x150 GB SATA RAID1	2x146 GB SAS RAID1	4x300 GB SSD RAID10	4x500 GB SSD RAID10	6x500 GB SSD RAID10	8x500 GB SSD RAID10	8x900 GB SSD RAID10
NIC	1 x 1Gbps	1 x 1Gbps	2 x 1Gbps	4 x 1Gbps	4 x 1Gbps	1 x 10Gbps	2 x 10Gbps	4 x 10Gbps	4 x 10Gbps	6 x 10Gbps	6 x 10Gbps

To avoid problems with traffic capture, use network adapters that do not have problems with RSS (Receive Side Scaling) support in drivers.

Network cards based on Intel X520, X710, XL710 and XXV710 series chipsets are guaranteed to work.

The disk subsystem is used only in special cases for caching captured network objects, as well as for storing analysis results in case of temporary unavailability of the system-consumer. Therefore, the disk subsystem size should be calculated by your own based on the possible downtime of the system-consumer.

EtherSensor runs on Windows Server 2012, Windows Server 2016 x64 or Windows Server 2019 x64, the file system is NTFS only. For maximum performance, we recommend using Windows Server 2016 or Windows Server 2019 on x64 platform.

You can also install EtherSensor on x64 desktop versions of Windows: Windows 8, Windows 8.1, and Windows 10. However, for EtherSensor Full Edition, you should consider possible performance degradation due to architectural limitations of these operating systems.

For EtherSensor correct work the latest Windows OS updates are required. If the update service is disabled, full operation of EtherSensor is not guaranteed.

1.4. Description of operation

Microolap EtherSensor consists of a set of services running on a separate hardware or virtual server ("sensor").

The sensor is connected to the Ethernet port of an active network device, which is configured to mirror network traffic (mirroring, RX and TX packets) from the specified ports, or to a network tap. Traffic capture technology developed in Microolap Technologies is used to capture traffic with zero packet loss under load in 50Gbps and above.

The sensor is a server with several network interfaces, one of which is administrative and the others are used to receive a copy of the network traffic. The OS network stack on the sensor is configured only on the administrative network interface, which is also used to transmit intercepted messages to external systems-consumers via the protocols set in the result delivery profiles.

The policy of the sensor is stored in its configuration files and is a set of rules defining IP packet capture, content analysis of captured application layer messages, and, depending on the result, the format, delivery method and direction of the captured result delivery.

The following services work on the sensor:

Services that work with data sources.

Service for extracting application-level messages PCAP files (EtherSensor PCAP, sensor_pcap.exe)

Designed to analyze PCAP files in tcpdump/libpcap/pcapng formats. EtherSensor EtherCAP service extracts application layer objects (L7 according to the OSI model) from processed files and passes these objects for further processing to the EtherSensor Analyser service.

Service for extracting application-level messages from Ethernet traffic (EtherSensor EtherCAP, sensor_ethercap.exe)

Designed to passively capture traffic on one or more Ethernet adapters. EtherSensor EtherCAP service extracts messages, events and other application-level objects from the processed traffic and then passes these objects for further processing to the EtherSensor Analyser message analysis service.

Service for extracting application-level messages from data provided by ICAP clients (EtherSensor ICAP, sensor_icap.exe)

Designed to receive traffic via ICAP protocol in REQMOD+RESPMOD mode from any ICAP clients (Squid, Blue Coat Proxy SG, Cisco WSA, etc.). Objects received from ICAP clients EtherSensor ICAP pass to the service EtherSensor Analyser for further analysis.

Service for extracting messages from Lotus Notes Transaction Log (EtherSensor LotusTXN, sensor_lotustxn.exe)

Designed for monitoring and reconstruction of Lotus Notes system messages by extracting them from Lotus Notes Transaction Log. EtherSensor LotusTXN extracts the messages from the Lotus Notes Transaction Log files and then sends these messages for further processing to the EtherSensor Analyser service.

Metadata retrieval service for enriching network events (EtherSensor Identity, sensor_identity.exe)

Designed for accumulation and processing of metadata used in EtherSensor to solve the problem of agentless binding of captured network traffic object to a specific host or user.

Service for analyzing messages extracted from Ethernet traffic (EtherSensor Analyser, sensor_analyser.exe)

Designed for detecting, analyzing and filtering captured messages and network events. EtherSensor Analyser analyzes application level protocol objects received from the EtherSensor PCAP, EtherSensor EtherCAP services, EtherSensor ICAP and EtherSensor LotusTXN to detect network events and messages transmitted over the network.

Then the filtering mechanism of the EtherSensor Analyser service makes a decision based on user-configurable rules:

1. About the termination of the message processing
2. About message delivery to the system-consumer (DLP, U(E)BA, archive, eDiscovery-system, Enterprise Search, etc.).
3. About forming a string with a predefined structure based on data extracted from the message and its metadata. For SIEM systems it is usually a syslog string in CEF format.

Also EtherSensor Analyser service is responsible for interaction with instances of EtherSensor Agent that mark sessions for binding to a specific workstation in case of using NAT, terminal services, etc.

Traffic analysis results delivery service (EtherSensor Transfer, sensor_transfer.exe)

EtherSensor Transfer is a EtherSensor subsystem designed to deliver the results of captured object analysis (the message itself, or a pre-configured syslog string) to various systems-consumers via various protocols according to predefined results delivery profiles.

EtherSensor Logging Service (EtherSensor Watcher, sensor_watcher.exe)

EtherSensor Watcher is a subsystem of EtherSensor intended for logging the current state and events of EtherSensor itself.

EtherSensor update service (EtherSensor Updater, sensor_updater.exe)

Designed to download and install Microolap EtherSensor update and license files when new versions and/or patches are released.

To view sensor operation statistics, use EtherSensor Management Console (sensor_console.exe) from the installation directory of EtherSensor.

To manage the sensor policy, also use EtherSensor Management Console, or edit the configuration files of sensor services in the directory [INSTALLDIR]\config.

1.5. Administrator skills

The system administrator of Microolap EtherSensor must:

- Have basic knowledge about the TCP/IP protocol stack, in particular about SMTP and HTTP application protocols.
- Have a basic knowledge of network interfaces administration and system services in the Windows Server family of operating systems.
- Have basic knowledge of using tcpdump and WireShark
- Have the knowledge and skills to administer Microolap EtherSensor services and be able to use them to implement corporate security policy as it relates to the use of external communication services.

1.6. List of operational documents

For operation and administration of Microolap EtherSensor, as well as for understanding the basic operations for putting the EtherSensor services into operation, the administrator should read the following documents:

- This Manual
- Documentation for external systems-consumers of messages and/or message-related events (depending on the purpose of Microolap EtherSensor use).

1.7. About EtherSensor PCAP Edition

EtherSensor PCAP Edition is a product-companion of Microolap EtherSensor designed for:

- Maintenance of the full working version of EtherSensor: development and testing of filters, rules and detectors, as well as analysis of PCAP and HAR-files without risky experiments on its production instance copy.
- Learning the functionality of EtherSensor without laborious creation of special conditions for this.

EtherSensor PCAP Edition does not contain any code to handle real-time network traffic from network adapters, and the performance of this edition is limited to 30 Mbps OR approximately 75 users.

EtherSensor PCAP Edition is licensed on the principle of "one specialist - one license", the number of installations is not limited.

1.8. Delivery of traffic analysis results

Immediately after installation, the default method for delivering traffic analysis results is the FILEDROP¹⁸⁸ profile. It saves the results to the local file system. The directory where the results are saved is set in the profile properties (**Path** field).

You can also immediately configure other transport profiles to deliver events to systems-consumers as you wish in the format they require.

There are no restrictions on the operation of delivery profiles in either the full version of EtherSensor or its PCAP Edition. But we recommend you to start with the easiest way to work with FILEDROP¹⁸⁸ profile when you first get acquainted with EtherSensor.

For more information on delivery profile settings for traffic analysis results, see Delivery of Results¹⁸⁴.

2. Microolap EtherSensor installation and configuring

To install Microolap EtherSensor, run the ethersensor_pcap_setup_x64_v6.1_en-us.exe installer included in the distribution. During the work of the installer, the software necessary for its correct operation will be installed, namely:

- Microsoft .Net Framework 4.0
- Microsoft Visual C++ 2019 redistributable runtime.

For correct installation of EtherSensor, you need administrator rights in the form in which they are set by default when installing Windows.

Additional restrictions on administrator rights may lead to incorrect operation.

During the installation of Microolap EtherSensor, the following services are installed:

EtherSensor PCAP (process sensor_pcap.exe).

Designed to analyze PCAP files in tcpdump/libpcap/pcapng formats.

EtherSensor EtherCAP (process sensor_ethercap.exe):

Designed to passively capture traffic on network adapters

EtherSensor ICAP (process sensor_icap.exe):

Designed to receive and process network traffic via ICAP from ICAP clients and extract messages.

EtherSensor LotusTXN (process sensor_lotustxn.exe):

Designed to receive Lotus Notes messages from the transaction log (Lotus Notes Transaction Log)

EtherSensor Analyser (process sensor_analyser.exe):

Designed to detect, analyze and filter captured messages

EtherSensor Transfer (process sensor_transfer.exe):

Designed to deliver the analysis results of objects extracted from traffic to systems-consumers

EtherSensor Watcher (process sensor_watcher.exe):

Designed for logging EtherSensor. Starts first, the rest of the EtherSensor services depend on it.

EtherSensor Updater Service (process sensor_updater.exe):

Designed to automatically update EtherSensor.

In addition to services, applications are installed:

EtherSensor Management Console (process sensor_console.exe):

EtherSensor management console: configuring, performance monitoring, and collecting information about the work EtherSensor to generate diagnostic reports.

After completing the EtherSensor configuration, you should configure the installed services.

2.1. Content of the software distribution kit

To prepare Microolap EtherSensor for work, you must first have the following distributions:

- Microolap EtherSensor distribution package.
- Windows Server 2012, Windows Server 2012 x64, Windows Server 2016 or Windows Server 2019. Recommended operating system is Windows Server 2019.

You can also install EtherSensor on x64 desktop versions of Windows: Windows 8, Windows 8.1, and Windows 10. However, for EtherSensor Full Edition, you should consider possible performance degradation due to architectural limitations of these operating systems.

EtherSensor distribution package contains:

- EtherSensor installer for Windows x64 platform
- Installers EtherSensor Updater (EtherSensor Updater) for Windows x64 platform
- Documentation.

EtherSensor installer contains all dependencies and installers of the platforms required for its operation. It should be used to initially install the product on a new server. This ensures that there are no problems with other software dependencies or prerequisites.

The update service does not update to the current version if the original baseline is not installed. Its installation is required in any case, only after that the update service will install the current version.

2.2. Connection Microolap EtherSensor to Ethernet

To connect the sensor to the network of the organization it is necessary to perform the following actions:

1. Connect the network management interface ⁽¹³⁾
2. Connect the network interfaces for traffic listening ⁽¹⁴⁾
3. Configure the switches. ⁽¹⁴⁾
4. Ensure compatibility with third-party information security means. ⁽¹⁵⁾

2.2.1. Management interface

EtherSensor requires a network connection of one management network interface and at least one network interface to which the copy of network traffic is directed.

The management interface is a standard network interface, configured with a TCP/IP address, subnet mask, and optionally a default router address.

For EtherSensor to work, it is desirable that when it is configured, the DNS servers on which reverse address resolution (from IP to hostname) of local network hosts is working: this will allow you to resolve the host names that participated in the connection, especially in the case of using DHCP to configure TCP/IP addresses on hosts on the local network.

Data received from the DNS server is cached, which decreases any added DNS server workload. Caching parameters are configured by the Management Console (sensor_console.exe) utility.

The management interface connection speed should be enough to pass all intercepted messages without accumulating them on the sensor. You can use 100/1000/10000 Mbit/s network interfaces.

When configuring the network management interface, take into account the special EtherSensor settings, which will filter the recognized data sent through the network management interface. For details on these settings, refer to "EtherSensor EtherCAP service" section.

2.2.2. Listening Interface

The network traffic listening interface does not require configuration of the TCP/IP protocol stack, as well as other protocols. Packets are captured from this network interface for further analysis and reconstruction of captured objects. Traffic to this interface can be directed from the Mirror port of the switch, from a network tap, or from a hub.

Normal operation requires that traffic on the listening interface contain all network packets, both from the server (RX) and the client (TX).

Please note:

The presence or absence of VLAN tags in the traffic does not matter for EtherSensor operation, since they are removed by the network adapter driver.

The bandwidth of the listening interface should be 20% more than the incoming data stream. Only then can you be sure that the packets will not be lost during capture on the network equipment and the quality of traffic capture will be high.

The sensor can work with several listening interfaces simultaneously. The maximum number of them on the server EtherSensor is not limited in any way, but depends on the performance of network interfaces, as well as on the performance of the server EtherSensor in general (amount of RAM, processor performance, etc.).

2.2.3. Switch configuration

Below is an example of Mirror port (SPAN) settings for Cisco Catalyst switches (such switches allow you to configure two active SPAN sessions):

```
monitor session 1 source interface gigabitEthernet 1/0/24 both
monitor session 1 destination interface gigabitEthernet 1/0/23
```

Here "1/0/24" is the switch port, a copy of traffic from which should be sent for analysis to EtherSensor, and "1/0/23" is the switch port to which the sensor is connected.

Below is a typical diagram of how the sensor is deployed in a local network:

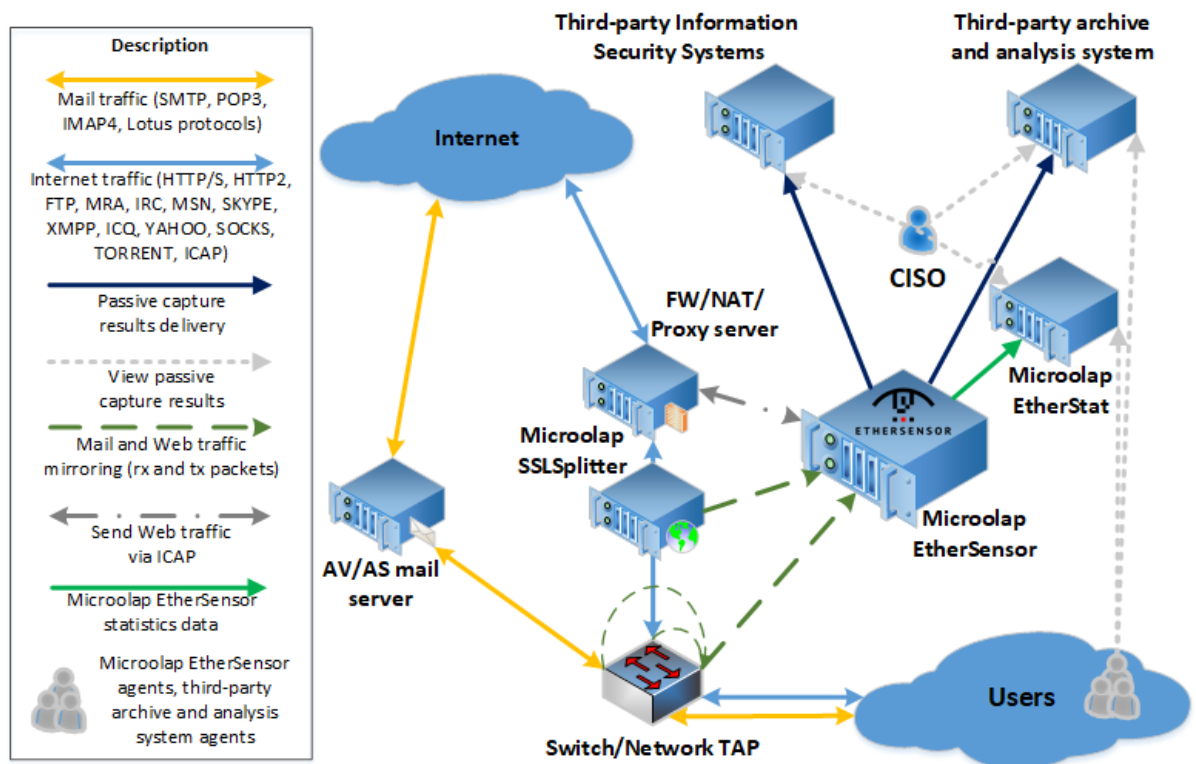


Fig.2. Typical scheme of Microolap EtherSensor deployment in the local network.

2.3. Using Third-Party Information Security Tools

To ensure the stable operation of Microolap EtherSensor, it is necessary to take into account compatibility with third-party information security systems/tools and other components of the organization's information infrastructure:

- The directory where EtherSensor is installed contains working directories. This path and all its subdirectories should be excluded from the control of tools similar to antiviruses, search indexers, as well as integrity control tools. No software should block files in these directories from being created, deleted, moved or modified.
- EtherSensor includes the following Windows services: EtherSensor EtherCAP, EtherSensor ICAP, EtherSensor LotusTXN, EtherSensor Analyser, EtherSensor Transfer, EtherSensor Watcher and an update service EtherSensor Updater. These services must be run with the local system rights because they require access to kernel functions.
- EtherSensor requires for normal operation sending messages via SMTP, FTP, SMB, SYSLOG or IMAP to the remote server. Information security means must not check, modify, or restrict connections to the server where EtherSensor sends data. Similarly, information security means should not prevent EtherSensor Transfer from opening connections on the ports used to send messages.
- EtherSensor requires connection to the NDIS system module for its work. Third party means of information protection should not prevent EtherSensor EtherCAP service from having access to the functions of this module.

- During installation and operation, software processes of EtherSensor use calls that require high privileges. The OS security policy should allow driver operations, process control and access to network interfaces for EtherSensor.
- During the interception process EtherSensor is able to operate with large volumes of files in the working directories. The file system EtherSensor server must have enough free space for data recording and storage.
- When working with large network flows, it is recommended to mount the [INSTALLDIR]\data directory as a separate partition on the raid controller with optimized access and write speeds (raid10).

2.4. Sensor settings

Before using the sensor, make the following settings:

1. **Ensure that the license file is available in the Microolap EtherSensor installation directory** ⁽²⁴⁹⁾ (for PCAP Edition is not required)
2. **Configure the data sources** ⁽¹⁶⁾
3. **Configure the metadata sources** ⁽⁴⁹⁾
4. **Configure capture results delivery** ⁽¹⁸⁴⁾
5. **Configure logging service** ⁽²²³⁾
6. **Configure messages analyser service** ⁽⁸¹⁾
7. **Configure HTTP objects filter** ⁽¹⁵⁰⁾
8. **Configure updater service** ⁽²³⁴⁾.

3. Sources of data and metadata

In the current version Microolap EtherSensor (6.1), the following services function on the sensor, designed to work with data sources: EtherSensor PCAP ⁽¹⁸⁾, EtherSensor EtherCAP ⁽²⁷⁾, EtherSensor ICAP ⁽⁴³⁾ and EtherSensor LotusTXN ⁽⁴⁶⁾.

In addition, EtherSensor includes the service EtherSensor Identity ⁽⁴⁹⁾, which is responsible for obtaining metadata to bind captured objects to a user or host.

The general scheme of services work is as follows:

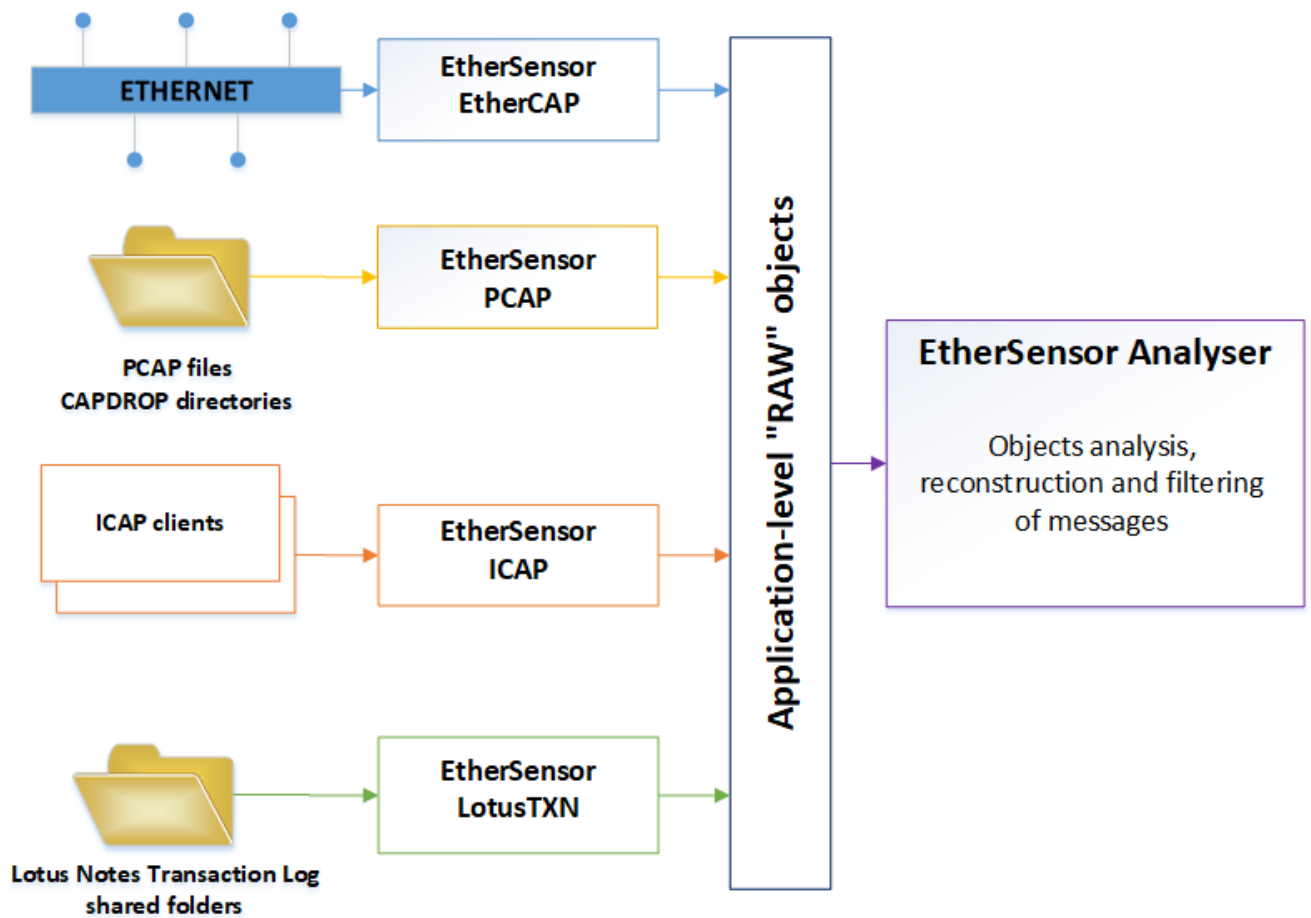


Fig.3. Scheme of work of services EtherSensor PCAP, EtherSensor EtherCAP, EtherSensor ICAP, EtherSensor LotusTXN.

The EtherSensor PCAP⁽¹⁸⁾ service is responsible for processing traffic from PCAP files in tcpdump/libpcap/pcapng formats.

The EtherSensor EtherCAP⁽²⁷⁾ service is responsible for passive traffic capturing on network adapters and reconstructing application layer protocol sessions.

The EtherSensor ICAP⁽⁴³⁾ service is responsible for receiving traffic via ICAP from any ICAP clients and further transferring the received objects to the EtherSensor Analyser⁽⁸¹⁾ service.

The EtherSensor LotusTXN⁽⁴⁶⁾ service is responsible for extracting messages from Lotus Notes Transaction Log files, it is used only when there is no access to unencrypted Lotus Notes traffic.

The result of these services work is the reconstructed objects transferred to the service of results analysis EtherSensor Analyser⁽⁸¹⁾.

To start and stop services, you can use both standard Windows Service snap-in and Management Console (utility sensor_console.exe) from the installation directory EtherSensor:

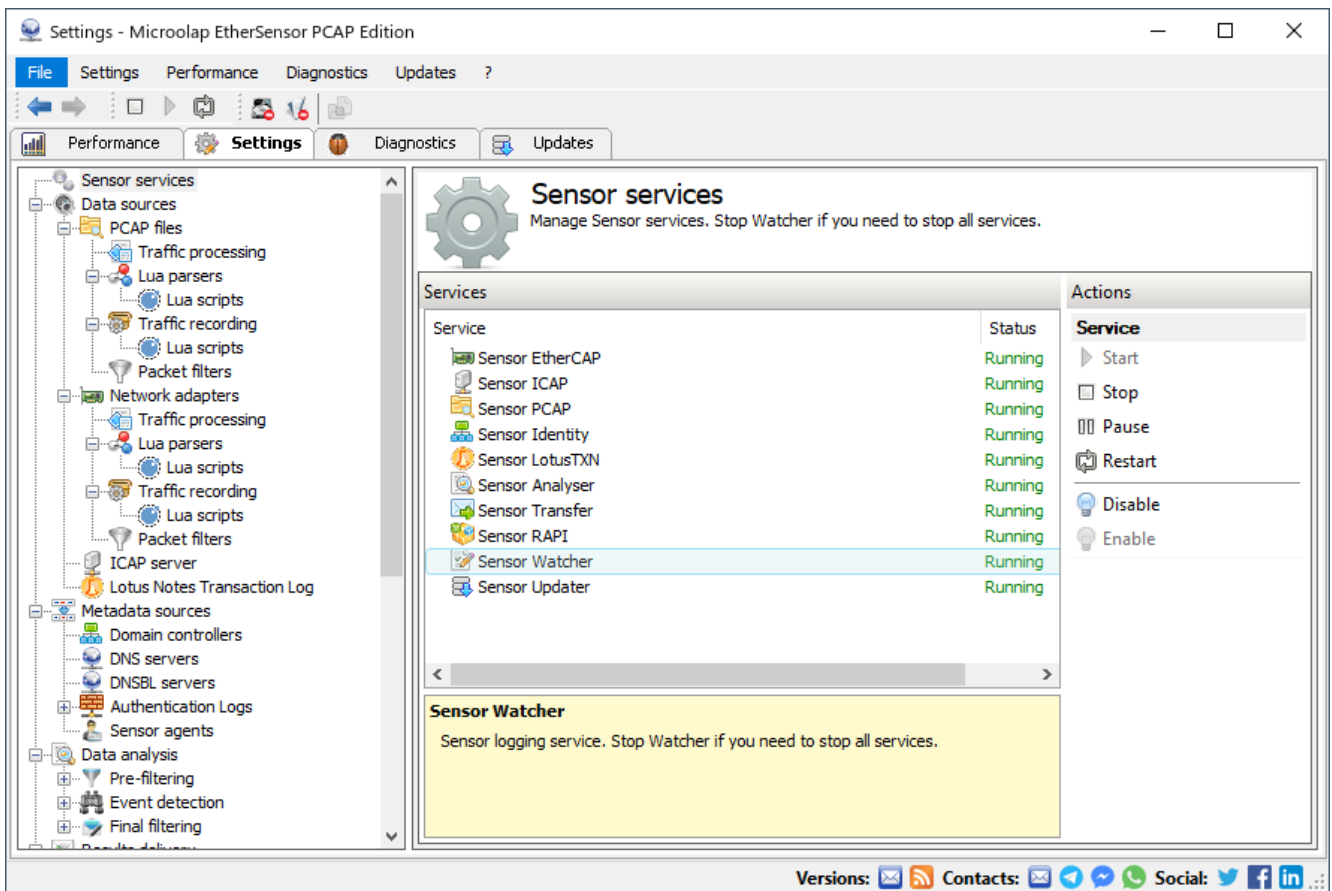


Fig.4. Start and stop the data source services.

Dependencies are set between the service of the logging subsystem EtherSensor Watcher and other EtherSensor services. Since no operation event EtherSensor should go unnoticed by the logging subsystem, no service EtherSensor can be launched before the logging service EtherSensor Watcher is launched.

And vice versa: to stop all services of EtherSensor, it is enough to stop only the logging service EtherSensor Watcher.

3.1. EtherSensor PCAP service

The EtherSensor PCAP service is responsible for processing traffic from PCAP files in tcpdump/libpcap/pcapng formats.

EtherSensor PCAP extracts application layer objects (L7 according to OSI model) from processed files and passes these objects for further processing to the service EtherSensor Analyser⁽⁸¹⁾.

EtherSensor PCAP can handle the same protocols of the third layer of OSI model as the service EtherSensor EtherCAP.

Interaction of EtherSensor PCAP with the service EtherSensor Analyser (and through it with the service EtherSensor Transfer) is the same, as in the EtherSensor EtherCAP service.

General working scheme of the EtherSensor PCAP service:

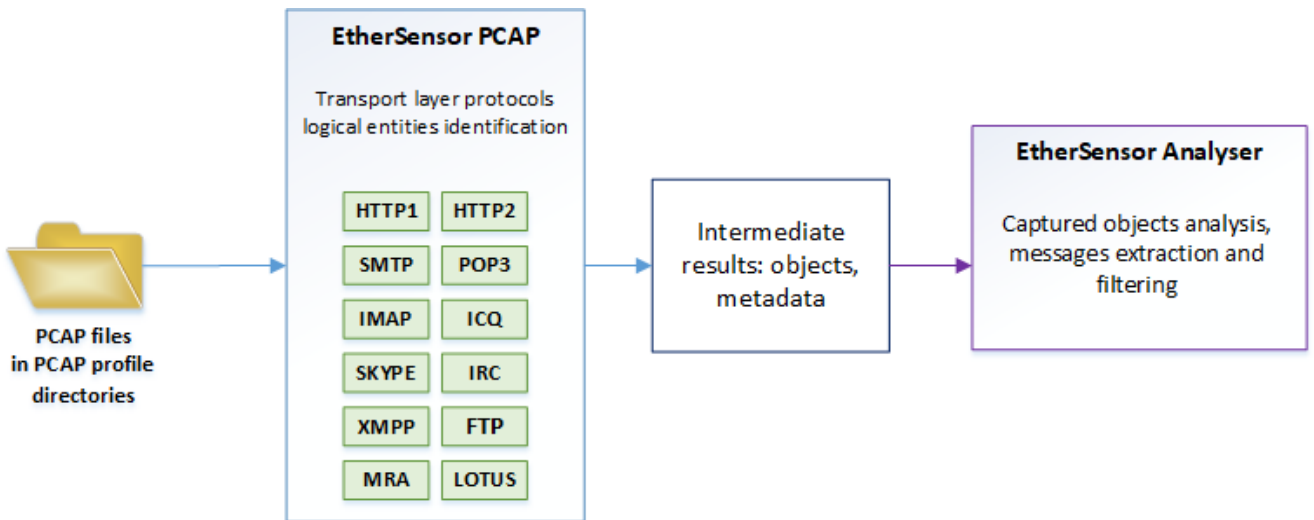


Fig.5. Scheme of the service EtherSensor PCAP work.

Options of EtherSensor PCAP work organization:

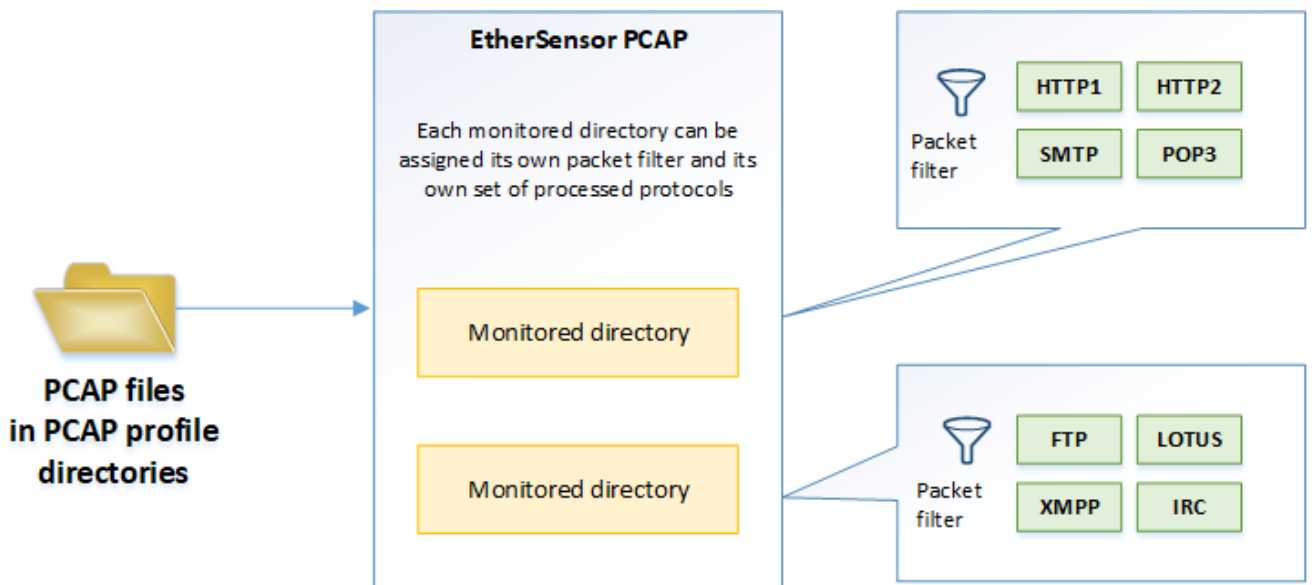


Fig.6. Variants of the EtherSensor PCAP service organization.

The EtherSensor PCAP service allows you to simultaneously monitor several local directories to process the PCAP files placed in them. It also allows you to assign a packet filter and network protocols to be analyzed for each monitored directory.

Command line parameters

The Windows service EtherSensor PCAP is installed during the installation of EtherSensor with automatic start. However, if necessary, the process `sensor_pcap.exe` can be run as a Windows application with the following command line parameters:

/process

Run the process `sensor_pcap.exe` as a normal Windows Win32 process (use for debugging is possible).

/service

Run as a Windows service.

/config

Save the default service configuration.

3.1.1. EtherSensor PCAP settings

EtherSensor PCAP service operates with a special virtual appliance designed to process traffic previously saved in PCAP files.

The settings of EtherSensor PCAP slightly differ from those of the EtherSensor EtherCAP⁽²⁷⁾ service:

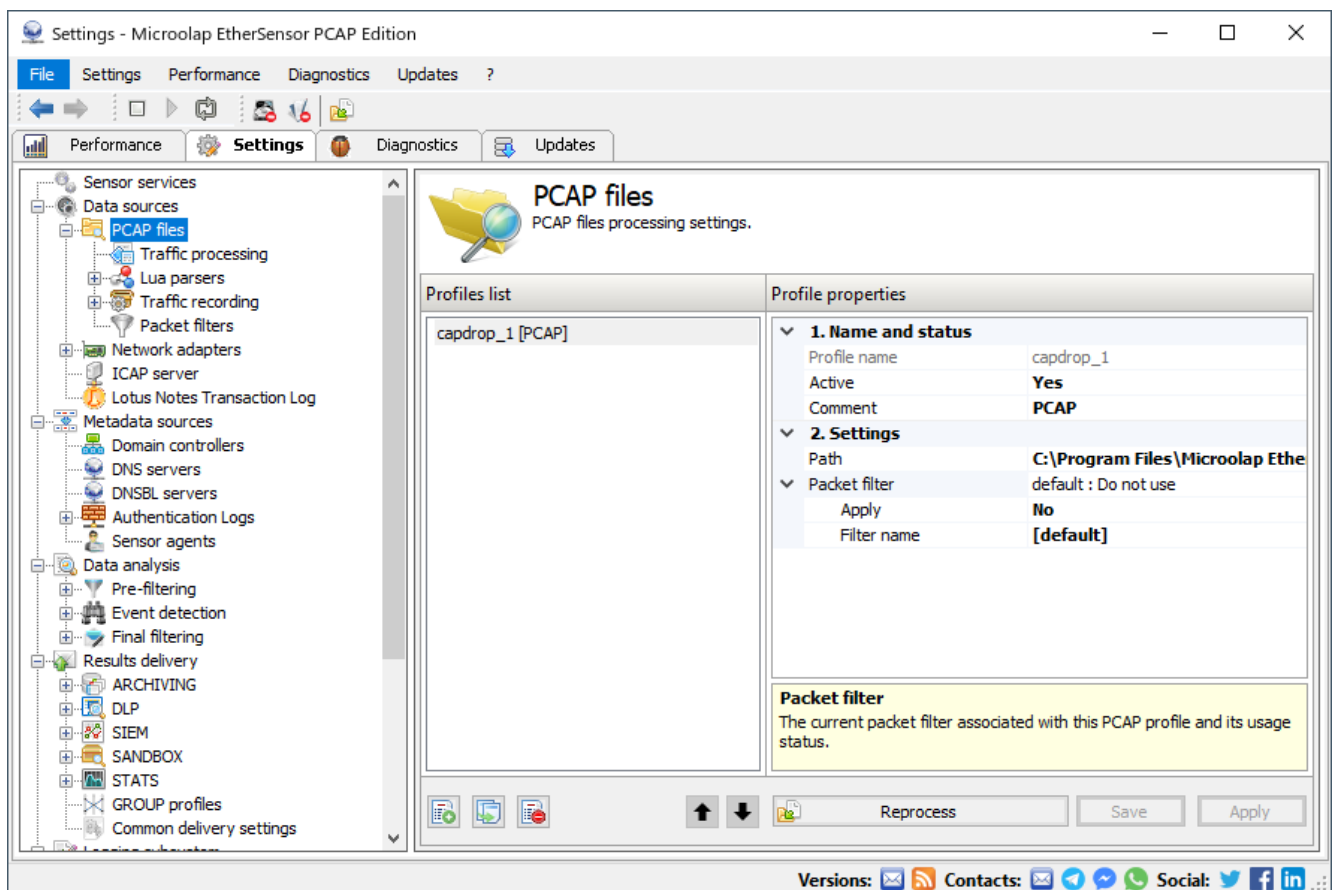


Fig.7. Settings of the EtherSensor PCAP service.

To process PCAP files, copy them to the directory of the PCAP profile and then allow the processing of PCAP files in the directory specified in **Path**. To do this, set the status of this PCAP profile **Active** to **Yes**.

PCAP files are "listened to" in FIFO order: a file placed in the directory of PCAP profile first (i.e. having an earlier modification date in terms of the file system) will be "listened to" first, placed there second will be "listened to" second, etc.

If the service profile flag **Active** was in the state **Yes**, the directory of the PCAP profile was empty and a PCAP file was placed in it, then this file will immediately start to be "listened".

If the service profile flag **Active** was in the state **No**, there were PCAP files in the directory of the PCAP profile, then "listening" to the files will start immediately after you put this service profile into active state.

All PCAP files processed in this way are placed in the subdirectory `.\processed` of the PCAP profile directory without any changes. This allows you to repeat the process of analyzing PCAP files as many times as you want using the **Reprocess** button when debugging filters and detectors.

Objects extracted by the EtherSensor PCAP service from PCAP files are then processed in the usual way: analyzed by the EtherSensor Analyser service, which passes them to the EtherSensor Transfer service to deliver the analysis results to the systems-consumers.

Tip: if you just need to check or debug the detectors or filters, the most convenient type of system-consumer is the local file system (FILEDROP delivery profile).

The EtherSensor PCAP service configuration is contained in the XML file `pcap.xml` located in the common configuration directory `Microolap EtherSensor [INSTALLDIR]\config`.

3.1.2. EtherSensor PCAP configuration file

The EtherSensor PCAP service configuration is contained in the XML file `pcap.xml` located in the common configuration directory `Microolap EtherSensor [INSTALLDIR]\config`.

Example `pcap.xml` configuration file:

```
<?xml version="1.0" encoding="utf-8"?>
<PcapConfig version="1.0">
  <PcapAdapters>
    <PcapAdapter enabled="true" id="capdrop_1" description="PCAP files processing adapter">
      <Comment>PCAP</Comment>
      <CapDrop>[[INSTALLDIR]]\data\capdrop</CapDrop>
      <Filter enabled="false" name="default" />
      <Protocol enabled="true" name="ftp" />
      <Protocol enabled="true" name="http" />
      <Protocol enabled="true" name="http2" />
      <Protocol enabled="true" name="icap" />
      <Protocol enabled="true" name="imap4" />
      <Protocol enabled="true" name="irc" />
      <Protocol enabled="true" name="lotus" />
      <Protocol enabled="true" name="pop3" />
      <Protocol enabled="true" name="skype" />
      <Protocol enabled="true" name="smb" />
      <Protocol enabled="true" name="smtp" />
      <Protocol enabled="true" name="socks" />
      <Protocol enabled="true" name="websocket" />
      <Protocol enabled="true" name="xmpp" />
    </PcapAdapter>
  </PcapAdapters>

  <Filters>
    <Filter name="default">
      <RuleGroup enabled="true" name="">
        <Rule type="accept" src="any" srcport="any" dst="any" dstport="any" proto="tcp" />
      </RuleGroup>
    </Filter>
    <Filter name="internet">
      <RuleGroup enabled="true" name="">
        <Rule
          type="reject"
          src="192.168.0.1"
          srcport="any"
          dst="any"
          dstport="any"
          proto="tcp" />
        <Rule
          type="reject"
          src="*"
          srcport="any"
          dst="192.168.0.1"
          dstport="any"
          proto="tcp" />
        <Rule
          type="accept"
          src="any"
          srcport="any"
          dst="any"
          dstport="any"
          proto="tcp" />
      </RuleGroup>
    </Filter>
  </Filters>
</PcapConfig>
```

Tag PcapConfig

EtherSensor PCAP configuration root tag. The "version" attribute contains the configuration version. In Microolap EtherSensor version 6.X it should always be equal to 1.0.

Tag PcapAdapters

Defines the settings for the monitored directories with PCAP files.

Tag PcapAdapter

The PcapAdapter tag is nested within the PcapAdapters tag and contains settings for the PCAP interface. The "enabled" attribute contains the active status of the PCAP interface: if it is set to false, the PCAP interface will not be used. The "id" attribute is used to indicate the name of the PCAP interface. This attribute should not be changed and is read-only. The "description" attribute contains your description of the PCAP interface.

Tag Comment

The Comment tag is nested within the PcapAdapter tag and contains your comment on configuring the PCAP interface profile.

CapDrop Tag

The CapDrop tag is nested within the PcapAdapter tag and contains the absolute path to the monitored PCAP directory.

Tag Filter

The Filter tag is nested within the PcapAdapter tag and contains a description of the IP filter used for this PCAP interface. The "enabled" attribute contains the status of using the IP filter: if it is set to false, then this filter will not be used for this PCAP interface.

The "name" attribute contains the name of the IP filter profile. IP filter profiles are specified in the Filters tag (see below).

Tag Protocol

The Protocol tag is nested within the PcapAdapter tag and contains the name of the Internet protocol. The "enabled" attribute contains the status of using the Internet protocol: if this attribute is "false", then this protocol will be ignored for this PCAP interface. The "name" attribute contains the name of the Internet Protocol Profile. This attribute is read-only and cannot be changed.

Example of PCAP-interface configuration for capturing HTTP, HTTP2, SMTP, WEBSOCKET client messages:

```
<PcapAdapter enabled="true" id="capdrop_1" description="PCAP files processing adapter">
<Comment>PCAP</Comment>
  <CapDrop>[[INSTALLDIR]]\data\capdrop</CapDrop>
    <Filter enabled="false" name="default" />
      <Protocol enabled="false" name="ftp" />
      <Protocol enabled="true" name="http" />
      <Protocol enabled="true" name="http2" />
      <Protocol enabled="false" name="icap" />
      <Protocol enabled="false" name="imap4" />
      <Protocol enabled="false" name="irc" />
      <Protocol enabled="false" name="lotus" />
      <Protocol enabled="false" name="pop3" />
      <Protocol enabled="false" name="skype" />
      <Protocol enabled="false" name="smb" />
      <Protocol enabled="true" name="smtp" />
      <Protocol enabled="false" name="socks" />
      <Protocol enabled="true" name="websocket" />
      <Protocol enabled="false" name="xmpp" />
    </PcapAdapter>
```

Tag Filter

The Filter tag is nested within the Filters tag and contains a description of the IP filter settings. The "name" attribute contains the name of the IP filter profile. The value of this attribute can be used as the value of the "PcapAdapter/Filter/name" attribute to indicate the IP filter to the PCAP interface.

RuleGroup tag

The RuleGroup tag is nested in the Filter tag and is used to group filtering rules related to a specific traffic filtering task.

The "name" attribute contains your description of the filter rule group. The value of this attribute can be empty.

Tag Rule

The Rule tag is nested within the RuleGroup tag and contains your description of the PCAP traffic filtering rule.

The "type" attribute contains the type of the rule: if it is equal to accept, then the network packets matching this rule will be accepted for further processing. Otherwise, if it is equal to reject, then network packets matching this rule will be rejected.

The "src" and "dst" attributes contain an IP address, a range of IP addresses, or network parameters for filtering IP addresses that match the specified value.

Example:

Reject packets that pass between computers 10.1.5.10, 10.1.5.15-10.1.5.59 and network 10.1.6.0/255.255.255.0:

```
<Rule
  type="reject"
  src="10.1.5.10, 10.1.5.15-10.1.5.59"
  dst="10.1.6.0/255.255.255.0"
  proto="tcp" />

<Rule
  type="reject"
  src="10.1.6.0/255.255.255.0"
  dst="10.1.5.10, 10.1.5.15-10.1.5.59"
  proto="tcp" />
```

In the "srcport" and "dstport" attributes specify the TCP ports or TCP port ranges required for filtering.

Example:

Reject packets passing between computers 10.1.5.10, 10.1.5.15-10.1.5.59 and network 10.1.6.0/255.255.255.0 on ports 80, 443-1024:

```
<Rule
  type="reject"
  src="10.1.5.10, 10.1.5.15-10.1.5.59"
  srcport="80, 443-1024"
  dst="10.1.6.0/255.255.255.0"
  proto="tcp" />

<Rule
  type="reject"
  src="10.1.6.0/255.255.255.0"
  dst="10.1.5.10, 10.1.5.15-10.1.5.59"
  dstport="80, 443-1024"
  proto="tcp" />
```

The rules are applied linearly from top to bottom. The top line is the first filter statement, the bottom line is the last. Each line rejects or accepts only the type of packets it describes.

Example:

Reject connection packets between two hosts or a group of hosts. In this case, packets transmitted to both sides of the connection should be rejected:

```
<Rule
  type="reject"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule
  type="reject"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />
```

It is also necessary to remember that if there are no filtering rules, then all traffic will be accepted. Conversely, if there are filtering rules, then only traffic that matches the filtering rules is processed.

Example:

Get traffic from all connections to a single host 10.31.5.57:

```
<Rule
  type="accept"
  src="10.31.5.57"
  srcport "*"
  dst "*"
  dstport "*"
  proto="tcp" />

<Rule
  type="accept"
  src "*"
  srcport "*"
  dst="10.31.5.57"
  dstport "*"
  proto="tcp" />
```

Example:

Cut off the hosts group. To do this, you must first cut this group off, and then make sure to accept all other packets, otherwise we will get nothing at all:

```
<Rule
  type="reject"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule
  type="reject"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />

<Rule
  type="accept"
  src "*"
  srcport "*"
  dst "*"
  dstport "*"
  proto="tcp" />
```

Example:

Get traffic of only two specific hosts, ignore the rest:

```
<Rule
  type="accept"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule
  type="accept"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />
```

3.1.3. EtherSensor PCAP packet filters

ATTENTION:

tcpdumpIn version 6.1 the format of packet filters has changed. If you are using packet filters, they must be converted to the new format (tcpdump/libpcap) manually. The old filters are saved in the \backup\DD.MM.YYYY\config directory.

Packet filters for the service EtherSensor PCAP are absolutely identical to packet filters for EtherSensor EtherCAP⁽³⁹⁾.

Therefore, after debugging the filters on PCAP files, they can be copied from the pcap.xml configuration file to the ethcap.xml⁽³³⁾ configuration file of the EtherSensor EtherCAP service and then used in its further work.

Use EtherSensor PCAP Edition in this process: it will save you a lot of time and effort.

3.2. EtherSensor EtherCAP service

The EtherSensor EtherCAP service is responsible for passive traffic capture on network adapters.

EtherSensor EtherCAP extracts application layer objects (L7 according to the OSI model) from the processed traffic and then transfers these objects for further processing to the EtherSensor Analyser service.

EtherSensor EtherCAP can handle the following protocols of the third layer of OSI model:

IP:

Ordinary traffic

GRE:

Tunneled traffic: e.g. unencrypted Ethernet over GRE or IP-over-IP connections

IPv6-over-IPv4:

Encapsulated IPv6. As a rule, it is found in large networks and on trunk channels.

In all cases EtherSensor handles mainly TCP and UDP streams.

In the current version EtherSensor (6.1) the EtherSensor EtherCAP service can recognize and process the following most frequently used application-level Internet protocols:

HTTPv1/HTTPv2:

All kinds of requests, message transfer, file transfer.

SMTP:

Sending outgoing mail messages.

POP3:

Sending incoming mail.

IMAP4:

Sending incoming and outgoing mail messages.

ICQ:

Sending incoming and outgoing instant messages, contact lists, files.

DC:

Sending of instant messages via NMDC and ADC (DC++) protocols.

LOTUS:

Sending mail messages, calendars, Lotus Notes system tasks.

MRA:

Sending instant messages, contact lists and files using the Mail.Ru Agent utility.

MSN:

Sending instant messages, contact lists and files using the MSN utility.

XMPP:

Sending instant messages, contact lists and files using XMPP clients (Google Talk, etc.).

IRC:

Sending instant messages and files.

SKYPE:

Detecting the fact of using skype clients, extracting messages.

SSL:

Detecting the fact of using the SSL protocol.

TORRENT:

Detecting the fact that Torrent clients are being used.

FTP:

Files transfer.

YAHOO:

Sending instant messages, files transfer.

ICAP:

Capture data transferred via ICAP protocol.

Microolap EtherSensor includes the service EtherSensor ICAP: this is an ICAP server designed to work with ICAP clients.

Passive capture of ICAP protocol is another Microolap EtherSensor function.

SOCKS:

Capture data transmitted via SOCKS protocol.

WEBSOCKET:

Capture data transmitted via WEBSOCKET protocol.

General scheme of the EtherSensor EtherCAP service work:

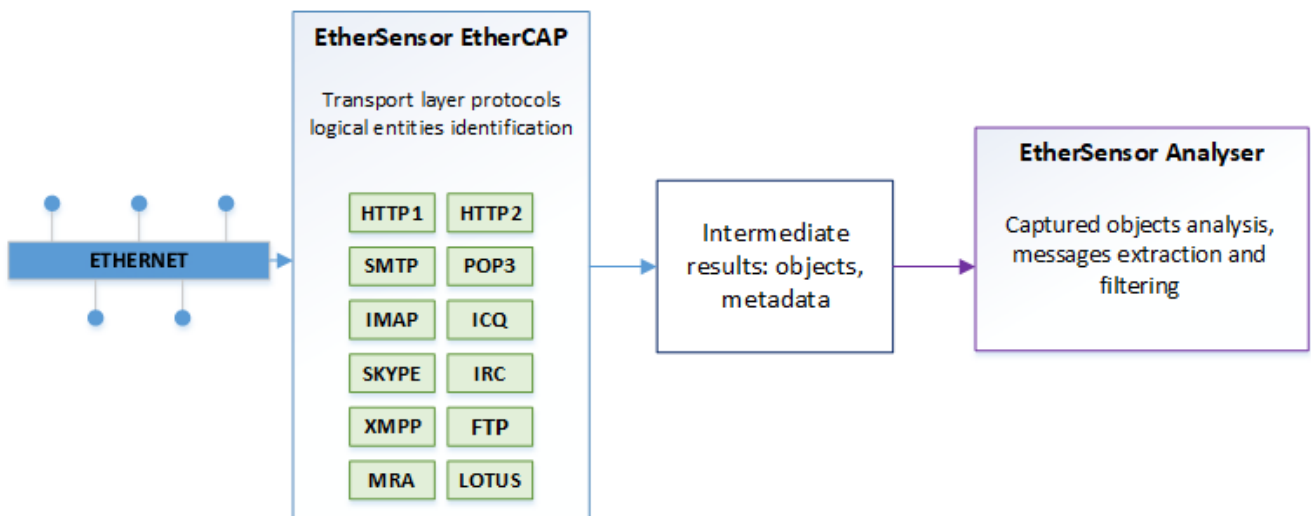


Fig.8. Scheme of the service EtherSensor EtherCAP work.

The EtherSensor EtherCAP service allows you to simultaneously capture traffic from all available Ethernet interfaces, as well as monitor local directories for processing PCAP files placed in them.

The amount of hardware resources required for each network interface to be monitored can be calculated based on the average amount of memory required to handle one TCP connection being approximately 40 Kbytes:

Network interface bandwidth	RAM consumption for processed packets cache
-----------------------------	---

10000 Mbps	2000 MB
5000 Mbps	1000 MB
1000 Mbps	200 MB
100 Mbps	50 MB
10 Mbps	10 MB

Thus, for simultaneous tracking of 10000 TCP sessions via a 1 Gbps network interface, the server EtherSensor needs the following amount of available physical memory:

200 MB + 10,000 * 40 KB - about 600 MB

The EtherSensor EtherCAP service allows you to assign a packet filter as well as network protocols to be monitored for each network interface you are listening to.

Variants of the EtherSensor EtherCAP service work organization:

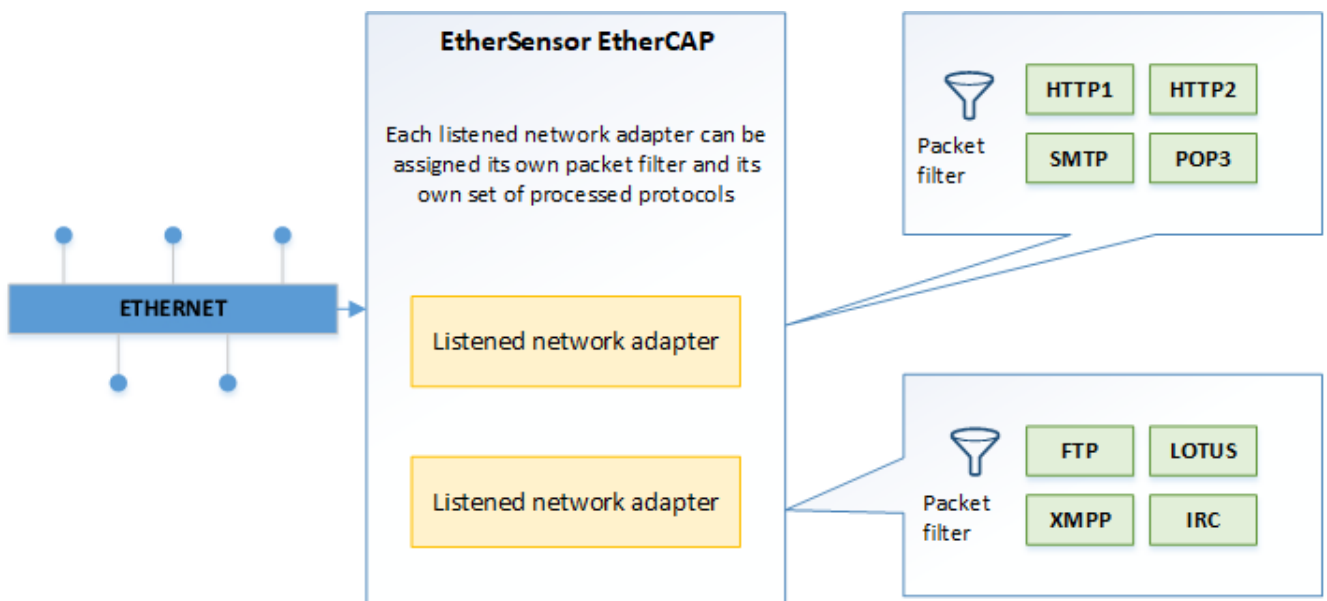


Fig.9. Variants of the EtherSensor EtherCAP service work organization:

Command line parameters

The Windows service EtherSensor EtherCAP is installed during the installation of EtherSensor with automatic start. However, if necessary, the process sensor_ethercap.exe can be run as a Windows application with the following command line parameters:

/process

Run the process sensor_ethercap.exe as a normal Windows Win32 process (use for debugging is possible).

/service

Run as a Windows service.

/config

Save the default service configuration.

3.2.1. EtherSensor EtherCAP settings

The service EtherSensor EtherCAP allows you to listen to hardware network interfaces installed on the EtherSensor server.

Hardware network interface settings

Information on network interface and packet filter settings can be found in ethcap.xml file located in [INSTALLDIR]\config directory, which can be edited either in the management console or with any text editor.

In the management console (file sensor_console.exe), setting up the service EtherSensor EtherCAP is done as follows:

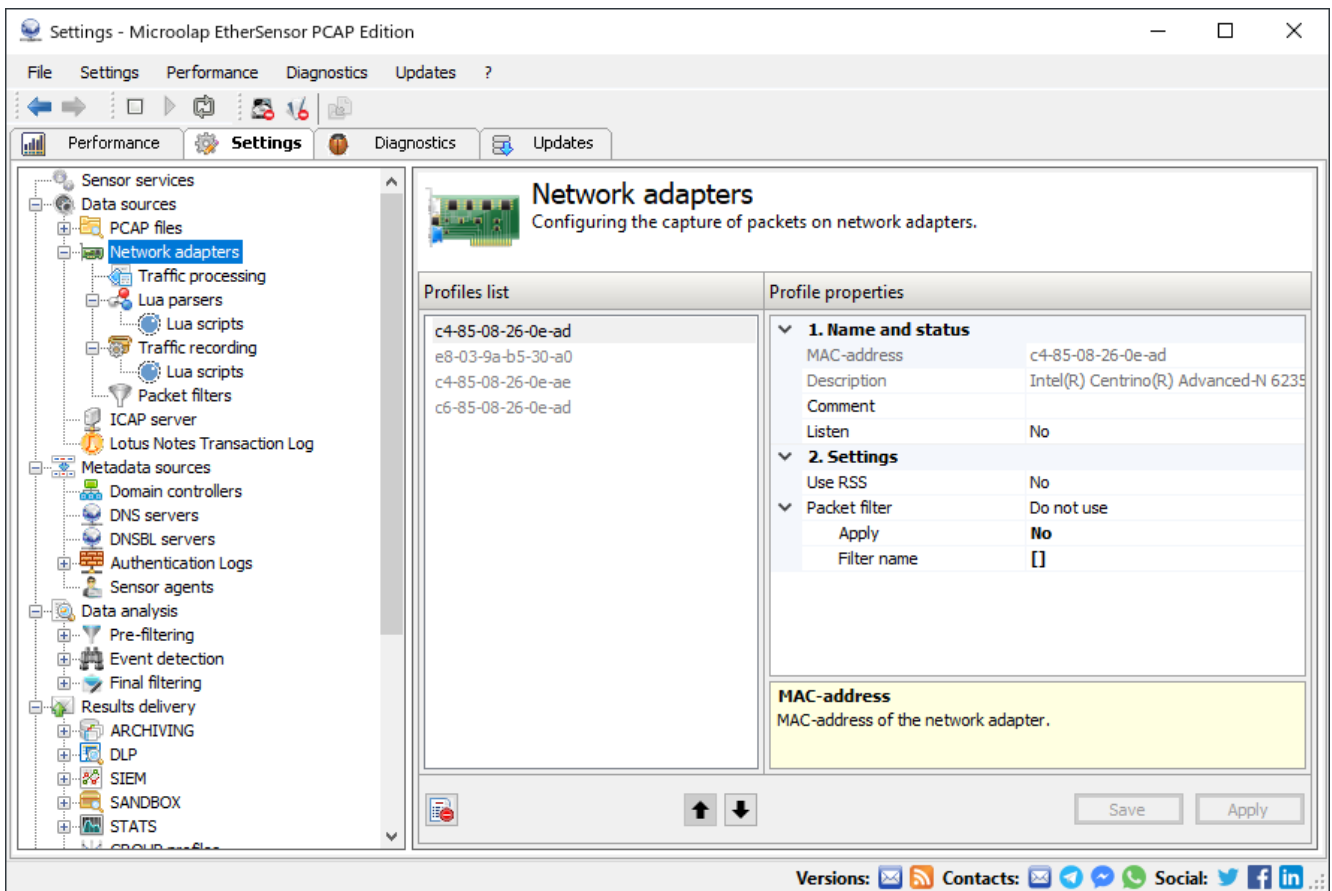


Fig.10. EtherSensor EtherCAP service settings.

MAC-address:

The MAC address of the network adapter that you are setting up. The value of the MAC address will be present in messages sent by EtherSensor to the system-consumer.

Listen:

Enable/disable listening to this interface by the EtherSensor EtherCAP service. It is possible to listen to several interfaces simultaneously. By default the EtherSensor EtherCAP service listens to all interfaces for which the OS network stack is not configured.

To listen to the interface for which the OS network stack is configured, you should set the **Listen** flag to **Yes**. In order not to listen to those interfaces, for which the OS network stack is not configured, you should set their flags **Listen** in **No**.

Use RSS:

Switch RSS technology on/off . Receive Side Scaling (RSS) is a technology that evenly distributes the network packet processing load among the processor cores to optimize performance.

Packet filter:

The name of the filter associated with this interface and the status of its use are displayed **Use/Do not use**). You can define in advance as many filters as you like, but you can apply only one of them at the same time on the same interface.

Filter name:

The field for selecting a packet filter from among those already available. The filters are stored in `[INSTALLDIR]\config\ethcap.xml`⁽³³⁾ file.

Apply:

Enable/disable the use of the specified filter on this network interface.

Protocols:

Allows you to disable unused protocol filters in order to save resources. For example, if you do not need to work with instant messages, you should disable the filters for the DC, ICQ, IRC, MRA, MSN, SKYPE and YAHOO protocols.

EtherSensor EtherCAP configuration file

The configuration for the service EtherSensor EtherCAP is contained in the ethcap.xml XML file located in the Microolap EtherSensor `[INSTALLDIR]\config` shared configuration directory.

3.2.2. EtherSensor EtherCAP configuration file

The EtherSensor EtherCAP service configuration is contained in the XML file ethcap.xml, located in the common configuration directory Microolap EtherSensor `[INSTALLDIR]\config`.

```
<?xml version="1.0" encoding="utf-8"?>
<EtherCapConfig version="4.2"
  flow_count="16"
  flow_buff_count="512"
  flow_buff_size="524288">
  <NetworkAdapters>
    <NetworkAdapter enabled="true" rss="true" mac="00-1F-C6-2D-EA-40"
      description="Marvell Yukon 88E8056 PCI-E Gigabit Ethernet Controller">
      <Filter enabled="true" name="internet" />
      <Protocol enabled="true" name="dc" />
      <Protocol enabled="true" name="ftp" />
      <Protocol enabled="true" name="http" />
      <Protocol enabled="true" name="icq" />
      <Protocol enabled="true" name="imap4" />
      <Protocol enabled="true" name="irc" />
      <Protocol enabled="true" name="lotus" />
      <Protocol enabled="true" name="mra" />
      <Protocol enabled="true" name="msn" />
      <Protocol enabled="true" name="pop3" />
      <Protocol enabled="true" name="skype" />
      <Protocol enabled="true" name="smtp" />
      <Protocol enabled="true" name="ssl" />
      <Protocol enabled="true" name="xmpp" />
      <Protocol enabled="true" name="yahoo" />
    </NetworkAdapter>
  </NetworkAdapters>

  <Filters>
    <Filter name="default">
      <RuleGroup enabled="true" name="">
        <Rule type="accept" src="any" srcport="any" dst="any" dstport="any" proto="tcp"
          comment="Comment to the rule" />
      </RuleGroup>
    </Filter>

    <Filter name="internet">
      <RuleGroup enabled="true" name="">
        <Rule type="reject" src="192.168.0.1" srcport="any" dst="any" dstport="any" proto="tcp"
          comment="Comment to the rule" />
        <Rule type="reject" src="*" srcport="any" dst="192.168.0.1" dstport="any" proto="tcp" />
        <Rule type="accept" src="any" srcport="any" dst="any" dstport="any" proto="tcp" />
      </RuleGroup>
    </Filter>
  </Filters>
</EtherCapConfig>
```

Tag EtherCapConfig

EtherSensor EtherCAP configuration root tag. Attribute "version" contains the version of configuration files.

The "flow_count" attribute contains the number of flows simultaneously processing the traffic. All captured traffic is evenly distributed among the processing flows. Traffic is distributed at the operating system kernel level, thus ensuring parallelism of data processing.

The "flow_buff_count" attribute contains the number of data buffers in the traffic processing flow.

This parameter together with the "flow_buff_size" attribute specifies a static amount of memory used for one processing flow.

The "flow_buff_size" attribute contains the size of data buffer in the traffic processing stream (bytes). This parameter together with "flow_buff_count" attribute sets the static memory size used for one processing thread. The amount of memory for one thread is "flow_buff_count". * "flow_buff_size" = 512 * 524288 = 256 MB.

The above attributes are used for engineering fine-tuning of Microolap EtherSensor and require a deep understanding of product functionality. Don't experiment with them on a production system.

Tag NetworkAdapters

Defines the settings of the network interfaces being listened to.

Tag NetworkAdapter

The NetworkAdapter tag is nested within the NetworkAdapters tag and contains settings for a specific network interface. The "enabled" attribute contains the activity status of the network interface, and if this attribute is "false", then traffic coming from this interface will be ignored.

The "rss" attribute (true/false) contains the usage status of Receive Side Scaling (RSS) technology. This is a technology that evenly distributes the network packet processing load among the processor cores to optimize performance.

The "mac" attribute contains the MAC address of the network interface. This attribute must not be changed and is read-only. The "description" attribute contains a description of the network interface and is filled in at the discretion of the administrator.

Tag Filter

The Filter tag is nested within the NetworkAdapter tag and indicates the description of the IP filter used for this network interface. The "enabled" attribute contains the status of using the IP filter, and if this attribute is "false", then the IP filter will not be used in data processing for this network interface. The "name" attribute contains the name of the IP filter profile. IP filter profiles are specified in the Filters⁽³³⁾ tag.

Tag Protocol

The Protocol tag is nested within the NetworkAdapter tag and contains a description of the Internet protocol that needs to be processed. The "enabled" attribute contains the Internet protocol processing status, and if this attribute is "false", then this Internet protocol will be ignored for this network interface. The "name" attribute contains the name of the Internet Protocol. This attribute is read-only and cannot be changed.

Example:

Network interface settings for capturing messages from DC, ICQ, IRC, MRA, MSN, XMPP/Jabber, YAHOO clients:

```
<NetworkAdapter enabled="true" mac="00-1F-C6-2D-EA-40"
  description="Marvell Yukon Controller
  88E8056 PCI-E Gigabit Ethernet">
  <Filter enabled="true" name="default" />
  <Protocol enabled="false" name="ftp" />
  <Protocol enabled="false" name="http" />
  <Protocol enabled="true" name="icq" />
  <Protocol enabled="true" name="irc" />
  <Protocol enabled="true" name="mra" />
  <Protocol enabled="true" name="msn" />
  <Protocol enabled="false" name="pop3" />
  <Protocol enabled="false" name="skype" />
  <Protocol enabled="false" name="smtp" />
  <Protocol enabled="true" name="xmpp" />
</NetworkAdapter>
```

Tag Filters

Defines IP filter profile settings.

Tag Filter

The Filter tag is nested within the Filters tag and contains the IP filter settings. The "name" attribute contains the name of the IP filter profile. The value of this attribute can be used as the value of the "NetworkAdapter / Filter / name" attribute to specify an IP filter for the network interface.

RuleGroup tag

The RuleGroup tag is nested in the Filter tag and is used to group filtering rules related to a specific traffic filtering task. The "name" attribute contains the name (or description) of the filtering rule group. The value of this attribute can be left blank.

Tag Rule

The Rule tag is nested within the RuleGroup tag and contains a description of the network traffic filtering rule. The "type" attribute contains the type of the rule, and if this attribute is equal to "accept", then network packets matching this rule will be passed for further processing, otherwise, if it is equal to "reject", network packets matching this rule will be rejected. The "src" and "dst" attributes contain an IP address, a range of IP addresses, or network parameters for filtering IP addresses that match the specified value. In the "comment" attribute, you can write a comment to the packet filtering rule.

Example:

Reject packets that pass between computers 10.1.5.10, 10.1.5.15-10.1.5.59 and network 10.1.6.0/255.255.255.0:

```
<Rule
  type="reject"
  src="10.1.5.10, 10.1.5.15-10.1.5.59"
  dst="10.1.6.0/255.255.255.0"
  proto="tcp"
  comment="" />

<Rule type="reject"
  src="10.1.6.0/255.255.255.0"
  dst="10.1.5.10, 10.1.5.15-10.1.5.59"
  proto="tcp" />
```

In the "srcport" and "dstport" attributes specify the TCP ports or TCP port ranges required for filtering.

Example:

Reject packets passing between computers 10.1.5.10, 10.1.5.15-10.1.5.59 and network 10.1.6.0/255.255.255.0 on ports 80, 443-1024:

```
<Rule
  type="reject"
  src="10.1.5.10, 10.1.5.15-10.1.5.59"
  srcport="80, 443-1024"
  dst="10.1.6.0/255.255.255.0"
  proto="tcp" />

<Rule
  type="reject"
  src="10.1.6.0/255.255.255.0"
  dst="10.1.5.10, 10.1.5.15-10.1.5.59"
  dstport="80, 443-1024"
  proto="tcp" />
```

The rules are applied linearly from top to bottom. The top line is the first filter statement, the bottom line is the last. Each line rejects or accepts only the type of packets it describes.

Example:

Reject connection packets between two hosts or a group of hosts. In this case, packets transmitted to both sides of the connection should be rejected:

```
<Rule
  type="reject"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule
  type="reject"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />
```

Also keep in mind that if no filtering rule is defined, we get all the traffic. And vice versa, if there are filtering rules, only the traffic described by these rules is processed.

Example:

Get all connections to host 10.31.5.57 only:

```
<Rule
  type="accept"
  src="10.31.5.57"
  srcport="*"
  dst="*"
  dstport="*"
  proto="tcp" />

<Rule
  type="accept"
  src="*"
  srcport="*"
  dst="10.31.5.57"
  dstport="*"
  proto="tcp" />
```

Example:

In order to cut off a group of hosts, you must first reject packets from that group, then make sure to pass all other packets, otherwise all traffic will be passed without analysis:

```
<Rule
  type="reject"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule
  type="reject"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />

<Rule
  type="accept"
  src="*"
  srcport="*"
  dst="*"
  dstport="*"
  proto="tcp" />
```

Thus, you can cut off traffic from the desired side of the proxy server.

Example:

If we describe the rules that allow traffic only for certain hosts, and deny the rest of the traffic, only traffic from these hosts will be processed:

```
<Rule
  type="accept"
  src="10.31.5.212"
  dst="10.31.5.57"
  dstport="1025"
  proto="tcp" />

<Rule type="accept"
  src="10.31.5.57"
  srcport="1025"
  dst="10.31.5.212"
  proto="tcp" />
```

3.2.3. EtherSensor EtherCAP packet filters

ATTENTION:

tcpdumpIn version 6.1 the format of packet filters has changed. If you are using packet filters, they must be converted to the new format (tcpdump/libpcap) manually. The old filters are saved in the \backup\DD.MM.YYYY\config directory.

To create and edit batch filters use the management console, section **Packet filters**:

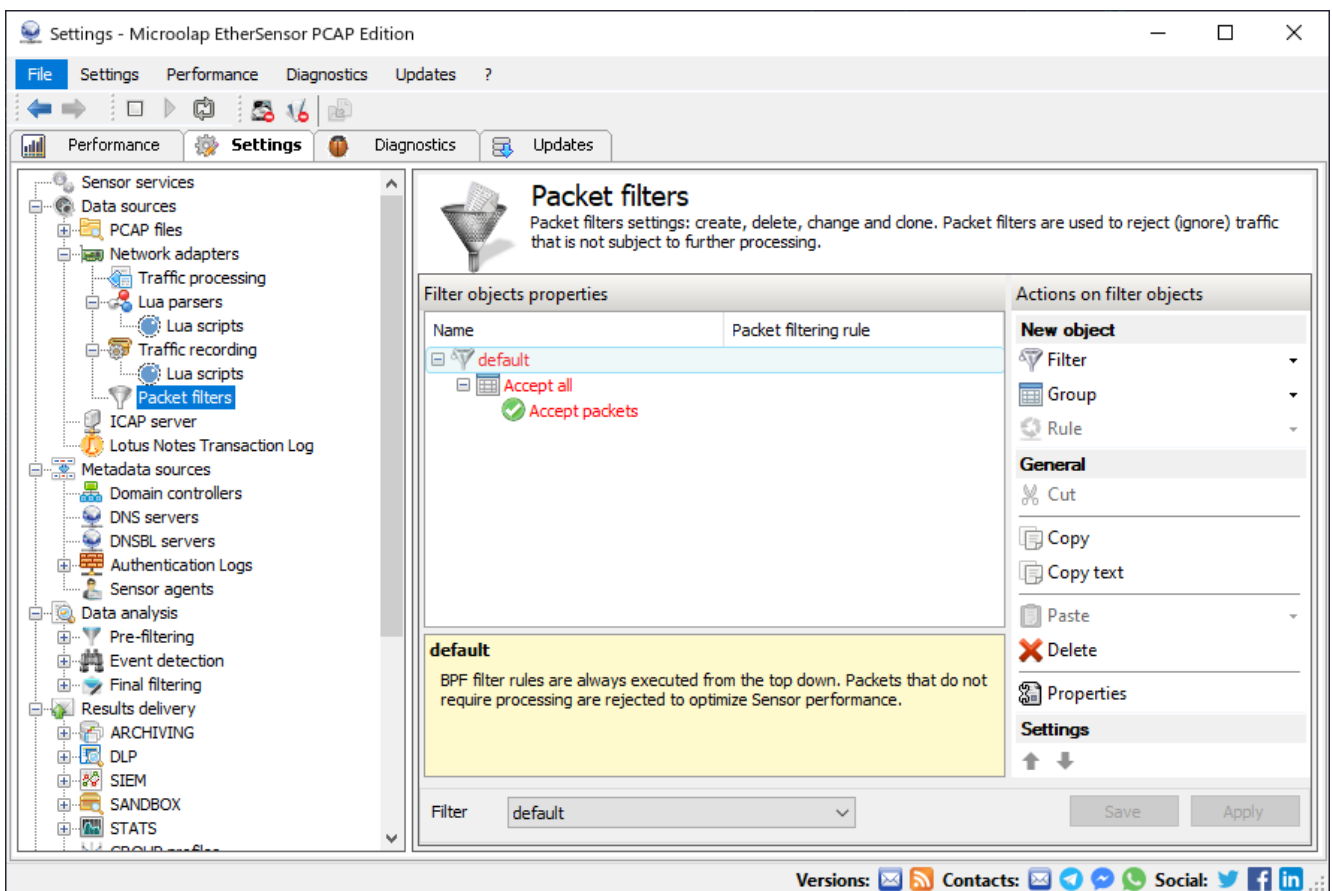


Fig.11. Packet filters settings.

Actions on filter objects panel:

Create, clone and delete filters and their objects: rules and rule groups.

Filter objects properties panel

Editing properties of the filter itself, groups and rules.

The panel for editing object properties is invoked by double-clicking on the corresponding object: filter, group of rules or rule:

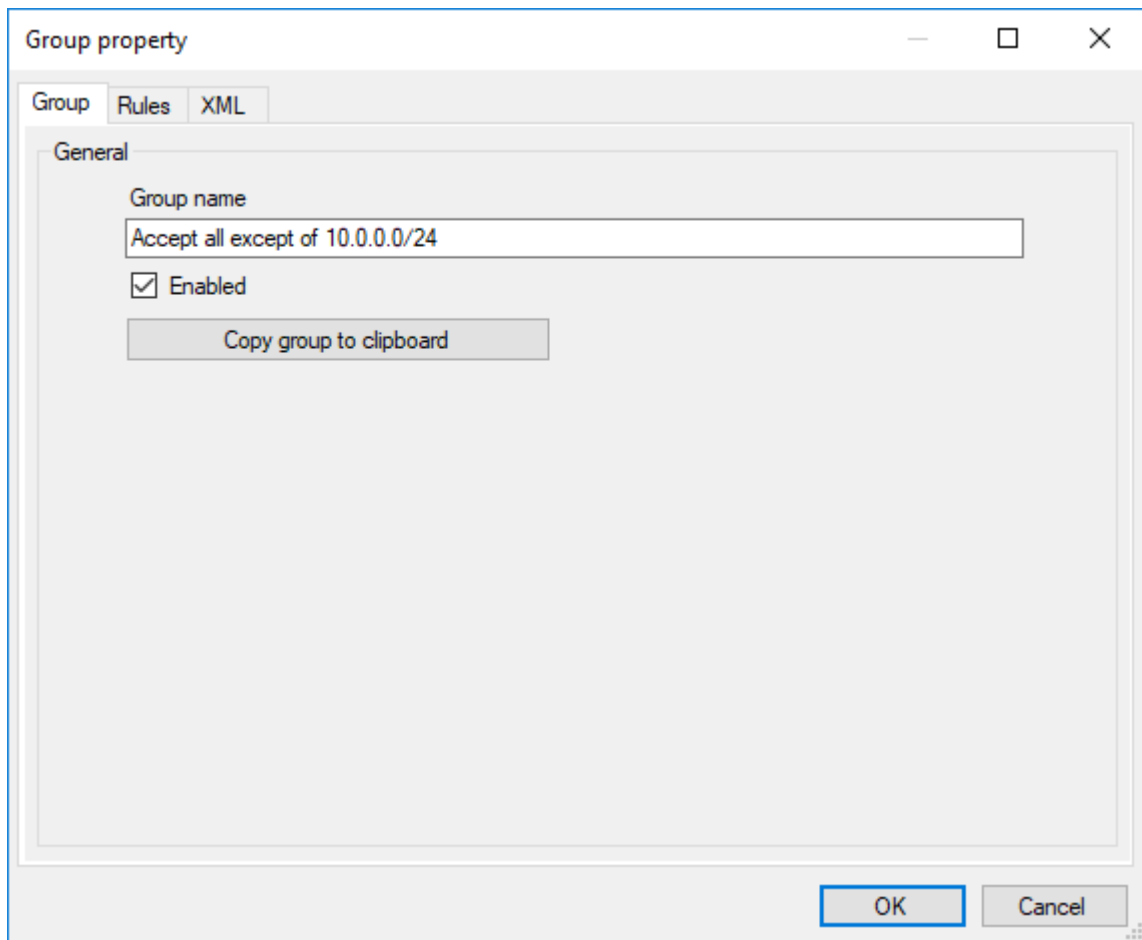


Fig.12. The panel of editing the filter's properties.

Filter name:

Filter name (at the discretion of the administrator).

Filtering rule groups tab:

The BPF program is not limited in length and the number of rules in the filter can be quite large. For the convenience of working with rules, use the ability to combine them into groups with informative names.

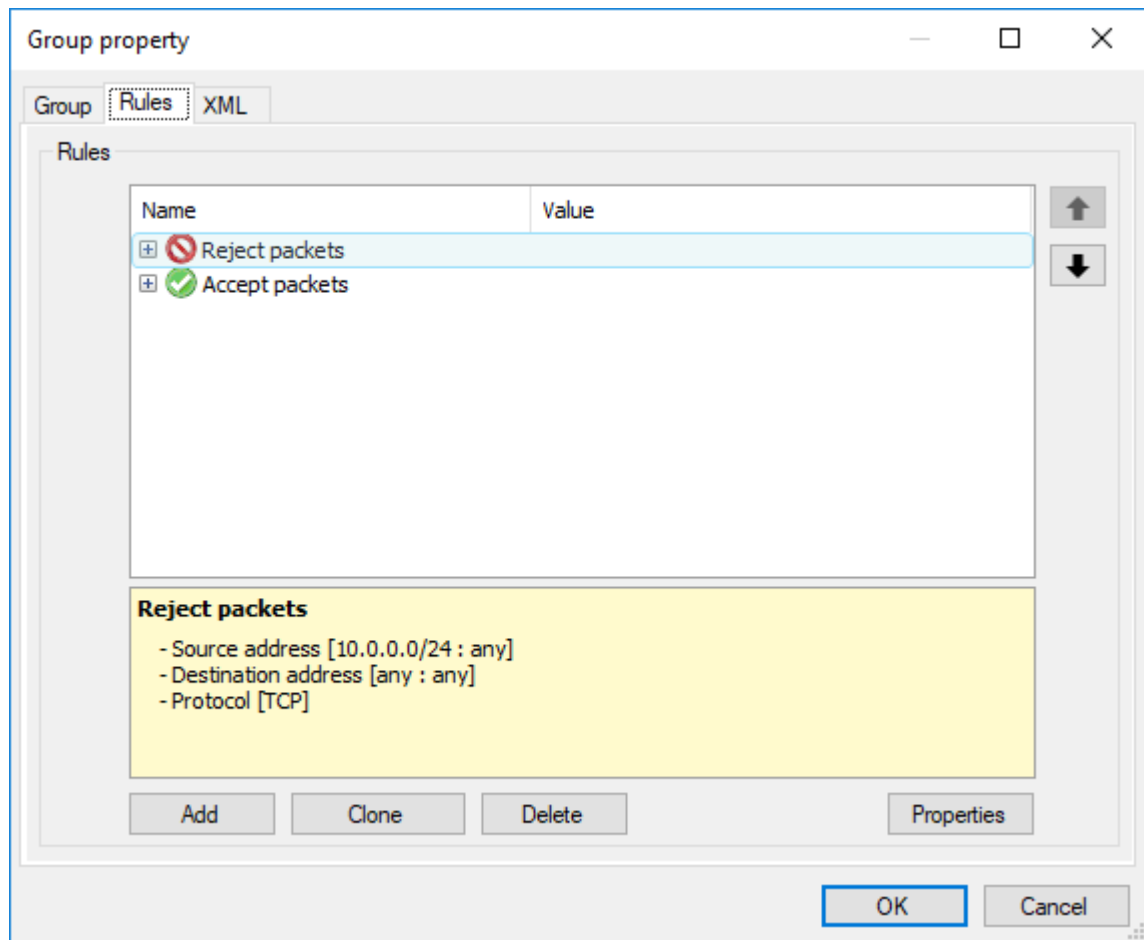


Fig.13. The panel of editing the properties of the rule group.

Group name:

The name of the rule group (at the discretion of the administrator).

Enabled checkbox:

Allows you to disable/enable a rule group.

Rules tab:

Rules can be edited both from the filter level and from the "Rules" tab of the rules group properties.

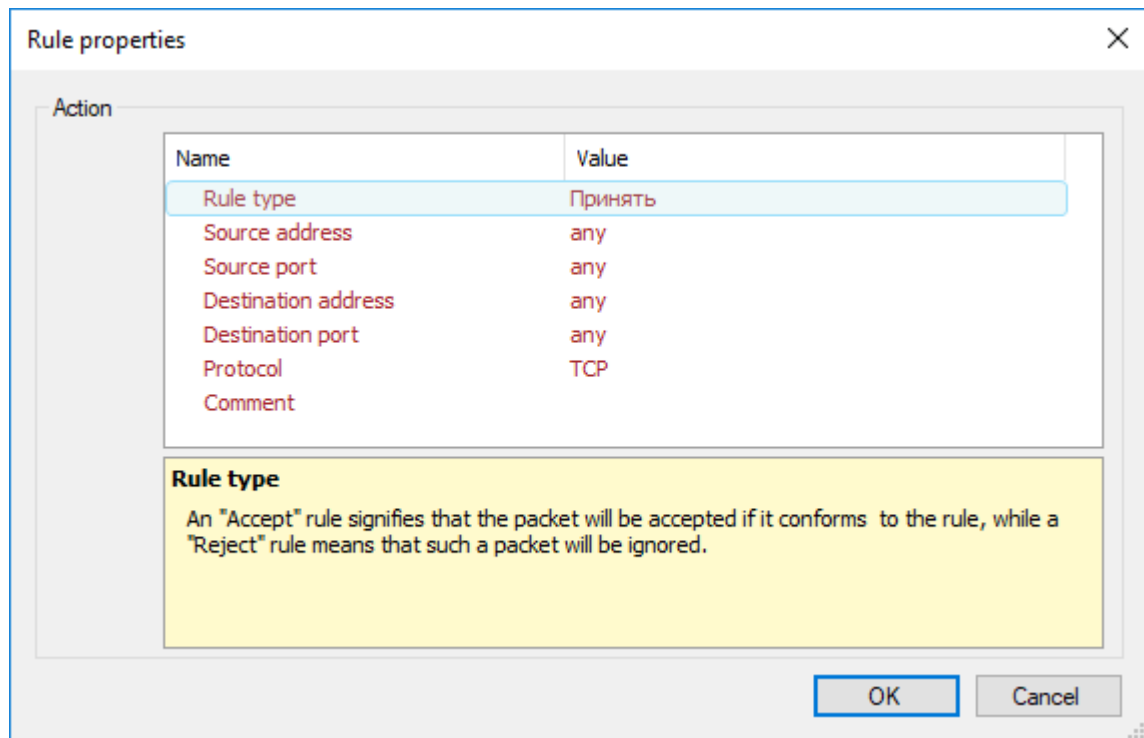


Fig.14. The panel of editing the filter's properties.

The panel for editing the action and conditions of the rule is invoked by double-clicking on the name of the rule.

Rule type:

Two types of rules have been defined: "Accept" and "Reject". A rule of type "Accept" means to accept a packet if it meets the conditions of a rule, and a rule of type "Reject" means to reject (ignore) such a packet.

The rules of the filter are always applied in sequence, shifting the rules in the filter can fundamentally change the result of its work.

Source address:

Source address. For example: any or 10.0.0.0/24 or 10.0.0.0-10.0.0.255 or a comma-separated list like 100.100.100.1-100.100.100.255, 192.168.0.0/8.

Source port:

Source port, for example: any or 80 or 80-8080 (80 to 8080), or 80-8000, 9000-10000 (80 to 8000 and 9000 to 10000).

Destination address:

Destination address. For example: any or 10.0.0.0/24 or 10.0.0.0-10.0.0.255, or a comma list of 100.100.100.1-100.100.100.255, 192.168.0.0/8.

Destination port:

Destination port, for example: any or 80 or 80-8080 (80 to 8080), or a comma list 80-8000, 9000-10000 (80 to 8000 and 9000 to 10000).

Protocol:

One of TCP, UDP, GRE, IP6, or any.

Comment

Your comment on this filtering rule.

Detailed settings of the EtherSensor EtherCAP service can be found in the Manual Setup section (configuration file)⁽³³⁾.

3.3. EtherSensor ICAP service

The EtherSensor ICAP service is an ICAP server designed to receive network traffic via ICAP from any ICAP clients in REQMOD mode.

ICAP (Internet Content Adaptation Protocol) is designed to work with the HTTP protocol only and is used for content filtering and detection of malicious content (viruses, spyware/malware).

The ICAP client is a system through which HTTP traffic is transmitted. Such system can be various HTTP proxies supporting ICAP (for example, SQUID, Blue Coat Proxy SG, Cisco IronPort S, Webwasher). After receiving data from the client, some ICAP servers process them and, if necessary, modify the data.

Then the data is returned to the ICAP client, and it passes it on to the server or client, depending on the direction in which it was transmitted.

Since the Microolap EtherSensor ICAP server uses traffic received from ICAP clients only for analysis, the traffic is always returned to the ICAP client unchanged.

System architecture when using ICAP protocol:

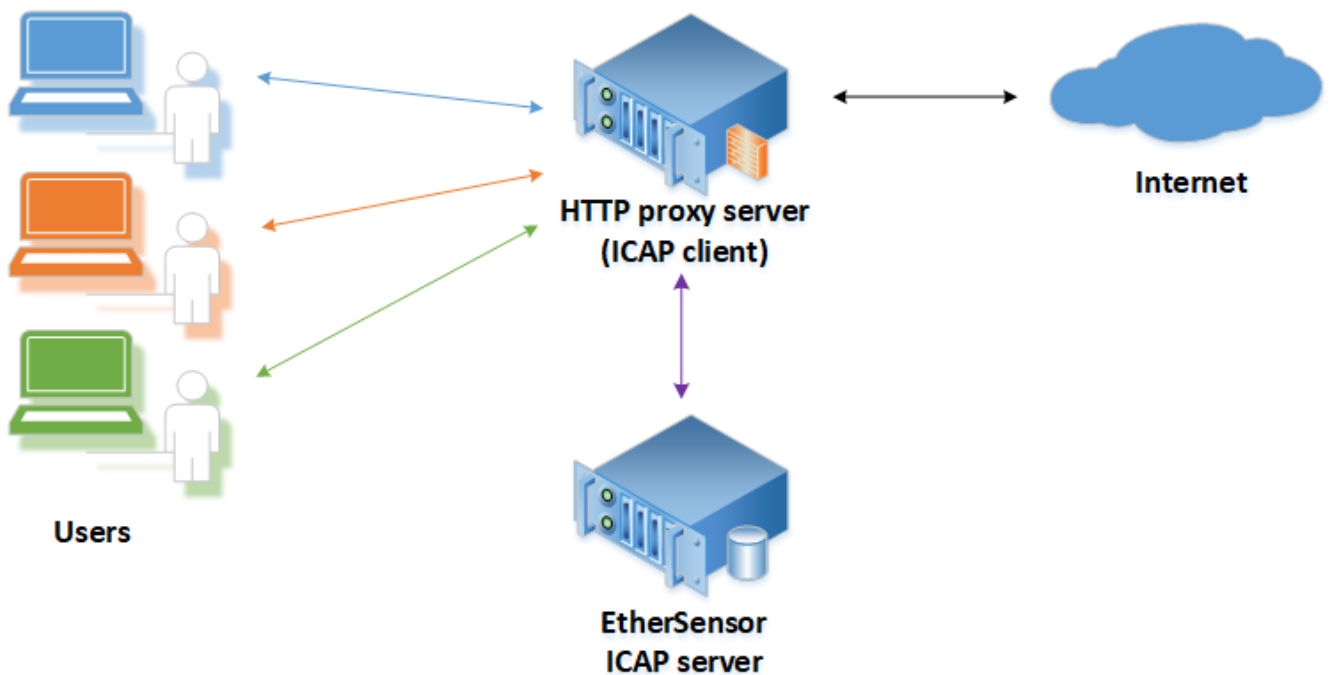


Fig.15. The scheme of interaction between the EtherSensor ICAP service and the ICAP client.

Some ICAP clients use header extensions that allow them to send information about the user who is logged in to the proxy server to the ICAP server. This information is taken into account in the further processing of messages in EtherSensor.

Command line parameters

The Windows service EtherSensor ICAP is installed during the installation of Microolap EtherSensor with automatic launch. However, if necessary, the `sensor_icap.exe` process can be run as a Windows application with the following command line parameters:

/process

Run the process `sensor_icap.exe` as a normal Windows Win32 process (use for debugging is possible).

/service

Run as a Windows service.

/config

Save the default service configuration.

3.3.1. EtherSensor ICAP settings

The EtherSensor ICAP service is an ICAP server designed to work with ICAP clients and has settings specific to this type of servers:

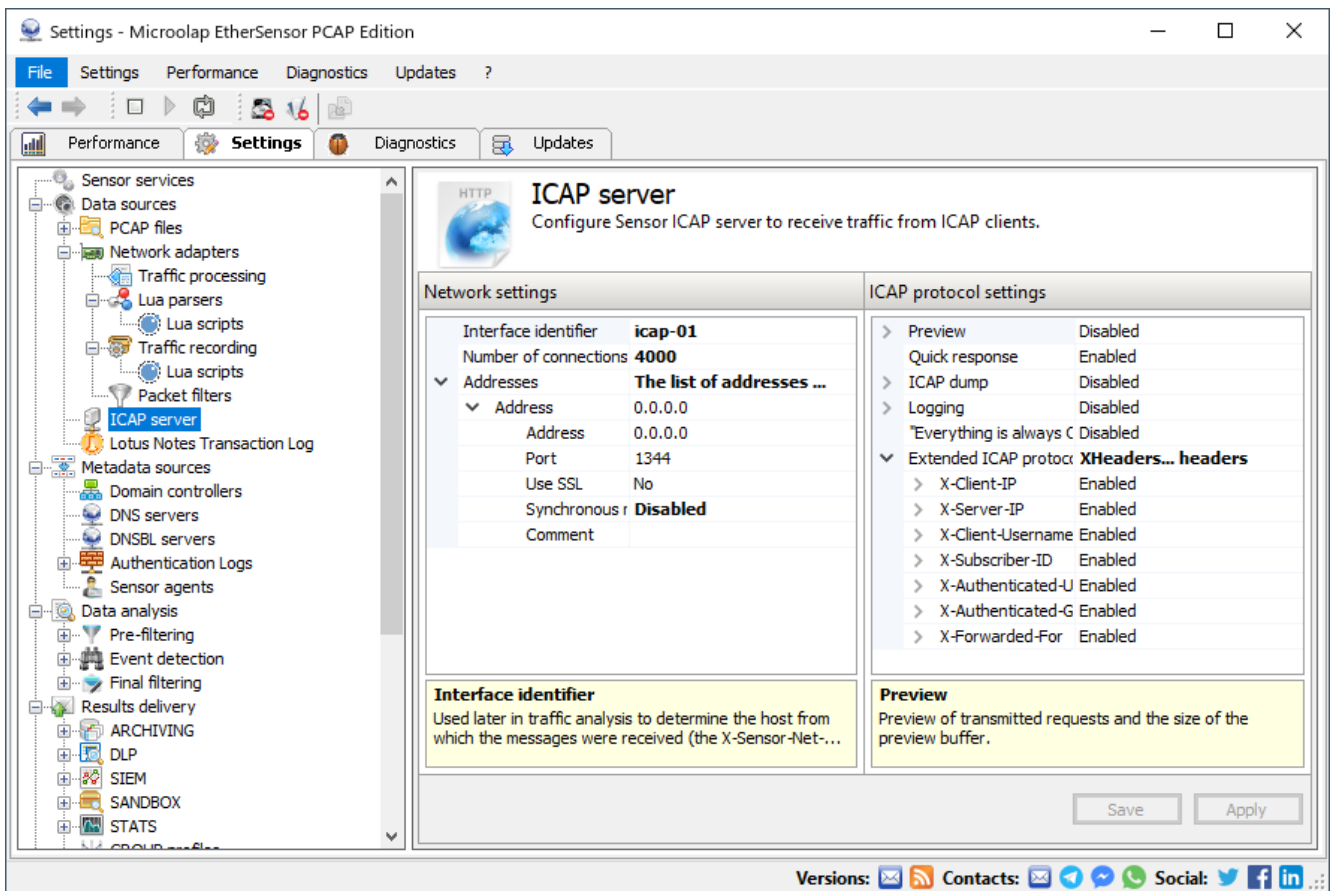


Fig.16. Settings of the EtherSensor ICAP service.

Number of connections:

Maximum number of ICAP server connections with clients. The approximate amount of RAM consumed per connection is about 8KB.

Addresses:

IP addresses/ports listened by the ICAP server.

Preview:

Enabling/disabling preview, buffer size.

Quick response:

Enable/disable fast short response Allow 204 (No modifications).

Synchronous mode:

Enable/disable the use of the ICAP server's synchronous operation mode. In synchronous mode, the ICAP server first receives the entire request and only then sends it back to the ICAP client, even if the request represents a significant amount of data, such as an ISO disk image, etc. The need for synchronous mode may result from ICAP client settings (e.g. Blue Coat, IronPort, etc.).

ICAP dump:

Dump of all data exchanged between ICAP client and server. It is used only for debugging interaction with third-party software.

Logging:

ICAP server logs HTTP requests for EtherSensor Watcher service.

"Everything is always OK" mode:

In this mode, the ICAP server, when errors are detected in the ICAP protocol from the client side, respond to it not with the appropriate error code, but with the Allow 204 (No modifications) code.

Extended ICAP protocol headers.:

Allows ICAP clients to be notified that the server supports the specified headers.

Supported extended ICAP header names:

X-Client-IP

X-Server-IP

X-Client-Username

X-Subscriber-ID

X-Authenticated-User

X-Authenticated-Groups.

These headers are converted into the following headers during the processing of ICAP traffic if a message signature is detected when sending a message to the system-consumer:

X-Sensor-Icap-Client-Username

X-Sensor-Icap-Subscriber-ID

X-Sensor-Icap-Authenticated-User

X-Sensor-Icap-Authenticated-Groups

X-Client-IP and X-Server-IP header values are saved in X-Sensor-Src-Address, X-Sensor-Dst-Address headers.

EtherSensor ICAP configuration file

The configuration of the EtherSensor ICAP service is contained in the XML file icap.xml, located in the common configuration directory Microolap EtherSensor [INSTALLDIR]\config.

3.4. EtherSensor LotusTXN service

The EtherSensor LotusTXN service is designed to extract Lotus Notes Transaction Log messages.

EtherSensor LotusTXN extracts messages from the Lotus Notes Transaction Log files and then passes these messages to the EtherSensor Analyser service for further processing.

In the current version Microolap EtherSensor (6.1) EtherSensor LotusTXN can track several Lotus Notes Transaction Log directories simultaneously. This allows the administrator to monitor several Lotus Notes systems at the same time with a single installation of EtherSensor, while the directories of Lotus Notes Transaction Log can be both local and remote.

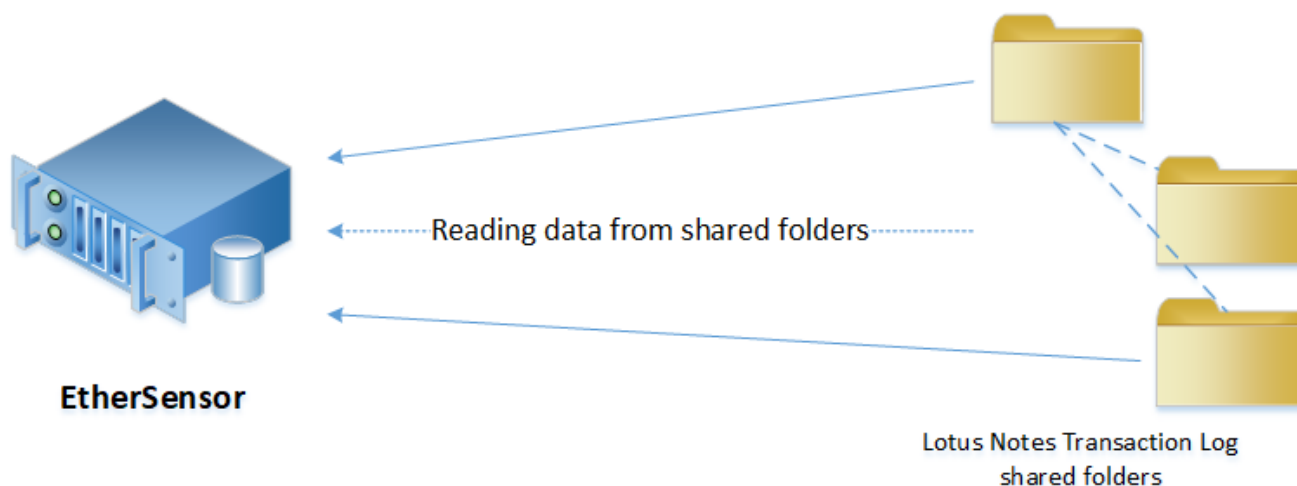


Fig.17. Scheme of the EtherSensor LotusTXN service work.

Use the EtherSensor LotusTXN service only if your Lotus Notes system uses encryption or if you need to monitor the Lotus Notes Transaction Log on remote servers.

Extraction of unencrypted Lotus Notes messages from the traffic along with SMTP, POP3 and IMAP4 is provided by the service EtherSensor EtherCAP.

Command line parameters

The Windows service EtherSensor LotusTXN is installed during the installation of Microolap EtherSensor with automatic launch. However, if necessary, the `sensor_lotustxn.exe` process can be run as a Windows application with the following command line parameters:

/process

Run the process `sensor_lotustxn.exe` as a normal Windows Win32 process (can be used for debugging).

/service

Run as a Windows service.

/config

Save the default service configuration.

3.4.1. EtherSensor LotusTXN settings

The EtherSensor LotusTXN service allows monitoring and reconstruction of Lotus Notes system messages by extracting them from Lotus Notes Transaction Log.

If the Lotus Notes system does not use encryption, the EtherSensor EtherCAP service is responsible for extracting Lotus Notes messages from traffic along with SMTP, POP3 and IMAP4.

In order to configure the EtherSensor LotusTXN service to track messages, it needs to specify the Lotus Notes transaction log directories to track.

Information about the settings of Lotus Notes Transaction Log directories is contained in the file `lotustxn.xml` located in the directory `[INSTALLDIR]\config`, and it can be edited directly in the configuration file by any text editor.

In the management console (file `sensor_console.exe`), the EtherSensor LotusTXN service is configured as follows:

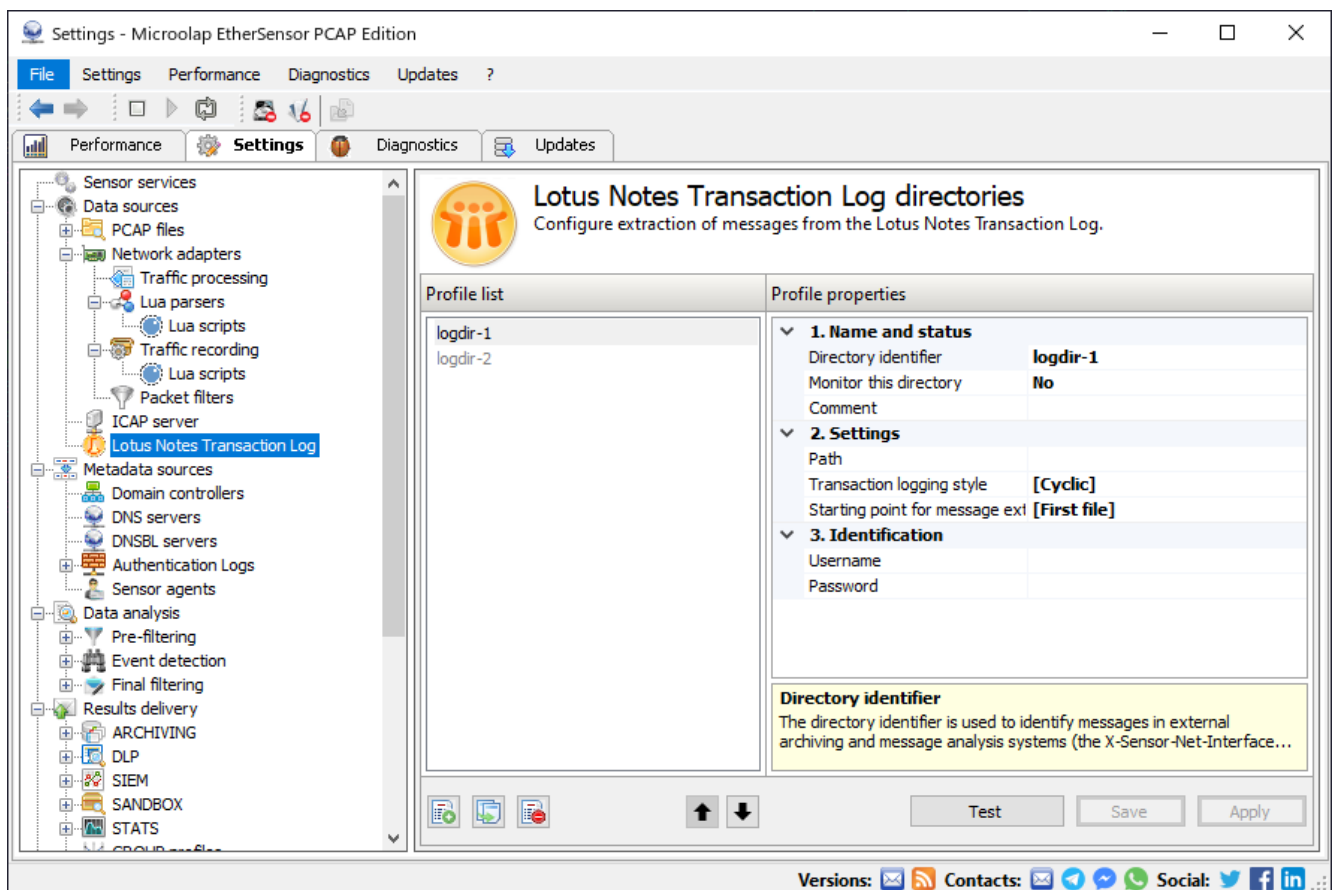


Fig.18. Settings of the EtherSensor LotusTXN service.

Directory identifier:

Identifier of the Lotus Notes Transaction Log directory. Used to identify the data source in EtherSensor.

Transaction logging style:

It serves to specify the transaction logging style used by the Lotus Notes system for this directory.

Starting point for message extraction:

Start point for extracting messages. It is used to specify EtherSensor from which file in the Lotus Notes Transaction Log directory the extraction process should start.

Username:

Defines the Windows user name to access the transaction directory. The value of this parameter must be specified in UPN (user principal name) format. Example: administrator@example.com, where administrator is the user name, example.com is the domain of the user. If this parameter is not specified, the rights of the account under which the EtherSensor LotusTXN service is running are used to access the transaction log files. It should be noted that when EtherSensor is installed, by default all services are launched with SYSTEM user rights.

Password:

Defines the user password to access the Lotus Notes transaction directory.

Path:

Full path to the directory with Lotus Notes transaction logs.

EtherSensor LotusTXN configuration file

The configuration of the service EtherSensor LotusTXN is specified in the XML file lotustxn.xml, located in the common configuration directory [INSTALLDIR]\config.

3.5. EtherSensor Identity service

The EtherSensor Identity service is responsible for accumulation and processing of metadata that is used in Microolap EtherSensor to solve the problem of agentless binding of captured network traffic object to a specific host or user.

In 6.1 EtherSensor Identity interacts with the following metadata sources:

Domain Controllers ⁽⁵²⁾

EtherSensor connects to domain controllers as a WMI client and receives data about network users who have been authenticated (an account with the appropriate rights is required to receive data from the domain controller). This way EtherSensor can see active users on the corporate network, their IP addresses and computer names, and then use this data to bind the captured object to a user or host.

Alternative:

To obtain this data, instead of connecting to a domain controller, you can use the logon script installed on workstations/servers that sends user authentication data to the SYSLOG server of the EtherSensor Identity service.

DNS servers⁽⁵⁵⁾

EtherSensor uses DNS servers to associate IP addresses of captured objects with hostnames in the organization's network and on the Internet.

DNSBL servers⁽⁵⁶⁾

EtherSensor uses DNSBL servers to detect objects from connections to insecure Internet resources (both by IP addresses and by DNS names).

Authentication Logs⁽⁵⁸⁾

EtherSensor supports receiving authentication messages from various external devices and systems via SYSLOG protocol. Supported transport protocols are UDP, TCP, and TLS over TCP.

External systems providing authentication data to the EtherSensor Identity service via the SYSLOG protocol may be:

Firewalls:

Palo Alto Networks PA, Check Point NGFW etc.

Proxy servers:

Symantec Proxy SG, Cisco IronPort etc.

Workstations/servers:

Linux, Mac OS etc.

Software agents:

DLP agents, EDR agents, etc.

Authentication messages from different sources differ in formats and data sets. To support a specific source, additional configuration of both the source itself and the EtherSensor Identity service is required.

Microolap EtherSensor Agents⁽⁶²⁾

EtherSensor can also interact with its own agents installed on workstations or servers. The agents provide the EtherSensor agent server with information about the network connections of the local user up to the name of the process participating in the connection: this information is especially relevant if terminal RDP servers are used in the organization.

General scheme of the EtherSensor Identity service work:

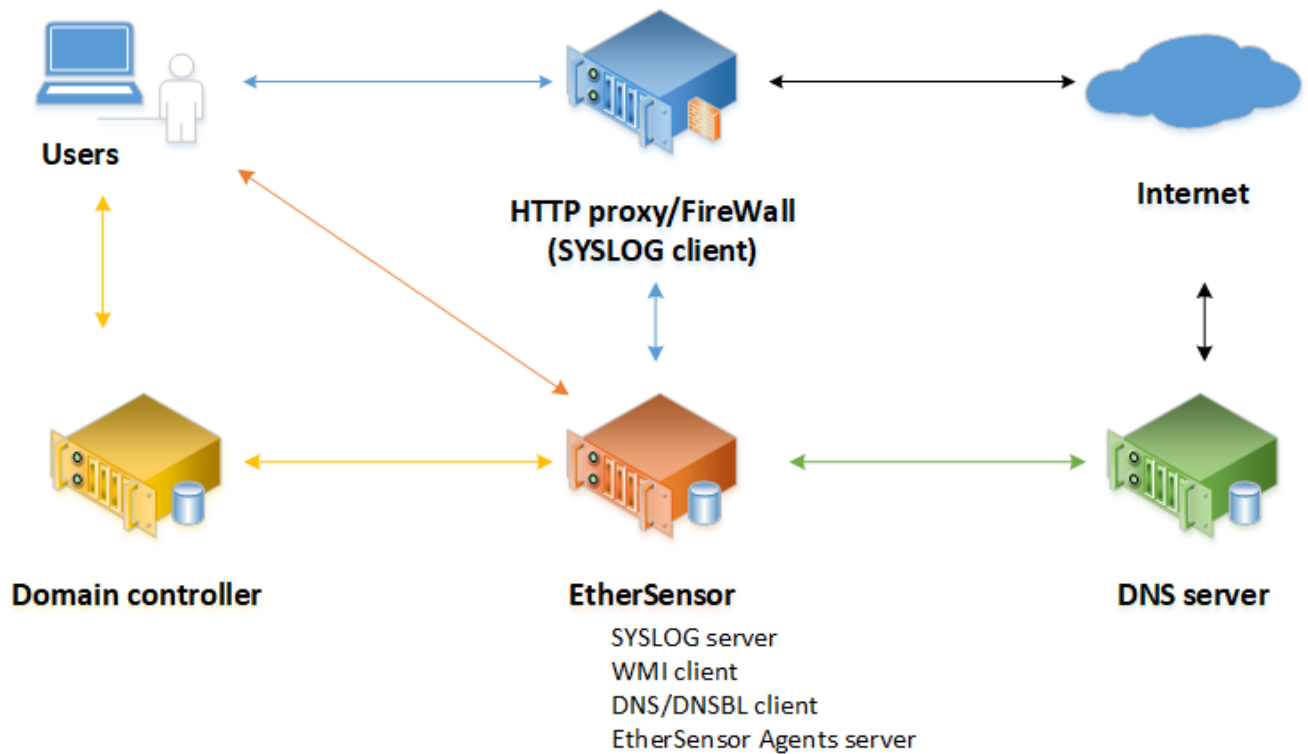


Fig.19. Scheme of the EtherSensor Identity service work.

EtherSensor Identity configuration file

The configuration of the EtherSensor Identity service is contained in the XML file `identity.xml`, located in the common configuration directory `[INSTALLDIR]\config`.

Command line parameters

The Windows service EtherSensor Identity is installed during the installation of Microolap EtherSensor with automatic launch. However, if necessary, the `sensor_identity.exe` process can be run as a Windows application with the following command line parameters:

/process

Run the process `sensor_pcap.exe` as a normal Windows Win32 process (use for debugging is possible).

/service

Run as a Windows service.

/config

Save the default service configuration.

3.5.1. EtherSensor Identity settings

The settings of the EtherSensor Identity service are based on the concept of predefined profiles. Profiles can be activated as needed and deactivated when they are no longer needed. There is no need to delete unused profiles.

In order to prepare the EtherSensor Identity service for operation, you should configure metadata provider profiles:

1. Domain controllers ⁽⁵²⁾
2. DNS servers ⁽⁵⁵⁾
3. DNSBL servers ⁽⁵⁶⁾
4. Authentication Logs ⁽⁵⁸⁾
5. EtherSensor Agents server ⁽⁶²⁾

In addition, you should consider installing on workstations and servers a logon script ⁽⁵⁴⁾ that sends user authentication data to the EtherSensor Agents server ⁽⁶³⁾.

3.5.1.1. Domain Controllers

EtherSensor connects to domain controllers as a WMI client and receives data about network users who have been authenticated (an account with the appropriate rights is required to get data from the domain controller).

Thus EtherSensor can see active users on the corporate network, their IP addresses and computer names, and then use this data to bind the captured object to a user or host.

Alternative:

To get this data instead of connecting to a domain controller, you can use the logon script ⁽⁵⁴⁾ installed on the workstations/servers that sends user authentication data to the EtherSensor Agents server.

Configuring the domain controller access profile:

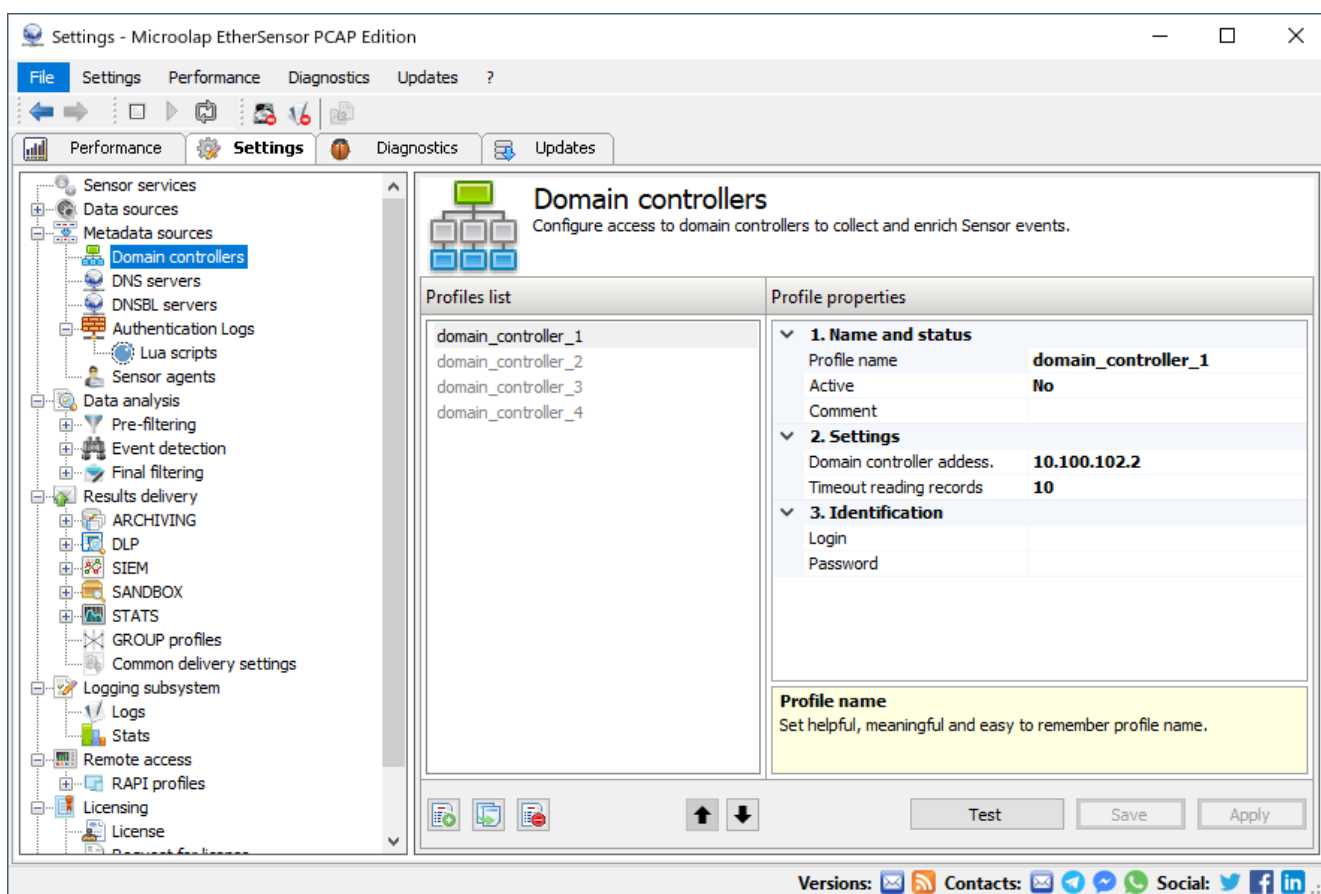


Fig.20. Configuring the domain controller profile for the EtherSensor Identity service.

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

The domain controller profile is not used in message analysis if it is disabled.

Comment

Your comment for this profile.

Domain controller address.

The IP address or domain controller name for accessing the OS logs.

Timeout reading records

Timeout in seconds to read entries from OS logs.

Login

Login for read access to domain controller logs in `<domain name>\<user name>` syntax.

Password

Password for read access to domain controller logs.

3.5.1.1.1. Alternative: logon script

An alternative way to obtain information about user authentication in the system or in the domain is to use a script that runs after a user logs on.

An example of such a script in the PowerShell language is given below. This script tells the EtherSensor Agents server⁽⁶³⁾ IP address, user name, computer name and user sid. This data allows EtherSensor to associate the IP address of the object extracted from traffic with the user or host.

Logon script and its launch must be configured through Group Policy Tools for the user logon event.

Sample logon script text:

```
function GetCUSID
{
    Param ( $CUIdentity )
    $MyID = new-object System.Security.Principal.NTAccount($CUIdentity)
    return $MyID.Translate([System.Security.Principal.SecurityIdentifier]).ToString()
}

# EtherSensor address
$Server = '10.0.1.123&apos

#0=EMERG, 1=Alert, 2=CRIT, 3=ERR, 4=WARNING, 5=NOTICE, 6=INFO, 7=DEBUG
$Severity = '5&apos

#(16-23)=LOCAL0-LOCAL7
$Facility = '16&apos

$HostName = $env:ComputerName
$UserName = $env:UserName
$UserSID = GetCUSID([Environment]::UserName)

# Create a UDP Client Object
$UDPClient = New-Object System.Net.Sockets.UdpClient
$UDPClient.Connect($Server, 10514)

# Calculate the priority
$Priority = ([int]$Facility * 8) + [int]$Severity

#Time format the SW syslog understands
$Timestamp = (Get-Date).ToString("MMM dd HH:mm:ss", [CultureInfo]::GetCultureInfo('en-US'))

# Assemble the full syslog formatted message
$FullSyslogMessage = "<{0}>{1} computername={2}, username={3}, sid={4}" -f $Priority,
$Timestamp, $HostName, $UserName, $UserSID

# Create a UTF8 Encoding object
$Encoding = [System.Text.Encoding]::UTF8

# Convert into byte array representation
$ByteSyslogMessage = $Encoding.GetBytes($FullSyslogMessage)

# Send the Message
$UDPClient.Send($ByteSyslogMessage, $ByteSyslogMessage.Length)
```

3.5.1.2. DNS servers

EtherSensor uses DNS servers to associate IP addresses of captured objects with hostnames in the organization's network and on the Internet.

When creating filtering rules, you may need to be able to use domain names, while only IP addresses are available in the attributes of the captured objects.

In order to be able to use this feature in real time during the filtering process (you should also take into account the dynamic assignment of IP addresses by DHCP), in the section **DNS** define and configure the available DNS servers.

An example of how to configure DNS server access profile:

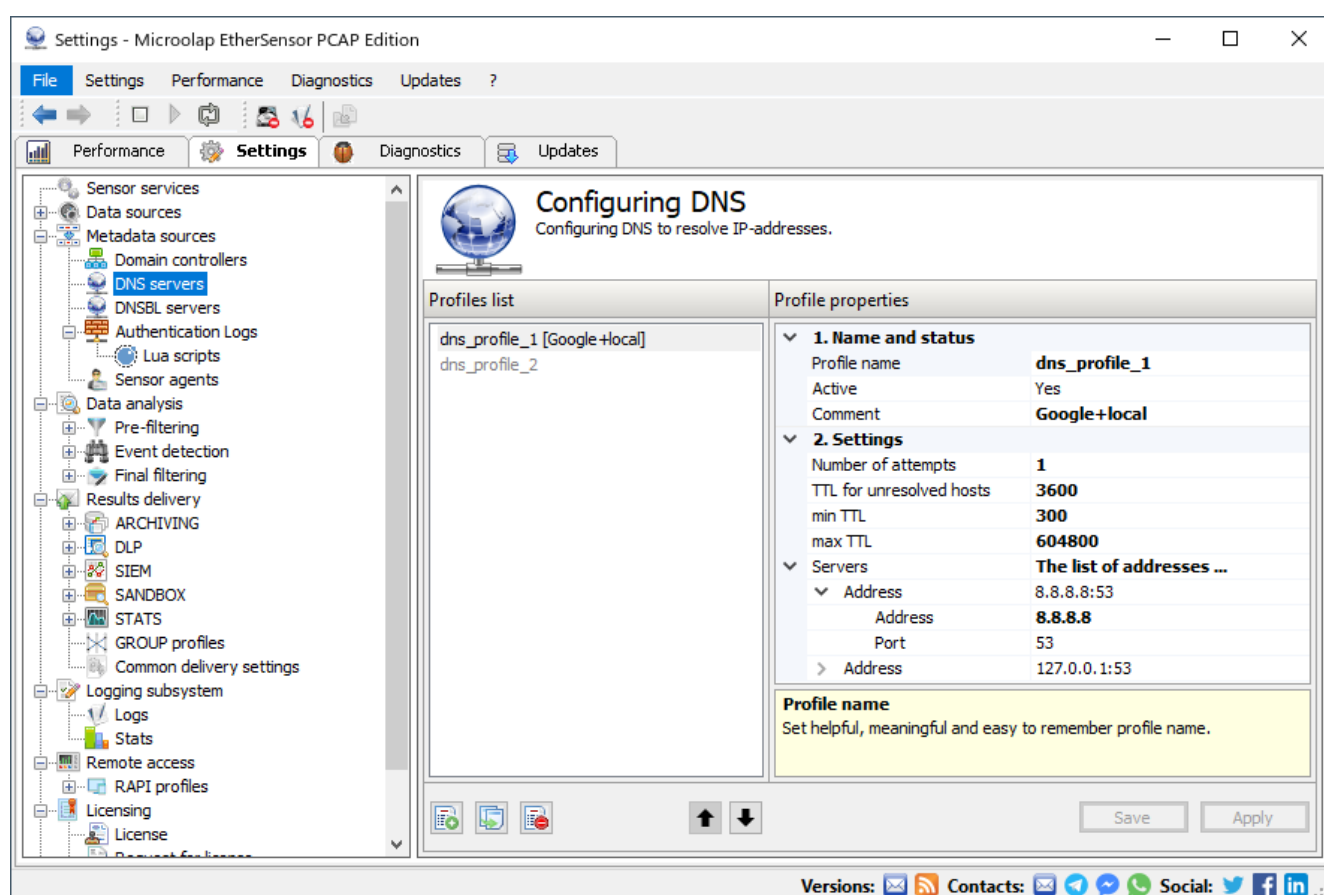


Fig.21. Setting up a DNS profile for the EtherSensor Identity service.

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

The DNS server profile is not used in message analysis if it is disabled.

Comment

Your comment for this profile.

Number of attempts

Number of attempts to poll each server.

TTL for unresolved hosts

TTL for unresolved hosts: this is the number of seconds that this host will be stored in the cache as unknown. (default: 3600 = 1 hour) (43200 = 12 hours, 86400 = 24 hours, 172800 = 48 hours).

min TTL

Minimum storage TTL in the cache for host names received from DNS servers.

max TTL

Maximum storage TTL in the cache for host names received from DNS servers.

Servers

List of DNS-servers.

Address

DNS-server address

Port

DNS-server port

3.5.1.3. DNSBL servers

While filtering messages, you may also need to categorize the messages. For this purpose, EtherSensor has a function of interaction with DNSBL servers to detect spam messages in the SMTP flow. The configuration of this function is completely identical to that of DNS servers.

EtherSensor uses DNSBL servers to detect objects from connections to insecure Internet resources (both by IP addresses and by DNS names).

An example of configuring the access profile to a DNSBL server:

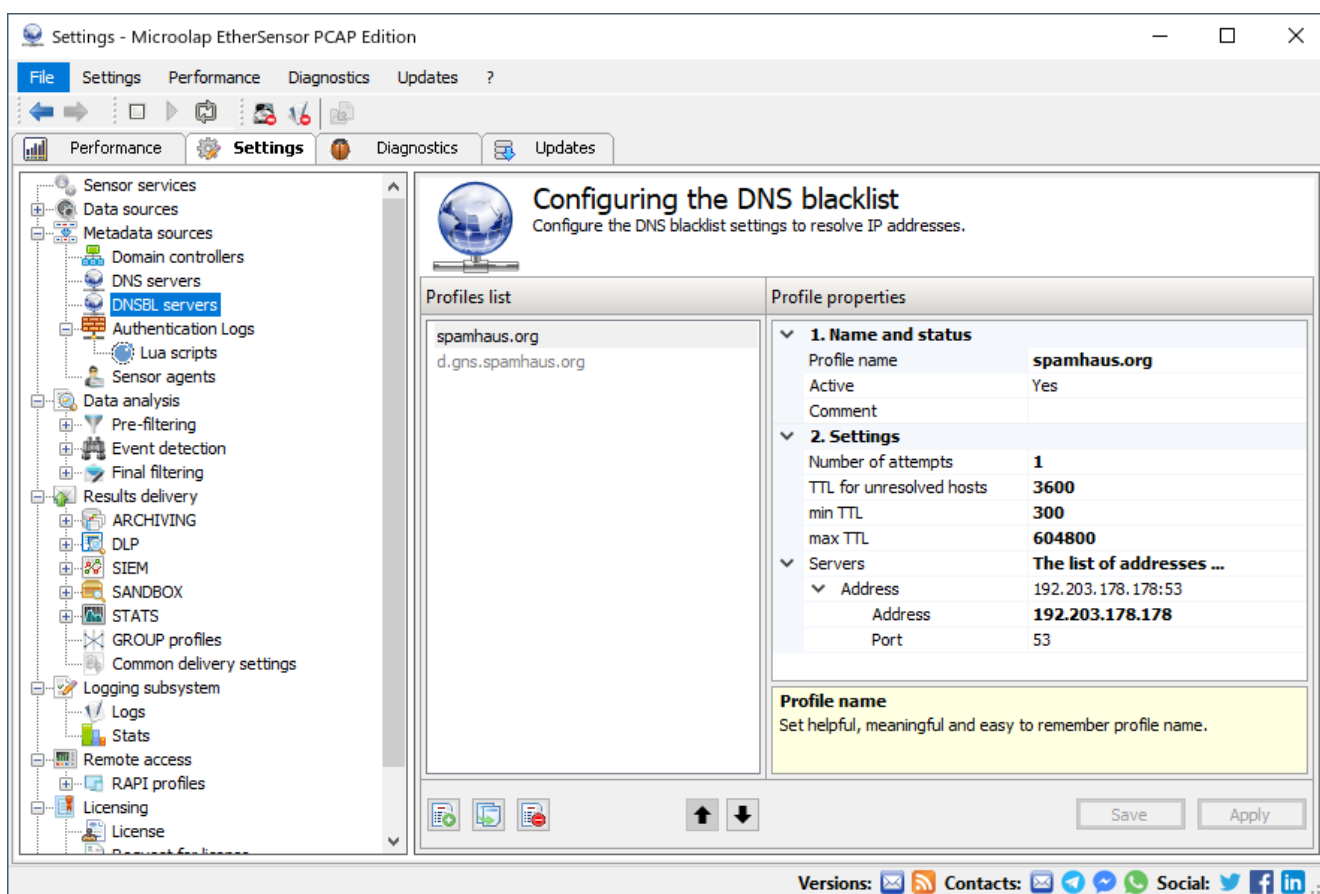


Fig.22. Setting up a DNSBL profile for the EtherSensor Identity service.

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

The DNSBL server profile is not used in message analysis if it is disabled.

Comment

Your comment for this profile.

Number of attempts

Number of attempts to poll each server.

TTL for unresolved hosts

TTL for unresolved hosts: this is the number of seconds that this host will be stored in the cache as unknown. (default: 3600 = 1 hour) (43200 = 12 hours, 86400 = 24 hours, 172800 = 48 hours).

min TTL

Minimum storage TTL in the cache for host names received from DNSBL servers.

max TTL

Maximum storage TTL in the cache for host names received from DNSBL servers.

Servers

List of DNSBL-servers.

Address

DNSBL-server address

Port

DNSBL-server port

3.5.1.4. Authentication Logs

The EtherSensor Identity service includes a SYSLOG server for receiving authentication log entries from various SYSLOG clients: firewalls, proxies, software DLP/EDR agents, etc.

According to the SYSLOG configuration profiles, the SYSLOG server can listen to different combinations of IP address:port or 0.0.0.0:port, in which case it will listen to all IP addresses on a given port.

An example of how to configure the authentication log profile:

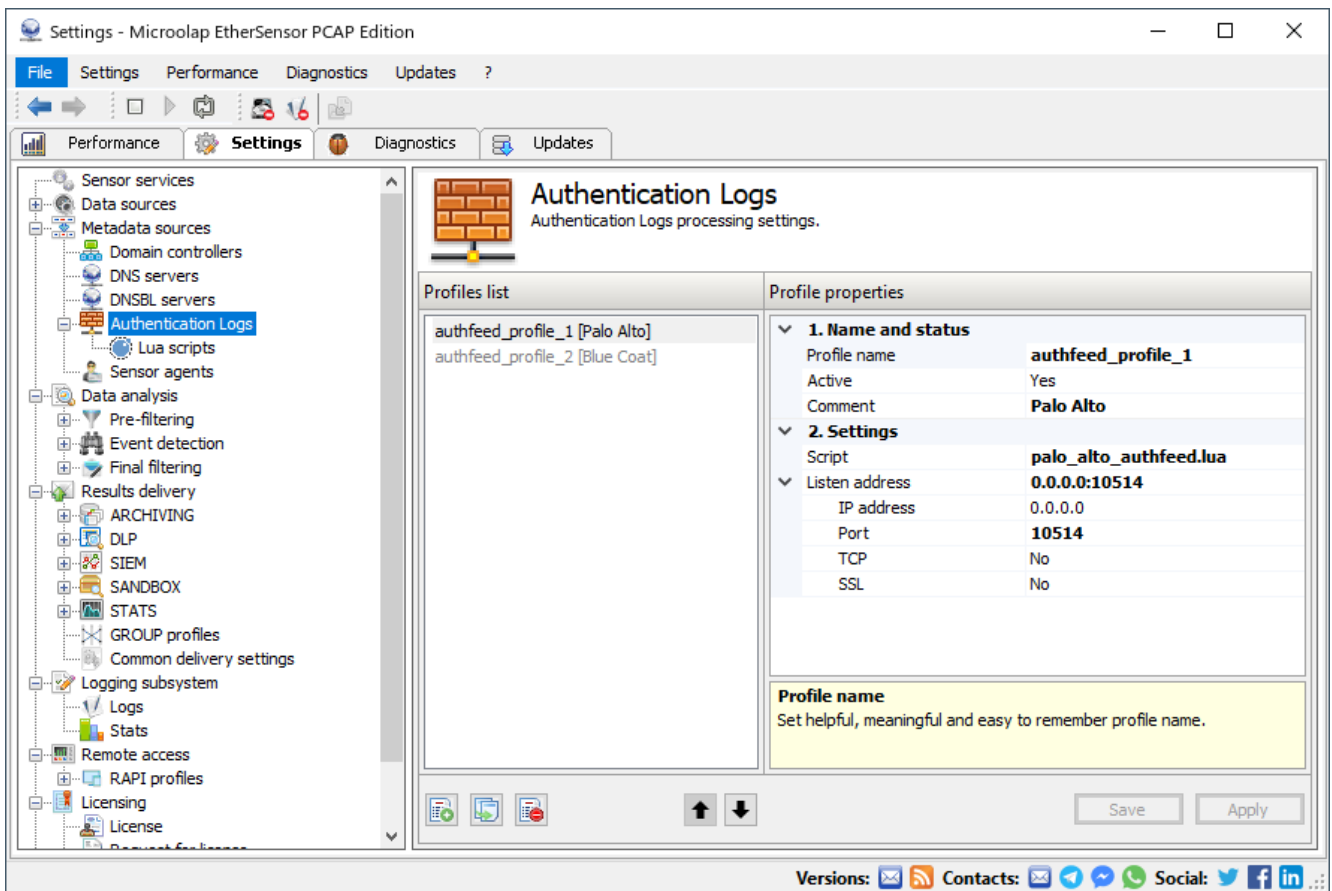


Fig.23. Configuring the authentication log profile for the EtherSensor Identity service.

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

Authentication Logs processing profile is not used in message analysis if it is disabled.

Comment

Your comment for this profile.

Script

A profile associated script that runs to process the Authentication Logs SYSLOG message.

Listen address

Set local address to listen the Authentication Logs.

IP address

Set IP address to listen the Authentication Logs.

Port

Set local port to listen the Authentication Logs.

TCP

Allows use of the TCP protocol to receive SYSLOG messages. This is necessary if SSL encryption is used.

SSL

Enables/disables the use of SSL encryption when receiving messages.

If an organization cannot provide the EtherSensor server with access to domain controllers or the Log Collector, there is another way to obtain data to bind domain users to objects extracted from traffic.

Install a free nxlog utility on your domain controllers or Log Collector and configure it to send Security Log events to the EtherSensor server.

In this case data are sent to the server EtherSensor via the SYSLOG protocol (**TCP**).

Below is an example of an nxlog configuration file sending Kerberos Ticket Authentication (EventID 4768) events to the EtherSensor server (in the example it is IP 10.100.0.100).

```

## This is a sample NXLog configuration file created by Loggly. June 2013
## See the nxlog reference manual about the configuration options.
## It should be installed locally and is also available
## online at http://nxlog.org/nxlog-docs/en/nxlog-reference-manual.html

## Please set the ROOT to the folder your nxlog was installed into,
## otherwise it will not start.
#define ROOT C:\\Program Files\\nxlog
#define ROOT_STRING C:\\Program Files\\nxlog
define ROOT C:\\Program Files (x86)\\nxlog
define ROOT_STRING C:\\Program Files (x86)\\nxlog
define CERTDIR %ROOT%\\cert

Moduledir %ROOT%\\modules
CacheDir %ROOT%\\data
Pidfile %ROOT%\\data\\nxlog.pid
SpoolDir %ROOT%\\data
LogFile %ROOT%\\data\\nxlog.log

<Extension json>
  Module xm_json
</Extension>
<Extension syslog>
  Module xm_syslog
</Extension>
<Input internal>
  Module im_internal
  Exec $Message = to_json()
</Input>
# Windows Event Log
<Input eventlog>
# Uncomment im_msvistalog for Windows Vista/2008 and later
  Module im_msvistalog
#Uncomment im_mseventlog for Windows XP/2000/2003
#Module im_mseventlog
#Send only EventID 4768 (Get Kerberos Ticket)
  Exec if $EventID NOT IN (4768) drop()
  Exec $Message = to_json()
</Input>
<Processor buffer>
Module pm_buffer
# 100Mb disk buffer
MaxSize 102400
Type disk
</Processor>
<Output out_ethersensor>
  Module om_tcp
  Host 10.100.0.100
  Port 516
  Exec to_syslog_ietf()
  Exec $raw_event =~ s/(\\[.*/g; $raw_event = replace($raw_event, '{',
  '[CUSTOMER_TOKEN@41058 tag="windows"] {' , 1)
#Use the following line for debugging (uncomment the fileop extension above as well)
#Exec file_write("C:\\Program Files (x86)\\nxlog\\data\\nxlog_output.log", $raw_event)
</Output>
<Route 1>
  Path internal, eventlog => buffer => out_ethersensor
</Route>

```

In the EtherSensor server management console under **Settings -- Metadata sources -- Authentication Logs** make the **authfeed_nxlog** profile active. To do this, set the flag **Active** and make sure that the correct **Port** and the protocol (**TCP**) are specified.

3.5.1.4.1. Lua scripts

The function of parsing SYSLOG messages using Lua scripts assigned to authentication log profiles is in the pre-release stage.

If you want to experiment with us, write us at support@microolap.com.

3.5.1.5. EtherSensor Agens server

EtherSensor Identity service includes the EtherSensor Agents server that interacts with agents⁽⁶³⁾ installed on workstations and servers.

Example of configuring EtherSensor Agents server:

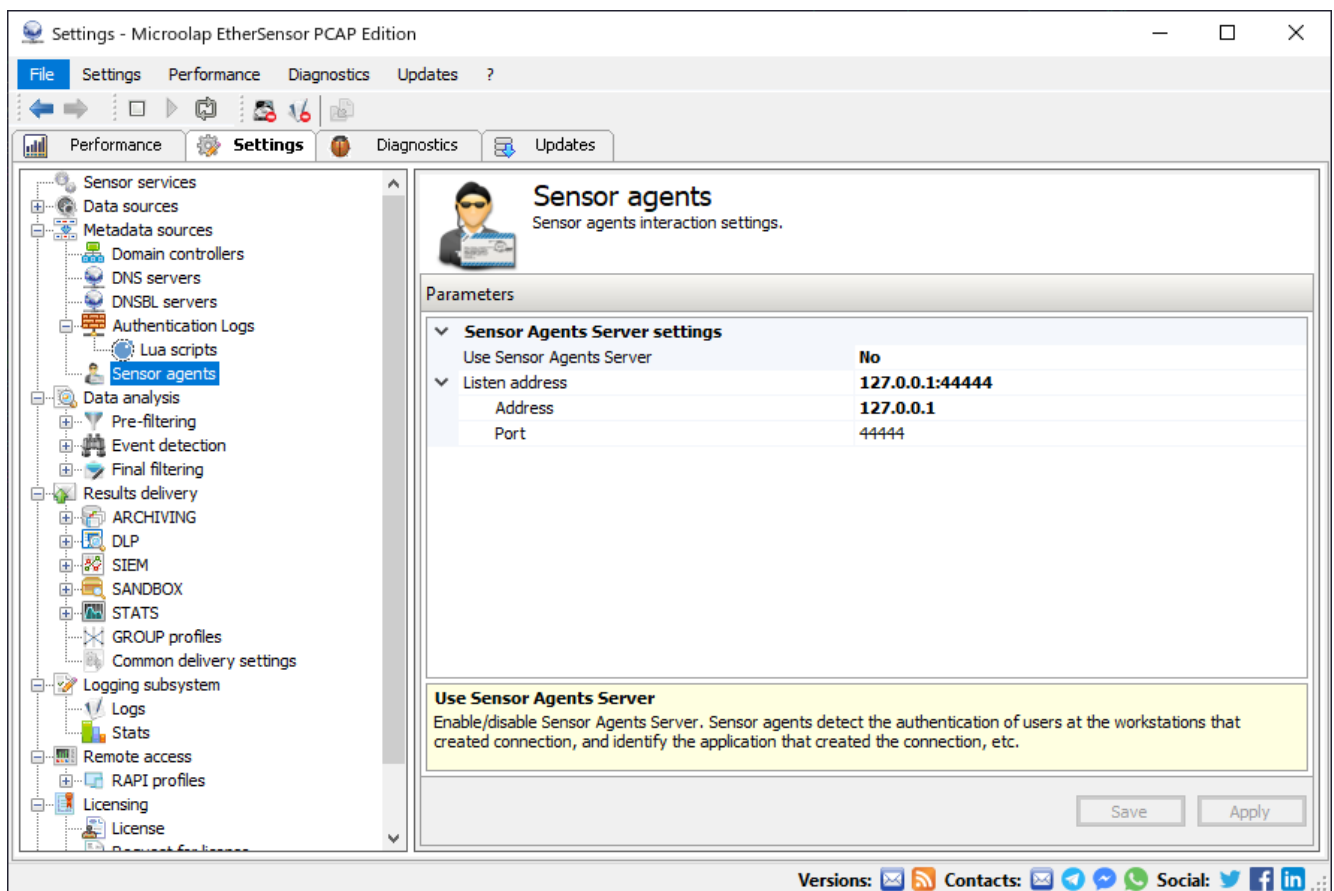


Fig.24. Setting up Agents server for the EtherSensor Identity service.

Use EtherSensor Agents Server

Enable/disable EtherSensor Agents Server. EtherSensor agents detect the authentication of users at the workstations that created connection, and identify the application that created the connection, etc.

Listen address

Use this option to specify the server local address to listen messages from EtherSensor agents.

Address

Use this option to specify the server IP-address to listen messages from EtherSensor agents.

Port

Use this option to specify the server port number to listen messages from EtherSensor agents.

3.6. EtherSensor Agent

EtherSensor Agent - Windows service installed on workstations. EtherSensor Agent solves two tasks:

- The Agent sends to the EtherSensor server via UDP protocol information⁽⁷⁰⁾ about the process that creates external TCP connections. This allows you to solve the problem of binding a TCP session to a specific workstation in the case of users working in a terminal session, behind NAT and other similar cases.
- The agent transmits information⁽⁶⁷⁾ about events on the workstation to the EtherStat server via the TCP protocol to its address and port specified in the configuration. If it is temporarily impossible to send data (no connection, etc.), they are accumulated in the local database. As soon as the Agent detects the ability to send data, it packs 64KB packets and sends them to the EtherStat server.

Agents provide the most complete and accurate information for EtherSensor. Of particular value is the name of the process involved in the connection.

If for any reason you cannot use agents on your workstations/servers, consider using a script for the user's domain or system logon event (logon script⁽⁵⁴⁾).

3.6.1. System requirements for the Agent

EtherSensor Agent operates on workstations that meet the following system requirements:

- OS (Windows 7, Windows 2008, Windows 8, Windows 8.1, Windows 2012, Windows 10) 32/64 bit.
- The required free space on your hard drive is at least 10 MB.
- Requirements to work with third-party information security means must be met.

Working with third-party information security means

To ensure stable operation of EtherSensor Agent you should take into account compatibility with third-party information security and other infrastructure components:

- The directory where EtherSensor Agent is installed contains working directories. This path and all its subdirectories should be excluded from the control of tools similar to antivirus, search indexers, as well as tools for controlling file changes. Such software should not block files in this directory and its subdirectories from being created, deleted, moved or modified.

- EtherSensor Agent contains the service sensor_agent.exe, which must be allowed to start and work with the rights of the local system.
- The EtherSensor Agent service requires the ability to exchange information via UDP and TCP protocols with the remote server for normal operation. Information security means must not check, modify or restrict connections to the server to which EtherSensor Agent sends data. Similarly, information security means should not prevent EtherSensor Agent from opening connections on the ports used to transfer information to the EtherSensor server.
- EtherSensor Agent requires registration of the Layered Service Provider module sensor_lsp.dll for its operation. Third-party security means must not interfere with the registration of sensor_lsp.dll and its operation.
- During installation and operation, the processes of EtherSensor Agent use calls that require high privileges. The OS security policy should allow driver operations, process control and access to network interfaces for EtherSensor Agent.

3.6.2. Agent Installation

Installation of EtherSensor Agent can be done either manually on each workstation or using Active Directory group policies (GPO).

To install the Agent manually, you need to run the MSI installer (32 or 64 bit) included in the distribution package.

To install EtherSensor Agent correctly, you need administrator rights as they are set by default when installing Windows.

Additional restrictions on administrator rights may result in incorrect operation.

During the Agent installation EtherSensor Agent service will be installed (process sensor_agent.exe), and the Layered Service Provider will be registered in the system: the sensor_lsp.dll module.

By default, the Agent is installed in [[INSTALLDIR]] Agent directory.

For correct operation of installed EtherSensor Agent instances it is necessary to configure the DNS server of the organization's network so that the server name specified in the EtherSensor Agent settings corresponds to the IP address of the EtherSensor server.

Installation can also be performed via the Windows Installer msiexec.exe utility in the command line with administrator rights, specifying the following parameters:

INSTALLDIR:

The path to the directory where EtherSensor Agent will be installed.

ETHERSTATSERVER:

The IP address and port of the server where EtherStat is located in address:port format.

KEY:

ZeroMQ protocol key for secure connection to EtherStat server, must be 40 characters long.

ETHERSENSORSERVER:

The IP address and port of the server where EtherSensor is located in the "address: port" format.

An example of a command line with which EtherSensor Agent will be installed EtherSensor:

```
msiexec /i [path to EtherSensor Agent MSI package] INSTALLDIR="[[INSTALLDIR]] Agent"  
ETHERSTATSERVER="etherstat:44445" KEY="0123456789ABCDEFGHIJabcdefgh!@#%^&*<>?-=+^"  
ETHERSENSORSERVER="ethersens:44444"
```

3.6.3. Agent files

Files included in the installation package of EtherSensor Agent:

[INSTALLDIR]\config\agent.xml

Text configuration file in XML format for assigning connection parameters to EtherStat and EtherSensor servers, data polling period on the workstation, and application filter for which TCP connection data should not be sent to the EtherSensor server.

[INSTALLDIR]\syslog.dll

The library needed to create and maintain all *.log files of EtherSensor Agent.

[INSTALLDIR]\sensor_agent.exe

Executable file EtherSensor Agent, implements the main functions of the Agent. When launching the service, you should specify one of the following command line parameters:

/service

Starting the Agent in Windows service mode.

/process

Starting the Agent in Windows process mode.

/install

Installing Agent service in the OS.

/remove

Remove Agent service from the OS.

If none of the parameters was specified, [INSTALLDIR]\log\svcagent.log will make a record about the incorrect command line with the corresponding hint, and the service will stop working.

Files generated during the Agent's work:

[INSTALLDIR]\log\svcagent.log

Text file in XML format, in which main actions and errors are logged EtherSensor Agent.

[INSTALLDIR]\log\sensor_agent.exe.log

Text file for recording information about events generated inside the EtherSensor Agent logging subsystem.

[INSTALLDIR]\log\processinfo.log

Text file containing information about current processes whose connections are tracked by EtherSensor Agent.

[INSTALLDIR]\data.db

Database file of messages about events that could not be sent over a TCP connection to the EtherStat server. When a connection to the server is established, the data will be re-sent, and if the delivery is successful, it will be deleted from the database.

3.6.4. Logic modules of the Agent

The module for tracking connections and marking HTTP connections (sensor_lsp.dll).

This module is designed as a Layered Service Provider. This means that when the module is installed, it is built into the application network stack and transparently proxies (while tracking) all TCP connections created by local processes. The module has settings stored in the Windows registry:

- UDP port of the EtherSensor EtherCAP service. The default port is 44444.
- HTTP traffic marking flag, default value is 1. If the flag is set to state 1, each HTTP request is marked with an X-Sensor-UID-type header: 554E4B4E-4F57-4E20-5555-494400000000, where 554E4B4E-4F57-4E20-5555-494400000000 is a unique user identifier bound to a particular machine, to a particular user on that machine, and is global in the context of the organization's network.

In the course of its work, the sensor_lsp.dll module locally communicates via the UDP protocol with the second main module EtherSensor Agent - the EtherSensor Agent service (process sensor_agent.exe), - and passes it information about the processes creating TCP connections.

EtherSensor server interaction module (service EtherSensor Agent).

This module is executed as a Windows system service in which the following functions are implemented:

- Collect and transmit workstation event data ⁽⁶⁷⁾ over an encrypted TCP connection to the EtherStat monitoring and statistics server. If the data could not be sent, the EtherSensor Agent service places the data in the local database [INSTALLDIR]\data.db.
- Transfer of data about TCP-connections of workstation processes ⁽⁷⁰⁾ via the UDP protocol to the EtherSensor server. The settings allow you to take into account the list of processes whose connections do not need to be monitored.
- Logging Agent service events and writing them to [INSTALLDIR]\log\svcagent.log.

The module sensor_agent.exe takes configuration data from [INSTALLDIR]\config\agent.xml. When changing settings, the EtherSensor Agent service must be restarted.

3.6.5. Data transferred to EtherStat

The agent sends data to the EtherStat server in two cases:

1. Periodically, according to the configuration.
2. When an event occurs, data about which EtherSensor Agent should be transmitted according to the configuration.

Periodically sent data

Hardware list

Sent to EtherStat only when starting the EtherSensor Agent service, then only the changes are sent (by change event). The list contains data structures describing the devices installed on the workstation. The data about the devices are extracted from their Properties provided by Windows Device Manager using the corresponding WinAPI functions:

- Device name
- Device description
- Manufacturer's name
- Corresponding Device Id
- GUID Class of the device in the {xxxxxxxx-xxxxx-xxxxx-xxxxx-xxxxxxxx} format.
- List of Hardware Ids of the device
- The name of the service the device communicates with.

List of installed applications and services

Sent to EtherStat only at startup of the EtherSensor Agent service, then only change data are sent. The list contains data structures describing the software installed on the workstation. All data about the installed software are extracted from the Windows registry using WinAPI and look like that:

- Product name
- Manufacturer's name
- Current product version

- Product installation path.

Registry branches from which information is taken:

- HKML\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
- HKML\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall
- HKEY_USERS\

OS information

Sent to the EtherStat server according to the OSMonitor configuration tag. This is OS Windows data on the workstation extracted using WinAPI:

- Name, current version, type and status of the operating system
- Serial number in XXXXX-XXXXXXX-XXXXXXX format
- Type of architecture (x86 or x64)
- Computer name in system and domain
- Physical drive partition and operating system directory
- Date and time of the last system reboot and date of the last system update (if there were no updates, the OS installation date). The current time on the workstation is also sent.
- Model and name of the motherboard manufacturer
- Number of running physical and logical processes
- The size of the operating system paging file.

Information on network adapters

Sent to the EtherStat server according to the NETMonitor configuration tag. A separate message is created for each adapter with its description and settings. Data on the network adapter and its configuration are obtained using WinAPI functions and contain:

- Name of the adapter
- Manufacturer's name
- MAC address, IP address, subnet mask, DNS address settings, etc.
- Name of the computer in the network and domain
- DHCP usage flag
- Network adapter GUID
- The maximum data transfer rate of the network adapter in bits per second.

Data on the current computer load

Extracted using WinAPI and contain:

- CPU current load in percent
- Current RAM usage in percent
- Current HDD Filling in percent

- Free space on HDD.

Data sent by event

Changing the list of installed software

Sent to the EtherStat server when new applications are discovered or when existing applications are removed from the workstation. All data about installed or removed software is extracted from the Windows registry using WinAPI and contain:

- Product name
- Manufacturer's name
- Current product version
- Product installation path.

Change the list of installed hardware

Sent to the EtherStat server when a new or remote device is detected on a workstation. All device data is extracted from their Properties, obtained from Windows Device Manager using the appropriate WinAPI features and contain:

- Device name
- Device description
- Manufacturer's name
- Corresponding Device Id
- GUID Class of the device in the {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx} format.
- List of Hardware Ids of the device
- The name of the service the device communicates with.

Start and stop the process with binding to the TCP session

Sent to the EtherStat server when a new process is detected or stopped. When a new process is detected, a data structure describing the process and containing the process is sent:

- Process name
- Command line with process start arguments
- The path to the directory of the running process
- User usage time
- ProcessID, SessionID and ParentID identifiers.

When the process is stopped, the ProcessIDs and SessionIDs of the session generated by the process are sent.

Data sent to the server when a user performs actions in the system

When a user logs on:

- Name of the domain to which the user belongs

- Name of the user account
- SID - unique user identifier in the OS
- SessionID - session number on the computer
- Name of the way the user works with the workstation: console or rdp
- Date and time of user login.

When a user logs out:

- SID - unique user identifier in the OS
- SessionID - session number on the computer
- Date and time of user logout.

When blocking or unblocking a user account:

- SID - unique user identifier in the OS
- SessionID - session number on the computer.

When changing the active window

- Current window title
- An identifier of the process that owns this window.

3.6.6. Data transferred to EtherSensor

EtherSensor Agent delivers the obtained information to the EtherSensor server over UDP.

The agent sends to the EtherSensor server information about the process that creates external TCP connections:

- Process Identifier
- Process name
- Name of the user under which the process is executed
- Computer name
- User ID.

The agent also sends information about the TCP connection itself:

- Process Identifier
- User ID
- Connection details.

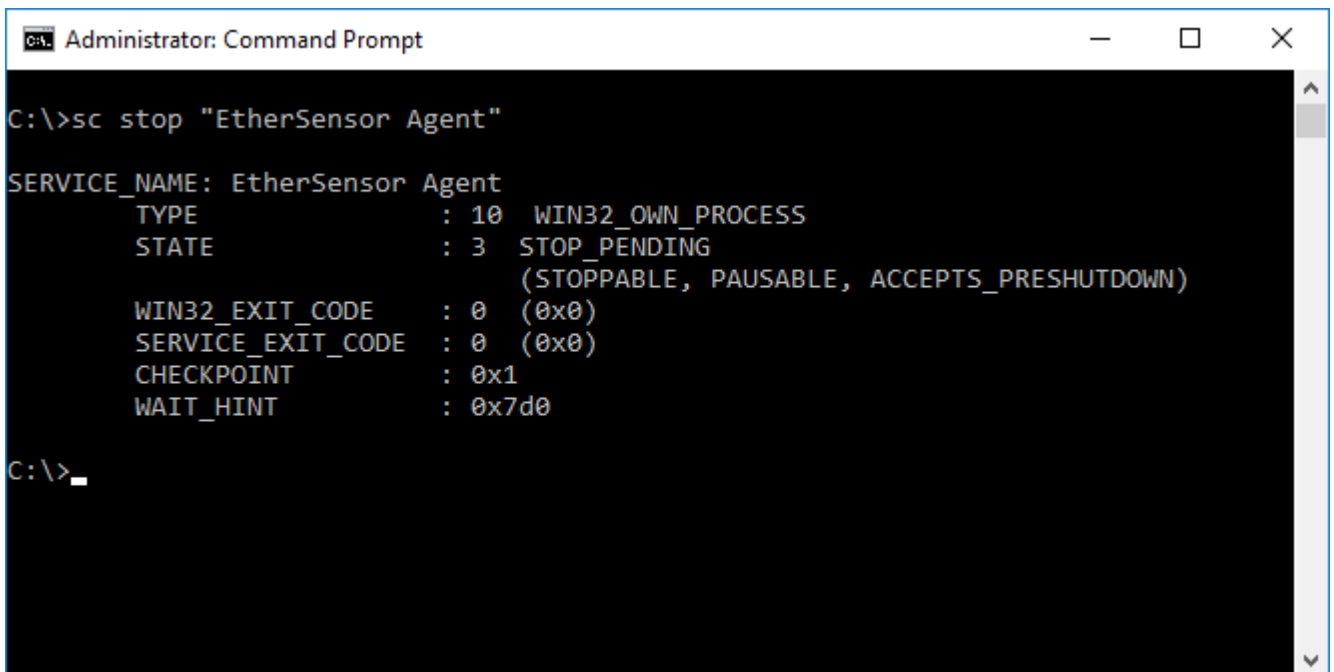
Based on the data transferred by the Agent, the server EtherSensor will mark the reconstructed traffic objects depending on the settings with the following properties:

- Name of the user under which the process is executed
- Computer name
- User ID in format X-Sensor-UID: 554E4B4E-4F57-4E20-5555-4944000000.

3.6.7. Working with the Agent

Before starting EtherSensor Agent do the following:

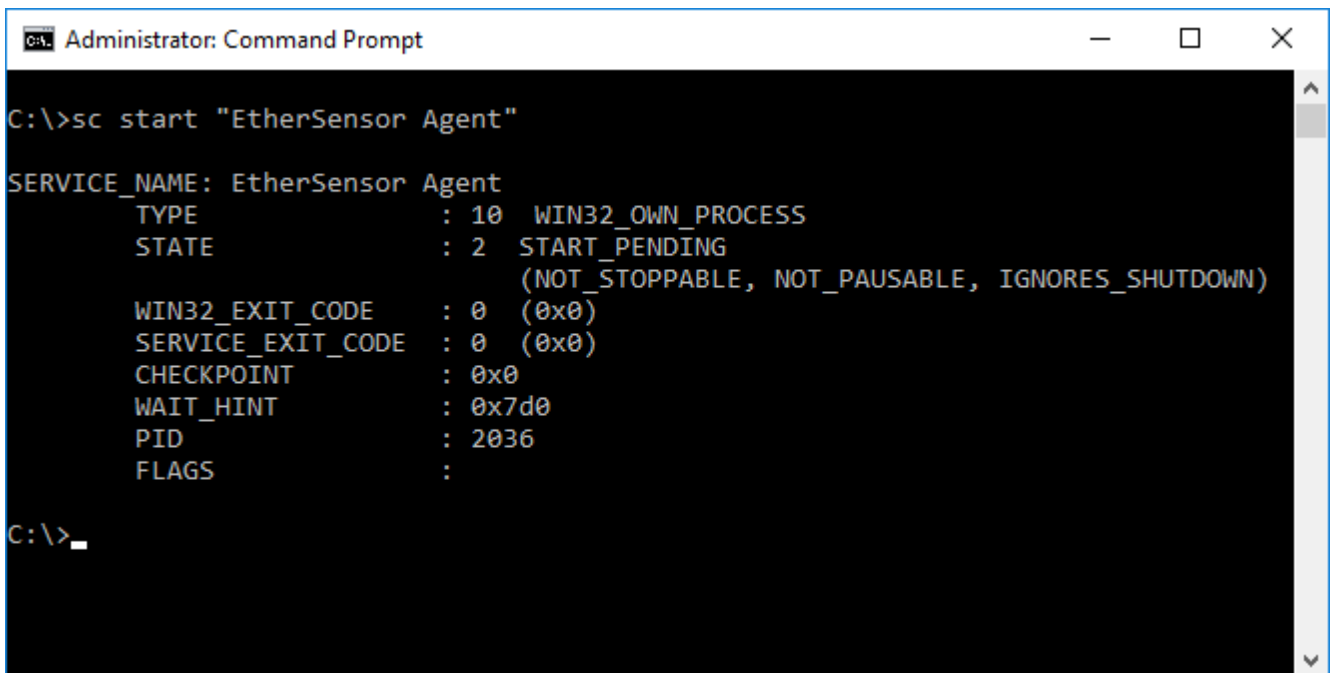
1. Make the necessary configuration settings in the file [INSTALLDIR]\config\agent.xml using any text editor.
2. Run cmd.exe with administrator rights.
3. Stop the EtherSensor Agent service with the command `sc stop "EtherSensor Agent"`.



```
Administrator: Command Prompt
C:\>sc stop "EtherSensor Agent"
SERVICE_NAME: EtherSensor Agent
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE                : 3   STOP_PENDING
                        (STOPPABLE, PAUSABLE, ACCEPTS_PRESHUTDOWN)
        WIN32_EXIT_CODE      : 0   (0x0)
        SERVICE_EXIT_CODE   : 0   (0x0)
        CHECKPOINT          : 0x1
        WAIT_HINT           : 0x7d0
C:\>_
```

Fig.25. Stop the "EtherSensor Agent" service.

4. Start the service EtherSensor Agent with the command `sc start "EtherSensor Agent"`.



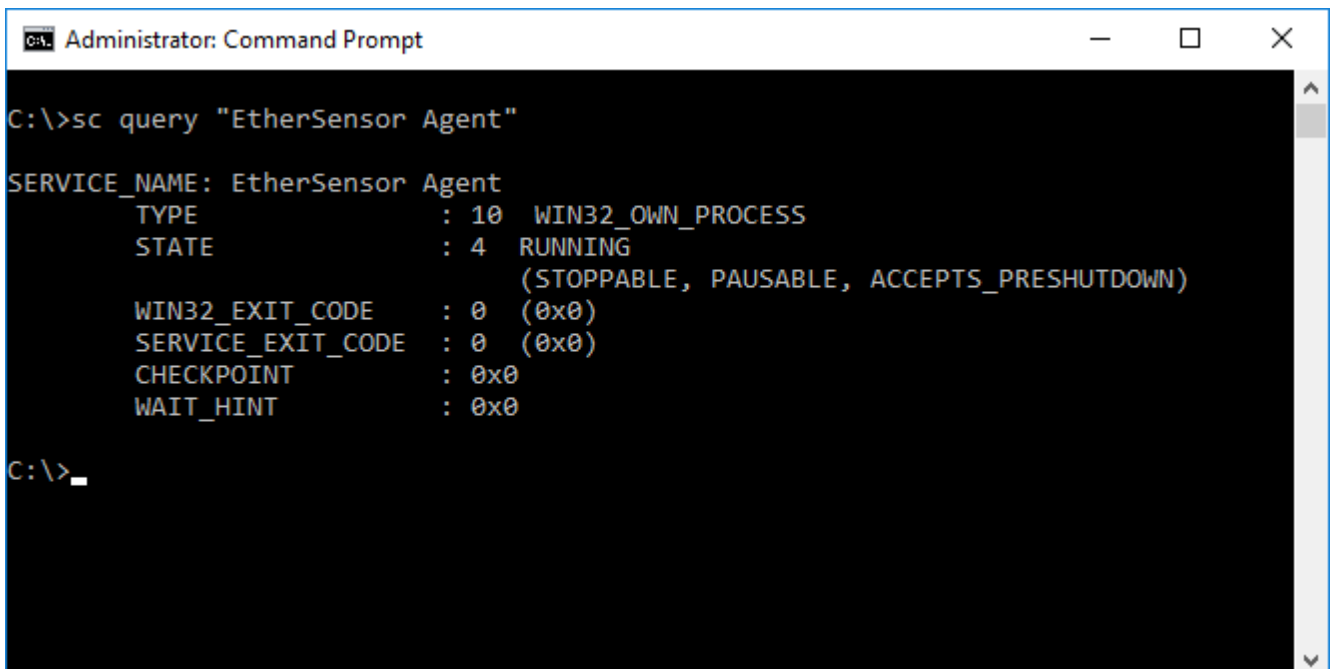
```
Administrator: Command Prompt
C:\>sc start "EtherSensor Agent"

SERVICE_NAME: EtherSensor Agent
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE                : 2   START_PENDING
                        (NOT_STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE      : 0   (0x0)
        SERVICE_EXIT_CODE   : 0   (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x7d0
        PID                 : 2036
        FLAGS                :

C:\>
```

Fig.26. Starting the "EtherSensor Agent" service.

5. Make sure that the service is started by typing `sc query "EtherSensor Agent"`. The state of the service must be `RUNNING`.



```
Administrator: Command Prompt
C:\>sc query "EtherSensor Agent"

SERVICE_NAME: EtherSensor Agent
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE                : 4   RUNNING
                        (STOPPABLE, PAUSABLE, ACCEPTS_PRESHUTDOWN)
        WIN32_EXIT_CODE      : 0   (0x0)
        SERVICE_EXIT_CODE   : 0   (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x0

C:\>
```

Fig.27. Checking the service "EtherSensor Agent".

3.6.7.1. Possible options for the Agent's work

Agent working with EtherStat server

For the Agent to work with the EtherStat server, it is necessary that the workstations on which EtherSensor Agent is installed are in the same network as the EtherStat server. EtherStat uses an encrypted TCP connection to collect and analyze information from the agents installed on the workstations. To identify the workstations, the agent generates a unique UHID and sends it in messages to the EtherStat server.

Agent work with EtherSensor server

Transparent proxy mode without traffic marking

In this mode, the Agent transparently proxies the connections of applications running on the user's computer. If the application successfully establishes a connection, the Agent sends to the EtherSensor server information about the establishment of a TCP connection by a specific application running under a specific network user.

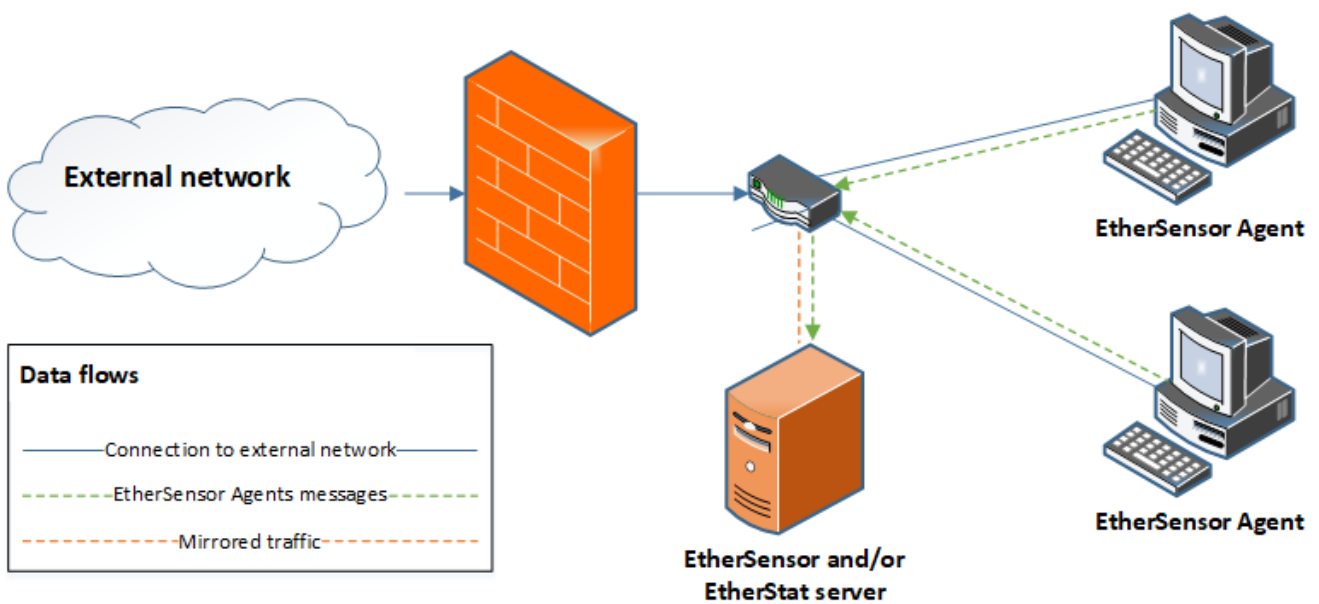


Fig.28. Transparent proxying without traffic marking

Thus EtherSensor during message reconstruction can fully identify the user who sent this message through any currently supported protocol (ICQ, MSN, MRA, IRC, XMPP, SMTP, POP3, LOTUS, HTTP, FTP, etc.).

In this case, the condition for diverting a copy of the analyzed traffic must be met: a copy of the traffic must be diverted to EtherSensor before changes in connection parameters are made.

For example:

- Before the proxy server

- Before NAT
- Before firewall.

Transparent proxying mode with HTTP traffic marking

This mode of the Agent's operation differs from the mode without marking traffic only in that the Agent modifies HTTP requests sent by applications on the client workstation by adding the X-Sensor-UID: <GUID> header to them, where <GUID> is a unique user identifier for a specific computer inside the local network. These actions are performed in full compliance with the HTTP protocol without disrupting its operation.

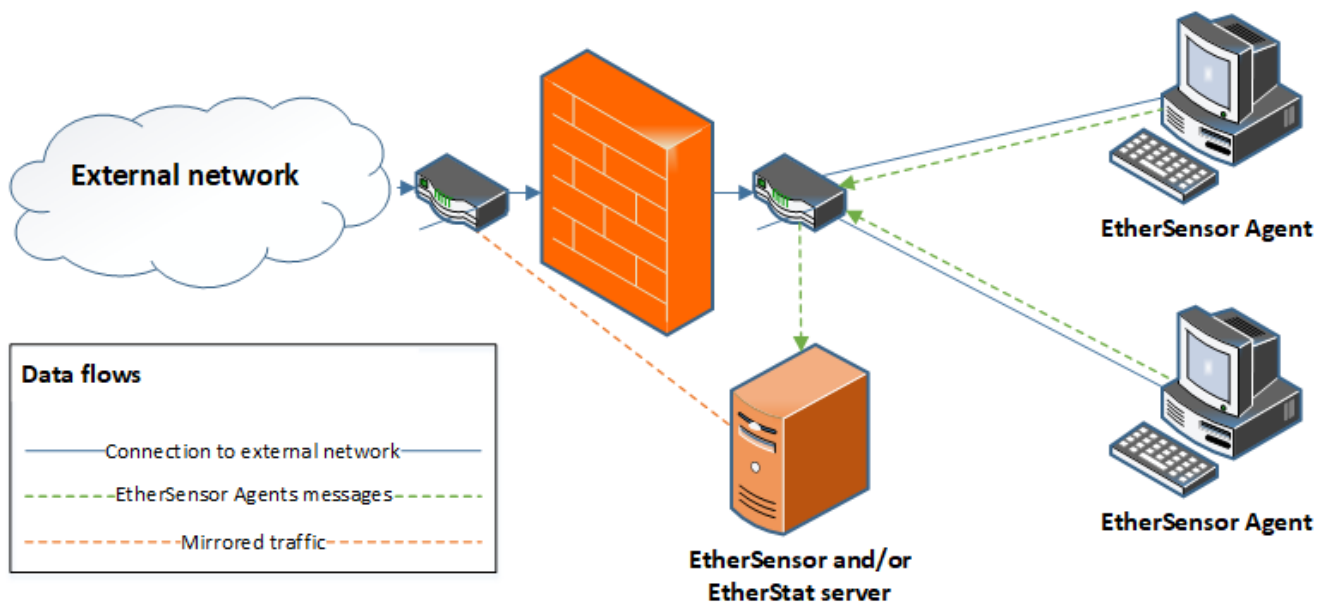


Fig.29. Proxying with traffic marking

This mode of operation can be used when EtherSensor can receive a copy of traffic for analysis only after modification of connection parameters. For example, after connections have passed through a proxy server, NAT or firewall.

In this case, EtherSensor when reconstructing the message, fully identifies the user who sent this message by the X-Sensor-UID headers from the HTTP protocol.

3.6.7.2. Configuration of the EtherSensor Agent service

EtherSensor Agent is configured by making changes to the configuration file [INSTALLDIR] \config\agent.xml.

An example of such a file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config version="1.1">
  <Local port="44444" markhttp="true" />

  <EtherSensor protocol="2">
    <server address="ethersensor.server1:44444" transport="udp" />
  </EtherSensor>
  <Filter>
    <Excludes>
      <application name="ethersensor_agent.exe" />
      <application name="mstsc.exe" />
      <application name="wmplayer.exe" />
      <application name="uTorrent.exe" />
      <application name="skype.exe" />
      <application name="wmpnetwk.exe" />
      <application name="winlogon.exe" />
      <application name="svchost.exe" />
      <application name="spoolsv.exe" />
      <application name="nissrv.exe" />
    </Excludes>
  </Filter>

  <EtherStat address="127.0.0.1:44445" ZMQKEY="" />
  <DataCollectionSetup>
    <Hardware duration_ms="10000" />
    <Software duration_ms="30000" />
    <OperatingSystem duration_ms="60000" />
    <Processes duration_ms="1000" />
    <Performance duration_ms="60000" />
    <Network duration_ms="60000" />
    <UserMonitor duration_ms="1000" />
    <DatabaseStore size="1" />
  </DataCollectionSetup>
</Config>
```

Tag Config

Configuration root tag. The "version" attribute inside the Config tag defines the version of the Agent configuration file format.

Tag Local

The Local tag is nested within the Config tag and defines the settings for the connection tracking module (sensor_lsp.dll). After loading the configuration, the EtherSensor Agent service saves these settings in the Windows registry.

The "port" attribute defines the local UDP port for the sensor_lsp.dll module to communicate with the EtherSensor Agent

The "markhttp" attribute allows/denies marking HTTP traffic by the module sensor_lsp.dll.

Tag EtherSensor

The EtherSensor tag is nested in the Config tag and defines the list of addresses of EtherSensor servers, to which the Agent sends information via UDP about the processes that create TCP connections.

The "protocol" attribute specifies the maximum protocol version used by EtherSensor Agent to send messages to server EtherSensor. At present, protocol version 3 is available. To support this

version of the protocol, use EtherSensor version 4.3.3 or later. For compatibility with previous versions of EtherSensor, set this field to 2.

Tag Server

The "server" tag is nested within the EtherSensor tag and defines the address and transport protocol of the EtherSensor server.

The "address" attribute defines the address and port for communication with the EtherSensor server. Possible address options are IP: Port or DNSNAME: Port.

The "transport" attribute defines the type of transport protocol for communication with the EtherSensor server. Possible options are udp.

Tag Filter

The Filter tag is nested within the Config tag and defines the filter settings for messages sent to the EtherSensor server.

Tag Excludes

The Excludes tag is nested within the Filter tag and defines a list of applications whose TCP connection information should not be sent to the EtherSensor server.

Tag Application

The application tag is nested within the Excludes tag and defines an application whose TCP connection information will not be sent to the EtherSensor server.

The "name" attribute specifies the exact name of the process that you want to exclude from tracking.

Thus, EtherSensor Agent informs the server EtherSensor only about TCP connections that are created to communicate with other workstations and servers on the local network and the Internet, and the settings allow taking into account the list of processes whose connections do not need to be monitored.

Tag EtherStat

Ter EtherStat is nested in the Config tag and defines the settings for connecting to the EtherStat monitoring and statistics system.

The "address" attribute defines the server address in the format "IP address:port".

The "ZMQKEY" attribute must contain the key to work with EtherStat in encrypted connection mode.

Tag DataCollectionSetup

The DataCollectionSetup tag is nested within the Config tag and defines the settings for data polling timers for the EtherSensor Agent service.

Tag Hardware

The Hardware tag is nested in the DataCollectionSetup tag and defines using the "duration_ms" attribute the timer settings in seconds for polling the current hardware.

Tag Software

The Software tag is nested within the DataCollectionSetup tag and defines using the "duration_ms" attribute to set the timer in seconds to poll the currently installed software.

Tag OperatingSystem

The OperatingSystem tag is nested within the DataCollectionSetup tag and uses the "duration_ms" attribute to set the timer in seconds to poll for operating system information.

Tag Processes

The Processes tag is nested in the DataCollectionSetup tag and defines using the "duration_ms" attribute to set the timer in seconds for monitoring current processes.

Tag Network

The Network tag is nested in the DataCollectionSetup tag and uses the "duration_ms" attribute to define the timer settings in seconds for polling data about network adapters and their settings.

Tag Performance

The Performance tag is nested within the DataCollectionSetup tag and defines, using the "duration_ms" attribute, the timer settings in seconds for polling OS performance data.

Tag Users

The Users tag is nested within the DataCollectionSetup tag and defines, using the "duration_ms" attribute, the timer settings in seconds for monitoring user actions.

Tag DatabaseSetup

The DatabaseSetup tag is nested within the DataCollectionSetup tag and defines the settings for the database size limit as a percentage of free space on the HDD on which EtherSensor Agent is installed.

3.6.7.3. Agent work logging

EtherSensor Agent logs its actions into files in the directory [INSTALLDIR]\log:

- The svcagent.log file records information about the basic actions performed by the EtherSensor Agent service.
- The file sensor_agent.exe.log contains information about events generated inside the logging service of EtherSensor Agent.
- The file processinfo.log contains information about current processes whose connections are monitored by EtherSensor Agent.

Log files (svcagent.log and sensor_agent.exe.log) are XML files of the following content:

```
<Message time="2012-03-23T17:47:48.148+04:00" level="information">
  <Client channelname="MICROOLAPAGENT"
    processname="sensor_agent.exe"
    modulename="sensor_agent.exe" />
  <Text>Start of the application.</Text>
</Message>
```

Tag Message

The Message tag is the root tag of the message written to the log file. The "time" attribute is the time when the message was sent, the "level" attribute is the priority with which the message was sent (for example, information - informational message, error - error message).

Tag Client

The Client tag describes the sender of the message. The "channelname" attribute contains the name of the message channel, the "processname" attribute - the name of the sender's process, the "modulename" attribute - the name of the module within the process that created the message.

Tag Text

The Text tag contains the text of the message.

The processinfo.log file is an XML file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<Processes>

  <Process pid="4136" name="chrome.exe">
    <User uuid="32014294-5bbf-11e1-b8f5-005056c00808"
      name="Home-PC\Home"/>
    <Sessions local="0" remote="311"/>
  </Process>

  <Process pid="636" name="svchost.exe">
    <User uuid="3a45de5b-5be6-11e1-b8f5-005056c00808"
      name="HOME\HOME-PC$"/>
    <Sessions local="0" remote="3"/>
  </Process>

  <Process pid="948" name="firefox.exe">
    <User uuid="32014294-5bbf-11e1-b8f5-005056c00808"
      name="Home-PC\Home"/>
    <Sessions local="2" remote="741"/>
  </Process>

  <Process pid="1584" name="googletalk.exe">
    <User uuid="32014294-5bbf-11e1-b8f5-005056c00808"
      name="Home-PC\Home"/>
    <Sessions local="0" remote="89"/>
  </Process>

  <Process pid="2860" name="uTorrent.exe">
    <User uuid="32014294-5bbf-11e1-b8f5-005056c00808"
      name="Home-PC\Home"/>
    <Sessions local="100" remote="27084"/>
  </Process>

  <Process pid="3076" name="vmware.exe">
    <User uuid="32014294-5bbf-11e1-b8f5-005056c00808"
      name="Home-PC\Home"/>
    <Sessions local="4" remote="1"/>
  </Process>

  <Process pid="3908" name="NisSrv.exe">
    <User uuid="fb91c5e7-5eec-11e1-b226-005056c00808"
      name="NT AUTHORITY\LOCAL SERVICE"/>
    <Sessions local="0" remote="1"/>
  </Process>
</Processes>
```

Tag Processes

The Processes tag is the root tag of the list of processes to be tracked.

Tag Process

The Process tag is nested within the Processes tag and describes the process being monitored. The "pid" attribute contains the process ID in the OS, the "name" attribute contains the name of the monitored process.

Tag User

The User tag is nested within the Process tag and describes the user on the local system under whose rights the monitored process is running. The "uuid" attribute contains the user ID, the "name" attribute contains the username.

Tag Sessions

The Sessions tag is nested within the Process tag and describes the monitored process connections. The "local" attribute contains the number of local connections made within a process or between processes, the "remote" attribute contains the number of remote connections established by this process.

3.6.7.4. Problems and solutions

The EtherSensor Agent service does not start.

- Check in the configuration EtherSensor Agent the port setting for the servers EtherSensor, EtherStat, as well as the local port setting: Local, EtherSensor and EtherStat tags. The ports must not match. If the ports match, the service cannot work correctly.
- Check the log files (svcagent.log, sensor_agent.exe. Log) for error messages EtherSensor Agent.
- Check Windows logs for EtherSensor Agent error messages.
- Report incidents to support.

After starting of EtherSensor Agent, the processinfo.log file does not display any monitored processes.

- No network card connected or no TCP/IP stack configured. Make sure the system makes TCP connections, for example by loading a browser and opening any page. After these steps, information about the browser process should be displayed in the processinfo.log file.
- This computer is a member of a domain. In this case, the EtherSensor Agent modules loaded into processes that create TCP connections try to get the domain username that the process is running under. In this case, it is very important that the DNS settings of the monitored OS are correctly set, since the system API, which provides information about the username in the domain, uses the DNS settings.

No application on the system can create a remote connection, but there were no such problems before installing EtherSensor Agent on the system.

- Go to the installation directory EtherSensor Agent and run the command `sensor_instlsp.exe -p> log.txt`. This command will write the list of installed OS network stack providers to the log.txt file.
- Analyze the log.txt file yourself and send it to support if necessary.
- Run the command `sensor_instlsp.exe -f -c b`. This command will disable the tracking module EtherSensor Agent `sensor_lsp.dll`. Start a fresh copy of the browser and open the remote page. If the page is opened, then the problem is really in the tracking module `sensor_lsp.dll`. Otherwise, the issue is not related to the tracking module EtherSensor Agent.

EtherSensor does not identify the user of the captured message.

- Check the EtherSensor Agent configuration: EtherSensor tag, then server tag. The "address" attribute of the server tag must contain the correct DNS name of the EtherSensor server, which is correctly defined on this workstation. If an IP address is specified, check that the EtherSensor server IP address is available (for example, using the ping utility).

EtherStat server does not receive messages from EtherSensor Agent:

- Check the svcagent.log file for the EtherSensor Agent error messages.
- Check the configuration⁽⁷⁴⁾ of the connection to the EtherStat server: EtherStat tag. The "address" attribute must contain the IP address to connect to. The "key" attribute contains the public key for the encrypted connection. This key must match the public key of the EtherStat server.
- Check the availability of the EtherStat server's IP address (for example, using the ping utility).
- Check the processing time configuration in the OSMonitor, HWMonitor, SWMonitor, UserMonitor, NetMonitor and ProcMonitor tags. If the "timer" attribute is set to 0, the corresponding event messages will not be processed, therefore, they will not be sent to the EtherStat server.

4. Analysis of events and objects

The EtherSensor Analyser service is designed to detect, filter and analyze objects extracted from network traffic.

The service parses application layer protocol objects received from EtherSensor PCAP, EtherSensor EtherCAP, EtherSensor ICAP and EtherSensor LotusTXN with the purpose of detecting objects and events, recognizing their belonging to certain Internet or intranet services.

Starting with version EtherSensor 6.0, an open subsystem for data analysis and event detection based on Lua scripts has been implemented.

Delivery of EtherSensor includes an extensive set of pre-prepared Lua scripts, and with the help of EtherSensor IDE it is possible to independently develop filters, detectors and delivery profiles of results.

General scheme of the EtherSensor Analyser service work:

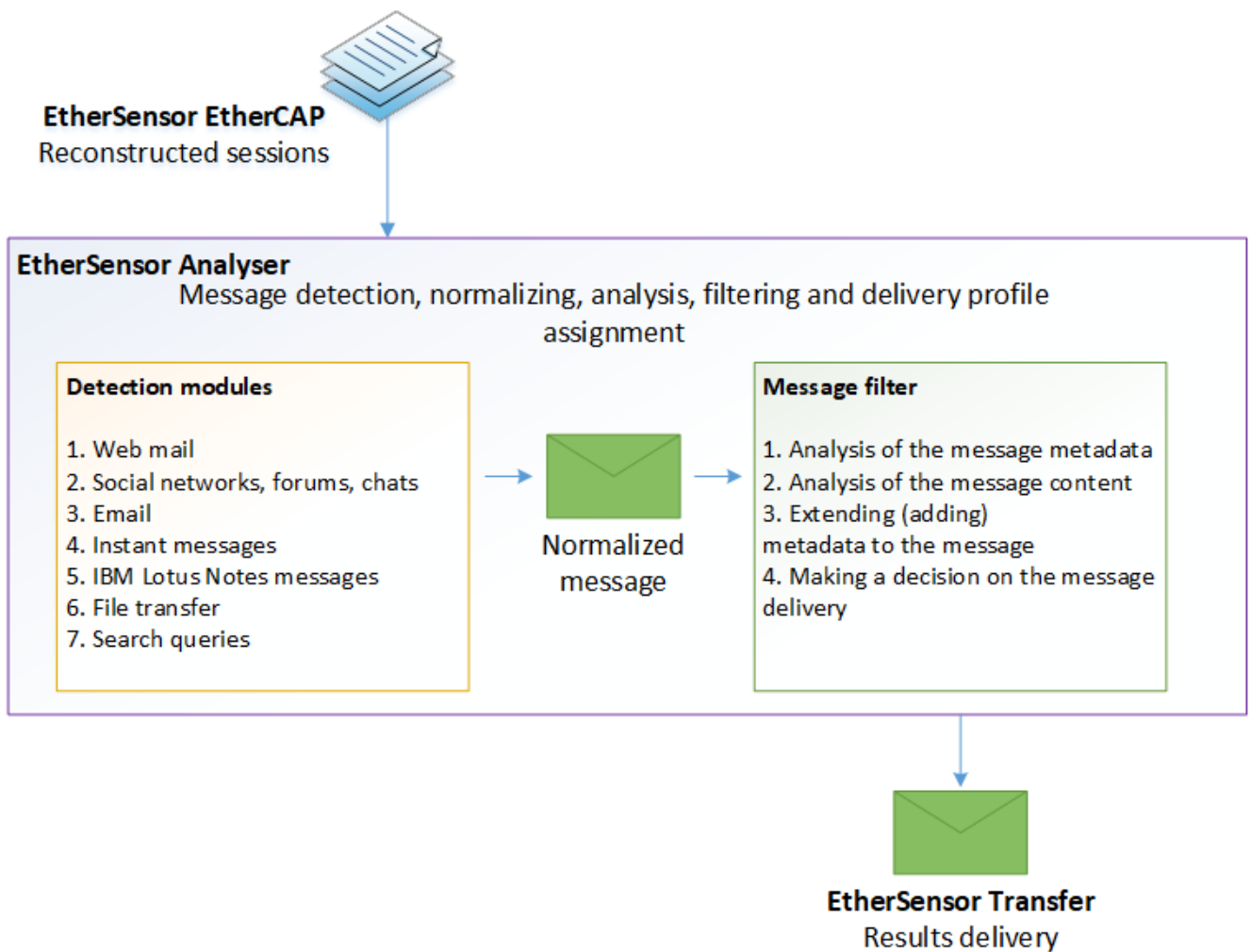


Fig.30. Scheme of the EtherSensor Analyser service work.

EtherSensor Analyser configuration file

The configuration of the service EtherSensor Analyser is stored in an XML file located in the common configuration directory Microolap EtherSensor [INSTALLDIR]\config.

Command line parameters

The Windows service EtherSensor Analyser is installed during the installation of Microolap EtherSensor with automatic launch. However, if necessary, the sensor_analyser.exe process can be run as a Windows application with the following command line parameters:

/process

Run the process sensor_analyser.exe as a normal Windows Win32 process (can be used for debugging).

/service

Run as a Windows service.

/config

Save the default service configuration.

4.1. EtherSensor Analyser settings

The analysis of objects extracted from network traffic performed by the EtherSensor Analyser service consists of the following steps:

1. Pre-filtration of objects/events. ⁽⁸⁵⁾

Pre-filtration of captured objects is mainly used to reduce the load on EtherSensor. At this stage the following object metadata obtained at the stage of traffic capture by EtherSensor EtherCAP service is supposed to be used:

- Network addresses of communication participants
- Protocol types
- Metadata of captured objects of specific protocols
- HTTP objects analysis (request methods, URL, request/response headers)
- Sizes and types of captured RAW data.

Also at the pre-filtering stage EtherSensor can log events and their content in various formats based on SYSLOG: ArcSight CEF, QRadar LEEF, ICSA/CISCO SDEE, DMTF CIM, CEE, etc.

2. Detection and normalization of objects/events ⁽⁸⁷⁾

At this stage, objects/events received from various sources are transformed using Lua scripts into normalized application level events with the same structure and encoding (UTF-8 by default).

The following functions are available when normalizing captured objects:

- URL-encoded objects analysis
- Content analysis of MIME parts
- JSON/XML/HTML objects analysis
- WebSocket object analysis: custom binary, json, xml, xmpp
- Unpack base64, gzip, brotli objects
- Object analysis with hyperscan and pcre-2
- Reconstruction of files transmitted in parts
- Analysis of binary objects, for example:

- DNS over HTTPS
- Protobuf.
- Accumulation and storage of metadata, for example:
 - Create and store user/device profiles for further binding of events
 - Cookie storage
 - Storing User Account Information.

3. Final filtration⁹⁰.

Objects/events received at the stage of detection and normalization can pass the stage of final filtering. At the stage of final filtering, the following parameters of the object/event are checked:

- Network addresses of communication participants
- Domain addresses of communication participants
- Object/event metadata (attributes and headings)
- Email addresses of communication participants (from, to, cc, bcc)
- Instant Message client user identifiers (ICQ, MRA, MSN, IRC, SKYPE, XMPP)
- Social network user IDs
- Contents of text fields of messages (subject, body)
- Names and types of attachments
- Sizes of files and messages.

The objects/events at this processing stage can be enriched with additional metadata/attributes:

- File types detected with libmagic
- Hashes of files and various object/event segments (crc32, md5, sha1, sha256)
- Metadata accumulated by the EtherSensor Identity service (network users, devices, etc.).
- Information about object/event delivery to the system-consumer (DLP, SIEM, UEBA, eDiscovery, Enterprise Search, other archives, etc.).

At each stage of data processing, you can specify an arbitrary set and order of calling Lua scripts to process the captured object/event.

This means that EtherSensor user can fully control the order of actions and processing logic of captured objects.

Each Lua script can:

- Stop processing the current object/event, after which it and its data will be deleted
- Continue processing the object/event. In this case, the object will be passed to the next script for further processing
- Create a new object/event and pass it to the final filtering stage.

4.1.1. Pre-filtration

To manage the pre-filtration settings, use the window *Pre-filtering* of the EtherSensor management console:

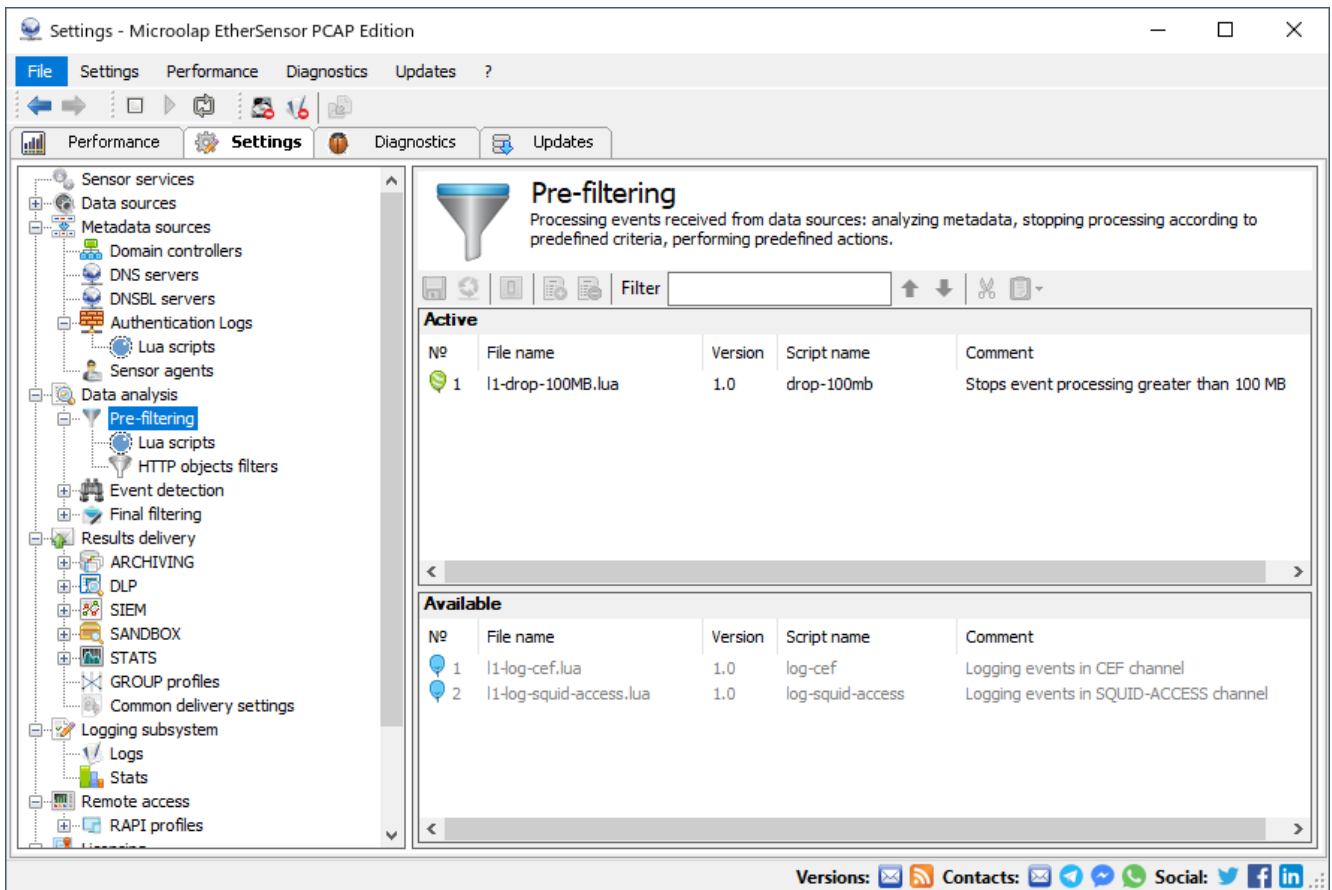


Fig.31. Configure pre-filtering of the captured objects.

In this window, Lua scripts are assigned for pre-filtering objects. Scripts are called from the first to the second, and so on. The number of scripts involved in pre-filtration can be as large as you want.

You can edit scripts using any text editor (scripts are located in the installation directory [INSTALLDIR]/scripts/an-prefilter) or directly in the management console window *Lua scripts*.

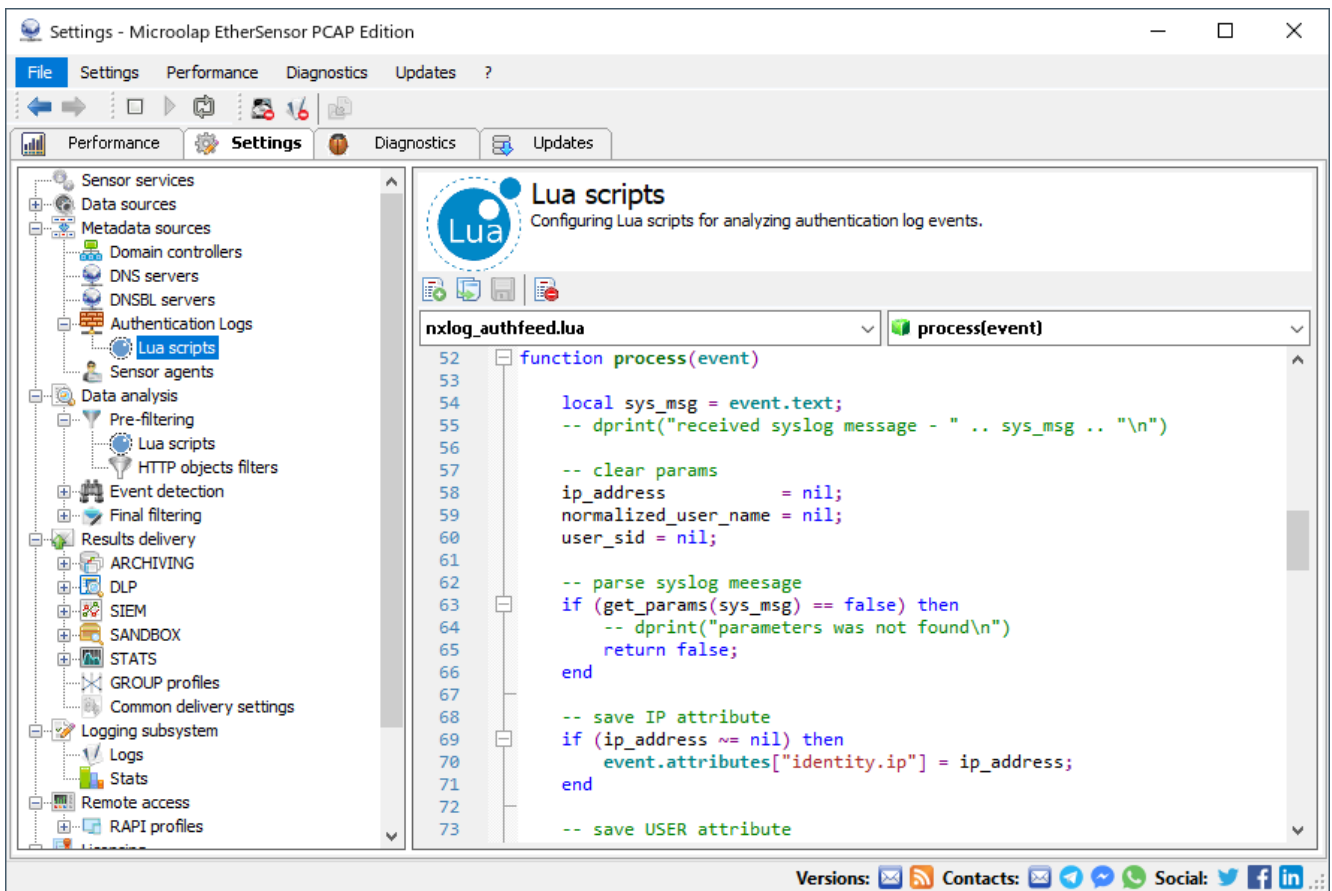


Fig.32. Editing captured objects pre-filtering scripts.

Filtering of captured HTTP requests can also be configured using the **HTTP objects filters** window of the management console. Filtering rules created using the HTTP object filter are stored in XML files located in [INSTALLDIR]\config\filter\http.

When the EtherSensor Analyser service is started, the active filter from XML format is transformed into a Lua script and always becomes the first in the pre-filter script call chain.

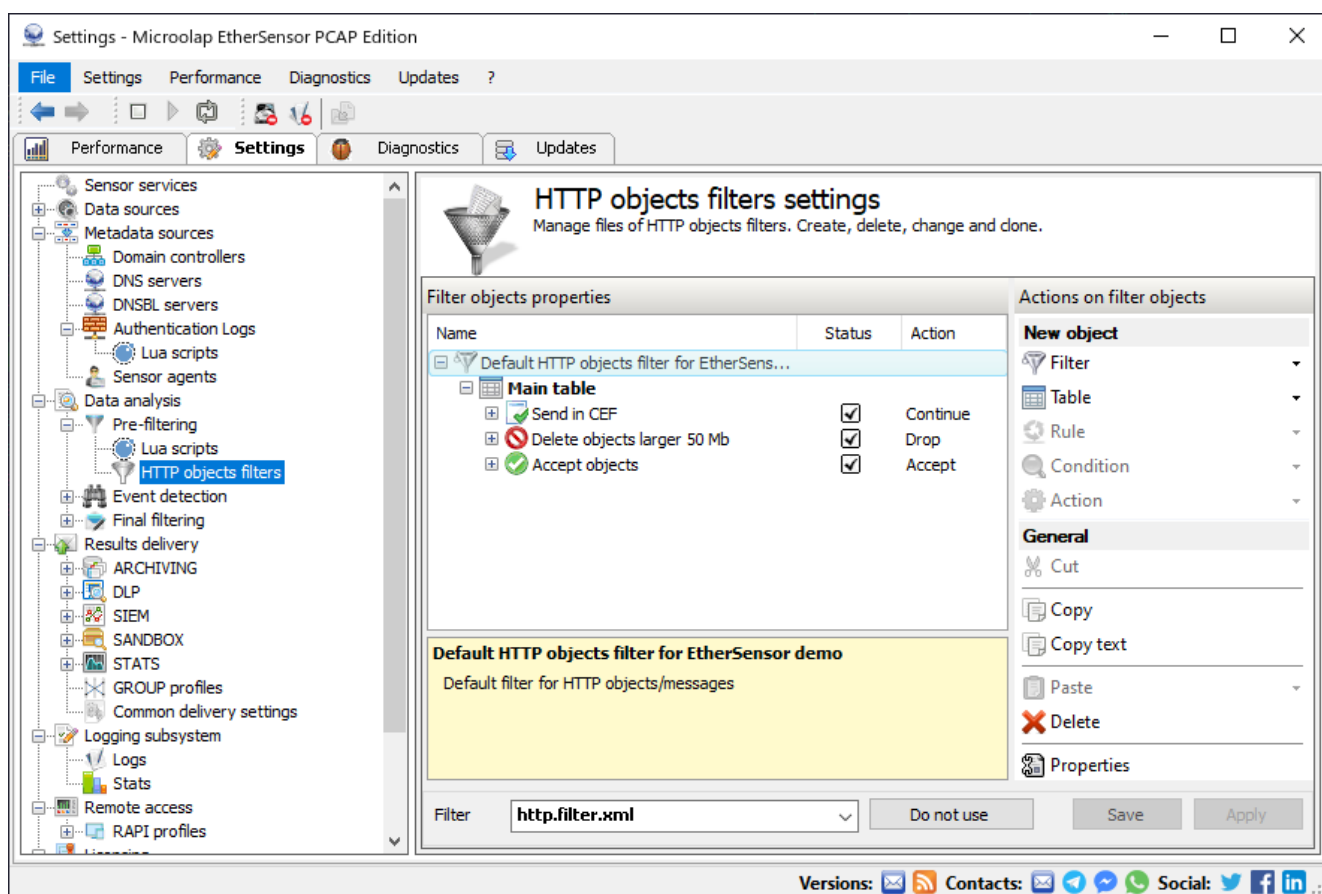


Fig.33. Managing HTTP objects filtering.

Use the right side of the window to edit the filter, you can edit the active filter. But for the service EtherSensor Analyser to start using the modified filter, this filter should be made active and the service should be restarted.

You can learn more about how this feature works in the section HTTP Request Pre-filtration¹⁵⁰.

4.1.2. Detection and normalization of events

To manage the settings of object/event detection and normalization use the window **Event detection** of the management console:

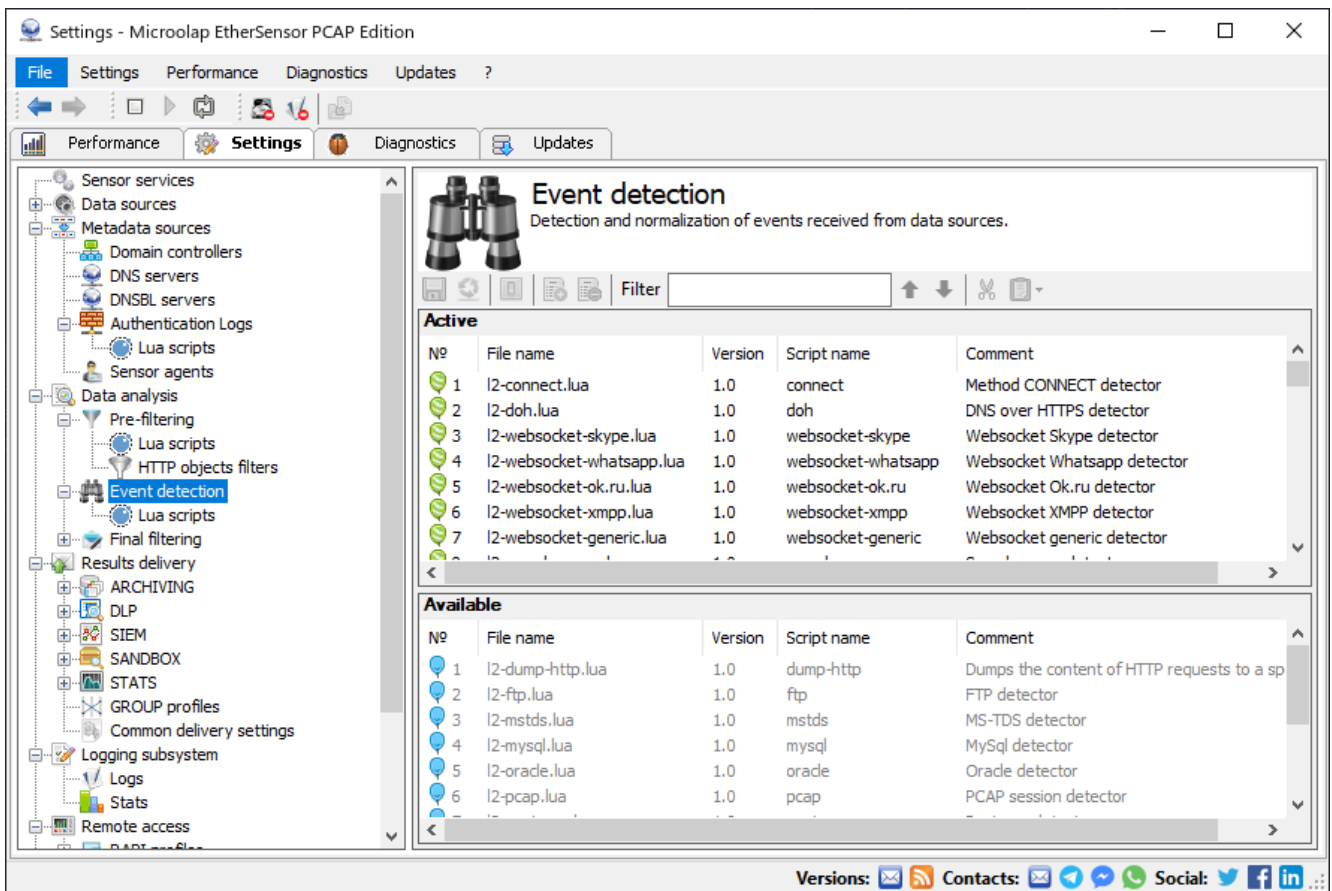


Fig.34. Management of object/event detectors.

In this window, Lua scripts are assigned for detecting and normalizing objects/events. Scripts are called from the first to the second and so on. The number of scripts involved in event detection can be as large as you want.

Scripts can be edited using any text editor (scripts are located in the installation directory [INSTALLDIR]/scripts/an-detect), or directly in the management console window **Lua scripts**.

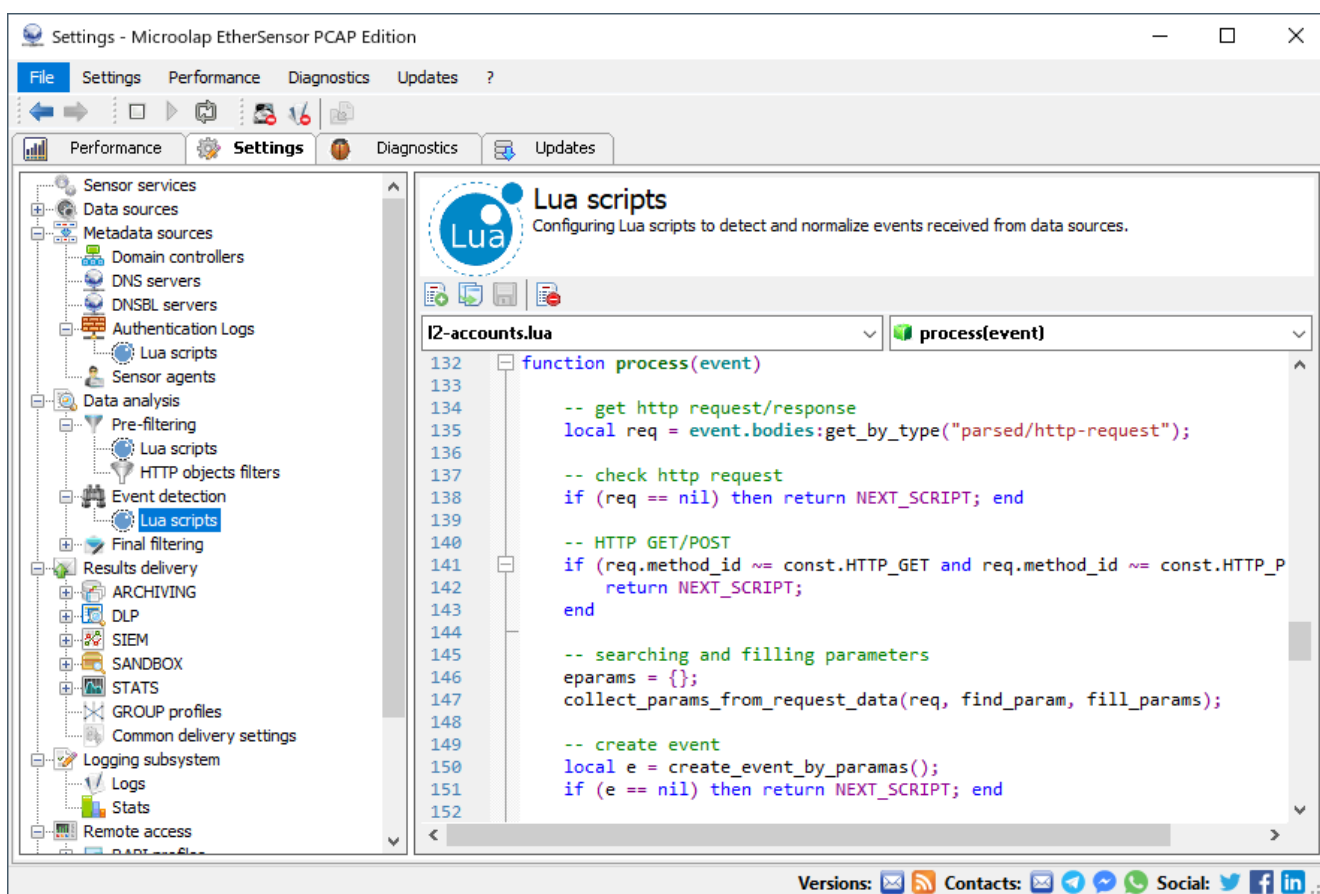


Fig.35. Editing scripts detecting application level events.

4.1.2.1. INCLUDE: Lua scripts functions

- URL-encoded objects analysis
- Content analysis of MIME parts
- JSON/XML/HTML objects analysis
- WebSocket object analysis: custom binary, json, xml, xmpp
- Unpack base64, gzip, brotli objects
- Object analysis with hyperscan and pcre-2
- Reconstruction of files transmitted in parts
- Analysis of binary objects, for example:
 - DNS over HTTPS
 - Protobuf.
- Accumulation and storage of metadata, for example:
 - Create and store user/device profiles for further binding of events
 - Cookie storage

- Storing User Account Information.

4.1.3. Final filtration

To manage the settings of the final filtering of captured objects/events, use the **Final filtering** window of the management console.

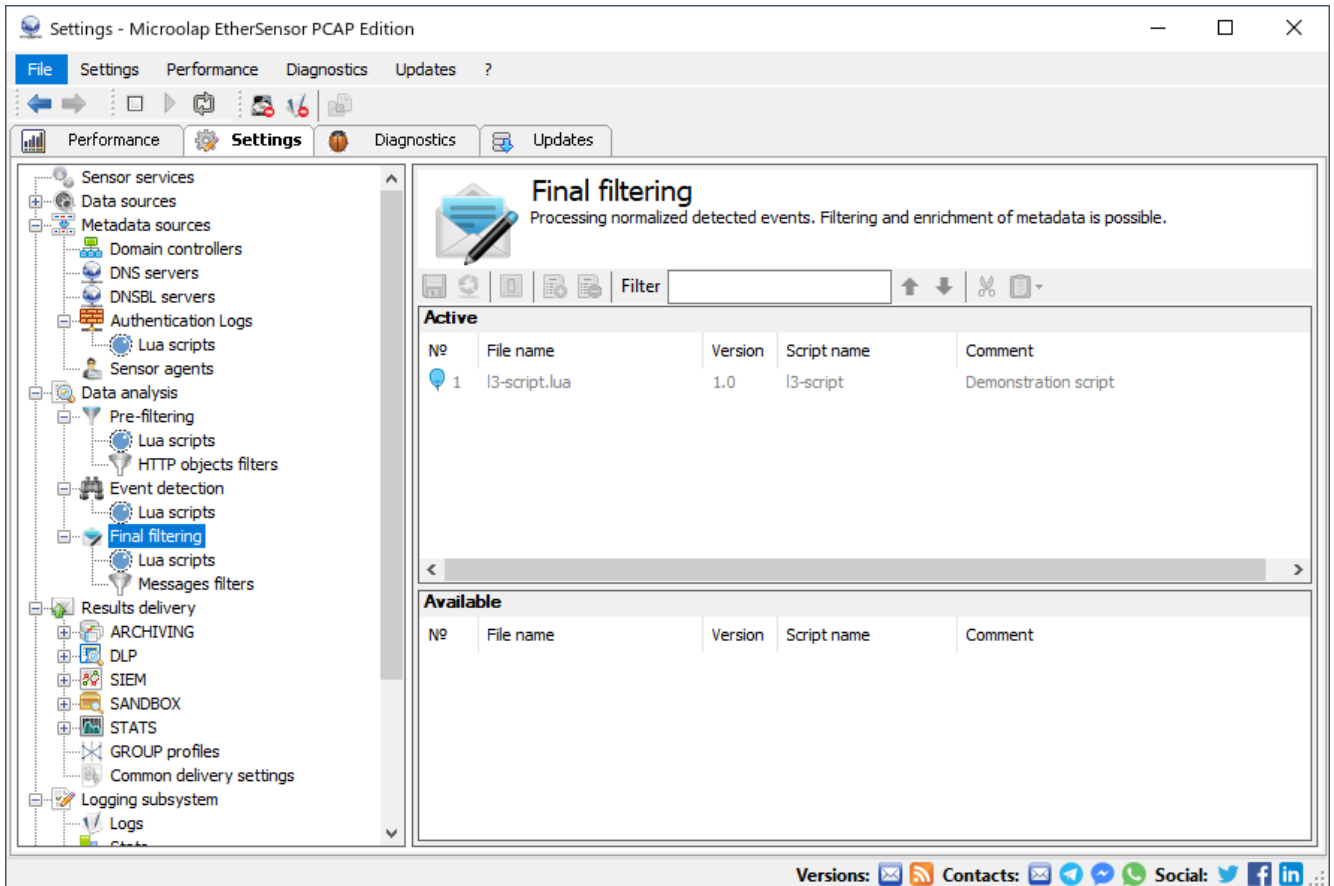


Fig.36. The final filtering of objects/events.

In this window, Lua scripts are assigned for final filtering of normalized objects/events. The scripts are called from the first to the second and so on. The number of scripts involved in the final filtering can be as large as you want.

Scripts can be edited using any text editor (scripts are located in the installation directory [INSTALLDIR]/scripts/an-postfilter) or directly in the management console window **Lua scripts**.

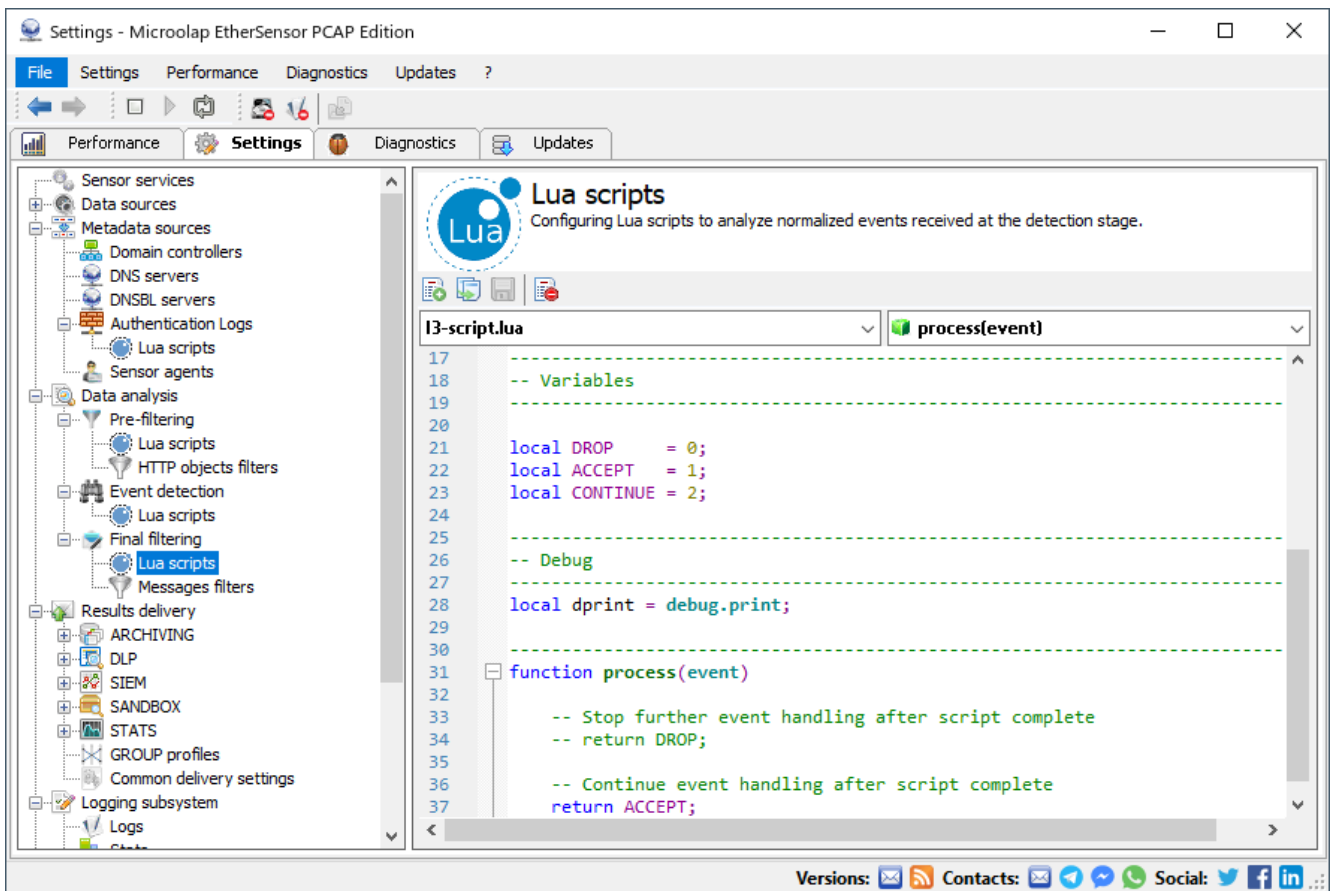


Fig.37. Editing scripts for final filtering of objects/events.

Setting up the final filtering of normalized events is also possible using the message filter in the **Messages filters** window of the management console. Filtering rules created using the message filter are stored in XML files located in the [INSTALLDIR]\config\filter directory.

When starting the EtherSensor Analyser service, the active filter from the XML format is transformed into a Lua script and always becomes the first in the chain of call of the final filtering scripts.

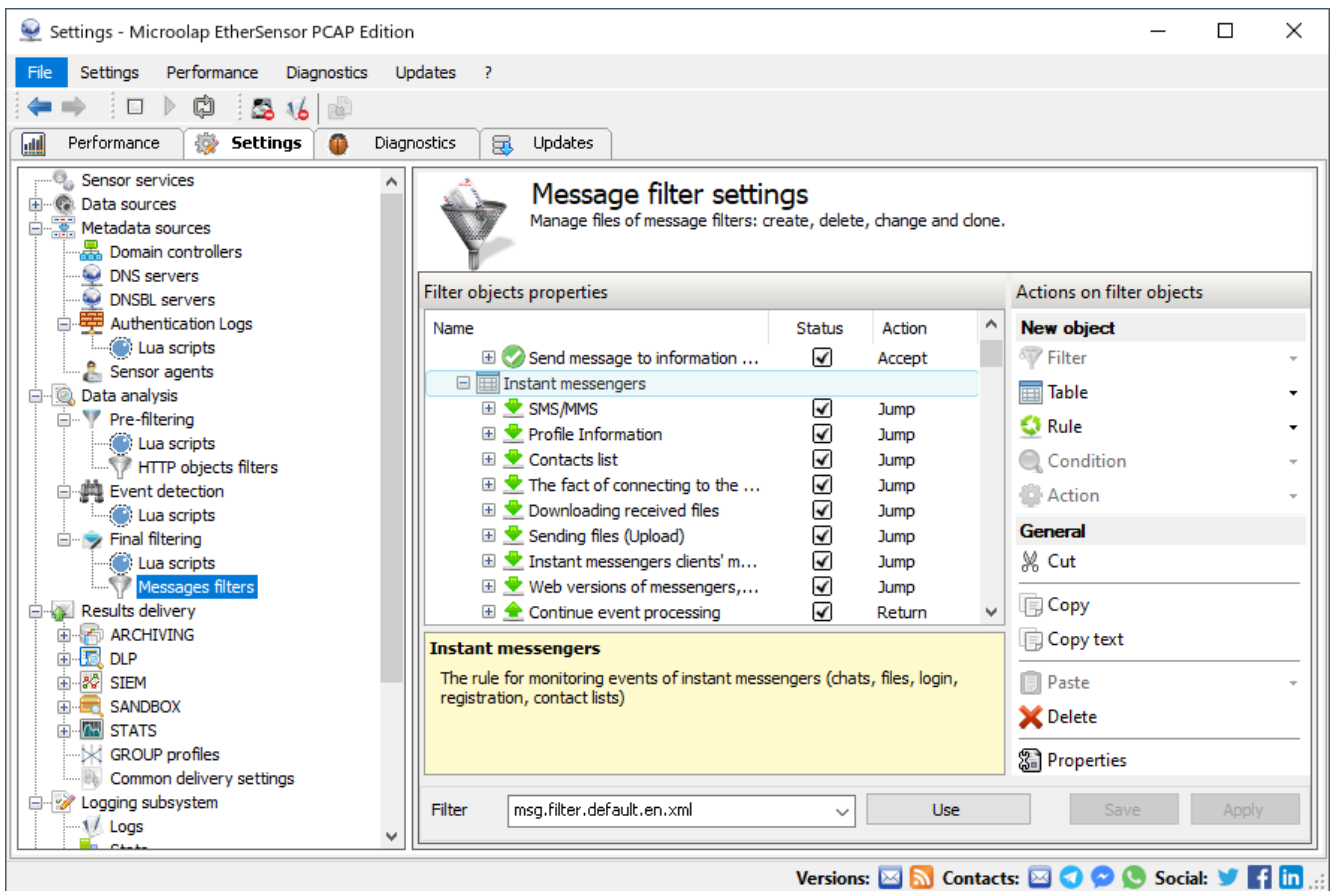


Fig.38. Configuring filters of objects/events.

Use the right side of the window to edit the filter, you can edit the active filter. But for the service EtherSensor Analyser to start using the modified filter, this filter should be made active and the service should be restarted.

For more information on creating filters, see the section Message filtering.⁽⁹⁹⁾

4.2. Generated events/messages

The result of EtherSensor are events/messages to systems-consumers generated from application level objects extracted from traffic. For further processing, EtherSensor delivers such messages to systems-consumers using the results delivery service⁽¹⁸⁴⁾.

The following mechanisms are used to generate a message to the system-consumer about the reconstructed communication:

1. A captured RAW event is formed into a binary object that describes known data about the nature of the captured objects, senders, receivers. In the process of RAW event processing the RAW event is filled with the collected data.
2. Upon completion of processing the object EtherSensor Analyser transfers it as a normalized event and other data files to the results delivery service.

3. The results delivery service, depending on the required type of transport and its settings, forms an object ready for delivery to the system-consumer from the normalized event. The structure of this object is determined by the delivery profiles.

For example, to deliver results via SMTP, an e-mail message is generated, where the captured data about the source and destination of the message, if any, are substituted into the sender and recipient addresses (FROM, TO, CC, BCC). The text of the message becomes the text of the email message, the transmitted files become attachments to the message. Other data accumulated by EtherSensor while processing the connection is stored as MIME message headers.

Message format for delivering the result

Messages delivered to the system-consumer must meet the requirements for messages transmitted via SMTP. Long headers can be splitted and encoded, the main encoding is UTF-8. EtherSensor detects object data types based on protocol headers.

1. For a number of web services it is allowed to capture several messages for one user operation in the web interface - this is the reaction of EtherSensor to sending drafts or attachments to the service.
2. It is acceptable that text and attachments are delivered as separate messages.
3. It is acceptable that for several messages one container with the content of these messages is transmitted to the consumer system (mainly for IM protocols to minimize traffic and related overhead).

Service MIME headers EtherSensor, example values

X-Sensor-Version: 6.1

Text string. It identifies the current version of EtherSensor.

X-Sensor-Id: sensor-01

Text string. Sensor ID, allows you to distinguish messages from different instances of EtherSensor or group them by this value. There can be any ASCII identifier, by default UHID is inserted.

X-Sensor-Session-Id: 6612456

An integer. Internal identifier of connections processed by EtherSensor. Issued by the message source (services EtherSensor ICAP or EtherSensor EtherCAP). It is logged in the capture.log.

X-Sensor-Net-Interface-Id: 00-21-28-10-58-80

Text string. If the message was captured by the EtherSensor EtherCAP service, the value is MAC address of the interface or capdrop - virtual identifier of the PCAP parsing driver.

In the case of an ICAP server, an identifier from the ICAP server configuration will be provided. This header allows you to trace the source of the message: from which interface or service the data was received for processing.

X-Sensor-Session-Level: 0

An integer. Shows how many different protocols it took to parse to get to the message. A protocol can be, for example, an HTTP connection, an HTTP-proxy connection through that connection, and GRE encapsulation.

X-Sensor-Src-Address: 10.31.90.22:47016**X-Sensor-Dst-Address: 193.203.100.139:8080**

IP addresses and connection ports: source and destination. Can be undefined if ICAP traffic is being processed without X-Client-IP and X-Server-IP headers. Example ICAP headers: X-Client-IP: 192.168.3.67, X-Server-IP: 123.45.67.89.

X-Sensor-Src-Host: pc-test.msk.su**X-Sensor-Dst-Host: nns-team.ru**

Text lines. Host names corresponding to X-Sensor-Src-Address and X-Sensor-Dst-Address headers. Since many networks provide internal addresses via DHCP, it is at the time the message is captured that the hostname should be resolved. EtherSensor does this by using a reverse DNS query to the EtherSensor Analyser DNS server specified in the configuration. If it was not resolved, the value will be <not resolved>.

X-Sensor-Protocol: HTTP

Text string. Name of the protocol detector that was used to parse the data. SMTP, ICQ, MRA and others are possible.

X-Sensor-Detector: phpbb

Text string. Name of the detector EtherSensor, which determined the presence of data in the connection.

X-Sensor-Attachments-Count: 0

An integer. If files were found in the message, their number is specified in this header.

X-Sensor-Object-Date: Fri, 17 Sep 2010 17:30:24 +0400

Text string. Time of connection capture (time of object creation in EtherSensor). The time zone is selected according to the settings of the sensor OS.

X-Sensor-Object-Size: 2735

An integer. The size of the captured object in bytes before processing.

X-Sensor-Object-MD5Hash: c326230de58279229862b18e818a3912

Text string, md5 hash of the captured object.

If messages have the same text but different size, hash, object or address date and source and destination ports, these are not duplicates, but very similar objects.

Matching md5 hashes of two different objects should not be in a normal situation. If it happens, then the same connection is processed EtherSensor several times (loop).

X-Sensor-Via: 1.1 off:1080 (squid/2.6.STABLE18)

X-Sensor-Forwarded-For: 10.255.241.31

Text lines. Headers from an HTTP connection are substituted with their values for existing headers. From these values it is possible to find out that the client of connection works behind a proxy server, and also sometimes to find out its address at the moment of message registration.

X-Sensor-Icap-Client-Username: user1

X-Sensor-Icap-Subscriber-Id: mike.smith@mycompany.com

X-Sensor-Icap-Authenticated-User:

TERBUDovLzE5Mi4xNjguMTluMTAwL289bXljb21wYW55LCBvdT1lbmdpbmVlcmluZywgY249bWlrZS5zbWl0aA==

X-Sensor-Icap-Authenticated-Group:

TERBUDovLzE5Mi4xNjguMTluMTAwL289bXljb21wYW55LCBvdT1lbmdpbmVlcmluZw==

The values of these headers are generated when processing ICAP traffic. For this, the corresponding headers of the ICAP protocol are used (X-Client-Username, X-Subscriber-ID, X-Authenticated-User, X-Authenticated-Groups).

X-Sensor-Filter-Name: TEST

X-Sensor-Tags: Filtered=1

X-Sensor-Labels: filter-begin-time="2010-09-17T17:30:24.6318125+04:00",

dns-begin="2010-09-17T17:30:24.6318125+04:00",

dns-end="2010-09-17T17:30:24.6318125+04:00",

Filtered="true",

filter-end-time="2010-09-17T17:30:24.6318125+04:00"

EtherSensor Analyser service headers. Allows you to determine the triggered filter, the nature of the message content, the tags and labels set, as well as to track the processing of the message in the filter. The resulting composition of these headers is highly dependent on the filter policy.

Date: Fri, 17 Sep 2010 17:30:24 +0400

This header duplicates the X-Sensor-Object-Date value.

From: anonymous@nns-team.ru

To: forum@nns-team.ru

CC, BCC and other headers

Subject: Re: World of Tanks

If possible, addresses or user IDs extracted from messages, request headers, etc. are inserted in the sender and receiver headers. These may be undefined and also depend on the detector: for example, if a protocol does not use the subject field of a message, it can be used for sensor information.

X-Sensor-RawSource-Type: LotusMail

All messages are marked with an X-Sensor-RawSource-Type header so that it is known from which data source the final message was received.

The values of this header in the current version EtherSensor (6.1) can be:

raw/http-request

Means that the primary data source was an HTTP request

raw/http-response

Means that the primary source of the data was the HTTP response.

raw/ftp-file

Means that the primary data source was an FTP file

raw/smtp-eml

Means that the primary data source was an SMTP message in EML format.

raw/pop3-eml

Means that the primary data source was a POP3 message in EML format.

raw/icq-contact-list

Means that the primary source of data was an ICQ contact list

raw/icq-message-list

Means that the primary data source was a list of ICQ messages

raw/icq-file

Means that the primary data source was a file transferred between ICQ clients

raw/icq-login-info

Means that the primary source of data was ICQ information about the user.

raw/mra-user-info

Means that the primary data source was MRA information about the user.

raw/mra-contact-list

Means that the primary source of data was the MRA contact list

raw/mra-message-list

Means that the primary data source was the MRA message list

raw/mra-file

Means that the primary data source was a file transferred between MRA clients

raw/msn-contact-list

Means that the primary source of data was MSN's contact list

raw/msn-message-list

Means that the primary data source was the MSN message list

raw/msn-file

Means that the primary data source was a file transferred between MSN clients

raw/xmpp-contact-list

Means that the primary data source was the XMPP contact list

raw/xmpp-message-list

Means that the primary data source was the XMPP message list

raw/xmpp-file

Means that the primary data source was a file transferred between XMPP clients

raw/irc-message-list

Means that the primary data source was an IRC message list

raw/irc-file

Means that the primary data source was a file transferred between IRC clients

raw/ssl-session-list

Means that the primary data source was a list of SSL sessions

raw/lotus-mail

Means that the primary data source was the LOTUS protocol message.

raw/lotus-attachment

Means that the primary data source was the LOTUS message attachments file.

X-Sensor-LicOption: Lotus

All messages are marked with an X-Sensor-LicOption header so that it is known which module processed the message. The values of this header in the current version EtherSensor (6.1) can be:

webmail

Means that the message was processed by the module with the "Web E-Mail" license option.

websocial

Means that the message was processed by the module with the "Social networks" license option.

email

Means that the message was processed by the module with the "E-Mail" license option.

im

Means that the message has been processed by the module with the "Instant messages" license option.

ft

Means that the message has been processed by the module with the "File transfer" license option.

webmailread

Means that the message has been processed by the module with the "Reading incoming web mail" license option.

lotus

Means that the message was processed by the module with the "Capture Lotus Notes" license option.

lotustxn

It means that the message was processed by the module with the "Extract messages from Lotus Notes Transaction Log" license option.

X-Sensor-UID: 0e515c8c-61eb-11e1-a529-000c29ff0707

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The value of the header uniquely identifies the user of the organization on a particular computer.

X-Sensor-UID-UserName: CN=Administrator,CN=Users,DC=bigbrother,DC=foo

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The value of the header uniquely identifies the user within the organization.

X-Sensor-UID-UserSID: S-1-5-21-86032015-1269853868-1024056280-1001

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The header value uniquely identifies the user within the organization and contains the current user security identifier (SID).

X-Sensor-UID-ComputerName: WS325-LOCK.bigbrother.foo

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The value of the header uniquely identifies the computer within the organization.

X-Sensor-UID-AdapterType: if_type_ethernet_csmacd

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The header value contains the type of network adapter through which the message was sent. The full list of possible options for the values of this field is available on the Microsoft website.

X-Sensor-UID-MacAddress: 00-1F-C6-2D-EA-40

This header is formed when the server EtherSensor and instances of EtherSensor Agent installed on workstations of network users interact.

The header value contains the MAC address of the network adapter through which the message was sent.

X-Sensor-UHID: UO2D-RNVO-JRN7-R1EN-91C0-61TA-1HP7-YRVF

Contains a unique identifier for each instance of EtherSensor and hardware set (Unique Hardware Identifier).

X-Sensor-Lotus-Messageld: <OF4D026078.B21F0C4F-ON44257C15.002E1625-44257C15.002E2225@LocalDomain>

Contains the unique identifier of the message transmitted through the Lotus Notes protocol.

X-Sensor-Lotus-Form: Reply

Contains the name of the form of the message transmitted via LOTUS protocol.

X-Sensor-Lotus-Mailer: Lotus Notes Release 8.5.2FP2 SHF236 October 24, 2011

Contains a string that identifies the client type of the Lotus Notes system.

X-Sensor-Lotus-INetPrincipal: UserName/OU/O@ServerName.Domain.com

Contains extended information about the sender of the message.

CNX-Sensor-Lotus-RouteServers: CN=lotus1/O=Company

Contains a list of Lotus Notes servers that participated in the transmission of the message.

X-Sensor-Lotus-References: <OF02B9DAC2.33CDF581-ON44257C15.002DF8CD@LocalDomain>

Contains a list of message IDs to which the current message refers.

4.3. Filtering interception results

The main idea of processing objects/events is to form rule chains that are combined into tables. The object is checked by rules that, depending on the rule triggering, can change the content of the object, its metadata, or the course of its further processing.

This idea is very similar to the filtering rules used in iptables.

The use of filters allows you to manage the process of event processing, direct the events to various external systems for further analysis, as well as to remove obviously not interesting events to reduce the load on EtherSensor.

4.3.1. Filtration basics

The key concepts of filtering captured messages/objects are:

Criterion

A logical expression consisting of one or more conditions combined by the OR, AND, XOR and NOT logical operators that checks the content and/or metadata of a message and determines whether the particular object matches the current rule.

Condition

Elementary condition for checking the message and/or its metadata. Checks one or more fields in a message or its metadata in a uniform way for a certain condition (equality, inequality, pattern match, etc.).

Action

Description of the action to be taken over the message in case the rule is triggered. For more details about possible actions on messages, see the Actions¹²⁶ section.

Rule

It consists of conditions and actions. If the message meets the conditions, actions are applied to it. There may be no conditions - then the criterion of "all messages" or "any message" is implicitly assumed.

Table

An ordered sequence of rules. The table must have a unique name. There is one system table named "main", which is the entry point for filtering the message. The "main" table must be present in every filter. Tables are discussed in more detail in the Tables¹⁰² section.

How filters work

All messages go through the filter tables, starting with the "main" table. When a message passes through a table, all the rules of this table are applied to it sequentially in the order in which they appear.

The application of the rule means:

- Checking a message (including metadata) against conditions
- Applying rule actions to it if the message matches this condition.

An action can mean both a basic operation (built-in action, for example, ACCEPT, DROP, JUMP, or RETURN), and a user-oriented action (these are actions of setting labels, tags, changing a message and / or its metadata, etc.). Basic operations can be both terminating, that is, stopping the processing of a message by a filter (this is ACCEPT, DROP), and non-terminating, that is, not interrupting the processing of a message (these are JUMP and RETURN).

The "main" table must always end with a rule with the "all messages" criterion and one of the terminating actions (ACCEPT or DROP). The rest of the tables must always end with a rule with the "all messages" condition and a terminal action (ACCEPT or DROP), or a RETURN action.

To jump a message from the current table to another, use the JUMP action. If the RETURN¹³⁰ action is applied to a message in this table, the message will return to the original table and continue traversing it starting with the next rule. Tables other than "main", and transitions to them may be needed to minimize the number of rules that a message must pass through in order for it to make an ACCEPT or DROP decision.

Also, the use of tables can be useful when combining groups of rules for processing messages, united by some "global" categories that cannot be described in the context of the criterion of one rule.

Based on this, it follows that the main table cannot end with a rule with a RETURN action, since there is simply nowhere to return from it. Also, it is not possible to make a JUMP to the main table.

It should be emphasized that it is forbidden to use explicit or implicit "looping" jumps, and this is checked at the compilation stage of the rules. For example, the script "main table" -> jump -> "boss-messages" table -> "shopping" table -> "spam" table -> "boss-messages" table will be locked at the filter compilation stage.

4.3.1.1. Filer configuration

Filter settings are stored in an XML file with a specific structure.

The filter configuration file starts with the standard XML document start tag with the XML 1.0 version and the document encoding.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
```

The root tag of the filter that contains the filter settings is then specified. The root tag has attributes "name" - the short name of the filter, and "version" - the version number of the filter. Now it is only "1.0".

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="main filter" version="1.0">
  ...
</filter>
```

You can write a comment for the filter - a text description to explain the content and purpose of the filter. Comments can contain any text and are ignored while working with the filter. A comment is added as a separate tag <comment>.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="main filter" version="1.0">
  <comment>This is a comment.</comment>
  ...
</filter>
```

Similarly, you can specify comments for tables and rules.

4.3.1.2. Table

A filter always consists of a "main" table and possibly other tables.

The processing of the message by the filter always starts with the "main" table. The table is defined in the filter with the "table" tag. Each table must have a unique name, specified in the "name" attribute of the XML tag "table". The table may have the XML tag "comment" as an optional comment.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="main filter" version="1.0">
  <comment>This is a comment.</comment>
  <table name="main">
    <comment>This is a comment for the table "main".</comment>
    ...
  </table>

  <table name="spam">
    ...
  </table>
</filter>
```

This filter specifies two tables - the required "main" table and the optional "spam" table.

4.3.1.3. Rules

The rules consist of a criterion consisting of one or more conditions and one or more actions to be taken if the message meets this set of criteria.

Rules are defined inside tables by the XML tag "rule".

A rule may have an optional name specified in the "name" attribute of the XML tag "rule".

A rule may have an optional comment - the XML tag "comment".

A rule can be active - it will be executed in the current filter configuration, or inactive - it will be ignored during the current filter configuration. The activity of a rule is determined by the mandatory attribute "enabled" of the XML tag "rule". The "enabled" attribute can take the following values:

1 or true:

Rule enabled, participates in message filtering

0 or false:

The rule is disabled, does not participate in message filtering, is ignored.

The rule criterion is described by the XML tag "match" inside the "rule" tag. The action of a rule is described by a sequence of XML "action" tags inside the "rule" tag.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="main filter" version="1.0">
  <comment>This is a comment.</comment>

  <table name="main">
    <comment>This is a comment for the table "main".</comment>
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>

    <rule name="spam" enabled="1">
      <comment>The rule for the messages of the SPAM category.</comment>
      <match ...> ... </match>
      <action ...> ... </action>
      <action ...> ... </action>
      <action ...> ... </action>
    </rule>

    <rule enabled="1">
      <action name="drop" />
    </rule>

  </table>
</filter>
```

In this example, there are three rules in the "main" table and all rules are enabled. The second rule (unlike the first one) has a name, a comment and several actions. The third rule is a mandatory termination rule: "reject all messages not accepted by the rules above".

If the "match" criterion is absent in a rule or is empty (<match />), the criterion "all messages" is implicitly implied: <c name="all"/>.

4.3.1.3.1. Criteria and conditions

A criterion in a rule is described by the XML tag "match" and determines whether the rule's actions for that message will be executed.

A criterion in a rule consists of one or more conditions that are associated with logical operations AND, OR, XOR or NOT. This allows you to build a criterion based on logical expressions that consist of conditions.

Example:

```
<rule>
  <match>
    <c name="all" />
  </match>
  <action name="drop" />
</rule>
```

This criterion consists of a single condition: "all messages."

Conditions in criteria

A condition is a simple check of a message or its metadata against something.

The condition is described by the XML tag "condition" or "c". The message either satisfies the condition, in which case the condition is TRUE, or does not satisfy the condition, in which case the result of the condition check will be FALSE. For a condition, its name is specified in the "name" attribute, which determines what and how to check in the message. The other attributes of the tag specify parameters for performing the action. The attribute names for additional parameters are context specific.

General structure of an XML tag describing a condition:

```
<condition name="The name of the condition." [additional parameters] />
```

or

```
<condition name="The name of the condition." [additional parameters] >
</condition>
```

or

```
<c name="The name of the condition." [additional parameters] />
```

or

```
<c name="The name of the condition." [additional parameters] ></c>
```

Most conditions use the value = "..." attribute, which specifies the values to check for compliance with the condition, or the data = "..." attribute, in which you can specify an external file to load data for such a test (these can be word sets, lists of domain names and other parameters). Which attributes are checked and how exactly are described in the sections of the corresponding conditions.

Boolean expressions in criteria

In the case when it is necessary to create a complex criterion for a rule, consisting of several conditions, these conditions can be combined into logical expressions using the logical operations AND, OR, XOR, NOT. Logical operations in the form of XML tags look like this:

<and> ... </and>:

Matches (... & ... & ... & ...) - all conditions that are inside the "and" tag are combined with the logical AND operation.

<or> ... </or>:

Matches (... | ... | ... | ...) - all conditions within the "or" tag are combined with the logical OR operation.

<xor> ... </xor>:

Matches (... ^ ... ^ ... ^ ...) - all conditions that are inside the "xor" tag are combined with a logical XOR operation.

<not> ... </not>:

Matches! (...) - Negation is applied to the condition.

Example:

```
<and>
  <c name="ccc1"/>
  <c name="ccc2"/>
</and>
```

means result (ccc1 & ccc2),

and the criterion:

```
<not><c name="ccc1"/></not>
```

means the result "not (ccc1)."

Any tags of logical operations can be nested.

Example:

```
<and>
  <c name="ccc1"/>
  <or>
    <c name="ccc2">
    <c name="ccc3">
  </or>
  <or>
    <c name="ccc4">
    <c name="ccc5">
    <not><c name="ccc6"></not>
  </or>
</and>
```

means result (ccc1 & (ccc2 | ccc3) & (ccc4 | ccc5 | !ccc6)).

That is, a message will meet the criteria specified in the example if: it meets the ccc1 condition and also meets (ccc2 or ccc3) and (ccc4 or ccc5) or does not meet (ccc6).

For clarity, this criterion can be splitted into condition blocks, which must necessarily return TRUE in the following tests:

1) the communication must meet the condition ccc1

AND

2) the communication must meet one of the conditions (ccc2 or ccc3)

AND

3) the message must satisfy one of the conditions (ccc4 or ccc5) OR it must not satisfy ccc6.

4.3.1.3.1.1. Condition ALL, *

A special condition to which all filtered objects satisfy.

Description

This condition is true for any objects, it is implicitly implied if the `<match> ... </match>` criterion is empty or is absent in the rule (the rule contains only "action" tags).

Format

```
<c name="all" />
<c name="*" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="all" or name="*".

Example:

The rule accepts all messages for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>

  <table name="main">

    <rule enabled="1">
      <comment>This rule accepts all messages for further processing.</comment>
      <match>
        <c name="all" />
      </match>
      <action name="accept" />
    </rule>
  </table>

</filter>
```

Example (implicit all condition):

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter comment.</comment>
  <table name="main">
    <rule enabled="1">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.2. Condition DETECTOR

Check the name of the detector that identified the message.

Description

When the detector is triggered, the system saves its name in the message metadata. The DETECTOR condition allows you to find out at the message filtering stage which detector has triggered it (it can be only one).

Format

```
<c name="detector" value="[detector-name(s)]" />
```

Attribute "name":

In the attribute "name" specify the name of the condition: name="detector".

Attribute "value":

In the value = "... " attribute, specify the name with which the name of the detector that triggered the message is compared. There can be several detector names (logical OR), in this case they are listed separated by commas ','. For readability, spaces are allowed after the comma.

Example:

Messages from the detectors mail.ru, yandex.ru.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Messages filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        Messages from detectors mail.ru,
        yandex.ru.
      </comment>
      <match>
        <c name="detector" value="mail.ru, yandex.ru" />
      </match>
      <action name="drop" />
    </rule>

  </table>
```

4.3.1.3.1.3. Condition PROTOCOL

Check which protocol was used to receive the message.

Description

This condition checks which protocol was used to receive the message.

Format

```
<c name="protocol" value="[protocol-name(s)]" />
```

Attribute "name":

In the attribute "name" specify the name of the condition - name="protocol".

Attribute "value":

In the value = "..." attribute, specify the name with which the name of the protocol by which the message was received is compared.

There can be several checked protocols (logical OR). In this case, they are listed separated by commas ','. For readability, it is acceptable to use spaces after the comma.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Messages filter.</comment>
  <table name="main">
    <rule enabled="1">
      <comment>
        Drop messages received by SMTP or IMAP.
      </comment>
      <match>
        <c name="protocol" value="smtp, imap" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.4. Condition MSG-SIZE, TOTAL-SIZE

Check the message size.

Description

MSG-SIZE and TOTAL-SIZE conditions check message size.

msg-size

Considers the size of extracted texts and the size of extracted attachments.

total-size

Considers the total size of the extracted text, the size of the extracted attachments and the size of the source data from which they are derived.

Format

```
<c name="msg-size" op="<operation>" value="<compare pattern>" />  
<c name="total-size" op="<operation>" value="<compare pattern>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="msg-size" or name="total-size".

Attribute "value":

In the attribute value="..." specify the number with which the message size is compared:

<number> or <number>B

Indicates the size in bytes

<number>K

Indicates the size in kilobytes

<number>M

Indicates the size in megabytes.

<number>G

Indicates the size in gigabytes.

Attribute "op":

The attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the size is EQUAL to the specified number

ne or != or <>

The condition is met if the size is NOT EQUAL to the specified number.

lt or <

The condition is met if the size is LESS than the specified number

gt or >

The condition is met if the size is GREATER than the specified number

le or <=

The condition is met if the size is LESS OR EQUAL than the specified number.

ge or >=

The condition is met if the size is GREATER OR EQUAL than the specified number.

Example:

The rule stops processing messages with the size more than 100KB without taking into account the source data or with the size more than 1MB.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Messages filter.</comment>

  <table name="main">

    <rule enabled="1">
      <match>
        <or>
          <c name="msg-size" op=">" value="100K"/>
          <c name="total-size" op="gt" value="1M"/>
        </or>
      </match>
      <action name="drop" />
    </rule>

  </table>
</filter>
```

4.3.1.3.1.5. Condition CHECK-MD5

Check the MD5 hash of a message for reappearance within a certain time interval (tracking duplicate messages).

Description

This condition checks if a message with the same MD5 hash as the currently being checked message has already occurred within the specified time period.

Format

```
<c name="check-md5" time="<timeout>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="check-md5".

Attribute "time":

In the time="..." attribute, specify the waiting time in milliseconds.

The waiting time may not be less than 1 millisecond and may not exceed 5 minutes, i.e. $5*60*1000 = 300000$ milliseconds.

Example:

If there were repeated messages with the same MD5 within two seconds, delete them.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="check-md5" time="2000" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.6. Condition CHECK-MESSAGE-ID

Check the Message-ID header (for LOTUS protocol the X-Sensor-Lotus-Messageld header is checked) of the message for reappearance during a certain time interval (tracking duplicates of messages).

Description

This condition checks if a message with the same Message-ID header (X-Sensor-Lotus-Messageld header is checked for LOTUS protocol) as the currently checked message has occurred during the specified period of time.

Format

```
<c name="check-message-id" time="<timeout>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="check-message-id".

Attribute "time":

In the time="..." attribute, specify the waiting time in milliseconds.

The waiting time may not be less than 1 millisecond and may not exceed 5 minutes, i.e. $5*60*1000 = 300000$ milliseconds.

Example:

If there were repeated messages with the same Message-ID header within two seconds, delete them.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>
  <table name="main">
    <rule enabled="1">
      <match>
        <c name="check-message-id" time="2000" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.7. Condition HOSTNAME

Check the source and destination host names in the message.

Description

This condition checks the source or destination host names against the string or against the wildcard or regexp template. For this condition to work properly, hostname resolution must be performed (see section DNS Action¹³⁶).

Tip: if you need to check only the destination host and only for the HTTP protocol, then resolving names through the "expensive" in terms of time and resources DNS action makes no sense, because the destination host name will be available from the Host header in the HTTP request.

Format

```
<c name="hostname"
  address="<address type>"
  op="<operation>"
  value="<compare pattern>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="hostname".

Attribute "address":

In the attribute address="..." specify the type of address to be checked. Possible values:

src or client

Check source address only

dst or server

Check destination address only

both or all or *

Check both addresses (source and destination).

If this attribute is missing, it means "both" to check both addresses.

Attribute "op":

The attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the value to be checked CONTAINS the specified value

ne or != or <>

The condition is met if the value to be checked DOES NOT CONTAIN the specified value.

wc or wildcard

The condition is met if the value being checked matches the specified wildcard template

regex or regexp

The condition is met if the value being checked matches the specified regexp template

Attribute "value":

In the attribute value="..." specify the string with which the value is compared, or a template for checking.

Example:

Messages sent to hosts *.yandex.ru should be ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="hostname" address="server" op="wc" value="*.yandex.ru" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.8. Condition IP

Check the IP addresses of the client or server for range entry or subnet entry.

Description

This condition checks the IP addresses of the client or server for range or subnet membership.

Tips:

1. Unwanted traffic should be cut off as soon as possible. Performance EtherSensor depends on it.
2. All traffic from a certain IP or range is best cut off in the IP filter of the EtherSensor EtherCAP service.
3. Certain HTTP traffic from some IP (if it is possible to define such criteria) is best cut off in the HTTP filter.
4. Certain messages from a certain IP address must be processed in the message filter.

Format

```
<c name="ip" address="<address type>" value="<ip-range>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="ip".

Attribute "address":

In the attribute address="..." specify the type of address to be checked. Possible values:

src or client

Check source address

dst or server

Check destination address

any or *

Matching any of the addresses.

If the attribute is omitted, it's implied by default "*".

Attribute "value":

In the attribute value="..." specify a value for comparison. Possible values:

ipaddress

Checks the IP address for equality. For example, value="192.168.0.10".

ip1-ip2

Checks for an IP address in the range. For example, value = "192.168.0.1-192.168.0.10"

ip/netmask

Checks that the IP address of the specified subnet belongs to it. For example, value="192.168.0.1/255.255.255.0"

ip/netmaskbits

Checks if the IP address of the specified subnet belongs to. For example, value = "192.168.0.1/24".

Example:

Ignore messages from the client's machine 192.168.0.15.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">

    <rule enabled="1">
      <match>
        <c name="ip" address="client" value="192.168.0.15" />
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.3.1.9. Condition HEADER

Checks the value of one of the message headers.

Description

This condition checks the value of the message header for the substring in the string or for the wildcard or regexp template. Message headers are any metadata headers generated by the detectors like "X-Sensor-...".

These are also headers from email messages, except for headers:

From

To

Cc

Bcc

Subject

Date

Content-Type

Content-Transfer-Encoding

Keep in mind that for more efficient filtering it is not necessary to filter metadata headers "X-Sensor-...", for which there are specialized verification conditions. For example, for the "X-Sensor-Detector" header it is more efficient to check not the value of this header through the HEADER condition, but to use the special condition `<c name="detector" value="...". />`

Format

```
<c name="header"  
  headername="<header name>"  
  op="<operation>"  
  value="<compare pattern>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="header".

Attribute "headername":

In the attribute headername="..." specify the name of the header to be checked.

Attribute "op":

The attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the value to be checked CONTAINS the specified value

ne or != or <>

The condition is met if the value to be checked DOES NOT CONTAIN the specified value.

wc or wildcard

The condition is met if the value being checked matches the specified wildcard template

regex or regexp

The condition is met if the value being checked matches the specified regexp template

Attribute "value":

In the attribute value="..." you should specify the string, with which the value is compared, or a template for checking.

Example:

```
<c name="header" headername="X-Priority" op="eq" value="3" />
```

The condition is met if the X-Priority header is present in the message and its value contains the line "3".

```
<c name="header" headername="X-Mailer" op="!=" value="Outlook" />
```

This condition is met if the X-Mailer header is present in the message and its value does not contain the "Outlook" line.

```
<c name="header"  
  headername="X-Sensor-Net-Interface-Id"  
  op="eq"  
  value="01-icap" />
```

The condition is met if the X-Sensor-Net-Interface-Id header is present in the message and its value contains the line "01-icap".

```
<c name="header"
  headername="X-Sensor-Net-Interface-Id"
  op="wc"
  value="*-icap" />
```

or

```
<c name="header"
  headername="X-Sensor-Net-Interface-Id"
  op="wildcard"
  value="*-icap" />
```

The condition is met if the message contains the X-Sensor-Net-Interface-Id header and its value corresponds to the wildcard template "*-icap".

Example:

Messages sent through Outlook should be ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="header"
          headername="X-Mailer"
          op="=="
          value="Outlook" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.10. Condition ATTACH-NAME

Check the names of attachments in the message.

Description

This condition checks the names of attachments for string matching or wildcard or regexp pattern matching.

Format

```
<c name="attach-name" op="<operation>" value="<compare pattern>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="attach-name".

Attribute "op":

In the attribute op="..." specify the type of comparison operation. Possible values:

eq or = or ==

The condition is met if the value to be checked CONTAINS the specified value

ne or != or <>

The condition is met if the value to be checked DOES NOT CONTAIN the specified value.

wc or wildcard

The condition is met if the value being checked matches the specified wildcard template

regex or regexp

The condition is met if the value being checked matches the specified regexp template

Attribute "value":

In the attribute value="..." specify the string with which the value is compared, or a template for checking.

Example:

```
<c name="attach-name" op="eq" value="instruction.doc" />
```

This condition is met if any message attachment name contains "instruction.doc".

```
<c name="attach-name" op="eq" value="instruction.doc" />
```

This condition is met if any name of the message attachment does not contain "instruction.doc".

```
<c name="attach-name" op="wc" value="*.doc" />
```

or

```
<c name="attach-name" op="wildcard" value="*.doc" />
```

This condition is met if any message attachment name matches the wildcard template "*.doc".

```
<c name="attach-name" op="re" value=".+\.doc" />
```

or

```
<c name="attach-name" op="regexp" value=".+\.doc" />
```

This condition is met if any message attachment name matches the ".+\.doc" regex template.

```
<c name="attach-name" op="re" value=".+((\.\doc)|(\.\exe)|(\.\zip))" />
```

This condition is met if any message attachment name matches the regex template ".+((\.\doc)|(\.\exe)|(\.\zip))".

Example:

Ignore messages with * .exe attachments.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="attach-name" op="re" value="+\.exe" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.11. Condition ATTACH-EXIST

Check the message for attachments.

Description

This is the condition that all messages containing any attachments are satisfied.

Format

```
<c name="attach-exist" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="attach-exist".

Example:

Messages with any attachments should be ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="attach-exist" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.12. Condition TAG

Check whether the tag is set and the counter value (see TAG Action ¹³³).

Description

This condition checks the existence of a tag or tag counter value.

If there is no tag, the condition is not met.

Format

```
<c name="tag" op="<operation>" value="<compare value>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="tag".

Attribute "op":

The attribute op="..." indicates the test criterion:

eq or = or ==

The condition is met if the counter value is EQUAL to the specified number.

ne or != or <>

The condition is met if the counter value is NOT EQUAL to the specified number.

lt or <

The condition is met if the counter value is LESS than the specified number

gt or >

The condition is met if the counter value is GREATER than the specified number.

le or <=

The condition is met if the counter value is LESS OR EQUAL than the specified number.

ge or >=

The condition is met if the counter value is GREATER OR EQUAL to the specified number.

exist

The condition is met if the tag exists (already set earlier for this object).

By default (if the attribute "op" is missing) the value "exist" is accepted. It is not necessary to specify the "value" attribute for the operation "exist".

Attribute "value":

In the attribute value="..." specify the number against which the tag counter value is compared.

Example:

```
<c name="tag" tag="SPAM" op="exist" />
```

or

```
<c name="tag" tag="SPAM" />
```

The condition is met if the object has the "SPAM" tag.

```
<c name="tag" tag="SPAM" op="eq" value="1" />
```

This condition is fulfilled if the "SPAM" tag is set for the message and its counter is 1.

```
<c name="tag" tag="SPAM" op=">" value="1" />
```

The condition is met if the "SPAM" tag is set for the message and its counter value is greater than 1.

```
<c name="tag" tag="SPAM" op=">=" value="3" />
```

This condition is fulfilled if the "SPAM" tag is set for the message and its counter value is greater than or equal to 3.

Example:

Messages marked with a SPAM tag with a value greater than 1 should be ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match>
        <c name="tag" tag="SPAM" op=">" value="1" />
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.1.13. Condition FROM, TO, CC, BCC, ADDRESS, SUBJECT

Check one of the fields from, to, cc, bcc, subject, address.

Description

This condition checks the value of the field for the content of the specified substring or for the wildcard or regexp pattern.

from

Checks FROM field (sender address)

to

Checks the TO field (recipient address)

cc

Checking the CC field

bcc

Checking the BCC field

address

Checks all address fields (from, to, cc, bcc). If a match is found in any of them, then the condition is met.

subject

Checks the SUBJECT field.

Format

```
<c name="from" op="<operation>" value="<compare pattern>" />
<c name="to" op="<operation>" value="<compare pattern>" />
<c name="cc" op="<operation>" value="<compare pattern>" />
<c name="bcc" op="<operation>" value="<compare pattern>" />
<c name="subject" op="<operation>" value="<compare pattern>" />
<c name="address" op="<operation>" value="<compare pattern>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="from", name="to", name="cc", name="bcc", name="subject" or name="address".

Attribute "op":

The attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the value of the field **CONTAINS** the specified value

ne or != or <>

The condition is met if the field value **DOES NOT CONTAIN** the specified value

wc or wildcard

The condition is met if the value of the field matches the specified wildcard pattern

regex or regexp

The condition is met if the value of the field matches the specified regexp-template.

Attribute "value":

In the attribute value="..." you should specify the string, with which the value is compared, or a template for checking.

Example:

```
<c name="from" op="eq" value="xxx@mail.ru" />
```

The condition is met if the FROM field of the message contains the string "xxx@mail.ru".

```
<c name="to" op="!=" value="xxx@mail.ru" />
```


Example:

Letters from the address or to *@mail.ru continue to be processed, everything else is ignored.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">

    <rule enabled="1">
      <match>
        <c name="address" op="wildcard" value="*@mail.ru" />
      </match>
      <action name="accept" />
    </rule>

    <rule enabled="1">
      <action name="drop" />
    </rule>

  </table>
</filter>
```

4.3.1.3.1.14. Condition TEXT

Check for keywords in the text of the message or in the subject.

Description

Check for keywords in the text of the message or in the subject.

The keyword check is based on the general occurrence of the keyword in the text, not just as individual words. In other words, it works as if you were looking for a wildcard template "*keyword*" for each keyword.

For example, under the criterion of the keyword "secret" will fall the words "secret", "sECrEt", "secretary", "insecretory", etc.

Character case is not taken into account when searching for keywords.

Format

```
<c name="text" op="<operation>" value="<compare pattern>" />
<c name="text" op="<operation>" data="<data source>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="text".

Attribute "op":

The attribute op="..." indicates the test criterion:

all

All specified keywords must be found in the message

one

It's enough to find at least one key word in the message.

By default (if the attribute "op" is missing) the value "one" is accepted.

Attribute "value":

In the attribute value="..." list the keywords. If there are several keywords, then list them by comma ','. Also keywords can be specified in the value of tag `<c>key-word-list</c>` itself.

Attribute "data":

The data="..." attribute may specify another source of keywords. This allows you to specify large sets of words when it is inconvenient to do so in the attribute value="...".

Possible values:

data="<external data name>"

Load the keyword list from the external block in the filter (tag `<data name="extern-data-name">...</data>`)

data="extern://<external data name>"

Load keyword list from external block in filter (tag `<data name="extern-data-name">...</data>`). Keywords are listed by commas

data="file://<full-file-path>"

Download the list of keywords from the specified file. Keywords are listed each on a separate line (no commas required).

Example:

```
<c name="text" op="one" value="secret1, secret2, secret3, secret4" />
```

or

```
<c name="text" op="one">secret1, secret2, secret3, secret4</c>
```

This condition is met if at least one of the specified keywords is found in the subject line or in the text of the message: both as separate words and as part of another word.

Examples of texts:

1. The subject of the e-mail is "RE: secret 1 secret2" - condition MET.

2. The subject of the e-mail is "RE: secret3 secret4" - condition MET.

3. The text of the letter is "I send you our secret 1 secret2" - condition MET.

4. The subject of the e-mail is "The quick brown fox jumps over the lazy dog" - condition NOT MET.

```
<c name="text" op="all" value="secret1, secret2" />
```

or

```
<c name="text" op="all">secret1, secret2</c>
```

This condition is met if both specified keywords are found in the subject line or in the text of the message as separate words, or as parts of other words.

Examples of texts:

1. The subject of the e-mail is "RE: secret1 secret2" - condition MET.
2. Subject of the e-mail - "RE: secret1 from the boil" - condition NOT MET.
3. The text of the e-mail - "I send you our secret2" - the condition NOT MET.
4. The text of the e-mail is "I send you our secret1 from secret2" - condition MET.

```
<c name="text" op="one" data="dictionary.txt" />
```

Load the list of keywords to check from "dictionary.txt" file.

Example:

Messages containing keywords from the keywords list should be accepted for further processing. Ignore everything else.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">

    <rule enabled="1">
      <comment></comment>
      <match>
        <c name="text" data="extern://keywords" />
      </match>
      <action name="accept" />
    </rule>

    <rule enabled="1">
      <action name="drop" />
    </rule>

  </table>

  <data name="keywords">
    secret1,
    secret2,
    secret3,
    secret4
  </data>

</filter>
```

4.3.1.3.2. Actions

All actions are described in XML-tags "action". For each action, its name is indicated in the "name" attribute, which determines what operation will be performed for the message. The other attributes of the tag specify parameters for performing the action. The attribute names for additional parameters are action-specific.

The general structure of the XML tag describing the action:

```
<action name="action name" [parameters] />
```

or

```
<action name="action name" [parameters]></action>
```

Most actions use the value = "..." attribute, which specifies the values for performing the action, or the data = "..." attribute, in which you can specify an external file to load data for performing the action (these can be word sets, lists of domain names and other parameters). The description of the attributes and their actions is detailed in the sections of the corresponding actions.

Basic actions

Basic actions are actions that change the order in which a message passes through the filter. Basic actions are ACCEPT, DROP, JUMP and RETURN.

Custom Actions

Custom actions are actions that modify the message or its metadata, as well as perform the processing of the message through external services (sandboxes, antivirus, URL-categorizers, DNSBL-servers, etc.).

4.3.1.3.2.1. Action ACCEPT

Accept the message for further processing.

Description

The ACCEPT action terminates the filters on the current message and immediately passes it to further processing (that is, to give it to the system-consumer according to a predefined delivery profile or default profile).

Format

```
<action name="accept" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="accept".

Example:

Messages that have reached this rule will be accepted for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>

    <rule enabled="1">
      <comment>
        All the messages reaching this rule are accepted for
        further processing.
      </comment>
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.1.3.2.2. Action DROP

Stop processing the message, delete the accumulated data.

Description

This action stops processing the current message and tells EtherSensor that all its data and metadata must be destroyed.

Format

```
<action name="drop" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="drop".

Example:

Destroy the data and metadata of messages that reach this rule.metadata

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <comment>
        Details/metadata of any message reaching this point
        are destroyed (discarded).
      </comment>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.1.3.2.3. Action JUMP

Continue processing the message in another table.

Description

This action continues to process the message in another table.

Format

```
<action name="jump" value="<table name>" /><action name="jump" value="<table name>" />
```

Attribute "name":

In the attribute "name" specify the name of the action: name="jump".

Attribute "value":

In the attribute value="..." specify the name of the table to which you want to go for further processing of the message.

Keep in mind that switching over JUMP to the main table is prohibited. Jumps that may cause explicit or implicit cycling are also prohibited.

Example:

From the main table, go to the yandex table for processing messages from mail.ru and yandex.ru. Yandex table: the only rule destroys messages larger than 100KB. After that, it returns to the main table.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <comment>
        Processing of messages from the mail.ru,
        yandex.ru detectors in the yandex table.
      </comment>
      <match>
        <c name="detector" value="mail.ru, yandex.ru" />
      </match>
      <action name="jump" value="yandex"/>
    </rule>

    <rule enabled="1">
      <action name="drop" />
    </rule>
  </table>

  <table name="yandex">
    <comment>
      The table processes messages from the mail.ru,
      yandex.ru detectors.
    </comment>

    <rule enabled="1">
      <comment>
        Discard messages larger than 100 Kb.
      </comment>
      <match>
        <c name="size" op=">" value="100K"/>
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <comment>
        Returning to the main table for further processing.
      </comment>
      <action name="return" />
    </rule>
  </table>
</filter>
```

4.3.1.3.2.4. Action RETURN

Return to the previous (called) table and continue executing it with the next rule.

Description

This action returns the processing of the message to the previous (called) table and continues with the next rule.

Format

```
<action name="return" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="return".

The action cannot be applied in the table main.

Example:

In this example, in the table "main" there is some processing of the message. When processing a message, the second rule is a rule named "spam": if the message meets the criteria of this rule, it is passed for further processing in the table "spam".

After passing the rules in the "spam" table, if the message processing is not interrupted by the previous rules, then having reached the rule with the name "return-to-main", the message processing will continue in the "main" table with the rule following the "spam" rule.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="main filter" version="1.0">

  <table name="main">
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>
    <rule name="spam" enabled="1">
      <match ...> ... </match>
      <action name="jump" value="spam"/>
    </rule>
    <rule enabled="1">
      <action name="drop" />
    </rule>
  </table>

  <table name="spam">
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>
    <rule name="return-to-main" enabled="1">
      <action name="return" />
    </rule>
  </table>
</filter>
```

4.3.1.3.2.5. Action LABEL

Adds a string label to the message metadata.

Description

This action sets a string label in the metadata of the currently processed message.

If this string label has already been set for a message (or the HTTP object from which the message was received), then the value of the string label is replaced by the newer one.

Used to add descriptions to messages.

Format

```
<action name="label" label="<label name>" value="<label value>" />
```

or

```
<action name="label" label="<label name>" > label value </action>
```

Attribute "name":

In the attribute "name" specify the name of the action: name="label".

Attribute "label":

In the label="..." attribute, specify the name of the string label to be set.

Attribute "value":

The value="..." attribute specifies the value (string) for the label.

The value can also be listed in the "action" tag itself.

Example:

```
<action name="label"
  label="VIRUS-DESCR"
  value="Win.32.BlackHorse.trojan.virus - mail worm, very dangerous!!!" />
```

or

```
<action name="label"
  label="VIRUS-DESCR">
  Win.32.BlackHorse.trojan.virus -
  mail worm, very dangerous!!!
</action>
```

Sets a string label with the name "VIRUS-DESCR" for the message and writes the line "Win.32.BlackHorse.trojan.virus - mail worm, very dangerous!!!" into it.

Example:

Tag messages processed by mail.ru, yandex.ru detectors with the CONTENT-DESCR label.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is the comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Mark messages detected by the mail.ru,
        yandex.ru detectors with the CONTENT-DESCR label.
      </comment>
      <match>
        <c name="detector" value="mail.ru, yandex.ru" />
      </match>
      <action name="label"
        label="CONTENT-DESCR"
        value="Russian mail services"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.6. Action TAG

Adds a tag (numeric label) to the message metadata.

Description

This action sets a tag (numeric label) in the message metadata. There can be several tags. In this case they are listed via semicolon ',' or semicolon ';'. If the same tag is set for an object several times, the "level" (its numeric value) is increased. In other words, each tag has an internal counter that shows how many times it was set for this object.

Whether a tag is set or not, and the value of its counter (how many times it has been set for the current object) can be checked later in filter conditions for decision-making.

By default, the counter value is increased by 1 when the tag is set. If the tag counter is to be increased by more than 1, you can set the value by which the counter is to be increased after the tag name in parentheses: "TAG(...)". For example, SPAM(3) - increases the counter value for a SPAM tag by 3, and SPAM(1) is simply SPAM. This may be necessary in cases where the conditions setting the same tag have different meanings, importance or priority.

The value for changing the counter may be negative. SPAM(-3) - reduces the counter value for the SPAM tag by 3.

Format

```
<action name="tag" value="<tag list>" />
```

or

```
<action name="tag" > tag list </action>
```

Attribute "name":

In the "name" attribute specify the name of the action: name="tag".

Attribute "value":

The value="..." attribute lists the tag names.

Tag names can also be listed in the "action" tag itself.

Example:

```
<action name="tag" value="SPAM" />
```

Sets a tag named "SPAM".

```
<action name="tag" value="SPAM(3)" />
```

Sets a tag named "SPAM" and increases its counter by 3.

```
<action name="tag" value="SPAM, shopping" />
```

Sets tags with the names "SPAM" and "shopping".

```
<action name="tag" value="SPAM(3), shopping(2)" />
```

Sets tags with the names "SPAM" and "shopping" and increases their counters by 3 and 2 respectively.

```
<action name="tag" value="SPAM, shopping">VIP-OFFICE</action>
```

Also sets tags with the names "SPAM" and "shopping".

```
<action name="tag" value="SPAM, shopping">VIP-OFFICE</action>
```

Sets tags with names "SPAM", "shopping", "VIP-OFFICE".

Example:

Tag RUS_MAIL to mark requests for popular Russian mail services.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is the comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Mark requests to popular Russian mail services
        with the RUS_MAIL tag.
      </comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
        <c name="req-header"
          headername="Host"
          op="eq"
          value="mail.yandex.ru" />
        <c name="req-header"
          headername="Host" op="eq"
          value="mail.rambler.ru" />
      </match>
      <action name="tag" value="RUS_MAIL"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.7. Action DATETIME

Set the message to a string label with the current date and time.

Description

This action sets a label for the message with the current date and time. If this label has already been set for a message, it will be replaced by a newer one. It is used to track the time of message passing through the filter.

Format

```
<action name="datetime" value="label-name" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="datetime".

Attribute "value":

In the attribute value="..." specify the name of the string label to be set.

Example:

Add a FILTER-BEGIN processing start label and a FILTER-END processing end label to all messages.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <comment>
        Set the FILTER-BEGIN processing start label
        for all the messages.
      </comment>
      <action name="datetime" value="FILTER-BEGIN" />
    </rule>

    <rule enabled="1">
      <comment>
        Set the FILTER-END processing end label
        for all the messages.
      </comment>
      <action name="datetime" value="FILTER-END" />
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.1.3.2.8. Action DNS

Performs DNS name resolution to IP addresses and IP addresses to names for unknown host addresses.

Description

This action resolves the hostnames and IP addresses of the session.

Each session has a source (client) and receiver (server) with their own names or IP addresses. It is possible that the host name of the client or server is known for the message but its IP address is not known or vice versa. The DNS action resolves the missing information about the client or server address.

If an IP address is known but the host name corresponding to that IP is not known, it is queried from the DNS server. The same is done if the host name is known, but the IP address is not known. If the message already contains the host name and its IP, no action is taken.

Also, if the names are successfully obtained, X-Sensor-Src-Host, X-Sensor-Dst-Host headers are added to the metadata and it becomes possible to use them in the "header" condition.

You should also keep in mind that a condition like "hostname" (hostname check) should only be applied after performing a DNS action.

Format

```
<action name="dns" address="<address type for resolving>" />
```

Attribute "name":

In the attribute "name" specify the name of the action: name="dns".

Attribute "address":

In the attribute address="..." specify the type of address for name resolution. Possible values:

src or client

Resolve name only for source address

dst or server

Resolve name only for the destination address

both or all or *

Resolve names for both addresses (source and destination).

If this attribute is missing, the action "both" is meant to perform name resolution for both addresses.

Example:

Perform name resolution for the message.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is the comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Resolve names for the message.
      </comment>
      <action name="dns" address="server"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.9. Action DNSBL-LH, DNSBL-RH

Verifies that the addresses of the message are in DNSBL lists, and if they are confirmed, sets the specified tag.

Description

dnsbl-rh - check IP address in DNSBL-RHSBL.

dnsbl-lh - check hostname in DNSBL-LHSBL.

When one of the addresses appears in one of the DNSBLs, this action increases the value of the specified tag by 1.

The list is always fully checked.

The presence of each address (or hostname) into each DNSBL increases the tag by 1. That is, if the same DNSBL includes both the source and destination addresses, then the tag is incremented for each entry.

You can then check the tag value under filter conditions and make decisions based on the entry of the message addresses into DNSBL.

Format

```
<action name="dnsbl-rh"
        address="<address-type>"
        tag="<tag-name>"
        value="<dns-bl domains list>" />
<action name="dnsbl-rh"
        address="<address-type>"
        tag="<tag-name>"
        data="<dns-bl domains list source>" />
<action name="dnsbl-lh"
        address="<address-type>"
        tag="<tag-name>"
        value="<dns-bl domains list>" />
<action name="dnsbl-lh"
        address="<address-type>"
        tag="<tag-name>"
        data="<dns-bl domains list source>" />
```

Attribute "name":

In the attribute "name" specify the name of the action: name="dns".

Attribute "address":

In the attribute address="..." specify the type of address for name resolution. Possible values:

src or client

Check source address only

dst or server

Check destination address only

both or all or *

Check both addresses (source and destination).

If this attribute is missing, the action "both" is meant to check both addresses.

Attribute "tag":

In the tag="..." attribute, specify the name of the tag to be increased.

Attribute "value":

The attribute value="..." lists DNSBL domains.

If there's more than one domain, they're separated by a comma ','.

Domains can also be specified in the value of <action>dns-bl-domains-list</action> tag itself.

Attribute "data":

The data="..." attribute may contain a different DNSBL list source. This allows you to specify long lists when it is inconvenient to do so in the attribute value="...".

Possible values:

data="<external data name>"

Load the list from the external block in the filter (tag <data name="extern-data-name">...</data>)

data="extern://<external data name>"

Load the list from the external block in the filter (tag <data name="extern-data-name">...</data>). DNSBL domains are listed by commas.

data="file://<full-file-path>"

Download the list from the specified file. DNSBL domains are listed on separate lines (no commas required).

Example:

```
<action name="dnsbl-rh" tag="SPAM">
  bl.spamcop.net, vote.drbl.sandy.ru,
  sbl.spamhaus.org, cblplus.anti-spam.org.cn
</action>
```

Increases the "SPAM" tag if the message addresses are part of a DNSBL. If any address is included in only one list - "SPAM" will be equal to 1. If the address is included in two lists - "SPAM" will be equal to 2, etc.

Later on you can check if ("SPAM" > 3) it is definitely spam, and if less it is just a little suspicious. If both addresses are part of the same DNSBL, the tag increases by 2.

```
<action name="dnsbl-rh" address="src" tag="SPAM">
  bl.spamcop.net, vote.drbl.sandy.ru,
  sbl.spamhaus.org, cblplus.anti-spam.org.cn
</action>
```

Only checks the source IP address.

```
<action name="dnsbl-lh" address="dst" tag="SPAM">
  bl.spamcop.net, vote.drbl.sandy.ru,
  sbl.spamhaus.org, cblplus.anti-spam.org.cn
</action>
```

Checking only the destination's hostname.

Example:

Check all message addresses via DNSBL list. Positive alarms are marked with the SPAM tag. Delete messages with SPAM tag, accept others.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <comment>
        Checks all message addresses against the DNS-BL list. Hits
        are to be marked with the SPAM tag.
      </comment>
      <action name="dnsbl-rh"
        address="both" tag="SPAM"
        data="extern://dns-bl-list" />
    </rule>

    <rule enabled="1">
      <comment>Delete spam.</comment>
      <match>
        <c name="tag-exist" value="SPAM" />
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <comment>Accept the rest.</comment>
      <action name="accept" />
    </rule>
  </table>

  <data name="dns-bl-list">
    bl.spamcop.net,
    vote.drbl.sandy.ru,
    sbl.spamhaus.org,
    cblplus.anti-spam.org.cn
  </data>
</filter>
```

4.3.1.3.2.10. Action SAVE RAW DATA

Enable/disable saving of source data from which the message was received.

Description

This action enables/disables saving the source data from which the message was received. In case the message was received from HTTP traffic capture, the source data will be two files containing the HTTP request and HTTP response. If the message was received from an SMTP or POP3 protocol capture, the source data will be a file containing the original email in EML format.

By default, saving the source data for each message is disabled. If you enable it, the source data files will be attached to the message.

It should be remembered that enabling the preservation of the original data more than doubles the amount of saved data, so this option should be used only for debugging purposes, or when the availability of the original data is extremely necessary.

Format

```
<action name="save-raw-data" value="<true/false/1/0>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="save-raw-data".

Attribute "value":

The value="..." attribute indicates whether or not the action is active.

true or 1

Enable saving of the original data for the message.

false or 0

Disable saving the original data for a message (if it was previously enabled in the filter).

Example:

Enable saving of source data of messages received via HTTP protocol.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">
    <rule enabled="1">
      <comment>
        Enable saving source data for messages
        intercepted over the HTTP protocol.
      </comment>
      <match>
        <c name="protocol" value="http" />
      </match>
      <action name="save-raw-data" value="true" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.11. Action TRANSPORT

Set a delivery profile for the message to be delivered to if it is successfully processed by the ACCEPT filter.

Description

If such a transport profile has already been installed for a message, the action does not perform anything.

Several transport profiles can be specified for a message, and several "transport" actions are specified for this purpose.

In this way, you can deliver the results of the same message analysis to multiple consumer systems at the same time. For example, you can send metadata to the SIEM system and the message itself to eDiscovery and DLP.

If no transport profile has been applied to the message during filtering, the message will be delivered by the default profile if accepted.

Format

```
<action name="transport" value="<transport-profile-name>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="transport".

Attribute "value":

In the attribute value="..." specify the name of the transport profile to be installed.

Example:

Set the transport profiles "smtp-archive" and "smtp-archive-rezerve" for all messages. If a message is accepted by the filter, it will be delivered using both profiles.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">
  <comment>Message filter.</comment>

  <table name="main">

    <rule enabled="1">
      <comment>
        Set the "smtp-archive" and "smtp-archive-rezerve"
        transport profiles for all messages. If the message
        is accepted by the filter, it will be delivered by
        BOTH profiles.
      </comment>
      <action name="transport" value="smtp-archive" />
      <action name="transport" value="smtp-archive-rezerve" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.12. Action HEADER

Adds a custom X-Sensor header to object metadata... or changes the value of an existing X-Sensor... header.

Description

This action adds a custom X-Sensor header to the message...

If a header with such a name already exists in the message, its value is replaced by a newer one.

Format

```
<action name="header" headername="<UserHeader>" value="<user header value>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="header".

Attribute "headername":

In the headername="..." attribute, specify the name of the string label to be added.

Attribute "value":

The value="..." attribute specifies the header value.

Example:

Adds X-Sensor-UserHeader: user header value to the message.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is the comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Adds the following header to the message:
        X-Sensor-UserHeader: user header value.
      </comment>
      <action name="header"
        headername="UserHeader"
        value="user header value" />
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.3.2.13. Action HEADER_EX

Adds a custom header to the object metadata or changes the value of an existing header, except for From, To, Cc, Bcc, Subject, Date headers.

Description

This action adds a custom header to the message.

If a header with such a name (regardless of case) already exists in the message, its value is replaced by a newer one.

Header name cannot be:

From
To
Cc
Bcc
Subject
Date

Format

```
<action name="header_ex"  
  headername="<UserHeader>"  
  value="<user header value>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="header_ex".

Attribute "headername":

In the headername="..." attribute, specify the name of the string label to be added/replaced.

Attribute "value":

The value="..." attribute specifies the header value.

Example:

Adds "X-SomethingElse-UserHeaderEx: user header value" heading to the message.

```
<?xml version="1.0" encoding="utf-8"?>  
<filter name="TEST" version="1.0">  
  <comment>This is the comment for the filter.</comment>  
  <table name="main">  
  
    <rule enabled="true">  
      <comment>  
        Adds the following header to the message:  
        X-SomethingElse-UserHeaderEx: user header value.  
      </comment>  
      <action name="header_ex"  
        headername="X-SomethingElse-UserHeaderEx"  
        value="user header value" />  
    </rule>  
  
    <rule enabled="true">  
      <match><c name="all"/></match>  
      <action name="accept" />  
    </rule>  
  
  </table>  
</filter>
```

4.3.1.3.2.14. Action LOG

Logging according to the assigned channel and receiver, used for filter debugging purposes.

Description

This action sends to the specified receiver (file, syslog server) a line of text, the value of the message tags, the value of the message labels or the metadata of the message.

Log string format:

```
[<timestamp>] <value>
```

You can send to the log:

- The line specified in the attribute value="..."
- List of labels and their values specified in field="#labels" attribute
- List of tags and their values specified in field="#tags" attribute
- The value of the message metadata header. To do this, specify the name of the header in the attribute field="X-Sensor-..."

The LOG action is useful for debugging filters. By placing it to the right filter rules, you can get a detailed report on how messages are processed and how the value of metadata, tags and labels changes.

Remember that this LOG action is used for filter debugging purposes when filtering messages. Continuous use of it can degrade the overall performance of EtherSensor if there is heavy writes to disk.

Format

```
<action name="log" dst="<log-destination>" value="<user string value>" />  
<action name="log" dst="<log-destination>" field="<field type>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="log".

Attribute "dst":

The dst="..." attribute specifies the logging receiver. It can be:

dst="syslog://<syslog-server-ip:port>"

Sends a message to a syslog server via the UDP protocol (RFC-3164).

dst="channel://<channel-name>"

Sends a message to a channel pre-configured in the service EtherSensor Watcher.

dst="file://<full-file-path>"

Saves the message as a file.

Attribute "field":

In the attribute field="..." specify the type of field of the message to be sent to the log. Possible values:

#labels

Output a list of message labels and their values

#tags

Output a list of message tags and their values

#from

FROM address list output

#to

TO address list output

#subject

Message subject output

X-Sensor-...

Output the message metadata header value.

Attribute "value":

The value="..." attribute specifies the line of the message to be sent to the log.

Example:

Action:

```
<action name="log" dst="file://d:\file\path.log.txt"
  value="user string value" />
```

Saves the following line to the log file d: \ file \ path.log.txt:

```
"[<timestamp>] user string value".
```

Action:

```
<action name="log" dst="file://d:\file\path.log.txt" field="#tags" />
```

Saves the following line to the log file d: \ file \ path.log.txt:

```
"[<timestamp>] Tags:"
"          <TAG = value>"
"          <TAG = value>"
"          <TAG = value>"
```

Each tag is saved on a separate line.

Action:

```
<action name="log" dst="channel://debug.log.txt" field="#labels" />
```

Sends the following line to the debug.log.txt channel:

```
"[<timestamp>] Labels:"  
"           <LABEL = "value">"  
"           <LABEL = "value">"  
"           <LABEL = "value">"
```

Each label is stored on a separate line.

Action:

```
<action name="log"  
  dst="syslog://192.168.0.1:514"  
  field="X-Sensor-Src-Address" />
```

Sends the following line to the syslog server with the address 192.168.0.1:514:

```
"[<timestamp>] X-Sensor-Src-Address: <metadata header value>"
```

Example:

Send message data to syslog server.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is the comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Sending message details to syslog.
      </comment>
      <action name="log"
        dst="syslog://192.168.0.1:514"
        value="Message info dump:" />
      <action name="log"
        dst="syslog://192.168.0.1:514"
        field="X-Sensor-Src-Address" />
      <action name="log"
        dst="syslog://192.168.0.1:514"
        field="X-Sensor-Dst-Address" />
      <action name="log" dst="syslog://192.168.0.1:514"
        field="#labels" />
      <action name="log"
        dst="syslog://192.168.0.1:514"
        field="#tags" />
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.1.4. Brief rules for writing filters

1. The filter must always contain the "main" table, from which the filter execution begins.
2. The table cannot be empty. Each filter table must end with a rule with the condition "any message" and one of the actions: ACCEPT, DROP or RETURN. This rule must always be enabled.

Example:

```
<rule name="end" enabled="1">
  <match>
    <c name="all"/>
  </match>
  <action name="drop" />
</rule>
```

or

```
<rule enabled="1">
  <action name="accept" />
</rule>
```

3. The RETURN action can be used in any table except main.
4. JUMP jumps forming cyclic references (explicit or implicit) are prohibited.
5. JUMP jumps to the "main" table are prohibited in order to avoid cyclic references.

4.3.1.5. Tips

This section describes techniques to help you debug and test your filters.

Time stamps

Time stamps (`<action name="datetime" value="label-name">`) can be used not only at the end and beginning of message processing, but also in each rule - then you will know when it was processed relative to the beginning of filtering the whole message. To do this, you can specify the "datetime" action just before the termination action.

Example:

```
<rule name="rule2" enabled="1">
  <match>
    <c name="attach-exist"/>
  </match>
  <action name="datetime" value="rule2-end" />
  <action name="accept" />
</rule>
```

In this example, after the message is processed by a rule, the message will have a "rule2-end" tag indicating the time the message was accepted by that rule for further processing (ACCEPT).

Some actions may take quite a long time (e.g. DNS name resolution, message verification by external antivirus, etc.). Since all actions are performed sequentially, you can measure the execution time of this long action by adding the "datetime" action to it.

Example:

```
<rule name="rule2" enabled="1">
  <match>
    <c name="attach-exist"/>
  </match>
  <action name="datetime" value="rule2-dns-start" />
  <action name="dns" />
  <action name="datetime" value="rule2-dns-end" />
</rule>
```

In this example, after processing the message with a rule, you can compare the value of the "rule2-dns-start" and "rule2-dns-end" tags and find out the time it took the DNS operation.

Rule tags

You can use the "tag" action (setting a tag in a message) in all rules, and set the tag name the same as the rule name. Then, after the message passes through the filter, it will contain the history of message processing by rules. This can be useful for testing filters and tracking the "paths" of objects being processed when the filter is complex and contains many rules and tables.

Example:

```
<rule name="rule-attach" enabled="1">
  <match>
    <c name="attach-exist" />
  </match>
  <action name="tag" value="rule-attach" />
</rule>

<rule name="rule-bad-words" enabled="1">
  <match>
    <c name="text" op="all" value="tel, respect" />
  </match>
  <action name="tag" value=" rule-bad-words" />
</rule>

<rule name="rule-only-for-dns" enabled="1">
  <action name="dns" />
  <action name="tag" value=" rule-only-for-dns" />
</rule>

<rule name="accept-all" enabled="1">
  <action name="tag" value="accept-all" />
  <action name="accept" />
</rule>
```

You can also do with timestamps, in this case it will be visible not only through which rules the message passed, but also at what time.

"From simple to complex."

Some actions can take quite a long time (DNS name resolution, message verification by external antivirus, etc.). You should place such actions as close as possible to the end of message processing so that there are as few situations as possible when a long operation is performed and then the DROP action is applied to the message because the FROM field did not pass the check. It's best to first check the FROM field and sift out some of the messages on it, and only then perform DNS or other lengthy operations like antivirus checks on it.

4.3.2. Pre-filtering HTTP requests

The HTTP request pre-filtering mechanism solves the following tasks:

1. Filtering out unnecessary HTTP traffic to reduce the load on EtherSensor at the message analysis stage (ACCEPT¹⁶³ and DROP¹⁶⁴ actions).
2. Accumulation of information about possible new trends in HTTP traffic to provide support service and/or developer Microolap EtherSensor (COPY¹⁶⁷ action).
3. Logging information about HTTP requests in SQUID-ACCESS-LOG format (ACCESS-LOG¹⁶⁸ action).
4. Manipulations with tags and labels at the preprocessing stage in order to use the information accumulated in them for message analysis (TAG¹⁶⁹ and LABEL¹⁷² actions).

The HTTP filter configuration is contained in an XML file stored in the [INSTALLDIR]\config\filter\http subdirectory.

To edit a filter, use any external text/XML editor or use the EtherSensor filter editor built into the management console, specially designed for this purpose:

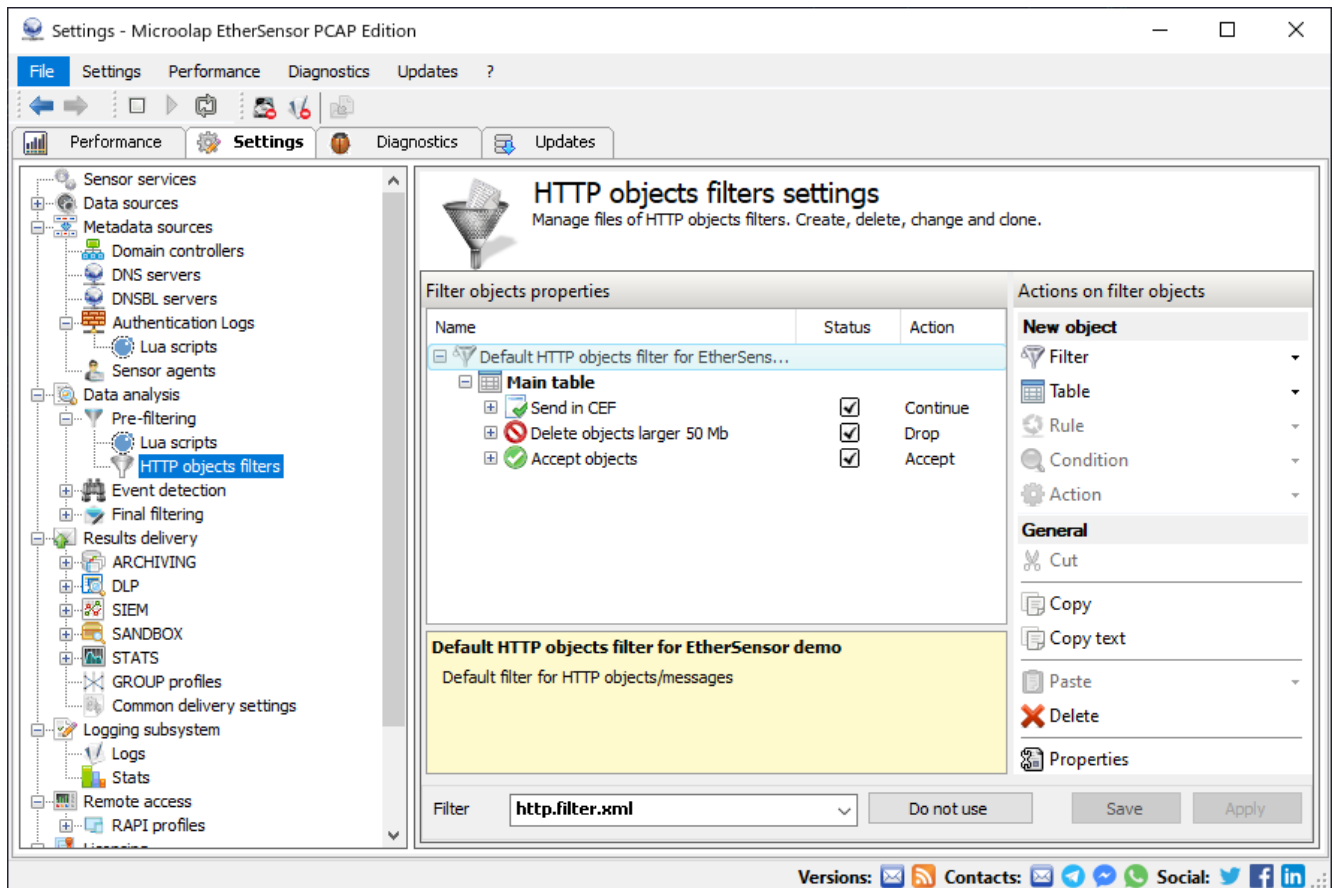


Fig.39. Editing the HTTP filter.

4.3.2.1. Conditions

Below are the conditions used in HTTP request pre-filtration rules.

4.3.2.1.1. Condition ALL, *

A special condition to which all filtered objects satisfy.

Description

This condition is true for any objects, it is implicitly implied if the <match> ... </match> criterion is empty or is absent in the rule (the rule contains only "action" tags).

Format

```
<c name="all" />
<c name="*" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="all" or name="*".

Example:

The rule accepts all messages for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>

  <table name="main">

    <rule enabled="1">
      <comment>
        This rule accepts all messages for further processing.
      </comment>
      <match>
        <c name="all" />
      </match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

Example (implicit all condition):

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.2.1.2. Condition METHOD

The condition checks the HTTP request method.

Description

This condition checks the name of the HTTP request method. If the method name matches, the condition is met.

Format

```
<c name="method" value="<HTTP method>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="method".

Attribute "value":

In the attribute value="..." specify a method name for comparison.

Example:

The rule stops processing all GET requests.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>

  <table name="main">
    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>

    <rule enabled="1">
      <comment>
        The rule stops further processing of GET requests.
      </comment>
      <match>
        <c name="method" value="GET"/>
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.2.1.3. Condition IP

Check the IP addresses of the client or server for range entry or subnet entry.

Description

This condition checks the IP addresses of the client or server for range or subnet membership.

Tips:

1. Unwanted traffic should be cut off as soon as possible. Performance EtherSensor depends on it.
2. All traffic from a certain IP or range is best cut off in the IP filter of the EtherSensor EtherCAP service.
3. Certain HTTP traffic from some IP (if it is possible to define such criteria) is best cut off in the HTTP filter.
4. Certain messages from a certain IP address must be processed in the message filter.

Format

```
<c name="ip" address="<address type>" value="<ip-range>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="ip".

Attribute "address":

In the attribute address="..." specify the type of address to be checked. Possible values:

src or client

Check source address

dst or server

Check the destination address.

Attribute "value":

In the attribute value="..." specify a value for comparison. Possible values:

ipaddress

Checks the IP address for equality. For example, value="192.168.0.10".

ip1-ip2

Checks for an IP address in the range. For example, value = "192.168.0.1-192.168.0.10"

ip/netmask

Checks if the IP address belongs to the specified subnet. For example, value = "192.168.0.1/255.255.255.0"

ip/netmaskbits

Checks if the IP address belongs to the specified subnet. For example, value = "192.168.0.1/24".

Example:

Ignore messages from the client's machine 192.168.0.15.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>

  <table name="main">

    <rule enabled="1">
      <comment>
        Discard messages from 192.168.0.15.
      </comment>
      <match>
        <c name="ip" address="client" value="192.168.0.15" />
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.2.1.4. Condition REQ-SIZE, RESP-SIZE, SIZE

The condition checks the full size (including headers) of the HTTP object.

Description

The size group conditions check the full (including headers) size of the HTTP request/response.

req-size

The condition is met if the size of the HTTP request matches the condition

resp-size

The condition is met if the size of the HTTP response matches the condition

size

The condition is met if the size of the HTTP request or HTTP response corresponds to the condition.

Format

```
<c name="req-size" op="<operation>" value="<compare pattern>" />
<c name="resp-size" op="<operation>" value="<compare pattern>" />
<c name="size" op="<operation>" value="<compare pattern>" />
```

Attribute "name":

In the attribute "name" specify the name of the condition: name="req-size" or name="resp-size" or name="size".

Attribute "value":

In the attribute value="..." specify the number with which the message size is compared:

<number> or <number>B

Indicates the size in bytes

<number>K

Indicates the size in kilobytes

<number>M

Indicates the size in megabytes.

<number>G

Indicates the size in gigabytes.

Attribute "op":

In the attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the size is EQUAL to the specified number

ne or != or <>

The condition is met if the size is NOT EQUAL to the specified number.

lt or <

The condition is met if the size is LESS than the specified number

gt or >

The condition is met if the size is GREATER than the specified number

le or <=

The condition is met if the size is LESS OR EQUAL than the specified number.

ge or >=

The condition is met if the size is GREATER OR EQUAL than the specified number.

Example:

The rule stops processing all HTTP request objects larger than 100KB or HTTP response objects larger than 1MB.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        The rule stops processing HTTP objects with the request size
        over 100KB or the response size over 1MB.
      </comment>
      <match>
        <or>
          <c name="req-size" op=">" value="100K"/>
          <c name="resp-size" op="gt" value="1M"/>
        </or>
      </match>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.2.1.5. Condition REQ-HEADER, RESP-HEADER

Checks the value of one of the HTTP request or response headers.

Description

This condition checks the value of the HTTP request header for substring in the string or for the wildcard or regexp pattern.

Format

```
<c name="req-header" headername="..." op="..." value="..." />
```

or

```
<c name="resp-header" headername="..." op="..." value="..." />
```

Attribute "name":

In the attribute "name" specify the name of the condition: name="req-header" or name="resp-header".

req-header

Checks HTTP request headers

resp-header

Checks HTTP response headers

Attribute "headername":

In the attribute headername="..." specify the name of the header to be checked.

Attribute "headername":

The string with which the value or template is compared for checking is specified in the value="..." attribute.

Attribute "op":

In the attribute op="..." indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the value of the field CONTENTS the specified value

ne or != or <>

The condition is met if the field value DOES NOT CONTAIN the specified value

wc or wildcard

The condition is met if the field value matches the specified wildcard template

regex or regexp

The condition is met if the value of the field corresponds to the specified regexp pattern

The following operations are available for Content-Length header:

eq or = or ==

The condition is met if the value of the field CONTENTS the specified value

ne or != or <>

The condition is met if the field value DOES NOT CONTAIN the specified value

lt or <

The condition is met if the size is LESS than the specified number

gt or >

The condition is met if the size is GREATER than the specified number

le or <=

The condition is met if the size is LESS OR EQUAL than the specified number.

ge or >=

The condition is met if the size is GREATER OR EQUAL than the specified number.

These operations are performed with the header value as NUMBER, not as a string.

Attribute "value":

In the attribute value="..." specify the value to be checked (string, wildcard or regexp template).

For the Content-Length header it is allowed to specify only a numerical value for comparison.

The number against which the size is compared is indicated in the form:

<number> or <number>B

Indicates the size in bytes

<number>K

Indicates the size in kilobytes

<number>M

Indicates the size in megabytes.

<number>G

Indicates the size in gigabytes.

Example:

Ignore requests with Content-Length over 100K. Accept requests for win.mail.ru and *.yandex.ru.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        Ignore requests where Content-Length is more than 100K.
      </comment>
      <match>
        <c name="req-header"
          headername="Content-Length"
          op=">" value="100K" />
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <comment>
        Accept requests to win.mail.ru and *.yandex.ru.
      </comment>
      <match>
        <or>
          <c name="req-header"
            headername="Host" op="eq"
            value="win.mail.ru" />
          <c name="req-header"
            headername="Host"
            op="wc"
            value="*.yandex.ru" />
        </or>
      </match>
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.2.1.6. Condition URL

Checks the value of the HTTP request URL.

Description

This condition checks the value of the HTTP request URL for substring in the string or for the wildcard or regexp template.

Keep in mind that the full request URL is checked, i.e. as `http://www.server.com/script-or-page-path.htm`.

If you only want to check the hostname, you should use the condition "HEADER". This will increase verification speed and save resources.

Format

```
<c name="url" op="..." value="..." />
```

Attribute "name":

In the "name" attribute specify the name of the condition: `name="url"`.

Attribute "op":

In the attribute `op="..."` indicates the type of comparison operation and can accept values:

eq or = or ==

The condition is met if the value of the field **CONTAINS** the specified value

ne or != or <>

The condition is met if the field value **DOES NOT CONTAIN** the specified value

wc or wildcard

The condition is met if the field value matches the specified wildcard template

regex or regexp

The condition is met if the value of the field corresponds to the specified regexp pattern

Attribute "value":

In the attribute `value="..."` specify the value to be checked (string, wildcard or regexp template).

Example:

```
<c name="url" op="eq" value="game" />
```

This condition is fulfilled if the URL contains the substring "game".

```
<c name="url" op="wc" value="http://*.mail.ru/*send*" />
```

The condition is met if the URL matches the wildcard template "http://*.mail.ru/*send*".

```
<c name="url" op="re" value="satan|shopping|dating|movies|hexogen" />
```

This condition is met if the URL matches the regexp template "satan|shopping|dating|movies|hexogen".

Example:

Detect requests with bad words (satan|shopping|dating|movies|hexogen), tag shopping. Ignore requests that are tagged with the shopping tag.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        Detect requests possibly related to bad content.
      </comment>
      <match>
        <c name="url"
          op="re"
          value="satan|shopping|dating|movies|hexogen" />
      </match>
      <action name="tag" value="shopping"/>
    </rule>

    <rule enabled="1">
      <comment>
        Ignore requests tagged as "shopping".
      </comment>
      <match>
        <c name="tag" tag="shopping"/>
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.2.1.7. Condition TAG

Checks whether the tag is set and the counter value (see Action TAG¹⁶⁹).

Description

This condition checks the activity of the specified tag for the object as well as its level.

Format

```
<c name="tag" tag="<tag name>" op="<operation>" value="<compare value>" />
```

Attribute "name":

In the "name" attribute specify the name of the condition: name="tag".

Attribute "tag":

In the tag="..." attribute, enter the name of the tag to be checked.

Attribute "op":

In the attribute op="..." specify the type of comparison operation. Possible values:

eq or = or ==

The condition is met if the value is EQUAL to the specified number

ne or != or <>

The condition is met if the value is NOT EQUAL to the specified number.

lt or <

The condition is met if the value is LESS than the specified number

gt or >

The condition is met if the value is GREATER than the specified number

le or <=

The condition is met if the value is LESS OR EQUAL to the specified number.

ge or >=

The condition is met if the value is GREATER OR EQUAL to the specified number.

exist

The condition is met if the tag exists (already set earlier for this HTTP object).

By default (if the attribute "op" is missing) the value "exist" is accepted. It is not necessary to specify the "value" attribute for the operation "exist".

Attribute "value":

In the attribute value="..." specify the value to be checked if the tag counter value is checked.

Example:

```
<c name="tag" tag="SPAM" op="exist" />
```

or

```
<c name="tag" tag="SPAM" />
```

The condition is met if the object has the SPAM tag.

```
<c name="tag" tag="SPAM" op=">=" value="3" />
```

The condition is met if the object has a SPAM tag and its counter value is greater than or equal to 3.

Example:

Detect requests with bad words (satan|shopping|dating|movies|hexogen), tag shopping. Ignore requests that are tagged with the shopping tag.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Detect requests possibly related to bad things.
      </comment>
      <match>
        <c name="url"
          op="re"
          value="satan|shopping|dating|movies|hexogen" />
      </match>
      <action name="tag" value="shopping"/>
    </rule>

    <rule enabled="true">
      <comment>
        Ignore a request tagged as "shopping".
      </comment>
      <match>
        <c name="tag" tag="shopping"/>
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="1">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.2.2. Actions

Below is a list of actions performed by HTTP pre-filtration rules.

4.3.2.2.1. Action ACCEPT

Accept the HTTP object for further processing.

Description

ACCEPT stops the filters with the current HTTP object and passes it for further processing by the detectors.

Format

```
<action name="accept" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="accept".

Example:

All messages that have reached this rule should be accepted for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <match ...> ... </match>
      <action ...> ... </action>
    </rule>

    <rule enabled="1">
      <comment>
        All messages that reach this rule are accepted for
        further processing.
      </comment>
      <action name="accept" />
    </rule>
  </table>
</filter>
```

4.3.2.2.2. Action DROP

Ignore the HTTP object without further processing.

Description

This action stops the processing of the current HTTP object in EtherSensor and informs it that the message should be ignored ("forgotten") and all accumulated data about it should be destroyed.

Format

```
<action name="drop" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="drop".

Example:

All messages that reach this rule should be ignored, their accumulated metadata destroyed.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        All messages that reach this rule will be discarded.
      </comment>
      <action name="drop" />
    </rule>
  </table>
</filter>
```

4.3.2.2.3. Action JUMP

Continue processing the HTTP object in another table.

Description

This action continues processing the HTTP object in another table.

Format

```
<action name="jump" value="<table name>" />
```

Attribute "name":

In the attribute "name" specify the name of the action: name="jump".

Attribute "value":

In the attribute value="..." specify the name of the table to which you want to go for further processing of the HTTP object.

It should be remembered that jump to the table "main" is prohibited to avoid an endless loop. Jumps that may lead to an explicit or implicit cycling are also prohibited.

Example:

GET requests are passed to the get-process table for processing, processed in it (ACCEPT for win.mail.ru requests), then returned to the "main" table for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">
    <rule enabled="1">
      <comment>Processing of GET requests is passed to the "get-process" table.</comment>
      <match>
        <c name="method" value="GET"/>
      </match>
      <action name="jump" value="get-process"/>
    </rule>
    <rule enabled="1">
      <action name="drop" />
    </rule>
  </table>

  <table name="get-process">
    <comment>GET requests are processed in the table get-process.</comment>
    <rule enabled="1">
      <comment>ACCEPT for requests for win.mail.ru.</comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
      </match>
      <action name="accept" />
    </rule>
    <rule enabled="1">
      <comment>Return to "main" table for further processing.</comment>
      <action name="return" />
    </rule>
  </table>
</filter>
```

4.3.2.2.4. Action RETURN

Return to the previous (called) table and continue executing it with the next rule.

Description

This action returns the processing of an HTTP object to the previous (called) table and continues with the next rule.

Format

```
<action name="return" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="return".

The action cannot be applied in the table "main".

Example:

GET requests are passed to the get-process table for processing, then GET requests are processed in the get-process table, then ACCEPT for win.mail.ru requests, then return to the "main" table for further processing.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <comment>HTTP filter.</comment>
  <table name="main">

    <rule enabled="1">
      <comment>
        Processing of GET requests is passed to the "get-process" table.
      </comment>
      <action name="jump" value="get-process"/>
    </rule>

    <rule enabled="1">
      <action name="drop" />
    </rule>
  </table>

  <table name="get-process">
    <comment>
      GET requests are processed in the table.
    </comment>

    <rule enabled="1">
      <comment>
        ACCEPT for requests for win.mail.ru.
      </comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
      </match>
      <action name="accept" />
    </rule>

    <rule enabled="1">
      <comment>
        Return to "main" table for further processing.
      </comment>
      <action name="return" />
    </rule>
  </table>
</filter>
```

4.3.2.2.5. Action COPY

Copy the current HTTP object being processed (request and response, if any) and its metadata to the specified directory.

Description

This action copies the current HTTP object being processed (request and response, if any) and its metadata to the specified directory.

Format

```
<action name="copy" value="<folder path>"/>
```

Attribute "name":

In the attribute "name" specify the name of the action: name="copy".

Attribute "value":

In the attribute value="..." specify the path to the directory, to which data should be copied.

If the directory does not exist, it will be created.

Example:

Copy win.mail.ru requests to the folder d:\data_from_mail.ru.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is a comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Copy requests for win.mail.ru to d:\data_from_mail.ru.
      </comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
      </match>
      <action name="copy" value="d:\data_from_mail.ru"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.2.2.6. Action ACCESS-LOG

Save information about the current HTTP object in the log channel in SQUID-ACCESS-LOG format.

Description

The access-log action saves information about the current HTTP object in the log channel in SQUID-ACCESS-LOG format.

Format

```
<action name="access-log" value="<log-channel-name>" />
```

Attribute "name":

In the "name" attribute specify the name of the action: name="access-log".

Attribute "value":

In the attribute value="..." specify the name of the log channel. This channel must be preset in the service EtherSensor Watcher.

Example:

Log all requests to the all-log-channel, and requests to win.mail.ru also to the mail-ru-log-channel.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is a comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Log requests for win.mail.ru to mail-ru-log-channel.
      </comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
      </match>
      <action name="access-log" value="mail-ru-log-channel"/>
    </rule>

    <rule enabled="true">
      <comment>
        Log all requests to all-log-channel.
      </comment>
      <action name="access-log" value="all-log-channel"/>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.2.2.7. Action TAG

Adds a tag (numeric label) to the metadata of the object.

Description

This action sets a tag (numeric label) in the metadata of the object. There can be several tags. In this case they are listed via comma ',' or semicolon ';'. If the same tag is set for an object several times, the "level" (its numeric value) is increased. Each tag has an internal counter that shows how many times it was set for that object.

The existence of a tag, as well as the value of its counter (how many times it was set for the current object) can then be checked in filter conditions for decision making.

By default, the counter value is increased by 1 when the tag is set. If the tag counter is to be increased by more than 1, you can set the value by which the counter is to be increased after the tag name in parentheses: "TAG(...)". For example, SPAM(3) - increases the counter value for a SPAM tag by 3, and SPAM(1) is simply equal to SPAM. This may be necessary in cases where the conditions setting the same tag have different meanings / importance / priority.

The value for changing the counter may be negative. SPAM(-3) - reduces the counter value for the SPAM tag by 3.

Tags set in this action will later be available in the message filter conditions if a message is extracted from this HTTP object.

Format

```
<action name="tag" value="<tag list>" />
```

or

```
<action name="tag" > tag list </action>
```

Attribute "name":

In the "name" attribute specify the name of the action: name="tag".

Attribute "value":

The value="..." attribute lists the tag names.

Tag names can also be listed in the <action> tag itself.

Example:

```
<action name="tag" value="SPAM" />
```

Sets a tag named "SPAM".

```
<action name="tag" value="SPAM(3)" />
```

Sets a tag named "SPAM" and increases its counter by 3.

```
<action name="tag" value="SPAM, shopping" />
```

Sets tags with the names "SPAM" and "shopping".

```
<action name="tag" value="SPAM(3), shopping(2)" />
```

Sets tags with the names "SPAM" and "shopping" and increases their counters by 3 and 2 respectively.

```
<action name="tag">SPAM, shopping</action>
```

Also sets tags with the names "SPAM" and "shopping".

```
<action name="tag" value="SPAM, shopping">VIP-OFFICE</action>
```

Sets tags with names "SPAM", "shopping", "VIP-OFFICE".

Example:

Tag RUS_MAIL to mark requests for popular Russian mail services.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>Filter comment.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Mark requests to popular Russian mail services
        with RUS_MAIL tag.
      </comment>
      <match>
        <c name="req-header"
          headername="Host"
          op="eq"
          value="win.mail.ru" />
        <c name="req-header"
          headername="Host"
          op="eq"
          value="mail.yandex.ru" />
        <c name="req-header"
          headername="Host"
          op="eq"
          value="mail.rambler.ru" />
      </match>
      <action name="tag" value="RUS_MAIL"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.2.2.8. Action LABEL

Adds a string label to the object's metadata.

Description

This action sets a string label in the metadata of the current HTTP object being processed. If this label has already been set for an HTTP object, it is replaced by a newer label. It is used to add descriptions to HTTP objects.

The set string labels will then be available in the message filter (if a message has been extracted from this HTTP object).

Format

```
<action name="label" label="<label name>" value="<label value>" />
```

or:

```
<action name="label" label="<label name>" > label value </action>
```

Attribute "name":

In the attribute "name" specify the name of the action: name="label".

Attribute "label":

In the label="..." attribute, specify the name of the string label to be set.

Attribute "value":

In the attribute value="..." specify a string value for the label.

The value can also be listed in the <action> tag itself.

Example:

```
<action name="label"
  label="VIRUS-DESCR"
  value="Win.32.BlackHorse.trojan.virus -
      mail worm, extremely dangerous!!!" />
```

This action sets a string label with the name "VIRUS-DESCR" for the message and writes into it the line "Win.32.BlackHorse.trojan.virus - mail worm, extremely dangerous!!!".

Example:

Mark requests for popular Russian postal services with the CONTENT-DESCR label.

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="TEST" version="1.0">
  <comment>This is a comment for the filter.</comment>
  <table name="main">

    <rule enabled="true">
      <comment>
        Mark requests for popular Russian mail services
        with CONTENT-DESCR label.
      </comment>
      <match>
        <or>
          <c name="req-header"
            headername="Host"
            op="eq"
            value="win.mail.ru" />
          <c name="req-header"
            headername="Host"
            op="eq"
            value="mail.yandex.ru" />
          <c name="req-header"
            headername="Host"
            op="eq"
            value="mail.rambler.ru" />
        </or>
      </match>
      <action name="label"
        label="CONTENT-DESCR"
        value="Russian mail services"/>
    </rule>

    <rule enabled="true">
      <match><c name="all"/></match>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

4.3.3. Examples of applying filters

Examples of applying filters to process capture results are described below.

4.3.3.1. Adding hostname

Task

It is necessary that the reconstructed messages, together with the IP addresses, contain the sender and receiver domain host names.

Description of solution logic

To do this, EtherSensor must resolve domain names through the DNS service.

Do the following:

1. In the EtherSensor Analyser service configuration, specify which DNS servers are to be used for name resolution.

2. Create a message filter that contains a rule with an action that means that DNS resolution of hostnames must be performed for the currently processed message.

Solution

1. Setting the parameters of DNS servers for name resolution in the EtherSensor Analyser service configuration.

Setup can be done in the management console or in a configuration file.

In the EtherSensor Analyser service configuration file:

The root tag <AnalyserConfig>, then the sub tag <Filter> of filter settings (message filter is enabled), then the sub tag <Dns> - DNS servers settings for resolving the names in the message filter.

```
<?xml version="1.0" encoding="utf-8"?>
<AnalyserConfig version="3.2">

<!-- specify other settings for the service here -->

  <Filter enabled="true" filename="msg_filter.xml">

<!-- specify other filter settings here -->

    <Dns>
      <AttemptsCount>3</AttemptsCount>
      <TtlForUnknown>3600</TtlForUnknown>
      <MinTtl>300</MinTtl>
      <MaxTtl>604800</MaxTtl>
      <Server ipaddress="127.0.0.1" port="53" />
    </Dns>

  </Filter>
</AnalyserConfig>
```

In this case it is assumed that the DNS server is located at 127.0.0.1:53.

2. Message filter configuration

For example, the file msg_filter.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="DNS resolve" version="1.0">

  <table name="main">

    <rule enabled="true">
      <comment>
        Resolve the sender and the recipient host names
        for the message.
      </comment>
      <action name="dns" address="both"/>
    </rule>

    <rule enabled="true">
      <comment>
        Accept all message that reached this point.
      </comment>
      <action name="accept" />
    </rule>

  </table>
</filter>
```

Comments and general recommendations

1. After performing the DNS action, the X-Sensor-Src-Host, X-Sensor-Dst-Host headers with the values of domain names or with the value "not resolved" will be added to the message metadata if it was not possible to get some of the domain names.
2. For faster resolution of DNS names in the configuration of the EtherSensor Analyser service, it is desirable to specify the fastest possible DNS servers. These can be either the servers of the Internet provider or your own DNS servers.
3. DNS action (DNS name resolution) is a relatively time-consuming operation, therefore try to perform it in the filter only for those messages for which it is necessary.
4. Try to apply the DNS action at the end of the filter and directly in the place where you need the results of its work.

For example, if you want the sender's and recipient's host addresses to just contain domain names in the message, it is enough to apply the DNS action right before the end of message filtering and the ACCEPT action. There is no need to do this at the beginning of the filter, since during the filtering some of the messages can be rejected by the DROP action and for these messages the DNS action (if it is set at the beginning of the filter) will produce an extra load on EtherSensor.

If the obtained DNS names of the sender and the recipient are required to be further used in the filter conditions to filter messages by host names, then DNS name resolution should be done in the rule that is located immediately before the rule in which the received DNS names will be used.

Troubleshooting

If there is no DNS resolution for sender or recipient IP addresses:

1. Check the availability of the DNS server directly from the sensor machine using the ping utility (eg ping <dns server ip address>) and telnet (eg telnet <dns server ip address>).
2. Make sure the DNS server being used can do name resolution for the sensor machine. This can be done with the nslookup utility.
 - At the command prompt, run the nslookup utility.
 - In the running utility, run the server <IP-address-dns-server> command to set the DNS server through which name resolution will be performed.
 - Enter the IP address whose name cannot be resolved.

If nslookup can resolve the hostname, then you need to check the filtering progress:

1. Make sure that the rule in the filter that contains the DNS action is enabled and that the conditions of that rule match the messages.
2. Make sure that the rule in the filter containing the DNS action will be executed at all, that is, there is no situation when messages are accepted by the rules above and the filtering process does not reach this rule (for example, above this rule there are no rules with the ACCEPT action or DROP).
3. Make sure that when starting EtherSensor there are no messages about incorrect loading of the message filter and errors in the logs of the service EtherSensor Analyser.

If all else fails, please send to the Microolap EtherSensor developer the following:

- Several examples of messages for which name resolution does not occur.
- The message filter used.
- Diagnostic report of EtherSensor generated by the utility sensor_console.exe from Microolap EtherSensor delivery.
- Your comments and observations on the above actions.

4.3.3.2. Filtering by hosts

Task

In case of requests to <*baender*.com, benderlog.ru, banderlog.biz> it is necessary to stop processing such a message and delete the accumulated data about it.

Description of solution logic

There are two possible solutions to the problem:

1. Completely ignore traffic to hosts with the specified names in the HTTP protocol filter. To do this, you should write a filtering condition by the HTTP request "Host" header.

2. In the message filter, delete messages sent to the specified hosts. To do this, you need to resolve DNS names and write the "hostname" filtering condition.

Option 1 is preferable because it allows you to filter out unnecessary data at an earlier stage, which significantly reduces the load on EtherSensor.

Solution

1. Option 1 - ignore traffic at an early stage using the HTTP protocol filter.

For example, the HTTP protocol filter file may look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">
  <table name="main">

    <rule enabled="true">
      <match>
        <or>
          <c name="req-header"
            headername="Host"
            op="wc"
            value="*baender*.com*" />
          <c name="req-header"
            headername="Host"
            op="eq"
            value="benderlog.ru" />
          <c name="req-header"
            headername="Host"
            op="eq"
            value="banderlog.biz" />
        </or>
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="true">
      <action name="accept" />
    </rule>

  </table>
</filter>
```

For a detailed description of the "req-header" filter condition, see REQ-HEADER, RESP-HEADER ⁽¹⁵⁷⁾.

2. Option 2 - delete messages in the message filter.

You should enable DNS name resolution, the message filter file may look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">

  <table name="main">

    <rule enabled="1">
      <match>
        <or>
          <c name="hostname"
            address="server"
            op="wc"
            value="*baender*.com*" />
          <c name="hostname"
            address="server"
            op="eq"
            value="benderlog.ru" />
          <c name="hostname"
            address="server"
            op="eq"
            value="banderlog.biz" />
        </or>
      </match>
      <action name="drop"/>
    </rule>

    <rule enabled="1">
      <action name="accept" />
    </rule>

  </table>
</filter>
```

For a detailed description of the "hostname" filter condition, see Condition HOSTNAME¹¹².

Comments and general recommendations

wildcard1. In Option 1, note that the hostname check from the "Host" header by wildcard mask "*baender*.com*" contains "*" at the beginning and at the end of the mask. This is necessary at the beginning of the mask, because the top-level domains (e.g. www.baender.com or 123.baender.com) may be at the beginning of the hostname. At the end of the mask, this is necessary because at the end of the hostname, the destination port may be specified in the "Host" header (e.g. baender.com:80). The equality check (an operation in the "eq" condition) simply involves finding a substring in the string. Therefore "eq" with the value "benderlog.ru" will be triggered for both www.benderlog.ru and benderlog.ru:80.

2. For variant 2 in the general case for the "hostname" condition to work it is necessary to perform DNS name resolution beforehand. However, in this case (we check only the destination host and only for the HTTP protocol) it is not necessary to do it, because the host value will be available from the "Host" header of the HTTP protocol.

4.3.3.3. Filtering by URL

Task

If the words <forum, phorum, post, submit> are present in the URL for the get method, set the mark "user reads forums" on the message.

Description of solution logic

Check the URL and HTTP request method using an HTTP protocol filter. To check the URL, use the "url" condition, to check the HTTP protocol method, use the "method" condition.

Solution

For example, the HTTP protocol filter file may look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">

  <table name="main">

    <rule enabled="true">
      <comment>
        Detect requests possibly related to forums.
      </comment>
      <match>
        <and>
          <c name="method"
            value="GET"/>
          <c name="url"
            op="re"
            value="http://.*/*.*(forum|phorum|post|submit).*" />
        </and>
      </match>
      <action name="tag" value="USER_READS_FORUMS"/>
    </rule>

    <rule enabled="true">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

For a detailed description of "url" and "method" filtering conditions, see **Condition URL**¹⁶⁰ and **Condition METHOD**¹⁵².

Comments and general recommendations

1. It should be remembered that at the stage of filtering the HTTP protocol there are no messages yet; the check for their presence in the traffic will be performed later. However, the tags and tags set for requests at this stage will be preserved in the message (if it is extracted from these requests). These labels and tags will be available for inspection in the message filter and later as the X-Sensor-Tags and X-Sensor-Labels headers.
2. It should be remembered that the "url" condition contains the full request URL, including the hostname (ie, in the form `http://www.mail.ru/mail/read.php?some=parameter¶m2`), this must be taken into account in the verification condition.

4.3.3.4. Filtering by HTTP + DNSBL

Task

If the message was received or transmitted by the GET method of HTTP protocol and the server address is in a dnsbl list (for example, dnsbl.sorbs.net), it is necessary to mark the message with the label "possible malware or proxy".

Description of solution logic

Check the HTTP request method using the HTTP protocol filter. To ensure that the server address is present in some DNSBL list, you should use the message filter. Thus, two filters will be used for the common task: the HTTP protocol filter and the message filter. The HTTP protocol filter will check the GET method and place a certain label on all GET requests (for example, "HTTP_GET").

This label will remain in the messages that will be extracted from these requests and will be available later in the message filter. In the message filter, for messages containing the "HTTP_GET" tag, the presence of the server address in the DNSBL list will be checked by performing the "dnsbl" action.

If the server address is included in the DNSBL list, then a label will be set on this message (for example, "DNSBL_EXIST"). Further, in the message filter, it will be necessary to check for the message already two labels - "HTTP_GET" and "DNSBL_EXIST" and set the "possible malware or proxy" label for messages with both labels.

Also, for the "dnsbl" action to work correctly, it is necessary to configure DNS servers for checking DNSBL in the EtherSensor Analyser service configuration.

Solution

1. Configuring DNS servers in the EtherSensor Analyser service configuration for DNSBL check.

Setup can be done in the management console or in a configuration file.

In the EtherSensor Analyser service configuration file:

The root tag <AnalyserConfig>, then the nested tag <RawHttpFilter> - enable the filter of the HTTP protocol. Filter settings tag <Filter> (enable message filter). Next, the <DnsBl> nested tag is the settings of DNS servers for checking addresses in DNSBL lists in the message filter.

```
<?xml version="1.0" encoding="utf-8"?>
<AnalyserConfig version="3.2">

<!-- other service settings -->
  <RawHttpFilter enabled="true" filename="http-filter.xml" />
  <Filter enabled="true" filename="msg_filter.xml">

<!-- other filter settings -->

  <DnsBl>
    <AttemptsCount>3</AttemptsCount>
    <TtlForUnknown>3600</TtlForUnknown>
    <MinTtl>300</MinTtl>
    <MaxTtl>604800</MaxTtl>
    <Server ipaddress="127.0.0.1" port="53" />
  </DnsBl>

</Filter>

</AnalyserConfig>
```

In this case it is assumed that the DNS server is located at 127.0.0.1:53.

2. The HTTP protocol filter file may look like this (http-filter.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">

  <table name="main">

    <rule enabled="true">
      <match>
        <c name="method" value="GET"/>
      </match>
      <action name="tag" value="HTTP_GET"/>
    </rule>

    <rule enabled="true">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

For a detailed description of the "method" filtering condition and the "tag" action, see Condition METHOD⁽¹⁵²⁾ and Action TAG⁽¹⁶⁹⁾.

3. The message filter file may look like this (msg_filter.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="Message filter" version="1.0">

  <table name="main">

    <rule enabled="true">
      <match>
        <c name="tag" tag="HTTP_GET"/>
      </match>
      <action name="dnsbl-rh"
        address="both"
        tag="DNSBL_EXIST"
        value="dnsbl.sorbs.net" />
    </rule>

    <rule enabled="true">
      <match>
        <and>
          <c name="tag" tag="HTTP_GET"/>
          <c name="tag" tag="DNSBL_EXIST"/>
        </and>
      </match>
      <action name="tag" value="MALWARE_OR_PROXY"/>
    </rule>

  </table>
</filter>
```

For a detailed description of the "tag" condition, the "tag" action and the "dnsbl" action, please refer to Condition TAG ⁽¹⁶¹⁾, Action TAG ⁽¹⁶⁹⁾ and Action DNSBL-LH, DNSBL-RH ⁽¹³⁷⁾.

Comments and general recommendations

2. For faster resolution of DNS names in the configuration of the EtherSensor Analyser service, it is desirable to specify the fastest possible DNS servers. These can be either the servers of the Internet provider or your own DNS servers.

2. The "dnsbl" action (DNS name resolution for DNSBL) is a relatively long operation (especially if multiple DNSBL services are specified), so try to do it in the filter only for those messages that need it.

3. Try to apply the "dnsbl" action at the end of the filter, where you need the results of its work.

4. Keep in mind that at the stage of HTTP protocol filtering there are no messages yet, check for their presence in the traffic will be performed later. However, the labels and tags set for the requests will be saved in the message at this stage (if it is extracted from these requests).

4.3.3.5. Filtering large HTTP objects

Task

Sometimes very large objects are transferred via HTTP (downloading files, watching movies online). With a large number of such objects transferred simultaneously, EtherSensor can consume a large amount of RAM. Such a situation may lead to general degradation of performance EtherSensor.

You should use filters to cut off large HTTP objects and remove them before starting analysis.

Description of solution logic

In order to remove large HTTP objects before they are completely analyzed (and uploaded to the whole memory for this purpose) you need to apply an HTTP filter. To check the size of an HTTP request or response, you should use the "size" condition, which checks both the request and response size.

Solution

For example, the HTTP protocol filter file may look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<filter name="HTTP filter" version="1.0">

  <table name="main">

    <rule enabled="1">
      <comment>
        The rule stops processing HTTP objects for which
        the request or response size is greater than 1MB.
      </comment>
      <match>
        <c name="size" op="gt" value="1M"/>
      </match>
      <action name="drop" />
    </rule>

    <rule enabled="true">
      <action name="accept" />
    </rule>
  </table>
</filter>
```

For detailed description of "size" HTTP request filtering conditions see REQ-SIZE, RESP-SIZE, SIZE⁽¹⁵⁵⁾.

Comments and general recommendations

1. Instead of the "size" condition which checks both the request size and the response size, you can use the "req-size" (for checking the size of the HTTP request only) or "resp-size" (for checking the size of the HTTP response only) conditions.

5. Delivery of results to systems-consumers

The EtherSensor Transfer service is responsible for delivering the results of EtherSensor Analyser work to external systems-consumers.

The main idea of the service EtherSensor Transfer is the concept of a predefined transport profile (results delivery profile) for the delivery of an object extracted from traffic.

The transport profile contains data about the server and authentication on it, if the system-consumer of the captured objects is a server, or the path to the local directory, as well as the requirements for stored objects, if we are talking about profiles of the ARCHIVING⁽¹⁸⁸⁾ group.

In the rules for messages filtering⁽¹⁰²⁾ of the service EtherSensor Analyser, one profile, if necessary, can be instantly replaced by another, and since the profiles were created by the administrator in advance and carefully checked, the probability of errors in this part of the settings is minimal.

The results of the analysis of the same captured object can be simultaneously delivered in many ways to many consumers using many transport profiles.

For example:

The content and metadata of the message are delivered simultaneously to the DLP system and to the eDiscovery system, the metadata to the SIEM system running in the SOC of the external MSSP, and the attachment file to the sandbox.

Types of transport profiles:

ARCHIVING⁽¹⁸⁸⁾

The ARCHIVING group profiles are used to deliver the content of captured objects and their metadata to systems such as eDiscovery, Enterprise Archiving, Enterprise Search, and many DLP systems. The delivery methods are SMTP/SMTPS⁽²⁰¹⁾, FTP/FTPS⁽¹⁹¹⁾, SFTP⁽¹⁹⁶⁾, IMAP⁽¹⁹³⁾, FILEDROP⁽¹⁸⁸⁾ (local file system) and SMB/CIFS⁽¹⁹⁸⁾ (network directory).

DLP⁽²⁰³⁾

DLP group profiles are used to deliver content of captured objects and their metadata to DLP systems. All supported DLP systems have their own archive, and it is to this archive that EtherSensor directly delivers captured objects using their proprietary data transfer protocols.

SIEM⁽²¹¹⁾

SIEM group profiles are used to deliver data about an captured object to SIEM systems via the SYSLOG protocol. If necessary, the content of the object is also delivered.

SANDBOX⁽²¹³⁾

The profiles of the SANDBOX group are used to deliver captured objects suspicious of malware to sandbox solutions for further analysis.

STATS⁽²¹⁷⁾

The STATS group profiles are used to deliver statistical data to NetFlow collectors, in particular, to NetFlow collectors of SIEM systems.

GROUP⁽²²⁰⁾

Group profiles are used to balance the load between systems-consumers and include the above transport profiles with preset weights.

The transport profile for the captured object is assigned in the message filter⁹⁹. If during the analysis of an object it turns out that no transport profile has been assigned to it, then it is delivered by the default profile. After successful delivery, the object is deleted from the cache of captured objects, all available information about it is destroyed.

The main format in which EtherSensor delivers the content of reconstructed objects to consumer systems is an EML envelope. Also, the EtherSensor Transfer service is capable of transmitting data in its own internal XML and/or JSON formats in cases where the EML envelope is not a mandatory delivery format (copying data in a directory or using the FTP protocol).

Architecture of the system when using unprotected data transfer protocols:

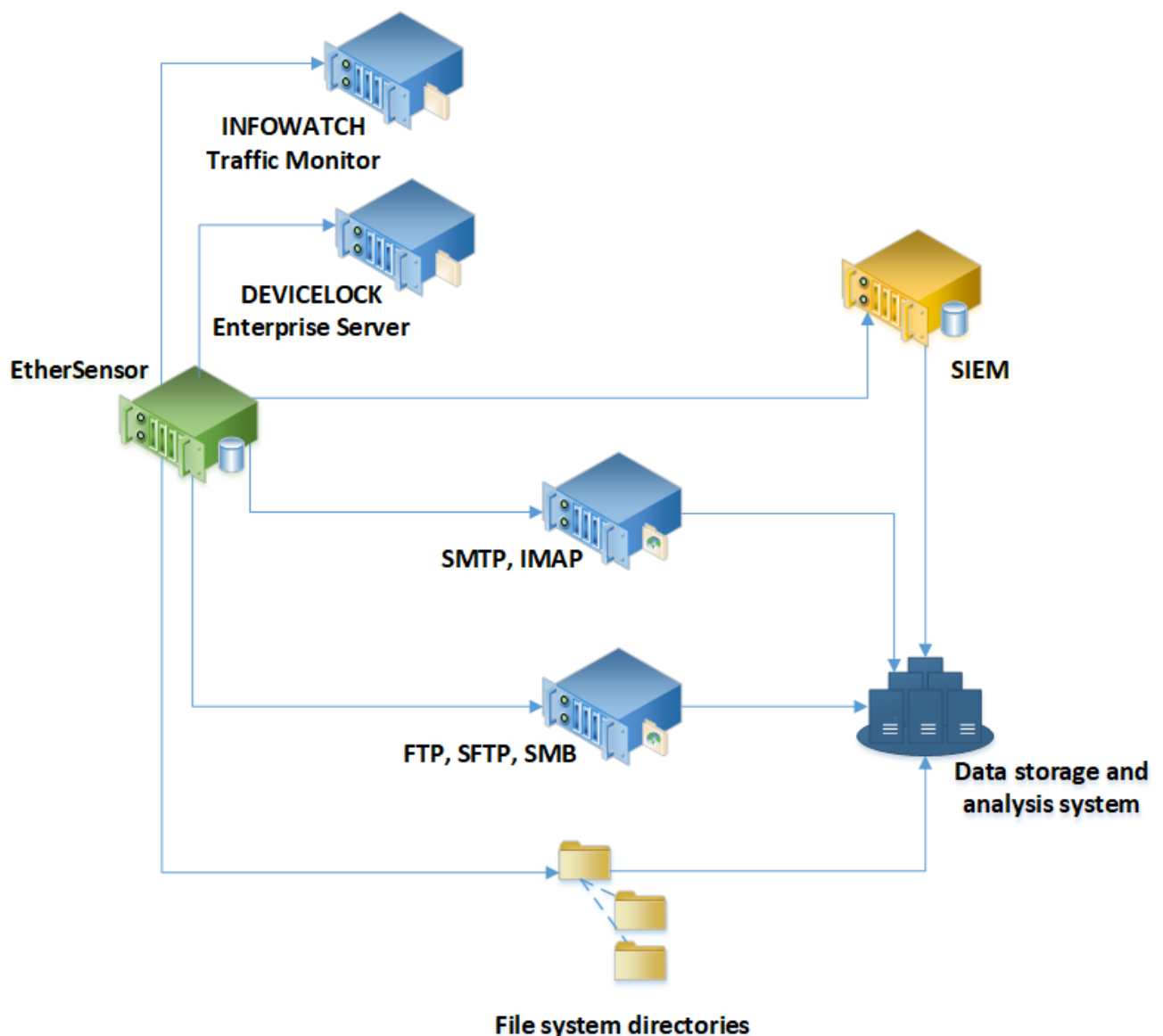


Fig.40. Work scheme of the service EtherSensor Transfer.

Architecture of the system when using protected data transfer protocols:

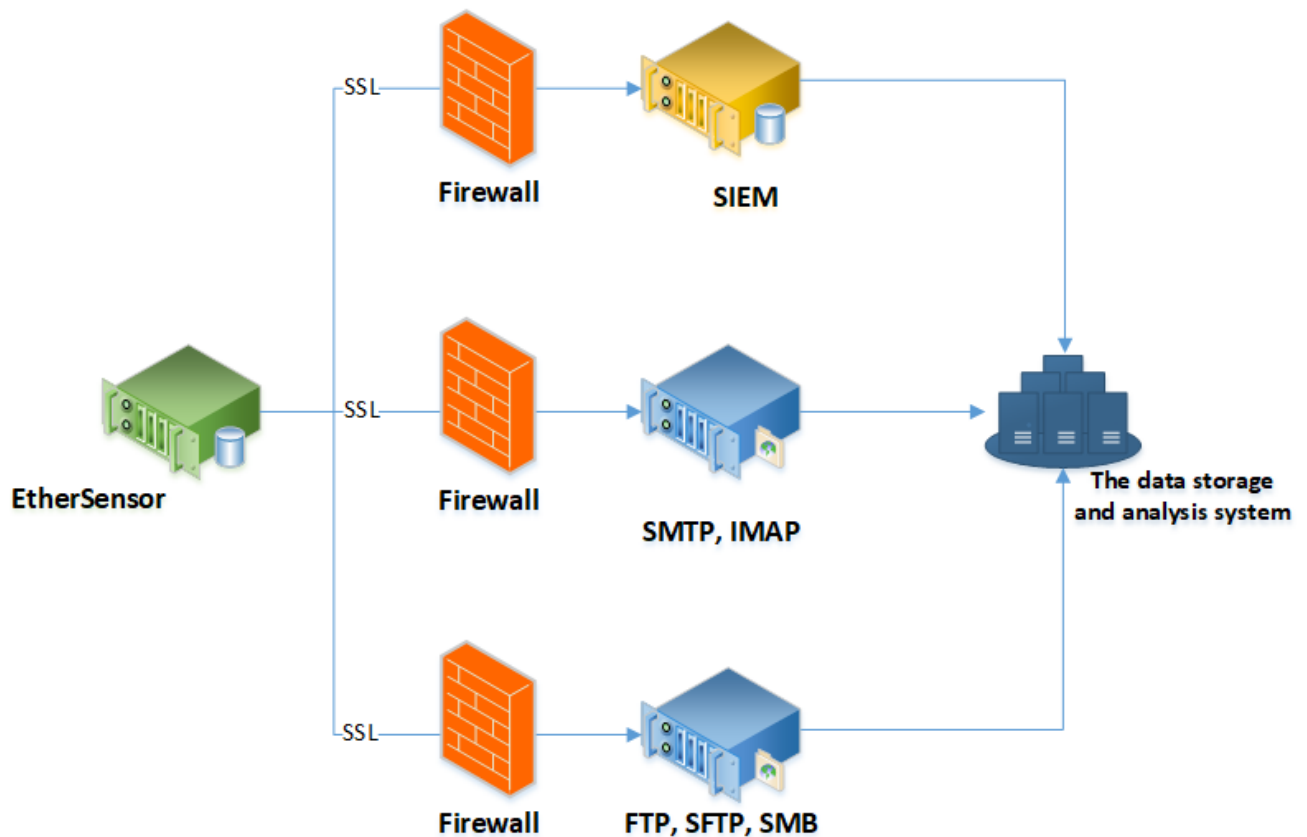


Fig.41. Operation of the EtherSensor Transfer service with secure protocols

EtherSensor Transfer service configuration file

The configuration of the service EtherSensor Transfer is contained in the transfer.xml XML file located in the general configuration directory Microolap EtherSensor [INSTALLDIR]\config.

Command line parameters

The service EtherSensor Transfer during the installation procedure Microolap EtherSensor is installed as a Windows service configured to start automatically. However, it can also be run as a Windows application sensor_transfer.exe and has the following command line parameters:

/process

Run the process sensor_transfer.exe as a normal Windows Win32 process (can be used for debugging).

/service

Run as a Windows service.

/config

Save the default service configuration.

5.1. ARCHIVING profiles

ARCHIVING profiles solve the problem of delivering captured objects to most eDiscovery, Enterprise Archiving, Enterprise Search and DLP systems.

FILEDROP¹⁸⁸

FILEDROP profiles allow you to save captured objects to the local file system, which can be very useful for debugging and testing purposes. In addition, access to the saved objects can be easily provided to external systems-consumers.

SMB/CIFS¹⁹⁸

SMB/CIFS profiles allow you to save captured objects directly to the network folders of systems-consumers.

SMTP/SMTPS²⁰¹, FTP/FTPS¹⁹¹, SFTP¹⁹⁶, IMAP¹⁹³

These delivery profiles of the analysis results are convenient for delivery of captured objects to remote/cloud systems-consumers.

When server EtherSensor processes large data flows, external systems-consumers may not be able to cope with receiving analysis results from EtherSensor due to the heavy load it creates.

In this case, use group profiles to load balance the systems-consumers.

5.1.1. FILEDROP profiles

FILEDROP profiles allow you to save captured objects to the local file system, which can be very useful for debugging and testing purposes. In addition, access to the saved objects can be easily provided to external systems-consumers.

FILEDROP undersea rocks:

1. With large traffic flow EtherSensor creates a large message flow. If you save these messages to disk using FILEDROP then the disk subsystem may be overloaded. Thus it can become a bottleneck for EtherSensor server in general. It is quite possible that sooner or later a system resource shortage will occur and the main function of EtherSensor - traffic capture and analysis - will suffer.
2. It is absolutely unacceptable to allow a large number of objects to accumulate uncontrollably on the file system: be sure to set the **Saved results TTL** parameter in the FILEDROP profile settings.

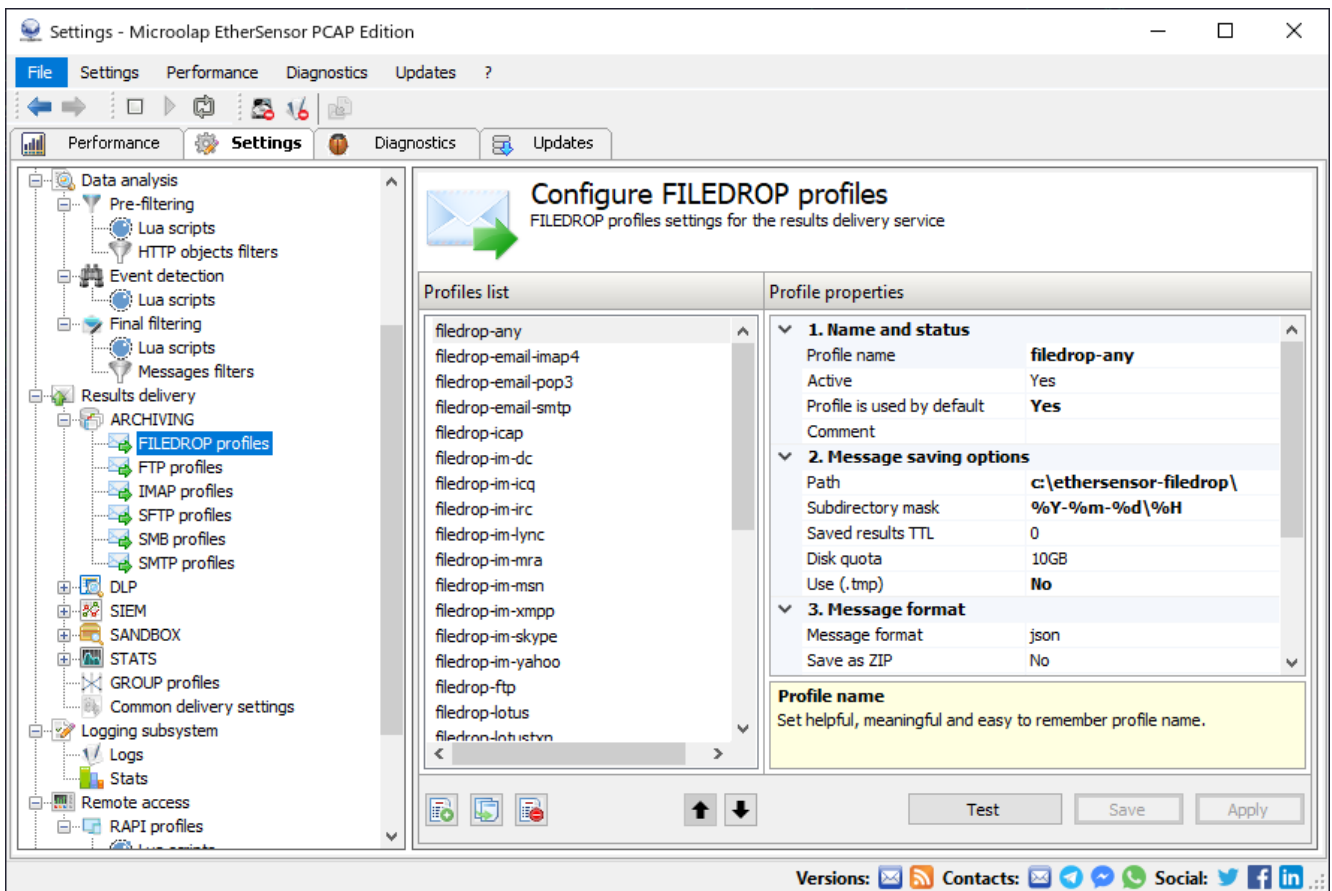


Fig.42. Settings of FILEDROP profiles.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. Message saving options

Path:

The path to the directory where captured messages will be saved.

Disk quota:

The size of the disk quota for storing messages. Example: 10GB or 500MB or 100KB or 10,000. If the quota is exhausted, files will no longer be saved until the quota again allows this. To resume saving files, either free up space or increase the quota.

Use (.tmp):

Enable/disable the use of 2-step moving of files to avoid collisions: 1) The file is moved with a temporary extension, for example, 2012-01-08-15-29-48-586.7.m.zip.tmp, then 2) After the transfer is complete, ".tmp" is deleted from the file name, the file is renamed in 2012-01-08-15-29-48-586.7.m.zip. May be helpful if a process is waiting for new files to appear in the directory (renaming is an atomic operation)."

3. Message format

Message format:

Stored or sent messages format (EML, XML, JSON ...)

Save as ZIP:

Whether to pack the saved messages in a ZIP archive (a file with a .zip extension). If this setting is enabled in conjunction with the "Save as EML" setting, EML message envelopes will be packed in a ZIP archives. If this option is enabled and the "Save as EML" setting is disabled, messages in the internal format will be packed into the ZIP archive.

ZIP file compression rate:

ZIP archive compression rate [0-9], where 0 means no compression, 1 means best compression speed, and 9 means best compression algorithm.

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.1.2. FTP profiles

FTP/FTPS profiles are used to deliver captured objects to remote/cloud systems-consumers.

When server EtherSensor processes large data flows, external systems-consumers may not be able to cope with receiving analysis results from EtherSensor due to the heavy load it creates.

In this case, use group profiles to load balance the systems-consumers.

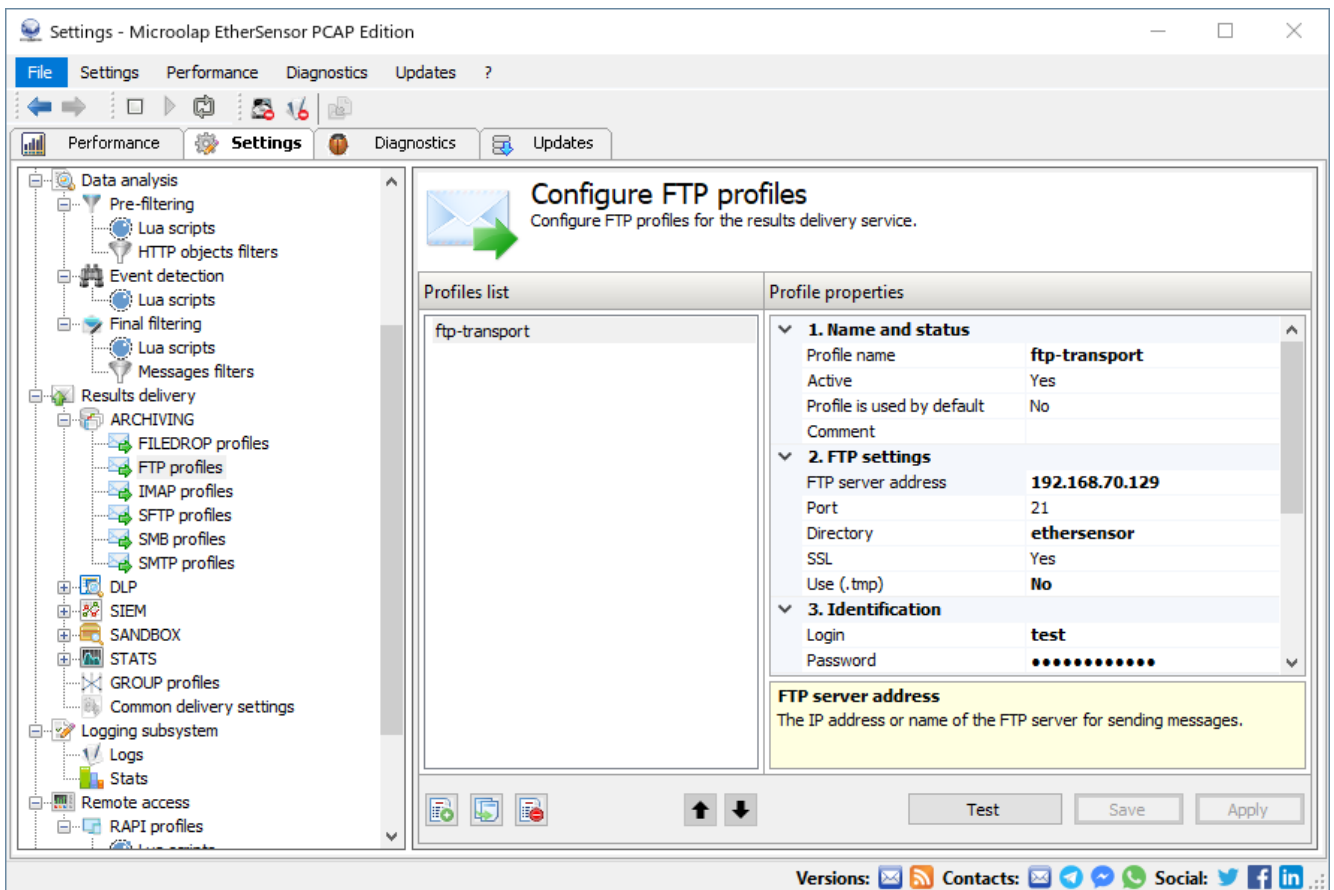


Fig.43. FTP profiles settings.

1. Name and status**Profile name:**

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. FTP settings

FTP server address:

The IP address or name of the FTP server for sending messages.

Port:

FTP-server port to send messages.

Directory:

The directory for storing messages.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Use (.tmp):

Enable/disable the use of 2-step moving of files to avoid collisions: 1) The file is moved with a temporary extension, for example, 2012-01-08-15-29-48-586.7.m.zip.tmp, then 2) After the transfer is complete, ".tmp" is deleted from the file name, the file is renamed in 2012-01-08-15-29-48-586.7.m.zip. May be helpful if a process is waiting for new files to appear in the directory (renaming is an atomic operation)."

3. Identification

Login:

Login to access the FTP-server.

Password:

Password for access to the FTP server.

4. Message format

Message format:

Stored or sent messages format (EML, XML, JSON ...)

Save as ZIP:

Whether to pack the saved messages in a ZIP archive (a file with a .zip extension). If this setting is enabled in conjunction with the "Save as EML" setting, EML message envelopes will be packed in a ZIP archives. If this option is enabled and the "Save as EML" setting is disabled, messages in the internal format will be packed into the ZIP archive.

ZIP file compression rate:

ZIP archive compression rate [0-9], where 0 means no compression, 1 means best compression speed, and 9 means best compression algorithm.

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.1.3. IMAP profiles

IMAP profiles are used to deliver captured objects to remote/cloud users.

When server EtherSensor processes large data flows, external systems-consumers may not be able to cope with receiving analysis results from EtherSensor due to the heavy load it creates.

In this case, use group profiles to load balance the systems-consumers.

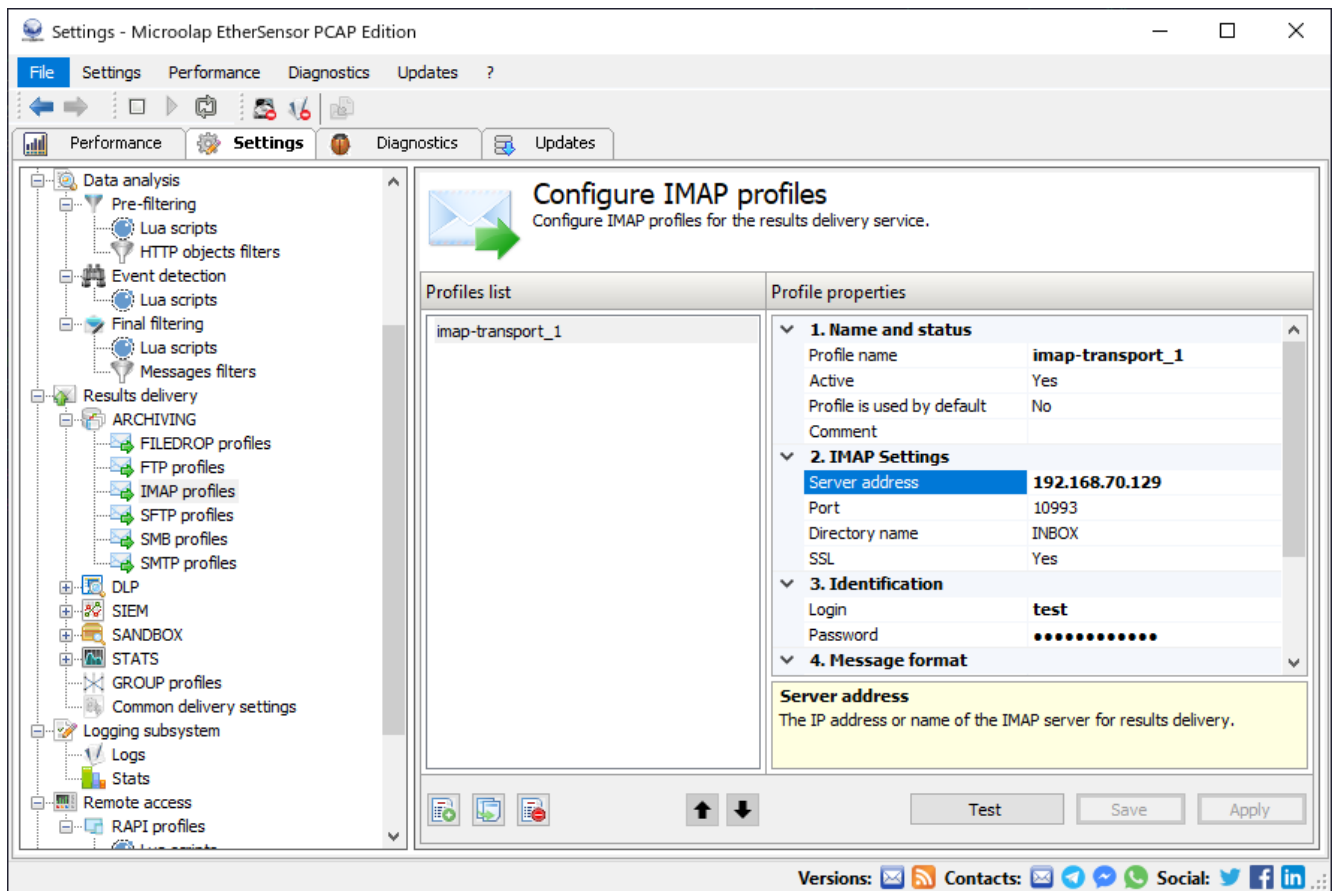


Fig.44. IMAP profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. IMAP Settings

Server address:

The IP address or name of the IMAP server for results delivery.

Port:

IMAP server port for messages delivery.

Directory name:

The name of the IMAP directory to which the message will be delivered.

SSL:

Enables/disables the use of SSL encryption when sending messages.

3. Identification

Login:

Login to access the IMAP server.

Password:

Password to access the IMAP server.

4. Message format

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.1.4. SFTP profiles

SFTP profiles are used to deliver captured objects to remote/cloud users.

When server EtherSensor processes large data flows, external systems-consumers may not be able to cope with receiving analysis results from EtherSensor due to the heavy load it creates.

In this case, use group profiles to load balance the systems-consumers.

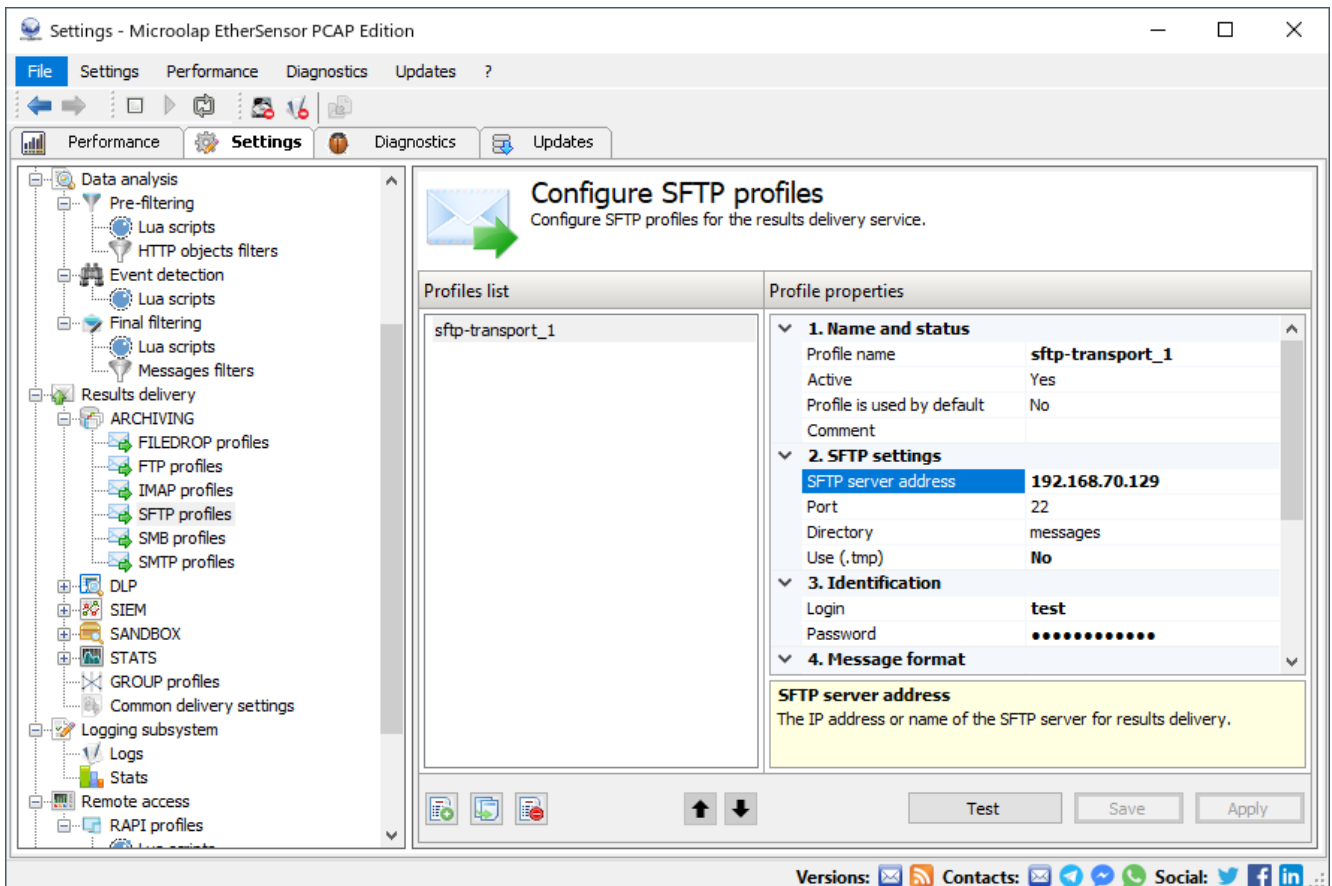


Fig.45. SFTP profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. SFTP settings

SFTP server address:

The IP address or name of the SFTP server for results delivery.

Port:

Port SFTP-server for sending messages.

Directory:

The directory for storing messages.

Use (.tmp):

Enable/disable the use of 2-step moving of files to avoid collisions: 1) The file is moved with a temporary extension, for example, 2012-01-08-15-29-48-586.7.m.zip.tmp, then 2) After the transfer is complete, ".tmp" is deleted from the file name, the file is renamed in 2012-01-08-15-29-48-586.7.m.zip. May be helpful if a process is waiting for new files to appear in the directory (renaming is an atomic operation)."

3. Identification

Login:

Login for access to the SFTP server.

Password:

Password for access to the SFTP-server.

4. Message format

Message format:

Stored or sent messages format (EML, XML, JSON ...)

Save as ZIP:

Whether to pack the saved messages in a ZIP archive (a file with a .zip extension). If this setting is enabled in conjunction with the "Save as EML" setting, EML message envelopes will be packed in a ZIP archives. If this option is enabled and the "Save as EML" setting is disabled, messages in the internal format will be packed into the ZIP archive.

ZIP file compression rate:

ZIP archive compression rate [0-9], where 0 means no compression, 1 means best compression speed, and 9 means best compression algorithm.

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.1.5. SMB profiles

SMB/CIFS profiles allow you to save captured objects directly to the network folders of systems-consumers.

The recommendations for disk subsystem performance and lifetime settings for saved objects are the same as for FILEDROP profiles.

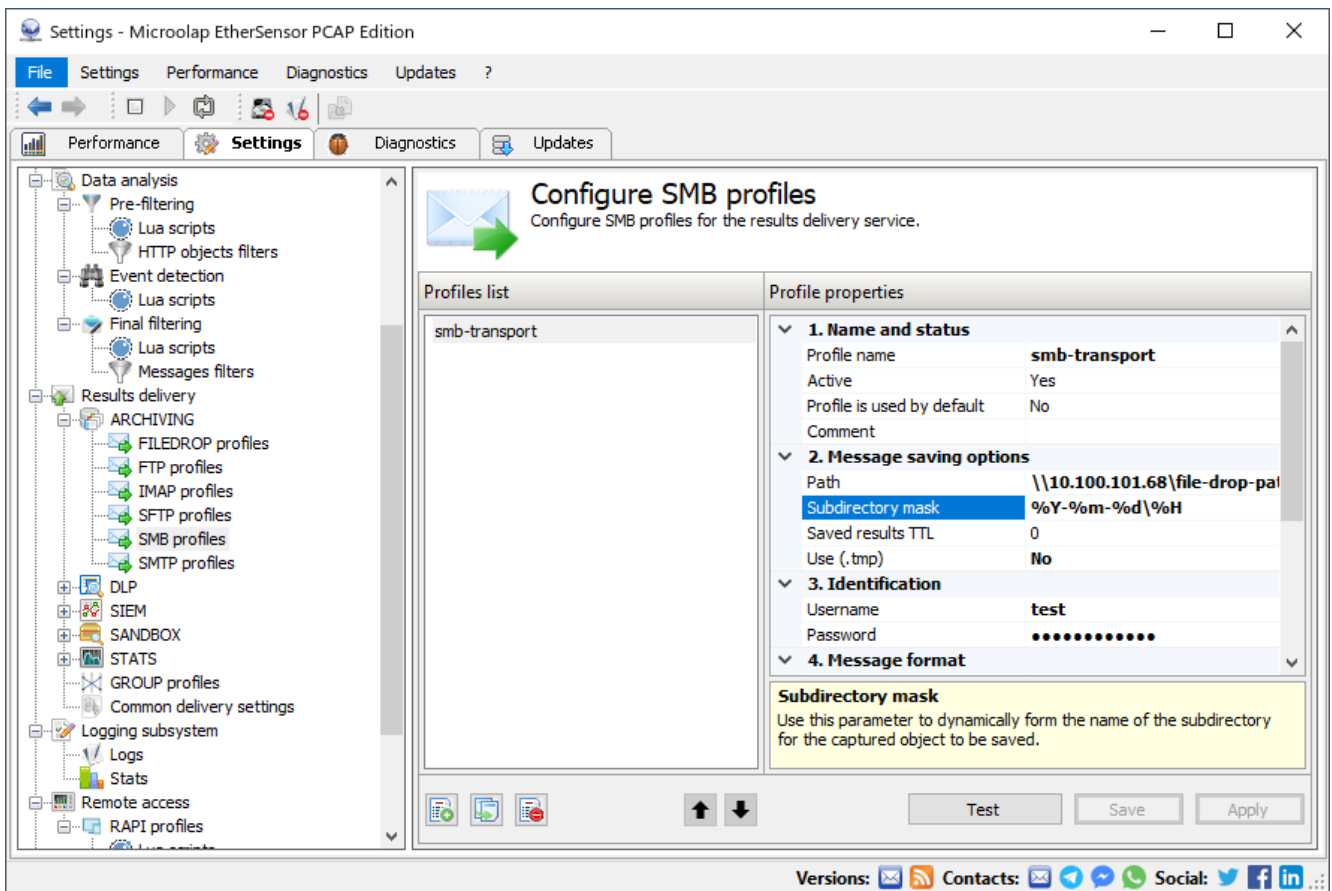


Fig.46. SMB profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. Message saving options

Path:

Path to the SMB/CIFS directory to save messages. Example: \\ARCHSERVER1\Messages

Use (.tmp):

Enable/disable the use of 2-step moving of files to avoid collisions: 1) The file is moved with a temporary extension, for example, 2012-01-08-15-29-48-586.7.m.zip.tmp, then 2) After the transfer is complete, ".tmp" is deleted from the file name, the file is renamed in 2012-01-08-15-29-48-586.7.m.zip. May be helpful if a process is waiting for new files to appear in the directory (renaming is an atomic operation)."

3. Identification

Username:

User name for accessing the SMB/CIFS directory to save messages.

Password:

Password for accessing the SMB/CIFS directory to save messages.

4. Message format

Message format:

Stored or sent messages format (EML, XML, JSON ...)

Save as ZIP:

Whether to pack the saved messages in a ZIP archive (a file with a .zip extension). If this setting is enabled in conjunction with the "Save as EML" setting, EML message envelopes will be packed in a ZIP archives. If this option is enabled and the "Save as EML" setting is disabled, messages in the internal format will be packed into the ZIP archive.

ZIP file compression rate:

ZIP archive compression rate [0-9], where 0 means no compression, 1 means best compression speed, and 9 means best compression algorithm.

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.1.6. SMTP profiles

SMTP profiles are used to deliver captured objects to remote/cloud users.

When server EtherSensor processes large data flows, external systems-consumers may not be able to cope with receiving analysis results from EtherSensor due to the heavy load it creates.

In this case, use group profiles to load balance the systems-consumers.

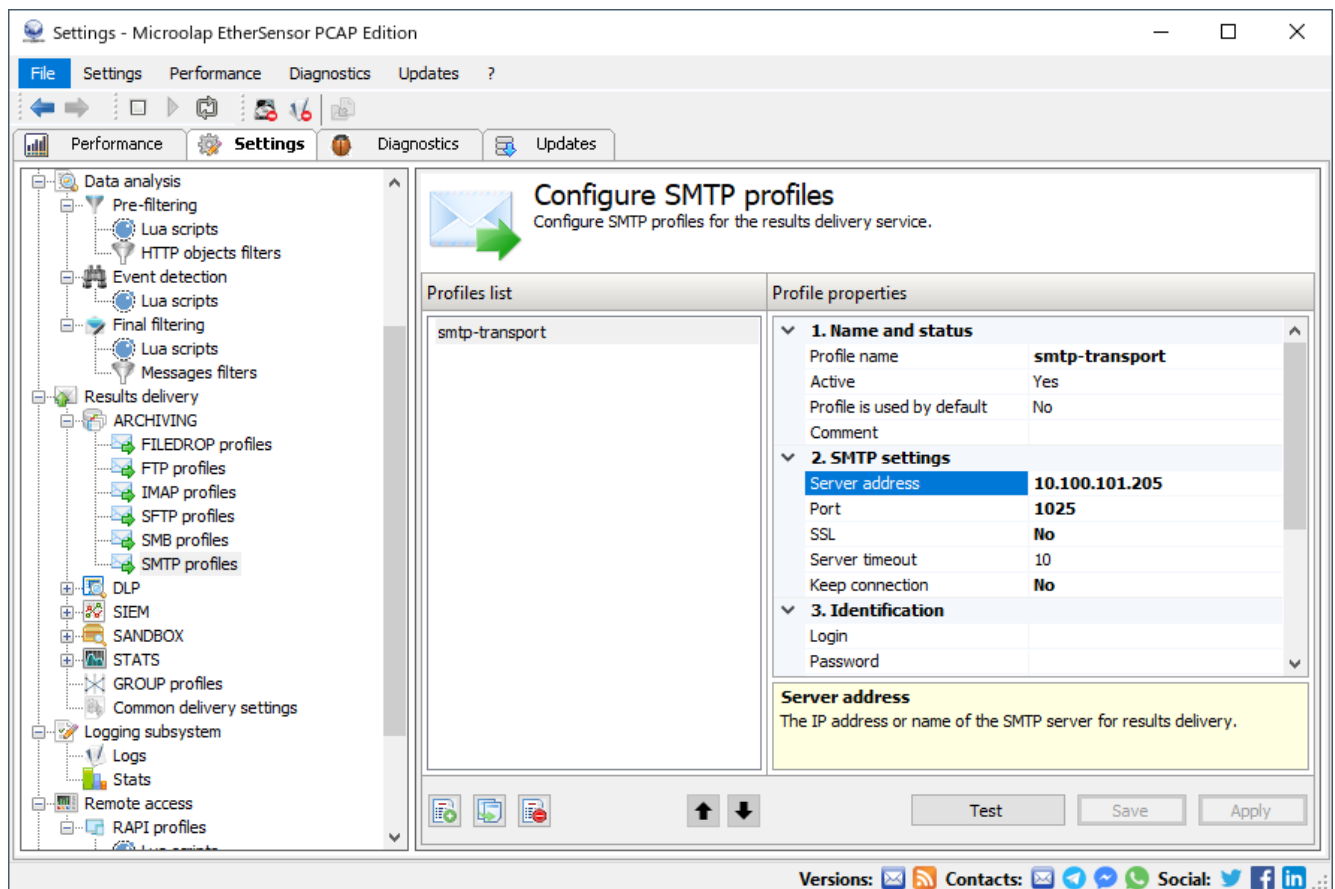


Fig.47. SMTP profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. SMTP settings

Server address:

The IP address or name of the SMTP server for results delivery.

Port:

SMTP server port for sending messages.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Server timeout:

Timeout (response timeout) of the SMTP server in seconds.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Identification

Login:

Login to access the SMTP server.

Password:

Password for access to the SMTP server.

Sender address:

The address of the sender of the message for the consumer system.

Your comment on this profile.

Recipient address:

A working email address of the consumer system (for example, the message archiving system).

4. Message format

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.2. DLP profiles

Integration of EtherSensor with DLP-systems solves the problem of receiving data from hosts on which it is impossible or undesirable to install advanced and resource-intensive endpoint-DLP software.

Most DLP systems support receiving captured messages using generic methods from the ARCHIVING¹⁸⁸ section, but the delivery of captured objects to some of them requires using their own proprietary protocols.

DeviceLock Enterprise Server²⁰⁴ (DLES²⁰⁴)

DeviceLock Enterprise Server is the central archive of the DeviceLock DLP Suite solution. The idea of integrating EtherSensor and DLES is that from the point of view of data exchange with DLES, the sensor behaves like an endpoint DeviceLock agent, using the proprietary DeviceLock protocol for

Fig.48. DEVICELOCK profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. Message saving options

Path:

The path to the directory where captured messages will be saved.

Used for intermediate storage of results sent to DEVICELOCK Enterprise Server.

Disk quota:

The size of the disk quota for storing messages. Example: 10GB or 500MB or 100KB or 10,000. If the quota is exhausted, files will no longer be saved until the quota again allows this. To resume saving files, either free up space or increase the quota.

3. Connection

Server address:

The IP address or name of the DLES (DeviceLock Enterprise Server) server for results delivery.

Server port:

The port used to notify the DEVICELOCK Enterprise Server about messages (events).

Service port:

The port through which the DEVICELOCK Enterprise Server "grabs" the results (messages/events).

4. Notification of DeviceLock

Time-out:

Sets the timeout for DLES (DeviceLock Enterprise Server) notification in seconds. If there are unsent messages after the timeout has elapsed, the server will be notified about the availability of results.

Quantity:

Sets the number of accumulated messages upon which the notification of DLES will be performed.

5. Message format

Duplicate headers:

Enables/disables the saving of standard message headers, as well as of "X-Sensor" headers, additionally, in a separate message attachment - the "microolap_msis_headers.txt" file. Available options: "all" - saves all message headers; "xsensor" - save headers with the "X-Sensor" prefix; "other" - the standard "From", "To", "Cc", "Bcc", and other headers that do not fall under the flag "xsensor"; "none" - disables saving message headers in a separate attachment.

6. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

7. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.2.2. FALCONGAZE profiles

Falcongaze Secure Tower Server is the core component of Falcongaze's DLP solution.

The idea of integrating EtherSensor and Falcongaze Secure Tower is that from the point of view of information exchange with Falcongaze, the sensor behaves as an endpoint agent of Falcongaze, using the proprietary Falcongaze protocol for direct delivery of captured objects to the archive.

Further work on applied problems takes place in the Falcongaze Secure Tower paradigm.

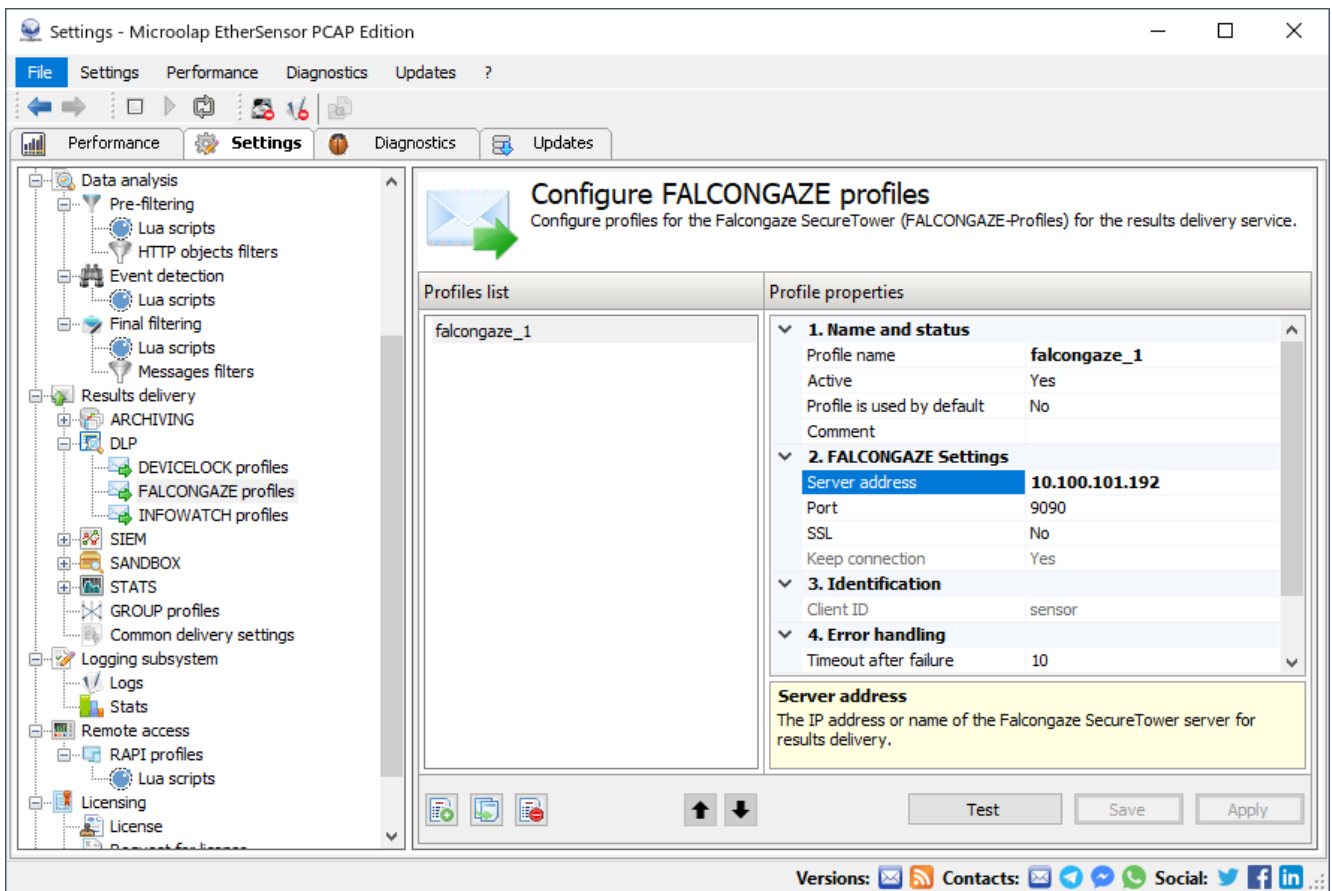


Fig.49. FALCONGAZE profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. FALCONGAZE Settings

Server address:

The IP address or name of the Falcongaze SecureTower server for results delivery.

Port:

Falcongaze SecureTower server port for results delivery..

SSL:

Enables/disables the use of SSL encryption when sending messages.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Identification

Client ID:

The unique EtherSensor client ID for Falcongaze server. It is designed to register a client in Falcongaze SecureTower for delivery of the traffic analysis results.

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.2.3. INFOWATCH profiles

InfoWatch Traffic Monitor archive accumulates objects important from the point of view of DLP tasks for further analysis.

The idea of integrating EtherSensor and InfoWatch Traffic Monitor is that from the point of view of exchanging information with the InfoWatch Traffic Monitor archive, the sensor behaves as an InfoWatch endpoint agent, using the proprietary InfoWatch protocol for direct delivery of captured objects to the archive.

Further work on applied tasks takes place in the InfoWatch Traffic Monitor paradigm.

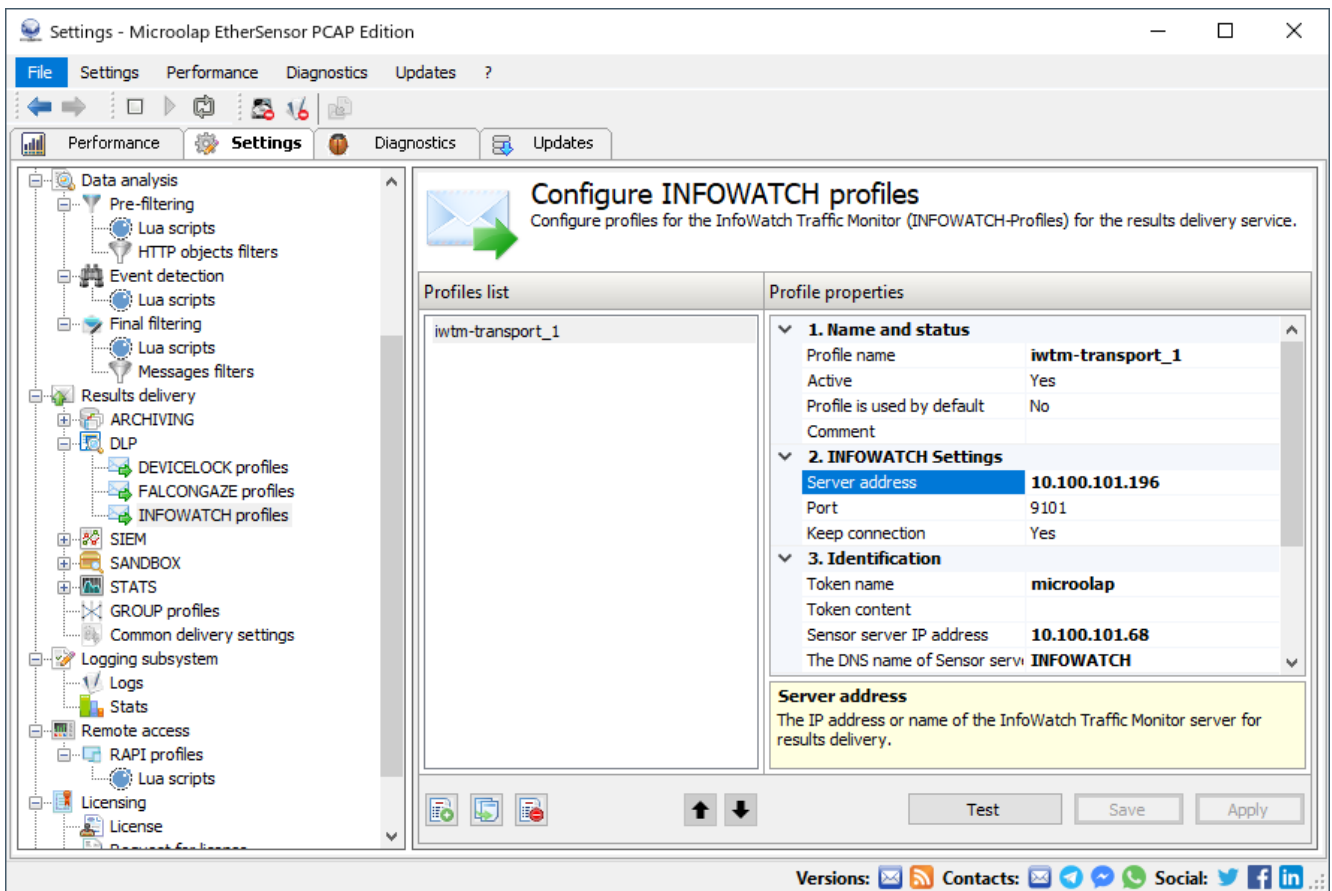


Fig.50. INFOWATCH profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. INFOWATCH Settings

Server address:

The IP address or name of the InfoWatch Traffic Monitor server for results delivery.

Port:

InfoWatch Traffic Monitor server port for sending messages, 9101 by default.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Identification

Token name:

A token name to access the InfoWatch Traffic Monitor server, "microolap" by default.

Token content:

The token contents (password) specified in the InfoWatch Traffic Monitor settings.

EtherSensor server IP address:

The IP address of EtherSensor server, allows the InfoWatch Traffic Monitor server to distinguish between different data sources.

The DNS name of EtherSensor server:

The DNS name of EtherSensor server, allows the InfoWatch Traffic Monitor server to distinguish between different data sources.

4. Message format

Message format:

Stored or sent messages format (EML, XML, JSON ...)

5. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

6. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.3. SIEM profiles

SIEM profiles are used to deliver the analysis results of captured objects to systems-consumers that receive event data via the SYSLOG server.

As a rule, these are SIEM systems, and any SIEM system includes a SYSLOG server. EtherSensor can dynamically generate an arbitrary SYSLOG string using a custom Lua script²¹³, adapting its format to the requirements of a specific SIEM system.

5.3.1. SYSLOG profiles

SYSLOG profiles are used to deliver the analysis results of captured objects to systems-consumers that receive data about events via a SYSLOG server (as a rule, these are SIEM systems).

The SYSLOG string can be formed both as a result of the filter of the service EtherSensor Analyser⁹⁹, and by processing the captured object with a prepared Lua script²¹³.

Using Lua scripts allows you to prepare data in real time in a format specific for any SIEM system without so-called "connectors".

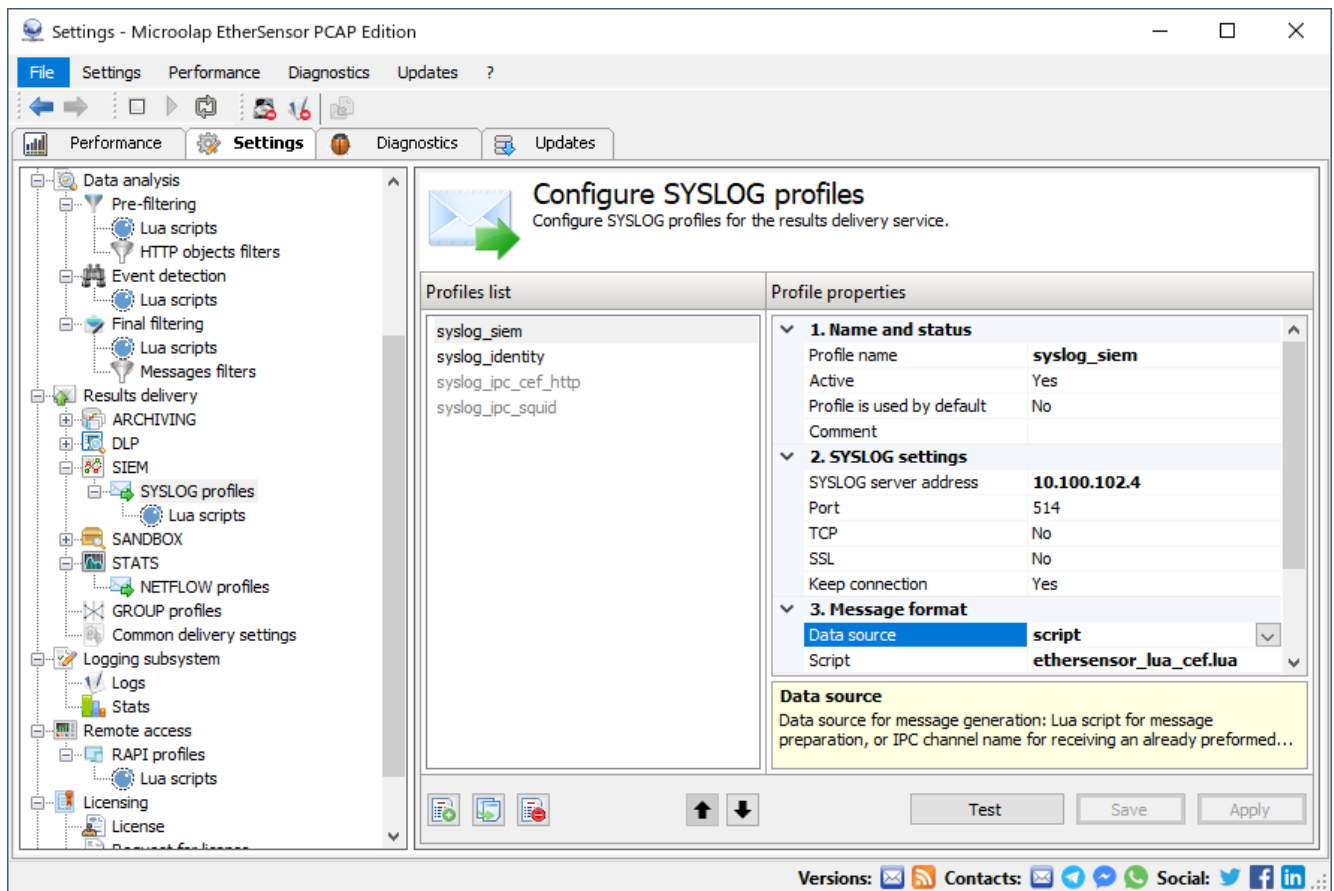


Fig.51. SYSLOG profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. SYSLOG settings

SYSLOG server address:

The IP-address or name of the SYSLOG server for results delivery.

Port:

SYSLOG server port for sending messages.

TCP:

Allows use of the TCP protocol to send messages to the SYSLOG server. This is necessary if SSL encryption is used when sending messages to the consumer system.

Sending to SYSLOG server via SSL only works when TCP is enabled.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Message format

Script:

The name of the Lua script file that will be used to generate the message to be sent to the SYSLOG server. The script file must be in the \scripts subfolder.

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.3.1.1. Lua scripts

The function of preparing SYSLOG messages using Lua scripts assigned in the delivery profile is in the pre-release stage.

If you want to experiment with us, write us at support@microolap.com.

5.4. SANDBOX profiles

VIRUSTOTAL²¹³

VIRUSTOTAL profiles are used to deliver captured objects to VirusTotal service for further analysis.

ATHENA²¹⁵

ATHENA profiles are used to deliver captured objects to the ATHENA server for scanning by both locally installed antiviruses and external analytical resources: VirusTotal, Spamhaus, etc.

5.4.1. VIRUSTOTAL profiles

VIRUSTOTAL profiles are used to deliver captured objects to VirusTotal service for further analysis.

According to VirusTotal licensing policy, you cannot use VirusTotal public service for commercial purposes. Please check VirusTotal terms and conditions before obtaining an access key: <https://developers.virustotal.com/reference>.

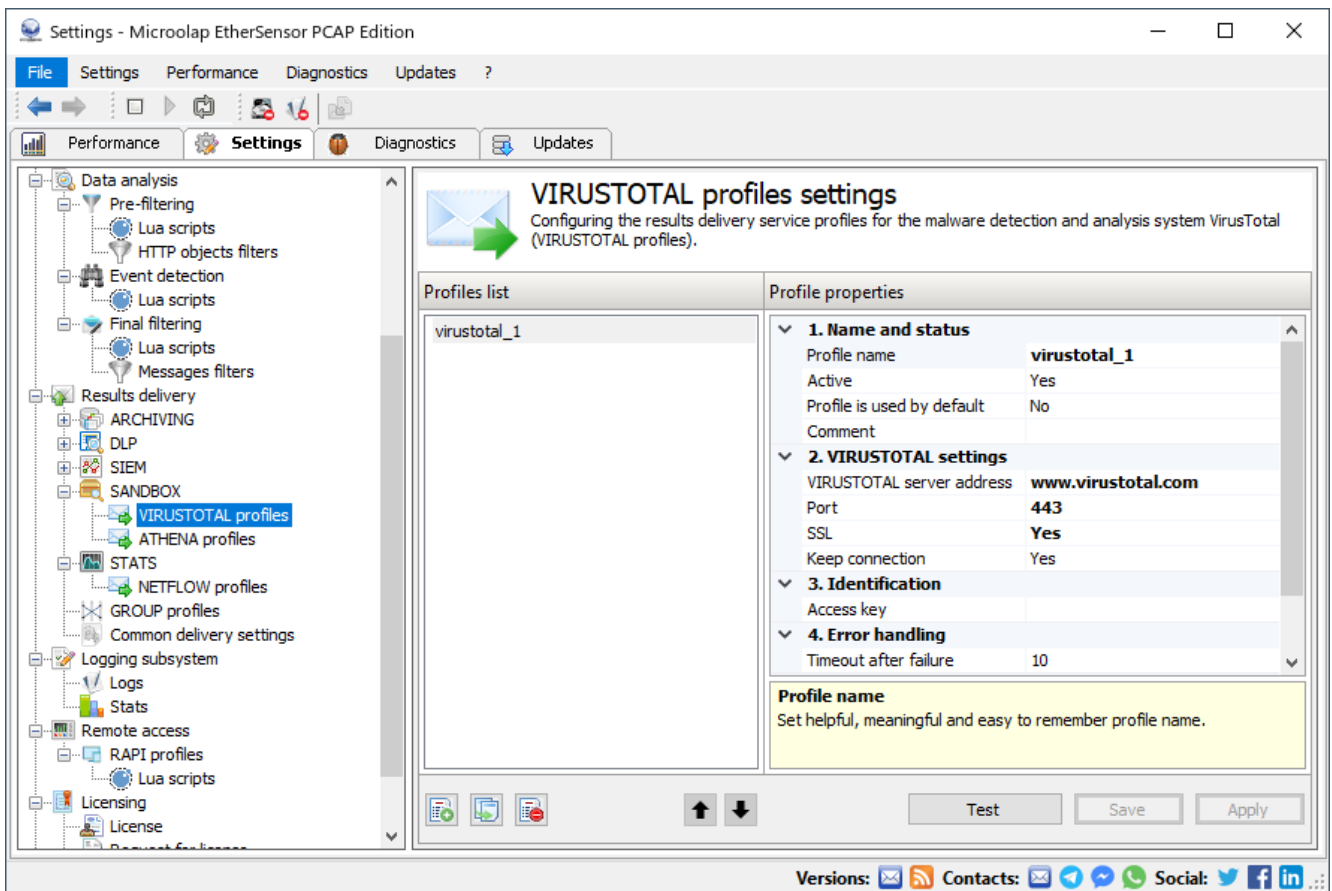


Fig.52. VIRUSTOTAL profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. VIRUSTOTAL settings

VIRUSTOTAL server address:

VIRUSTOTAL server IP address or name to deliver captured objects.

Port:

VIRUSTOTAL server port to deliver captured objects.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Identification

Access key:

VIRUSTOTAL server resources access key (undoubtedly, you honor the license policy of VirusTotal: <https://developers.virustotal.com/reference>).

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.4.2. ATHENA profiles

ATHENA profiles are used to deliver captured objects to the ATHENA server for scanning by both locally installed antiviruses and external analytical resources: VirusTotal, Spamhaus, etc.

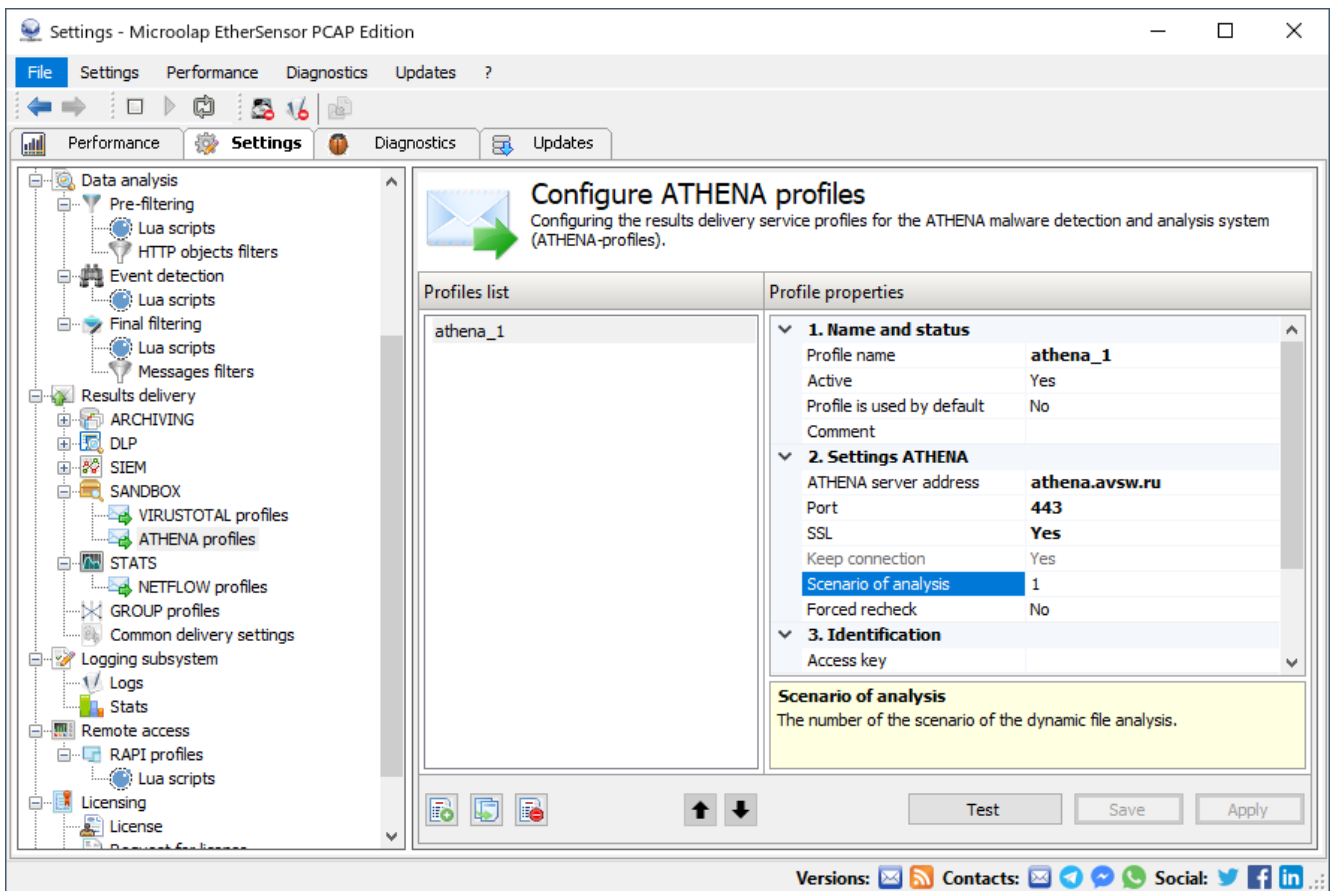


Fig.53. ATHENA profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. Settings ATHENA

ATHENA server address:

The IP address or name of the ATHENA server for results delivery.

Port:

ATHENA server port for results delivery.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

Scenario of analysis:

The number of the scenario of the dynamic file analysis.

Forced recheck:

Forced recheck flag.

3. Identification

Access key:

ATHENA server resources access key.

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.5. STATS profiles

NETFLOW²¹⁸

The STATS group profiles are used to deliver statistical data to NetFlow collectors, in particular, to NetFlow collectors of SIEM systems.

5.5.1. NETFLOW profiles

NETFLOW profiles are used to deliver statistical data to NetFlow collectors, usually to NetFlow collectors of SIEM systems.

NetFlow collectors can also get statistics on slices of active TCP sessions from network equipment, and this is a commonly used approach. But there is a known problem: the shorter the time interval for obtaining such slices, the more resources are required from the network equipment - it is not useful for its main function. In addition, the shorter slice time slot increases the cost of resources on the NetFlow collector side of the data collection itself.

EtherSensor to extract an object from a TCP session in any case must reconstruct it, and after that it has all the necessary data about the TCP session itself. At this point, it may well transmit session data to the NetFlow collector assigned in the NETFLOW profile.

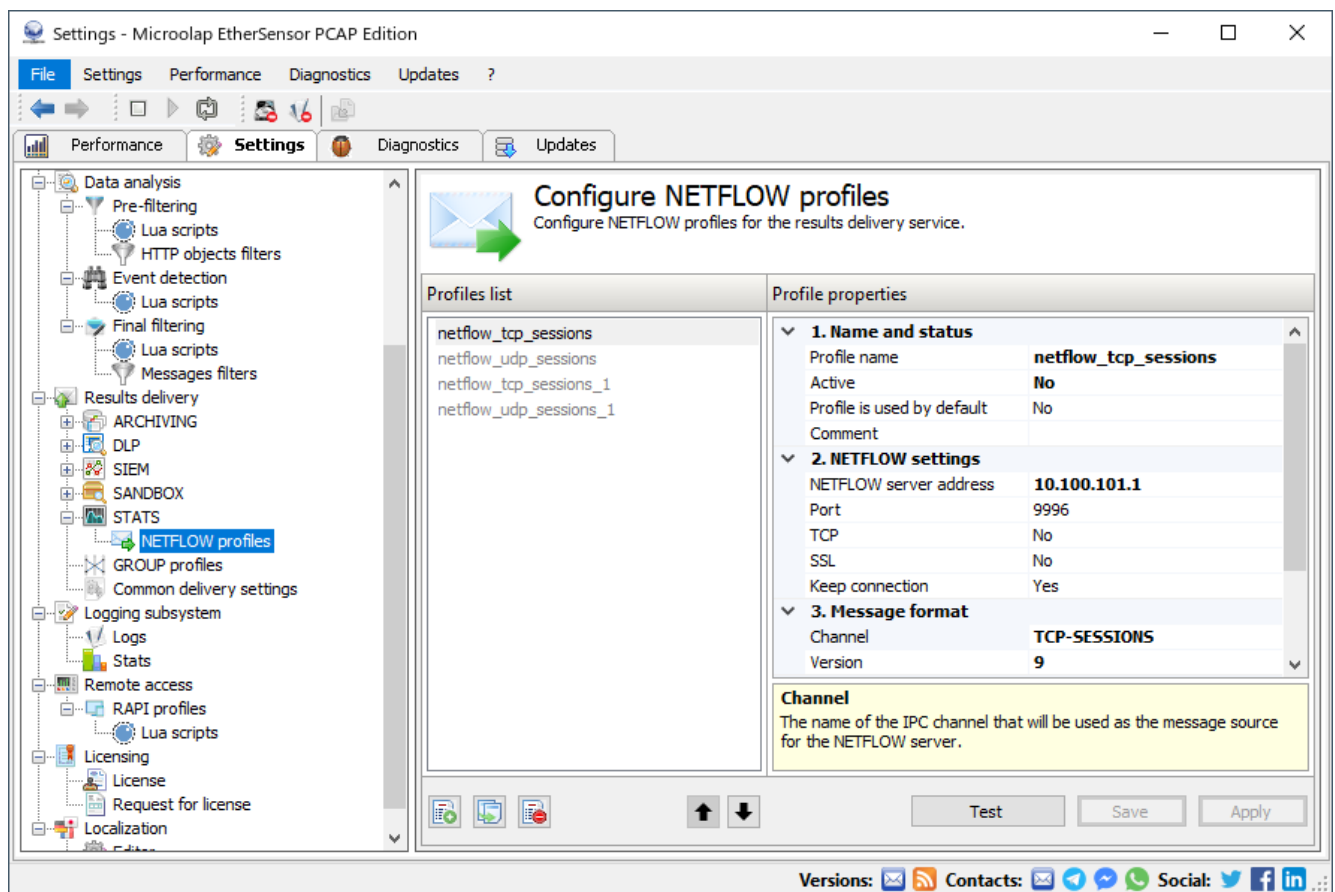


Fig.54. NETFLOW profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Active:

The transport profile is not used for message delivery if it is turned off.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. NETFLOW settings

NETFLOW server address:

The IP-address or name of the NETFLOW server for results delivery.

Port:

NETFLOW server port for sending messages.

TCP:

Allows use of the TCP protocol to send messages to the NETFLOW server. This is necessary if SSL encryption is used when sending messages to the consumer system.

SSL:

Enables/disables the use of SSL encryption when sending messages.

Keep connection:

Send all messages in the same connection with the server. If this option is disabled, then each message will be sent in a separate TCP connection.

3. Message format

Channel:

The name of the IPC channel that will be used as the message source for the NETFLOW server.

Version:

NETFLOW protocol version.

4. Error handling

Timeout after failure:

Timeout in seconds to retry message delivery in the event of receiver rejection.

5. For GROUP profiles

Weight:

The weight of the profile (from 1 to 10) specifies a proportion for the distribution of messages among profiles and is used only if this profile is included in a GROUP profile.

Reserve profile:

Enable/disable the use of the profile as a reserve one. If this setting is enabled and the main (non-reserve) delivery profiles fail, this profile will be used for message delivery. The setting is valid only when used in a GROUP profile.

5.6. GROUP profiles

In the case of large input traffic flows EtherSensor can create such captured objects flows for systems-users that they will no longer be able to cope with receiving them. As a rule, in this case you need to install several systems-consumers of this type and distribute the results EtherSensor between them.

Group profiles are used to balance the load between systems-consumers and include delivery profiles with pre-set weights.

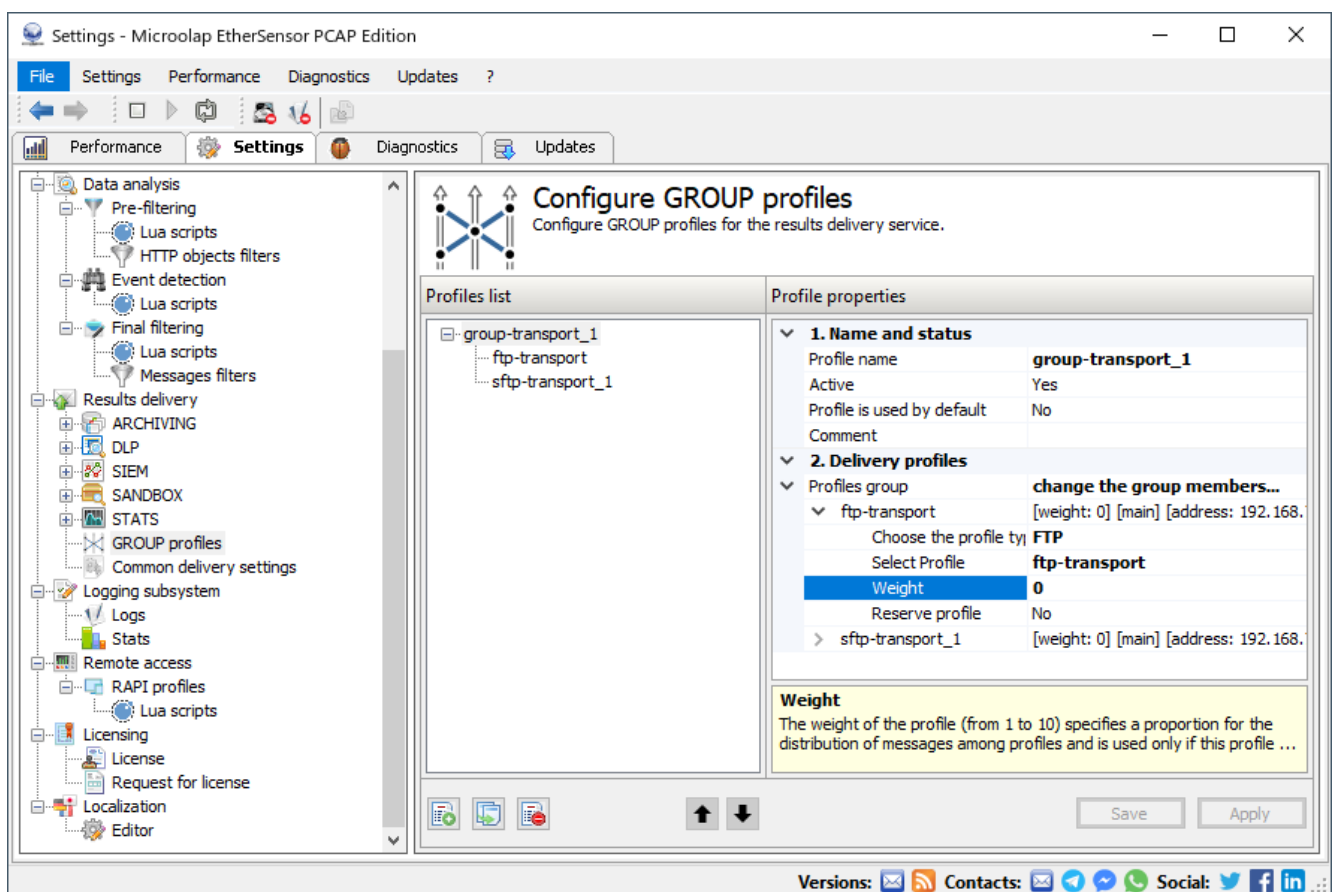


Fig.55. GROUP profiles settings.

1. Name and status

Profile name:

Set helpful, meaningful and easy to remember profile name.

Profile is used by default:

"Yes" means that this delivery profile is used by default.

Comment:

Description of the transport profile.

2. Delivery profiles

Profiles group:

A list of pre-existing delivery profiles used in the GROUP profile.

How group transport profile works

Just like a normal transport profile, a group transport profile can be assigned as the default profile. It can also be directly assigned for use in message filtering rules.

In contrast to the standard transport profile, which contains fine settings of delivery of EtherSensor results according to the method used to deliver them to the system-consumer, the group profile contains only the names of pre-created standard transport profiles.

In a group profile, standard profiles may have the status of main and reserve profiles. Main profiles are used to deliver messages in normal operation mode. Reserve profiles are used to deliver messages when none of the main profiles can deliver the message.

Also in the group profile, standard profiles can be assigned weights to deliver messages to systems-consumers in the required proportion. Weights can be assigned to both main and reserve profiles.

Delivery of results using a group profile

Let's consider an example of delivery of results using a group profile in which three standard transport profiles are assigned as main (SMTP 1, SMTP 2, SMTP 3) with equal weights, and one standard transport profile (FILEDROP 1) is assigned as reserve.

In normal operation, messages are delivered to message receivers in equal proportions.

If one of the message receivers described in the standard transport profile is unable to receive messages, the load on receiving messages falls on the other main transport profiles.

If all main transport profiles (SMTP 1, SMTP 2, SMTP 3) cannot accept messages, then the reserve transport profile (FILEDROP 1) will be used for message delivery until at least one main transport profile is able to deliver messages.

5.7. General delivery settings

General settings for delivering EtherSensor work results to systems-consumers.

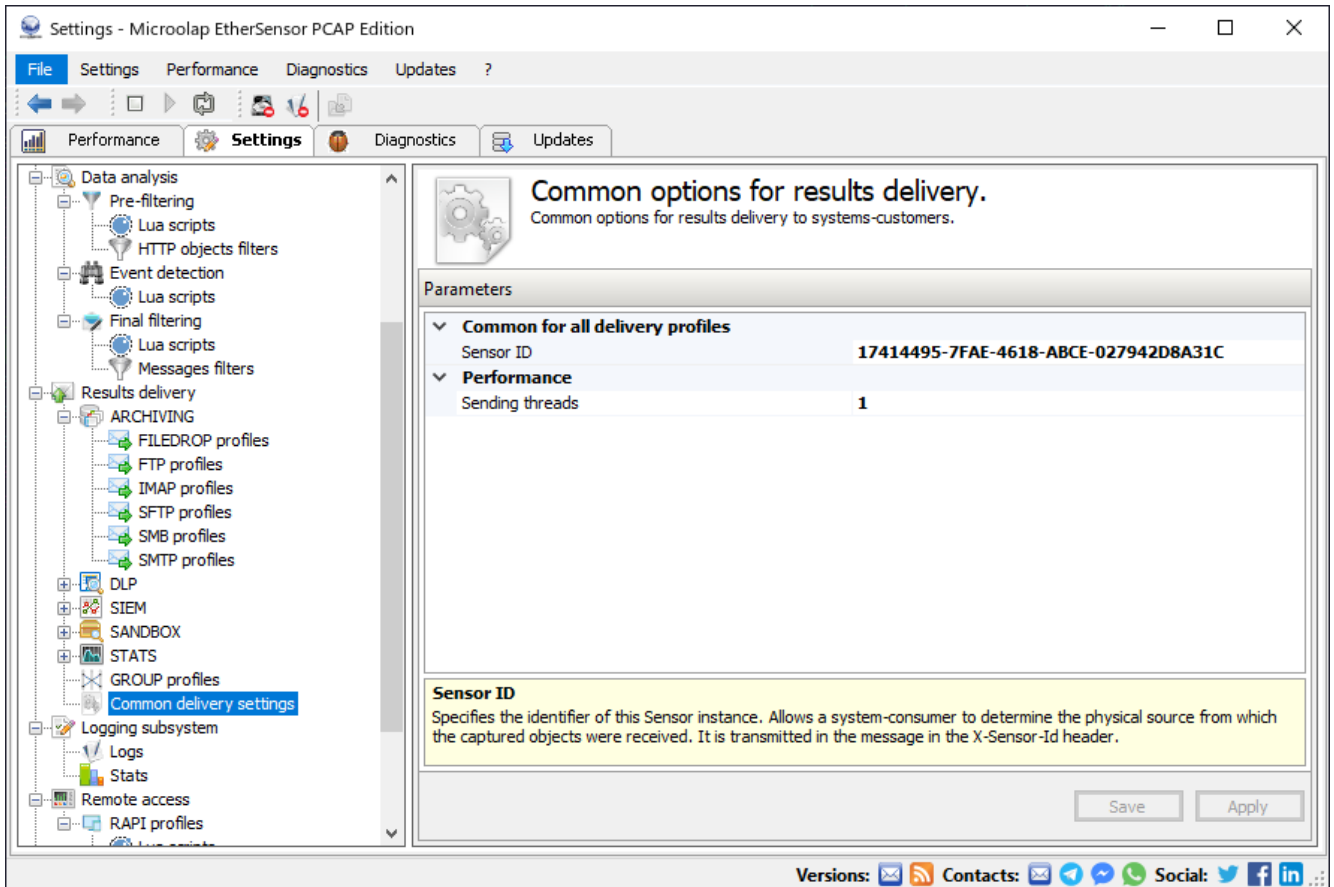


Fig.56. General settings of EtherSensor Transfer service.

Common for all delivery profiles

Sensor ID:

Specifies the identifier of this EtherSensor instance. Allows a system-consumer to determine the physical source from which the captured objects were received. It is transmitted in the message in the X-Sensor-Id header.

Performance

Sending threads:

Specifies the number of threads that send messages. The maximum number of sending threads cannot be greater than the current number of CPUs*2 and cannot be less than 1.

6. EtherSensor Watcher logging

EtherSensor starts its work by launching the logging subsystem EtherSensor Watcher. Logging subsystem is designed to log messages of the EtherSensor service, to work with logs, to track the current state of EtherSensor and valid licenses.

The EtherSensor Watcher service collects system messages from other services of EtherSensor and writes them, depending on the named logging channel, log level and other criteria, to files or syslog servers designated by the administrator.

EtherSensor Watcherservice configuration file.

The configuration of the service EtherSensor Watcher is contained in the XML file watcher.xml, located in the common configuration directory EtherSensor [INSTALLDIR]\config.

Command line parameters

The service EtherSensor Watcher during the installation procedure Microolap EtherSensor is installed as a Windows service configured to start automatically. However, if necessary, it can be run as a Windows application sensor_watcher.exe with the following command line parameters:

/process

Run the process sensor_watcher.exe as a normal Windows Win32 process (can be used for debugging).

/service

Run as a Windows service.

/config

Save the default service configuration.

6.1. Configure EtherSensor Watcher logging

In the window shown below, you can create or edit existing EtherSensor event logging rules.

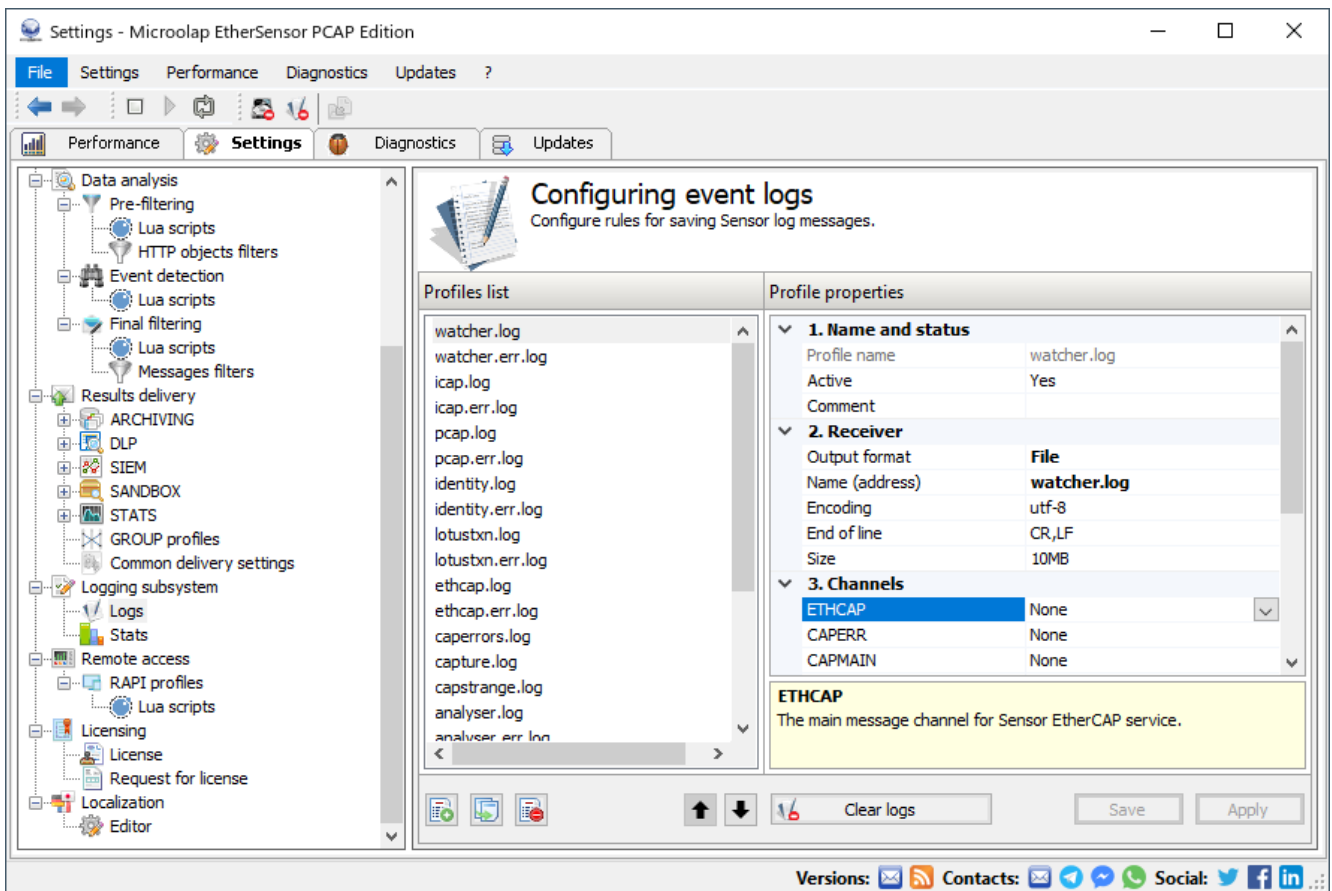


Fig.57. EtherSensor Watcher service logging profiles settings.

Named channels:

The logging subsystem "listens" to so-called named channels (ETHERCAP, ANALYSER ...) through which other services EtherSensor transmit messages of various importance levels - info, warning, error and criterror. The names of such named channels are generally arbitrary and can be set by the administrator at his discretion in the settings of the corresponding services.

According to the settings, messages received from named channels are either ignored or sent to consumers: these can be SYSLOG servers or local/network files specified in the settings.

The settings can also specify the size limit for log files.

Logging profile parameter setting:

If you need to delete existing profiles or define additional logging profiles, use the buttons **New**, **Clone** and **Delete**:

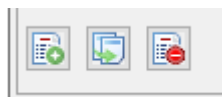


Fig.58. Buttons for creating, deleting and cloning log profiles.

1. Name and status

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

The log profile is not used if it is disabled.

Comment

Description of the log profile.

2. Receiver

Output format

Select a format for the output data. file, win-file - data output format is utf-8, unix-file - data output is in ASCII, syslog - data output, in ASCII, is sent to a remote SYSLOG-Server.

Name (address)

The file name or address of the remote SYSLOG-Server. For example, file: events.log, located in a subdirectory log of EtherSensor installation directory, SYSLOG-Server: 192.0.0.168:514.

Encoding

Data output encoding.

End of line

Set characters that mark the end of the saved message in the event log.

Size

The maximum size for the event log. Syntax: 1000B or 100KB or 50MB or 1GB.

3. Channels

ETHCAP

The main message channel for EtherSensor EtherCAP service.

CAPERR

The logging channel for EtherSensor EtherCAP service traffic analysis errors.

CAPMAIN

The logging channel for the processed connections of EtherSensor EtherCAP service.

ICAP

The main logging channel for EtherSensor ICAP service.

ICAP-REQUEST

The logging channel for EtherSensor ICAP service HTTP requests.

IDENTITY

IDENTITY service logging channel.

PCAP

PCAP service logging channel.

LOTUSTXN

The main logging channel for EtherSensor LotusTXN service.

ANALYSER

The main logging channel for EtherSensor Analyser service.

FILTER

The logging channel for EtherSensor Analyser service data filtering system.

TRANSFER

The main logging channel for EtherSensor Transfer, the results delivery service.

WATCHER

The main logging channel for EtherSensor Watcher service.

SQUID-ACCESS

The logging channel for processed HTTP-requests in SQUID-ACCESS-LOG format.

CEF-HTTP

Logging channel for processed HTTP requests in CEF format.

6.2. EtherSensor Watcher stats settings

In the process of message detection EtherSensor allows you to accumulate various connection statistics: MAC address of the interface on which the connection was captured, time of creation, time of connection termination, IP addresses and ports of connection, the amount of data transferred from the client to the server and back, the application layer protocol used (HTTP, ICQ, SMTP, POP3...), etc.

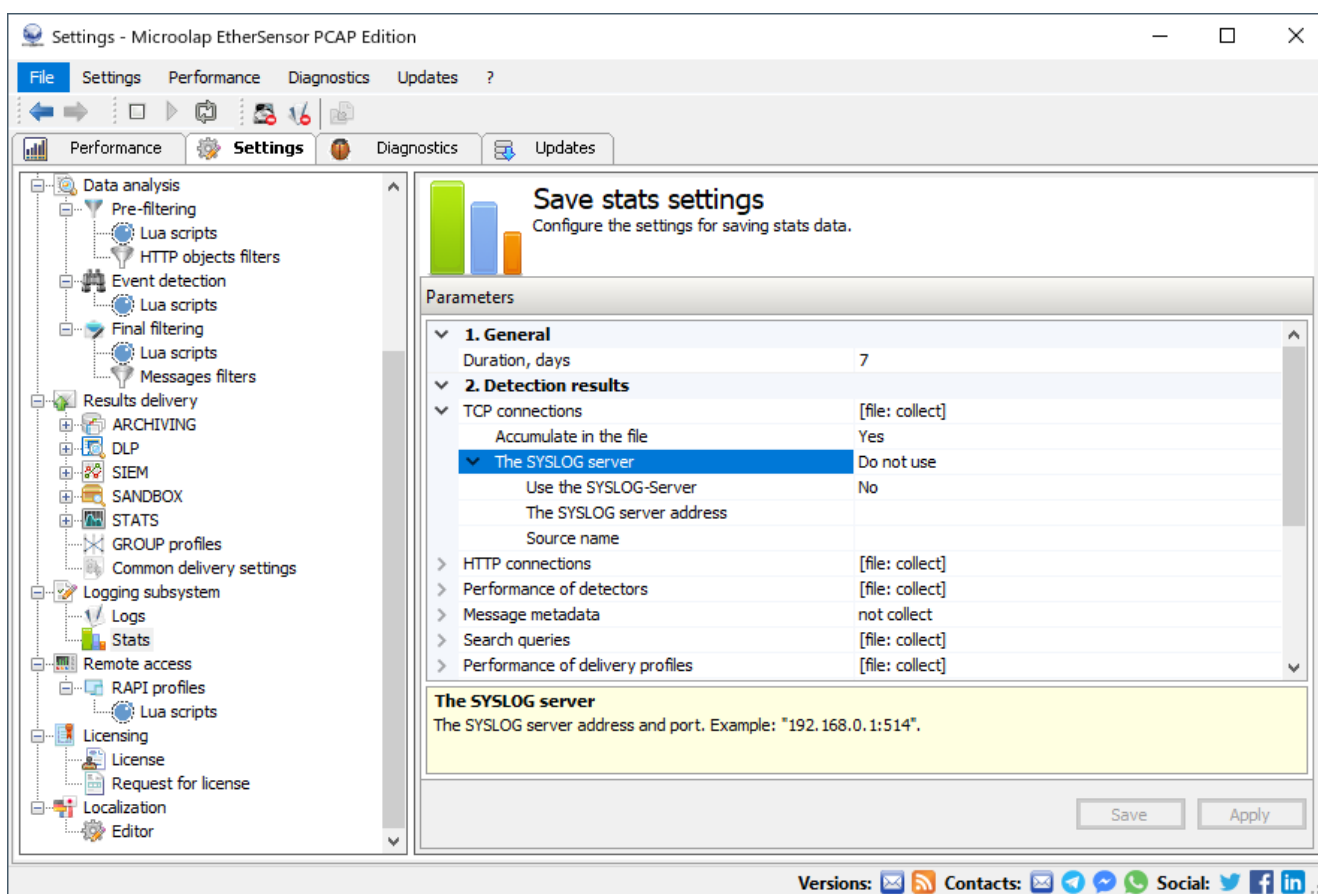


Fig.59. Setting the parameters of saving statistics.

The obtained statistics are accumulated either in the local directories intended for this, or can be sent to the syslog server.

1. General

Duration, days

The number of days during which stats data are to be accumulated: from 0 to 62 days. A value of 0 means that the accumulation of stats has been suspended.

2. Detection results

TCP connections

It is accumulated in the data\statistics\YYYY-MM-DD\sessions directories in CSV format. Includes the following parameters for monitored connections:" - the MAC address of the interface where the connection was captured" - the connection creation and termination time" - the IP-addresses and ports of the connections" - the amount of data sent from client to server and from server back to client" - the protocol used over TCP/IP (HTTP, ICQ, SMTP, POP3...) - other options (see documentation).

Accumulate in the file

Enables/disables the accumulation of EtherSensor stats data.

Use the SYSLOG-Server

Enables/disables SYSLOG server use to log EtherSensor events.

The SYSLOG server address

The SYSLOG server address and port. Example: "192.168.0.1:514".

Source name

Used to parse messages on the SYSLOG server side. Example: ethersensor_syslog_sessions (ASCII symbols only).

HTTP connections

It is accumulated in the data\statistics\YYYY-MM-DD\hosts directories in CSV format . Includes the following parameters for monitored connections:" - the MAC address of the interface, where the connection was captured" - the IP addresses and ports of the connection" - the DNS-name of the server to the connection was established.

Accumulate in the file

Enables/disables the accumulation of EtherSensor stats data.

Use the SYSLOG-Server

Enables/disables SYSLOG server use to log EtherSensor events.

The SYSLOG server address

The SYSLOG server address and port. Example: "192.168.0.1:514".

Source name

Used to parse messages on the SYSLOG server side. Example: ethersensor_syslog_sessions (ASCII symbols only).

Further the similar settings of statistical recipients are omitted.

Performance of detectors

It is accumulated in the data\statistics\YYYY-MM-DD\detectors directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - the name of the detector" - the detection status" - the number of detected messages.

Message metadata

Accumulates in the data\statistics\YYYY-MM-DD\messages directories in XML format and includes the metadata of the reconstructed messages:"- the service headers of the protocol via which the message was sent"- metadata generated by EtherSensor when processing messages (X-Sensor headers ...)"- Headers From, To, Cc, Bcc, Subject.

Search queries

It is accumulated in data\statistics\YYYY-MM-DD\squeries directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - the IP address and port of the search query source" - the DNS-name of the search service" - the search query phrase.

Performance of delivery profiles

It is accumulated in the data\statistics\YYYY-MM-DD\transports directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - the name of the delivery profile" - the protocol used to send the message" - the message sending status.

TCP connections processed by EtherSensor agents

It is accumulated in the data\statistics\YYYY-MM-DD\agents directories in CSV format. Includes the following parameters for monitored connections:" - the MAC address of the interface where the connection was captured" - the time of the connection establishing" - the IP addresses and ports of the connection" - the name of the process that created the connection" - the name of the user with whose rights the process was launched and the connection created" - Other parameters (see documentation.).

3. Counter values

Resource usage counters

It is accumulated in the data\statistics\YYYY-MM-DD\performance directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - CPU utilization (current, average, peak)" - RAM utilization (current, average, peak)" - system threads usage" - system objects descriptors usage(files, events, etc.)" - general resources utilization parameters (CPU, RAM).

ICAP server counters

It is accumulated in the data\statistics\YYYY-MM-DD\icap directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - the number of connections to the ICAP-server" - the number of processed requests (GET, POST, PUT).

Traffic processing counters on NICs

It is accumulated in the data\statistics\YYYY-MM-DD\interfaces directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - counters of processed packets" - counters of processed TCP connections.

Detecting connections by protocol

It is accumulated in the data\statistics\YYYY-MM-DD\parsers directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - counters of processed TCP connections by protocol (SMTP, POP3, HTTP, FTP...).

Analyser cache counters

It is accumulated in the data\statistics\YYYY-MM-DD\cache directories in CSV format and includes the following parameters:"- the timestamp of the event;"- counters of processed objects of EtherSensor Analyser service cache.

Messages detection counters

It is accumulated in the data\statistics\YYYY-MM-DD\analysers directories in CSV format and includes the following parameters:"- the timestamp of the event"- counters for detecting messages by EtherSensor Analyser service.

Filters counters

It is accumulated in the data\statistics\YYYY-MM-DD\filter directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - EtherSensor Analyser service filters counters (RAW-filter, message filter).

Disk quota counters

It is accumulated in the data\statistics\YYYY-MM-DD\quotas directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - counters of disk quota usage.

Results delivery counters

It is accumulated in the data\statistics\YYYY-MM-DD\quotas directories in CSV format. Includes the following parameters for monitored connections:" - the timestamp of the event" - counters of disk quota usage.

7. Remote management and monitoring of EtherSensor

The EtherSensor RAPI service is responsible for remote monitoring and management of the EtherSensor server. For this purpose, it uses pre-configured profiles²³¹.

EtherSensor RAPI service profiles provide independent and isolated from each other http2 web services.

Each profile is associated with a specific user account on the local system on which EtherSensor is installed and has exactly the same amount of rights that this user has.

The entry point for each profile EtherSensor RAPI is a Lua script to which an HTTP request is sent, and the result of the profile is an HTTP response generated in accordance with the script developer's intent.

The functionality of the Lua script of a particular profile EtherSensor RAPI is limited only to the current implementation of the Lua language.

Also, the Lua-script of any EtherSensor RAPI profile can use the entire API implemented for the stage of event detection and analysis⁽⁸¹⁾.

EtherSensor RAPI service configuration file

The configuration of the EtherSensor RAPI service is contained in the XML file `rapi.xml` located in the common configuration directory `Microolap EtherSensor [INSTALLDIR]\config`.

Command line parameters

The service EtherSensor RAPI during the installation procedure Microolap EtherSensor is installed as a Windows service configured to start automatically. However, if necessary, it can be run as a Windows application `sensor_rapi.exe` with the following command line parameters:

/process

Run the process `sensor_rapi.exe` as Windows Win32 process (can be used for debugging).

/service

Run as a Windows service.

/config

Save the default service configuration.

7.1. EtherSensor RAPI profiles settings

Each profile of EtherSensor RAPI provides independent http2 web service. The number of profiles EtherSensor RAPI can be as large as you want.

All http2 web services defined by profiles are isolated from each other and perform only those functions that define the associated Lua scripts. In this way, the administrator can organize remote access to the EtherSensor server based on any of his own role models.

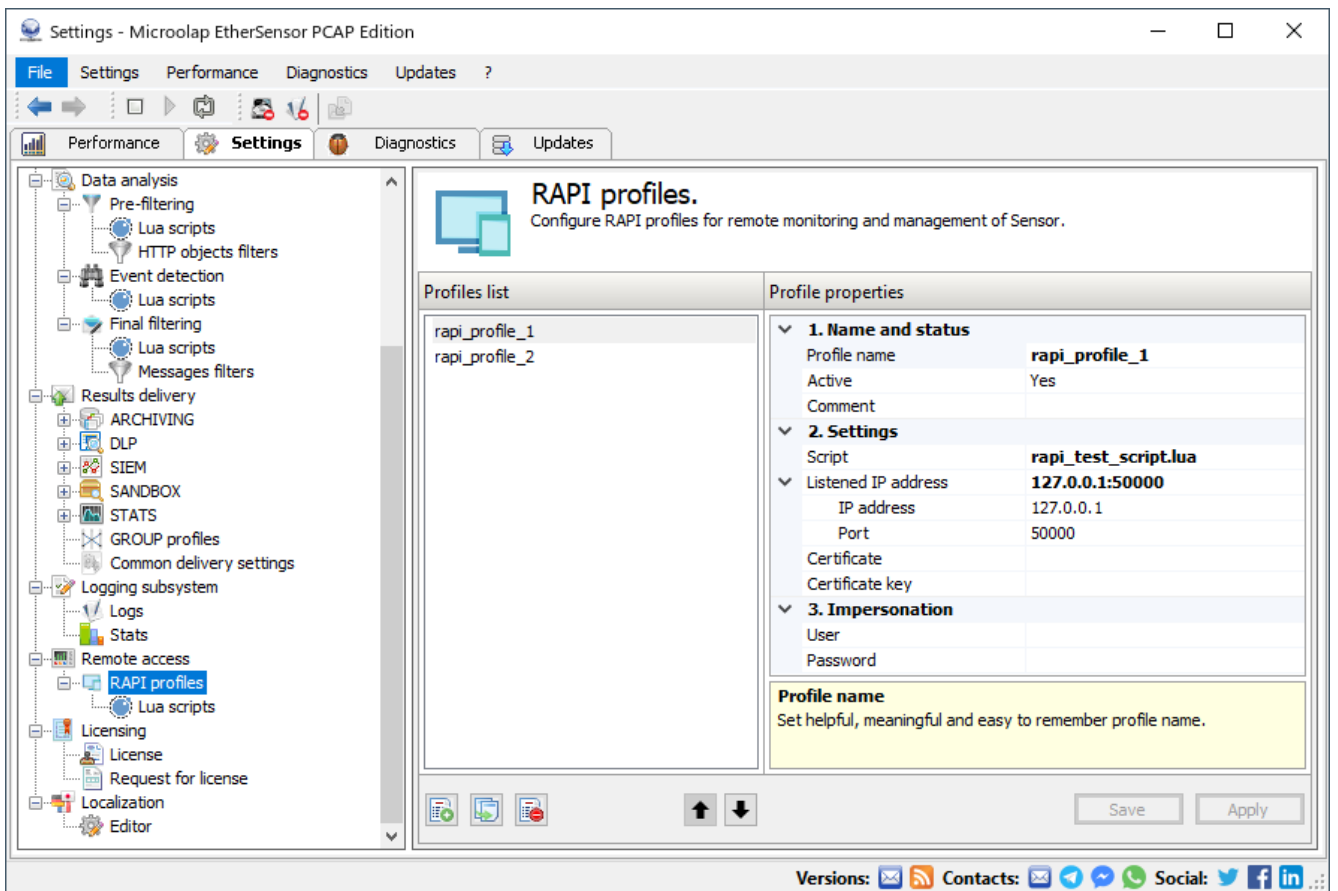


Fig.60. Settings of EtherSensor RAPI service profiles.

If you need to delete or create EtherSensor RAPI profiles, use the **New**, **Clone** and **Delete** buttons:

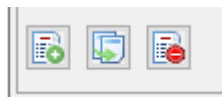


Fig.61. Buttons of creating, deleting and cloning EtherSensor RAPI profiles.

EtherSensor RAPI parameters settings:

1. Name and status

Profile name

Set helpful, meaningful and easy to remember profile name.

Active

RAPI profile is not used if it is disabled.

Comment

RAPI profile description.

2. Settings

Script

A script bound to the profile, which is the entry point for processing HTTP requests at the listening IP address.

Listened IP address

Configure a local address to accept HTTP2 connections.

IP address

Configure a local address to accept HTTP2 connections.

Port

Configure a local address to accept HTTP2 connections.

Certificate

To create secure SSL connections, use your own certificate file. Certificates: *.crt and *.cer, keys - *.key and .pem.

Certificate key

To create secure SSL connections, use your own certificate file. Certificates: *.crt or *.cer, keys - *.key or .pem.

3. Impersonation

User

The name of the local OS user to impersonate HTTP requests processing flow.

Password

The password of the local OS user to impersonate HTTP request processing flow.

The Lua scripts of the profile can be edited using any text editor (scripts can be found in the installation directory [INSTALLDIR]\scripts\rapi) or directly in the management console window **Lua scripts**.

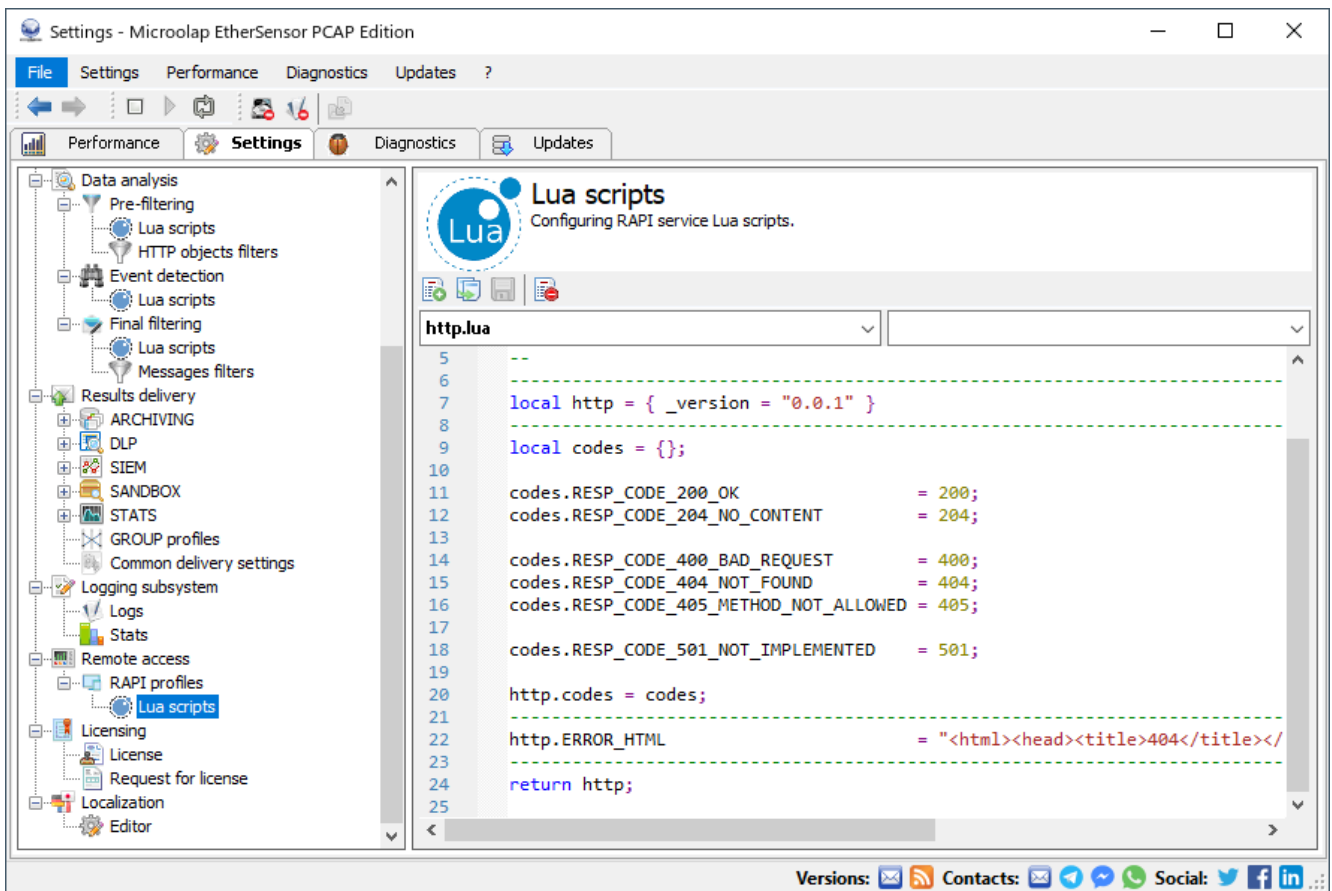


Fig.62. EtherSensor RAPI script editing.

8. Microolap EtherSensor update service

EtherSensor Updater update service (process sensor_updater.exe) is designed for automated download and installation of update files and patches. EtherSensor Updater is a part of developer's service to support Microolap EtherSensor in normal mode and update its functions.

Functions of EtherSensor Updater>:

- Check for updates and patches for Microolap EtherSensor, download and install them.
- Promptly update Microolap EtherSensor license file when it is changed.
- Create an archive copy of the previous version of the product and put all the necessary files into it.
- Restore the previous version from the archive if an update installation error occurs.

You can specify an update check interval or use the default settings. EtherSensor Updater also allows you to flexibly set up a network connection, supports work with a proxy server.

How EtherSensor Updater works:

The update service checks both for new versions of EtherSensor and for updated versions of licenses. To do this, at the time intervals specified in the settings, it establishes a connection with the update server and sends it information about the version and license Microolap EtherSensor.

If newer versions are available for the installed software, the server sends a response file to the service specifying the files to be downloaded and installed.

EtherSensor Updater downloads these files from the update server and places them in the installation queue. During the time specified in the service configuration, these files are started and the software is updated.

EtherSensor Updater installation:

The initial installation EtherSensor Updater is performed by launching the file `ethersensor_updater_X.X.XXXX_x64.msi` from the distribution kit. In future the service will be updated by itself when new versions are released.

To install into a directory other than the default directory, you should use the `msiexec.exe` utility of Windows.

Example:

```
msiexec.exe /i ethersensor_updater_4.6.3.12232_x64.msi INSTALLDIR="[ [INSTALLDIR] ]\updater"
```

Technical requirements:

- EtherSensor Updater service must have Internet access on port 80 or 443 to the servers `license.microolap.com` and `kpps-downloads.microolap.com`.
- If a proxy server is used, it must support the correct operation of the SOAP protocol.
- Windows Server 2012, Windows Server 2016 or Windows Server 2019 operating system.

EtherSensor Updater consists of the following parts:

Service EtherSensor Updater (process `sensor_updater.exe`).

Starts in the background and performs all actions necessary to check for available updates and their further installation. In the process, a `status.xml` file is created, which contains information about the current state of the service.

"Updates" section of the EtherSensor management console (application `sensor_console.exe`).

Used to configure and display the service status, a separate tab displays the main update log file.

Note:

1. The management console EtherSensor connects to the service sensor_updater.exe via TCP/IP protocol, port 52076.
2. If the EtherSensor management console detects that the sensor_updater.exe service is not running, no connection attempt is made. If you want to establish a connection to a service running in console mode, you can use the command line parameter /ignore-service-status
- 3 You can also use the /no-connect command line parameter, which prohibits any attempts to contact the service. In this case, the management console EtherSensor receives service status information only via the status.xml file.

Directory [INSTALLDIR]\config

Contains service configuration files (service setup).

Directory [INSTALLDIR]\downloads

When configured by default, files required for EtherSensor update are uploaded to this directory.

Directory [INSTALLDIR]\log

Contains log files and status file:

status.xml

Status file containing information about the current status of the service.

log.txt

The main log of the service.

uupdater_log.txt

Logs of the special program sensor_uupdater.exe, which is called if necessary to install a new version of EtherSensor Updater.

Configuration file EtherSensor Updater

The EtherSensor Updater configuration is stored in the system.xml file located in [INSTALLDIR]\config.

8.1. Microolap EtherSensor update service settings

Current status

The Current status section displays information from the status file EtherSensor Updater concerning the operation of the update service itself.

This section also displays the list of software detected on the sensor EtherSensor, for which it is possible to check for updates, the list of downloaded files (if something is downloaded right now) and the list of ready to install updates.

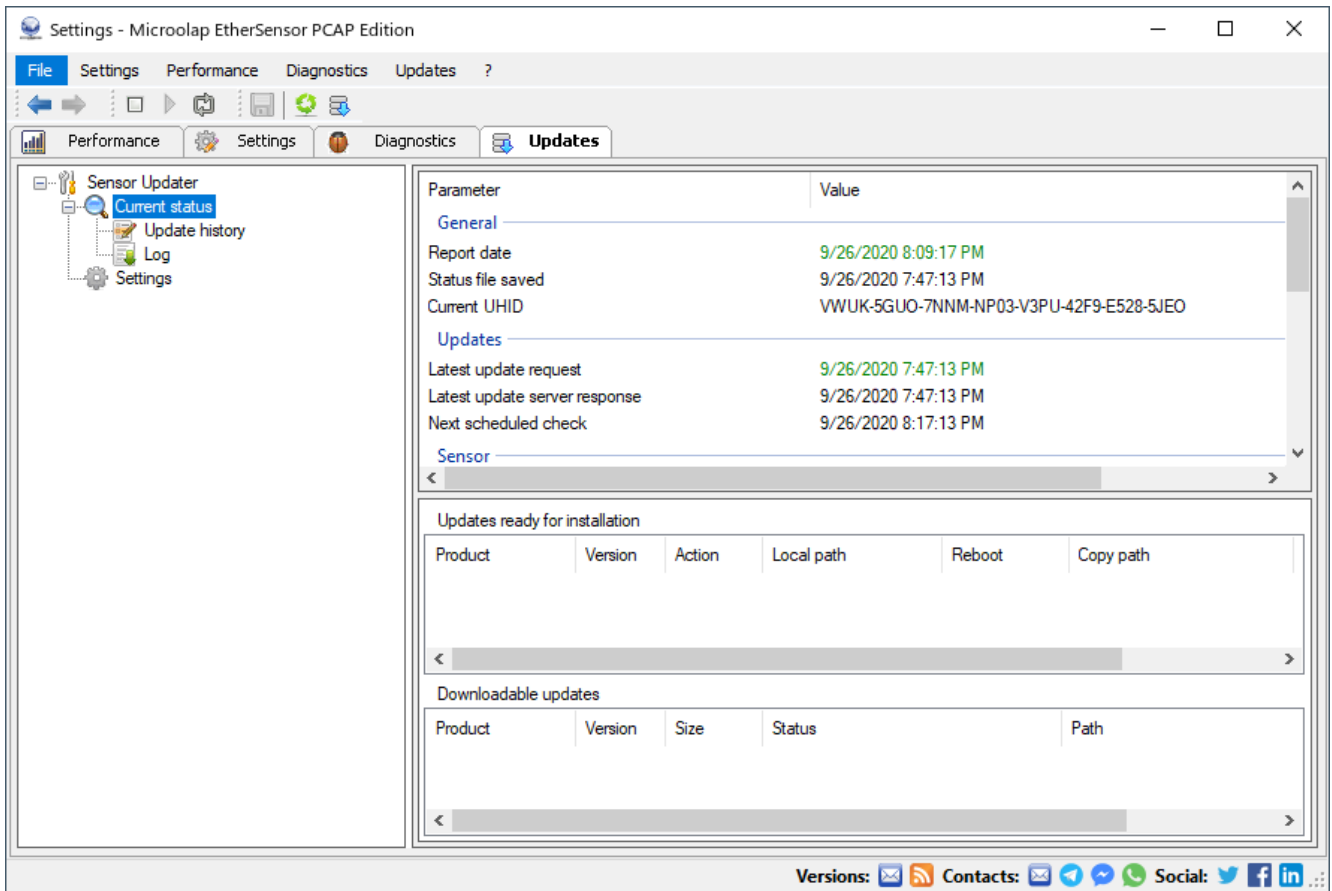


Fig.63. Section "Current status"

Settings, tab Download updates

The Download updates section sets the modes of operation EtherSensor Updater and the frequency of updates check. You can also enable the test mode in this section, where more detailed messages about the service operation are added to log files.

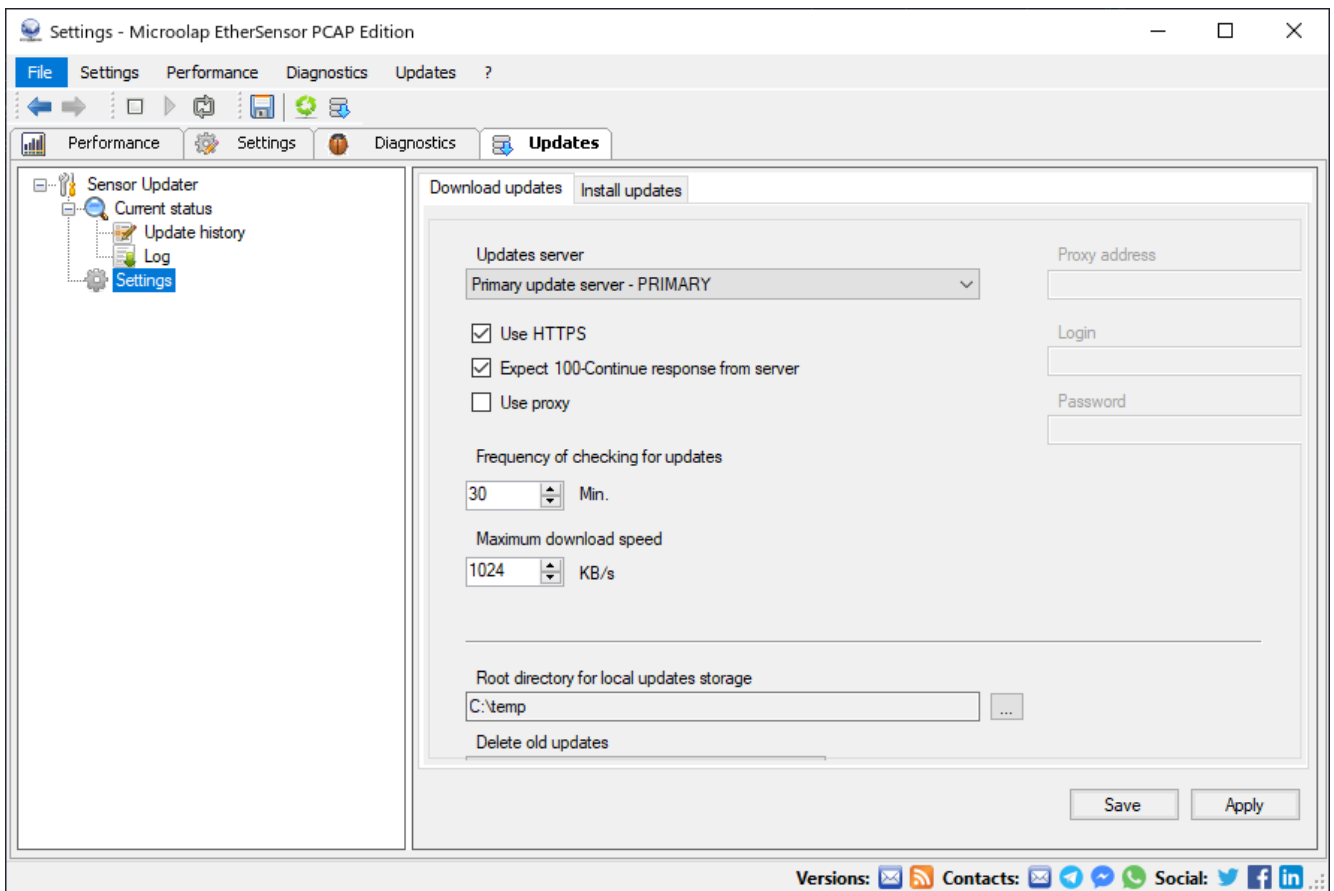


Fig.64. Section "Download updates"

Settings, tab Install updates

In Install updates section you can set the time of automatic updates installation. After installation of updates, OS is rebooted, if necessary.

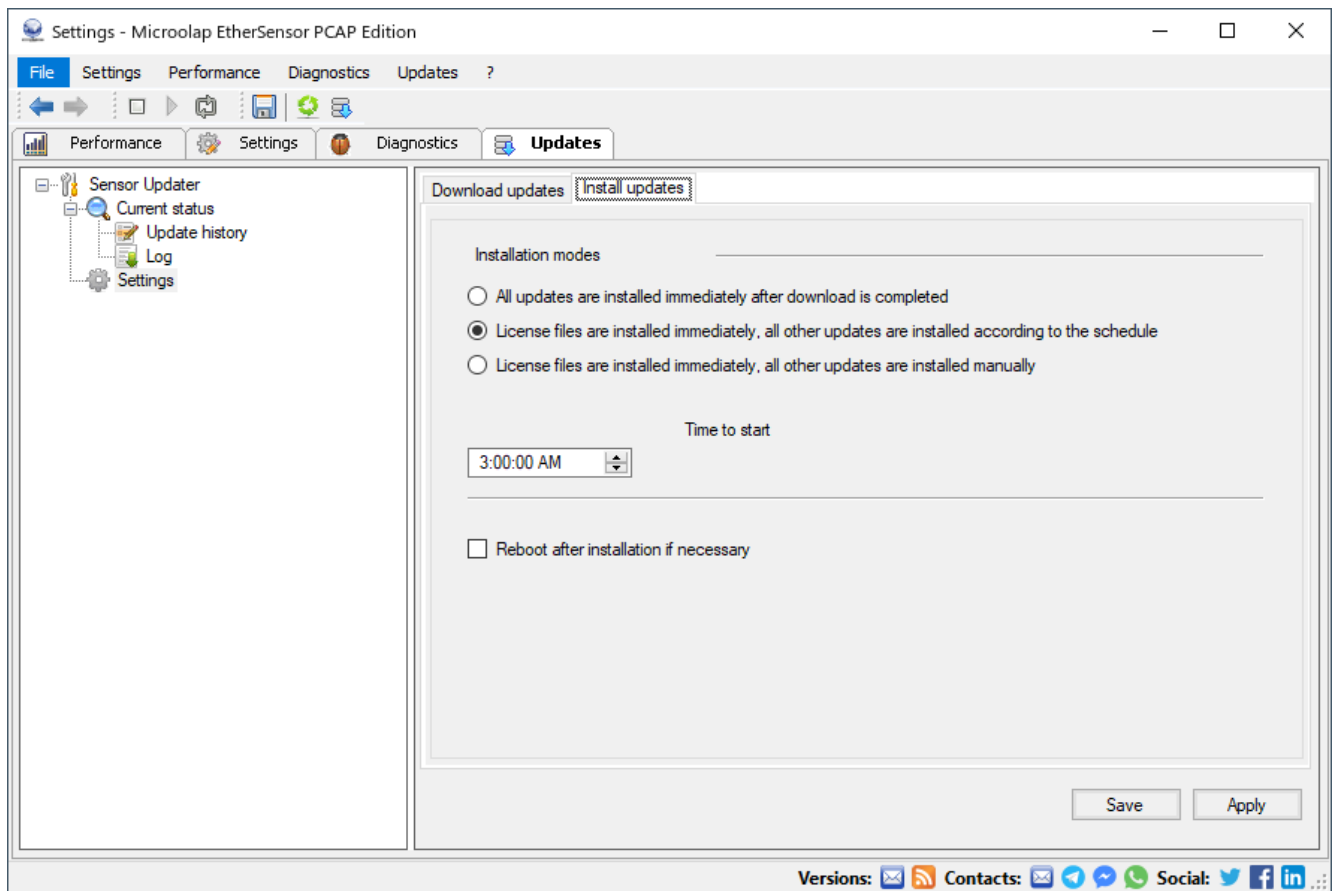


Fig.65. Section "Install updates"

Log

The Log section displays the latest log file entries of EtherSensor Updater.

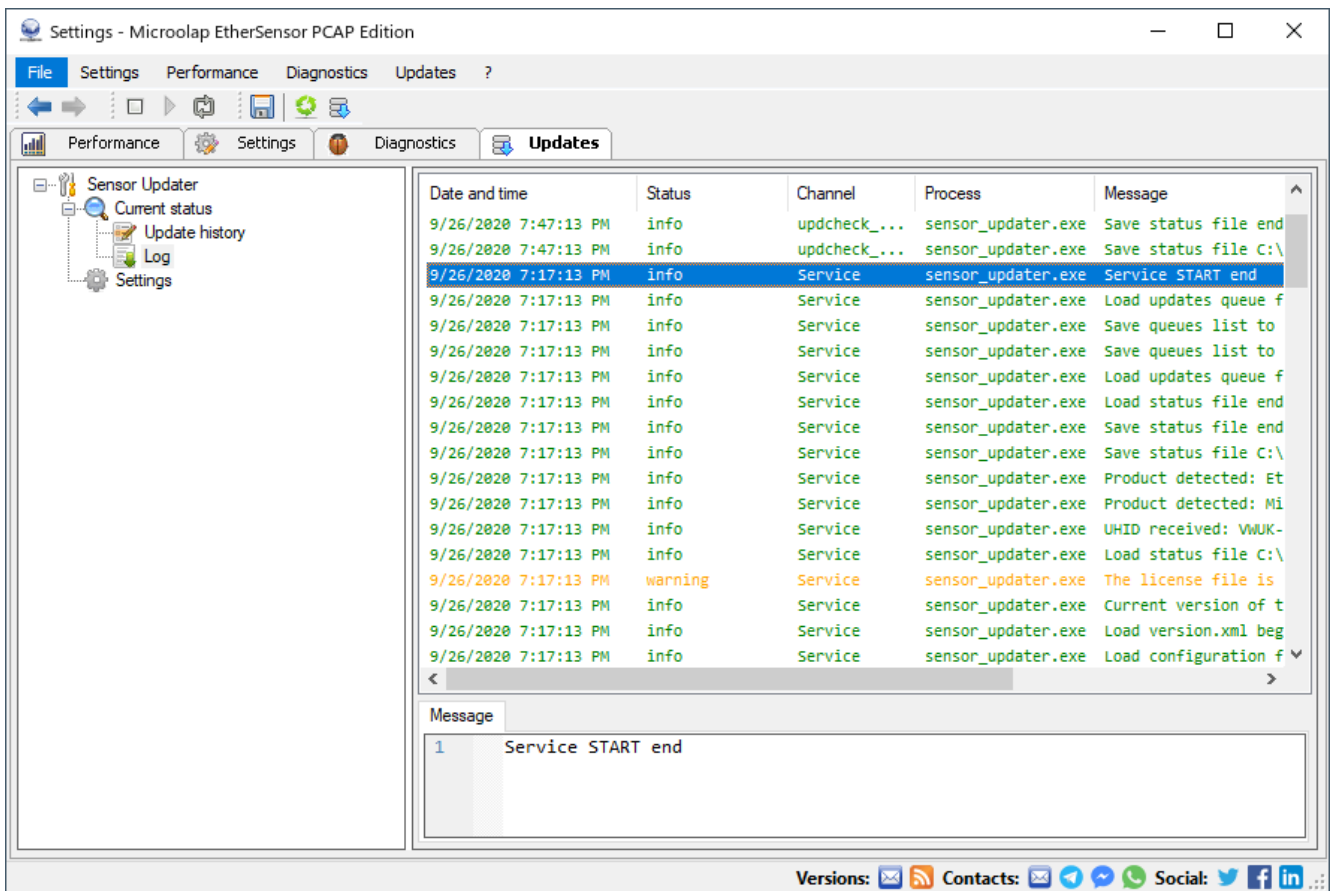


Fig.66. Section "Log"

Update history

The Update history section displays the list of previously installed updates.

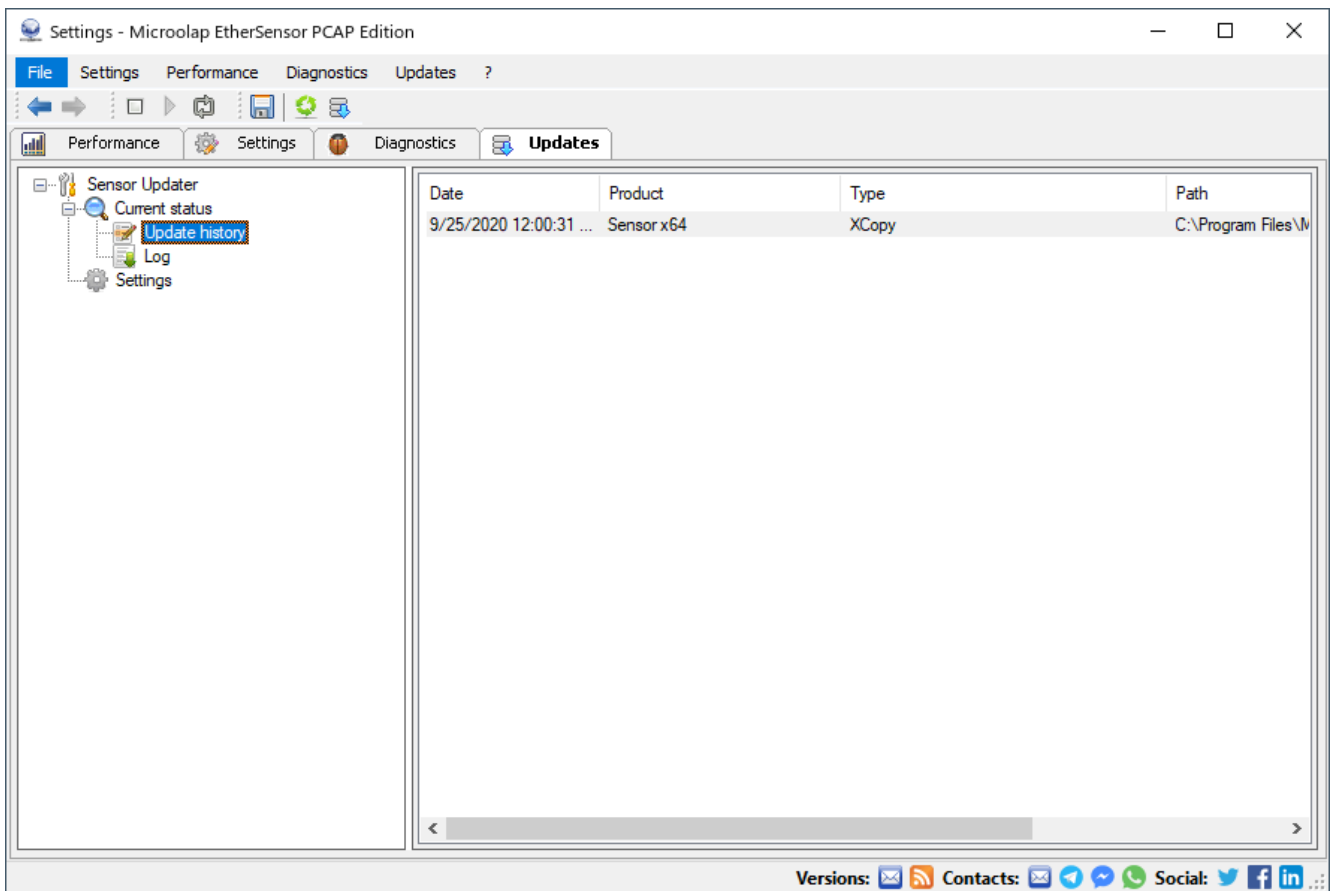


Fig.67. Section "Update history"

9. Sensor routine maintenance

In order to maintain an optimal operating condition of the sensor and to detect problems at an early stage, it is recommended to perform the following operations regularly.

1. Check the sensor event logs for alerts and service errors.
2. Performing backups. Make sure that the sensor configuration file is regularly backed up to an external storage medium.
3. Checking performance indicators of the server on which EtherSensor operates, such as free disk space, CPU utilization, RAM usage. For this purpose, you can use the Windows Performance Monitor tool and the application sensor_console.exe installed in the installation directory EtherSensor. Using the notification mechanism, you can notify employees of the support group EtherSensor about sudden changes or problems with server performance.
4. Checking log files on a network device. The log files should be analyzed for correct operation of the transport service, and the volumes of branching traffic should be evaluated to prevent overloading of its consumers.
5. Checking the availability of captured messages to consumers by the sensor. Make sure that messages generated by the sensor services are delivered to the message consumers' servers. If it

is not possible to deliver messages to consumers, there may be disk space overflow on the sensor (depends on disk quotas set by the administrator).

9.1. Sensor maintenance issues

- How to remove Microolap EtherSensor services manually?

1. In the command line execute the following commands:

```
sc delete "EtherSensorEtherCAP"  
sc delete "EtherSensorICAP"  
sc delete "EtherSensorLotusTXN"  
sc delete "EtherSensorAnalyser"  
sc delete "EtherSensorTransfer"  
sc delete "EtherSensorWatcher"
```

2. Run the regedit utility and remove the registry branches:

```
HKLM/System/CurrentControlSet/EtherSensorAnalyser  
HKLM/System/CurrentControlSet/EtherSensorEtherCAP  
HKLM/System/CurrentControlSet/EtherSensorICAP  
HKLM/System/CurrentControlSet/EtherSensorLotusTXN  
HKLM/System/CurrentControlSet/EtherSensorTransfer  
HKLM/System/CurrentControlSet/EtherSensorWatcher
```

3. Delete old version files.

4. Reboot the server.

- Why many duplicates in messages?

1. Web services can send forms more than once, for example when saving drafts or adding attachments.

2. In some complex network conditions (for example, when using a chain of proxy servers or load balancers) it is possible that the sensor sees the same connection by several interfaces simultaneously. In this case, you should use the check-md5 filter, this will allow you to decide whether or not to process this message again.

- I see traffic from the mirror port by another sniffer, but your sensor does not catch anything.

- Traffic counters are increasing, but the sensor is not intercepting anything.

- In the capture record, return packets from HTTP services are not visible, i.e. packets from client IP to remote server/port are visible, but responses from remote server/port are not. At the same time, ACK, FIN/ACK packets are sent from the client, i.e. this means that these are working sessions with real traffic that reaches the client.

Checking is done in the following order:

1. Make sure the services are up and running.
2. Check the IP filtering rules, remembering that services must be restarted to apply the rules.

3. Check the traffic counters. The amount received must not be equal to the amount Rejected.
4. Check for captured data in the data subdirectory of the EtherSensor installation directory. If filters are used, also check the contents of [INSTALLDIR]\data\filter.
5. Check the directory [INSTALLDIR]\data\result - are there any captured messages that have not been sent?
6. Check the logs and counters for errors. If they are not, the services work fine.
7. Check the mirror port settings. For example, on Cisco devices either RX or TX packets are mirrored by default. You need to specify the keyword "both" in the configuration:

```
monitor session 1 source interface <interface-id> both
```
8. Check whether the profile is configured in the transport service settings, whether the correct profile is specified as the default profile.
9. If you can't find out the reason yourself, contact technical support.

10. Emergency response

Technical failure

Possible types of failure of the technical means on which the Microolap EtherSensor is running are listed below, actions to eliminate them and then restore the sensor.

Unable to start Microolap EtherSensor services if server restarts or stops incorrectly.

The symptom of this alarm is the system message (Error 1053) that the service cannot be started.

If it is not possible to start any EtherSensor service, the following steps should be performed:

- Try to start the services again, and remember that for successful start of services EtherSensor it is necessary to start the service EtherSensor Watcher.
- If the restart of services was not successful, you should contact technical support.

Overflow of disk space allocated for message capture spools.

The symptoms of this emergency are:

- Operating system alerts for lack of free disk space
- Records in the operating system log that there is no free disk space.

The system administrator must determine the reason for the lack of free space. To do this, use the Disk Management snap-in. If the reason is spool overflow ([INSTALLDIR]\data directory), delete or move spool data to a free disk partition. Also, the reasons for overflow can be:

- Unavailability of the server to which the sensor should deliver messages - in this case, you need to troubleshoot communication problems
- Too much traffic being processed by the sensor - in this case additional sensors and load sharing between them are required.

Emergency power-off of the server, which runs EtherSensor.

Once power is restored and the server on which EtherSensor is running is turned on, the system administrator must ensure that all services are running successfully.

If there are deviations from normal start-up, you should review the service logs. If error messages are found there, you should contact technical support.

Breakdown of network connection of the server where EtherSensor operates.

The symptoms of this emergency are:

- The network card of the server on which the sensor operates indicates a mechanical violation of the communication channel
- Return of packets sent via ICMP to the servers of systems-consumers of results EtherSensor does not occur or occurs partially

The system administrator must check and restore the communication channel of the server where EtherSensor operates. Then you should check the network settings in the operating system. After fixing network connection problems, make sure that all services EtherSensor are running.

Unauthorized access to Microolap EtherSensor or OS

Unauthorized tampering with Microolap EtherSensor or the operating system can be determined by the following attributes:

- Windows security log contains entries about unauthorized access attempts to the system
- The system is spontaneously deleting or creating files, starting arbitrary services.

If you find such signs, you must perform the following actions:

1. Disable sensor network access.
2. Stop the operation of Microolap EtherSensor by stopping all services.
3. Remove thereasons of unauthorized intervention in the system.
4. Restore Microolap EtherSensor.

11. What's new

What's new in version 6.1 versus 6.0

Runtime environment:

Windows Server 2012, Windows Server 2016, Windows Server 2019.

Data sources and objects capture:

PCAP files processing:

- [+] Added ability to detect and process protocols using Lua-parsers. Now the processing of new application level protocols is fully defined in the EtherSensor configuration.
- [+] The following protocol parsers in Lua have been added as examples:
 - TCP: MySQL, Postgres, MSSQL (TDS), Oracle
 - UDP: DNS
 - IP: ICMP.

- [+] Added ability to record traffic to PCAP files. Recorded traffic has all metadata obtained during session processing:
 - Protocol type: NDPI attributes, EtherSensor attributes
 - Traffic is bound to the user.

Captured objects analysis:

- [+] The lua detectors became event oriented. In the settings of the detector you can specify in the Settings.EventFilter field, which type of events is processed by this detector.
For example: EventFilter = "raw/mysql" or EventFilter = "raw/xmpp".
- [+] The following Lua scripts for the event detection stage have been implemented:
 - Messengers:
 - l2-xmpp.lua
 - l2-slack.com.lua
 - l2-websocket-slack.lua
 - Operations with files:
 - l2-ftp.lua
 - Detectors of exchange events between the client and the DBMS server:
 - l2-mysql.lua
 - l2-mstds.lua
 - l2-postgres.lua
 - l2-oracle.lua
 - PCAP-files processing:
 - l2-pcap.lua
 - SIP request processing:
 - l2-sip.lua
 - Processing DNS queries:
 - l2-dns.lua
 - ICMP-events processing:
 - l2-icmp.lua.

Delivery of analysis results to consumer system:

- [+] Added integration with Dataplan (Angara) analytical platform.
- [+] Updated libcurl.dll module to version 7.73.0.0.
- [+] Updated libssh2.dll module to version 1.9.0.0.

Management Console:

- [+] Configuring traffic capturing and processing of PCAP files became more detailed:
 - Adjustment of built-in parsers
 - Adjustment of Lua parsers
 - Configure traffic recording in PCAP files.
- [+] Packet filters now use syntax of tcpdump and libpcap library filters syntax to generate BPF (Berkeley Packet Filters).
Please note:
In version 6.1 the format of packet filters has changed. If you are using packet filters, they must be converted to the new format (tcpdump/libpcap) manually. The old filters are saved in the \backup\DD.MM.YYYY\ config directory.
- [+] Lua detectors are grouped according to the types of events being processed.

12. GUI localization

The user interface can be localized into your language in a very short time directly in the management console of EtherSensor PCAP Edition.

If you click on the **Localization--Editor** node, the GUI item localization window will open:

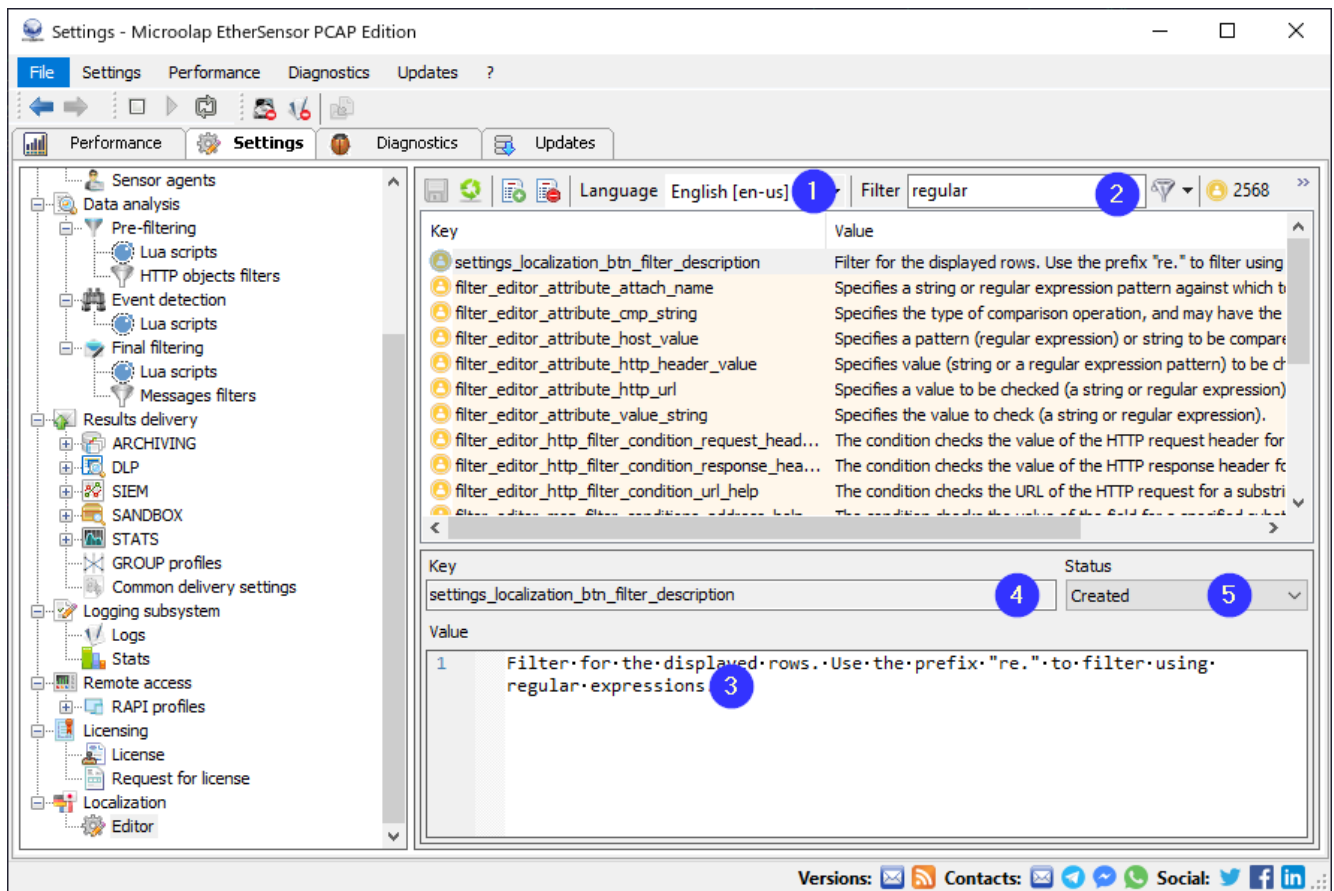


Fig.68. Localization--Editor node window

Text strings of GUI elements are stored in XML language files, you can edit them directly in the node window **Editor**.

To save the modified text in a language XML file, press Ctrl-S. To see the updated GUI elements, press Ctrl-R. The same actions can be performed using the buttons on the toolbar to the left of the **Language** (1) dropdown.

Basic controls of the localization window:

1. Language:

Editable language XML file switcher (not to be confused with the GUI language switchable by Settings - Language). The labels on this element are determined by the name and content of the language XML files (see below).

2. Filter:

Filter to search for string values. The search can take place in the columns **Key**, **Value** and **Status** or all at once. The search area is set in the dropdown next to the funnel icon to the right of the **Filter** element.

3. Value:

Window to edit the field *Value*. The content of this window is used when displaying an interface element.

4. Status:

Interface element status switch. The names of the statuses can be changed at your discretion here. Each status has its own background color for the string of the interface element. At the top right of the localization interface window, there are counters of elements with different statuses, which allows you to estimate the amount of work remaining.

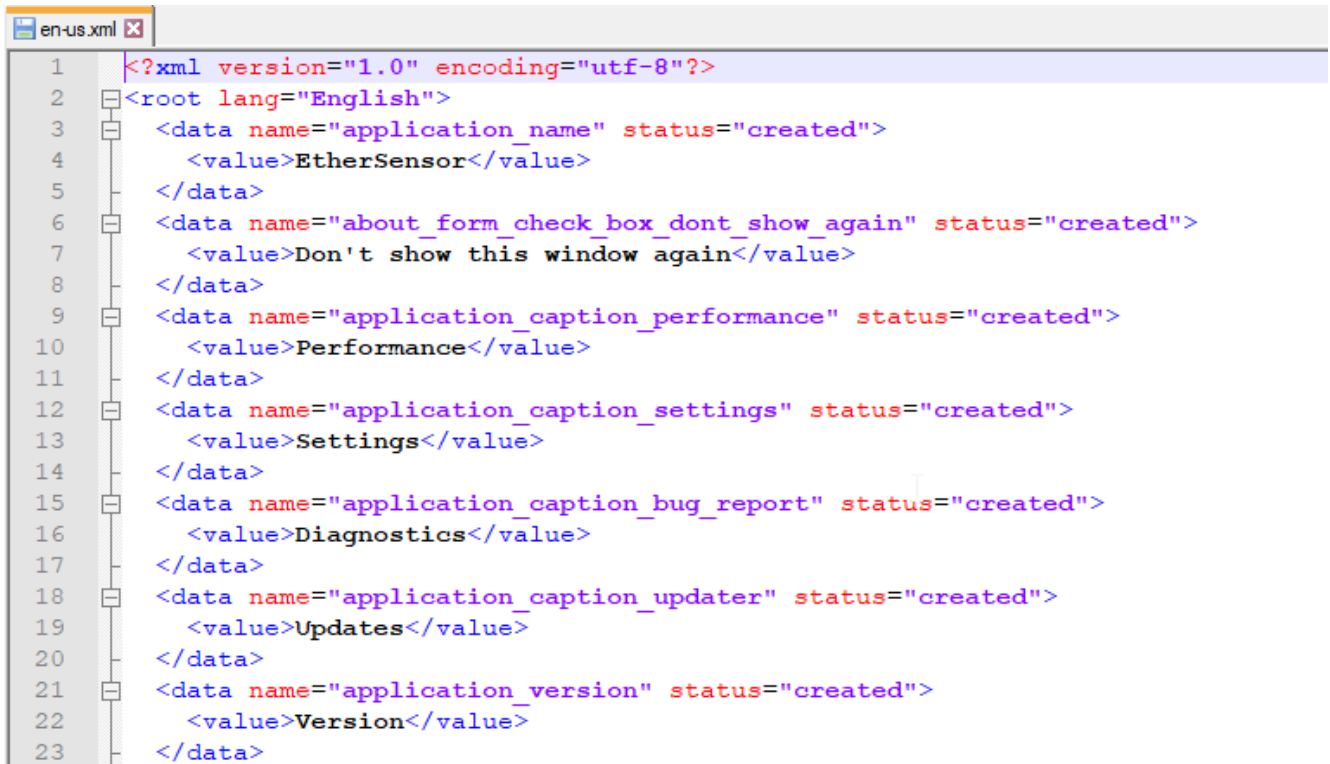
5. Key:

GUI text elements are stored in XML files as "key-value" pairs, where **Key** field is a unique key and **Value** field contains the text displayed in the GUI element.

Language XML files

The language files are located in the lang subdirectory of the Microolap EtherSensor installation directory and are named by the principle <language>-<culture>.xml, e.g. en-us.xml, pt-br.xml and so on.

Inside, the language XML file looks like this:



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <root lang="English">
3   <data name="application_name" status="created">
4     <value>EtherSensor</value>
5   </data>
6   <data name="about_form_check_box_dont_show_again" status="created">
7     <value>Don't show this window again</value>
8   </data>
9   <data name="application_caption_performance" status="created">
10    <value>Performance</value>
11  </data>
12  <data name="application_caption_settings" status="created">
13    <value>Settings</value>
14  </data>
15  <data name="application_caption_bug_report" status="created">
16    <value>Diagnostics</value>
17  </data>
18  <data name="application_caption_updater" status="created">
19    <value>Updates</value>
20  </data>
21  <data name="application_version" status="created">
22    <value>Version</value>
23  </data>
```

Fig.69. The content of the GUI language XML file.

The value of attribute "lang" of tag "root" together with the file name are used to display the current language file in the element **Language**.

In order to start localizing the GUI into a new language, you need to do the following:

1. Copy an existing language file, e.g. en-us.xml, to a new one, e.g. pt-br.xml
2. Open the newly created pt-br.xml file with any text editor and fix English with, for example, Português brasileiro.
3. Close the file and restart the sensor_console.exe management console. A new language and a new file will appear in the **Language** dropdown.

Now you can start editing GUI text elements in a pt-br.xml file.

Editing GUI elements

To edit the text of a GUI element, click on the window **Value**, change the text, press Ctrl-S - the new value of the element will be saved in the language file, then Ctrl-R - the new value of the element will be read from the language file and displayed in the corresponding element. The same actions can be performed by buttons on the toolbar to the left of **Language**.

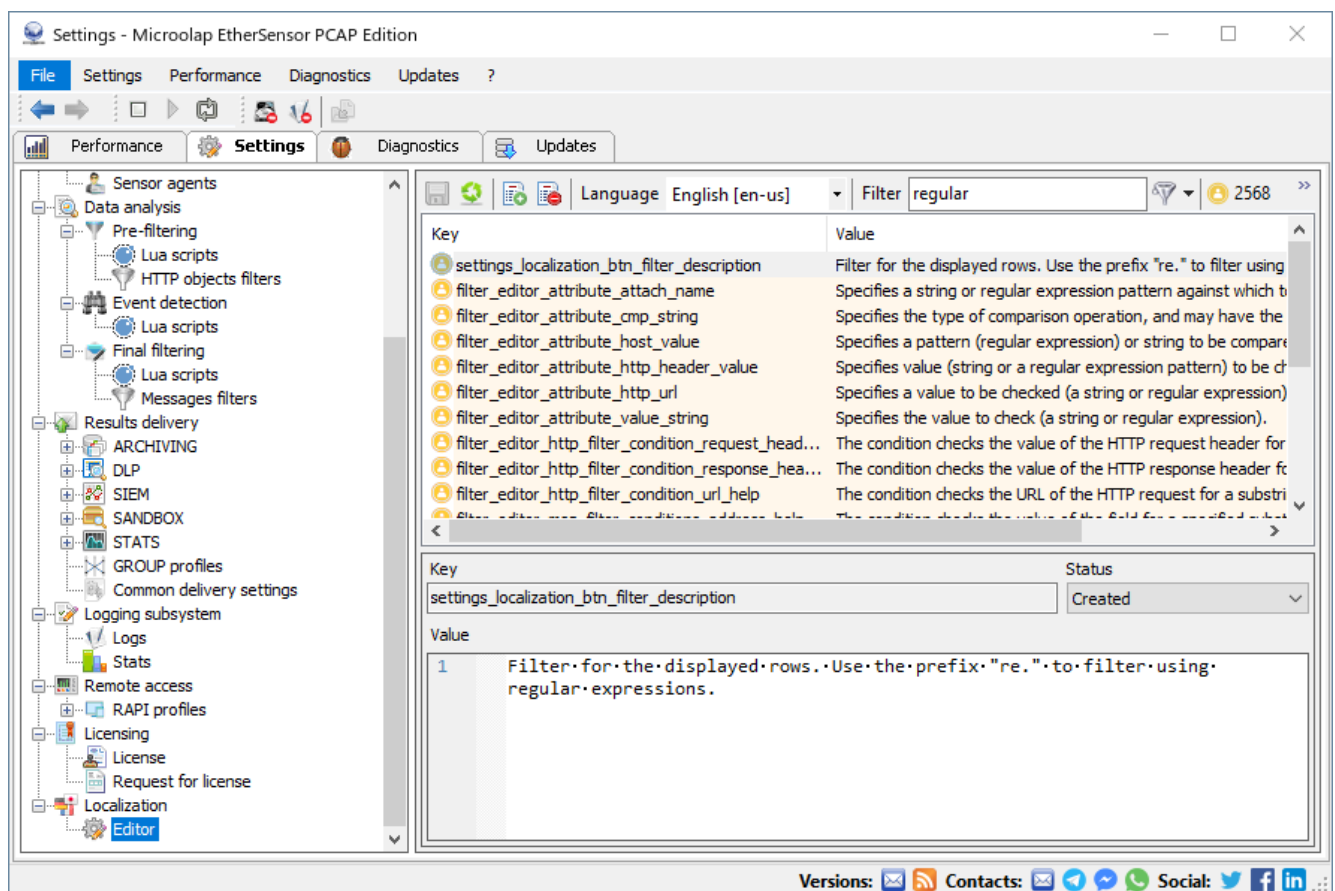


Fig.70. Editing the text of the GUI element.

Remember to change the status of the element immediately, if necessary.

Simultaneous work with two GUI languages.

In order not to switch between the two languages of the GUI language pair constantly, it is better to have two management console windows open at the same time. To do this, do the following:

1. Create a folder to work with the first language, for example, en-us.xml. Name it C:\EtherSensor Agent-en-us.
2. Create a folder to work with a second language, such as pt-br.xml. Name it C:\EtherSensor Agent-pt-br.
3. Copy all files from the installation directory EtherSensor into these folders, leaving only en-us.xml in the first folder in the lang subdirectory and only pt-br.xml in the second folder.

You can now open two windows of the management console application simultaneously, one from C:\EtherSensor Agent-en-us and the other from C:\EtherSensor Agent-pt-br, and edit the files to avoid any errors or confusion.

When you are done, move the pt-br.xml file to the lang subdirectory of the EtherSensor installation directory, and simply delete these two new directories.

Using regular expressions when searching in the fields of elements (Filter).

If the string with the sought text in the element *Filter* is preceded by a combination of characters re. ("r", "e" and "."), then it will be interpreted as a regular expression.

Example:

re.[0-9]\$ - Find all elements where the value ends in a number

re.: \$ - Find all elements where the value ends in a "colon space"

re.^.{4,6}\$ - Find all elements that consist of strings from 4 to 6 characters long.

If you send us your language file, we will add it to the distribution kit of the next release of EtherSensor. In addition, this file will be added by the update system to all currently running instances of the program.

13. Licensing Microolap EtherSensor

Microolap EtherSensor PCAP Edition of the current version is distributed free of charge, no license file is required.

How to purchase Microolap EtherSensor license

Supplier: Microolap Technologies

Email: sales@microolap.com

Phone: +7 495 748 8105

WWW: <https://www.microolap.com/about/contacts>

How to buy Microolap EtherSensor

To get a quote email Microolap Technologies at sales@microolap.com.

Purchase from partners: <https://www.microolap.com/order/resellers/>.

Purchase online on www.microolap.com.

13.1. License file

Information about Microolap EtherSensor licenses is contained in the file `license.licx`, which must be located in the root directory of the installation.

The license contains information about the modules licensed for this installation, the expiration date of the license for each module, the date of expiration of the subscription for updates and other data. The content of the license file is open for viewing, but is protected by a checksum and a digital signature. The license is issued for installation of EtherSensor on specific hardware, for which a special check is provided.

The license status can be viewed through the management console (section *Licensing*).

Depending on the presence of the `license.licx` file in the installation directory and license types for EtherSensor, all modules work in one of the modes:

Demo mode

All messages are detected in the demo mode, but only every fifth message is sent to the results delivery service in its original form.

There is no time limit for EtherSensor in demo mode: you can work on deploying it on your organization's network as long as it takes.

To switch EtherSensor to full mode, place the correct `license.licx` file in the EtherSensor installation directory.

Full mode

In full mode EtherSensor operates for the period of time specified explicitly in the license file, receiving and installing available updates.

At the end of the specified time period:

- If a permanent license has been purchased, the installation of updates is discontinued until the `license.licx` file with updated license data is received. In this case EtherSensor continues to work in full mode.
- If the license for EtherSensor was purchased by subscription for a fixed term, the system switches to demo mode.

When starting in demo mode version EtherSensor, released after the expiration date of the

subscription for updates, a message about the latest version, which is allowed by the current license, will be written to the log.

Microolap EtherSensor makes a log entry about the license expiration. The license file is automatically transferred to the running sensor through the update system.

License request

The license.licx file is generated by the Microolap EtherSensor vendor based on the license request.licx file created on a specific instance of the sensor hardware.

When transferring EtherSensor to another server or replacing out-of-order equipment, EtherSensor will recognize the license as invalid. In this case, you should submit a new license request and obtain a new license file license.licx.

Each license is bound to the server hardware (UHID, HardwareID) for which it was created.

Therefore, to avoid problems with UHID changes during operation, before creating the request.licx file, it is necessary to bring all server hardware to the state in which it will be used, i.e. disable all temporary devices (flash cards, USB-HDD, etc.), disable all temporary, virtual or unused network cards, configure MAC addresses, etc.

For more details on the concept of UHID, see the "UHID (HardwareID) of the server" section.

To create the request.licx file, open the **Request for license** section in the management console and fill in the corresponding fields in the window that appears, then click **Save**.

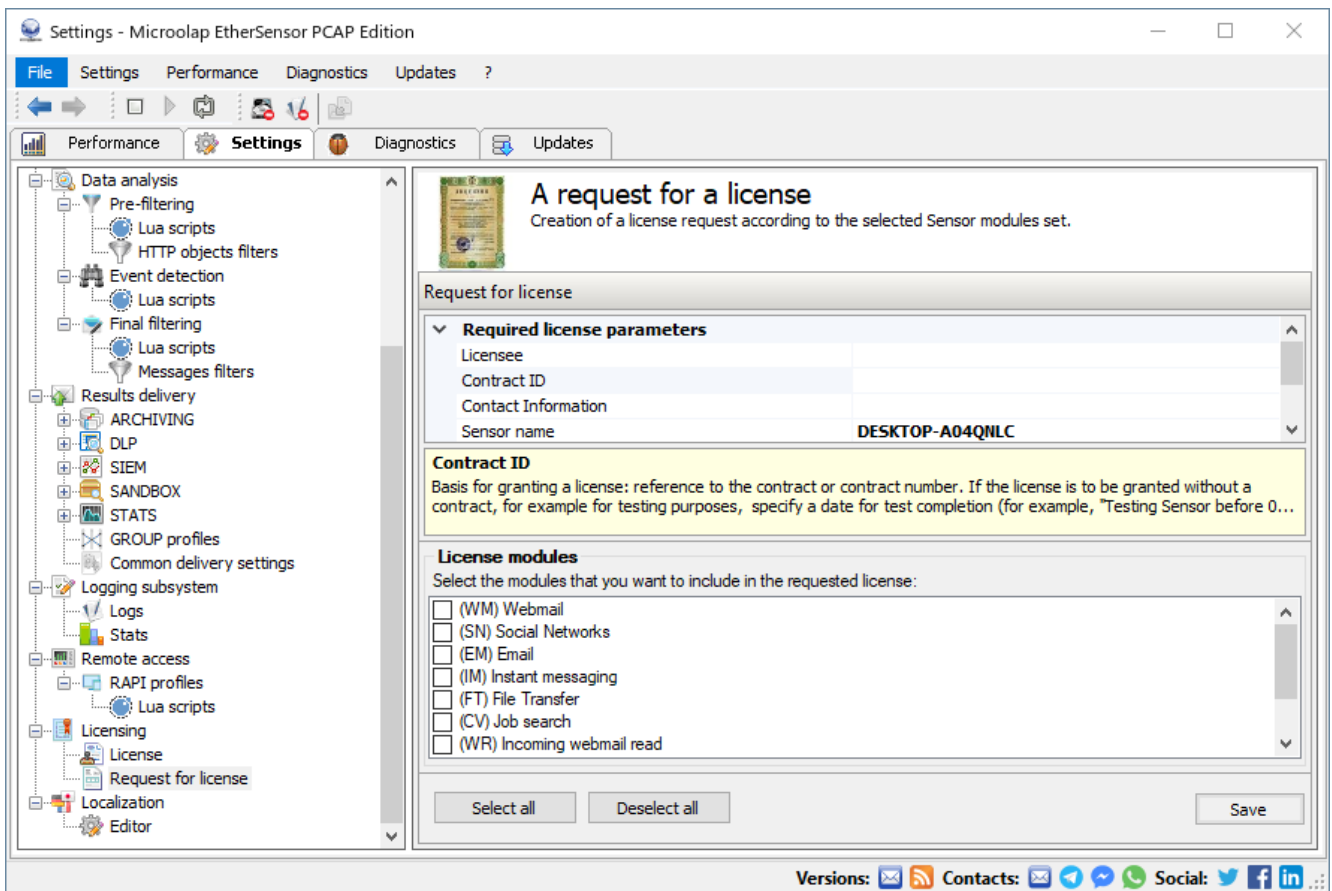


Fig.71. License request form.

The license request file will be saved in the installation directory with the name request.licx.

Required license parameters

Licensee

Name of the organization-licensee (to which the license will be issued).

Contact Information

Contact information for communication with the representative of the licensee (name, email, phone).

Contract ID

Basis for granting a license: reference to the contract or contract number. If the license is to be granted without a contract, for example for testing purposes, specify a date for test completion (for example, "Testing EtherSensor before 01.12.2011").

End date of registration to receive updates

Usually this is the date before which payment is to be made under a contract or an additional agreement (time of day is not taken into account).

Beta License

Select "Yes" if, during the upgrade, the license must allow for installation of the latest beta versions of EtherSensor. If "No" is selected, updates will include stable versions only.

Sensor name

The name of EtherSensor server. Used to distinguish between different instances of EtherSensor within a single organization. Here, you can enter the host name or any other provisional name for the sensor that uniquely identifies it within the licensee organization.

License modules

Modules EtherSensor to be licensed.

After pressing **Save** button, the request.licx file will be created in the root directory of the EtherSensor installation. It should be sent to the EtherSensor provider to create a new license.licx file based on it.

How to buy Microolap EtherSensor

To get a quote email Microolap Technologies at sales@microolap.com.

Purchase from partners: <https://www.microolap.com/order/resellers/>.

Purchase online on www.microolap.com.

13.2. Uhid (HardwareID) of the runtime environment

Each Microolap EtherSensor license (license.licx file) is bound to the EtherSensor server hardware for which it is intended. This binding is done through the so-called UHID - a special code, unique for each system and equipment set (Unique Hardware Identifier). The UHID is added to the license file, and then EtherSensor checks if the UHID of the system and the UHID in the license file match.

If the UHID in the license file does not correspond to the current UHID of the system, EtherSensor automatically goes into demo mode.

When calculating the current UHID, the list of system NICs and the list of system storage devices are taken into account. This means that if you change the list of network cards or the list of storage devices of the system, the current UHID may also change.

To avoid problems with UHID modification during operation, before creating the request.licx file, it is necessary to bring all server hardware EtherSensor to the state in which it will be used: disable all temporary devices (flash cards, USB-HDD, etc.), disable all temporary, virtual or unused network cards, configure MAC addresses, etc.

During EtherSensor operation you can temporarily connect storage devices (flash card, portable HDD, etc.). EtherSensor monitors such situations and continues to work correctly with the license file even though the UHID is temporarily changed.

However, changes to the UHID may become permanent when equipment is scheduled to be replaced or added. In this situation, you need to re-generate the request.licx file and send it to the address

Generating a license requires confirmation from the supplier of EtherSensor in order for the end user to obtain it. It takes at least three hours to receive confirmation.

It should be remembered that the operation of the EtherSensor update system directly depends on the state of the license. With an expired license, updates cannot be obtained. The license data determines when and what updates EtherSensor will receive.

Microolap EtherSensor requires its installation server access to the update service for normal operation. If there is no such access, correct operation of EtherSensor is not guaranteed and support is not provided.

Beta licenses

The licensing system allows some licenses to be designated as beta licenses. Installations with beta licenses receive updates before they are released for all other installations. This allows you to have test installations of the product to study new versions.

13.4. After purchasing the license

You have purchased Microolap EtherSensor license.

With one license you may use EtherSensor on one computer or virtual machine at the same time.

Each license is linked to a specific hardware and cannot be used on another computer.

Please follow the instructions below to get EtherSensor up and running:

1. Download the appropriate distribution package (x86 or x64 platform) of EtherSensor from the link provided by Microolap Technologies.
2. Put the downloaded archive on the computer intended to run EtherSensor.
3. Right-click on the archive and select Properties.
4. If there is a Security section on the General tab, check Unblock field and click OK. If there is no Security section and no Unblock field, no action is required, close the Properties window.
5. Unpack the archive and run ethersensor_pcap_setup_x64_v6.1_en-us.exe file.
6. Follow the installation instructions.

To get the license file, the following actions must be performed on each computer where EtherSensor is planned to be installed:

1. From the Start menu, start the EtherSensor management console (sensor_console.exe).
2. Open the tab **Settings**, section **Licensing**, subsection **Request for license**.
3. Fill out the form on the right side of the window and click **Save**.
4. The request.licx file will be created and saved in the EtherSensor installation directory.

5. Please send us this file by e-mail at support@microolap.com, so that we create a license file for you.
6. We will send you the `license.licx` file, which should be placed in the installation directory of EtherSensor.
7. It's done. Now you can use the registered version of EtherSensor!

It means that during the subscription period you can download and install the latest versions of Microolap EtherSensor from our website, and the resulting license file will work with new versions of the product.

In addition, if you do not forget to enable automatic updates, new versions will be installed without your participation.

PLEASE DO NOT TRY TO OPEN OR RENAME THE LICENSE FILE!

Latest information and updates: <https://www.microolap.com/support/>.

13.5. License agreement

Microolap EtherSensor End User License Agreement (EULA)

NOTICE TO USER:

THIS IS AN AGREEMENT GOVERNING YOUR USE OF **Microolap EtherSensor 6.X**, FURTHER DEFINED HEREIN AS "PRODUCT". THE LICENSOR OF THE PRODUCT IS WILLING TO PROVIDE YOU WITH ACCESS TO THE PRODUCT ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS AND CONDITIONS CONTAINED IN THIS AGREEMENT. BELOW, YOU ARE ASKED TO ACCEPT THIS AGREEMENT AND CONTINUE TO INSTALL OR, IF YOU DO NOT WISH TO ACCEPT THIS AGREEMENT, TO DECLINE THIS AGREEMENT, IN WHICH CASE YOU WILL NOT BE ABLE TO INSTALL OR OPERATE THE PRODUCT. BY INSTALLING THIS PRODUCT YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT.

THIS EULA CAN BE FOUND AT <https://www.microolap.com> SHOULD YOU WISH TO CONSULT IT IN THE FUTURE.

This Electronic End User License Agreement (the "Agreement") is a legal agreement between you (either an individual or an entity), as a licensee, and Microolap Technologies and its affiliates (collectively, the "Licensor"), regarding the Product and related support service you are about to install and Operate and/or other related services, including without limitation:

- a. all of the contents of the files, including private assembly files, disk(s), CD-ROM(s) or other media with which this Agreement is provided and including all forms of code, such as Source Code and Object Code as provided and in a form that is provided by Licensor to you (the "Software"). For the avoidance of doubt, by way of example, but not exclusion, if a specific file is provided by Licensor in Object Code only, the Source Code for such files shall not be deemed a part of the Software provided by Licensor to you. For purposes hereof "Source Code" shall mean the human-

readable form of the computer programming code and related system documentation including all comments and any procedural code such as job control language and "Object Code" shall mean computer programs assembled or compiled in magnetic or electronic binary form on software media, which are readable and usable by machines, but not generally readable by humans without reverse-assembly, reverse-compiling, or reverse-engineering;

b. any sample programs or samples of the code made available on Licensor Site, as defined below (the "Samples");

c. all support services provided to you by Licensor in connection with the Software (the "Services");

d. and all successor upgrades, modified versions, modified modules, revisions, patches, enhancements, fixes, modifications, copies, additions or maintenance releases of the Software, if any, licensed to you by the Licensor (collectively, the "Updates") provided that the Updates shall not include a new subsequent releases of the Product bearing a new first (6.0 or 7.0 or 8.0 or 9.0 or 10.0 or 11.0) (collectively, "New Releases") but include any minor revisions of the Product version indicated by a change in the number after the period, such as 7.1 or 7.2, and

e. related user documentation and explanatory materials or files provided in written, "online" or electronic form (the "Documentation" and together with the Software, Samples, Updates, and Services the "Product").

For the purposes of this Agreement, "Licensor Site" shall mean the Internet website maintained by or on behalf of Licensor from which the Software is available for download pursuant to a license from Licensor. The Licensor Site is currently located at <https://www.microolap.com>.

You are subject to the terms and conditions of this End User License Agreement whether you access or obtain the Product directly from the Licensor, or through any other source. For purposes hereof, "you" means the individual person installing or using the Product on his or her own behalf; or, if the Product is being downloaded or installed on behalf of an organization, such as an employer, "you" means the organization for which the Product is downloaded or installed, then the person accepting this agreement represents hereby that such organization has authorized such person to accept this agreement on the organization's behalf. For purposes hereof the term "organization," without limitation, includes any partnership, limited liability company, corporation, association, joint stock company, trust, joint venture, labor organization, unincorporated organization, or governmental authority.

By accessing, storing, loading, installing, executing, displaying, copying the Product into the memory of a Client Device, as defined below, or otherwise benefiting from using the functionality of the Product ("Operating"), you agree to be bound by the terms and conditions of this Agreement. If you do not agree to the terms and conditions of this Agreement, the Licensor is unwilling to license the Product to you. In such event, you may not Operate or use the Product in any way.

IF THE LICENSOR AND YOU HAVE AGREED ON AND PROPERLY EXECUTED A SEPARATE CONTEMPORANEOUS OR SUBSEQUENT TERMS OF USE OR EXHIBITS (the "Terms of Use"), WHICH ARE SUPPLEMENTAL, DIFFERENT OR INCONSISTENT WITH THE TERMS OF THIS AGREEMENT, SUCH TERMS OF USE SHALL CONTROL, PROVIDED THAT (i) SUCH TERMS OF USE SPECIFICALLY ACKNOWLEDGE AND REFER TO THIS AGREEMENT, AND (ii) ALL OTHER TERMS AND CONDITIONS OF THIS AGREEMENT REMAIN IN FULL FORCE AND EFFECT.

BEFORE YOU PUT A CHECKMARK AT THE "I ACCEPT THE AGREEMENT" BUTTON AND PRESS "NEXT," PLEASE CAREFULLY READ THE TERMS AND CONDITIONS OF THIS AGREEMENT, AS SUCH ACTIONS ARE A SYMBOL OF YOUR SIGNATURE AND BY CLICKING ON THE "I ACCEPT THE AGREEMENT" AND "NEXT" BUTTONS, YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PARTY TO THIS AGREEMENT AND AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE "CANCEL" BUTTON AND THE PRODUCT WILL NOT BE INSTALLED ON YOUR CLIENT DEVICE, AS SUCH TERM IS DEFINED BELOW. You may also receive a copy of this Agreement by contacting Licensor at: formal@microolap.com.

1. Proprietary Rights and Non-Disclosure.

1.1. Ownership Rights. You agree that the Product and the authorship, systems, ideas, methods of operation, documentation and other information contained in the Product, are proprietary intellectual properties and/or the valuable trade secrets of the Licensor and are protected by civil and criminal law, and by the law of copyright, trade secret, trademark and patent of United States, other countries and international treaties. You may use trademarks only insofar as to identify printed output produced by the Product in accordance with accepted trademark practice, including identification of trademark owner's name. Such use of any trademark does not give you any rights of ownership in that trademark. The Licensor and its suppliers own and retain all right, title, and interest in and to the Product, including all copyrights, patents, trade secret rights, trademarks, and other intellectual property rights therein, including without limitation the mark Microolap. Your possession, installation or use of the Product does not transfer to you any title to the intellectual property in the Product, and you will not acquire any rights to the Product except as expressly set forth in this Agreement. All copies of the Product made hereunder must contain the same proprietary notices that appear on and in the Product. Except as stated herein, this Agreement does not grant you any intellectual property rights in the Product.

1.2. Source Code and Modifications. You acknowledge that the Source Code for the Product is proprietary to the Licensor and constitutes trade secrets of the Licensor. You agree not to disassemble, decompile or "unlock", decode or otherwise reverse-translate or reverse-engineer, or attempt in any manner to reconstruct or discover any Source Code or underlying algorithms of the Product or any part thereof provided solely in Object Code form but you may change, add or delete any files of the licensed copy of the Products and you may adapt or modify the Source Code solely for purposes of Operating a licensed copy of the Product by you and as expressly permitted pursuant to the Documentation provided that you may not, in any event, remove or alter any copyright notices or other proprietary notices on any copies of the Product, whether so modified or not, and further provided that any such change, addition, deletion, adaptation or modification voids any express warranty provided herein and terminates any right to support services.

1.3. License Key File and Confidential Information. You agree that, unless otherwise specifically provided herein or agreed by the Licensor in writing, the Product, including the specific design and structure of individual programs and the Product, including without limitation the License Key File provided to you by the Licensor and/or its authorized resellers or distributors, constitute confidential proprietary information of the Licensor. For purposes hereof, "License Key File" shall mean a unique key identification file or a combination of unique electronic characters provided to you by the Licensor confirming the purchase of the license from the Licensor, which may carry the information about the license and the allowed network traffic intensity, or the allowed number of the users monitored, or the allowed number of the simultaneously reconstructed network

sessions, or the allowed number of the network objects analysis results to be delivered to a consuming system per unit of time, and enabling the full functionality of the Product in accordance with the license granted under this Agreement. You agree not to transfer, copy, disclose, provide or otherwise make available such confidential information in any form to any third party without the prior written consent of the Licensor. You agree to implement reasonable security measures to protect such confidential information, but without limitation to the foregoing, shall use best efforts to maintain the security of the License Key File provided to you by the Licensor and/or its authorized resellers or distributors.

2. Grant of License.

2.1. License. The Licensor grants you the following rights:

2.1.1. Trial Version. If you have received, downloaded and/or installed a trial version of the Product and are hereby granted an evaluation license for the Software and you may Operate the Product only for evaluation purposes and only during the single applicable evaluation period of thirty (30) days, unless otherwise indicated, from the date of the initial installation. Any use of the Product for other purposes or beyond the applicable evaluation period is strictly prohibited, provided however that, subject to the restrictions contained herein, you may copy and distribute a trial version of the Software without any modifications whatsoever to any third party subject to this Agreement and further provided that you have no technical support rights during the trial period.

2.1.2. Grant of License. Unless otherwise specifically indicated under a valid license (e.g. volume license) pursuant to an applicable Terms of Use granted by the Licensor you are granted a non-exclusive and non-transferable license to install one (1) copy of the Product and during the term of your license, subject to the payment of the applicable fees and your compliance with the terms hereof, this Agreement permits you or any of your employees to Operate one copy of the specified version of the Product, for internal purposes only, on one computer, workstation, or other electronic device for which the software was designed (each a "Client Device") and implemented solely within one network traffic analyzer (the "Sensor") using a single copy of the Product licensed hereunder, the initial text of the Product, and any Documentation, and in combination with Microolap identifier to group the Sensor's objects including but not limited to network traffic objects detectors, network packet filters, samples, results delivery profiles, etc., provided that the network traffic intensity, or the number of the users monitored by the Sensor, or the number of the simultaneously reconstructed network sessions, or the number of the network objects analysis results to be delivered to a consuming system per unit of time do not exceed the allowed number according to the purchased product edition. To extend the allowed network traffic intensity, or the number of the users monitored by the Sensor, or the allowed number of the simultaneously reconstructed network sessions, or the allowed number of the network objects analysis results to be delivered to a consuming system per unit of time, you are required to upgrade to the next tier product edition in the manner provided in this End User License Agreement. If the Product is licensed as a suite or bundle with more than one specified software product, this license applies to all such specified software products, subject to any restrictions or usage terms specified on the applicable price list or product packaging that apply to any of such software products individually. The Licensor reserves all rights not expressly granted herein.

2.1.3. Volume Use. If the Product is licensed with volume license terms specified in the applicable product invoicing or packaging for the Product, you may make use and install as many additional copies of the Product on the number of Client Devices as the volume license terms specify. You

must have a reasonable mechanism in place to ensure that the number of Client Devices on which the Product has been installed does not exceed the number of licenses you have obtained.

2.1.4. Test copy. You may also make a copy of the Product solely for purposes of testing, adjusting and similar tasks provided that such copy shall be located only on the local Client Device or local area network (LAN) without any internet or remote access of any party and further provided that such copy is deleted upon consummation of the Sensor.

2.1.5. Multiple Environment Product; Multiple Language Product; Dual Media Product; Multiple Copies; Bundles. If the Product supports multiple platforms or languages, if you receive the Product on multiple media, if you otherwise receive multiple copies of the Product, or if you received the Product bundled with other software, the total number of your Client Devices on which all versions of the Product are installed may not exceed the number of licenses you have obtained from the Licensor. You may not rent, lease, sublicense, lend or transfer any versions or copies of the Product you do not use.

2.1.6. Back-up Copies. You can make one (1) copy of the Product for backup and archival purposes, provided, however, that the original and each copy is kept in your possession or control, and that your installation and use of the Product does not exceed that which is allowed in this Section 2.

3. Redistributable Elements

3.1. In addition to the license and rights granted in Section 2, Licensor grants you the right to use and reproduce the copyrightable elements of the Product such as network objects detectors, filtering scripts, and results delivery profiles (collectively "Redistributables") provided:

3.1.1. you may use the Redistributables to create your own network services detectors, filtering scripts, and results delivery profiles as long as, it is NOT the basis for creating a product that provides the same, or substantially the same, functionality as any product of Licensor; and

3.1.2. in the event you develop any modifications, enhancements, derivative works and/or extensions to the Redistributables, either independently or jointly with Licensor, such modifications, enhancements, derivative works and/or extensions and all rights associated therewith will be the exclusive property of Licensor. You will not grant, either expressly or impliedly, any rights, title, interest, or licenses to any such modifications, enhancements, derivative works and/or extensions to any third party. You will, however, be entitled to use such modifications, enhancements, derivative works and/or extensions under the terms set forth in this Agreement. You hereby assign all right, title and interest in and to such modifications, enhancements, derivative works and/or extensions to the Redistributables to Licensor. You also agree to execute, acknowledge and deliver to Licensor all documents and do all things Licensor deems necessary or desirable, at no cost to but at expense of Licensor, to enable Licensor to obtain, secure, register or prosecute such modifications, enhancements, derivative works and/or extensions anywhere in the world. You agree to secure all necessary rights and obligations from relevant employees, or third parties in order to satisfy the above obligations.

3.2. In addition to the other requirements set forth in this Section 3, you hereby agree to indemnify, hold harmless, and defend Licensor from and against any and all liabilities, damages, losses, costs and expenses (including reasonable attorneys' fees) arising from or related to any alleged or actual claim, action, proceeding or allegation that arises or results, either directly or indirectly, from the exercise of your rights relating to the Redistributables.

4. Term and Termination.

4.1. The term of this Agreement ("Term") shall begin when you download, access or install the Product or pay the applicable license fees (whichever is earlier) and shall continue for the term specified in your order. Without prejudice to any other rights, this Agreement will terminate automatically if you fail to comply with any of the limitations or other requirements described herein. Upon any termination or expiration of this Agreement, you must immediately cease Operating the Product and all of its components and destroy, uninstall and erase all copies of the Product and all of its components, including without limitation on all systems and all types of media and in computer memory.

4.2. No Rights Upon Termination. Upon termination of this Agreement you will no longer be authorized to Operate or use the Product in any way.

5. Support and Updates.

5.1. Terms of Support. During the Warranty Period as defined below, you are entitled to technical services and support for the Product which is provided to you by Licensor during the regular business hours (GMT+3), except for locally-observed holidays, and includes the support provided through a special technical support section of the Licensor Website, email (support@microolap.com), or phone as listed on the Site <https://www.microolap.com>. During such period of one year (this period could be different and it depends on license(s) you have), email support, is unlimited and includes technical and support questions and patch fixes. Any question submitted to the Site support channel will be responded to within two (2) business days.

5.2. Updates. During the Warranty Period hereunder, you may download Updates to the Product when and as the Licensor publishes them on the Site, or through other online services. If the Product is an Update to a previous version of the Product, you must possess a valid license to such previous version in order to use the Update. You may continue to use the previous version of the Product on your Client Device after you receive the Update to assist you in the transition to the Update, provided that: (i) the Update and the previous version are installed on the same Client Device; (ii) the previous version or copies thereof are not transferred to another party or Client Device unless all copies of the Update are also transferred to such party or Client Device; (iii) you acknowledge that any modification that you made to the Product may be lost, altered, distorted or destroyed rendering such modifications, Product or the part thereof inoperable or non-usable; and (iv) you acknowledge that any obligation the Licensor may have to support the previous version of the Product may be ended upon availability of the New Release. Except for the rights to free Updates during the Warranty Period, as further defined herein, nothing in this Agreement shall be construed as to grant you any rights or licenses with regard to the new releases of the Product or to entitle you to any new release. This Agreement does not obligate Microolap Technologies to provide any Updates. Notwithstanding the foregoing, any Updates that you may receive become part of the Product and the terms of this Agreement apply to them (unless this Agreement is superseded by a succeeding agreement accompanying such Update or modified version of the Product).

5.3. Limitations. In the event Product Updates and Support aren't purchased, the Product's functionality will be limited as stated at <https://www.microolap.com/products/ethersensor/limitations/>. Microolap Technologies reserves the right to amend the limitations descriptions at any time without prior notice.

6. Restrictions.

6.1. No Transfer of Rights. You may not transfer any rights pursuant to this Agreement nor rent, sublicense, lease, loan or resell the Product. You may not permit third parties to benefit from the use or functionality of the Product via a timesharing, service bureau or other arrangement, except to the extent such use is specified in the application price list, purchase order or product packaging for the Product. Except as otherwise provided in Section 1.2 hereof, you may not, without the Licensor's prior written consent, reverse engineer, decompile, disassemble or otherwise reduce any part of the Product to human readable form nor permit any third party to do so, except to the extent the foregoing restriction is expressly prohibited by applicable law. Notwithstanding the foregoing sentence, decompiling the Software is permitted to the extent the laws of your jurisdiction give you the right to do so to obtain information necessary to render the Software interoperable with other software; provided, however, that you must first request such information from the Licensor and the Licensor may, in its discretion, either provide such information to you (subject to confidentiality terms) or impose reasonable conditions, including a reasonable fee, on such use of the Software to ensure that the Licensor's and its affiliates' proprietary rights in the Software are protected. Except for the modification permitted under Section 1.2, you may not modify, or create derivative works based upon the Product in whole or in part.

6.2. Proprietary Notices and Copies. You may not remove any proprietary notices or labels on the Product. You may not copy the Product except as expressly permitted in Section 2 above.

6.3. Compliance with Law. You agree that in Operating the Product and in using any report or information derived as a result of Operating this Product, you will comply with all applicable international, national, state, regional and local laws and regulations, including, without limitation, privacy, trademark, patent, copyright, export control and obscenity law and you shall not use the Product for unethical or illegal business practices or in violation of any obligation to a third party in using, operating, accessing or running any of the Product and shall not knowingly assist any other person or entity to so violate any obligation to a third party.

6.4. Additional Protection Measures. Solely for the purpose of preventing unlicensed use of the Product, the Software may install on your Client Device technological measures that are designed to prevent unlicensed use, and the Licensor may use this technology to confirm that you have a licensed copy of the Product. The update of these technological measures may occur through the installation of the Updates. The Updates will not install on unlicensed copies of the Product. If you are not using a licensed copy of the Product, you are not allowed to install the Updates. The Licensor will not collect any personally identifiable information from your Client Device during this process.

7. WARRANTIES AND DISCLAIMERS.

7.1. Limited Warranty. The Licensor warrants that for one year (the "Warranty Period") from the date the License Key File is provided to you by Licensor (i) the media on which Product has been provided will be free from defects in materials and workmanship, and (ii) the Software will perform substantially in accordance with the Documentation or generally conform to the Product's specifications published by the Licensor. Non-substantial variations of performance from the Documentation do not establish a warranty right. THIS LIMITED WARRANTY DOES NOT APPLY TO UPDATES AS APPLIED TO ANY MODIFIED PRODUCT, WHETHER OR NOT SUCH MODIFICATION IS PERMISSIBLE HEREUNDER, TRIAL AND EVALUATION VERSIONS, UPDATES, PRE-RELEASE, TRYOUT,

PRODUCT SAMPLER, OR NOT FOR RESALE (NFR) COPIES OF PRODUCT. This limited warranty is void and your support right terminate if the defect has resulted from accident, abuse, or misapplication or any modification, whether or not such modification is permitted hereunder. No warranty is made as to the integrity, protection or safekeeping of any modification to the Products made by you upon installation of any of the Updates. To make a warranty claim, you must return the Product to the location where you obtained it along with proof of purchase within such sixty (60) day period of the license fee you paid for the Product. THE LIMITED WARRANTY SET FORTH IN THIS SECTION GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE ADDITIONAL RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

7.2. Customer Remedies. The Licensor and its suppliers' entire liability and your exclusive remedy for any breach of the foregoing warranty shall be at the Licensor's option: (i) return of the purchase price paid for the license, if any, (ii) replacement of the defective media in which the Product is contained, or (iii) correction of the defects, "bugs" or errors within reasonable period of time. You must return the defective media to the Licensor at your expense with a copy of your receipt. Any replacement media will be warranted for the remainder of the original warranty period.

7.3. NO OTHER WARRANTIES. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT TO WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, THE PRODUCT IS PROVIDED "AS-IS" WITHOUT ANY WARRANTY WHATSOEVER AND THE LICENSOR MAKES NO PROMISES, REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESSED OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE, REGARDING OR RELATING TO THE PRODUCT OR CONTENT THEREIN OR TO ANY OTHER MATERIAL FURNISHED OR PROVIDED TO YOU PURSUANT TO THIS AGREEMENT OR OTHERWISE. YOU ASSUME ALL RISKS AND RESPONSIBILITIES FOR SELECTION OF THE PRODUCT TO ACHIEVE YOUR INTENDED OR EXPECTED RESULTS, AND FOR THE INSTALLATION OF, USE OF, AND RESULTS OBTAINED FROM THE PRODUCT. THE LICENSOR MAKES NO WARRANTY THAT THE PRODUCT WILL BE ERROR FREE OR FREE FROM INTERRUPTION OR FAILURE, OR THAT IT WILL MEET YOUR OR YOUR CUSTOMERS' EXPECTATIONS, OR THAT IT IS COMPATIBLE WITH ANY PARTICULAR HARDWARE OR SOFTWARE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT OF THIRD PARTY RIGHTS, INTEGRATION, SATISFACTORY QUALITY OR FITNESS FOR ANY PARTICULAR PURPOSE WITH RESPECT TO THE PRODUCT AND THE ACCOMPANYING WRITTEN MATERIALS OR THE USE THEREOF. SOME JURISDICTIONS DO NOT ALLOW LIMITATIONS ON IMPLIED WARRANTIES, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU. YOU HEREBY ACKNOWLEDGE THAT THE PRODUCT MAY NOT BE OR BECOME AVAILABLE DUE TO ANY NUMBER OF FACTORS INCLUDING WITHOUT LIMITATION PERIODIC SYSTEM MAINTENANCE, SCHEDULED OR UNSCHEDULED, ACTS OF GOD, TECHNICAL FAILURE OF THE SOFTWARE, TELECOMMUNICATIONS INFRASTRUCTURE, OR DELAY OR DISRUPTION ATTRIBUTABLE TO VIRUSES, DENIAL OF SERVICE ATTACKS, INCREASED OR FLUCTUATING DEMAND, AND ACTIONS AND OMISSIONS OF THIRD PARTIES. THEREFORE, THE LICENSOR EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY REGARDING SYSTEM AND/OR SOFTWARE AVAILABILITY, ACCESSIBILITY, OR PERFORMANCE. THE LICENSOR DISCLAIMS ANY AND ALL LIABILITY FOR THE LOSS OF DATA DURING ANY COMMUNICATIONS AND ANY LIABILITY ARISING FROM OR RELATED TO ANY FAILURE BY THE LICENSOR TO TRANSMIT ACCURATE OR COMPLETE INFORMATION TO YOU.

7.4. LIMITED LIABILITY; NO LIABILITY FOR CONSEQUENTIAL DAMAGES. YOU ASSUME THE ENTIRE COST OF ANY DAMAGE RESULTING FROM YOUR USE OF THE PRODUCT AND THE INFORMATION CONTAINED IN OR COMPILED BY THE PRODUCT, AND THE INTERACTION (OR FAILURE TO INTERACT PROPERLY) WITH ANY OTHER HARDWARE OR SOFTWARE WHETHER PROVIDED BY THE LICENSOR OR A THIRD PARTY. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT WILL THE LICENSOR OR ITS SUPPLIERS OR LICENSORS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF DATA, LOSS OF GOODWILL, WORK STOPPAGE, HARDWARE OR SOFTWARE DISRUPTION IMPAIRMENT OR FAILURE, REPAIR COSTS, TIME VALUE OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT, OR THE INCOMPATIBILITY OF THE PRODUCT WITH ANY HARDWARE SOFTWARE OR USAGE, EVEN IF SUCH PARTIES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL LICENSOR'S TOTAL LIABILITY TO YOU FOR ALL DAMAGES IN ANY ONE OR MORE CAUSE OF ACTION, WHETHER IN CONTRACT, TORT OR OTHERWISE EXCEED THE AMOUNT PAID BY YOU FOR THE PRODUCT TO THE LICENSOR. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT THAT APPLICABLE LAW PROHIBITS SUCH LIMITATION. FURTHERMORE, BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

8. Indemnification

8.1. Indemnification for Violations. Your Operating of the Product, your accessing your account with Licensor and your entering into this Agreement constitutes your consent and agreement to defend, indemnify and hold harmless Licensor and its affiliated companies, employees, contractors, officers and directors from any claim or demand, including reasonable attorney's fees arising out of your use of the Product in violation of this Agreement.

9. U.S. Government-Restricted Rights.

9.1. Notice to U.S. Government End Users. The Product and accompanying Documentation are deemed to be "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," respectively, as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. § 227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights, including any use, modification, reproduction, release, performance, display or disclosure of the Product and accompanying Documentation, as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States.

9.2.Export Restrictions. You acknowledge and agree that the Confidential Information, Product, its use, export or transshipment may be subject to restrictions and controls imposed by various government authorities and you agree to assure that, in connection with performance of its obligations pursuant to this Agreement or arising or relating therefrom, no Software, Product, Documentation, Confidential Information or any portion thereof, and any information relating thereto or to this Agreement, is exported, transshipped or re-exported, directly or indirectly, in violation of any applicable law and ensure that neither the Software, Products nor the

Documentation, underlying information or technology may be downloaded or otherwise exported or re-exported in violation of applicable embargo or export/import law, regulation or international treaty. You acknowledge that it is your sole responsibility to comply with any and all government export and other applicable laws and that the Licensor has no further responsibility for such after the initial license to you.

10. Your Information and the Licensor's Privacy Policy

10.1. Privacy Policy. You acknowledge receipt of and agree to the Licensor's privacy statement which is made available to you in connection with installation and is set forth in full at <https://www.microolap.com/privacy/>. You here by expressly consent to the Licensor's processing of your personal data (which may be collected by the Licensor or its distributors) according to the Licensor's current privacy policy as of the date of the effectiveness here of which is incorporated into this Agreement by reference. By entering into this Agreement, you agree that the Licensor may collect and retain information about you, including your name, email address and credit card information. The Licensor employs other companies and individuals to perform certain functions on its behalf. Examples include fulfilling orders, delivering packages, sending postal mail and email, removing repetitive information from customer lists, analyzing data, providing marketing assistance, processing credit card payments, and providing customer service. They have access to personal information needed to perform their functions, but may not use it for other purposes. The Licensor publishes a privacy policy on its web site and may amend such policy from time to time in its sole discretion. You should refer to the Licensor's privacy policy prior to agreeing to this Agreement for a more detailed explanation of how your information will be stored and used by the Licensor. If "you" are an organization, you will ensure that each member of your organization (including employees and contractors) about whom personal data may be provided to the Licensor has given his or her express consent to the Licensor's processing of such personal data. Personal data will be processed by the Licensor or its distributors in the country where it was collected, and possibly in the United States and Germany. United States laws regarding processing of personal data may be less or more stringent than the laws in your jurisdiction.

10.2. Public Announcements. The Licensor may identify you to the public as a customer of the Licensor and describe in a customer case study the services and solutions delivered by the Licensor to you. The Licensor may also issue one or more press releases, containing an announcement of the execution and delivery of this Agreement and/or the implementation of the Product by you. Nothing contained in this Section shall be construed as an obligation by you to disclose any of your proprietary or confidential information to any third party. In addition, you may opt-out from this Section by writing an opt-out request to the Licensor at formal@microolap.com.

11. Miscellaneous.

11.1. Governing Law; Jurisdiction and Venue. This Agreement shall be governed by and construed and enforced in accordance with the laws of the Russian Federation without reference to conflicts of law rules and principles. To the extent permitted by law, the provisions of this Agreement shall supersede any provisions of the Uniform Commercial Code as adopted or made applicable to the Products in any competent jurisdiction. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly disclaimed and excluded. The courts within the Russian Federation shall have exclusive jurisdiction to adjudicate any dispute arising out of this Agreement. You agree that this Agreement is to be performed in the Russian Federation and that any action, dispute, controversy,

or claim that may be instituted based on this Agreement, or arising out of or related to this Agreement or any alleged breach thereof, shall be prosecuted exclusively in the courts in of the Russian Federation and you, to the extent permitted by applicable law, hereby waive the right to change venue to any other state, county, district or jurisdiction; provided, however, that the Licensor as claimant shall be entitled to initiate proceedings in any court of competent jurisdiction.

11.2. Period for Bringing Actions. No action, regardless of form, arising out of the transactions under this Agreement, may be brought by either party hereto more than one (1) year after the cause of action has occurred, or was discovered to have occurred, except that an action for infringement of intellectual property rights may be brought within the maximum applicable statutory period.

11.3. Entire Agreement; Severability; No Waiver. This Agreement is the entire agreement between you and Licensor and supersedes any other prior agreements, proposals, communications or advertising, oral or written, with respect to the Product or to subject matter of this Agreement. You acknowledge that you have read this Agreement, understand it and agree to be bound by its terms. If any provision of this Agreement is found by a court of competent jurisdiction to be invalid, void, or unenforceable for any reason, in whole or in part, such provision will be more narrowly construed so that it becomes legal and enforceable, and the entire Agreement will not fail on account thereof and the balance of the Agreement will continue in full force and effect to the maximum extent permitted by law or equity while preserving, to the fullest extent possible, its original intent. No waiver of any provision or condition herein shall be valid unless in writing and signed by you and an authorized representative of Licensor provided that no waiver of any breach of any provisions of this Agreement will constitute a waiver of any prior, concurrent or subsequent breach. Licensor's failure to insist upon or enforce strict performance of any provision of this Agreement or any right shall not be construed as a waiver of any such provision or right.

11.4. Contact Information. Should you have any questions concerning this Agreement, or if you desire to contact the Licensor for any reason, please contact us at formal@microolap.com.

2001 - 2020 Microolap Technologies. All rights reserved. The Product, including the Software and any accompanying Documentation, are copyrighted and protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. Microolap® is a registered trademark of Microolap Technologies. By using Microolap Technologies trademark, in whole or in part, you are acknowledging that Microolap Technologies is the sole owner of the trademark and promising that you will not interfere with Microolap Technologies' rights in the trademark, including challenging its use, registration of, or application to register such trademark, alone or in combination with other words, anywhere in the world, and that you will not harm, misuse, or bring into disrepute any Microolap Technologies' trademark. The goodwill derived from using any part of Microolap Technologies' trademark exclusively inures to the benefit of and belongs to Microolap Technologies. Except for the limited right to use as expressly permitted under these Agreement, no other rights of any kind are granted hereunder, by implication or otherwise. If you have any questions regarding these guidelines, please email us at: formal@microolap.com.