



we protect your digital worlds

ESET File Security

*Installation Manual
and User's documentation*

Table of contents

1. Introduction	3
2. Terminology and abbreviations	5
3. Installation	9
4. Product's Roadmap	11
5. Integration with File System services	15
5.1. On-demand scanner	16
5.2. On-access scanner powered by Dazuko	16
5.2.1. Operation principle.....	17
5.2.2. Installation and configuration	17
5.2.3. Tips	17
5.3. On-access scanner using preload LIBC library.....	18
5.3.1. Operation principle.....	18
5.3.2. Installation and configuration	19
5.3.3. Tips	19
6. Important ESET File Security mechanisms	21
6.1. Handle Object Policy.....	22
6.2. User Specific Configuration.....	22
6.3. Samples Submission System.....	23
6.4. World WideWeb Interface.....	24
7. ESET Security system update	25
7.1. ESETS update utility.....	26
7.2. ESETS update process description	26
8. Let us know	29
Appendix A. PHP License	31

ESET File Security

Copyright © 2007 ESET, spol. s r.o.

ESET File Security was developed by ESET, spol. s r.o. For more information visit www.nod32.com.hk

All rights reserved. No part of this documentation may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise without a permission in writing from the author.

ESET, spol. s r.o. reserves the right to change any of the described application software without prior notice.

This product includes PHP software, freely available from <http://www.php.net/software/>.

REV.20071112-003



Chapter 1:

Introduction



Dear user, you have acquired ESET File Security - probably the best security system running under the Linux/BSD OS. As you will soon find out, the system using the state-of-the-art ESET scanning engine, has unsurpassed scanning speed and detection rate, combined with a very small footprint that makes it the ideal choice for any Linux/BSD OS server.

In the rest of this chapter we review a key features of the system.

- The ESET anti-virus scanning engine algorithms provide the highest detection rate and the fastest scanning times.
- The ESET File Security is developed to run on the single-processor as well as on the multi-processor units.
- It includes unique advanced heuristics for Win32 worms and back-doors.
- Inbuilt archivers unpack archived objects without the need for any external programs.
- In order to increase speed and efficiency of the system, its architecture is based on the running daemon (resident program) where all the scanning requests are sent to.
- The system supports selective configuration specific for user or client/server identification.
- Six logging levels can be configured to get information about system activity and infiltrations.
- The ESET File Security installation does not require external libraries or programs except for LIBC.
- The system can be configured to notify any person in case of detected infiltration.

To run efficiently, ESET File Security requires just 16MB of hard-disk space and 32MB of RAM. It works smoothly under the 2.2.x, 2.4.x and 2.6.x Linux OS kernel versions and also under 5.x, 6.x FreeBSD OS kernel versions.

From lower-powered, small office servers to enterprise-class ISP servers with thousands of users, the system delivers the performance and scalability you expect from a UNIX based solution and the unequalled security of ESET products.



Chapter 2:

Terminology and abbreviations



In the following text we review terms and abbreviations used in this documentation. Note that in this documentation (PDF format only) a boldface font is reserved for product components names and in this chapter also for newly defined terms and abbreviations. Note also that terms and abbreviations defined in this chapter are emphasized later in this documentation (PDF format only).

ESETS

ESET Security is a common acronym for all security products developed by ESET, spol. s r.o. for Linux OS (resp. for BSD OS). It is also the name (or its part) of the software package containing the products.

RSR

Abbreviation of 'RedHat/Novell(SuSE) Ready'. Note that we support also so called RedHat Ready and Novell(SuSE) Ready variation of the product. The difference from the "standard" Linux version is that the *RSR* package meets criteria defined by FHS (File-system Hierarchy Standard defined as a part of Linux Standard Base) document required by the RedHat Ready and Novell(SuSE) Ready certificate. This means for instance that the *RSR* package is installed as an add-on application, i.e. the primary installation directory is `/opt/eset/esets`.

ESETS daemon

Main *ESETS* system control and scanning daemon `esets_daemon`.

ESETS base directory

The directory where *ESETS* loadable modules containing for instance virus signatures database are stored. Further in this documentation we use abbreviation `@BASEDIR@` for the directory. The directory location is as follows:

```
Linux: /var/lib/esets
Linux RSR: /var/opt/eset/esets/lib
BSD: /var/lib/esets
```

ESETS configuration directory

A directory where all files related with the ESET File Security configuration are stored. Further in this documentation we use abbreviation `@ETCDIR@` for the directory. The directory location is as follows:

```
Linux: /etc/esets
Linux RSR: /etc/opt/eset/esets
BSD: /usr/local/etc/esets
```

ESETS configuration file

Main ESET File Security configuration file. The absolute path of the file is as follows:

```
@ETCDIR@/esets.cfg
```

ESETS binary files directory

The directory where the relevant ESET File Security binary files are stored. Further in this documentation we use abbreviation `@BINDIR@` for the directory. The directory location is as follows:

```
Linux: /usr/bin
Linux RSR: /opt/eset/esets/bin
BSD: /usr/local/bin
```

ESETS system binary files directory

The directory where the relevant ESET File Security system binary files are stored. Further in this documentation we use abbreviation @SBINDIR@ for the directory. The directory location is as follows:

```
Linux: /usr/sbin  
Linux RSR: /opt/eset/esets/sbin  
BSD: /usr/local/sbin
```

ESETS object files directory

The directory where the relevant ESET File Security object files and libraries are stored. Further in this documentation we use abbreviation @LIBDIR@ for the directory. The directory location is as follows:

```
Linux: /usr/lib/esets  
Linux RSR: /opt/eset/esets/lib  
BSD: /usr/local/lib/esets
```





Chapter 3:

Installation



This product is distributed as a binary file:

```
eSETS.i386.ext.bin
```

where 'ext' is a Linux/BSD OS distribution dependent suffix, i.e. 'deb' for Debian, 'rpm' for RedHat and SuSE, 'tgz' for other Linux OS distributions, 'fbs5.tgz' for FreeBSD 5.xx and 'fbs6.tgz' for FreeBSD 6.xx distributions.

Note that the Linux *RSR* binary file format is:

```
eSETS-rsr.i386.rpm.bin
```

In order to install or update the product, use statement:

```
sh ./eSETS.i386.ext.bin
```

resp. for Linux *RSR* variation of the product, use statement:

```
sh ./eSETS-rsr.i386.rpm.bin
```

As a result the product's User License Acceptance Agreement is shown. Once you have confirmed the Acceptance Agreement, the installation package is placed into the current working directory and relevant information regarding the package's installation, un-installation or update is printed into terminal.

Once the package is installed and the main *ESETS* service is running, in Linux OS you can check its operation by using command:

```
ps -C eSETS_daemon
```

In case of BSD OS you can use a similar command:

```
ps -ax eSETS_daemon | grep eSETS_daemon
```

You will see the following (or similar) message on return:

PID	TTY	TIME	CMD
2226	?	00:00:00	eSETS_daemon
2229	?	00:00:00	eSETS_daemon

where at least two *ESETS daemon* processes running in the background have to be present. One of the processes is so-called process and threads manager of the system. The other serves as *ESETS* scanning process.



Chapter 4:

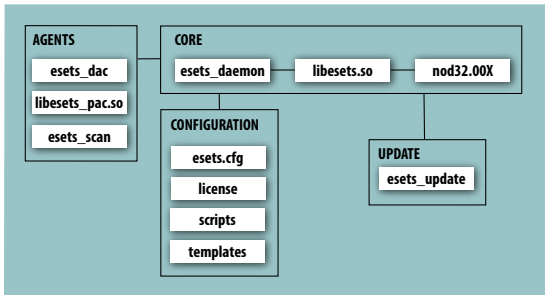
Product's Roadmap



Once the product package has been successfully installed, it is time to become familiar with its content.

The structure of ESET File Security is shown in the figure 4-1. The system is composed of the following components.

Figure 4-1. Structure of ESET File Security.



CORE

Core of ESET File Security consists of *ESETS daemon* **esets_daemon**. The daemon uses *ESETS* API library **libesets.so** and *ESETS* loading modules **nod32.00X** to provide base system tasks: scanning, maintenance of the agent daemon processes, maintenance of the samples submission system, logging, notification, etc.. Please refer to **esets_daemon(8)** manual page for details.

AGENTS

The purpose of *ESETS* agent modules is to integrate *ESETS* with the Linux/BSD Server environment. Please note a special chapter in this document devoted to the topic.

UPDATE

The update utility is a particular fraction of the system. It is developed to update *ESETS* loading modules containing for instance virus signatures database, archives support, advanced heuristics support etc. Please note a special chapter in this document devoted to the topic.

CONFIGURATION

Proper configuration is the most important condition for the system operation. Therefore we describe all the related components in the rest of this chapter. We also strongly recommend to read **esets.cfg(5)** manual page, an essential information source regarding *ESETS* configuration. After the product is successfully installed, all its configuration components are stored in *ESETS configuration directory*. The directory consists of the following files.

@ETCDIR@/esets.cfg

This is the most important configuration file as it maintains the major part of the product functionality. After exploring the file you can see that it is built from various parameters distributed within sections. Note the section names always enclosed in square brackets. In the *ESETS configuration file* there is always one global and several so-called agent sections.

Parameters in global section are used to define configuration options of *ESETS daemon* as well as default values of *ESETS* scanning engine configuration options. Parameters in agent sections are used to define configuration options of so-called agents, i.e. modules used to intercept various data flow types in the computer and/or its neighborhood and prepare this data for scanning. Note that besides the number of parameters used for the system configuration, there is also a number of rules determining organization of the file. To become familiar with this knowledge, please refer to *esets.cfg(5)*, *esets_daemon(8)* manual page and also to manual pages related to relevant agents.

@ETCDIR@/certs

This directory is used to store the certificates used by *ESETS WWW* Interface for authentication (see *esets_wwwi(8)* for details).

@ETCDIR@/license

This directory is used to store the product(s) license key(s) you have acquired from your vendor. Note that the *ESETS daemon* will always check only this directory to evaluate license key validity unless it is redefined by *ESETS configuration file* parameter 'lic_dir'.

@ETCDIR@/scripts/license_warning_script

This script, if enabled by *ESETS configuration file* parameter 'license_warn_enabled', is executed since 30 days (once per day) before product license expiration. It is used to send e-mail notification about the expiration status to system administrator.

@ETCDIR@/scripts/daemon_notification_script

This script, if enabled by *ESETS configuration file* parameter 'exec_script', is executed in case the infiltration has been detected by the anti-virus system. It is used to send e-mail notification about the event to system administrator.



Chapter 5:

Integration with File System services

This chapter describes process of configuration of ESET File Security system in order to provide an efficient protection from virus and worm infections of the file systems by using on-demand and on-access scanning techniques. The ESET File Security is composed from the so-called on-demand scanner `esets_scan` and so-called on-access scanner `esets_dac`. The Linux version of the products implements also additional on-access scanner technique using preload library module `libesets_pac.so`. All concerned components are described in the following sections.

5.1. On-demand scanner

On-demand scanner is scanner that can be invoked by privileged user (usually system administrator) using command line interface or by operating system using periodic command scheduler. This is also an explanation of the term 'on-demand' that the file system objects are scanned on user and/or system demand.

Concerning on-demand scanner there are not special requirements for its operation. After proper installation of the *ESETS* package and after valid license has been provided within the license keys directory feel free to run on-demand scanner by using command line interface or scheduler tool.

In order to run on-demand scanner from command line the following syntax is expected

```
@SBINDIR@/esets_scan [option(s)] INCL_DIR -- -EXCL_MASK
```

where `INCL_DIR` (resp. `EXCL_MASK`) is list of directories and/or files to be scanned (resp. excluded from scanning).

Multiple command line options are implemented within *ESETS* on-demand scanner. To get full list of them, read the manual page `esets_scan(8)`.

Please note that there is no configuration file interface supported for this module.

5.2. On-access scanner powered by Dazuko

On-access scanner is scanner invoked by predefined triggered access of user(s) and/or operating system to the file system objects. This also explains the term 'on-access'; the scanner is started on attempt to access selected file system object.

The technique used by *ESETS* on-access scanner is based on kernel calls interception which is powered by Dazuko (read da-tzu-ko) kernel module. Dazuko project is a Free software, by means that it is distributed as a free source code, in order to allow users compilation of the kernel module for their own custom kernels. Note that the Dazuko kernel module is not a part of the *ESETS* product and thus it must be compiled and installed into the kernel prior the *ESETS* on-access controller `esets_dac` initialization. On the other hand the Dazuko technique make on-access scanning independent of used file system type. It is also suitable for controlling file system objects via Network File System (NFS), Nettalk and Samba.

IMPORTANT: Before we provide user with the detailed information related with the on-access scanner configuration and operation, we would like to point out that any *ESETS* on-access scanner is not assumed to provide protection of whole file system where installed. It has been developed and tested to protect primarily the file systems mounted externally. If this is not your case, you will have to count on exclusion of multiple directories from file access control to

prevent system from hang-up. Typical directory to be excluded in this case is '/dev' directory and directories used by *ESETS*.

5.2.1. Operation principle

On-access scanner **esets_dac** (*ESETS* Dazuko powered file Access Controller) is a resident program providing continuous monitoring and control over the file system. Scanning of each file system object is performed upon customizable file access event. The following file access types are supported by the current version:

ON_OPEN events

This file access type is controlled once the first bit of the integer parameter 'event_mask' in the *ESETS configuration file* (section [dac]) is 1. In this case ON_OPEN bit of Dazuko access mask is set on.

ON_CLOSE events

This file access type is controlled once second bit of the integer parameter 'event_mask' in the *ESETS configuration file* (section [dac]) is 1. In this case ON_CLOSE bit and ON_CLOSE_MODIFIED bit of Dazuko access mask is set on.

Note that some of the kernel versions do not support interception of the ON_CLOSE events. In this case problems could be detected when running **esets_dac** module.

ON_EXEC events

This file access type is controlled once third bit of the integer parameter 'event_mask' in the *ESETS configuration file* (section [dac]) is 1. In this case ON_EXEC bit of Dazuko access mask is set on.

By using this mechanism all opened, closed and executed regular files are scanned by **esets_daemon** for viruses. Based on the result of this scanning the access to the files is denied or allowed.

5.2.2. Installation and configuration

It has been already discussed that prior any **esets_dac** initialization, so-called Dazuko kernel module has to be compiled and installed within the running kernel. In order to compile and install Dazuko, read <http://www.dazuko.org/howto-install.shtml>.

Once Dazuko installed, read and edit [global] and [dac] sections of *ESETS configuration file*. Note that for the proper functioning of on-access scanner it is necessary to enable configuration option 'agent_enabled' within a [dac] section of *ESETS configuration file*. Furthermore it is necessary to define file system objects (i.e. directories and files) that are required to be under control of on-access scanner. This can be achieved by defining the parameters of 'ctl_incl' and 'ctl_excl' configuration options within [dac] section of *ESETS configuration file*. For reread of newly created configuration, reload the *ESETS daemon*.

5.2.3. Tips

To provide Dazuko module loading prior each **esets_dac** daemon initialization, follow the next steps:

Provide copy of Dazuko module in some of the directories located within the directory

reserved for the kernel modules

```
    /lib/modules  
or  
    /modules
```

Use the kernel utilities 'depmod' and 'modprobe' (resp. in BSD OS 'kldconfig' and 'kldload') to handle dependencies and proper loading of the newly added Dazuko module.

Insert the following line into **esets_daemon** initialization script '/etc/init.d/esets_daemon' before the daemon initialization statement.

```
    /sbin/modprobe dazuko
```

Note that in BSD OS an appropriate line

```
    /sbin/kldconfig dazuko
```

must be inserted into script '/usr/local/etc/rc.d/esets_daemon.sh'.

IMPORTANT: It is highly important to execute the individual steps above exactly in order as they are written. The reason is that in case of kernel module not located within the kernel modules directory, 'modprobe' (resp. 'kldload' in BSD OS) will not be able to handle loading of the module and the system can hang-up.

5.3. On-access scanner using preload LIBC library

In the previous sections we have described integration of on-access scanner powered by Dazuko with the Linux/BSD file system services. In this place we would like to point out that the technique using Dazuko may be non-wished for Linux OS system administrators which carry on the critical systems where source code and/or configuration file appropriate to the currently running kernel is not available or the kernel is rather monolithic than modular. In this case the second discussed on-access scanning technique based on the preload LIBC library comes in handy.

Note that this section is relevant only for Linux OS users. This section contains information concerned with operation, installation and configuration of on-access scanner using preload library **libesets_pac.so**.

5.3.1. Operation principle

On-access scanner **libesets_pac.so** (*ESETS* Preload library based file Access Controller) is a shared objects library that is used as a preload library of LIBC and can become functional during the system start-up. It is thus applicable for file system servers using LIBC calls, for instance ftp server, Samba server etc. Scanning of each file system object is performed upon customizable file access event. The following file access types are supported by the current version:

ON_OPEN events

This file access type is controlled once first bit of the integer parameter 'event_mask' in the *ESETS configuration file* (section [pac]) is 1. In this case all 'open' or 'open64' calls of the LIBC are intercepted.

ON_CLOSE events

This file access type is controlled once second bit of the integer parameter 'event_mask' in

the *ESETS configuration file* (section [pac]) is 1. In this case all 'close', 'dup' and 'dup2' calls of the LIBC are intercepted.

By using this mechanism all opened and closed descriptors tied to regular files are scanned by main *ESETS daemon* for viruses. Based on the result of this scanning the access to the files is denied or allowed.

5.3.2. Installation and configuration

The `libesets_pac.so` installation is done using standard installation mechanism of the preload libraries. One has just to define the environment variable 'LD_PRELOAD' with absolute path of the `libesets_pac.so` library. Please refer also to the manual page `ld.so(8)` to get further information.

IMPORTANT: It is important to note that the 'LD_PRELOAD' environment variable has to be defined just for the network server daemon process (ftp, samba, etc.) we would like to have under control of on-access scanner. Generally it is not recommended to preload LIBC calls in all operating system processes as it can dramatically slow down the performance of the system or even cause the system hang-up. In this sense all mechanisms using '/etc/ld.so.preload' configuration file are not correct as well as mechanisms using 'export LD_PRELOAD' statement. Both would override all relevant LIBC calls in the whole system that will lead to the system hang-up during its initialization.

Thus in order to intercept just relevant file access calls related with just objects within selected file system area, one has to override an executable statement of an appropriate network file system server with the following line

```
LD_PRELOAD=libesets_pac.so COMMAND COMMAND-ARGUMENTS
```

where 'COMMAND COMMAND-ARGUMENTS' is the original executable statement.

Read and edit [global] and [pac] sections of *ESETS configuration file*. Note that for the proper run of on-access scanner it is necessary to define file system objects (i.e. directories and files) that are required to be under control of the preload library. This can be achieved by defining the parameters of 'ctl_incl' and 'ctl_excl' configuration options within [pac] section of the configuration file. For reread of newly created configuration, reload the *ESETS daemon*.

5.3.3. Tips

In order to provide on-access scanner functionality immediately after network file system server start-up, it is good to define environment variable 'LD_PRELOAD' directly within an appropriate network file server initialization script.

EXAMPLE: Let's assume we would like to have on-access scanner catching all file system access events immediately after starting the samba server. Thus within the initialization script concerned with samba daemon (/etc/init.d/smb), we replace the statement

```
daemon /usr/sbin/smbd $SMBDOPTIONS
```

responsible for initialization of `smbd` daemon by the following line

```
LD_PRELOAD=libesets_pac.so daemon /usr/sbin/smbd $SMBDOPTIONS
```

In this manner selected file system objects controlled by Samba will be checked immediately after Samba initialization, i.e. during the system start-up.



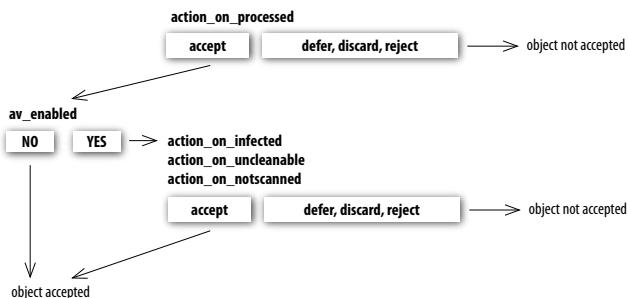
Chapter 6:

Important ESET File Security mechanisms

6.1. Handle Object Policy

The Handle Object Policy (see figure 6-1) is a mechanism that provides handling of the scanned objects depending on their scanning status. The mechanism is based on so-called action configuration options ('action_on_processed', 'action_on_infected', 'action_on_uncleanable', 'action_on_notscanned') combined with Anti-Virus enabling configuration option ('av_enabled'). For detailed information on the options, please refer to the esets.cfg(5) manual page.

Figure 6-1. Scheme of Handle Object Policy mechanism.



Every object processed is at first handled with respect to the setting of the configuration option 'action_on_processed'. Once the option is set to 'accept', the object is handled according to the setting of configuration option 'av_enabled'. Once 'av_enabled' is enabled the object is scanned for virus infiltrations and set of action configuration options 'action_on_infected', 'action_on_uncleanable' and 'action_on_notscanned' is taken into account to evaluate further handling of the object. If action 'accept' has been taken as a result of the three above options or 'av_enabled' is disabled the access to the object is allowed, otherwise the object is blocked.

6.2. User Specific Configuration

User Specific Configuration mechanism is implemented in the product in order to provide administrator with enhanced configuration functionality. It allows to define *ESETS* anti-virus scanner parameters selectively for user accessing file system objects.

Please note that the detailed description of this functionality can be found in esets.cfg(5) manual page and manual pages referenced there. Thus in this section we will only provide short example of user specific configuration definition.

Let's say we use `esets_dac` to control `ON_OPEN` and `ON_EXEC` access events for external disc mounted under directory `'/data'`. The module is subjected to configuration section `[dac]` in the *ESETS configuration file*. The section is as follows:

```
[dac]
agent_enabled = yes
```

```
event_mask = 5
ctl_incl = "/home"
action_on_processed = accept
```

In order to provide individual parameters setting one has to define 'user_config' parameter with the path to the special configuration file where the individual setting will be stored. In the next example we create reference to the special configuration file 'esets_dac_spec.cfg' located within the *ESETS configuration directory*.

```
[dac]
agent_enabled = yes
event_mask = 5
ctl_incl = "/home"
action_on_processed = accept
user_config = "esets_dac_spec.cfg"
```

Once special configuration file referenced from within [dac] section we have to create the file in the *ESETS configuration directory* and provide it with an appropriate individual settings. The next example shows individual setting of parameter 'action_on_processed' for user 'username'.

```
[username]
action_on_processed = reject
```

Note that the section header name of the special section contains identification of the user for which we have created individual setting. The section body then contains individual parameters specified for this identification. Thus with this special configuration all users attempting to access file-system will be processed, i.e. all file system objects accessed by the users will be scanned for infiltrations, with exception to the user 'username' whose access will be rejected, i.e. blocked, in any case.

6.3. Samples Submission System

Samples submission system is an intelligent ThreatSense.NET technology that provides catching of the infected objects found by advanced heuristics method and delivering these objects to the samples submission system server. All virus samples caught by the sample submission system will be processed by the team of ESET virus laboratory department and consequently added into the ESET virus database, if necessary.

NOTE: ACCORDING TO OUR LICENSE AGREEMENT: BY ENABLING SAMPLE SUBMISSION SYSTEM YOU ARE AGREEING TO ALLOW THE COMPUTER AND/OR PLATFORM ON WHICH THE ESETS_DAEMON IS INSTALLED TO COLLECT DATA (WHICH MAY INCLUDE PERSONAL INFORMATION ABOUT YOU AND/OR THE USER OF THE COMPUTER) AND SAMPLES OF NEWLY DETECTED VIRUSES OR OTHER THREATS AND SEND THEM TO OUR VIRUS LAB. THIS FEATURE IS TURNED OFF BY DEFAULT. WE WILL ONLY USE THIS INFORMATION AND DATA TO STUDY THE THREAT AND WILL TAKE REASONABLE STEPS TO PRESERVE THE CONFIDENTIALITY OF SUCH INFORMATION.

In order to turn on Samples Submission System, the samples submission system cache has to be initialized. This can be achieved by enabling configuration option 'samples_enabled' in [global] section of *ESETS configuration file*. In order to enable process of samples delivery to ESET virus laboratory servers it is yet necessary to enable parameter 'samples_send_enabled' in the same section.

User may decide to provide the ESET virus laboratory team with the additional optional information using configuration options 'samples_provider_mail' and/or 'samples_provider_country'. This information will help us to get overview on the infiltration spreading throughout the Internet.

In order to get detailed information on the Samples Submission System, refer to `esets_daemon(8)` manual page.

6.4. World WideWeb Interface

WWW Interface allows user-friendly ESETS configuration, administration and license management.

This module is a separate agent and must be explicitly enabled. For quickstart, set all of these options in *ESETS configuration file* and restart *ESETS daemon*:

```
[wwwi]
agent_enabled = yes
listen_addr = address
listen_port = port
username = name
password = pass
```

(enter all four values as your own ones) and direct your browser to 'https://address:port' (note the *https*) and login with 'name/pass'. There are basic usage instructions on the help page. For more technical details about **esets_wwwi** see the `esets_wwwi(1)` manual page.

Chapter 7:

ESET Security system update

7.1. ESETS update utility

In order to keep the ESET File Security effective, it is necessary to keep its virus signatures database up to date. The `esets_update` utility has been developed for this purpose (see `esets_update(8)` manual page for details). In order to launch update one has to define configuration options 'username' and 'password' in [update] section of *ESETS configuration file*. Note that in case you access the Internet via HTTP proxy additional configuration options 'proxy_addr', 'proxy_port' and optionally 'proxy_username' and 'proxy_password' have to be specified there as well. To trigger an update, enter command:

```
@SBINDIR@/esets_update
```

To provide the highest security for the user, the ESET team collects the virus definitions continuously from all over the world. The new patterns can appear within the database in very short intervals. It is therefore recommended, to trigger an update on a regular basis. Note that *ESETS daemon* is able to provide the periodic update of the system once 'av_update_period' configuration option specified in [update] section of *ESETS configuration file* and the daemon is up and running.

7.2. ESETS update process description

The update process is composed of two stages. First, the mirror of all relevant so-called pre-compiled modules have to be created from the origin ESET server. The pre-compiled modules are downloaded by default into directory

```
@BASEDIR@/mirror
```

Note that the mirror directory path can be redefined using configuration option 'mirror_dir' in section [update] of *ESETS configuration file*.

The *ESETS* modules are divided into two categories; engine category and component category. The modules of component category are currently only for use on the MS Windows OS. Currently the following types of engine category modules are supported: base scanning modules (prefix engine) containing virus signatures database, archives support modules (prefix archs) supporting various file system archive formats, advanced heuristics modules (prefix advheur) containing implementation of so-called advanced heuristics method of virus and worm detection, packed worm scanner modules (prefix pwscan) used on MS Windows OS, utilities modules (prefix utilmod) used on MS Windows OS and ThreatSense.NET technology support modules (prefix charon). These modules are always necessary and therefore are all downloaded by default at each download process. On the other hand the component category modules are platform dependent and language localization dependent and thus the download of component category modules is optional.

After download of the pre-compiled modules the 'update.ver' file is created in the mirror directory as well. This file contains the information about the modules currently stored in the newly created mirror. The newly created mirror thus serves as fully functional modules download server and can be used to create subordinate mirrors, however, some more conditions have to be fulfilled yet. First, there must be a http server installed on the computer where the modules are going to be downloaded from. Second, the modules to be downloaded by other computers have to be placed at the directory path

```
/http-serv-base-path/nod_upd
```

where 'http-serv-base-path' is a base http server directory path, as this is the first place where update utility looks the modules for.

Second part of the update process is the compilation of modules loadable by the ESET File Security scanner from those stored in the local mirror. Typically the following *ESETS* loading modules are created: base module (nod32.000), archives support module (nod32.002), advanced heuristics module (nod32.003), packed worm scanner module (nod32.004), windows utilities module (nod32.005) and ThreatSense.NET support module (nod32.006) in the directory:

```
@BASEDIR@
```

Note that it is exactly the directory where *ESETS daemon* loads modules from and thus can be redefined by using configuration option 'base_dir' in section [global] (resp. [update]) of *ESETS configuration file*.





Chapter 8:

Let us know



Dear user, this guide should have given you a good knowledge about the ESET File Security installation, configuration and maintenance. However, writing a documentation is a process that is never finished. There will always be some parts that can be explained better or are not even explained at all. Therefore, in case of bugs or inconsistencies found within this documentation, please report a problem to our support center

<http://www.nod32.com.hk/support>

We are looking forward to help you solve any problem concerning the product.



Appendix A. PHP License



The PHP License, version 3.01 Copyright (c) 1999 - 2006 The PHP Group. All rights reserved. Redistribution and use in source and binary forms, with or without modification, is permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name "PHP" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact group@php.net.
4. Products derived from this software may not be called "PHP", nor may "PHP" appear in their name, without prior written permission from group@php.net. You may indicate that your software works in conjunction with PHP by saying "Foo for PHP" instead of calling it "PHP Foo" or "phpfoo".
5. The PHP Group may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number. Once covered code has been published under a particular version of the license, you may always continue to use it under the terms of that version. You may also choose to use such covered code under the terms of any subsequent version of the license published by the PHP Group. No one other than the PHP Group has the right to modify the terms applicable to covered code created under this License.
6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes PHP software, freely available from <http://www.php.net/software/>".

THIS SOFTWARE IS PROVIDED BY THE PHP DEVELOPMENT TEAM ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE PHP DEVELOPMENT TEAM OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.