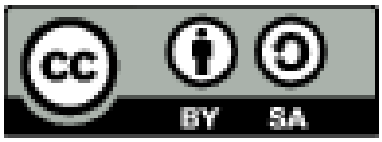
OpenVAS Compendium

A Publication of the OpenVAS Project

Jan-Oliver Wagner, Michael Wiegand, Tim Brown, Carsten Koch Mauthe

Version 1.0.1 as of 20090115

Imprint



Contents

1 Introduction

1.1 About this Compendium

(by Jan-Oliver Wagner)

This compendium was compiled by people involved in the OpenVAS project. The intention is to provide a comprehensive documentation for all aspects of network vulnerability scanning with OpenVAS.

This ranges from instructions on how to use the OpenVAS-Client graphical user interface, run specific test methods and write NASL vulnerability tests up to details on the internal architecture of the actual scan server software.

This compendium is permanently being improved and extended. You may find some sections not comprehensive enough and you may miss some topics entirely.

Further authors are welcome and if you identify important aspects that need to be added here, please coordinate with the OpenVAS team if you plan to get an author for this compendium. It is important that you coordinate before starting to write to avoid concurrent works on the same subject.

The source format of this document is LaTeXwith HyperLaTeXextensions for HTML output. The sources are available as module "openvas-compendium" at the OpenVAS development platform[1].

1.2 About the OpenVAS Project

(by Michael Wiegand)

OpenVAS stands for Open Vulnerability Assessment System and represents a comprehensive tool-chain for network security scanning including a graphical user front-end and incorporating various third-party security applications. The core is a server component with a set of Network Vulnerability Tests (NVTs) to detect security problems in remote systems and applications.

The OpenVAS development team consists of various interested parties from academia and commercial entities as well as individuals. The majority of the team members has a long professional record in security consulting and/or software development.

The OpenVAS project is open to new members. Formal processes exist only where helpful, e.g. getting a team member just means to take part in the processes (design, development, tests, reviews, packaging, support).

All OpenVAS products are Free Software under the GNU General Public License (GNU GPL).

OpenVAS derives from the Nessus project which turned into a proprietary product, but has progressed on its own since then.

[1] http://wald.intevation.org/projects/openvas

1.3 About the OpenVAS Software

(by Michael Wiegand)

The OpenVAS software consists of five distinct parts which are provided and maintained by the OpenVAS projects. The individual parts are:

OpenVAS-Server: This is the core component of OpenVAS. It contains the functionality used for scanning a large number of target servers at a high speed. Scans will always originate from the host where OpenVAS-Server is running; therefore, this machine has to be able reach the intended targets.

The server requires three other modules:

OpenVAS-Libraries: This module contains functionality used by OpenVAS-Server.

OpenVAS-LibNASL: The NVTs are written in the "Nessus Attack Scripting Language" (NASL). This module contains the functionality needed by OpenVAS-Server to interface with NASL.

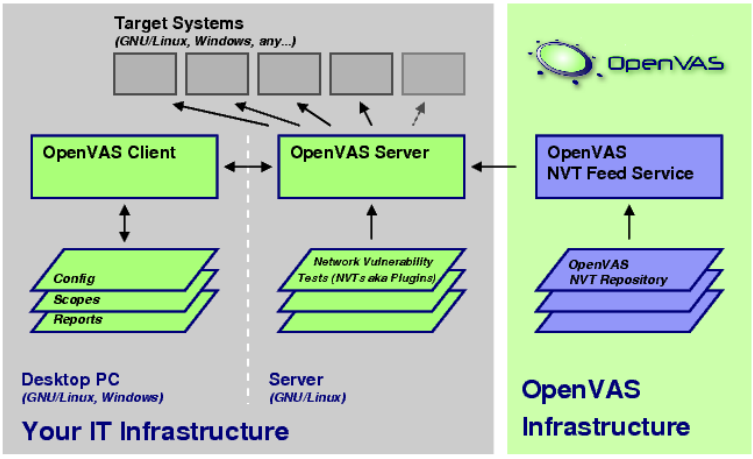
OpenVAS-Plugins: This module contains a base set of NVTs. Please be aware that the update cycle of this module is not intended to ensure the availability of the most recent NVTs. If you need up-to-date NVTs you should consider subscribing to a NVT feed as described in section 2.1.

OpenVAS-Client: OpenVAS-Client controls the OpenVAS server, processes the scan results and displays them to the user. OpenVAS-Client can run on any machine able to connect to the OpenVAS-Server and can control multiple servers.

2 Planning OpenVAS-based Network Auditing

(by Jan-Oliver Wagner)

2.1 Consider Coverage of Available Vulnerability Tests



As one of the first tasks of planning security audits based on OpenVAS, you should compare your targets with the coverage of the currently available OpenVAS vulnerability tests.

Please be aware that the OpenVAS server releases (actually the releases of its module “openvas-plugins”) delivers a base set of tests. The update cycle of this base set is quite long compared to the occurrence of new vulnerabilities and respective NVTs. New or changed tests are promptly distributed via so-called “feed services”.

If you want to test your network against the latest threats, a successful outcome will depend on the quality of the feed service(s) you subscribed to. Although the set of tests provided by “openvas-plugins” will detect a large range of older, well-known vulnerabilities, it will most probably be outdated by the time you use it and will not be able to detect the most recent vulnerabilities. In order to stay up-to-date with the latest security threats, you will need a feed service that provides you with the most recent tests for these threats.

The OpenVAS project maintains a feed of its own:

http://www.openvas.org/openvas-nvt-feed.html

To evaluate your need for an up-to-date feed service, you should think about the following questions:

- To what extent do the available feeds and their maintained families cover my audit target?
- In case you are planning a permanent auditing solution and not just a single shot: How sustainable is the feed service organized?
- Does the feed deliver signatures on the NVTs with a trust level for quality you are comfortable with?

2.2 Choose Location of Scan-Server

If you are planning to use the OpenVAS security scanner in your network, the best location for the machine running the server module depends on the targets you want to evaluate:

- Target is a public server:

  Several tests do follow the very same path as various attacks do: from a remote network. If you are only interested in these tests, you may use a arbitrary location of your OpenVAS server outside of the targeted network.

  However, you are advised to contact the administration of the target systems beforehand and inform them that you are planning on running OpenVAS against their machines. Because OpenVAS will actively look for vulnerabilities on the target system, a scan will under certain circumstances look like a real attack on the target system and might be acted upon legally and/or technically by the administration of the system in question.

- Targets are intranet desktops and servers:

  In this case you should directly coordinate with your system administration.

  Depending on the complexity of the intranet, you may need to find out how to reach specific subnets from your OpenVAS Server installation. In some cases it might be an option to install several OpenVAS Servers.

  For local security checks you need to prepare the target systems for remote access. For unixoid systems this is usually via ssh connections, for Windows it is about SMB shares.

2.3 Choose Type of Scan-Server

2.3.1 Hardware

OpenVAS usually does not need outstanding hardware performance nor does it consume considerable disk space. A standard server configuration will be sufficient in most cases.

Most users use OpenVAS with a IA32 architecture. 64bit and PowerPC architectures are also supported, but not tested to the same extent.

2.3.2 Operating System

Only unixoid systems are supported by OpenVAS and out of this group, Linux distributions are tested most extensively. xBSD should work but has not been extensively tested.

When selecting the operating system you should consider some aspects:

- OpenVAS can either be compiled out of source code or be installed via prepared binary packages. In case you prefer the latter method, you should check which operating systems are supported best with installation packages.
- Several security tools are integrated into OpenVAS. If you plan to use them, you should make sure you have them available with your selected operating system. Again, some platforms are better supported with binary installation packages than others.

3 Installing and Configuring OpenVAS-Server

(by Tim Brown, Jan-Oliver Wagner and Michael Wiegand)

3.1 Installing Binary Packages

Binary packages for the major Linux distributions and some other platforms are available for download from the OpenVAS website or from services provided by third parties.

Please note that the amount of configuration that is done during installation depends on the distribution and the package maintainers. Refer to section 3.3 for the complete configuration instructions.

If there are no (or not all) OpenVAS modules packaged for the distribution of your choice, installation from source (see section 3.2) is usually possible on most Unix-based platforms. You might also want to let the maintainers of this distribution know that you would like to see OpenVAS packaged for their distribution and that they can turn to the OpenVAS development team should they need more information.

3.1.1 Debian and Ubuntu

OpenVAS-Server is currently being integrated into Debian and Ubuntu. This means that you can install OpenVAS-Server using the `apt-get` mechanism.

While the integration into Debian and Ubuntu is in progress, some parts of OpenVAS may not yet be available from the official repositories, but might be available from other repositories. Please refer to the OpenVAS website for up-to-date information regarding package availability.

3.1.2 Gentoo

OpenVAS packages for Gentoo are available in the Gentoo portage. Please refer to the Gentoo documentation and the OpenVAS website for information on installing these packages.

3.1.3 RPM-Based Distributions

Packages for some versions of OpenSUSE, Fedora, Mandriva and other RPM-based distributions are available from unofficial repositories that can be found at

http://download.opensuse.org/repositories/security:/openvas:/ and http://www.atomicorp.com/channels/atomic/

Please refer to the documentation provided with your distribution for information on adding repositories and installing these packages.

3.1.4 FreeBSD

The FreeBSD Ports and Packages Collection provides ports and packages for all OpenVAS modules.

The following commands can be used to compile and install the FreeBSD ports on your FreeBSD system:

```
cd /usr/ports/security/openvas-libraries/ && make install clean
cd /usr/ports/security/openvas-libnasl/ && make install clean
cd /usr/ports/security/openvas-server/ && make install clean
cd /usr/ports/security/openvas-plugins/ && make install clean
```

If you would rather use binary packages, you will want to use the following commands:

```
pkg_add -r openvas-libraries
pkg_add -r openvas-libnasl
pkg_add -r openvas-server
pkg_add -r openvas-plugins
```

3.2 Compiling OpenVAS-Server from Source Packages

3.2.1 Latest source code release

The download link for the latest source code release can be found in the “Download” section on the OpenVAS website.

If you are using a source code release please be aware that the installation process might copy files to places different from the ones expected by you or the distribution you are using. This may lead to unexpected results, especially if you install releases from different series or try to remove old copies of OpenVAS if you are not careful. Because of this, we recommend that you configure the modules with a prefix (i.e. `./configure prefix=/opt/openvas`) and use different prefixes for different series. This way you can easily isolate all the files belonging to this particular OpenVAS version; if you want to remove a specific OpenVAS installation from you can simply remove the particular subdirectory.

Download the four “.tar.gz” source code archives and unpack with “tar -xzf openvas-MODULE-N.N.N.tar.gz”. Compiling from source is currently geared towards GNU/Linux systems, but may work as well in other environments.

You have to compile and install the packages in the the following sequence:

1. openvas-libraries
2. openvas-libnasl
3. openvas-server
4. openvas-plugins

Now read the file `INSTALL_README` inside the directory “openvas-libraries” for the next steps.

Repeat for each module and read the corresponding `INSTALL` or `README` files.

3.2.2 Most current state of development (directly from the source code management system)

You need subversion to retrieve the code:

```
$ svn checkout
https://svn.wald.intevation.org/svn/openvas/trunk/openvas-libraries
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-libnasl
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-server
$ svn checkout https://svn.wald.intevation.org/svn/openvas/trunk/openvas-plugins
```

Now read the file `INSTALL_README` inside the directory "openvas-libraries" for the next steps.

Repeat for each module and read the corresponding `INSTALL` or `README` files.

Although the OpenVAS team is committed to maintaining a high code quality, please be aware that you are using a development state that may be incomplete and unstable and should not be used in production environments.

3.3 Configuring OpenVAS-Server

(by Michael Wiegand)

After installing OpenVAS-Server some additional steps are needed to get your OpenVAS installation up and running. This section provides the information you need to generate a server certificate and add users to your installation. These instructions are relevant for both the 1.0 and the 2.0 series.

If you have installed OpenVAS from packages provided by your distribution, be aware that there might be differences between the instructions provided by this compendium and the steps necessary on your distribution. Please refer to the documentation provided by your distribution for additional information.

Also note that you might need to install additional software if you are planning to use the tools described in chapter 8.

3.3.1 Generating a Server Certificate

For security reasons, communication between the OpenVAS server and client is only possible through SSL encrypted connections. In order to establish an SSL encrypted connection, the server needs to have an SSL certificate. If the machine OpenVAS-Server is running on does not have a certificate, you will have to generate one yourself.

The easiest way to do this is through the `openvas-mkcert` script provided by the OpenVAS-Server package. This will generate two certificates: one certificate for a local certificate authority (CA) and a second certificate for the OpenVAS server which is signed by the CA and is presented to connecting clients.

However, in case you want or have to consider a X.509 Public Key Infrastructure (PKI), you may of course use a certificate signed by the respective CA, which is e.g. done by your trust center.

3.3.2 Adding New Users

In order to use an OpenVAS server, a client needs to have a user account on the server. The OpenVAS-Server package provides the `openvas-adduser` script to simplify the creation of user accounts. Using `openvas-adduser`, you can specify whether the user should use a password or a certificate to authenticate himself and optionally restrict the access rights of the user.

Restricted access rights can be useful to prevent users from scanning arbitrary hosts or networks. You can specify rules that restrict an user to certain hosts or subnets or even prevent him from scanning any host but his own.

The correct syntax for user rules is:

```
accept|deny ip/mask
```

and

```
default accept|deny
```

Where `mask` is the CIDR netmask of the rule.

The `default` statement must be the last rule and defines the policy for the user.

The following rule set will allow the user to test 192.168.1.0/24, 192.168.3.0/24 and 172.22.0.0/16, but nothing else:

```
accept 192.168.1.0/24
accept 192.168.3.0/24
accept 172.22.0.0/16
default deny
```

The following rule set will allow the user to test whatever he wants, except the network 192.168.1.0/24:

```
deny 192.168.1.0/24
default accept
```

The keyword `client_ip` is replaced at runtime by the IP address of the user. If you want to restrict the user to be able to scan only the system he is connecting from, you can use the following ruleset:

```
accept client_ip
default deny
```

3.3.3 Advanced Configuration

If you need to make changes to the default OpenVAS-Server configuration, you can do so in the configuration file located at `/etc/openvas/openvasd.conf`.

The following settings can be configured through the openvasd.conf configuration file (note: the default values for your distribution may differ from the default values described here):

plugins_folder This setting configures the path where the NVT scripts can be found.
(default value: /lib/openvas/plugins)

max_hosts The maximum number of hosts that will be tested simultaneously.
(default value: 30)

max_checks The maximum number of checks that will run simultaneously against a given host.
(default value: 10)

be_nice Niceness. If set to 'yes', openvasd will renice itself to 10.
(default value: no)

logfile The file used to log activity. If this value is set to 'syslog', OpenVAS-Server will use syslogd for logging.

(default value: /var/log/openvas/openvasd.messages)

log_whole_attack This setting controls how detailed the log should be. If this option is set to 'no', only the start and end time of the scan is logged. If set to 'yes', OpenVAS-Server will log more information, including the time each plugin took to execute. Be aware that this may cause OpenVAS-Server to use more hard disk space and to access the hard disk more often during the scan.

(default value: no)

log_plugins_name_at_load This setting controls whether the names of the plugins that are loaded by the server should be logged.

(default value: no)

dumpfile This option configures the name of the file that should be used for debugging output. If this option is set to '-', debugging output will be written to stdout.

(default value: /var/log/openvas/openvasd.dump)

rules The filename for the server rules file.

(default value: /etc/openvas/openvasd.rules)

users The filename for the user database.

(default value: /etc/openvas/openvasd.users)

cgi_path The default CGI paths to check, separated by colons(':').

(default value: /cgi-bin:/scripts)

port_range The range of ports that will be scanned by the port scanners. If this setting is set to 'default', OpenVAS-Server will scan the ports specified in the file found at `/var/lib/openvas/openvas-services`.

(default value: default)

optimize_test Security tests may request to be launched if and only if certain information gathered by other tests exists in the knowledge base, or if and only if a given port is open. If this option is set to 'yes', it will speed up the test, but may cause the OpenVAS server to miss some vulnerabilities.

(default value: yes)

language The language to use in plugin description. Currently the values 'english' and 'french' are supported.

(default value: english)

checks_read_timeout The read timeout (in seconds) for the sockets used while scanning.

(default value: 5)

non_simult_ports This option can be used to specify a list of ports or services against which plugins should not be run simultaneously.

(default value: 139, 445)

plugins_timeout The maximum lifetime of a plugin (in seconds).

(default value: 320)

safe_checks Some security checks may harm the target server, by disabling the remote service temporarily or until a reboot. If this option is set to 'yes', the OpenVAS server will rely on banners instead of actually performing a security check. This will result in a less reliable report, but is less likely to disrupt functionality on the target system during a test.

(default value: yes)

auto_enable_dependencies If this option is set to 'yes', OpenVAS-Server will automatically enable plugins which are needed by the plugins selected by the user.

(default value: yes)

silent_dependencies If this option is set to 'yes', output from plugins which were enabled automatically will not be send to the client.

(default value: yes)

use_mac_addr Designate hosts by MAC address, not IP address; this can be useful in DHCP networks.

(default value: no)

save_knowledge_base This option controls whether the knowledge base created during the scan should be saved to disk.

(default value: no)

kb_restore This setting controls whether the knowledge base should be restored for each test.

(default value: no)

only_test_hosts_whose_kb_we_dont_have If this option is set to 'yes', OpenVAS-Server will only test the hosts that are not yet in the knowledge base. This can be used to scan new hosts once if they appear in a subnet for the first time, for example.

(default value: no)

only_test_hosts_whose_kb_we_have If this option is set to 'yes', OpenVAS-Server will only test the hosts that are already in the knowledge base. This is useful for scanning only a set of host that are already known to the server.

(default value: no)

kb_dont_replay_scanners If this option is set to 'yes' and the option kb_restore has been enabled, scanner plugins will not be launched if they have already been launched in the past.

(default value: no)

kb_dont_replay_info_gathering If this option is set to 'yes' and the option kb_restore has been enabled, information gathering plugins will not be launched if they have already been launched in the past.

(default value: no)

kb_dont_replay_attacks If this option is set to 'yes' and the option kb_restore has been enabled, attack plugins will not be launched if they have already been launched in the past.

(default value: no)

kb_dont_replay_denials If this option is set to 'yes' and the option kb_restore has been enable, denial of service plugins will not be launched if they have already been launched in the past.

(default value: no)

kb_max_age This option sets the maximum age of the knowledge base (in seconds).

(default value: 864000)

slice_network_addresses If this option is set to 'yes', OpenVAS will not scan a network sequentially (10.0.0.1, 10.0.0.2, 10.0.0.3), but will attempt to slice the workload throughout the whole network (i.e.: 10.0.0.1, 10.0.0.127, 10.0.0.2, 10.0.0.128).

(default value: no)

nasl_no_signature_check If this option is set to 'yes', OpenVAS-Server will not check the signatures of the NASL scripts and will run scripts even if they have no or no valid signature. Be aware that setting this option to 'yes' does pose a security risk. However, at the current stage of OpenVAS development, signatures are not yet included in the openvas-plugins releases available from the OpenVAS website. If this option is set to 'no', you will only be able to use a very limited number of plugins until you have synchronized your plugin collection with an NVT Feed Service providing signatures. For this reason, this option will default to 'yes' until signatures are included with all plugins.

(default value: yes)

3.4 Configuring NVT Feeds

(by Jan-Oliver Wagner and Tim Brown)

This section explains how NVT Feed Services work in general and how you can use a Feed Service to keep up-to-date with the latest NVTs.

An OpenVAS NVT Feed Service provides a set of NVTs (i.e. ".nasl" and ".inc" files) which can be downloaded into your OpenVAS server installation.

In fact, only changed and new NVTs will be downloaded along with their signature files (".asc") and an overall "md5sums". This synchronization process uses the RSYNC technology. The signatures are only relevant for you if you configure your OpenVAS server to execute only trusted NVTs.

3.4.1 Prerequisites

Apart from openvas-plugins (version 0.9.1 or higher), which contains the "openvas-nvt-sync" script, you need to have the standard `rsync` and `md5sum` tools available on the system where your OpenVAS server instance is running. If you installed OpenVAS from a binary package, the package management of your distribution should have taken care to meet these dependencies already.

3.4.2 Performing a synchronization with an OpenVAS NVT Feed

To synchronize your NVT repository with an OpenVAS NVT Feed, you need to follow these steps:

1. Check the configuration of the synchronization command: Usually you will find this shell script installed as `/usr/sbin/openvas-nvt-sync`.

   You should verify that the variables "NVT_DIR" and "FEED" are correct. This should be the case for NVT_DIR if you did not deviate from the standard build and install routine. For FEED there is currently only the pre-configured one available anyway. So, just don't change it.

2. Run the synchronization command:

   ```
   # openvas-nvt-sync
   ```

   It will connect to the currently only available NVT feed. At the end, it will verify the md5 checksums of all synchronized files. If any of them fails, an error is reported. In this case you should retry a couple of minutes later (reasons for failures could be network lags or that the feed was updated at the same time). If the problem occurs again, please report it to the OpenVAS discussion mailing list. If you want to automatically update you NVT collection, you can a description of the required steps in chapter 3.4.4.

3. Restart the OpenVAS server (openvasd):

```
# kill -1 PID
```

Where PID is the process ID of the main openvasd. You may see in the "openvas-nvt-sync" script how this should work ideally, but currently it does not work. You might consider using the "killall openvasd" command if you really know what this means.

3.4.3 Available NVT Feed Services

For demonstration purposes, the OpenVAS project offers a simple NVT feed at rsync://rsync.openvas.org:/nvt-feed. It is pre-configured in the "openvas-nvt-sync" tool.

However, the NVTs are signed with the OpenVAS Transfer Integrity certificate.

3.4.4 Automatically Updating an NVT Feed

- Create a script called "openvas-update" and save it somewhere such as "/usr/local/bin":

```
#!/bin/sh

temp=`tempfile`
openvas-nvt-sync 2>&1> $temp
if [ $? -ne  0 ]
then
cat $temp
fi
rm $temp
if [ -f /var/lib/run/openvasd.pid ]
then
pid=`cat /var/lib/run/openvasd.pid`
kill -1 $pid 2>/dev/null
fi
```

- Update your crontab to contain a line such as:

```
25 4 * * * root /usr/local/bin/openvas-update
```

3.5 Managing NVT signatures

(by Jan-Oliver Wagner)

This section explains what you need to do to allow your OpenVAS-Server to execute only signed NVTs with a trust level you decide.

Currently, some signed NVTs are available by using the command `openvas-nvt-sync` which is included in openvas-plugins. The signatures correspond to the certificate "OpenVAS Transfer Integrity" available on the OpenVAS website at http://www.openvas.org/trusted-nvts.html.

3.5.1 What is a Signature?

A clever method is applied to compute a unique checksum for a file. If only a single character in the file changes, the checksum will change as well. This checksum is digitally signed in a way that you can test with a public

certificate whether a certain key was used to create the signature. Such a key and certificate do always form a pair that relates them to each other. If the signed file has been modified by a third party, the signature will be broken. In this case you should not trust the file.

If the signature is not broken, the question remains if you trust the owner of the key. If you decided to do so (and there any many ways and supporting technologies to manage this), you can accept the file as trustworthy.

In other words, the checksum ensures the integrity of the file and will change if the file was changed between the NVT feed server and your system. The signature on the other hand indicates the authenticity of the file – by signing the checksum, the manager of the Feed Service signifies that the file available from the feed server has been tested and is authentic.

This way you can verify that the file in your possession is indeed the same file that was tested by the feed manager. It is your responsibility to verify that the manager of the Feed Service is indeed the person he or she claims to be and to make sure the tests performed by this person are sufficient for you.

3.5.2 The Signature Format

The signatures for OpenVAS NVTs and associated files (.nasl, .inc and .nes) are standard so-called "ASCII-armored detached OpenPGP signatures" created with GnuPG. This format features:

- Multiple signer keys possible
- Site administrators can decide which keys to trust
- Signatures can be created and verified with widely available tools like GnuPG
- detached signatures do not require changes to the signed file (like inline signatures would)

The name of the signature file is the name of the signed file with the added extension ".asc". That is, the name of the signature file for a file "myscript.nasl" is "myscript.nasl.asc".

Please note the difference to Nessus: Nessus signatures were inline x509-based signatures. This concept does not consider multiple signatures. Please be aware that OpenVAS no longer supports Nessus signatures and will consider plugins unsigned even if the have a valid Nessus signature.

3.5.3 The Signature Verification Process

The signature verification of the OpenVAS server is activated by setting "nasl_no_signature_check = no" in the OpenVAS-Server configuration (see section 3.3.3).

At start-up the openvas daemon (openvasd) checks all signatures for validity. Only fully trusted files are considered by the server and thus loaded and made available to OpenVAS client.

The trust check uses a special list of certificates managed for the OpenVAS server. It is a standard GnuPG keyring located by default in /etc/openvas/gnupg.

When OpenVAS verifies the signature for a file it checks all signatures contained in the signature file and all signatures must be fully valid. This means that all of the following criteria must be fulfilled for all signatures which have signed this particular file:

- The certificate must be present in the keyring.
- The key must be fully valid.
- The signature must be valid.

If any of the signatures does not meet all of these criteria, that file is considered untrustworthy and will not be executed at all. If all signatures meet the criteria, the script is trusted fully and may execute all functions. If no signature file exists, the script is not executed at all.

Again, please note the difference to Nessus: For Nessus signatures, three levels were distinguished: no signature, a bad signature and a good signature. Plugins with no signature were still executed, but in a "restricted" mode where no functions that were regarded critical could be executed. OpenVAS explicitly only distinguishes between fully trusted and untrusted files.

3.5.4 How to Add a Certificate

To add a certificate to the OpenVAS Server keyring, use this command:

```
# gpg --homedir=/etc/openvas/gnupg --import certificate-file.asc
```

See the OpenVAS website at http://www.openvas.org/trusted-nvts.html for available certificate files.

3.5.5 How to Set Trust

To express trust into keys that signed NVTs you need a signing key for your OpenVAS installation. You can use an existing key, or you can generate a new one:

```
# gpg --homedir=/etc/openvas/gnupg --gen-key
```

This needs to be done only once for an OpenVAS-Server installation.

For OpenVAS to trust a signature, the key used to create the signature has to be valid. A certificate corresponding to this key that was just imported has an unknown validity and thus is considered not valid.

In order to trust a certificate for your purpose, you have to sign it. The recommended way is to use local signatures that remain only in the keyring of your OpenVAS Server installation.

To sign a certificate you need to know its KEY_ID. You can get it either from the OpenVAS website or via a "list-keys" command. Then you can locally sign:

```
# gpg --homedir=/etc/openvas/gnupg --list-keys

# gpg --homedir=/etc/openvas/gnupg --lsign-key KEY_ID
```

Before signing you should be absolutely sure that you are signing correct certificate. You may use its fingerprint and other methods to convince yourself.

3.5.6 How to Remove a Certificate

```
# gpg --homedir=/etc/openvas/gnupg --delete-keys KEY_ID
```

3.5.7 Manual Signature Verification

In case you want to manually verify the validity of a .nasl file, you can either run GnuPG:

```
$ gpg --homedir=/etc/openvas/gnupg gpg --verify script.nasl.asc script.nasl
```

Or you can use the standalone nasl interpreter:

```
$ openvas-nasl -p script.nasl
```

The -p option means that the script is only parsed and not executed.

To debug the signature verification done by the nasl interpreter, use the -T option to enable the trace mode. The signature verification will leave more detailed information about the verification and the signatures found in the trace file.

4 OpenVAS File Locations

(by Michael Wiegand)

The OpenVAS components will install files to different locations on your system during installation. This chapter aims to give you a rough idea of what will be installed where.

Please note that the locations presented here refer to a standard installation from source with prefix=/usr/local. These locations may differ according to the choices made when configuring and compiling the individual components; if you installed OpenVAS from binary packages provided for your distribution, the package maintainers may have chosen different locations according to the distributions guidelines.

Be aware that most of these directories will of course contain files additional to the ones described in the following sections; this chapter only points out which files will be placed by OpenVAS in certain locations.

4.1 Executables for users (PREFIX/bin)

This directory holds executables that are either relevant for the user or for the source code compilation procedure:

- `libopenvas-config`: compile parameters for modules that use openvas-libraries
- `openvas-libnasl-config`: compile parameters for modules that use openvas-libnasl
- `openvasd-config`: compile parameters for modules that use openvas-server
- `openvas-mkcert-client`: a tool for generating client certificates
- `openvas-nasl`: a standalone NASL interpreter for users who want to test a NASL script for syntax errors.
- `OpenVAS-Client`: The graphical user interface.

4.2 Server configuration (PREFIX/etc/openvas)

This directory holds the system-wide configuration data for OpenVAS including the file for the access rules and the user database.

- `openvasd.conf`: the main server configuration as described in section 3.3.3.
- `openvasd.rules`: rules to restrict sets of target systems for OpenVAS users as described in chapter 3.3.2.
- `gnupg`: the keyring and trust levels for defining which NVTs are allowed for execution.

4.3 Compilation files (PREFIX/include/openvas)

This directory contains the C header files for OpenVAS. They are only relevant for compilation of OpenVAS on the respective system.

4.4 Libraries (PREFIX/lib)

This directory holds the library files, static (.a) and dynamic (.so) of the OpenVAS components openvas-libraries and openvas-libnasl.

4.5 NVTs (PREFIX/lib/openvas/plugins)

This is the directory where all `.nasl` and `.inc` files are located.

4.6 Executables for server (PREFIX/sbin)

This directory holds several executables useful for OpenVAS system administrators:

- `openvasd`: The OpenVAS server itself.
- `openvas-adduser`: Routine to add a OpenVAS user.
- `openvas-mkcert`: Routine to create a server certificate.
- `openvas-rmuser`: Routine to remove a OpenVAS user.
- `openvas-nvt-sync`: Routine for synchronization with NVT Feeds.

4.7 Manual pages for users (PREFIX/share/man)

This directory contains the standard unixoid manual pages for the individual executables relevant for OpenVAS users and for the source code compilation process.

4.8 Manual pages for server (PREFIX/man)

This directory contains the standard unixoid manual pages for the individual executables relevant for OpenVAS administration.

4.9 Server installation specific data (PREFIX/var/lib/openvas)

This directory contains data which is specific to each OpenVAS installation:

- `CA`: Public certificates created for OpenVAS Server SSL connections.
- `private/CA`: Corresponding private certificates.
- `users/`: Files for each configured OpenVAS user.
- `openvas-services`: List of known services with their port numbers.
- `services.tcp`: TCP services.
- `services.udp`: UDP services.

4.10 Log files (PREFIX/var/log/openvas)

This directory holds the log files of the OpenVAS-Server component:

- `openvasd.dump`
- `openvasd.messages`

4.11 Server process information (PREFIX/var/run)

This directory contains runtime data of a running OpenVAS system:

- `openvasd.pid`: Process identifier file.

4.12 User data (HOME/)

All user specific runtime data (managed by OpenVAS-Client) are located in the home directory of the user:

- `.openvas/TTT/`: Directory with all files that relate to task „TTT”.
- `.openvas/TTT/nessusrc`: [OpenVAS 1.0] Task-specific configuration.
- `.openvas/TTT/openvasrc`: [OpenVAS 2.0] Task-specific configuration.
- `.openvas/TTT/SSS/`: Directory with all files that relate to scope „SSS” of task „TTT”.
- `.openvas/TTT/SSS/nessusrc`: [OpenVAS 1.0] Scope-specific configuration.
- `.openvas/TTT/SSS/openvasrc`: [OpenVAS 2.0] Scope-specific configuration.
- `.openvas/TTT/SSS/RRR/`: Directory with all files that relate to report „RRR” of scope „SSS” of task „TTT”.
- `.openvas/TTT/SSS/RRR/nessusrc`: [OpenVAS 1.0] Configuration that was applied when creating this report „RRR”.
- `.openvas/TTT/SSS/RRR/openvasrc`: [OpenVAS 2.0] Configuration that was applied when creating this report „RRR”.
- `.openvas/TTT/SSS/RRR/report.nbe`: The actual contents of report „RRR” in a text format.
- `.openvas/TTT/SSS/RRR/report.nbe.cnt`: Counter for security holes, warnings and hints collected with this report (can be recalculated from report.nbe).
- `.openvas/TTT/SSS/RRR/nessus_plugin_cache`: [OpenVAS 1.0] Cached NVT descriptions that define the NVTs called to produce report „RRR”.
- `.openvas/TTT/SSS/RRR/openvas_nvt_cache`: [OpenVAS 2.0] Cached NVT descriptions that define the NVTs called to produce report „RRR”.
- `.openvasrc`: Default configuration for OpenVAS-Client.
- `.openvasrc.cert`: Certificates of OpenVAS Servers connected in the past that were accepted by the user as correct.

5 Installing and Configuring OpenVAS-Client

(by Tim Brown, Jan-Oliver Wagner and Michael Wiegand)

5.1 Installing Binary Packages

Binary packages for the major Linux distributions and some other platforms are available for download from the OpenVAS website or from services provided by third parties.

If OpenVAS-Client is not yet packaged for the distribution of your choice, installation from source (see section 5.2) is usually possible on most Unix-based platforms. You might also want to let the maintainers of this distribution know that you would like to see OpenVAS packaged for their distribution and that they can turn to the OpenVAS development team should they need more information.

5.1.1 Debian and Ubuntu

OpenVAS-Client is an official package for the Debian distributions "unstable" ("Sid") and "testing" ("Lenny") and for Ubuntu releases onward from release 8.10 ("Intrepid Ibex"). This means that you can install OpenVAS-Client using the `apt-get` mechanism if you are using one of these distributions. Please refer to the OpenVAS website for up-to-date information regarding package availability for older versions.

5.1.2 Gentoo

OpenVAS-Client packages for Gentoo are available in the Gentoo portage. Please refer to the Gentoo documentation and the OpenVAS website for information on installing these packages.

5.1.3 RPM-Based Distributions

Packages for some versions of OpenSUSE, Fedora, Mandriva and other RPM-based distributions are available from unofficial repositories that can be found at

http://download.opensuse.org/repositories/security:/openvas:/ and http://www.atomicorp.com/channels/atomic/

Please refer to the documentation provided with your distribution for information on adding repositories and installing these packages.

5.1.4 Windows XP SP2

Packages for Microsoft Windows XP SP2 are available from the OpenVAS website.

5.1.5 FreeBSD

The FreeBSD Ports and Packages Collection provides ports and packages for OpenVAS-Client.

The following commands can be used to compile and install the FreeBSD port on your FreeBSD system:

```
cd /usr/ports/security/openvas-client/ && make install clean
```

If you would rather use the binary package, you will want to use the following commands:

```
pkg_add -r openvas-client
```

5.2 Compiling OpenVAS-Client from Source Packages

5.2.1 Latest source code release

If you are using a source code release please be aware that the installation process might copy files to places different from the ones expected by you or the distribution you are using. This may lead to unexpected results, especially if you install releases from different series or try to remove old copies of OpenVAS-Client if you are not careful. Because of this, we recommend that you configure the modules with a prefix (i.e. `./configure prefix=/opt/openvas`) and use different prefixes for different series. This way you can easily isolate all the files belonging to this particular version of OpenVAS-Client; if you want to remove a specific OpenVAS-Client installation from you can simply remove the particular subdirectory.

Download the ".tar.gz" source code archive from the download section of the OpenVAS website and unpack with "tar -xzf openvas-client-N.N.N.tar.gz". Compiling from source is currently geared towards GNU/Linux systems, but may work as well in other environments.

Now read the README file inside the new directory for further instructions.

5.2.2 Most current state of development (directly from the source code management system)

You need subversion to retrieve the code:

```
 $ svn checkout
https://svn.wald.intevation.org/svn/openvas/trunk/openvas-client
```

Change to the new directory and follow the instructions of the README file.

Although the OpenVAS team is committed to maintaining a high code quality, please be aware that you are using a development state that may be incomplete and unstable and should not be used in production environments.

6 Using OpenVAS-Client

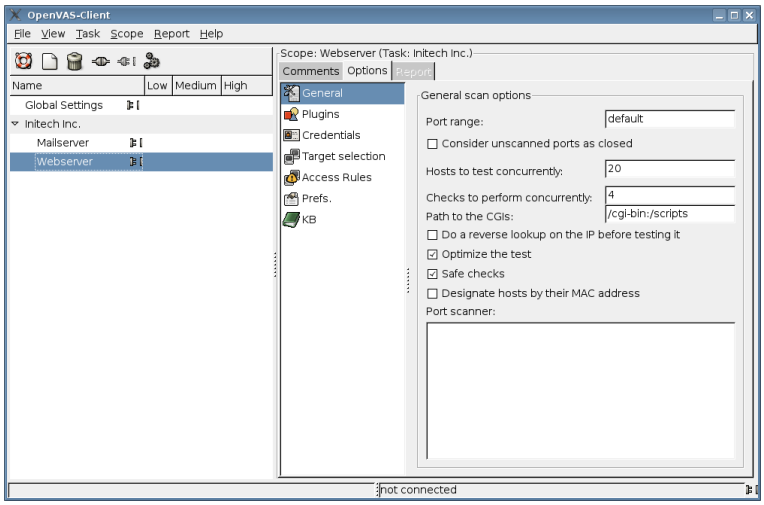
(by Jan-Oliver Wagner)

This section describes the basic components of OpenVAS-Client and how to use them for day-to-day use as well as more specific features that might be of interest for advanced users.

This documentation assumes OpenVAS-Client in version 2.0-beta1. Newer version might offer additional or changed functionality. In this case, please refer to the OpenVAS website for information or support.

6.1 The Main Window

The main window of OpenVAS-Client is split into two major sections: On the left-hand side is the treelist with an overview of the locally stored tasks, scopes and reports. On the right-hand side there is a notebook with pages for comments, options and reports. This is the place where a security scan can be configured, commented upon and its result reviewed.



When you first start OpenVAS-Client, you will see only one entry in the list: Global Settings. These settings you see on the first start-up are the default settings shipped with OpenVAS-Client. They do not cover a specific selection of plugins since a connection to an OpenVAS server is required to make a plugin selection. You can establish a connection with a server and then specify a global default plugin selection for later use.

6.1.1 Tasks

Tasks are intended to cover all activities of a major topic. A task could be “Test the machines of our headquarter” or “Customer XYZ Inc.”.

A task can contain a comment that explains the task in more detail. Also any type of additional info or reminder can be entered in the comment area, e.g. when to run the next series of scans or based on which contract the scans are performed.

A task has neither options nor a report. Apart from the comment, it just contains a number of scopes.

Possible operations for tasks are:

New Adds a new task with the title “unnamed”.

Rename Allows to edit the title in the treelist either by clicking on the title or by selecting the corresponding menu item.

Remove This means the removal of all scopes associated with this task and thus the removal action prompts for a confirmation.

6.1.2 Scopes

A scope can be seen as a sub-task. It defines a certain security scan. The title should indicate the scope of this scan, e.g. “Careful scan of web server production system”, “Aggressive scan of web server alpha test system” or “All Sun workstations”.

Comments can also be specified for each scope and may explain the scope in more detail as well as contain any other helpful hints regarding the respective scope.

The scope is associated with a full set of options for the security scan. When creating a new scope, the general settings are copied. The scan options are explained in detail in a later section.

It should be noted that a connection to an OpenVAS server is established within the context of a specific scope. If a scope is connected to the server, a scan based on the settings for this scope can be executed. An icon to the right of the scope title indicates the connection status of this scope. This means a task can contain a selection of scopes that connect to different OpenVAS servers with different plugins.

Next, a scope may contain a number of reports. Whenever a scope is successfully executed, the resulting report is added to its list of reports. Also, importing a report from a file or from an OpenVAS server will add the report to the currently selected scope.

Please note that changes to a scope are always and only stored when executing a scan. If you make changes to a scope like a new plugin selection and leave OpenVAS-Client without running a scan, these changes will be discarded.

Possible operations for scopes are:

Execute The scope configuration is stored to disk and a security scan is executed with the currently specified options using the currently connected OpenVAS server.

New Adds a new scope entitled “unnamed” as part of the currently selected task. As a default the global settings are copied. Note, that only explicitly saved global settings are taken as defaults. If you changed them inside OpenVAS-Client without saving them, they will have no effect.

Rename Allows to edit the title in the treelist either by clicking on the title or by selecting the corresponding menu item.

Remove This means the removal of all associated reports and thus the removal action prompts for a confirmation.

Move to task It is possible to move a scope with all of its reports from one task to another. This menu item has subitems which represent the other tasks. Select one of them and the scope will be moved.

Open You can load a scope file and add it to the current task with this menu command. Note that here only the parameter sets are covered but not the reports which are represented by files of their own. So, opening and saving (see below) scopes is a method to transfer your settings to someone else or to create a copy of the current scope for yourself.

Save As Saves the current scope to a file (which is of openvasrc type). Note that only the parameter sets are stored but not the reports. See the description of "Open" above for more hints.

6.1.3 Reports

A report is the result of a security scan. It contains the results of the executed plugins associated with the corresponding subnet, host, port and severity.

Managed within OpenVAS-Client, additionally a comment and, if available, the scan options leading to the report, can be stored. This additional information is not contained in the plain OpenVAS report files and thus gets lost when being exported. This also means that imported reports have no comments or scan options associated.

Possible operations for reports are:

Remove Deletes the report and its comments and options. The user is prompted to confirm the removal.

Import Allows to import a report from a file. The standard exchange format is NBE (files suffixed ".nbe"). The file selection dialog allows to select the desired report file. An error hint will be displayed if the file format was not NBE. Else, the report is added to the currently selected scope. Neither comments nor options will be there for a report imported from a NBE file.

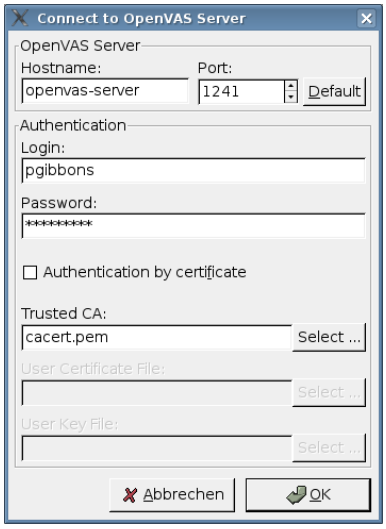
Export Allows to export the currently selected report either in a re-importable format (NBE) or in a format for further processing or presentation (XML, HTML, HTML with Pies and Graphs, LaTeX, ASCII Text and PDF). It is recommended to use NBE if re-importing is planned and to use PDF for creating simple report documents that need no further editing. Use one of the others if you want to further process the report or integrate it into your own document style.

Print Selecting the Print command will create a PDF version of the current report and tries to run a PDF viewer installed on your system. OpenVAS-Client will try a number of well-known PDF viewers; if there is a PDF viewer installed on your system and this menu item does not work, please check if the executable file for your PDF viewer is available in your system path.

6.2 Authentication

OpenVAS-Client needs to connect to an OpenVAS server in order to retrieve the available plugins and to actually execute a security scan. Starting with OpenVAS-Client 2.0.0, the client will display a notification whenever new NVTs are found on the server.

OpenVAS-Client can handle multiple connections to different servers. Each scope has a connection of its own. Additionally, the Global Settings can be connected to an OpenVAS server to define default plugin selections and plugin parameters. Note that only explicitly saved Global Settings are used as defaults for new scopes.



The connection status is indicated with a icon in the tasks/scopes/reports treelist next to the title of the global settings or a scope. Only scopes are connected with the OpenVAS server.

More information on the connection status is shown in the statusbar at the bottom of the main window. There, the connection information is displayed, e.g. “Connection: username@host.test.example”. At bottom right there is an icon indicating the connection status.

The connection dialog allows to specify the following settings for establishing a connection to an OpenVAS server:

Host The hostname or IP address of the server where an OpenVAS server is running.

Port The port where the OpenVAS Server waits for connections. Older versions of the OpenVAS server up to and including version 2.0.0 used port 1241 as the default port. The default port used for communication via the OpenVAS Transfer Protocol (OTP) by more recent versions is 9390. You can reset this option to the default port using the “default” button.

Login Your username on the selected OpenVAS server. To use an OpenVAS server you have to have an account on the OpenVAS server. Please contact the administrator of the server if you need an account.

Password The password for your account on the OpenVAS server.

Authentication by Certificate: If you use this method you have to have a key/certificate pair created for you. This is usually done by the administrator of OpenVAS server using the available scripts. The administrator will give you the two files you need to specify (User Certificate File and User Key File). The administrator may create a key without a password or with a password. If you have a password for the User Key File you must enter the password in the corresponding text field when connecting to the OpenVAS server.

Trusted CA: This certificate defines a certificate authority (CA) you trust. With this certificate you will be able to check that you are connecting to a trusted OpenVAS server. This is checked if you have the "Paranoia Level" set to 2 or 3 and is is not checked with a "Paranoia Level" of 1. Note that you can set the Paranoia Level by hand in the openvasrc files or when first connecting to an OpenVAS server where you are asked explicitly.

The default path for the Trusted CA is the filename used by the OpenVAS server itself. Thus, if you are running OpenVAS-Client on the same machine or have the same volume mounted, you can just use the default.

If you are running OpenVAS-Client from a remote machine, you need to have a copy of the CA certificate and set the location of the certificate file manually.

6.3 Scan Options

This section explains the most important configuration options for a security scan.

6.3.1 General

This page covers all the general scan options. See the screenshot for the main window in section 6.1.

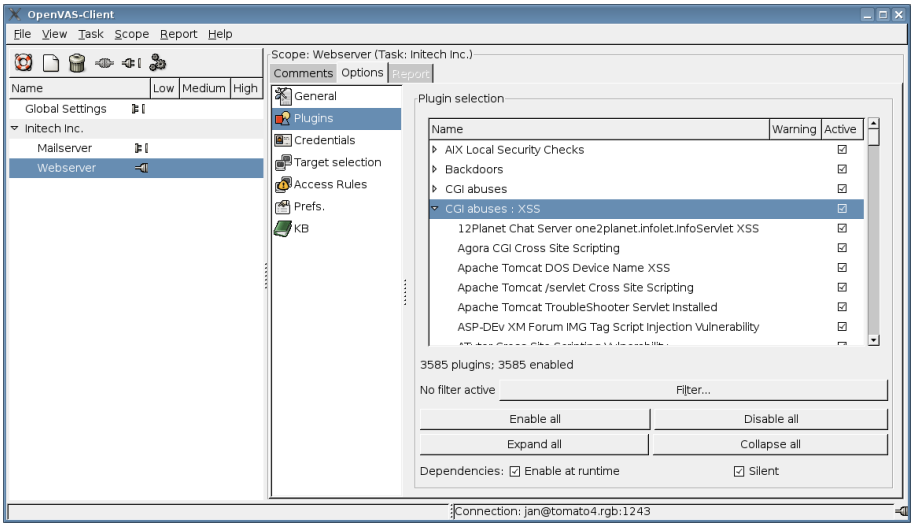
Port range Ports that will be scanned by the OpenVAS server. You can enter single ports, such as "1-8000" or more complex sets, such as "21,23,25,1024-2048,6000". Put "-1" for no portscan, or put "default" to scan the default ports in the OpenVAS services file.

Consider unscanned ports as closed To save scanning time, you may ask the OpenVAS server to declare TCP ports it did not scan as closed. This will result in an incomplete audit but it will reduce scanning time and prevent the OpenVAS server from sending packets to ports you did not specify. If this option is disabled, the OpenVAS server will consider ports whose state it does not know as open. Be aware that enabling this option might cause you to miss vulnerabilities in services that available on other ports than the ones you have scanned.

Number of hosts to test at the same time Maximal number of hosts that the OpenVAS server will test at the same time. Be aware that the OpenVAS server will spawn max_hosts x max_checks processes!

Number of checks to perform at the same time Maximal number of security checks that will be launched at the same time against each host. Be aware that the OpenVAS server will spawn max_hosts x max_checks processes!

Path to CGIs It is possible to check for the presence of CGIs in multiple paths like "/cgi-bin", "/cgis", "/home-cgis" and so on. In that case, put all your paths here separated by colons. For instance: "/cgi-bin:/cgi-aws:/cgi".



Screenshot of OpenVAS-Client 1.0 showing the plugin- selection view.

Do a reverse lookup of the IP before testing it If this option is set, the OpenVAS server will do a reverse lookup on the IP addresses before it tests them. This may slow down the whole test.

Optimize the test Security tests may ask the OpenVAS server to be launched if and only if some information gathered by other tests exists in the knowledge base, or if and only if a given port is open. This option speeds up the test, but may cause the OpenVAS server to miss some vulnerabilities. If you are paranoid, disable this option.

Safe checks Some security checks may harm the target server, by disabling the remote service temporarily or until a reboot. If you enable this option, the OpenVAS server will rely on banners instead of actually performing a security check. You will obtain a less reliable report, but you are less likely to disrupt functionality on the target system by doing a test. From a security point of view, we recommend you disable this option; from a system administrator point of view, we recommend you enable it.

Designate hosts by their MAC address If you enable this option, the hosts on the local network will be designated by their ethernet MAC address instead of their IP address. This is especially useful if you are using the OpenVAS server in a DHCP network. If unsure, disable this option.

Port Scanner This is the list of available port scanners. Port scanners are a special category of plugins and therefore presented separately from the other plugins. The list is only available if a connection to an OpenVAS server has been established. You can activate one or more of the scanners. Clicking on an entry shows the details for the respective scanner plugin.

6.3.2 Plugins

The plugins are stored on the OpenVAS server. Thus, to make a selection of the plugins to apply you need to connect to a server. Otherwise this page will remain empty.

The Plugins are separated into a number of families which can be activated or deactivated as a whole by checking the box to the right of family name. Also, a family can be expanded to show all of its member plugins where

you can refine the selection by activating or deactivating single plugins using the checkbox to the right of the plugin name.

The column “Warning” contains a warning sign for some plugins. The warning sign means that this plugin may harm the target host by disabling the attacked service or by crashing the host. You should be careful when you enable it since it may disrupt functionality on the target server.

Below the plugin list the total number of plugins loaded from the server is shown, together with the total number of currently selected plugin as well as the number of plugins shown due to an applied filter.

The following actions are possible:

Enable all Enables all plugins.

Disable all Disables all plugins.

Expand all Expands the plugin tree-list to maximum so that the list contains all plugins.

Collapse all Only show the plugin families.

Enable dependencies at runtime If you enable this option, the OpenVAS server will automatically enable the plugins needed by the set of plugins you selected.

Silent dependencies If you enable this option, the OpenVAS server will not report data coming from the plugins that you did not specifically enable.

Filter The filter dialog lets you select plugins with the characteristics you want. **Note** that you will erase your previous selection by applying a filter.

Automatically enable new plugins Since version 2.0 of OpenVAS-Client the user may choose whether new plugins should be enabled by default. Directly after fetching any new plugins, a notification will be displayed, showing how many new plugins were found and whether they have been en- or disabled. Earlier versions *do* automatically enable new plugins and do not show this notification.

Plugin information dialog Double-clicking on a specific plugin title will raise an information dialog for the respective plugin.

The values shown are the ones specified within the corresponding plugin, like its description, copyright information.

The following actions are possible in this dialog:

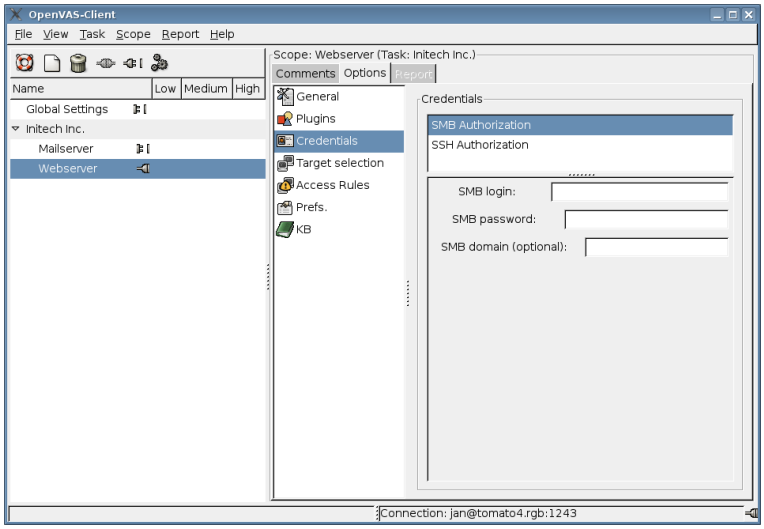
Set plugin timeout Allows you to specify a timeout for the plugin.

Show dependencies This lists the dependencies for the selected plugin. It also provides information on whether the dependencies are currently enabled or disabled.

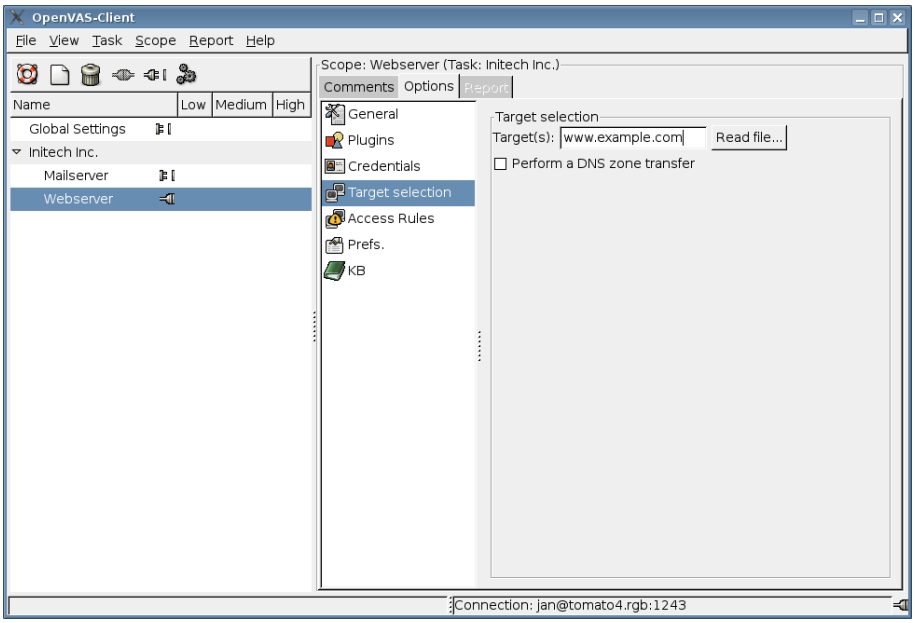
Certificate information Since OpenVAS-Client 2.0.0beta2 (which needs a server version 2.0.0beta2 or newer) the servers certificates can be viewed from within the client. If the plugin that is currently displayed in the plugin information dialog has one or more signatures, the names of and trust relation the signer(s) will be displayed. Next to name and trust level is a button that allows to see the signers certificate.

6.3.3 Credentials

Some of the plugins allow to enter credentials to test certain applications, for example Samba or websites (HTTP). These plugins work the same way as the plugins listed in the “Plugin Preferences” list. For better handling they are collected under “Credentials”.



6.3.4 Target Selection



Target(s) The first host(s) that will be attacked by the OpenVAS server. The options below allow you to extend the test to a larger set of targets. You may define several targets by separating them with a comma (,). i.e. : “host1,host2”.

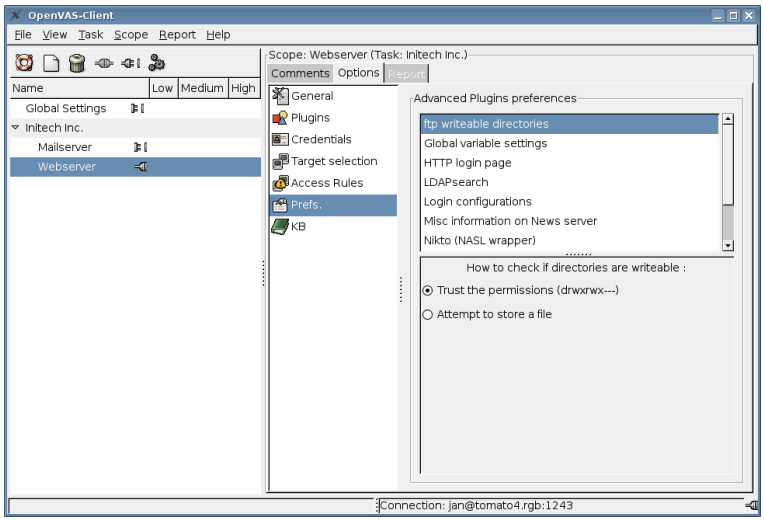
A special syntax is “file:/some/where/targetlist.txt” which means that the actual target names are loaded from that list.

Read from file A textfile can be specified that contains the list of targets. This textfile may contain comma-separated lists of host and also may contain many of such lines.

Perform a DNS Zone transfer The OpenVAS server will perform an AXFR request (that is, a zone transfer) to the target name server and will attempt to obtain the list of the hosts in the target domain. Then, it will test each host.

6.3.5 Plugin Preferences

Some of the plugins allow to refine with specific parameters. All of the configurable plugins’ parameters are collected on this page where the user may modify the default values.



Only a comparably small number of plugins offer a configuration.

6.3.6 Access Rules

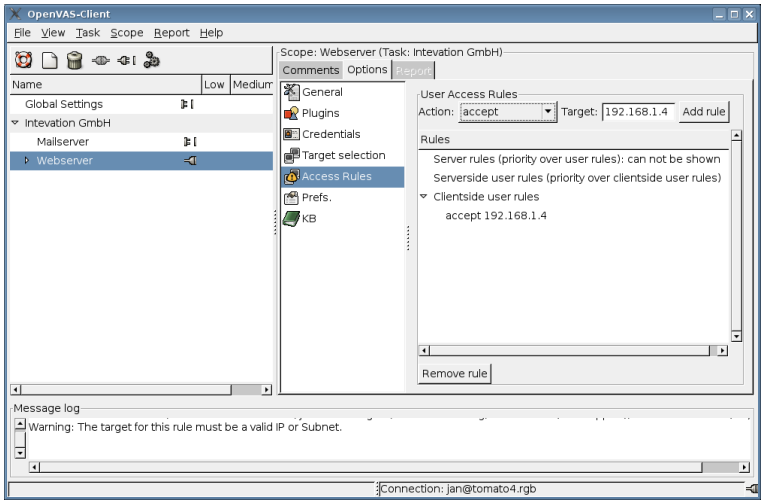
In this section you can view and manage the access rules for your scan. These rules determine which host you may scan. Note that there are three kinds of access rules:

Server rules These rules are global to the server and will affect all users that connect to this server.

Serverside user rules These rules are specific to a user and affect only this user, no matter from which client he connects to this server.

Clientside user rules These rules are specific to the client. They will affect only the scope in which they are defined.

The first two rulesets are sent by the server only to inform the client about possible restrictions and cannot be changed by the client. Only the last ruleset can be changed by the client.

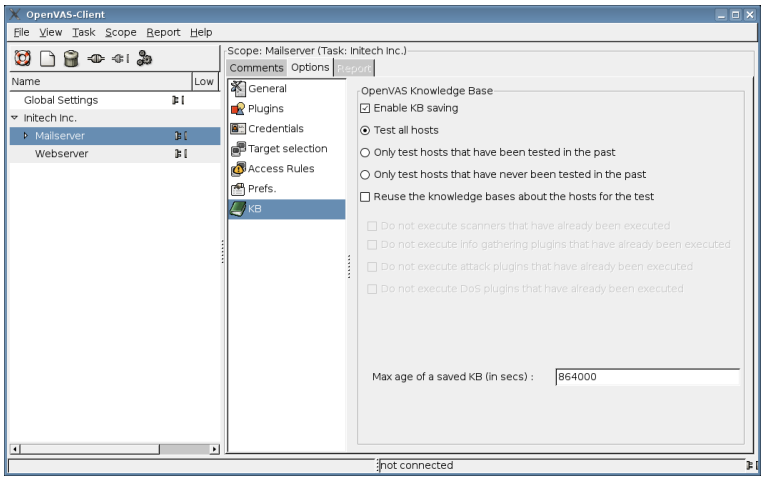


A rule consists of an action and a target. To add a rule to the clientside user rules, select the appropriate action from the drop-down menu, enter the the IP of the target or a netmask and press the “Add rule” button. You can remove clientside user rules by selecting the rule and pressing the “Remove rule” button.

For more information about rules and the ruleset syntax, please refer to section 3.3.2 which explains the definition of serverside rules.

6.3.7 Knowledge Base

The configuration section for the Knowledge Base (KB) allows you to control the management of the serverside scan results. Information retrieved by plugins is collected in a KB during a scan. This is done on a per-host basis, meaning there is one KB for every host scanned. The default is to discard the KB once all plugins have finished, but under certain circumstances it can be quite useful to tell the server to keep the KBs generated during the scan and use them again at a later time.



The following options are available to control KB handling:

Enable KB saving If you want the server to save the KB after the scan is done, you have to enable this option.

Test all hosts If this option is set, the server will not use the KB to determine which hosts should be scanned, but will rather scan all hosts supplied.

Only test hosts that have been tested in the past If KB saving is enabled, there is one KB saved on the server for every host the server has scanned in the past. This can be used to restrict the server to scan only hosts that have been scanned before. This might be useful if you want to keep an eye on a certain set of machines and their configuration. Be aware that this setting might cause you to miss new hosts on the network since the server will not scan them.

Only test hosts that have never been tested in the past Another way of using the existence of KBs is to exclude all hosts that have already been scanned. This way a scan will automatically discover hosts that have been added to the network since the last scan. Be aware that this setting cause hosts to be scanned only once (the first time they appear on the network), meaning you will not discover security issues that have recently developed or are only detected by new NVTs.

Reuse the knowledge bases about the hosts for the tests This setting controls if the server should restore the KB that was saved for this host during the last scan. The default behavior is to create a new KB every time a host is scanned and to replace an existing KB with the new results.

Do not execute scanners that have already been executed If the server has been instructed to reuse the existing KB, this will prevent scanning plugins from running if their results have already been recorded in the KB.

Do not execute info gathering plugins that have already been executed If the server has been instructed to reuse the existing KB, this will prevent information gathering plugins from running if their results have already been recorded in the KB.

Do not execute attack plugins that have already been executed If the server has been instructed to reuse the existing KB, this will prevent attack plugins from running if their results have already been recorded in the KB.

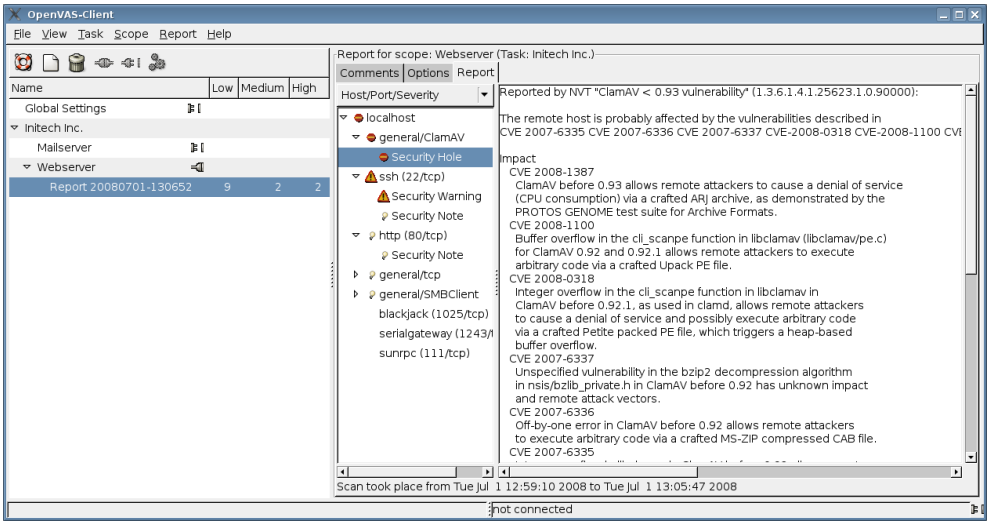
Do not execute DoS plugins that have already been executed If the server has been instructed to reuse the existing KB, this will prevent denial-of-service (DoS) plugins from running if their results have already been recorded in the KB.

Max age of a saved KB This setting controls the maximum age of the KB (in seconds). A KB older than this value is automatically discarded.

6.4 Reports

6.4.1 Report Page of OpenVAS-Client

The report page consists of three elements. On the left hand a tree list allows you to browser via hosts, ports and severity to single reports. On top of this treelist is a selection for re-ordering the tree structure. On the right hand the text area contains the actual report text. The whole design is focused on supporting an explorative understanding of the scan results.



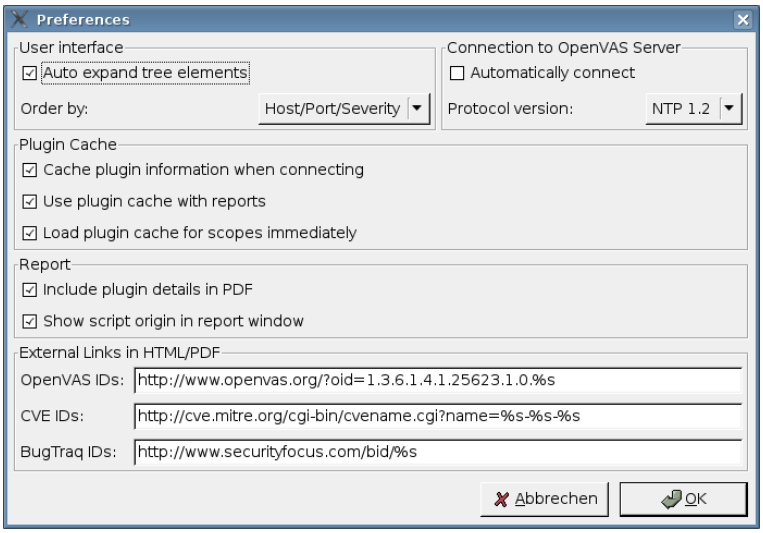
6.4.2 Report Formats

The scan results can be exported into a number of formats. Basically it can be distinguished between three types of formats: interchange formats, editable documents and read-only documents. The last type is currently represented by the PDF Report file. With a capable viewer it allows you to browse back and forth through the document using the various inserted hyperlinks.

For further information see the section 6.1.3 about the menu command “Report->Export”.

6.5 OpenVAS-Client Preferences

OpenVAS-Client allows you to specify some individual preferences that determine some ways how the client GUI works.



The following selection are available:

6.5.1 User Interface

Auto expand tree elements In the left-hand treeview clicking on a task or a scope automatically expands the corresponding tree if checked.

If not checked, the user has to manually click on the expand icon each time.

Order by This setting controls how the scan results are sorted in the report window. The default is to sort the results by host first, then by port and severity. Depending on your context, it might make more sense to choose the second option and sort the results by port first; for example, you might be more interested in a quick overview over which hosts are running a service on a certain port than in which ports are open on a certain host.

6.5.2 Connection to the OpenVAS server

Automatically connect If this setting is enabled, OpenVAS-Client will try to connect to the server when a scope is executed. For user certificates without a password, this will work immediately. For password protected user certificates or simple password based authentication, the password will be stored in memory after a successful login until OpenVAS-Client is closed.

Protocol version This setting controls which protocol the client will ask the server to use for communication between client and server. Please be aware that the server will close the connection if the client asks for an unsupported protocol.

6.5.3 Plugin Cache

Cache plugin information when connecting If this setting is enabled, OpenVAS-Client will create a cache for the respective scope containing all plugin information. This has essentially three effects:

First, reconnecting the same scope might be much faster because MD5 checksums are used to discover changed and new plugins. Only the changes will be downloaded. Of course, connecting to a different OpenVAS server will usually force a new download of all plugins.

Second, all plugin information is available even when the client has no connection to the server. Thus you can review which plugins are selected and what the current plugin preferences are. Note that the selection might change after connecting for a actual scan because new plugins might become available and others might disappear. Loading the cache may take a couple of seconds. If you don't want this, switch off the option "Load plugin cache for scopes immediately".

Third, the downside of caching: The cache will consume several megabytes for each scope. If you do not have sufficient storage space available, you should disable this feature. If you want to remove the caches, search for the files "nessus_plugin_cache" in your OpenVAS directory (the directory ".openvas" in your home directory). Simply deleting them is sufficient.

Use plugin cache with reports Enabling this setting will make OpenVAS-Client attach all Plugin Information to newly created scan reports. This allows to review the plugin selection and the plugin preferences for a report in the OpenVAS-Client GUI. So, this cache is for increasing transparency not performance.

Again, the downside is that several megabytes of cache per report will be generated. Disable this option if you do not have sufficient storage space available. If you want to remove the caches, search for the files "nessus_plugin_cache" in your Nessus directory (the directory ".openvas" in your home directory). Simply deleting them is sufficient.

Load plugin cache for scopes immediately Disabling this option will cause OpenVAS-Client to not automatically load a scope's cache when made active. You won't see the plugin selection nor the plugin preferences. So, in fact this option could remove the second effect of the above described option "Cache plugin information when connecting" for the benefit of avoiding to load possibly huge caches once clicking on a scope entry.

6.5.4 Report

Include plugin details in PDF Enabling this setting will make OpenVAS-Client add an appendix to each PDF Report containing the details of those plugins that produced relevant results for the report. They are linked within the PDF so that the information can easily be browsed.

Be aware, however, that this could significantly increase the size of your PDF file. On average, you can expect two plugin descriptions to consume one page.

Show script origin in report window Enabling this option will cause OpenVAS-Client to show additional information in the report window regarding the origin of the reported security issue. If this option is enabled, these reports will contain the name and the OID of the NVT that reported the issue.

6.5.5 External Links in HTML/PDF

These settings determine the URLs for linking more information on OpenVAS NVTs, CVE/CAN and BugTraq IDs in HTML or PDF reports. The defaults as shown in the screenshots are recommended since these are the definitive sources for up-to-date information. The defaults are restored when the fields are left empty.

In case you want to package an OpenVAS report with e.g. CVE/CAN details for offline-reading, you may enter an appropriate definition like "mitre/%s/%s/%s.html" in case you have a directory structure relative to the report file with mitre/CVE/yyyy/nnnn.html and mitre/CAN/yyyy/nnnn.html where yyyy is the year and nnnn is the number of the record. Then you could package all files together.

Note, that the strings defined here are inserted into the html link parameter "href" as they are. The tool "html-doc" is used to produce a PDF out of this html report. Depending on the version and features the created links in the PDF file may be created differently.

6.5.6 Installing SLAD using SLADinstaller

If you are planning to use the Security Local Auditing Daemon (SLAD) with OpenVAS (as described in section 8.1.1), there is a convenient way to install SLAD on a local or remote machine. Using the "SLAD install" option in the file menu, you can launch the `sladinstaller` binary from within OpenVAS-Client, which will the guide you through the installation process.

Please note that the `sladinstaller` binary needs to be available in your system path for you to be able to use this feature. More information on installing and using `sladinstaller` is available in section 8.1.1.

7 Performing Local Security Checks

(by Jan-Oliver Wagner)

7.1 Debian Local Security Checks

This section explains how to run local security checks with OpenVAS. So far, this procedure has been tested only with Debian local security checks.

7.1.1 Prerequisites

To perform local security checks, you need a working OpenVAS-Server installation. Information on setting up and configuring OpenVAS-Server is available in chapter 3 on page 13.

7.1.2 Create users for local security checks

First, you need a key with certificate:

```
$ ssh-keygen -t rsa -f ~/.ssh/id_rsa_sshovas -C
"OpenVAS-Local-Security-Checks-Key"
$ openssl pkcs8 -topk8 -v2 des3 -in ~/.ssh/id_rsa_sshovas -out sshovas_rsa.p8
```

Note: The comment (here: “OpenVAS-Local-Security-Checks-Key”) must not contain spaces. Currently, you need a rsa pkcs8 key for OpenVAS local security checks.

Now, for each target system:

```
# adduser --disabled-password sshovas
Name: OpenVAS Local Security Checks
# su - sshovas
$ mkdir .ssh
$ cp /some/path/id_rsa_sshovas.pub .ssh/authorized_keys
$ chmod 500 .ssh
$ chmod 400 .ssh/authorized_keys
```

7.1.3 Configure the local security checks in OpenVAS-Client

In Preferences, configure SSH Authorization with the key and the certificate you have just created:

```
SSH login name: sshovas
SSH private key: ~/.ssh/sshovas_rsa.p8
SSH key passphrase: ********
SSH public key: ssh/id_rsa_sshovas.pub
```

Note: It is actually not necessary to submit the public key, but currently this is necessary due to a bug inherited from Nessus.

Next, make sure you select at least these plugins:

- Debian Local Security Checks/*
- Misc/Determine List of installed packages via SSH login
- Service Detection/Services
- Settings/Global variable settings
- Settings/SSH Authorization

or ensure dependencies are resolved at runtime (see checkboxes) if you select only some local security checks.

7.2 Windows Local Security Checks

In order to provide - analogous to the Linux Local Security Check - a framework for checking and testing for Microsoft Windows vulnerabilities, a new Application Programming Interface (API) was introduced by the OpenVAS team.

This API (for the Nessus Attack Scripting Language, NASL) allows to check and examine binary files (e.g. DLLs) as well Microsoft Windows meta-data on Microsoft Windows machines by accessing the complete file system using default Windows administrative shares.

The functionality is similar to the Nessus Windows Local Security checks. However, the OpenVAS solution is using samba (smbclient) and does not re-implement the binary SMB protocol in NASL.

The advantage of this smbclient integration is to act more flexibly on protocol changes on the SAMBA/CEFIS protocol side.

7.2.1 Preparing the OpenVAS Server

To install the WLSC, a few steps are required to be taken on the server that hosts and runs the actual scan daemon “openvasd”.

It depends on the operating system where you run the OpenVAS server, how the steps are carried out technically.

1. Install SAMBA (http://www.samba.org)
   Installation packages should be readily available for your operating system.
2. Make sure that the binary “smbclient” in is the installation/execution path.
3. Check the permissions of openvasd and smbclient: smbclient must be executable with the permissions of openvasd.

7.2.2 Preparing the Microsoft Windows target

The WLSC implementation has been tested on the following Microsoft Windows Operating Systems:

- Windows NT 4.0
- Windows 2000

- Windows XP SP2
- Windows XP SP3
- Windows Vista

Probably the WLSC is also compatible with other Microsoft Operating Systems. Once the SMB Port of a Windows target is accessible, the Operating System (OS) and SAMBA Version could be detected immediately and will be reported. For deeper tests the following steps are required:

You need the Windows credentials for an administrative user. Usually this is the user name (Default is "Administrator") and the correct password for this user. There is no default password, this has been defined before during the Windows installation process.

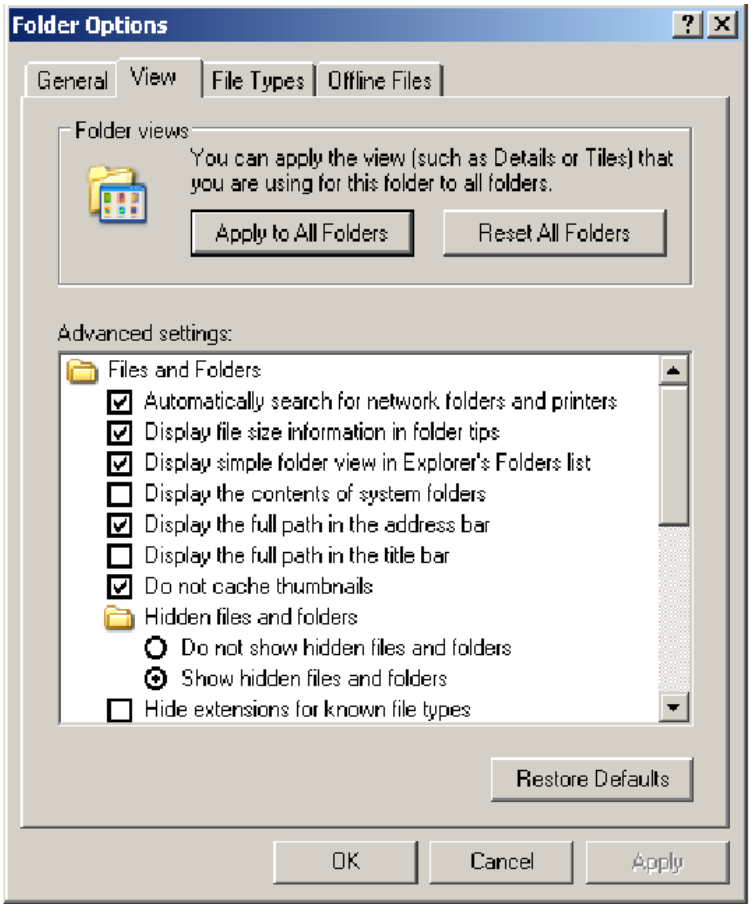
These credentials are entered in the OpenVAS-Client GUI as SMB Credentials and are used on every host in the target list.

If you plan to scan a whole Windows Domain, you can enter the Domain-Administrative user and password instead of the target host credentials.

Make sure the Windows-(personal) Firewall is disabled for the OpenVAS Server host, or a correct rule for the Test-Network is entered.

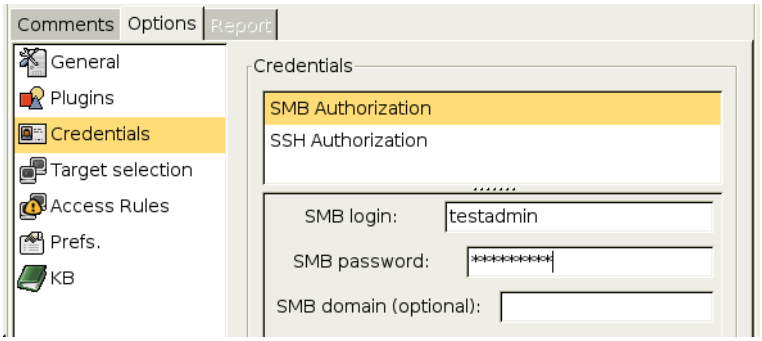
Additional Note for Windows XP For Windows XP it is important that "Easy Filesharing" is switched off. To disable this, the Windows Click-Path is: Windows Explorer/Tools/Folder Options/View (see screenshot below).

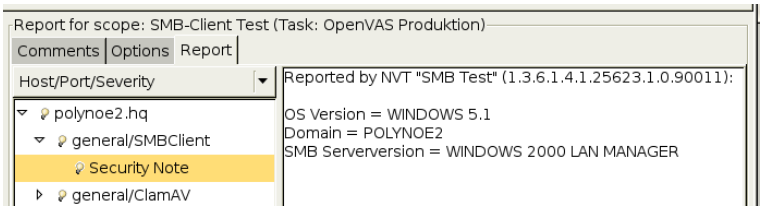
Without this setting, "smbclient" is not able to retrieve files from the Windows System shares (`C$, D$...`).



7.2.3 Executing the checks via OpenVAS-Client

Using the OpenVAS-Client you specify the credentials and the target host information will be reported as illustrated by the following two screenshots:





8 Using Integrated Tools

8.1 Security Local Auditing Daemon (SLAD)

(by Jan-Oliver Wagner)

8.1.1 How to use Security Local Auditing Daemon (SLAD) with OpenVAS

If you want to install SLAD on remote machines, you may want to use the `sladinstaller` tool which will automatically retrieve the latest SLAD release and install it on the target machine.

The `sladinstaller` integration into OpenVAS is currently under development; up-to-date information on installing `sladinstaller` is available on the OpenVAS website.

8.1.2 SLAD plugins

The sladd is just a program to run other programs from inside a daemon and provide an unified interface to their output. The current package of SLAD contains the following plugins:

chkrootkit

The chkrootkit package is a tool to locally check for signs of installed rootkits.

clamav

The clamav plugin provides a GPLed virus scanner for linux. The options include scanning with or without archive (.zip, .tar.gz, etc) scanning, and removing infected files or putting them into quarantine.

john

John the ripper is a fast password cracker. This tool is meant to find weak user passwords, which could compromise system security. It comes with three options:

Fast crack mode: In this mode, John only tries the usernames and derived words against the hashed passwords.

Dictionary mode: In this mode all words from the installed dictionary are tried to attack the hashed passwords.

Full crack mode: This slowest mode tries all words from the dictionary, as well as rules generated variations of these, against the user's passwords

The "normal" version of John exposes cracked passwords in clear-text. This makes John difficult to operate with in a professional environment. Therefore, SLAD uses a John the Ripper version which has been patched. In this version cracked passwords are not exposed anymore, instead only the user-accounts with crackable passwords are identified.

lsof

The Unix system utility lsof simply shows a list of files currently open on the system and which programs use them. This can assist an administrator to find unusual activity on the system.

tiger

The tiger suite is a package to analyze the host's security. Out of the many checks the suite can perform four groups have been created. These are:

Users: The users check covers accounts, checks for mail aliases, FTP login users and the like.

Permission: This selection checks users and group access permissions on device nodes, logfiles and other important files and directories.

Config: This script checks for weaknesses and mistakes in common system and application specific configuration files.

System: The system check looks for open deleted files, processes that are waiting for incoming connections, and other "unusual" things.

Full system check: This runs all of the above checks.

tripwire

Tripwire is an open source file integrity checker. It initially stores hashes of system files in a database for comparison on subsequent runs. Modifications performed by a potential intruder can be easily spotted this way.

The default installation of tripwire contains a rule set for Debian. For details on how to adapt these to a different operating system or distribution, see the SLAD 2 Developers and Administrators Guide.

Snort

Snort is a network intrusion detection and prevention system that provides real time traffic analysis and packet logging on IP networks. It is capable of detecting a large number of attacks such as buffer overflows, stealth port scans, CGI attacks, SMB probes or OS fingerprinting attempts by doing both protocol analysis and content checks. Once an attack has been detected Snort is also capable of counteracting it by dropping the according connections. The SLAD plugin selects all relevant Snort messages from a MySQL Database and sends them to the management platform.

To use the SLAD Snort plugin, Snort needs to be installed with MySQL support. Information on how to do this is usually provided with the Snort package for your distributions or can be found in the Snort installation itself.

LMSensors

This fetches the events from your hardware monitoring (for example someone opening the chassis of the server). The management system supports hardware sensor logging. An alert will be shown describing the physical incident on the SLAD managed server. This features is supported from the most mid-range server boards like Intel BX440 and newer.

LogWatch

Logwatch extracts events from the system log, like the syslog files present at "/var/log". All important information like login users, SSH and PAM Sessions, etc. are filtered and aggregated and returned to the calling SLAD. Three different levels of detail are supported:

Low Returns logfile values in a low detail level and with the highest aggregation.

Medium Returns logfile values in a medium detail level.

High Detailed logfile values with the lowest aggregation level.

TrapWatch

TrapWatch is a special version of Logwatch and listens on SNMP hardware traps. The Simple Network Management Protocol (SNMP) is the most common protocol for managing all kinds of network devices and is implemented in almost all currently available network devices. An SNMP trap is a message sent out by a network device to report an incident such as loss of link, failed authentication attempts etc. TrapWatch catches these messages and puts them into the report. This can be useful to detect changes in the network, like machines being unplugged or added to the network. Support for Netscreen firewall traps, HP-Procure switches and Cisco hardware is installed out of the box. If non-standard MIBs are used, it might necessary to configure TrapWatch accordingly. Please note that to enable TrapWatch, you need to install an SNMP trap handler that puts the TRAP results into your syslog file.

8.2 Nikto

(by Michael Wiegand)

Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 3500 potentially dangerous files/CGIs, versions on over 900 servers, and version specific problems on over 250 servers. Scan items and plugins are frequently updated and can be automatically updated (if desired).

OpenVAS is able to recognize an installed version of Nikto and can integrate the results of a Nikto scan in the scan results.

8.2.1 Prerequisites

In order to be able to perform a Nikto scan from within OpenVAS, the following requirements must be met:

- There has to be a version of Nikto that can be found in the system path. The OpenVAS integration of Nikto is optimized for Nikto versions >= 2.0, but will probably work with older versions as well.
- The OpenVAS plugin for Nikto integration (*nikto.nasl*) needs to be present and enabled. You can find the plugin in the section *CGI abuses* in the plugin section of your client.

8.2.2 Starting a Nikto scan

If the Nikto plugin is present and enabled, it will be executed with your next scan. The results returned by Nikto will be available together with the rest of the scan results.

8.2.3 Understanding Nikto results

Some web servers are (intentionally or unintentionally) configured to respond to requests for non-existent files with an HTTP status code other than 404. This can be used to direct these requests from human users to a page with helpful information (like a sitemap), but tends to confuse security assessment tools like Nikto checking whether possibly sensitive or dangerous content can be accessed on the target server.

The Nikto plugin is able to recognize this condition in most web servers and will (in the default setting) refuse to launch Nikto under these circumstances. You can however force the Nikto plugin to launch Nikto by enabling the option *Force scan even without 404s* in the plugin preferences.

If you enable this option, please be aware that the results of the Nikto scan are likely to contain false positives; because of the web server configuration described above Nikto may be convinced that certain files exist on the web server, even though the server simply redirected these requests to a generic page.

This is especially true for older versions of Nikto (< 2.0); but even with newer versions you may need to manually evaluate whether the threats reported by Nikto are real threats or simply the result of the web server configuration.

8.3 Ovaldi (OVAL support in OpenVAS)

(by Michael Wiegand)

The Open Vulnerability and Assessment Language (OVAL) is a standard that can be used – among other things – to describe known vulnerabilities and tests that can be used to assess whether a vulnerability is present on a target system. It uses XML documents e.g. to describe components of a potentially vulnerable system and to describe the state in which their components are. Other XML documents – the so called vulnerability definitions – describe particular states in which these components should be considered vulnerable. In contrast to NASL, OVAL definitions only formally describe how a vulnerable system is expected to look and are not in themselves programs describing a way to actively look for those vulnerabilities.

The OVAL community has created ovaldi, an open source reference implementation of an OVAL definition interpreter. Although ovaldi initially only supported checks of a local system, the OpenVAS project has created a patch that enables ovaldi to make use of the information collected by OpenVAS about remote systems.

Starting from OpenVAS 2.0.0, ovaldi support is present in OpenVAS. To enable ovaldi support, the use of ovaldi in the SVN revision 138 is recommended. Please refer to the OpenVAS website for the patch needed for ovaldi and up-to-date information regarding ovaldi integration. The latest information is available at http://www.openvas.org/integrated-tools.html.

Using ovaldi, you will be able to access hundreds of additional security checks provided as definitions in the OVAL standard such as security announcements regarding the Red Hat Enterprise Linux distribution. Be aware that the ovaldi integration into OpenVAS only supports a limited subset of the tests available in OVAL. Support for OVAL tests will be extended as the ovaldi integration matures.

Once you have successfully enabled support for OVAL plugins, the plugins will show up in the OpenVAS-Client in the “OVAL definitions” family. Most plugins will return one of the following values: “true”, “false” or “unknown”. These values are defined as follows:

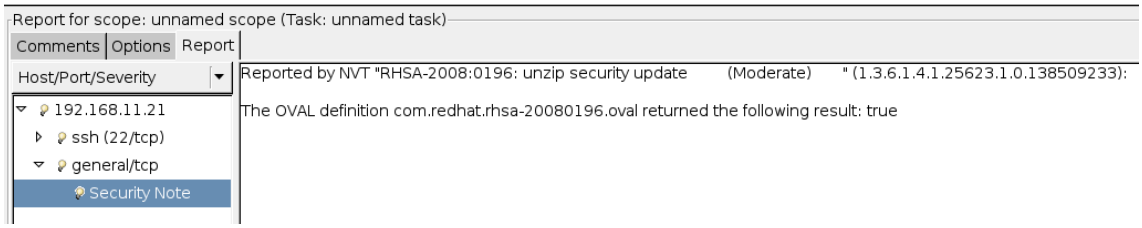
true The tests evaluated by ovaldi indicated that the vulnerability described in the definition is very likely to be present on the system.

false The tests evaluated by ovaldi indicated that the vulnerability described in the definition is not likely to be present on the system.

unknown The tests evaluated by ovaldi were inconclusive or ovaldi was unable to execute all tests required for this evaluation.

Note that a large number of tests will return “unknown” until extended OVAL support in OpenVAS has been established.

The results of the OVAL definitions will be shown in the same way as the results for other plugins, allowing you to assess the results conveniently from within OpenVAS-Client.



You can find more information about the OVAL project and the OVAL language at http://oval.mitre.org/. The project page for ovaldi can be found at http://sourceforge.net/projects/ovaldi/.

9 Developers Guide for Network Vulnerability Tests

The Network Vulnerability Tests (NVTs) used by OpenVAS to check for existing security issues on remote systems are written in the scripting language NASL. NASL (short for Nessus Attack Scripting Language) was originally designed for the Nessus security scanner by Renaud Deraison.

The motivation behind NASL was to create a language enabling even users with limited programming experience to write vulnerability tests in a short amount of time and to allow users to easily add new vulnerability tests to their existing installation without having to worry about compatibility issues.

The first version of NASL (also known as NASL1) was created in 1998 by Renaud Deraison. In 2002, Michel Arboi and Renaud Deraison developed an improved NASL parser which extended the range of built-in functions and operators. This improved version is generally referred to as NASL2.

If not indicated otherwise, this compendium describes NASL2 as it is implemented in OpenVAS.

The NASL syntax was inspired by C. Users with experience in C or related programming languages should be able to pick up the basics of NASL development in a relatively short amount of time.

Starting from OpenVAS 2.0.0, support has been added to OpenVAS for the Open Vulnerability and Assessment Language (OVAL) as documented in section 8.3. This means that OpenVAS will also understand vulnerability tests specified in OVAL. Even though support for OVAL in OpenVAS is very limited at this time, you might want to consider OVAL as an alternative when writing NVTs. More information about OVAL can be found at http://oval.mitre.org/language/about/index.html.

9.1 Basic Structure of NASL Scripts

All NASL scripts have to contain a set of information about themselves by which they can be clearly distinguished from other plugins and referenced by other tools like the client. This information is contained in the `description` or `register` section which is mandatory for all NASL scripts and is usually right at the beginning of any NASL script.

A basic NASL script might start like this:

```
#
# This is an example NASL script.
#

if(description)
{
 script_oid("1.3.6.1.4.1.25623.1.0.12345")
 script_version ("1.2");
 name["english"] = "Foo Bar 2.5 vulnerability";
 script_name(english:name["english"]);

 desc["english"] = "
```

```
   This plugin checks for the vulnerability in the Foo Bar 2.5 server
   component as described in CVE 2009-4321.

   Risk factor : None";

 script_description(english:desc["english"]);

 summary["english"] = "Check for vulnerability in Foo Bar 2.5";
 script_summary(english:summary["english"]);

 script_copyright(english:"This script is under GPLv2+");

 ...

 exit(0);
}

...
```

The plugin description has to be contained in the `if (description)` block so the OpenVAS server can retrieve it. The first time the server encounters a new plugin, it will be called with the global variable `description` set to `TRUE`. The information provided by the plugin will be cached in the `.desc` subdirectory in the plugins directory. When the script is called during a scan, it will be called with `description` set to `FALSE`.

For a complete list of NASL commands that can be used in the script description, please refer to the section 9.3 of the NASL API documentation.

9.2 Basic NASL Syntax

9.2.1 Comments

Comments in NASL start with the # character. If a # character is encountered, the remaining line is ignored by the NASL interpreter.

Example:

```
# This is a comment.
a = b + c; # This is a comment as well.
```

9.2.2 Variables and Declarations

Variables are implicitly declared in NASL; this means that it is not necessary to declare any variables before use. You can however use the `local_var` keyword to declare a variable as local to a certain function and avoid collisions with external variables. By default, NASL considers a variable to be local to the context in which it was declared; if you want to declare a variable as global, you can use the `global_var` keyword.

In contrast to C (and other languages), variable types do not have to be declared in advance; the NASL interpreter will use the appropriate type the first time the variable is assigned. Memory allocation is automatically handled by the interpreter as well.

9.2.3 Data Types

- Integers
- Strings
- Arrays
- Booleans

9.2.4 Numbers and Strings

9.2.5 Function Arguments

9.2.6 Loops

- for
- foreach
- while
- repeat .. until
- break
- continue
- return

9.2.7 User-defined Functions

9.2.8 Operators

- =
- `[]`
- +
- -
- *
- /
- %
- **
- ++
- +=
- -=
- *=
- /=
- %=
- «=

- »=
- »>=
- ><
- >!<
- =
- !
- ==
- !=
- >
- >=
- <
- <=
- !
- &&
- ||
- ~
- &
- |
- ^
- <<
- >>
- >>>

Operator Precedence

9.3 NASL API Documentation

9.3.1 Pre-defined Constants

9.3.2 Built-In Functions

Socket Manipulation

- `open_sock_tcp()`
- `open_sock_udp()`
- `open_priv_sock_tcp()`
- `open_priv_sock_udp()`
- `close()`
- `shutdown()`
- `recv()`
- `recv_line()`
- `send()`

Network Operations

- start_denial()
- end_denial()
- get_port_transport()
- get_source_port()
- get_tcp_port_state()
- get_udp_port_state()
- islocalhost()
- islocalnet()
- join_multicast_group()
- leave_multicast_group()
- scanner_add_port()
- scanner_get_port()
- this_host_name()

FTP Operations

- ftp_log_in()
- ftp_get_pasv_port()

HTTP Operations

- is_cgi_installed()
- http_get()
- http_head()
- http_post()
- cgibin()
- http_delete()
- http_close_socket()
- http_open_socket()
- http_recv_headers()
- http_put()

Packet Manipulation

- forge_ip_packet()
- get_ip_element()
- set_ip_elements()
- dump_ip_packet()
- forge_tcp_packet()
- set_tcp_elements()
- send_packet()
- pcap_next()
- get_tcp_element()
- forge_udp_packet()
- set_udp_elements()
- get_udp_elements()
- forge_icmp_packet()
- get_icmp_element()
- set_icmp_elements()
- forge_igmp_packet()
- dump_tcp_packet()
- dump_udp_packet()

Utilities

- this_host()
- get_host_name()
- get_host_ip()
- get_host_open_port()
- get_port_state()
- telnet_init()
- tcp_ping()
- getrpcport()

String Manipulation

- ereg()
- ereg_replace()
- egrep()
- crap()
- string()
- strlen()
- raw_string()
- tolower()
- chomp()
- display()
- eregmatch()
- hex()
- hexstr()
- insstr()
- int()
- match()
- ord()
- str_replace()
- strcat()
- stridx()
- strstr()
- split()
- substr()
- toupper()

Knowledge Base

- get_kb_item()
- set_kb_item()
- get_kb_list()
- replace_kb_item()
- replace_or_set_kb_item()

Plugin Description

- `script_id()`
- `script_oid()`

  (by Tim Brown)

  This function is intended to replace `script_id`, the current method of uniquely identifying NASL scripts. The logic behind this is that `script_id` has only a single global namespace. With plans by several organisations to develop and contribute plugin feeds it was deemed necessary to introduce a new namespace that could be shared between each organisation.

  The current proposed implementation of this function is as follows. Any plugin that contains a `script_id` call will automatically be given an OID from the namespace allocated for legacy plugins. Moreover `script_oid` can only be used on plugins that do not have a `script_id` set. The OID namespace for legacy plugins has the prefix "1.3.6.1.4.1.25623.1.0". The OpenVAS OID namespace is currently administered by Tim Brown.

  Both the client and server as well as the libraries on which they depend are being updated to support this functionality. You can detect if it is supported by checking OPENVAS_NASL_LEVEL is greater than 2206.

  `script_oid` should be called like so:

```
...
if(description)
{
  if (OPENVAS_NASL_LEVEL >= 2206)
  {
    script_oid("1.3.6.1.4.1.25623.1.0.90010");
  }
  else
  {
    script_id(90010);
  }
...
```

- `script_version()`
- `script_name()`
- `script_description()`
- `script_summary()`
- `script_category()`
- `script_copyright()`
- `script_family()`
- `script_dependencies()`
- `script_cve_id()`
- `script_require_ports()`
- `script_require_keys()`
- `script_exclude_keys()`
- `scanner_status()`

- `script_get_preference()`
- `script_add_preference()`
- `script_bugtraq_id()`
- `script_dependencie()`
- `script_get_preference_file_content()`
- `script_get_preference_file_location()`
- `script_require_udp_ports()`
- `script_timeout()`

Report Functions

- `security_warning()`
- `security_hole()`
- `security_info()`
- `security_note()`
- `log_message()`
- `debug_message()`

Crypto Functions

- `HMAC_DSS()`
- `HMAC_MD2()`
- `HMAC_MD4()`
- `HMAC_MD5()`
- `HMAC_RIPEMD160()`
- `HMAC_SHA()`
- `HMAC_SHA1()`
- `MD2()`
- `MD4()`
- `MD5()`
- `RIPEMD160()`
- `SHA()`
- `SHA1()`

Miscellaneous Functions

- `cvsdate2unixtime()`
- `defined_func()`
- `dump_ctxt()`
- `func_has_arg()`
- `func_named_args()`
- `func_unnamed_args()`
- `gettimeofday()`
- `isnull()`
- `localtime()`
- `max_index()`
- `mktime()`
- `safe_checks()`
- `sleep()`
- `type_of()`
- `usleep()`
- `unixtime()`
- `make_list()`
- `make_array()`
- `get_preference()`

"Unsafe" Functions

NB: Under Nessus, these functions were considered "unsafe" since they access the file system and were only available to a special selection of "trusted" plugins. Since OpenVAS takes a different approach and will not permit untrusted plugins to run alongside trusted plugins, these functions are available to all plugins. Classifying these functions as "unsafe" is done purely for legacy reasons. Nevertheless, developers should be aware that these functions could potentially harm the system which is running OpenVAS and are expected to take extra care when using these functions.

- `find_in_path()`
- `file_close()`
- `file_open()`
- `file_read()`
- `file_seek()`
- `file_stat()`
- `file_write()`
- `fread()`
- `fwrite()`
- `get_tmp_dir()`
- `unlink()`
- `pread()`

9.3.3 Functions from the NASL Library

Apart from the built-in functions provided by NASL itself it is also possible to use functions defined in "include" files. As a convention, these files have the extension ".inc" and reside in the plugin directory. Before you start writing functions for your own NASL scripts, you might want to check if the functionality is already available in the NASL Library. If you find yourself implementing a function that could be useful in other plugins as well, please consider adding it to the library.

Quite a number of functions have been added to the NASL library already; the following list contains the names and declared functions of the files currently in the library. Keep in mind that this information is subject to change as plugins are added or updated.

backport.inc get_backport_banner(), get_php_version()

debian_package.inc deb_check(), deb_str_cmp(), deb_ver_cmp()

default_account.inc check_account(), _check_telnet(), recv_until()

dump.inc dump(), hexdump(), isprint (), line2string ()

ftp_func.inc ftp_authenticate(), ftp_close(), ftp_pasv(), ftp_recv_data(), ftp_recv_line(), ftp_recv_listing(), ftp_send_cmd(), get_ftp_banner()

global_settings.inc debug_print(), log_print()

http_func.inc can_host_asp(), can_host_php(), cgi_dirs(), check_win_dir_trav(), do_check_win_dir_trav(), get_http_banner(), get_http_port(), headers_split(), hex2dec(), __hex_value(), http_40x(), http_is_dead(), http_recv(), http_recv_body(), http_recv_headers2(), http_recv_length(), http_send_recv(), php_ver_match()

http_keepalive.inc check_win_dir_trav_ka(), enable_keepalive(), get_http_page(), http_keepalive_check_connection(), http_keepalive_enabled(), http_keepalive_recv_body(), http_keepalive_send_recv(), is_cgi_installed_ka(), on_exit()

imap_func.inc get_imap_banner ()

misc_func.inc add_port_in_list(), base64(), base64_code(), base64_decode(), cvsdate2unixtime(), dec2hex(), get_mysql_version(), get_rpc_port(), get_service_banner_line(), get_unknown_banner(), hex2raw(), known_service(), pow2(), rand_str(), register_service(), replace_or_set_kb_item(), report_service(), service_is_unknown(), set_mysql_version(), set_unknown_banner()

netop.inc ip_dot2raw(), ip_raw2dot(), netop_banner_items(), netop_check_and_add_banner(), netop_each_found(), netop_kb_derive(), netop_log_detected(), netop_product_ident(), netop_spacepad(), netop_zeropad()

network_func.inc htonl(), htons(), ip_checksum(), is_private_addr(), ms_since_midnight(), ntohl(), test_udp_port()

nfs_func.inc cwd(), mount(), open(), padsz(), read(), readdir(), rpclong(), rpcpad(), str2long(), umount()

nntp_func.inc nntp_article(), nntp_auth(), nntp_connect(), nntp_make_id(), nntp_post()

pingpong.inc udp_ping_pong()

pkg-lib-deb.inc isdpkgvuln()

pop3_func.inc get_pop3_banner ()

qpkg.inc qpkg_check(), qpkg_cmp(), qpkg_ver_cmp()

revisions-lib.inc isdigit(), revcomp()

slackware.inc slack_elt_cmp(), slack_ver_cmp(), slackware_check()

slad_ssh.inc slad_ssh_login ()

smbcl_func.inc bin_dword(), bin_word(), fileread(), GetPEFileVersion (), GetPEProductVersion (), get_windir(), is_domain(), PEVersion(), smbclientavail(), smbgetdir(), smbgetfile(), smbversion()

smb_hotfixes.inc hotfix_check_dhcpserver_installed(), hotfix_check_domain_controler(), hotfix_check_excel_version(), hotfix_check_exchange_installed(), hotfix_check_iis_installed(), hotfix_check_nt_server(), hotfix_check_office_version(), hotfix_check_outlook_version(), hotfix_check_powerpoint_version(), hotfix_check_sp(), hotfix_check_wins_installed(), hotfix_check_word_version(), hotfix_check_works_installed(), hotfix_data_access_version(), hotfix_get_commonfilesdir(), hotfix_get_programfilesdir(), hotfix_get_systemroot(), hotfix_missing()

smtp_func.inc get_smtp_banner(), smtp_close(), smtp_from_header(), smtp_open(), smtp_recv_banner(), smtp_recv_line(), smtp_send_port(), smtp_send_socket(), smtp_to_header()

ssh_func.inc base64decode(), check_pattern(), crypt(), decrypt(), derive_keys(), dh_gen_key(), dh_valid_key(), get_data_size(), get_ssh_banner(), get_ssh_error(), get_ssh_server_version(), get_ssh_supported_authentication(), getstring(), init(), is_sshd_bugged(), kb_ssh_login(), kb_ssh_passphrase(), kb_ssh_password(), kb_ssh_privatekey(), kb_ssh_publickey(), kb_ssh_transport(), kex_packet(), load_array_from_kb(), load_data_from_kb(), load_intarray_from_k load_int_from_kb(), mac_compute(), ntol(), packet_payload(), putbignum(), putstring(), raw_int32(), raw_int8(), recv_ssh_packet(), register_array_in_kb(), register_data_in_kb(), register_intarray_in_kb(), register_int_in_kb(), reuse_connection_init(), send_ssh_packet(), set_ssh_error(), ssh_close_channel(), ssh_close_connection(), ssh_cmd(), ssh_cmd_error(), ssh_dss_verify(), ssh_exchange_identification(), ssh_hex2raw(), ssh_kex2(), ssh_login(), ssh_login_or_reuse_connection(), ssh_open_channel(), ssh_recv(), ssh_reuse_connection(), ssh_rsa_verify(), ssh_userauth2(), update_window_size()

telnet_func.inc get_telnet_banner(), set_telnet_banner()

tftp.inc tftp_get(), tftp_put()

ubuntu.inc deb_str_cmp(), ubuntu_check(), ubuntu_ver_cmp()

uddi.inc create_uddi_xml ()

version_func.inc find_bin(), get_bin_version(), get_string_version(), version_is_equal(), version_is_greater(), version_is_greater_equal(), version_is_less(), version_is_less_equal(), version_test()

9.3.4 Knowledge Base

In order to facilitate the exchange of information between different NVTs and to speed up the scanning process, information collected by plugins can be stored in a Knowledge Base (KB). This enables plugins to build upon the results of other plugins and can help to avoid duplicate scans.

Below is a list of known KB entries and the NASL/NES scripts that are known to set these entries. Keep in mind that this list is likely to be incomplete and subject to change.

Amanda/running (amanda_detect.nasl): 1 = Amanda is running on the remote host

Amanda/version (amanda_version.nasl):

Amap/*/FullBanner (amap.nasl):

Amap/*/PrintableBanner (amap.nasl):

Amap/*/Svc (amap.nasl):

bind/version (bind_version.nasl): Version of the remote BIND server

cfengine/running (cfengine_detect.nasl):

cheopsNG/password (cheopsNG_detect.nasl):

cheopsNG/unprotected (cheopsNG_detect.nasl):

CVE-2003-1011 (apple-sa-2004-08-09.nasl):

ebola/banner/* (find_service2.nasl):

fake_identd/113 (w32_spybot_worm_variant.nasl):

fake_identd/* (slident.nasl, find_service1.nasl, ident_backdoor.nasl):

FindService/tcp/*/get_http (doublecheck_std_services.nasl, apache_SSL_complain.nasl):

FindService/tcp/*/help (doublecheck_std_services.nasl, find_service2.nasl, find_service_3digits.nasl):

FindService/tcp/*/spontaneous (find_service_3digits.nasl, doublecheck_std_services.nasl):

fsp/banner/* (fsp_detection.nasl):

ftp/backdoor (ftp_kibuv_worm.nasl):

ftp/fw1ftpd (ftpserver_detect_type_nd_version.nasl): 1 = The remote server is FW/1 FTPd

ftp/login (logins.nasl): Login to use when connecting to ftp

ftp/msftpd (ftpserver_detect_type_nd_version.nasl): 1 = The remote server is IIS FTPd

ftp/ncftpd (ftpserver_detect_type_nd_version.nasl): 1 = The remote server is NcFTPd

ftp/password (logins.nasl): Password to use when connecting to ftp

ftp/*/AnyUser (DDI_FTP_Any_User_Login.nasl):

ftp/*/backdoor (ftp_kibuv_worm.nasl):

ftp/*/syst (find_service_3digits.nasl):

ftp/vxftpd (ftpserver_detect_type_nd_version.nasl):

ftp/writeable_dir (ftp_writeable_directories.nasl, logins.nasl, ftp_write_dirs.nes):

ftp/wuftpd (ftpserver_detect_type_nd_version.nasl):

global_settings/debug_level (global_settings.nasl):

global_settings/experimental_scripts (global_settings.nasl):

global_settings/http_user_agent (global_settings.nasl):

global_settings/log_verbosity (global_settings.nasl):

global_settings/network_type (global_settings.nasl):

global_settings/report_paranoia (global_settings.nasl):

global_settings/report_verbosity (global_settings.nasl):

global_settings/thorough_tests (global_settings.nasl):

Host/accept_lsrr (source_routed.nasl):

Host/dead (3com_nbx_voip_netset_detection.nasl, blackice_dos.nasl, dont_scan_printers.nasl, ftp_w98_devname_dos.nasl, fw1_udp_DoS.nasl, http_w98_devname_dos.nasl, jolt2.nasl, jolt.nasl, labrea.nasl, linux_icmp_sctp_DoS.nasl, open_all_ports_DoS.nasl, p-smash.nasl, spank.nasl, TLD_wildcard.nasl, vxworks_ftpdDOS.nasl): 1 = The remote host is dead

Host/Debian/dpkg-l (ssh_get_info.nasl):

Host/firewall (securemote_info_leak.nasl, securemote.nasl): (firewall name) The remote host is a firewall

Host/full_scan (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, openvas_tcp_scanner.nes, synscan.nes):

Host/ident_scanned (ident_process_owner.nasl, netstat_portscan.nasl, nmap.nasl):

Host/num_ports_scanned (openvas_tcp_scanner.nes, synscan.nes):

Host/OS (nmap.nasl, nmap_wrapper.nes):

Host/OS/ntp (ntp_open.nasl):

Host/ping_failed (nmap.nasl, nmap_wrapper.nes):

Host/processor/ntp (ntp_open.nasl):

Host/protocol_scanned (ip_protocol_scan.nasl):

Host/scanned (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, nmap_tcp_connect.nes, nmap_wrapper.nes, synscan.nes, openvas_tcp_scanner.nes, ike-scan.nasl): 1 = The remote host has been portscanned

Host/scanners/amap (amap.nasl):

Host/scanners/netstat (netstat_portscan.nasl):

Host/scanners/nmap (nmap.nasl):

Host/scanners/openvas_tcp_scanner (openvas_tcp_scanner.nes):

Host/scanners/snmpwalk (snmpwalk_portscan.nasl):

Host/scanners/synscan (synscan.nes):

Host/scanners/ike-scan (ike-scan.nasl):

Host/tcp_reverse_lsrr (source_routed.nasl):

Host/tcp_seq_idx (nmap.nasl):

Host/tcp_seq (nmap.nasl, nmap_wrapper.nes):

Host/udp_scanned (amap.nasl, netstat_portscan.nasl, nmap.nasl, snmpwalk_portscan.nasl, nmap_wrapper.nes):

http/auth (logins.nasl): login to use when doing HTTP requests

http/login (logins.nasl):

http/password (logins.nasl): password to use when doing HTTP requests

Hydra/cisco-enable/* (hydra_cisco_enable.nasl):

Hydra/cisco/* (hydra_cisco.nasl):

Hydra/cvs/* (hydra_cvs.nasl):

Hydra/ftp/* (hydra_ftp.nasl):

Hydra/http/* (hydra_http.nasl):

Hydra/http-proxy/* (hydra_http_proxy.nasl):

Hydra/icq/* (hydra_icq.nasl):

Hydra/imap/* (hydra_imap.nasl):

Hydra/ldap/* (hydra_ldap.nasl):

Hydra/mssql/* (hydra_mssql.nasl):

Hydra/nntp/* (hydra_nntp.nasl):

Hydra/pcnfs/* (hydra_pcnfs.nasl):

Hydra/rexec/* (hydra_rexec.nasl):

Hydra/sapr3/* (hydra_sapr3.nasl):

Hydra/smtp-auth/* (hydra_smtp_auth.nasl):

Hydra/snmp/* (hydra_snmp.nasl):

Hydra/socks5/* (hydra_socks5.nasl):

Hydra/ssh2/* (hydra_ssh2.nasl):

Hydra/telnet/* (hydra_telnet.nasl):

Hydra/vnc/* (hydra_vnc.nasl):

Ident/*/* (nmap.nasl):

Ident/tcp/* (ident_process_owner.nasl, netstat_portscan.nasl):

iis/global.asa.download (DDI_GlobalASA_Retrieval.nasl):

imap/banner/* (find_service2.nasl):

imap/login (logins.nasl): Imap login to use

imap/password (logins.nasl): Imap password to use

imap/*/Cyrus (cyrus_imap_prelogin_overflow.nasl):

IPProtocol/* (ip_protocol_scan.nasl):

kazaa/username (kazaa_morpheus_detect.nasl):

mssql/SQLVersion (mssql_version.nasl):

mssql/udp/1434 (mssql_ping.nasl):

MSSQL/UDP/Ping (mssql_ping.nasl):

Nmap/*/svc (nmap.nasl):

Nmap/*/version (nmap.nasl):

nntp/local_distrib (nntp_info.nasl):

nntp/login (logins.nasl):

nntp/password (logins.nasl):

nntp/*/cancel (nntp_info.nasl):

nntp/*/noauth (nntp_info.nasl):

nntp/*/posted (nntp_info.nasl):

nntp/*/posting (nntp_info.nasl):

nntp/*/ready (nntp_info.nasl):

nntp/*/supersed (nntp_info.nasl):

nntp/x_no_archive (nntp_info.nasl):

NTP/Running (ntp_open.nasl):

oracle_tnslsnr/*/version (oracle_tnslsnr_version.nasl):

pop2/login (logins.nasl):

pop2/password (logins.nasl):

pop3/login (logins.nasl):

pop3/password (logins.nasl):

/rip/*/broken_source_port (rip_detect.nasl):

/rip/*/version (rip_detect.nasl):

Secret/hydra/logins_file (hydra_options.nasl):

Secret/hydra/passwords_file (hydra_options.nasl):

Secret/SSH/login (ssh_authorization.nasl):

Secret/SSH/passphrase (ssh_authorization.nasl):

Secret/SSH/password (ssh_authorization.nasl):

Secret/SSH/privatekey (ssh_authorization.nasl):

Secret/SSH/publickey (ssh_authorization.nasl):

Services/data_protector/build (hp_data_protector_installed.nasl):

Services/data_protector/version (hp_data_protector_installed.nasl):

Services/three_digits (find_service.nes):

Services/unknown (find_service.nes): Port of unknown service(s)

Services/wrapped (find_service.nes):

Services/www/*/broken (http_func.inc, no404.nasl):

Services/www/*/embedded (3com_nbx_voip_netset_detection.nasl, ciscoworks_detect.nasl, clearswift_mimesweeper_smtp_
cobalt_web_admin_server.nasl, DDI_Cabletron_Web_View.nasl, DDI_F5_Default_Support.nasl, embedded_web_server_detect.nasl, imss_detect.nasl, interspect_detect.nasl, intrushield_console_detect.nasl, iwss_detect.nasl, linksys_multiple_vulns.nasl, raptor_detect.nasl, securenet_provider_detect.nasl, sitescope_management_
sun_cobalt_adaptive_firewall_detect.nasl, tmcm_detect.nasl, websense_detect.nasl, xedus_detect.nasl):

Services/www/*/kerio_wrf (kerio_wrf_management_detection.nasl):

Services/www/*/working (http_func.inc):

Settings/disable_cgi_scanning (global_settings.nasl):

Settings/ExperimentalScripts (global_settings.nasl):

Settings/ThoroughTests (global_settings.nasl):

sip/banner/5060 (sip_detection.nasl):

SMB/domain_filled/0 (smb_authorization.nasl):

SMB/DOMAIN (smbcl_func.inc):

SMB/dont_send_in_cleartext (logins.nasl):

SMB/dont_send_ntlmv1 (logins.nasl):

SMB/ERROR (smbcl_func.inc):

SMB/FILEVERSION/* (smbcl_func.inc):

SMB/login_filled/0 (smb_authorization.nasl):

SMB/name (netbios_name_get.nasl): NetBIOS name of the remote host

SMB/OS (smbcl_func.inc):

SMB/password_filled/0 (smb_authorization.nasl):

SMB/PRODUCTVERSION/* (smbcl_func.inc):

SMB/samba (netbios_name_get.nasl): 1 = The remote SMB server is running Samba

SMB/SERVER (smbcl_func.inc):

SMB/smbclient (smbcl_func.inc):

SMB/transport (cifs445.nasl):

SMB/username (netbios_name_get.nasl):

SMB/WinXP/ServicePack (smb_reg_service_pack_XP.nasl):

SMB/workgroup (netbios_name_get.nasl): Name of the remote host workgroup/domain

SMTP/3comnbx (smtpserver_detect.nasl):

SMTP/domino (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/exim (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/firewall-1 (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/intermail (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/interscan (smtpscan.nasl):

SMTP/microsoft_esmtp_5 (smtpscan.nasl, smtpserver_detect.nasl): 1 = The remote SMTP server is MS SMTP 5

smtp/*/broken (check_smtp_helo.nasl):

smtp/*/denied (check_smtp_helo.nasl):

smtp/*/helo (check_smtp_helo.nasl):

smtp/*/real_banner (smtpscan.nasl):

smtp/*/temp_denied (check_smtp_helo.nasl):

SMTP/postfix (smtpscan.nasl, smtpserver_detect.nasl): 1 = The remote SMTP server is Postfix

SMTP/postoffice (smtpscan.nasl):

SMTP/qmail (smtpscan.nasl, smtpserver_detect.nasl): 1 = The remote SMTP server is qmail

SMTP/sendmail (smtpscan.nasl): 1 = The remote SMTP server is Sendmail

SMTP/sendmail (smtpserver_detect.nasl): 1 = The remote SMTP server is Sendmail

SMTP/snubby (smtpserver_detect.nasl):

SMTP/wrapped (check_smtp_helo.nasl): 1 = The remote sendmail is wrapped

SMTP/wrapped (smtpserver_detect.nasl): 1 = The remote sendmail is wrapped

SMTP/xmail (smtpscan.nasl, smtpserver_detect.nasl):

SMTP/zmailer (smtpserver_detect.nasl):

SNMP/community (snmp_default_communities.nasl): Name of a valid SNMP community

SNMP/port (snmp_default_communities.nasl):

SNMP/running (snmp_detect.nasl):

socks/*/auth/* (socks.nasl):

SSH/banner/* (ssh_detect.nasl):

ssh/login/freebsdpatchlevel (gather-package-list.nasl):

ssh/login/freebsdpkg (gather-package-list.nasl):

ssh/login/freebsdrel (gather-package-list.nasl):

ssh/login/gentoo (gather-package-list.nasl):

ssh/login/packages (gather-package-list.nasl):

ssh/login/pkg (gather-package-list.nasl):

ssh/login/release (gather-package-list.nasl):

ssh/login/rpms (gather-package-list.nasl):

ssh/login/slackpack (gather-package-list.nasl):

ssh/login/solosversion (gather-package-list.nasl): Solaris OS version as reported by uname -r

ssh/login/solhardwaretype (gather-package-list.nasl): Solaris hardware type as reported by uname -p

ssh/login/solpackages (gather-package-list.nasl): Solaris packages as repored by pkginfo

ssh/login/solpatches (gather-package-list.nasl): Solaris patches as reported by showrev -p

ssh/login/uname (gather-package-list.nasl, ssh_get_info.nasl):

SSH/supportedauth/* (ssh_detect.nasl):

SSH/textbanner/* (ssh_detect.nasl):

TCPScanner/NbPasses (openvas_tcp_scanner.nes):

TCPScanner/OpenPortsNb (openvas_tcp_scanner.nes):

TCPScanner/ClosedPortsNb (openvas_tcp_scanner.nes):

TCPScanner/FilteredPortsNb (openvas_tcp_scanner.nes):

TCPScanner/RSTRateLimit (openvas_tcp_scanner.nes):

tftp/get_file (tftp_grab_file.nes):

tftp/backdoor (tftpd_backdoor.nasl):

tftp/*/backdoor (tftpd_backdoor.nasl):

tftp/*/filecontent/* (tftpd_dir_trav.nasl):

tftp/*/filename/* (tftpd_dir_trav.nasl):

tftp/*/get_file (tftpd_dir_trav.nasl):

tftp/*/overflow (tftpd_overflow.nasl):

tftp/*/smalloverflow (tftpd_small_overflow.nasl):

/tmp/cgibin (DDI_Directory_Scanner.nasl):

/tmp/http/auth/* (http_login.nasl):

/tmp/hydra/empty_password (hydra_options.nasl):

/tmp/hydra/exit_ASAP (hydra_options.nasl):

/tmp/hydra/login_password (hydra_options.nasl):

/tmp/hydra/tasks (hydra_options.nasl):

/tmp/hydra/timeout (hydra_options.nasl):

/tmp/rpc/noportmap/* (misc_func.inc):

/tmp/UnableToRun/14254 (nikto.nasl):

/tmp/UnableToRun/14255 (nmap.nasl):

/tmp/UnableToRun/14663 (amap.nasl):

/tmp/UnableToRun/91984 (ldapsearch.nasl):

/tmp/UnableToRun/80000 (ike-scan.nasl):

Transport/SSL (find_service.nes):

webmin/*/version (webmin.nasl):

www/abyss (http_version.nasl):

www/akamaighost (http_version.nasl):

www/alchemy (http_version.nasl):

www/alibaba (http_version.nasl):

www/allegro (http_version.nasl):

www/anti-OpenVAS/*/rand-url (anti_nessus.nasl):

www/anti-OpenVAS/*/user-agent (anti_nessus.nasl):

www/anweb (http_version.nasl):

www/aolserver (http_version.nasl):

www/apache (http_version.nasl): 1 = The remote server is running Apache

www/appleshareip (http_version.nasl):

www/badblue (http_version.nasl):

www/banner/* (apache_SSL_complain.nasl, http_version.nasl):

www/BitKeeper (http_version.nasl):

www/buggy_post_crash (monkeyweb_post_DoS.nasl):

www/caudium (http_version.nasl):

www/cern (http_version.nasl): 1 = The remote server is running CERN httpd

www/communigatepro (http_version.nasl):

www/communique (http_version.nasl):

www/compaq (http_version.nasl):

www/cougar (http_version.nasl):

www/cups (http_version.nasl):

www/doc_browseable (doc_browsable.nasl):

www/domino (http_version.nasl): 1 = The remote server is running domino

www/domino/*/db (domino_default_db.nasl):

www/emweb (http_version.nasl):

www/filemakerpro (http_version.nasl):

www/firstclass (http_version.nasl):

www/goahead (http_version.nasl):

www/hmap/*/banner_ok (www_fingerprinting_hmap.nasl):

www/hmap/*/banner_regex (www_fingerprinting_hmap.nasl):

www/hmap/*/description (www_fingerprinting_hmap.nasl):

www/hmap/*/raw_signature (www_fingerprinting_hmap.nasl):

www/hmap/*/signature (www_fingerprinting_hmap.nasl):

www/ibm-http (http_version.nasl):

www/ideawebserver (http_version.nasl):

www/iis (http_version.nasl):

www/iplanet (http_version.nasl):

www/ipswitch-imail (http_version.nasl):

www/jetty (http_version.nasl):

www/jigsaw (http_version.nasl):

www/KFWebServer (http_version.nasl):

www/linuxconf (http_version.nasl):

www/miniserv (http_version.nasl):

www/multiple_get/* (www_multiple_get.nasl):

www/ncsa (http_version.nasl):

www/netcache (http_version.nasl):

www/netscape-commerce (http_version.nasl):

www/netscape-fasttrack (http_version.nasl):

www/netware (http_version.nasl):

www/novell (http_version.nasl):

www/omnihttpd (http_version.nasl):

www/OracleApache (http_version.nasl):

www/oracle-web-listener (http_version.nasl):

www/pi3web (http_version.nasl):

www/*/generic_xss (cross_site_scripting.nasl):

www/*/punBB (punBB_detect.nasl):

www/*/vBulletin (vbulletin_detect.nasl):

www/*/webmin (webmin.nasl):

www/real_banner/* (http_version.nasl):

www/resin (http_version.nasl):

www/roxen (http_version.nasl):

www/sambar (http_version.nasl):

www/savant (http_version.nasl):

www/simpleserver (http_version.nasl):

www/squid (http_version.nasl):

www/statistics-server (http_version.nasl):

www/stronghold (http_version.nasl):

www/stweb (http_version.nasl):

www/theserver (http_version.nasl):

www/thttpd (http_version.nasl):

www/tigershark (http_version.nasl):

www/tomcat (http_version.nasl):

www/too_big_post_crash (monkeyweb_too_big_post.nasl):

www/too_long_url_crash (oracle9iAS_too_long_url.nasl, ws4e_too_long_url.nasl):

www/tux (http_version.nasl):

www/visualroute (http_version.nasl):

www/vnc (vnc_http.nasl):

www/vqserver (http_version.nasl):

www/wdaemon (http_version.nasl):

www/weblogic (http_version.nasl):

www/webserver4everyone (http_version.nasl):

www/websitepro (http_version.nasl):

www/webstar (http_version.nasl):

www/wwwfileshare (http_version.nasl):

www/xitami (http_version.nasl):

www/zeus (http_version.nasl):

www/zope (http_version.nasl):

X11/*/answer (X.nasl):

X11/*/open (X.nasl):

X11/*/version (X.nasl):

xedus/*/running (xedus_detect.nasl):

zebra/banner/* (find_service2.nasl):

zebra/banner/* (zebra_dos.nasl):

zonealarm/version (zone_alarm_local_dos.nasl):

9.4 Test and debugging procedures

There are different approaches to test your OpenVAS NVTs; for example, the network TCP/IP tests differs from the local security tests. For the local security test, it is important to see the command line and what was queried on a local system using a shell. An excellent start is using the `openvas-nasl` tool to execute your script on the target environment to see if any error messages come up.

9.4.1 Testing a local vulnerability

Here is an example of using the `openvas-nasl` tool to perform a test:

First test if your script written in NASL is syntactically correct. This could be done by using openvas-nasl with the -p option, e.g.:

```
# openvas-nasl -p broken-example.nasl
syntax error, unexpected IDENT, expecting ')'
Parse error at or near line 17
```

This is telling us that this script has a syntax error. Test the functionality of your script only after you've made sure your script contains only syntactically correct NASL.

Now you can test on your target host, if the LVT is correct, by writing debug-output into a debug file:

```
 openvas-nasl  -T /tmp/debug-lvt.txt -X  example-lvt.nasl
```

The debug output will be written into the debug-lvt.txt file, which in this example will look like this:

```
[...]
NASL:0196> make_list(...)
[9831]() NASL> [080c9968] <- "qpkg"
[9831]() NASL> [080c9a00] <- "-nc"
[9831]() NASL> [080c9f38] <- "-I"
[9831]() NASL> [080c9f98] <- "-v"
[9831](example-lvt.nasl) NASL> Call make_list(1: "qpkg", 2: "-nc", 3: "-I", 4:
"-v")
[9831](example-lvt.nasl) NASL> Return make_list: ???? (DYN_ARRAY (64))
[9831]() NASL> [080c9e88] <- (VAR2_ARRAY)
[9831](example-lvt.nasl) NASL> Call pread(cmd: "qpkg", argv: ???? (DYN_ARRAY
(64)))
[9831](example-lvt.nasl) NASL> Return pread: "qpkg: invalid option -- n
Usage: qpkg <opt..."
[9831]() NASL> [080c95c0] <- "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
NASL:0199> if (! (qpkg_list)) { ... }
[9831](example-lvt.nasl) NASL> [080c95c0] -> "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
```

```
NASL:0201> if (((arch) && (my_arch)) && (my_arch >!< arch)) { ... }
[9831](example-lvt.nasl) NASL> [080c9948] -> undef
NASL:0201> l=egrep(...);
NASL:0201> egrep(...)
[9831](example-lvt.nasl) NASL> [080c95c0] -> "qpkg: invalid option -- n
Usage: qpkg <opts> <misc args> : manipulate Gentoo binpkgs

Options: -[cpP:vqChV]
  -c, --clean          * clean pkgdir of unused binary files
  -p, --pretend        * pretend only
  -P, --pkgdir   <arg> * alternate package directory
  -v, --verbose        * Make a lot of noise
  -q, --quiet          * Tighter output; suppress warnings
  -C, --nocolor        * Don't output color
  -h, --help           * Print this help and exit
  -V, --version        * Print version and exit
"
[9831]() NASL> [080c9890] <- "qpkg: invalid option ? n
[...]
```

The last line tells us that an incorrect syntax for the qpkg tool was given to the LVT.

9.4.2 Testing a network vulnerability

Here is an example using the openvas-nasl tool to perform a test:

First test if your script written in NASL is syntactically correct. This could be done by using the openvas-nasl tool with the -p option, e.g.:

```
# openvas-nasl -p broken-example.nasl
syntax error, unexpected IDENT, expecting ')'
Parse error at or near line 17
```

This is telling us that this script has a syntax error. Test the functionality of your script after making sure that your script contains only correct NASL.

The test on the network is a bit more complicated. To test if the right packet was sent, you can use TCPDUMP to capture the communication between the host and the client, e.g.:

```
# tcpdump -i lo -w debug.pcap -s 1450
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 1450 bytes
10 packets captured
20 packets received by filter
0 packets dropped by kernel
```

Now it is possible to decode the content of the network communication.

```
 # tcpdump -vvvv -n -r debug.pcap
reading from file debug.pcap, link-type EN10MB (Ethernet)
15:45:52.474613 IP (tos 0x0, ttl 64, id 60969, offset 0, flags [DF], proto TCP
(6), length 60) 127.0.0.1.53655 > 127.0.0.1.24: S, cksum 0x80c9 (correct),
1315997236:1315997236(0) win 32792 <mss 16396,sackOK,timestamp 5466141
0,nop,wscale 6>
15:45:52.474618 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6),
length 60) 127.0.0.1.24 > 127.0.0.1.53655: S, cksum 0x64b5 (correct),
1311860089:1311860089(0) ack 1315997237 win 32768 <mss 16396,sackOK,timestamp
5466141 5466141,nop,wscale 6>
15:45:52.474638 IP (tos 0x0, ttl 64, id 60970, offset 0, flags [DF], proto TCP
```

```
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x4bd8 (correct),
1:1(0) ack 1 win 513 <nop,nop,timestamp 5466141 5466141>
15:45:52.474797 IP (tos 0x0, ttl 64, id 3431, offset 0, flags [DF], proto TCP
(6), length 72) 127.0.0.1.24 > 127.0.0.1.53655: P, cksum 0xfe3c (incorrect (->
0x941e), 1:21(20) ack 1 win 512 <nop,nop,timestamp 5466141 5466141>
15:45:52.474829 IP (tos 0x0, ttl 64, id 60971, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x4bc3 (correct),
1:1(0) ack 21 win 513 <nop,nop,timestamp 5466142 5466141>
15:45:52.475572 IP (tos 0x0, ttl 64, id 60972, offset 0, flags [DF], proto TCP
(6), length 68) 127.0.0.1.53655 > 127.0.0.1.24: P, cksum 0xfe38 (incorrect (->
0xefa4), 1:17(16) ack 21 win 513 <nop,nop,timestamp 5466142 5466141>
15:45:52.475586 IP (tos 0x0, ttl 64, id 3432, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.24 > 127.0.0.1.53655: ., cksum 0x4bb3 (correct),
21:21(0) ack 17 win 512 <nop,nop,timestamp 5466142 5466142>
15:45:57.479223 IP (tos 0x0, ttl 64, id 60973, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: F, cksum 0x46ce (correct),
17:17(0) ack 21 win 513 <nop,nop,timestamp 5467393 5466142>
15:45:57.479279 IP (tos 0x0, ttl 64, id 3433, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.24 > 127.0.0.1.53655: F, cksum 0x41eb (correct),
21:21(0) ack 18 win 512 <nop,nop,timestamp 5467393 5467393>
15:45:57.479296 IP (tos 0x0, ttl 64, id 60974, offset 0, flags [DF], proto TCP
(6), length 52) 127.0.0.1.53655 > 127.0.0.1.24: ., cksum 0x41ea (correct),
18:18(0) ack 22 win 513 <nop,nop,timestamp 5467393 5467393>
```

If a deeper packet analysis is needed, tools like `wireshark` are able to read such files in `pcap` format, and perform a close analysis of all type of network communication packets.

The `openvas-nasl` interpreter also provides us with a logfile at `/tmp/debug-nvt.txt`. This file helps us to debug NASL based NVTs:

```
[...]
NASL:0277> register_int_in_kb(...)
[9905](ssh_detect24.nasl) NASL> [0811e310] -> 0
[9905]() NASL> [08120328] <- 0
[9905]() NASL> [08120358] <- "Secret/SSH/bugged_sshd"
[9905](ssh_detect24.nasl) NASL> Call register_int_in_kb(int: 0, name:
"Secret/SSH/bugged_sshd")
NASL:0055> if ((! (defined_func(...))) || (! (_reuse_connection))) { ... }
NASL:0054> defined_func(...)
[9905]() NASL> [081203a8] <- "replace_kb_item"
[9905](ssh_detect24.nasl) NASL> Call defined_func(1: "replace_kb_item")
[9905](ssh_detect24.nasl) NASL> Return defined_func: 1
[9905](ssh_detect24.nasl) NASL> [0811e2d8] -> undef
NASL:0054> return 0;
[9905](ssh_detect24.nasl) NASL> Return register_int_in_kb: 0
[9905](ssh_detect24.nasl) NASL> Return init: FAKE
NASL:1771> server_version=ssh_exchange_identification(...);
NASL:1771> ssh_exchange_identification(...)
[9905](ssh_detect24.nasl) NASL> [0811fde0] -> 1000000
[9905]() NASL> [08120688] <- 1000000
[9905](ssh_detect24.nasl) NASL> Call ssh_exchange_identification(socket:
1000000)
NASL:0377> local_var ...
NASL:0379> buf=recv_line(...);
NASL:0379> recv_line(...)
[9905](ssh_detect24.nasl) NASL> [08120688] -> 1000000
[9905]() NASL> [081207b0] <- 1000000
[9905]() NASL> [081207d0] <- 1024
[9905](ssh_detect24.nasl) NASL> Call recv_line(socket: 1000000, length: 1024)
[9905](ssh_detect24.nasl) NASL> Return recv_line: "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081202d0] <- "SSH-2.0-FreeSSH_9.9
"
NASL:0388> if (! (buf)) { ... }
```

```
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
NASL:0394> if (! (ereg(...))) { ... }
NASL:0388> ereg(...)
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081206a8] <- "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081207b0] <- "^SSH-*[0-9]\.*[0-9]-*[^\n]"
[9905](ssh_detect24.nasl) NASL> Call ereg(string: "SSH-2.0-FreeSSH_9.9
", pattern: "^SSH-*[0-9]\.*[0-9]-*[^\n]")
[9905](ssh_detect24.nasl) NASL> Return ereg: 1
NASL:0394> sshversion=split(...);
NASL:0394> split(...)
[9905](ssh_detect24.nasl) NASL> [081202d0] -> "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [08120638] <- "SSH-2.0-FreeSSH_9.9
"
[9905]() NASL> [081207b0] <- "-"
[9905]() NASL> [0811fff8] <- 0
[9905](ssh_detect24.nasl) NASL> Call split(1: "SSH-2.0-FreeSSH_9.9
", sep: "-", keep: 0)
[9905](ssh_detect24.nasl) NASL> Return split: ???? (DYN_ARRAY (64))
[9905]() NASL> [081202f0] <- (VAR2_ARRAY)
NASL:0395> num=split(...);
NASL:0395> split(...)
[...]
```

This information should be sufficient to solve the problem. If not, it might be an OpenVAS bug in the script engine. To detect this, compile OpenVAS NASL with debug symbols and use GDB. More information on GBD can be found at: http://www.gnu.org/software/gdb/gdb.html.

9.5 Writing SMBclient-based WLSC NASL Scripts

(by Carsten Koch Mauthe)

The SMB-Client API is made available as `smbcl_func.inc`. This file has to be included in any WLSC Script.

smbclientavail()

This function returns TRUE if smbclient can be used from within openvasd. It also sets the knowledge base item “SMB/smbclient”. It should be called first in any WLSC Script. Example:

```
include("smbcl_func.inc");
if(!get_kb_item("SMB/smbclient"))
{
  smbclientavail();
}
if(get_kb_item("SMB/smbclient"))
{
  do something;
}
else
{
```

```
  report = string("SMBClient not found on this host !");
  security_note(port:0, proto:"SMBClient", data:report);
  exit(0);
}
```

smbversion()

This function returns TRUE if successful and writes the DOMAIN, OS Version and SMB Server version to the knowledge base as items “SMB/DOMAIN”, “SMB/OS” and “SMB/SERVER”.

smbgetfile(share, filename, tmp_filename)

Use this function to get a file from the target host and save this file locally using tmp_filename. Returns TRUE if successful. Example:

```
tmp_filename = get_tmp_dir() + "tmpfile" + rand();
orig_filename = "C:\Windows\systems32\ntdll.dll";
smbgetfile(share: "C$", filename: orig_filename, tmp_filename: tmp_filename)
```

smbgetdir(share, dir, typ)

Use this function to get directory entries from the SMB source.

typ = 0: All entries

typ = 1: Only file entries

typ = 2: Only directory entries

With this it is possible to check for one or more files or directories. Returns NULL or array of Strings containing found entries. Example:

```
r = smbgetdir(share: "C$", dir: "C:\Windows\systems32\*.dll", typ: 1);
```

GetPEFileVersion (tmp_filename, orig_filename)

This function returns the version of Windows PE/32 executables like .exe or .dll. Together with smbgetfile, this can be used to check for Windows vulnerabilities.

Example:

```
tmp_filename = get_tmp_dir() + "tmpfile" + rand();
orig_filename = "C:\Windows\systems32\ntdll.dll";
if(smbgetfile(share: "C$", filename: orig_filename,
              tmp_filename: tmp_filename))
{
  v = GetPEFileVersion(tmp_filename:tmp_filename,
                       orig_filename:orig_filename);
  if(v = "1.2.3.4")
  {
    ...
```

get_windir()

This function returns the standard Windows folder WINNT or WINDOWS, depending on the OS found by “smbversion”.

9.5.1 Example

This is a complete NASL test for a Windows Local Security Check. It can be found in the OpenVAS plugins as win_CVE-2007-0043.nasl.

```
#
# This script was written by
# Carsten Koch-Mauthe
# <c.koch-mauthe at dn-systems.de>
#
# This script is released under the GNU GPLv2
#
# $Revision: 01 $

if(description)
{
  if (OPENVAS_NASL_LEVEL >= 2206)
  {
    script_oid("1.3.6.1.4.1.25623.1.0.90010");
  }
  else
  {
    script_id(90010);
  }
  script_version ("$Revision: 01 $");
  script_cve_id("CVE-2007-0043");
  name["english"] = ".NET JIT Compiler Vulnerability";
  script_name(english:name["english"]);

  desc["english"] =
"The remote host is affected by the vulnerabilities described in CVE-2007-0043

Checking if System.web.dll version is less than 2.0.50727.832

Impact
    The Just In Time (JIT) Compiler service in Microsoft
    .NET Framework 1.0, 1.1, and 2.0 for Windows 2000, XP,
    Server 2003, and Vista allows user-assisted remote
    attackers to execute arbitrary code via unspecified
    vectors involving an unchecked buffer, probably a
    buffer overflow, aka .NET JIT Compiler Vulnerability.
    Checking if System.web.dll version is less than 2.0.50727.832

References:
    http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0043

Solution:
    All users should upgrade to the latest version.


Risk factor : High";

  script_description(english:desc["english"]);
  summary["english"] = "Test for .NET JIT Compiler Vulnerability";
  script_summary(english:summary["english"]);
  script_category(ACT_GATHER_INFO);
  script_copyright(english:"This script is under GPLv2");
  family["english"] = "Windows.NET";
```

```
  script_family(english:family["english"]);
  exit(0);
}

#
# The code starts here
#

include("version_func.inc");
include("smbcl_func.inc");

if(!get_kb_item("SMB/smbclient"))
{
  smbclientavail();
}
test_version = "2.0.50727.832";

if(get_kb_item("SMB/smbclient"))
{
  if(smbversion() == 0)
  {
    report = string("Error getting SMB-Data -> " +
                    get_kb_item("SMB/ERROR"));
    security_note(port:0, proto:"SMBClient", data:report);
    exit(0);
  }
}
else
{
  report = string("SMBClient not found on this host !");
  security_note(port:0, proto:"SMBClient", data:report);
  exit(0);
}

win_dir = get_windir();
if(!isnull(win_dir))
{
  path = win_dir+"Microsoft.NET\Framework\";
  filespec = "v2*";
  r = smbgetdir(share: "C$", dir: path+filespec, typ: 2);
  if(!isnull(r))
  {
    filespec = r[0] + "\" + "system.web.dll";
    r = smbgetdir(share: "C$", dir: path+filespec, typ: 1);
    if(!isnull(r))
    {
      tmp_filename = get_tmp_dir() + "tmpfile" + rand();
      orig_filename = path + filespec;
      if(smbgetfile(share: "C$", filename: orig_filename,
                    tmp_filename: tmp_filename))
      {
        v = GetPEFileVersion(tmp_filename:tmp_filename,
                             orig_filename:orig_filename);
        unlink(tmp_filename);
        if(version_is_less(version: v, test_version: test_version))
        {
          security_hole(port:0, proto:"SMB");
          report = report + "Fileversion : C$ " +
                    orig_filename + " "+ v + string("\n");
          security_hole(port:0, proto:"SMB", data:report);
        }
      }
      else
      {
        report = string("Error getting SMB-File -> " +
                        get_kb_item("SMB/ERROR")) + string("\n");
```

```
        security_note(port:0, proto:"SMB", data:report);
      }
    }
  }
  else
  {
    report = string(".NET V2xx not found/no access -> " +
                    get_kb_item("SMB/ERROR")) + string("\n");
    security_note(port:0, proto:"SMB", data:report);
  }
}

exit(0);
```

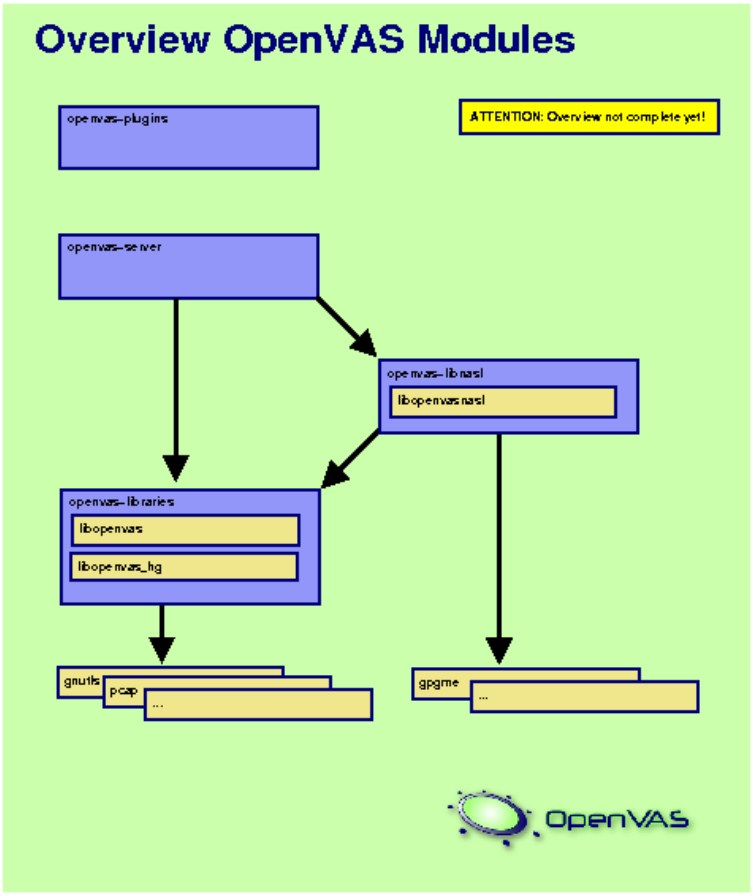
10 Developers Guide for OpenVAS Server and Client

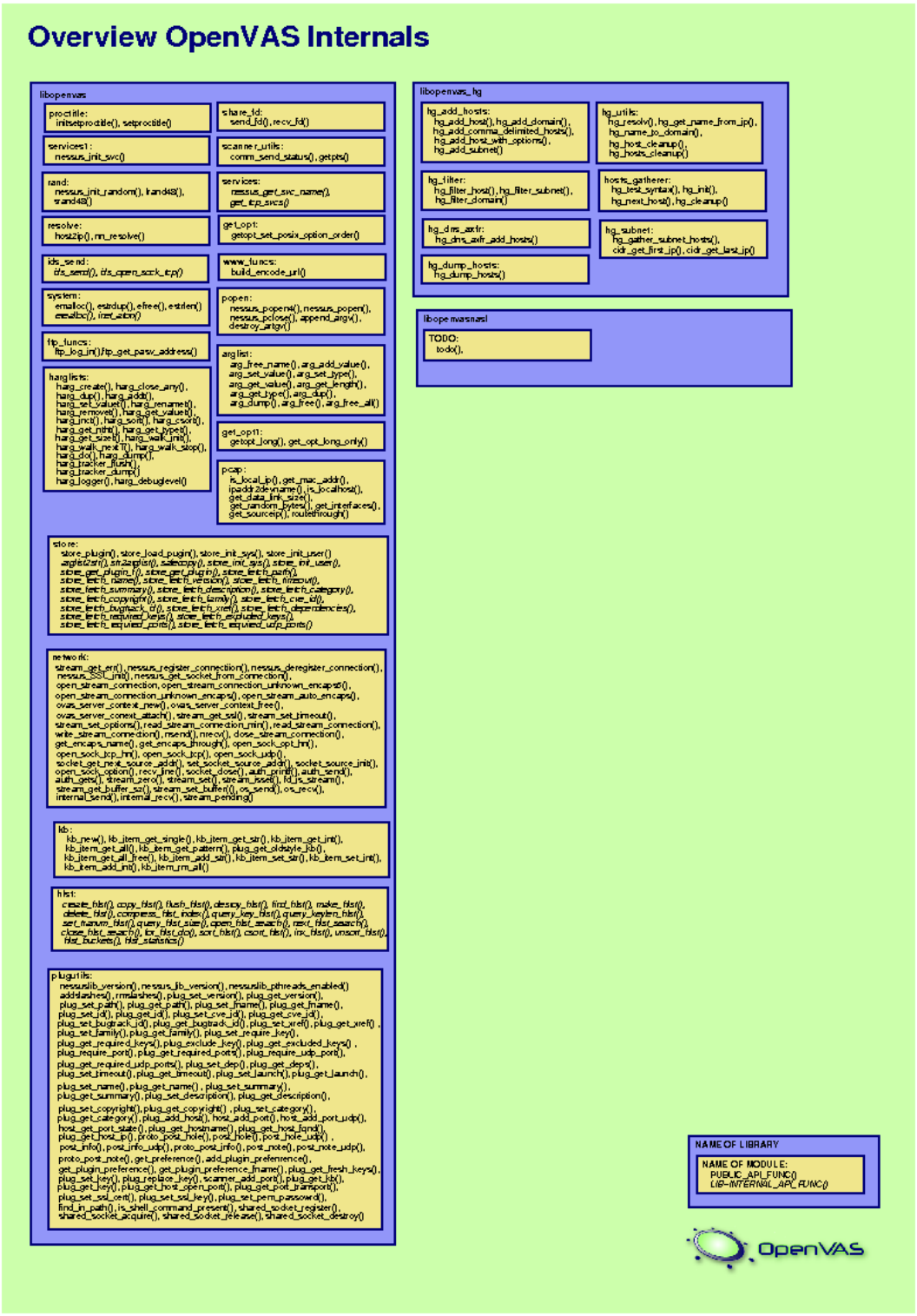
(by Jan-Oliver Wagner)

10.1 The OpenVAS Source Code Map

A large chunk of the OpenVAS codebase was inherited from Nessus. As with any project of this size, it can be quite difficult to find your way around if you are new to the code. While the Nessus code was very well structured in some places, it proved to be rather chaotic and difficult to understand in other places. In order to better visualize the codebase, the OpenVAS project has begun to create maps of the source code and its structure.

Below you will find a first draft of the overview and internal structure. Keep in mind that this is a work in progress; updated maps are likely to be available on the OpenVAS website.




Overview OpenVAS Internals
libopenvas
proctitle:
initsetproctitle(), setproctitle()
services1:
nessus_init_svc()
rand:
nessus_init_random(), krand48(), srand48()
resolve:
host2ip(), rm_resolve()
kb_send:
kb_send(), kb_open_sock_kb()
system:
emalloc(), estrdup(), efree(), estrlen()
emallbc(), inet_aton()
ftp_funcs:
ftp_log_in()ftp_get_pasv_address()
harglists:
harg_create(), harg_close_any(),
harg_dup(), harg_add(),
harg_set_value(), harg_rename(),
harg_removet(), harg_get_value(),
harg_incr(), harg_soft(), harg_csort(),
harg_get_nthi(), harg_get_type(),
harg_get_sizet(), harg_walk_init(),
harg_walk_next1(), harg_walk_stop(),
harg_dot(), harg_dump(),
harg_tracker_flush()
harg_tracker_dump()
harg_logger(), harg_debuglevel()
share_fd:
send_fd(), recv_fd()
scanner_utils:
comm_send_status(), getpts()
services:
nessus_get_svc_name(),
get_tcp_svcs()
get_opt:
getopt_set_posix_option_order()
www_funcs:
build_encode_url()
popen:
nessus_popen4(), nessus_popen(),
nessus_pclose(), append_argv(),
destroy_argv()
arglist:
arg_free_name(), arg_add_value(),
arg_set_value(), arg_set_type(),
arg_get_value(), arg_get_length(),
arg_get_type(), arg_dup(),
arg_dump(), arg_free(), arg_free_all()
get_opt1:
getopt_long(), get_opt_long_only()
pcap:
is_local_ip(), get_mac_addr(),
ipaddr2devname(), is_localhost(),
get_data_link_size(),
get_random_bytes(), get_interfaces(),
get_sourceip(), routethrough()
store:
store_plugin(), store_load_plugin(), store_init_sys(), store_init_user()
arglist2str(), str2arglist(), safeempty(), store_init_sys(), store_init_user(),
store_get_plugin_t(), store_get_plugin(), store_fetch_path(),
store_fetch_name(), store_fetch_version(), store_fetch_timeout(),
store_fetch_summary(), store_fetch_description(), store_fetch_category(),
store_fetch_copyright(), store_fetch_family(), store_fetch_cve_id(),
store_fetch_bugtraq_id(), store_fetch_xref(), store_fetch_dependencies(),
store_fetch_required_keys(), store_fetch_excluded_keys(),
store_fetch_required_ports(), store_fetch_required_udp_ports()
network:
stream_get_err(), nessus_register_connection(), nessus_deregister_connection(),
nessus_SSL_init(), nessus_get_socket_from_connection(),
open_stream_connection, open_stream_connection_unknown_encaps(),
open_stream_connection_unknown_encaps(), open_stream_auto_encaps(),
ovas_server_context_new(), ovas_server_context_free(),
ovas_server_conext_attach(), stream_get_ssl(), stream_set_timeout(),
stream_set_options(), read_stream_connection_min(), read_stream_connection(),
write_stream_connection(), nsend(), nrecv(), close_stream_connection(),
get_encaps_name(), get_encaps_through(), open_sock_opt_hn(),
open_sock_tcp_hn(), open_sock_tcp(), open_sock_udp(),
socket_get_next_source_addr(), set_socket_source_addr(), socket_source_init(),
open_sock_option(), recv_line(), socket_close(), auth_printf(), auth_send(),
auth_gets(), stream_zero(), stream_set(), stream_nsend(), fd_is_stream(),
stream_get_buffer_siz(), stream_set_buffer(), os_send(), os_recv(),
internal_send(), internal_recv(), stream_pending()
kb:
kb_new(), kb_item_get_single(), kb_item_get_str(), kb_item_get_int(),
kb_item_get_all(), kb_item_get_pattern(), plug_get_oldstyle_kb(),
kb_item_get_all_free(), kb_item_add_str(), kb_item_set_str(), kb_item_set_int(),
kb_item_add_int(), kb_item_rm_all()
hlst:
create_hlst(), copy_hlst(), flush_hlst(), destroy_hlst(), first_hlst(), make_hlst(),
delete_hlst(), compress_hlst_index(), query_key_hlst(), query_keyitem_hlst(),
set_maxvm_hlst(), query_hlst_size(), open_hlst_search(), next_hlst_search(),
close_hlst_search(), for_hlst_do(), sort_hlst(), csort_hlst(), ins_hlst(), unsort_hlst(),
hlst_buckets(), hlst_statistics()
plugutils:
nessuslib_version(), nessus_lib_version(), nessuslib_pthreads_enabled()
addslashes(), rmslashes(), plug_set_version(), plug_get_version(),
plug_set_path(), plug_get_path(), plug_set_fname(), plug_get_fname(),
plug_set_id(), plug_get_id(), plug_set_cve_id(), plug_get_cve_id(),
plug_set_bugtraq_id(), plug_get_bugtraq_id(), plug_set_xref(), plug_get_xref() ,
plug_set_family(), plug_get_family(), plug_set_require_key(),
plug_get_required_keys(), plug_exclude_key(), plug_get_excluded_keys() ,
plug_require_port(), plug_get_required_ports(), plug_require_udp_port(),
plug_get_required_udp_ports(), plug_set_dep(), plug_get_deps(),
plug_set_timeout(), plug_get_timeout(), plug_set_launch(), plug_get_launch(),
plug_set_name(), plug_get_name() , plug_set_summary(),
plug_get_summary(), plug_set_description(), plug_get_description(),
plug_set_copyright(), plug_get_copyright() , plug_set_category(),
plug_get_category(), plug_add_host(), host_add_port(), host_add_port_udp(),
host_get_port_state(), plug_get_hostname(), plug_get_host_fqdn(),
plug_get_host_ip(), proto_post_hole(), post_hole(), post_hole_udp() ,
post_info(), post_info_udp(), proto_post_info(), post_note(), post_note_udp(),
proto_post_note(), get_preference(), add_plugin_preference(),
get_plugin_preference(), get_plugin_preference_fname(), plug_get_fresh_keys(),
plug_set_key(), plug_replace_key(), scanner_add_port(), plug_get_kb(),
plug_get_key(), plug_get_host_open_port(), plug_get_port_transport(),
plug_set_ssl_cert(), plug_set_ssl_key(), plug_set_key_password(),
find_in_path(), is_shell_command_present(), shared_socket_register(),
shared_socket_acquire(), shared_socket_release(), shared_socket_destroy()
libopenvas_hg
hg_add_hosts:
hg_add_host(), hg_add_domain(),
hg_add_comma_delimited_hosts(),
hg_add_host_with_options(),
hg_add_subnet()
hg_utils:
hg_resolv(), hg_get_name_from_ip(),
hg_name_to_domain(),
hg_host_cleanup(),
hg_hosts_cleanup()
hg_filter:
hg_filter_host(), hg_filter_subnet(),
hg_filter_domain()
hosts_gatherer:
hg_test_syntax(), hg_init(),
hg_next_host(), hg_cleanup()
hg_dns_axfr:
hg_dns_axfr_add_hosts()
hg_subnet:
hg_gather_subnet_hosts(),
cidr_get_first_ip(), cidr_get_last_ip()
hg_dump_hosts:
hg_dump_hosts()
libopenvasnasl
TODO:
todo(),
NAME OF LIBRARY
NAME OF MODULE:
PUBLIC_API_FUNC()
LIB-INTERNAL_API_FUNC()
OpenVAS

10.2 Source Code Branches for Stable and In-Development

If you look at the OpenVAS source code for the first time, you might wonder what all those branches and tags are for and why most changes tend to occur in the "trunk" section.

Similar to many other projects, the OpenVAS project uses the concepts of branches and tags to signify important steps in the source code management. As you might have already noticed, almost all changes to the source code happen in the trunk section. The trunk is the place where bug fixes, new features and other additions appear first. [1]

When a component has reached a stable state and the changes to this component have been evaluated by other developers, a decision is made to release a new version of this component. An intent to release a new version is usually announced on the developer mailing list. If the developers agree with this intent, a release is prepared. As part of the release process, the revision of the component that will become the release is tagged and appears in the tags section with the new version number, like `openvas-client-release-1.1.0`.

A tagged version is a snapshot of the source code as it was at the time of the release; this, among other things, allows users to use the source code management to check out older versions should this become necessary. No further development will occur on a tagged version. If changes to a tagged version are needed, these changes will occur in the trunk or the branch where this release originated from and will be released as a new release with a new tag.

As you have guessed from the last sentence, branches are different from tags in that further development can occur in a branch. The code is usually branched right before major changes are made that might render the trunk unstable for some time and cause incompatibilities with previous versions. One reason for branching is enabling these major changes to the codebase while still retaining a separate branch where minor maintenance changes to already released versions can occur.

For example, after releasing the 1.1.5 version of openvas-client, it might be decided that upcoming major changes justify a new branch. This leads to the creation of a branch name `openvas-client-1-1`. Future releases of the 1.1 series (1.1.6, 1.1.7) will come from this branch, while the trunk undergoes major changes that will ultimately lead to the release of 1.2.0.

If errors are found in the code which affect both the trunk and one or more branches, the are usually first fixed in the trunk and then propagated to the branches. This propagation is called "backporting" the fixes. Be aware that these backports make up the bulk of the changes to most branches. Only under very rare circumstances should it become necessary to propagate changes in the reverse direction, i.e. merge changes in the branch back into the trunk. Developers should avoid making changes to a branch that have not yet been incorporated into to the trunk unless there is a very good reason to do so.

10.3 Code Quality and Code Security

Especially as an IT security product, the OpenVAS project is committed to a high level regarding code quality and security. While the code inherited from Nessus certainly left room for improvements in these two aspects, the OpenVAS project is confident in its abilities to mitigate these shortcomings and to further improve code quality and security.

The OpenVAS project makes use of a number of automated tools in order to measure the quality of the codebase and to identify potential issues in the code itself. Tools used by the OpenVAS project include Flawfinder and RATS (Rough Auditing Tool for Security).

The latest results of the code quality tests is available at:

[1] Even though it might be tempting to use the trunk version to get the latest features, be aware that the trunk is under constant development and intended for developers. This means that the trunk version is not recommended for any production use.

http://www.openvas.org/code-quality.html

Please be aware that you should make yourself familiar with the tools used for generating these results before interpreting the absolute numbers. While every issue reported by these tools is evaluated by the OpenVAS developers, some of the issues reported do not have a significant impact on code quality and security.

10.4 Management of OpenVAS Change Requests

OpenVAS change requests describe proposed changes to one of the OpenVAS components. Though this is a formalized approach, this does not replace open discussion among interested developers on the mailing lists. In fact, these open discussions are the main source of change requests. This approach is intended to make the OpenVAS development process as transparent as possible to the OpenVAS developers as well as to the OpenVAS user community and other interested parties.

You can find the most current change requests at:

http://www.openvas.org/openvas-crs.html.

A complete change request consists of the following sections:

Status: A general description of the current status of this request. This could be something like “in discussion”, “agreed (voted +3) for release 1.4” or “Step 1 and 2 implemented”.

Purpose: A concise summary of the reasons for this change request.

References: Links to corresponding issue tracker entries or mailing list discussions.

Rationale: A more verbose explanation as to why this change request should be implemented.

Effects: The effects should this change request be implemented, describing changes to API, compatibility, user experience and so on.

Design and Implementation: Technical details regarding the implementation of this change request.

History: Date, name and description of changes to this change request in ChangeLog format.

Change requests can be created by anybody; although most change requests are created by OpenVAS developers, this is not in any way meant to exclude anyone from creating their own change request and submitting it to the openvas-discuss mailing list. Authors of change requests are encouraged to discuss their ideas for change requests on this mailing list or in the OpenVAS IRC channel (#openvas on irc.oftc.net) before compiling the actual request.

The OpenVAS developer community regularly votes on new change requests. Every OpenVAS developer can vote for or against a certain change request by voting “+1” (for), “+/-0” (somewhat for/against) or “-1” (against). After a number of days has passed, the votes are counted; a positive result means that the change request has been accepted and will be implemented in future versions of OpenVAS.

The voting process itself is not yet fully formalized and subject to change; ideas and suggestions are welcome.

10.5 Submitting Patches

If you have found (and fixed) a bug in the OpenVAS source code, implemented a change request or added a new feature, you are welcome to send a patch containing your changes to the OpenVAS developers.

To facilitate the inclusion of your changes into the code, please keep in mind the following guidelines:

- If possible, try to send a patch against the latest SVN revision.
- Send your patch as a unified context diff, in the format the GNU `diff` tool would produce when called with the -u3 parameter. If you are working on a revision you checked out from the SVN repository, the `svn diff` command will produce the correct output.
- Please detail your changes in the appropriate ChangeLog. This will help other people (especially other developers) to understand what you changed and why you changed it.
- Try to keep your patches atomic. That means that each patch should contain only one feature, bugfix or addition. Please do not submit a patch containing dozens of features as it is highly unlikely that the developers will want to add all your changes simultaneously. Your patch is far more likely to be accepted if you send in a number of small patches instead and leave it at the discretion of the developers to determine the best time to include your changes.

If you have followed these guidelines your patch should now be ready for submission to the OpenVAS project. The best way to do this is to send your patch to the OpenVAS developers mailing list at

`openvas-devel@wald.intevation.org`

with a short explanation as to what you did and why you did it.

10.6 Write-Access to Source Code Repository

Write access to the source code repository is granted by the project coordinators. If you want to able to commit changes to the source code repository, just drop a message on the openvas-devel mailing list. Usually you are asked to send your first changes as patches to the mailing list to demonstrate you know what you are doing. If these are accepted, it is also usual that you are immediately approved for write access to the source code repository.

Note that that changes to the code repository are watched by several developers. Low quality changes or uncoordinated high-impact-changes might get reverted immediately.

10.7 Maintaining ChangeLog

Any main module of OpenVAS maintains a “ChangeLog” file in its root directory. It is a GNU-style ChangeLog and syntax highlighting for your favorite editor should work out of the box.

Some rules for the ChangeLog file:

- Keep the GNU-style syntax for your new entries. Your syntax-highlighting editor will support you.
- Make atomic commits: Always include the updated ChangeLog file together with the changed files.
- List any changed file explicitly in the ChangeLog, even if there were many. See the ChangeLog file for examples how to write the change information.

Note: The ChangeLog is intentionally not created automatically. Before committing, developers are forced to reflect on the changes they did. This regularly leads to detection of suboptimal changes or identification of intermediate (try-out/debugging) changes that of course should not go to the main repository.

The developer’s ChangeLog is the base for writing the user’s CHANGES when a new release is prepared.

10.8 Source Code Style Guide

Because several styles have been mixed into OpenVAS over the last years (most inherited from the old Nessus times), the OpenVAS developers searched an agreement on source and documentation formats.

The result of this Change Request was to follow the formatting recommendations of the GNU Project

(http://www.gnu.org/prep/standards/standards.html#Formatting) and to use doxygen with in-code Javadoc style comments

(http://www.stack.nl/ dimitri/doxygen/docblocks.html).

This defines how both new and changed OpenVAS source code should ideally be formatted and documented.

The features in a nutshell

- File: Use a limit of 80 chars per line.
- Indention: Do not use tabulators but regular spaces.
- Indention: Indent with a step of 2 space characters.
- Function definitions: Do not use old K&R style.
- Function definitions: Add a space before opening brackets.
- Function definitions: Break function return types into a own line.
- Function documentation: Document each function.

Below, an example file with some comments can be seen (not meant to be compiled and not available in the repository).

```
/* OpenVAS-Client
 * $Id$
 * Description: Proposed formating example (altered slad_install.c)
 *
 * Authors:
 * Felix Wolfsteller <felix.wolfsteller@intevation.de>
 *
 * Copyright:
 * Copyright (C) 2008 by Intevation GmbH
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2,
 * or, at your option, any later version as published by the Free
 * Software Foundation
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 * In addition, as a special exception, you have
```

```
 * permission to link the code of this program with the OpenSSL
 * library (or with modified versions of OpenSSL that use the same
 * license as OpenSSL), and distribute linked combinations including
 * the two. You must obey the GNU General Public License in all
 * respects for all of the code used other than OpenSSL. If you
 * modify this file, you may extend this exception to your version
 * of the file, but you are not obligated to do so. If you do not
 * wish to do so, delete this exception statement from your version.
 */


#include <unistd.h>
#include <gtk/gtk.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "nessus_i18n.h"
#include "module.h"


/**
 * Defines follow the includes which follow the header, are capitalized and can
 * be documented.\n
 * Documentation is done in the source file, not the header file.
 */
#define SOMEDEFINE "fine"


/**
 * Static functions do not neccessarily begin with the module name.
 * @param str Parameters can be described like this.
 */
static void
statfunc (gchar *str)
{
  /* Indentation by 2 blanks, single lines should not exceed 80 chars */
  GtkWidget* m = gtk_message_dialog_new (GTK_WINDOW(NULL),
                                         (GTK_DIALOG_MODAL
                                          | GTK_DIALOG_DESTROY_WITH_PARENT),
                                         GTK_MESSAGE_ERROR, GTK_BUTTONS_OK,
                                         str);
  /* (If 80 chars per line do not suffice, try to break it up nicely) */
  gtk_dialog_run (GTK_DIALOG (m));
  gtk_object_destroy (GTK_OBJECT (m));
}


/**
 * @brief For an overview, I can describe what happens here in one sentence.
 *
 * Non-static functions begin with the name of the module + underscore.
 * @param win Widget to work on.
```

```
 * @param data Data to work with.
 * @return String that has to be freed.
 */
char*
module_func (GtkWidget* win, gpointer data)
{
  char buf[1024];
  pid_t p;
  FILE* f;
  /* Structuring by newlines is fine. */

  f = popen (SLADINSTALLER " --ping", "r");

  /* Conditional and loop intendation same as for functions. */
  if (!f)
    {
      statfunc (_("Could not check SLAD installer."));
      return;
    }
  else /* f != NULL */
    {
      statfunc (_("An else"));
    }
  fgets (buf, sizeof (buf), f);
  pclose (f);

#ifndef CYGWIN
  /* Compiled with flag -mwindows, sladinstaller
   * does not answer on stdout.
   * Therefore, this test is only performed on
   * non-CYGWIN systems and simply dropped for
   * CYGWIN. */
  if (strcmp (buf, "pong\n"))
    {
      statfunc (_("Could not execute SLAD installer."));
      return;
    }
#endif /* CYGWIN */

  // Single-line intermediate comments can also start by double slashes.
  p = fork ();
}
```

11 OpenVAS Transfer Protocol (OTP)

The client and the server module in an OpenVAS installation communicate through the OpenVAS Transfer Protocol (OTP). Earlier versions of OpenVAS have used the Nessus Transport Protocol (NTP) inherited from Nessus, but in order to address shortcomings of NTP and to facilitate further improvements in the OpenVAS modules it became necessary to make changes to the protocol. Since NTP was specified by the Nessus project and changes to NTP by the Nessus project are to be expected, a decision was made to switch to a new protocol to avoid collisions with future protocol specifications by the Nessus project and to avoid confusion with other well-established protocols. The initial specification of OTP is very close to the NTP implementation in the last versions available under the GNU General Public License (GPL).

11.1 Changes from NTP 1.2 to OTP 1.0

This section describes the changes between NTP 1.2 and OTP 1.0. If you already have some experience with NTP, this section should help you to spot the main differences between the two protocols.

Plug-in upload Section 10 of the NTP Extensions describes the `ATTACHED_PLUGIN` message type. Using this message type, it was possible for a client to upload a plug-in to a server. Due to security considerations described in the OpenVAS change request #4, this message type has been removed from the protocol.

Version information The undocumented `NESSUS_VERSION` message type has been replaced with the `OPENVAS_VERSION` message type. When an `OPENVAS_VERSION` message is issued by the client, the server is expected to respond with a message containing the current server version.

New message types In addition to the existing message types `HOLE`, `INFO` and `NOTE` two new message types have been added to the protocol: `DEBUG` and `LOG`. Their purpose is to give clients the possibility to display log and debug messages raised by plugins and allow the user to control the verbosity of the messages displayed.

Detached scans This functionality has been dropped due to design decisions. This means the following commands have been removed from the protocol: `DETACHED_SESSIONS_LIST` and `DETACHED_STOP`. The following preferences have been removed from the protocol as well: detached_scan, continuous_scan, delay_between_scan_loops, detached_scan_email_address.

Plugin order information The server command `PLUGINS_ORDER` was defined for NTP 1.2 but not implemented in the server. This command has been removed from the protocol.

Starting a scan NTP offered two ways of starting a scan, `NEW_ATTACK` and `LONG_ATTACK`. The latter allowed arbitrary long list of targets while the first was limited to 4000 bytes. The OpenVAS-Client (and so did NessusClient) used only `LONG_ATTACK` anyway.

Reporting preferences errors NTP allowed the reporting of preferences errors with the `PREFERENCES_ERRORS` message type. Since this message type was never properly implemented in either server or client and due to design decisions, this message type has been removed from the protocol.

Protocol extensions These protocol extensions have been made standard in the OTP protocol: "timestamps", "dependencies", "plugins_version", "plugins_cve_id", "plugins_bugtraq_id" and "plugins_xrefs".

11.2 General Aspects of OTP

The OpenVAS Transfer Protocol is text-based, human readable and line-oriented. Each line consists of fields separated by " <|> ". The first field indicates whether it is a command sent by the client or by the server ("`CLIENT`" vs. "`SERVER`").

11.3 Protocol Initialization

The client will start a protocol session by sending a version string specifying the requested protocol version to the server. Upon receiving this string, the server may answer with the same version string if it supports the requested protocol version or terminate the connection.

Syntax:

```
< OTP/1.0 >
User : user_name
Password : user_password
```

11.4 Protocol Commands

11.4.1 ATTACHED_FILE

Description: This command corresponds to the plugin preferences type "file". It follows the command `PREFERENCES` to upload the specified files from client to server.

Syntax:

```
CLIENT <|> ATTACHED_FILE
name: file_name
content: octet/stream
bytes: file_length
file_content
```

where

file_name The path and name of the file. It is an identifier to reference the file in the plugin preferences.

file_length the number of bytes that will follow after the newline

file_content the actual file as a byte stream.

11.4.2 BYE

Description: This command is used by the server to indicate that a scan session has finished.

The client is expected to acknowledge this command.

Syntax:

```
SERVER <|> BYE <|> BYE <|> SERVER
CLIENT <|> BYE <|> ACK
```

11.4.3 CERTIFICATES

Description: This command is used by the client to request certificate information and by the server to send those. Included are the certificate owners name, the trust level and the public key.

Syntax:

```
CLIENT <|> CERTIFICATES <|> CLIENT
SERVER <|> CERTIFICATES
fingerprint <|> owner_name <|> trust_level <|> length_in_bytes <|> pubkey
<|> SERVER
```

where

fingerprint is a 48 bytes field containing the fingerprint of the certificate.

owner_name denotes the owner name.

trust_level is either “trusted” or “notrust”.

length_in_bytes contains the length of the public key in bytes.

pubkey is the ascii-armored public key itself, where newlines have been replaced by semicolons.

11.4.4 COMPLETE_LIST

Description: This command can be used by the client to request the complete list of plugins from the server.

It usually follows the `PLUGINS_MD5` commands of the server in case the server side md5sum is not equal to the md5sum of the client side cached NVTs. Alternatively, the client can use the command `SEND_PLUGINS_MD5`.

The server will answer with command `PLUGIN_LIST`.

Syntax:

```
CLIENT <|> COMPLETE_LIST <|> CLIENT
```

11.4.5 DEBUG

Description: With this command the server reports an identified problem of class “debug”. The “general” version is applied if no port relates to the note.

Syntax:

```
SERVER <|> DEBUG <|> host <|> service_name (port_number/protocol_type) <|>
description <|> oid <|> SERVER

SERVER <|> DEBUG <|> host <|> general <|> description <|> oid <|> SERVER
```

where

host the target system

service_name the name of the service (like in /etc/services)

port_number the port number the problem relates to.

protocol_type “tcp” or “udp”.

description the problem description where newlines have been replaced by semicolons.

oid the OID of the NVT that identified the problem.

11.4.6 ERROR

Description: In case of problems the server sends an error message with this command. In case of unrecoverable problems, the server will then close the connection with the `BYE` command.

Syntax:

```
SERVER <|> ERROR <|> error description <|> SERVER
```

11.4.7 FINISHED

Description: The server will send this information each time when a scan of a single host is finished. This will only be done if requested by the client via setting the preferences option “ntp_opt_show_end”.

Syntax:

```
SERVER <|> FINISHED <|> host <|> SERVER
```

11.4.8 GO ON

Description: This command can be used by the client to confirm that the plugin information has been received and to signal to the server that it may proceed. It usually follows the `PLUGINS_MD5` commands of the server in case the server side md5sum is equal to the md5sum of the client side cached NVTs. The server will answer with command `PREFERENCES` and communication will continue.

Syntax:

```
CLIENT <|> GO ON <|> CLIENT
```

11.4.9 HOLE

Description: With this command the server reports an identified problem of class “security hole”. The “general” version is applied if no port relates to the hole.

Syntax:

```
SERVER <|> HOLE <|> host <|> service_name (port_number/protocol_type) <|>
description <|> oid <|> SERVER

SERVER <|> HOLE <|> host <|> general <|> description <|> oid <|> SERVER
```

where

host the target system

service_name the name of the service (like in /etc/services)

port_number the port number the problem relates to.

protocol_type “tcp” or “udp”.

description the problem description where newlines have been replaced by semicolons.

oid the OID of the NVT that identified the problem.

11.4.10 INFO

Description: With this command the server reports an identified problem of class “security info”. The “general” version is applied if no port relates to the info.

Syntax:

```
SERVER <|> INFO <|> host <|> service_name (port_number/protocol_type) <|>
description <|> oid <|> SERVER

SERVER <|> INFO <|> host <|> general <|> description <|> oid <|> SERVER
```

where

host the target system

service_name the name of the service (like in /etc/services)

port_number the port number the problem relates to.

protocol_type “tcp” or “udp”.

description the problem description where newlines have been replaced by semicolons.

oid the OID of the NVT that identified the problem.

11.4.11 LOG

Description: With this command the server reports an identified problem of class “log”. The “general” version is applied if no port relates to the note.

Syntax:

```
SERVER <|> LOG <|> host <|> service_name (port_number/protocol_type) <|>
description <|> oid <|> SERVER

SERVER <|> LOG <|> host <|> general <|> description <|> oid <|> SERVER
```

where

host the target system

service_name the name of the service (like in /etc/services)

port_number the port number the problem relates to.

protocol_type “tcp” or “udp”.

description the problem description where newlines have been replaced by semicolons.

oid the OID of the NVT that identified the problem.

11.4.12 LONG_ATTACK

Description: With this command the client requests the server to attack target system(s) “hosts”. “hosts” is one or many (comma-separated) IP or FQDN.

“length” is the number of bytes of “hosts”. In case this does not match, the server will close the connection.

Before the client sends `LONG_ATTACK`, the commands `PREFERENCES` and `RULES` should be applied.

Syntax:

```
CLIENT <|> LONG_ATTACK
length
hosts
```

11.4.13 NOTE

Description: With this command the server reports an identified problem of class “security note”. The “general” version is applied if no port relates to the note.

Syntax:

```
SERVER <|> NOTE <|> host <|> service_name (port_number/protocol_type) <|>
description <|> oid <|> SERVER

SERVER <|> NOTE <|> host <|> general <|> description <|> oid <|> SERVER
```

where

host the target system

service_name the name of the service (like in /etc/services)

port_number the port number the problem relates to.

protocol_type “tcp” or “udp”.

description the problem description where newlines have been replaced by semicolons.

oid the OID of the NVT that identified the problem.

11.4.14 OPENVAS_VERSION

Description: With this command the client asks the server to send its version.

The server will answer as shown in the syntax.

Syntax:

```
CLIENT <|> OPENVAS_VERSION <|> CLIENT

SERVER <|> OPENVAS_VERSION <|> version <|> SERVER
```

11.4.15 PLUGINS_DEPENDENCIES

Description: The `PLUGINS_DEPENDENCIES` message is sent after the `RULES` messages.

Syntax:

```
SERVER <|> PLUGINS_DEPENDENCIES
nvt_name1 <|> dependency1 <|> dependency2 <|> ... <|>
nvt_name2 <|> dependency1 <|> dependency2 <|> ... <|>
...
<|> SERVER
```

11.4.16 PLUGINS_MD5

Description: Attention: This command occurs in two ways.

1. This command can be used instead of the `PLUGIN_LIST` command. “md5sum” is the MD5 sum over all NVTs.
2. This command follows the `SEND_PLUGINS_MD5` command of the client and delivers the md5sums for each NVT.

Syntax:

1. `SERVER <|> PLUGINS_MD5 <|> md5sum <|> SERVER`
2. ```
   SERVER <|> PLUGINS_MD5
   nvt_oid1 <|> md5sum1
   nvt_oid2 <|> md5sum2
   ...
   <|> SERVER
   ```

11.4.17 PLUGIN_INFO

Description: This command is issued by the client to request information of the NVT specified by its oid.
   ```

Syntax:

```
CLIENT <|> PLUGIN_INFO <|> oid <|> CLIENT
```

The server answers with this line (analogous to `PLUGIN_LIST` command):

```
oid <|> name <|> category <|> copyright <|> description <|> summary <|>
family <|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
```

The last field, fprs is a comma-separated list of fingerprints of signatures, if any.

In case no plugin with OID=oid is found, the server will not answer at all.

For cve_id, bugtraq_id, xrefs and fprs symbolic values (NOCVE, NOBID, NOXREFS, NOSIGNKEYS) are sent, if no cve_id, bugtrac_id etc. is known.

11.4.18 PLUGIN_LIST

Description: With this command the server sends detailed information about the available NVTs.

The server will send `PREFERENCES` and `RULES` right after this command.

The client might request individual NVT information via the `PLUGIN_INFO` command.

Syntax:

```
SERVER <|> PLUGIN_LIST <|>
oid <|> name <|> category <|> copyright <|> description <|> summary <|> family
<|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
oid <|> name <|> category <|> copyright <|> description <|> summary <|> family
<|> plugin_version <|> cve_id <|> bugtraq_id <|> xrefs <|> fprs
...
<|> SERVER
```

In this case, fprs is a comma-separated list of fingerprints of signatures, if any. For cve_id, bugtraq_id, xrefs and fprs symbolic values (NOCVE, NOBID, NOXREFS, NOSIGNKEYS) are sent, if no cve_id, bugtrac_id etc. is known.

11.4.19 PORT

Description: With this command the server reports on open port “port_number” on target system “host”.

Syntax:

```
SERVER <|> PORT <|> host <|> port_number <|> SERVER
```

11.4.20 PREFERENCES

Description: With this command the values for the preferences are communicated. The server uses the command to inform about defaults, the client uses the command to send the user selections.

Note that besides some general preferences, the syntax definition describes also per-NVT preferences and its special way of applying these.

Available preferences:

ntp_save_sessions If set to “yes”, the server will support server-side saving of scan sessions and the following commands will be available: `SESSIONS_LIST`, `SESSION_DELETE` and `SESSION_RESTORE`.

save_session If set to “yes”, the server will save the scan as a session.

save_empty_sessions Only considered if save_session is set to “yes”. If set to “yes” even empty scans will be saved as a session.

max_threads

test_file

ping_hosts

reverse_lookup

outside_firewall

host_expansion

port_range

max_hosts

save_knowledge_base Activates KB saving when set to “yes”

only_test_hosts_whose_kb_we_have Only scans host for which the KB is filled when set to “yes”.

only_test_hosts_whose_kb_we_dont_have Only scans host for which the KB is empty when set to “yes”.

kb_restore Restore the KB contents for tested hosts when when set to “yes”.

kb_dont_replay_scanners Don’t run scanners in case kb_restore is set to “yes” and there is contents in the KB when set to “yes”.

kb_dont_replay_info_gathering Don’t run gatherers in case kb_restore is set to “yes” and there is contents in the KB when set to “yes”.

kb_dont_replay_attacks Don’t run attack scripts in case kb_restore is set to “yes” and there is contents in the KB when set to “yes”.

kb_dont_replay_denials Don’t run DoS attack scripts in case kb_restore is set to “yes” and there is contents in the KB when set to “yes”.

kb_max_age This sets the maximum age (in seconds) of a KB until it gets disregarded.

timeout.<nvt_id> = <timeout> Set the timeout <timeout> for NVT <nvt_id>. Timeout of “-1” means no specific timeout.

Only sent by CLIENT:

plugin_set empty means all NVTs

ntp_opt_show_end Tell server to send `FINISHED` messages

ntp_keep_communication_alive Tell server to keep the connection even after a scan was finished.

ntp_short_status Tell server send shorter `STATUS` message in order to save bandwidth.

Syntax:

```
SERVER <|> PREFERENCES <|>
pref_name <|> value
pref_name <|> value
pref_name <|> value
...
<|> SERVER

CLIENT <|> PREFERENCES <|>
pref_name <|> value
pref_name <|> value
pref_name <|> value
...
<|> CLIENT
```

For preference of individual NVTs these lines can occur inside the list:

```
nvt_name[pref_type]:pref_name <|> value
```

where

nvt_name This references the NVT for which the preferences are set

pref_type Defines the variable type of the preference which ultimately determines the widget type in the client GUI.

pref_name This references the Preference and at the same time is used as the visible string for the user in the GUI.

value The default value for this preference when sent by SERVER, the user selected value if send by CLIENT

and pref_type is one of these:

checkbox value is “yes” or “no”

entry value is a text string

password value is a text string but should not be shown in GUI or in cleartext in local files

radio value is a list of semicolon-separated options when sent by SERVER and only the user-selected option name when sent by CLIENT

file value is “<none>” when sent by SERVER and a file path when sent by CLIENT. The client has to submit the file under the very same path name using the command `ATTACHED_FILE`.

11.4.21 RULES

Description: Rules define restrictions for target systems. Client-side rules self-restrict target host patterns, server-side rules are just for information to the client. These rule sets are independent of each other.

Syntax:

```
SERVER <|> RULES <|>
rule_1;
rule_2;
rule_3;
...
<|> SERVER

CLIENT <|> RULES <|>
rule_1;
rule_2;
rule_3;
...
<|> CLIENT
```

11.4.22 SEND_PLUGINS_MD5

Description: This command usually follows the `PLUGINS_MD5` commands of the server in case the server side md5sum is not equal to the md5sum of the client side cached NVTs. Alternatively, the client can use the command `COMPLETE_LIST`.

The server will answer with command `PLUGINS_MD5`.

Syntax:

```
CLIENT <|> SEND_PLUGINS_MD5 <|> CLIENT
```

11.4.23 SESSIONS_LIST

Description: The CLIENT requests with this command the list of sessions stored on the server side for the logged in user.

The SERVER will answer with the same command and provide the list of sessions. The session names are derived from time stamps. The hosts are the targets applied for the respective session. The hosts are just there to help identify the sessions. It is cut after 4000 bytes of length.

Syntax:

```
CLIENT <|> SESSIONS_LIST <|> CLIENT

SERVER <|> SESSIONS_LIST
session_name1 hosts1
session_name2 hosts2
...
<|> SERVER
```

11.4.24 SESSION_DELETE

Description: With this command the client deletes the session identified with “session_name” from the server-side storage. The server will not answer in case of success, else it will answer with an `ERROR` command.

Syntax:

```
CLIENT <|> SESSION_DELETE <|> session_name <|> CLIENT
```

11.4.25 SESSION_RESTORE

Description: With this command the client tells the server to pick up again the session identified with "session_name". The server will act as if a LONG_ATTACK command has issued and will send all results that were collected so far for this session immediately (and naturally rapidly) and then continue the scan where it stopped.

Syntax:

```
CLIENT <|> SESSION_RESTORE <|> session_name <|> CLIENT
```

11.4.26 STATUS

Description: With this command, the server informs the client about the progress of the scan for target system "host". "attack_state" is either "portscan" or "attack". (or just "p" and "a" in case the client has set preferences option "ntp_short_status"). "current" is the currently processed port and "max" the last port number to be tested.

Syntax:

```
SERVER <|> STATUS <|> host <|> attack_state <|> current/max <|> SERVER
```

In case the client has set "ntp_short_status".

```
SERVER <|> STATUS <|> attack_state:host:current:max <|> SERVER
```

11.4.27 STOP_ATTACK

Description: With this command, the client tells the server to stop scanning target "host".

Syntax:

```
CLIENT <|> STOP_ATTACK <|> host <|> CLIENT
```

11.4.28 STOP_WHOLE_TEST

Description: With this command the client tells to stop the currently running test.

Syntax:

```
CLIENT <|> STOP_WHOLE_TEST <|> CLIENT
```

11.4.29 TIME

Description: The `TIME` messages are sent by the server to inform about the duration of scanning a host and of the whole scan.

Syntax: After completion of scanning a target host the server sends:

```
SERVER <|> TIME <|> HOST_START <|> host <|> time_string <|> SERVER
SERVER <|> TIME <|> HOST_END <|> host <|> time_string <|> SERVER
```

or, in case `STOP\_ATTACK` was issued by the client:

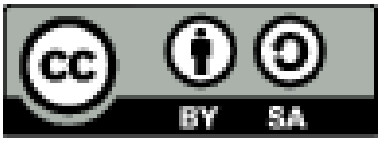
```
SERVER <|> TIME <|> HOST_START <|> host <|> time_string <|> SERVER
SERVER <|> TIME <|> HOST_INTERRUPTED <|> host <|> time_string <|> SERVER
```

After completion of the whole scan the server sends:

```
SERVER <|> TIME <|> SCAN_START <|> time_string <|> SERVER
SERVER <|> TIME <|> SCAN_END <|> time_string  <|> SERVER
```

where time_string is of the form “Wed Jun 30 21:49:08 1993”.

12 Document License: CC by SA



Creative Commons Attribution-ShareAlike 3.0 Unported

You are free to copy, distribute and transmit the work under the following conditions:

Attribution. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of the above conditions can be waived if you get permission from the copyright holder.

Nothing in this license impairs or restricts the author's moral rights.
To view the full licensing agreement, visit

http://creativecommons.org/licenses/by-sa/3.0/

or mail to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.