NessusClient User's Manual

$Date: 2006/05/19 09:58:15 $

Renaud Deraison <renaud@nessus.org>

Jan-Oliver Wagner <jan@intevation.de>

Contents

About This Document

This document describe how to use a Nessus Server via the GTK+ GUI "NessusClient". This may not cover the whole functionality of the Nessus Server nor does it claim to be complete. However, the text should be up-to-date and refers to the upcoming NessusClient 1.2 release.

1 General Overview

The "Nessus" Project provides powerful, up-to-date and easy to use remote security scanner as Free Software under GNU General Public License (GNU GPL). Nessus will allow you to audit remotely a given network and determine whether bad guys (aka 'crackers') may break into it, or misuse it in some way.

Intelligent Scanning Unlike many other security scanners, Nessus does not take anything for granted. That is, it will not consider that a given service is running on a fixed port - that is, if you run your web server on port 1234, Nessus will detect it and test its security. It will also not determine a security vulnerability is present by just regarding the version number of the remote service, but will really attempt to exploit the vulnerability.

Modular Architecture The client/server arquitecture allows you flexibility to deploy the scanner (server) and the GUI (client) in multiple configurations reducing management costs (one server can be used by multiple clients) Some other features of the Nessus Security Scanner are:

CVE compatible Each plugin links to CVE for administrators to retrieve further information on published vulnerabilities. They also includes references to **CERT, Bugtraq**, and vendor security alerts.

Plug-in architecture Each security test is written as an external plugin. This way, you can easily add your own tests without having to read the code of the Nessus Server engine `nessusd`. The complete list of the Nessus plugins is available at http://cgi.nessus.org/plugins.

NASL The Nessus Security Scanner includes NASL, (Nessus Attack Scripting Language) a language designed to write security test easily and quickly. (security checks can also be written in C)

Up-to-date security vulnerability database We mostly focus on the development of security checks for recent security holes. Our security checks database is updated on a daily basis, and all the newest security checks are available at http://www.nessus.org/scripts.php and on the FTP servers and mirrors.

Can test an unlimited amount of hosts at the same time Depending of the power of the station you run the Nessus Server onto, you can test two, ten or forty hosts at the same time.

Smart service recognition Nessus does not believe that the target hosts will respect the IANA assigned port numbers. This means that it will recognize a FTP server running on a non-standard port (31337 say), or a web server running on port 8080.

Multiples services Imagine that you run two web servers (or more) on your host, one on port 80 and another on port 8080. When it will come to testing their security, Nessus will test both of them.

Tests cooperation The security tests performed by Nessus cooperate so that nothing useless is made. If your FTP server does not offer anonymous logins, then anonymous-related security checks will not be performed.

Complete reports Nessus will not only tell you what's wrong on your network, but will, most of the time, tell you how to prevent crackers from exploiting the security holes found and will give you the risk level of each problem found (from Low to Very High)

Exportable reports The Unix client can export Nessus reports as XML, HTML, text, LaTeX, PDF and an easy-to-parse file format.

Full SSL support Nessus has the ability to test SSLized services such as https, smtps, imaps, and more. You can even supply Nessus with a certificate so that it can integrates into a PKI-fied environement.

Smart plugins (optional) Nessus will determine which plugins should or should not be launched against the remote host (for instance, this prevents the testing of Sendmail vulnerabilities against Postfix). This option is called "optimizations".

Non-destructive (optional) If you don't want to take the risk to bring down services on your network, you can enable the "safe checks" option of Nessus, which will make Nessus rely on banners rather than exploiting real flaws to determine if a vulnerability is present.

Independent developers The Nessus developers are independent from the rest of the world, so we will not hide a security vulnerability in the program XYZ because we have a contract with them.

Easy-to-reach developers You feel that there is a missing feature? Just contact ushttp://www.nessus.org/contact/ or use the Nessus mailing lists. We reply and implement what makes sense.

Open bug tracking system. Found a bug? Report it here: http://bugs.nessus.org.

Huge number of tests There are more than 7000 nessus tests (and constantly growing!) which are divided into 23 different families:

- Backdoors
- CGI abuses
- CISCO
- Denial of Service
- Finger abuses
- Firewalls
- FTP
- Gain a shell remotely
- Gain root remotely
- General
- Misc.
- Netware

- NIS
- Port scanners
- Remote file access
- RPC
- Settings
- SMTP problems
- SNMP
- Untested
- Useless services
- Windows

2 Installation

Note that only the source is what the Nessus developer team primarily tests in depth. Binary installation packages are likely to be available for most standard operating systems but not necessarily be tested by the Nessus development team.

Operating systems not listed here may offer well-maintained packages as well.

2.1 From Source

The usual way of installing from scratch is to get a current CVS snapshot or a source tar-ball of the module "NessusClient" and perform the usual sequence "./configure ; make ; make install". Please also as usual read the files about installation and configuration options in the packages first.

2.2 Debian GNU/Linux

Please refer to the Debian documentation on how to install standard Debian packages. Debian actively support Nessus packages.

2.2.1 Debian "Woody" 3.0

The official version of Nessus for Woody is 1.0.10. This is by far outdated, you should neither use Nessus Server nor Client as shipped with Woody!

The build- and run-dependencies of NessusClient are conservative so that you do not need to backport any other package for backporting NessusClient. Thus it might work to create a so-called "backport" for Woody from a newer Debian source package of NessusClient.

2.2.2 Debian “Sarge” 3.1

The official version of Nessus for Sarge is 2.2.3. NessusClient is compatible with this Nessus Server, but it is recommended to update the Nessus Server to get the newest bug-fixes.

The build- and run-dependencies of NessusClient are conservative so that you do not need to backport any other package for backporting NessusClient. Thus it might work to create a so-called “backport” for Sarge from a newer Debian source package of NessusClient.

2.2.3 Debian “Etch”

At time of writing, Etch is the current Testing version of Debian and thus in flux. At least Nessus 2.2.7 is included. It is expected that NessusClient packages will appear, once NessusClient is released.

2.3 RedHat Linux

In the download section of www.nessus.org you might also find RPM packages for RedHat and Fedora, created by Tenable Network Security.

2.3.1 Fedora Core 4 and 5

This distribution actively maintains Nessus as extra packages. In case you need a newer version than shipped, you may find backports. Backports for RPM packages maybe found via rpmfind.net or similar services. At time of writing, NessusClient is not yet officially part of Fedora or Fedora extras.

The build- and run-dependencies of NessusClient are conservative so that you do not need to backport any other package for backporting NessusClient. It might even work to install newer packages directly.

2.4 SUSE Linux

In the download section of www.nessus.org you might also find RPM packages for SUSE, created by Tenable Network Security.

2.4.1 SUSE Linux 9.2

Nessus is part of this distribution with the latest 2.0 release. It is highly recommended you update Nessus Server to latest 2.2 release. Backports for RPM packages maybe found via rpmfind.net or similar services.

For NessusClient there is currently no official package for SUSE.

The build- and run-dependencies of NessusClient are conservative so that you do not need to backport any other package for backporting NessusClient. It might even work to install newer packages directly.

2.5 MS Windows

2.5.1 Windows XP SP2

For Windows a regular setup package is available for NessusClient, usually called *NessusClientN.N.N-setup-LL.exe*. Either execute it directly or use this in the Windows Manager for Software. This is also the tool where you can easily and cleanly deinstall NessusClient again. N.N.N defines the version of Nessus, e.g. 1.0.0. LL defines the language where “en” means english, “de” german and “sv” swedish version.

2.6 Migrating Nessus GTK Client from version 2.2

NessusClient has a new way to store session data locally compared to the GTK client that was included in Nessus 2.2. In fact, the whole concept of tasks, scopes and reports is new since then.

The main difference is that Nessus GTK Clients upto 2.2 consider a single file “.nessusrc” in the users home directory. The newer versions use a directory structure with the main directory “.nessus” in the users home directory.

Nonetheless, there will be no conflict when running the new version because the new version uses the .nessusrc file as the reference for the global setting (see below). In other words, the configuration of your Nessus GTK Client 2.2 will be the default configuration for NessusClient.

It is even possible to run a Nessus GTK Client 2.2 after a newer version has been launched. Version 2.2 will simply use the same file “.nessusrc”, the new version uses for the global settings.

3 The First Report

This section is intended to lead a novice user quickly through a first security scan with Nessus. After a successful first scan it is recommended to proceed with the chapter on the general usage of NessusClient.

3.1 Invoking NessusClient

On any unixoid system you can simply type “NessusClient” in a command shell. Of course you should have X running.

For Debian GNU/Linux, SUSE Linux, ReadHat/Fedora and MS Windows you will additionally find a menu entry for NessusClient if installed with one of the above explained install packages.

3.2 The Nessus Scan Assistant

If you are new to Nessus and NessusClient you may want to use the Nessus Scan Assistant to perform you first scan. The intention of this assistant is summarize the core parameters you need to enter for a scan and to be self-explanatory.

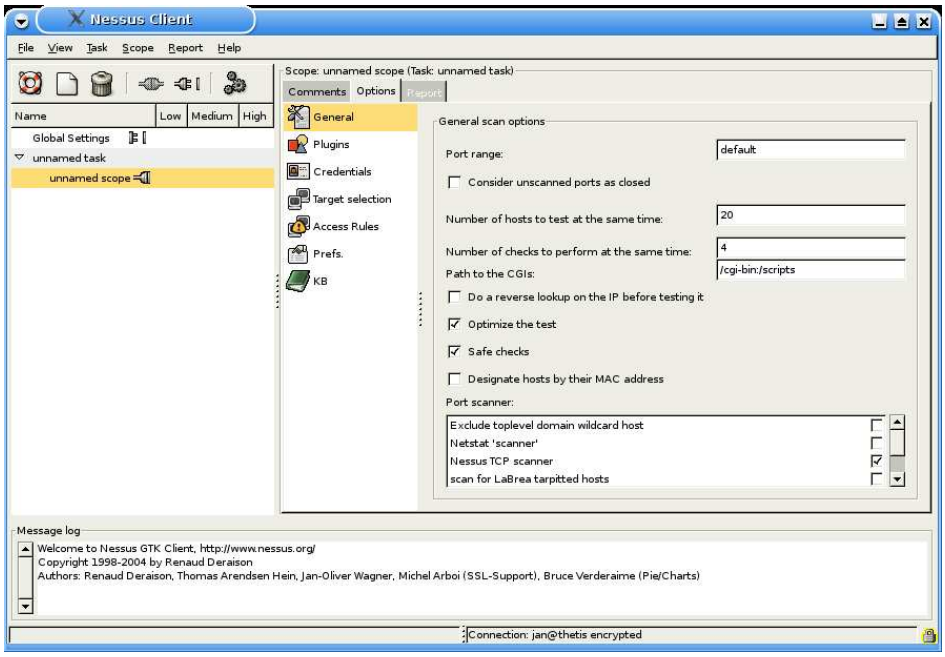
However, **you should note** that the scan performed through the Assistant will be based on the explicitely saved global settings. If you have switched on some dangerous options already in the global settings, these will be inherited!

4 General Usage of NessusClient

This section is intended to explain all general elements of NessusClient and how to use it in a standard way. A later chapter will then address more specific features suitable for advanced users.

4.1 The Main Window

The main window of NessusClient is separated into two major sections: On the left-hand side 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. That is the place where a security scan is to be configured, commented upon and the result reviewed.



At the first start of NessusClient, you will see only one entry in the list: Global Settings. These settings are defaults coming with NessusClient. They are not covering a specific selection of plugins since a connection to a Nessus Server is required for that. You can establish a connection with a server and specify the global defaults for a plugin selection. You should then save your preferred defaults via menu item “Save Global Settings” of menu “File”.

4.1.1 Tasks

Tasks are intended to cover all activities of a major topic. E.g., a task could be “Test the maschines 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. It just contains a number of scopes.

Possible operations for tasks are:

New Adds a new task entitled “unnamed”.

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

Remove This means the removal of all associated scopes and thus the removal action prompts for an additional confirmation.

4.1.2 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 sytem” 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 hint regarding the respective scope.

The scope is associated with a full set of options for the security scan. Creating a new scope, the general preferences are copied. The scan options are explained in detail in a later chapter. However, for each scope a connection to a Nessus Server can be established. Has this been done, the actual selection of the plugins being a part of the options can be performed (a plug icon right-hand of the title indicates the active connection). The reason is that each Nessus Server may contain its own set of plugins and thus with establishing a connection, NessusClient retrieves the list of available plugins.

Next, a scope may contain a number of reports. Whenever a scope is successfully executed, the resulting report is added in its list of reports. Also, importing a report from a file or from a Nessus 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 NessusClient 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 issueing the currently connected Nessus 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 explicitely saved global settings are taken as defaults. If you changed them inside NessusClient without saving them, they will have no effect.

Rename Allows to edit the title right 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 an additional 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 you 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 nessusrc type). Note that only the parameter sets are stored but not the reports. See above the description of “Open” for more hints.

4.1.3 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 NessusClient, additionally a comment and, if available, the scan options leading to the report, are stored. These additional information are not contained in the plain Nessus report files and thus get 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, if associated, 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”). Older releases of Nessus used the format NSR (files suffixed “.nsr”). It is recommended to use NBE format only. The file selection dialog allows to select the desired report file. A 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.

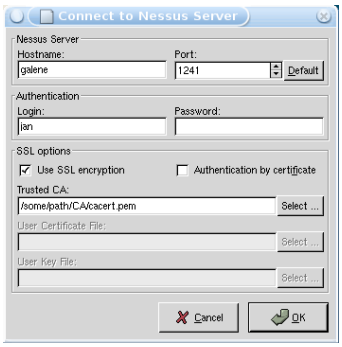
Export Allows to export the currently selected report either in a re-importable format (NBE or the deprecated NSR) or in a format for further processing or presentation (XML, old deprecated XML, HTML, HTML with Pies and Graphs, LaTeX, ASCII Text and PDF). It is recommended to use NBE if re-importing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

Print Selecting the Print command will issue the creation of a PDF report and run a PDF Viewer installed on your system. A number of well-known PDF Viewers is tried. If you think you have one installed but it still does not work, please check whether the executable file is covered by the search paths.

4.2 Authentication

NessusClient needs to connect to a Nessus Server in order to retrieve the available plugins and to actually execute a security scan.

NessusClient can handle mutiple connections to different servers. Each scope has a connection of its own. Additionally, the Global Settings can be connected to a Nessus Server to define default plugin selections and plugin parameters. Note that only explicitly saved Global Settings are taken 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 Nessus Server.

More information on the connection status is shown in the statusbar at the bottom of the main window. There, the actual connection information is displayed, ie. “Connection: username@host.test.example”. At bottom right there is an icon indicating whether the connection is encrypted or not, just like most web browsers indicate this.

The connection dialog allows to specify the following data for establishing a connection to a Nessus Server:

Host The domain name or IP of the server where a Nessus Server is running.

Port The port where the Nessus Server waits for connections. Older Nessus Servers used 3001, but the official port now is 1241. With the default button you can always return to this default quickly.

Login Your username at the selected host. To use a Nessus Server you have to have an account for the Nessus Server. The administrator of the server should create one for you.

Password The password for your Nessus Server account. Don't use the same one you have for your other user accounts and especially if you ever use unencrypted connections.

SSL Options

Use SSL encryption: For the authentication there are two basic methods, via login/password combination or via certificate (with or without password). In any case you should switch on SSL encryption to not have the password transferred as cleartext. However, if you are in an evironment where you don't get SSL to work, switching it off is the work around to run Nessus at last.

Trusted CA: This certificate defines a certificate authority (CA) you trust. With this certificate you will check that you are connecting to a trusted Nessus Server. This is checked if you have the Paranoia Level set to 2 or 3, it is not checked with a Paranoia Level of 1. Note, that you can set the Paranoia Level by hand in the nessusrc files or when first connecting to a Nessus Server where you are asked explictly.

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

If you are running Nessus Client from a more remote maschine, you need to have a copy of the CA certificate and place it to some arbitrary place in your home directory.

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 Nessus Server using the corresponding 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 textentry.

4.3 Scan Options

This section explains the most important configuration options for a security scan. More special option (access rules, knowledge base detached scans) are explained later in the chapter on special features.

4.3.1 General

This page covers all the general scan options. See the screenshot for the main window above.

Port range Ports that will be scanned by Nessus 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 Nessus services file.

Consider unscanned ports as closed To save scanning time, you may ask Nessus 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 Nessus Server from sending packets to ports you did not specify. If this option is disabled, then Nessus Server will consider ports whose state it does not know as open.

Number of hosts to test at the same time Maximal number of hosts that the Nessus Server will test at the same time. Be aware that the Nessus Server will spawn max_hosts 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 Nessus 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:/~deraison/cgi”.

Do a reverse lookup of the IP before testing it If this option is set, Nessus 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 Nessus Server to be launched if and only if some information gathered by other test exist in the knowledge base, or if and only if a given port is open. This option speeds up the test, but may make Nessus Server miss some vulnerability. If you are paranoid, disable this option.

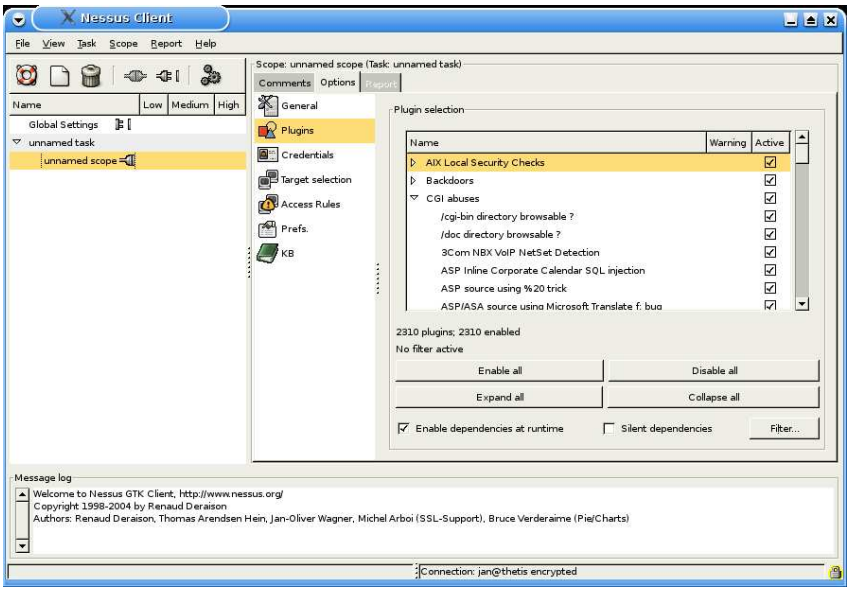
Safe checks Some security checks may harm the target server, by disabling the remote service temporarily or until a reboot. If you enable this option, Nessus Server will rely on banners instead of actually performing a security check. You will obtain a less reliable report, but you will less likely disrupt the network users by doing a test. From a security point of view, we recommend you disable this option. From a sysadmin point of view, we recommand 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 Nessus 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 filled if a connection to a Nessus Server has been established. You can switch on one or more of the scanners. Clicking on an entry shows the details for the respecitve scanner plugin.

4.3.2 Plugins

The plugins are stored on the Nessus 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 as a whole activated or deactivated by checking the box right of family title. 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.

The column “Warning” contains 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 force you to reboot your servers or restart some services manually.

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 a applied filter.

The following actions are possible:

Enable all Enables all plugin categories.

Disable all Disables all plugin categories.

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, then Nessus Server will enable the plugins that are depended on by the set of plugins you selected.

Silent dependencies If you enable this option, then Nessus 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 immedialty with applying a filter.

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

The information shown are the ones specified within the corresponding plugin.

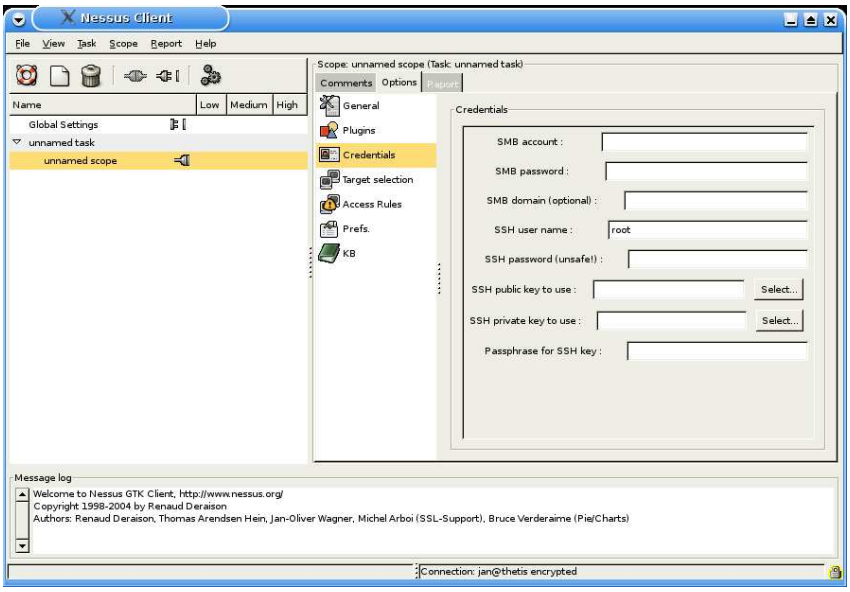
The following actions are possible in this dialog:

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

Show dependencies Another info dialog is raised listing up the dependencies for the plugin. Also it is provided a hint whether the dependencies are currently enabled or disabled.

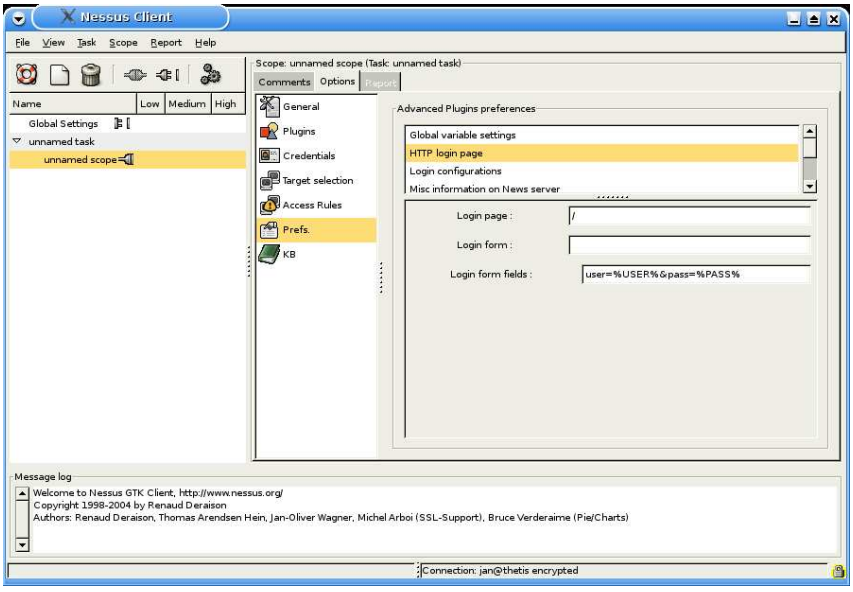
4.3.3 Credentials

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



4.3.4 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 offers a configuration.

4.3.5 Target Selection

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

A special syntax is “file:/some/where/targetlist.txt” which means that the actual target names are loaded from that list. See for the file syntax also below the description for the button “Read from file”.

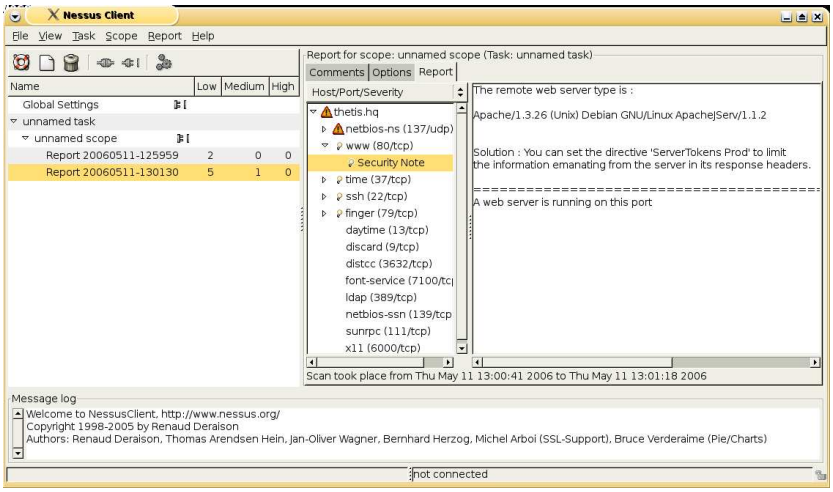
Read from file A textfile can be specified that contains the list of targets. This textfile may contain comma-separated lists of host and also may contain many of such lines.

Perform a DNS Zone transfer Nessus 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 of the target domain. Then, it will test each host.

4.4 Reports

4.4.1 Report Page of NessusClient

The report page consists of 3 elements. On the left hand a tree list allows to browser via hosts, ports and severity to single reports. Ontop 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.



4.4.2 Report Formats

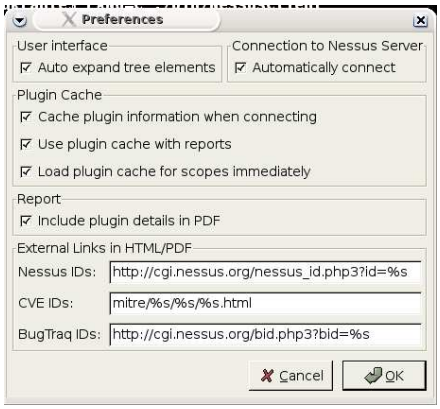
The scan results can be exported into a number of formats. Basically it can be distinguished between 3 types of formats: Interchange-Formats, Editable Documents and Read-only Documents. The last type currently is the PDF Report file. With a capable view it allows to browse back and forth through the document using the various inserted hyperlinks.

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

5 Special Features

5.1 NessusClient Preferences

NessusClient allows to specify some individual preferences that determine some ways how the client GUI works.



The following selection are available:

5.1.1 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.

5.1.2 Connection to Nessus Server

Automatically connect If this setting is enabled, NessusClient 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 NessusClient is closed.

5.1.3 Plugin Cache

Cache plugin information when connecting If this setting is enabled, NessusClient 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 check sums are used to find out about changed and new plugins. Only the changes will be downloaded. Of course, connecting to a different Nessus server will usually force a new download of all plugins.

Second, all plugin information are available even when the server has not been connected. 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 occur and some might disappear. The latter usually doesn't happen. Loading the cache may take a couple of seconds. If you don't want this, switch off option "Load plugin cache for scopes immediately".

Third, the downside of caching: The cache will consume several MBytes for each scope. If this means a problem, you should disable this feature. If you want to remove the caches, search for the files "nessus_plugin_cache" in your Nessus directory ("~/.nessus"). Simply deleting them is sufficient.

Use plugin cache with reports Enabling this setting will make NessusClient 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 NessusClient GUI. So, this cache is for increasing transparency not performance.

Again, the downside is that several MBytes of cache per report will be generated. Disable this option if this means a problem. If you want to remove the caches, search for the files "nessus_plugin_cache" in your Nessus directory ("~/.nessus"). Simply deleting them is sufficient.

Load plugin cache for scopes immediately Disabling this option will cause NessusClient 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.

5.1.4 Report

Include plugin details in PDF Enabling this setting will make NessusClient 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.

5.1.5 External Links in HTML/PDF

These settings determine the URLs for linking more information on Nessus Plugin, CVE/CAN and BugTraq ID in Reports of format HTML or PDF. The defaults as shown in the screenshots are recommended since there up-to-date information are to be found. The defaults are reactivated when field is left empty.

In case you want to package a Nessus report with e.g. CVE/CAN details for offline-reading, you may enter an apropriate 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 filled into the html link parameter “href” as it is. The tool “htmldoc” is issued 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.

5.2 User-defined Access Rules

5.2.1 General

Nessus Server keeps a list of access rules similar to usual firewall rules. In no way can these be overridden by the Nessus Client.

Furthermore, the Nessus Server Administrator can add specific rules for each user while applying the tool `nessus-adduser`.

And finally, the User may add another set of rules in NessusClient.

5.2.2 Syntax

A rule might either be a default definition: “default deny”, “default reject” or “default accept”.

Or it is a target specific definition that consists of the action (“deny”, “reject” or “accept”) and a target specified with its IP (eg. 192.168.10.10).

Note that it must be an IP. The use of hostnames is not allowed due to security reasons.

5.2.3 Manage rules in NessusClient

A rule consists of a action and (in case no default action is dfined) a target. There are in total 6 actions available where the user can select one in the selection box. The target is to be specified manually in a text entry field.

Once clicking the button “Add rule”, the specified action/target is added to the group “Clientside userrules”.

If a rule in this section is selected, the button “Remove rule” will delete it from the list. No serverside rule can be removed of course.

5.2.4 Example for User-defined Access Rule

You are the security team of BigBank Corp. The management authorized you to scan the network, BUT you’re not allowed to scan 172.20.142.2 under any circumstance, because it’s a SWIFT server and if it goes down, so does the bank (and your company and legal responsability). So you add :

```
deny 172.20.142.2
```

as a user rule in the GUI, and you are now SURE that you’ll never scan this IP - even if you later on ask nessusd to scan it.

5.2.5 Example for a Adminstrator-defined Access Rule

By editing nessusd.rules (only the administrator for Nessus Server can do that), you make sure that nessusd will not be used to scan any system it should not scan. If your internal network is 10.0.0.0/8, you basically want to make sure that nobody will scan anything else (either by fat-fingering an IP or just by malice), so you edit `nessusd.rules` and you add :

```
allow 10.0.0.0/8
allow 127.0.0.1
default deny
```

5.3 Local Security Checks: Theory

5.3.1 What are "local" security checks?

Starting with Nessus 2.1.0, it is possible to give Nessus SSH credentials to log into the remote Unix systems and determine which patchs need to be applied. What this really means is that if you give Nessus the proper SSH key, it will litterally log into the remote host, extract the list of installed software, and tell you which packages have an update ready for them.

The real goal of this feature is to easily keep track of all the missing patches of a given system, without the potential hazards and false positives of remote security checks. It is really important to understand that enabling local security checks do not discover flaws such as unauthorized suid binaries, they simply determine which patches are missing from the remote operating system.

5.3.2 Which operating systems are supported at this time?

At this time, Nessus is able to determine the missing patches of the following operating systems :

- AIX 5.x
- Debian
- Fedora Core 1 and 2
- FreeBSD 4.x, 5.x (including the port collection)
- Gentoo Linux (all advisories from 2004)
- Mac OS X (and Mac OS X Server)
- Mandrake Linux (8.0 to 10.0)
- RedHat Enterprise Linux (2.1 and 3.0)
- Solaris (2.5.1, 2.6, 7, 8 and 9)
- SuSE Linux (7.0 to 9.1)
- Microsoft Windows NT, 2000, XP, 2003

We hope to support more systems in the future (HP-UX, Tru64, etc...).

5.3.3 Which Solaris patches are checked for?

We have implemented checks for Solaris 2.5.1, 7, 8 and 9 (both on sparc and i386 architectures). We check for every patch listed by Sun in the "Patch containing security fixes" of their patch page.

5.3.4 Is it intended to support more systems?

We do not intend to support every operating system which has ever surfaced on the face of earth. However, we intend to create “working groups” of people working on a given operating system. Basically, we are looking for people who would be responsible for producing the security checks relevant to the operating system they use (and love). If you are interested in taking part in such a group (or to even lead a given OS group), please contact the Nessus Development Team.

5.3.5 What are "local" security checks NOT?

At this time, Nessus does not check the local security policy of the remote system, like Tiger does (ie: directories writeable by any users, wrong permissions on a sensitive configuration file, and so on...). Read about SLAD (Security Local Auditing Daemon) for extensive on-site testing of remote targets that porentially are compromised. SLAD can be accessed via respective NASL scripts.

5.3.6 Why do we need to perform local security checks, is not Nessus good enough to determine security issues?

Nessus 2.0 is, at its heart, a network based scanner. This means that it will connect to the remote host and attempt to determine the flaws affecting every service on it. However, testing everything from the point of view of the network has its own shortcomings. In particular, a 100% network-based security scanner can not do the following :

- Determine client-side flaws: There are more and more client-side vulnerabilities published every day, ranging from flaws in mail clients to XML libraries. It simply is not possible to check for those from the point of view of an active scanner. The only way to detect such flaws is to deploy a passive scanner (ie: NeVO) or to locally check that every host has been patched.
- Determine local flaws: Obviously, the security level of local utilities can not be determined from the point of view of the network. By running local security checks, Nessus will tell you if a user logged into the remote host can use an overflow in a suid binary to gain root privileges.
- Determine flaws in disabled services: If a host has a vulnerable service installed, but it's not running at the time of the scan, a vulnerability will be missed. Having an unpatched disabled service is a potential security hazard - if the service is ever enabled in the future, it might be exploited.

So traditional network-based scanners have their inherent limitations, which can now be overcomed thanks to this new feature.

5.3.7 Will Nessus recognize individually compiled and installed packages?

If you e.g. compiled your DNS server by hand instead of using the system's one: Will Nessus see it ? No!

Nessus sticks to the patches provided by your OS vendor. However, note that it usually is a good administrative practice to stick to your vendor packages, as it makes the upgrade of your system much easier.

In the future, Nessus might be able to detect if a running server is not of the same version as the one installed by RPM (or any other packaging system) and may improve its output this way.

5.3.8 Why care about local security if there is no local user?

E.g. why should one care if the web server has local overflows in suid binaries but there's no local user on it, apart from oneself?

At the time this feature starts to be discussed, many people would object that they don't really care about local security because their server simply is a web server with no local users, therefore nobody can exploit a local vulnerability. The answer is a simple security practice: **containment**. If your web server is vulnerable to a given flaw (be it a misconfiguration, a badly written PHP script, or an 0day in Apache), an attacker will be able to obtain a shell on your system. The question really is: do you want to

make it easy for the attacker to gain super-user privileges and gain the full control of your system, or do you want to contain him to the privileges your web server is running with? (These privileges will prevent the attacker to modify your kernel, load a rogue kernel module, and generally will make it harder for him to cover his tracks). So **local security really matters**, since it's your last line of defense.

5.3.9 How can enabling local security checks improve my scanning experience?

Depending on how you use them, local security checks can dramatically improve your scanning experience:

- By reducing false positives: it is not always possible to remotely determine if a given service is vulnerable to a given flaw or not (sometimes the only way to check for it is to actually crash the service, which is not an option in a production network). Sometimes, vendors do not upgrade a faulty product to the newest version but backport the fix instead (see Red Hat's statement about this), which also makes life more difficult for network-based scanners .

- By speeding up the checks: if all you are interested in is to determine if all the patches have been installed on your remote hosts, you may want to only enable local security checks and dramatically reduce the time it takes to scan your systems. It only takes 6 seconds to find all the local flaws of a remote system (over a VPN link) whereas it takes over one minute to scan the same system entirely with all the network checks.

- By scanning more often: Every network-based security scanner create the risk of crashing devices/services on your network. This is not specific to Nessus, this really is a hard fact about security scanning in general. The reason is simple: a scanner attempts to connect to all the services on the remote host, and some services (and TCP/IP stacks) out there are so poorly written that connecting to them too quickly or not sending them the exact data they are waiting for may crash them. No matter how gentle the scanner is, there is always a risk of crashing a custom application, a loaded server which is low on memory, and so on. This is especially true for embedded devices, most of which being designed without any consideration for security. There are scanners vendors out there who are trying to claim that they have reached "100% unintrusive scanning", and this is simple not true. If a scanner interacts with a network service, there is always a risk for crashing it.
  As a result, many people only scan once a month or once a quarter, even though security issues appear every day!
  By only enabling local checks and disabling port scanners and every other plugins, Nessus Server will only connect to the SSH port of the remote hosts - no stress on the network, no bad surprises.

So wether you administrate a small number of hosts or a huge number of systems, local security checks can definitely make your job much easier.

5.3.10 Does this feature slow down the scan?

No. Nessus will minimize the number of SSH logins into the remote host, and will actually log in once, then load its knowledge base with various information found on the remote host. Please view the source code of the plugin ssh_get_info.nasl for more details about what is gathered.

5.3.11 What are the caveats of this new feature?

At this time, the following caveats are known:

- Only SSHv2 is supported. You can not log into the remote host using SSHv1.
- You have to use SSH public key authentication. It's slightly harder to deploy than a password, but it's much more secure.
- This documentation only deals with OpenSSH.

5.4 Local Security Checks: How to enable

5.4.1 The principle

To enable local security checks, the idea is to:

- Create a SSH private/public key pair for Nessus to use
- Create a user account on every system for which you want to perform a local scan
- Copy the SSH public key Nessus will use in the directory of the new user
- Tell Nessus to use this SSH private and public keys and perform the scan

As you can see this is pretty simple. Nessus can not use password authentication, otherwise it would be vulnerable to a man-in-the-middle attack when scanning the remote systems (we do not want to handle key management for the remote systems just yet).

If you are not familiar with SSH public key authentication and understand what it does, you should consult the ssh documentation or corresponding publications.

5.4.2 Generating a SSH public key

To generate a SSH public key, use ssh-keygen and save the key in a safe place.

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.  Enter file in which to save the key
(/Users/renaud/.ssh/id_dsa):
/home/renaud/Nessus/ssh_key
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
Your identification has been saved in /home/renaud/Nessus/ssh_key.
Your public key has been saved in /home/renaud/Nessus/ssh_key.pub.
The key fingerprint is:  06:4a:fd:76:ee:0f:d4:e6:4b:74:84:9a:99:e6:12:ea
renaud@myth.local
```

As you can see in the example above, two files have been created:

- /home/renaud/Nessus/ssh_key is the private SSH key. Do not transfer it on a system other than the one running your Nessus client.
- /home/renaud/Nessus/ssh_key.pub is the public SSH key. You will copy this file on every system you want to scan with Nessus.

When ssh-keygen asks you for a passphrase, you may want to enter return twice (ie: don't set any passphrase). It's safer to protect a key with a passphrase, however, given the way Nessus is implemented it will not buy you extra security, as the passphrase will have to circulate between your Nessus Client and the Nessus Server, and will be stored in clear text in your .nessusrc configuration file. If you feel safer, feel free to enter a passphrase.

5.4.3 Creating a user account and setting up the SSH key

On every platform you want to test, you want to create a new user account dedicated to Nessus. We will call this user nessus in this document, but you can set his name to any value. Once you created an account for the user (using adduser or any utility that the system provides you with), make sure that he has no valid password set (ie: when editing the file with 'vipw' make sure that the password field only contains a star):

```
# vipw
(...)
nessus:*:502:502::0:0:Nessus Test Account:/home/nessus:/bin/bash
(...)
```

Now that the user has been created, you simply need to copy the key to his home and set the correct permissions on the .ssh directory:

```
# mkdir ~nessus/.ssh
# mv ssh_key.pub ~nessus/.ssh/authorized_keys
# chown -R nessus:nessus ~nessus/.ssh/
# chmod 0600 ~nessus/.ssh/authorized_keys
# chmod 0700 ~nessus/.ssh/
```

5.4.4 Setting up Nessus

First, make sure that you are running Nessus 2.1.0 or newer and that it has been compiled with SSL support. You can do that by typing nessusd -d:

```
[root@myth Home]$ nessusd -d
This is Nessus 2.1.0 for Darwin 7.4.0
compiled with gcc version 3.3 20030304 (Apple Computer, Inc.  build 1495)
Current setup :
nasl:  2.1.0
libnessus:  2.1.0 SSL support:  enabled
```

Then start the NessusClient, and go to the "Credentials" section and set the SSH user name and SSH keys properly:

Make sure not to mix the public key (keyname.pub) and the private one. Make sure that the option "enable dependencies" at runtime (in the plugins) section is checked, and run a scan against the remote host. The Nessus report should now tell you that login in was successful, and should show you all the missing patches on the remote system:

Repeat the operation for every server for which you want to do a local scan, and you are ready to go!

5.4.5 Acknowledgements

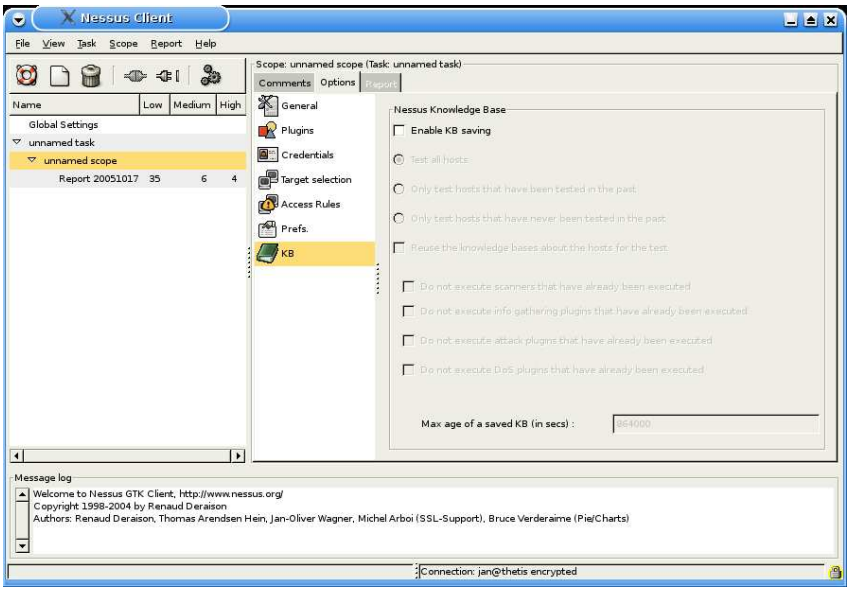
The SSH local checks functionality is a courtesy of:

- Nicolas Pouvesle who wrote a lightweight SSH client implementation in NASL
- Tenable Network Security. Tenable wrote over 1200 plugins performing local security audits.

6 Optimizing and Fine Tuning

6.1 Knowledge Base

Knowledge Base saving allows you to not disturb the users of your network by doing daily portscans of a /24 network and to not waste the result of other tests.



The Nessus Server as well as the Nessus Clients have a compile option to switch on Knowledge Base support. Usually that is the case but you may find servers or clients installations without this feature.

In the NessusClient you find the configuration of the knowldege base on the page “KB” of the “Options” notebook (provided the respective client support the KB feature).

6.1.1 Introduction

The knowledge base (KB) is the list of information gathered about a tested host. It contains the list of open ports, the operating system type of the host, and many more information. Its first purpose was to reduce the redundancy of the tests, so that a plugin A which finds a fact (for instance, a way to log into the remote FTP server) writes can share its result with the other plugins (for instance, a plugin which needs to log into the remote FTP server will look in the KB if there is a way to do so). After a test, the KB would be freed from memory, and would be rebuilt from scratch during the next test. The idea behind the saving of the knowledge base is to re-use it for another audit, to reduce the bandwidth consumption of a test. For instance, if you have not touched the configuration of your web server during the last 5 hours, it’s very unlikely that a new port will be opened when you do a new port scan on it. So why waste your bandwidth and time re-doing a portscan when you could just re-use the result of the previous portscan you did against this host ? Saving the KB allows you to:

- save bandwidth
- save time
- test the hosts of your network for new vulnerabilities only

You can define a lifetime for the knowledge bases attached to the tested hosts. This lifetime can range between a few seconds to a hundred of years - you are free to configure it the way you want. The purpose of this section is to explain you how to enable and use this feature to increase your security by making more tests more often, thus having a day-to-day view of your network.

6.1.2 Using the KB saving

If you want the knowledge bases to be saved on the Nessus Server side, you must enable the option "**Enable KB saving**" in the client:

This option enables all the knowledge base saving, and by enabling this checkbox, you know that all the information collected about the remote hosts will be saved on disk. If you enable this option only, then the data gathered from the hosts you are testing for will be saved on disk, and that's all. That is, nessusd will not use it when you test the same host again. If you want to take advantage of it for the rest of your tests, continue to read.

Technically speaking, each knowledge base is saved on the remote host as a different file for each tested host. A database may be pluggable here in the future, but at this time, this is fine. The files are located in "/usr/local/var/nessus/username/kbs/" where username is the Nessus Server login name in use.

6.1.3 Reducing the hosts tested to (re)populate the knowledge bases

When "**Enable KB saving**" is set, then it is possible to reduce the set of tested hosts to populate the knowledge bases of the current user. You may want to just test the hosts that already have a knowledge base attached to, or at the opposite, you may want to only tests the hosts that have no, or an outdated, knowledge base, so that the set of knowledge bases is up-to-date and complete. That's where the three options "**Test all hosts**", "**Only test hosts that have been tested in the past**" and "**Only tests hosts that have never been tested in the past**" are handy.

- If "**Test all hosts**" is set, then all the hosts present in the "Target" field of the target selection panel will be tested; that is, no filtering will be done, and everything will be tested as it used to be.
- If you set the option "**Only test hosts that have been tested in the past**", only the hosts to which an up-to-date knowledge base is attached will be tested. This allows you to make sure that, for instance, no port scan will be done against hosts that have never been tested in your selection, so you are sure that network noise will be low.
- Last but not least, the option "**Only test hosts that have never been tested in the past"** is set, then only the hosts that have no knowledge base, or an outdated one, will be tested, and will be given a fresh new KB.

These three options allow you to do what you want :

- test everything

- make sure that no noise will be generated during this test
- populate the set of knowledge bases

6.1.4 Re-using the knowledge base between multiple tests

Re-using a knowledge base saved in the past may be dangerous, suppose for instance that the host you have audited yesterday was broken into during the night, and that the attacker left a root shell open on a port that used to be closed. If you re-use the knowledge base, then you will miss this fact. On the other hand, on a well-secured network, you may want to re-use a previously saved KB to save time and test a restricted set of new security checks against the host.

If you enable the option "**Reuse the knowledge bases about the hosts for the test**", then the older knowledge base will be reloaded in memory, and Nessus Server will make an active use of it. You can control what Nessus Server will make of it using the four following options:

...

If "Do not execute scanners that have already been executed" is set, then scanners you used in an older test won't be used again. The same for the other three options. If you just want to execute the new checks you gathered using the command "nessus-update-plugins", then it is suggested you enable all these options. If you just want to reduce the noise of your network and don't do any port scan, just enable the first option. And so on...

6.1.5 Defining the meaning of up-to-date

Throughout this section, it is said that a KB can be declared as being up-to-date or outdated. What is meant by this is what you want. Each time a KB is created, it is attributed a date tag. The option "**max age of a saved KB**" defines, in seconds, the amount of time after which a KB is declared as being obsolete and will be regenerated from scratch.

The default value is 864000 (10 days), but you are free to lower this to 3600 (one hour), or less, or at the opposite, raise this value to whatever you want.