SecurityFocus™

**Infocus**

**< http://www.securityfocus.com/infocus/1741 >**

# Introduction to Nessus

*by* Harry Anderson

last updated October 28, 2003

## 1.0 Introduction

Nessus is a great tool designed to automate the testing and discovery of known security problems. Typically someone, a hacker group, a security company, or a researcher discovers a specific way to violate the security of a software product. The discovery may be accidental or through directed research; the vulnerability, in various levels of detail, is then released to the security community. Nessus is designed to help identify and solve these known problems, before a hacker takes advantage of them. Nessus is a great tool with lots of capabilities. However it is fairly complex and few articles exist to direct the new user through the intricacies of how to install and use it. Thus, this article shall endeavor to cover the basics of Nessus setup and configuration. The features of the current versions of Nessus (Nessus 2.0.8a and NessusWX 1.4.4) will be discussed. Future articles will cover Nessus in more depth.

Nessus is a public domain program released under the GPL. Historically, many in the corporate world have ridiculed such public domain software as being a waste of time, instead choosing "supported" products developed by established companies. Typically these packages cost hundreds or thousands of dollars, and are often purchased using the logic that you get what you pay for. Some people are starting to realize that public domain software, such as Nessus, isn't always inferior and sometimes it is actually superior. Paid technical support for Nessus is even available from www.tenablesecurity.com. Nessus also has a great community of developers anchored by the primary author, Renaud Deraison. When allowed to fairly compete in reviews against other vulnerability scanners, Nessus has equaled or outshined products costing thousands of dollars. [ref: Information Security, Network Computing]

One of the very powerful features of Nessus is its client server technology. Servers can be placed at various strategic points on a network allowing tests to be conducted from various points of view. A central client or multiple distributed clients can control all the servers. The server portion will run on most any flavor of Unix. It even runs on MAC OS X and IBM/AIX, but Linux tends to make the installation simpler. These features provide a great deal of flexibility for the penetration tester. Clients are available for both Windows and Unix. The Nessus server performs the actual testing while the client provides configuration and reporting functionality.

## 2.0 Installation

Nessus server installation is fairly simple even for a Windows jockey like me. First an installed version of Unix is required. Secondly, prior installation of several external programs is recommended: NMAP is the industry standard for port scanners, Hydra is a weak password tester and Nikto is a cgi/.script checker. While not required, these external programs greatly enhance Nessus' scanning ability. They are included because they are the best applications in their class. If installed in the PATH$ before

Nessus installation, they will automatically be available.

The simplest installation method is using the Lynx automatic install. Lynx is included on many of the linux versions. The Lynx command is (logged in as a user, and not root) :

```
lynx -source http://install.nessus.org | sh
```

This should install the server on most platforms with no other steps necessary. Note that the latest install script can also be downloaded and run locally. Whether you install directly off the Website or using the same install script offline, either way the script will setup a temporary suid and ask for your root password when required -- if you don't like this feature you can download, compile and install the four required tarballs individually. The above command should also be used periodically to upgrade Nessus as new versions are regularly released. You will be questioned about proxy servers, a download method (www or CVS), and the branch of the development tree to use; most of the time the defaults are the best choice. This is the simplest method of installation however; you are effectively giving the install.nessus.org server temporary root privileges. Thus there is a security risk with this method albeit a low one. So if you are paranoid, and paranoid is not always a bad thing in the security field, installation can be done the old-fashioned way by downloading and compiling the source. For information on performing an install from scratch see: www.nessus.org/nessus_2_0.html.

## 3.0 Setup

Once the server is installed, some basic setup steps are required. The first task to complete in the new install is to add a user. A new user can be added by the "nessus-adduser" command. The script will question you for the authentication method. Authentication can be performed by several means, however a password is the simplest and is recommended. The next question queries about rules to restrict the user account. When used across an enterprise, a user can be restricted and only allowed to scan specified IP addresses. However, for most uses this will be left blank, allowing the user to scan anything. A certificate also needs to be generated as well to be used to encrypt the traffic between the client and server. The `nessus-mkcert` command accomplishes this.

### 3.1 Update plug-ins

Before a scan is done, the plug-ins should be updated. Nessus plug-ins are very much like virus signatures in a common virus scanner application. Each plug-in is written to test for a specific vulnerability. These can be written to actually exploit the vulnerability or just test for known vulnerable software versions. Plug-ins can be written in most any language but usually are written in the Nessus Attack Scripting Language (NASL). NASL is Nessus' own language, specifically designed for vulnerability test writing. Each plug-in is written to test for a specific known vulnerability and/or industry best practices. NASL plug-ins typically test by sending very specific code to the target and comparing the results against stored vulnerable values. There are a few built-in plug-ins that do not use NASL. These are C and Perl scripts to perform special purposes that can not easily be done in NASL. Among these is the Services plug-in which identifies port-to-program mappings.

Plug-in updates should be done frequently. New vulnerabilities are being discovered and disseminated all the time. Typically after a new vulnerability is released to the public, someone in the Nessus community writes a NASL plug-in, releases it to the public and submits it to www.nessus.org. It is then reviewed by the developers and added to the approved plug-in list. For high risk, high profile vulnerabilities a plug-in is often released the same day the vulnerability information is publicly

released. Updating plug-ins from the maintained list is fairly simple involving a simple command: `nessus-update-plugins`. This command must be done as root. By no means however, are you limited to the list of plug-ins from www.nessus.org. New and special purpose plug-ins can be written relatively easily using NASL, so you can write your own custom plug-ins as well.

**3.2 Launch the daemon**

Nessus is now installed, updated and ready to go. The simplest way to get the server running is (as root) issue the `nessusd -D` command. In order to use it, one must use a client. There are three primary Nessus clients. The native Unix GUI version is installed at server install time. Alternatively, Nessus can be controlled from the command line. A third option, a Windows version also exists called NessusWX. The binaries for NessusWX can be found here . The NessusWX install is a straightforward Windows install. All three clients work well. Personally I prefer NessusWX. It is better organized, allows for easier reporting, and has a better facility for managing different sessions (groups of hosts to scan) then its Unix counterparts. To run the native Unix GUI client, run the nessus command or for NessusWX click the eye icon after installation.

**3.3 Client connection**

Since Nessus is a client server technology, once running the client a connection must be made to the server. In the native client, enter the server IP, username and password (created with the `nessus-adduser` command) and hit login. The process in NessusWX is similar but uses the communications | connect menus. The client is connected to the server thru an SSL connection and a list of the currently installed plug-ins is downloaded. On the first run the SSL certificate is also downloaded and verification is requested. This verification ensures that in the future you are actually communicating with the server intended. Figures 1 and 2 shows the connection using the Unix and Windows GUI tools, respectively. Figure 3 shows user authentication using the NessusWX client.

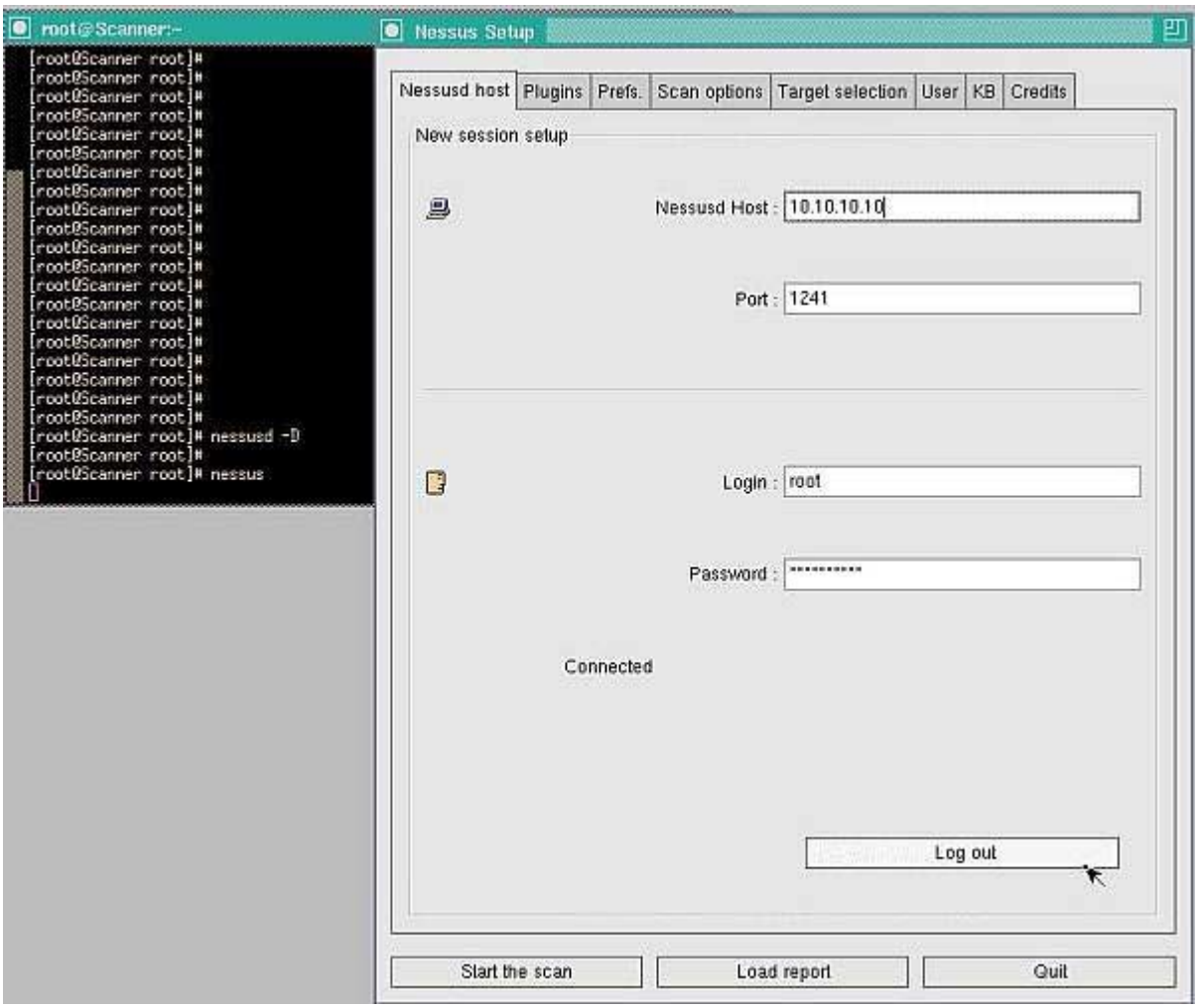


**Figure 1: Starting the Nessus server and connecting with the Unix GUI**

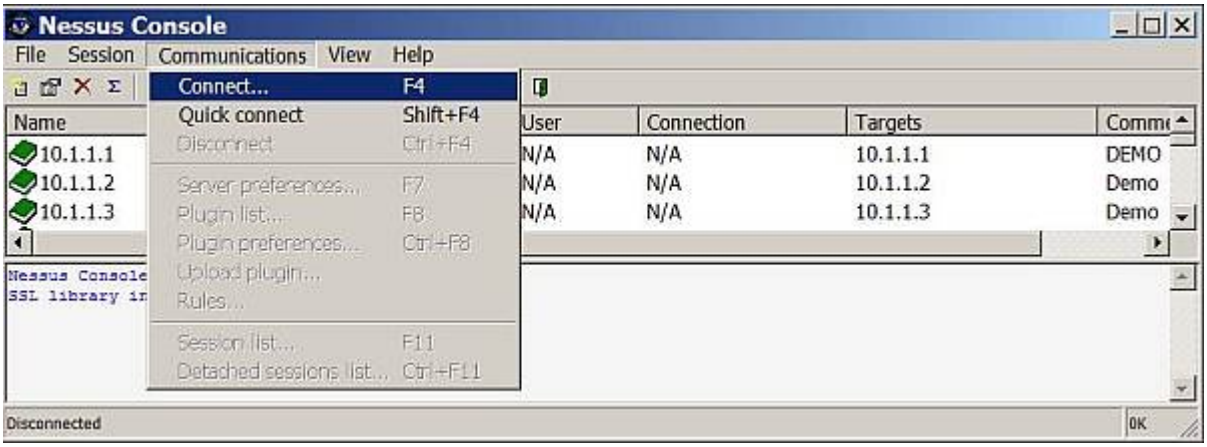


**Figure 2: Connecting to the Nessus server with NessusWX (Windows Client)**

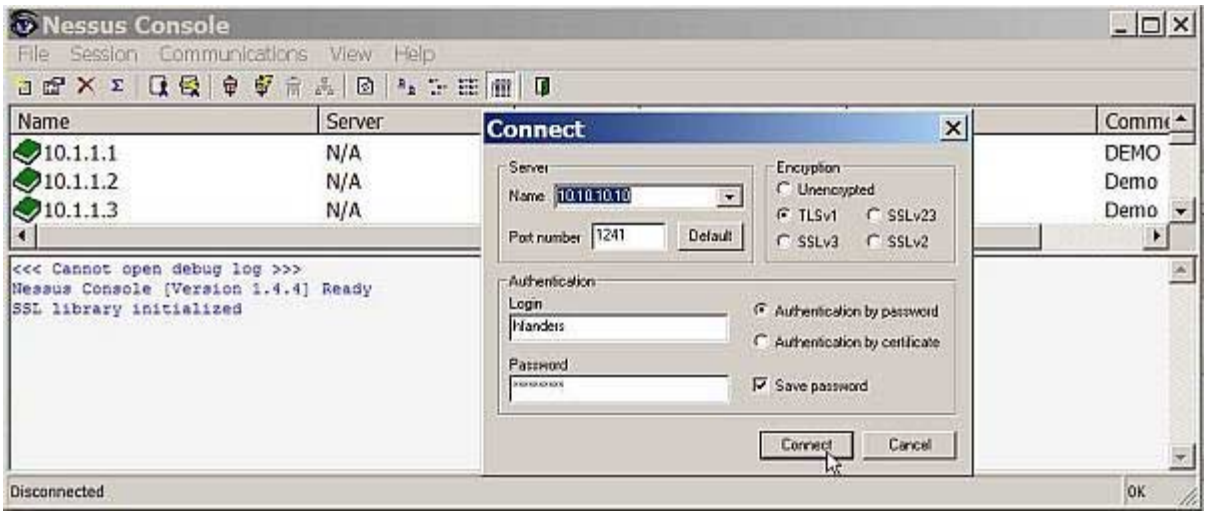


**Figure 3: Enter in the server IP and the login and password setup with nessus_adduser**

## 4.0 Using Nessus

Now that we have installed and connected to Nessus lets explore some of the options available. The most obvious and powerful aspect of Nessus is its plug-in feature. The choice of plug-ins is critical to the success of a scan. Most plug-ins are written very well and rarely trigger false positives or negatives; however, a few are not. One example of a poorly written plug-in is the test for the classic Windows IIS hack RFP's MSDAC /RDS vulnerability. Rain Forest Puppy (RFP) publicized this vulnerability in 1999. The vulnerability makes use of the %system%/msadc/msadcs.dll file and leads to total system compromise on un-patched IIS 4.0 servers. The problem is that the Nessus plug-in only tests for the existence of the /msadc/msadcs.dll file. It does not take into account patches, windows versions etc. Thus with this plug-in enabled, a false positive will show up on many IIS servers and must be sorted out manually.

Before anyone yells out, "see the problems of public domain software," one should note that the same types of problems exist with the high priced "supported" vulnerability scanners as well. This problem is a result of the current state of the technology. The difference is that typically in purchased products you cannot easily examine the exact "proprietary" testing methodology as can be done with Nessus, thus making resolution of the false positive difficult.

### 4.1 Choosing dangerous/non-dangerous plug-ins

Plug-ins are categorized in several different and sometimes confusing ways. One method of plug-in grouping is by category. Most importantly, some plug-ins are categorized as Dangerous/Denial of Service (DOS). These plug-ins will actually perform a DOS attack and crash systems that have these particular problems. Needless to say these should not be blindly run on production systems. They won't cause long term damage, but at least a reboot will be required. In both clients, there are buttons to "Enable all plug-ins" or just "Enable all but dangerous plug-ins" (called "Enable Non-DOS" in NessusWX). Note that the author of the plug-in decides if it is dangerous or not. Most of the time, this has been very well chosen. However there are instances (Exmple: the rpc_endpoint mapper plug-in), where the plug-in causes a DOS but it is not listed as dangerous. The native client denotes dangerous plug-ins with a caution triangle. NessusWX has no special way to notate a dangerous plug-in other then using the enable "Enable Non-DOS" button. One other thing to be aware of is that sometimes even a "non-dangerous" plug-in can crash software. Since the plug-ins are sending non-standard data, there is always the risk, albeit rare, that a new undiscovered DOS will be stumbled upon. Therefore

anytime systems are being scanned one should be aware that system crashes, although unlikely, are possible even with "non-dangerous" plug-ins. Figure 4, below, shows plug-in selection using the Unix GUI. Similarly, Figures 5 and 6 show plug-in selection using NessusWX for Windows:

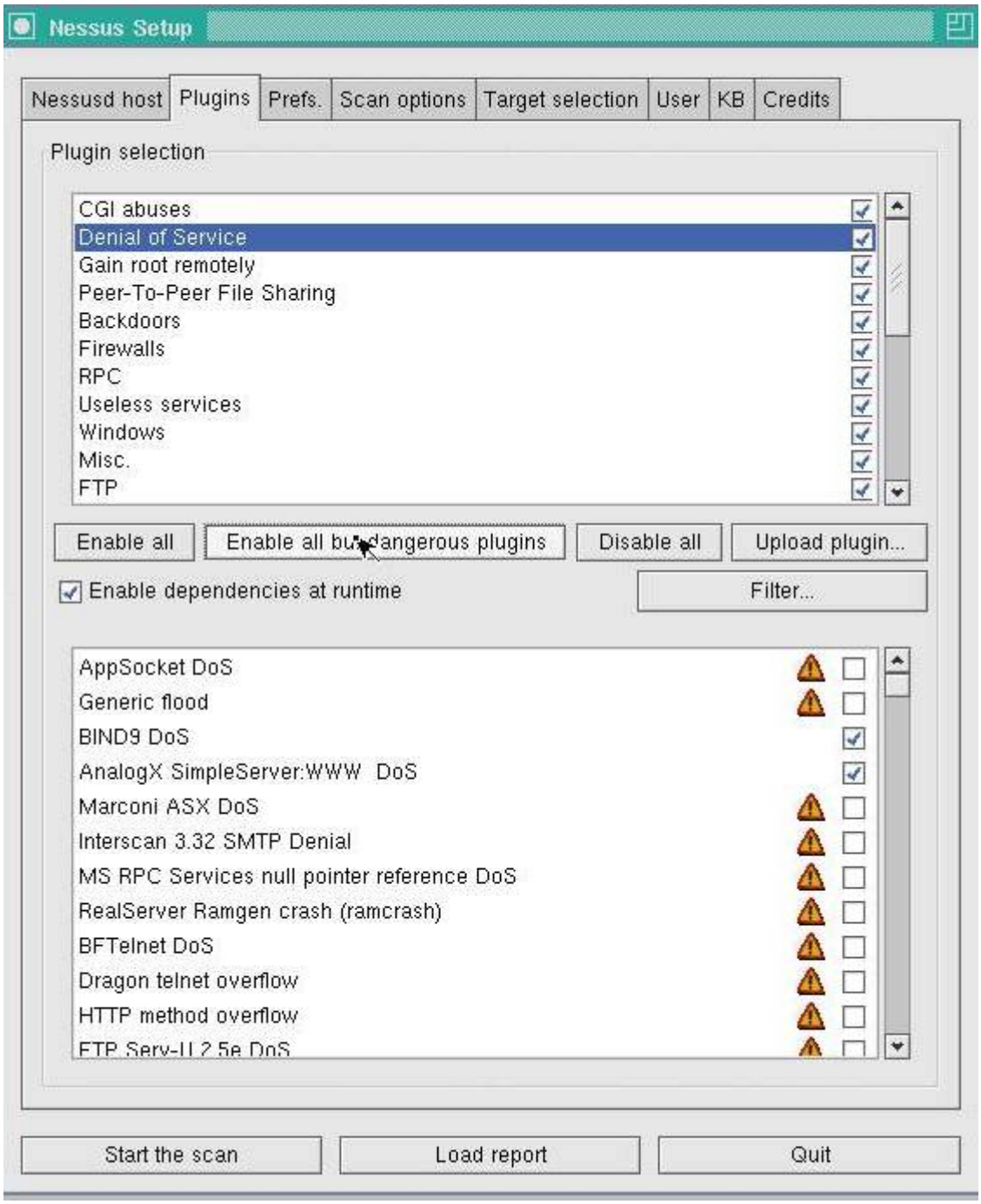


**Figure 4: Enabling all but dangerous plugins with the Unix Nessus GUI**

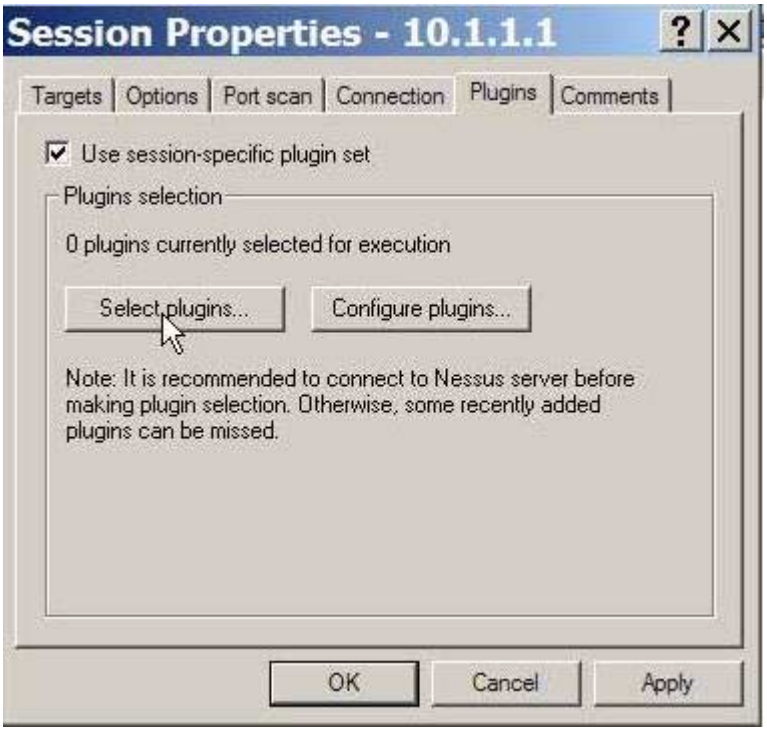


**Figure 5: Selecting plug-ins with the Windows NessusWX Client**

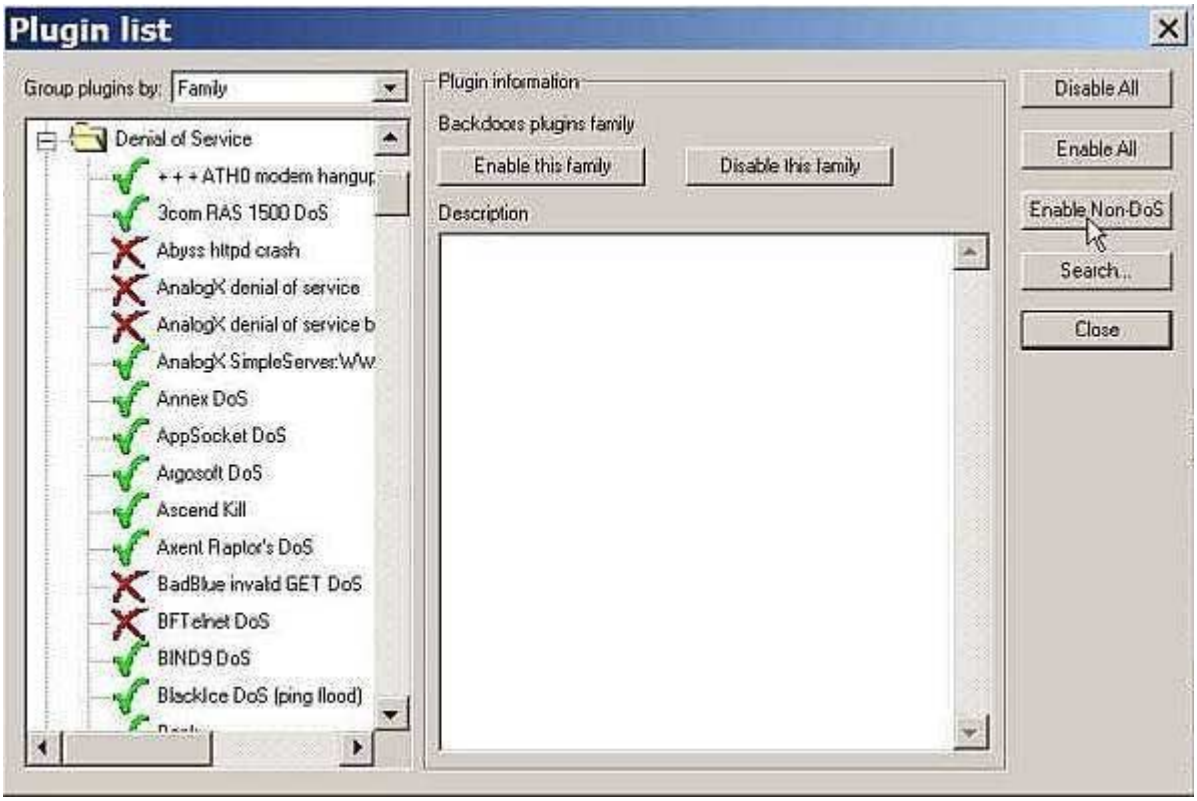


**Figure 6: Enabling non-dangerous plug-ins with the Windows NessusWX Client**

### 4.2 Safe-checks

This is a good place to mention the related concept of safe-checks. Safe-checks disables the dangerous parts of safe-check compatible plug-ins and causes them to check just through passive methods such

as version numbers in banners. Since a patch or workaround may be installed, safe-checks are not as reliable as actually exploiting the vulnerability. They might cause false positives or false negatives. The valuable trade off is that they should not crash a machine. The safe-check option is on the scan options tab (the options tab in NessusWX). Figure 7 shows the safe-check option in the NessusWX interface:

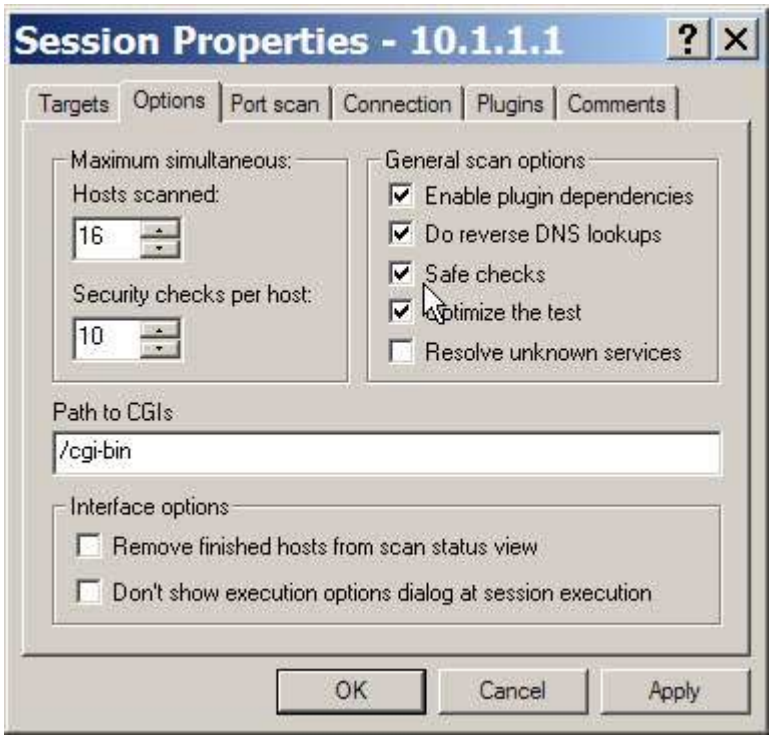


**Figure 7: Choosing Safe Checks**

The second method of plug-in organization is in families such as Windows, FTP SNMP, SMB, Cisco, etc. I find this to be a rather unhelpful grouping due to the arbitrary categorizing process. Should an FTP vulnerability that only exists on a Windows box go in the Windows family or the FTP family? Since this decision is left up to the plug-in writer with little guidance, there are examples of both. The filtering/search mechanism is very helpful to isolate certain vulnerabilities. A filter can be initiated on name, plug-in number, etc. Clicking on the family and then the plug-in will give details of what the plug-in tests for. If intricate details are needed, the actual NASL code at cgi.nessus.org/plugins/ can be referenced. Note the DOS family is not the same as the dangerous/DOS category of plug-ins. A dangerous/DOS category plug-in actually exploits the vulnerability while a plug-in in the DOS family may just check for the vulnerability by checking software versions, for example. To perform a simple noisy scan on a non-production system, enabling all plug-ins is the best choice. If stealth or a production system is involved, choices can get complicated. We will talk in-depth about plug-ins selection in a future article.

### 4.3 Port scanning

The other critical part of the scanning process is port scanning. Port scanning is the process by which the active ports for an IP address are identified. Each port is tied to a specific application. Nessus is a smart scanner and only runs a test if the specific program for that test is available. For example, only Web server plug-ins are run if a Web server is found. Since often ports are changed from their default to hide them, Nessus has a plug-in called services. The services plug-in attempts to identify the program running on each port. Once the program is identified, only the user-selected and pertinent

plug-ins are run against it.

Nessus has several options to scan for ports. There is the built-in wrapper for NMAP, widely acknowledged as the best port scanner around. There is also an internal scanner and a custom ping scan. As with plug-in selection, port scanning is very dependent on the situation. For a simple scan, the internal "sync" scan using default parameters with pings enabled, found on the "Perf" tab of the Unix GUI and the Port scan tab of NessusWX, is sufficient. Figures 8 and 9, below, show the internal SYN scan option using NessusWX and the Unix GUI client, respectively:

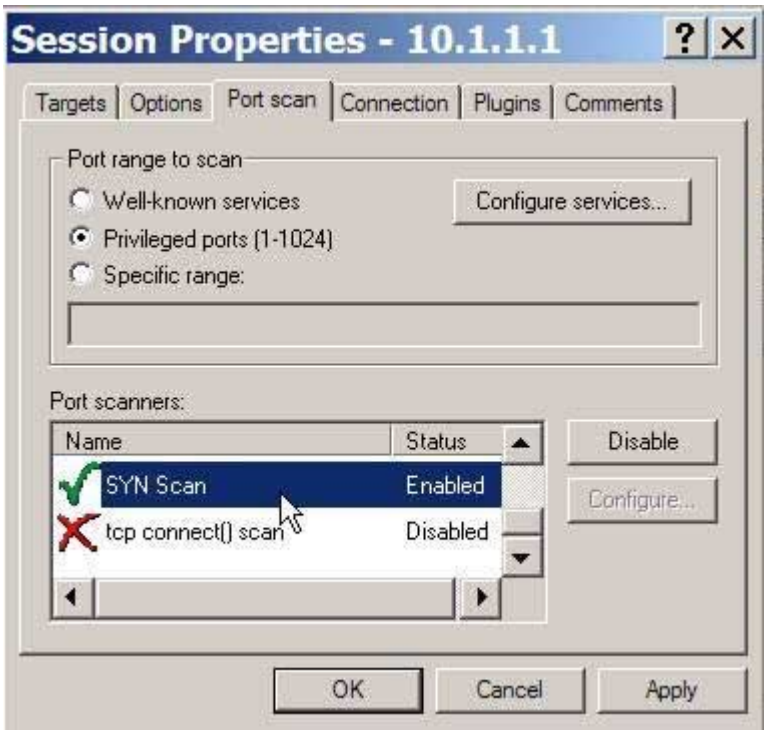


**Figure 8: Configuring the internal SYN scan for a simple port scan on NessusWX**

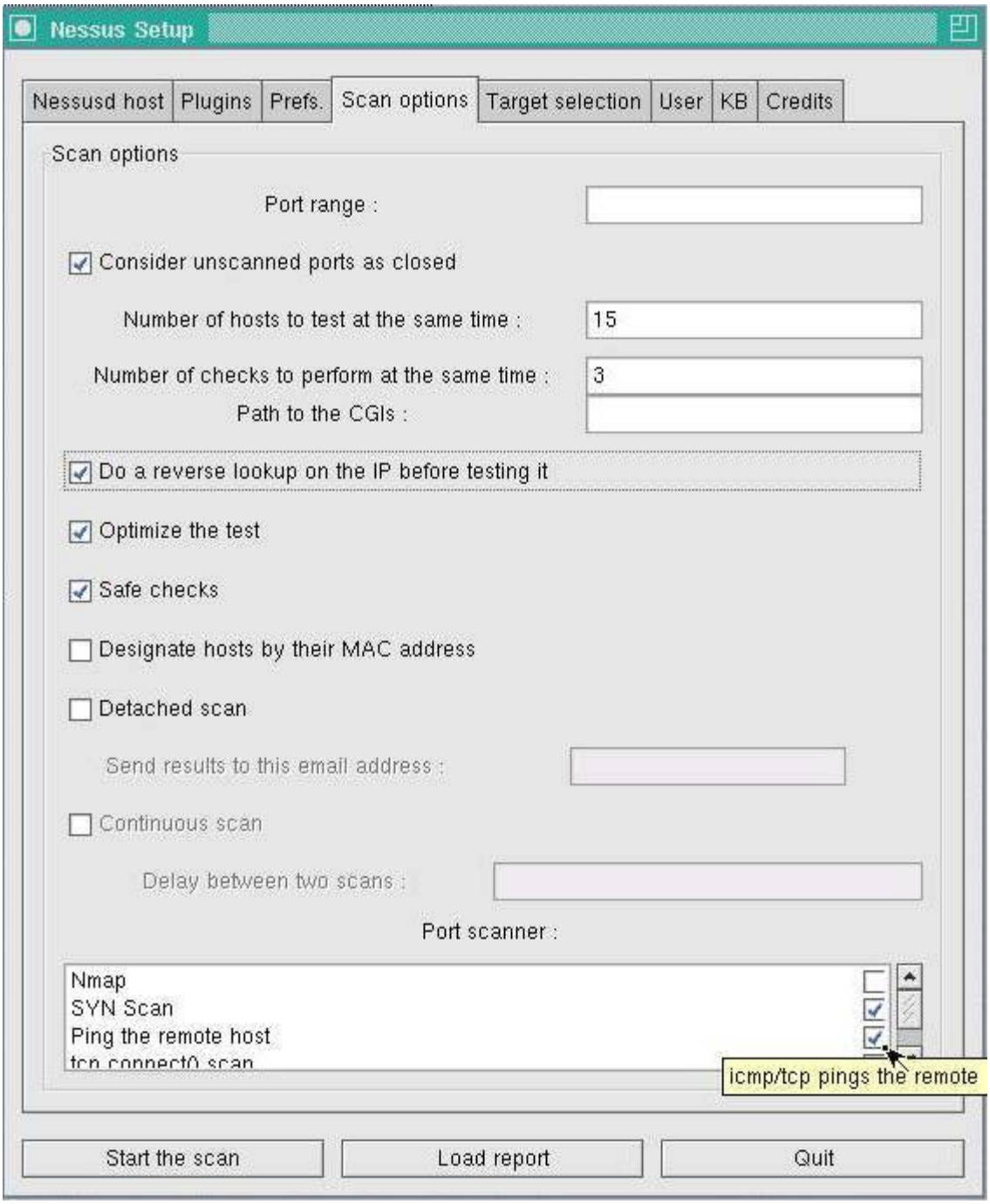


**Figure 9: Configuring the internal SYN scan for a simple port scan on the Unix Client**

### 4.3 Identify targets

The final task is to identify your targets. This is done on the targets tab. Targets can be specified as a single IP Address, as a subnet or as a range of IP addresses. I normally try to break them down into logical groups. It is typically easier to deal with smaller groups at one time. Figures 10 and 11 show how to select targets in both client environments:

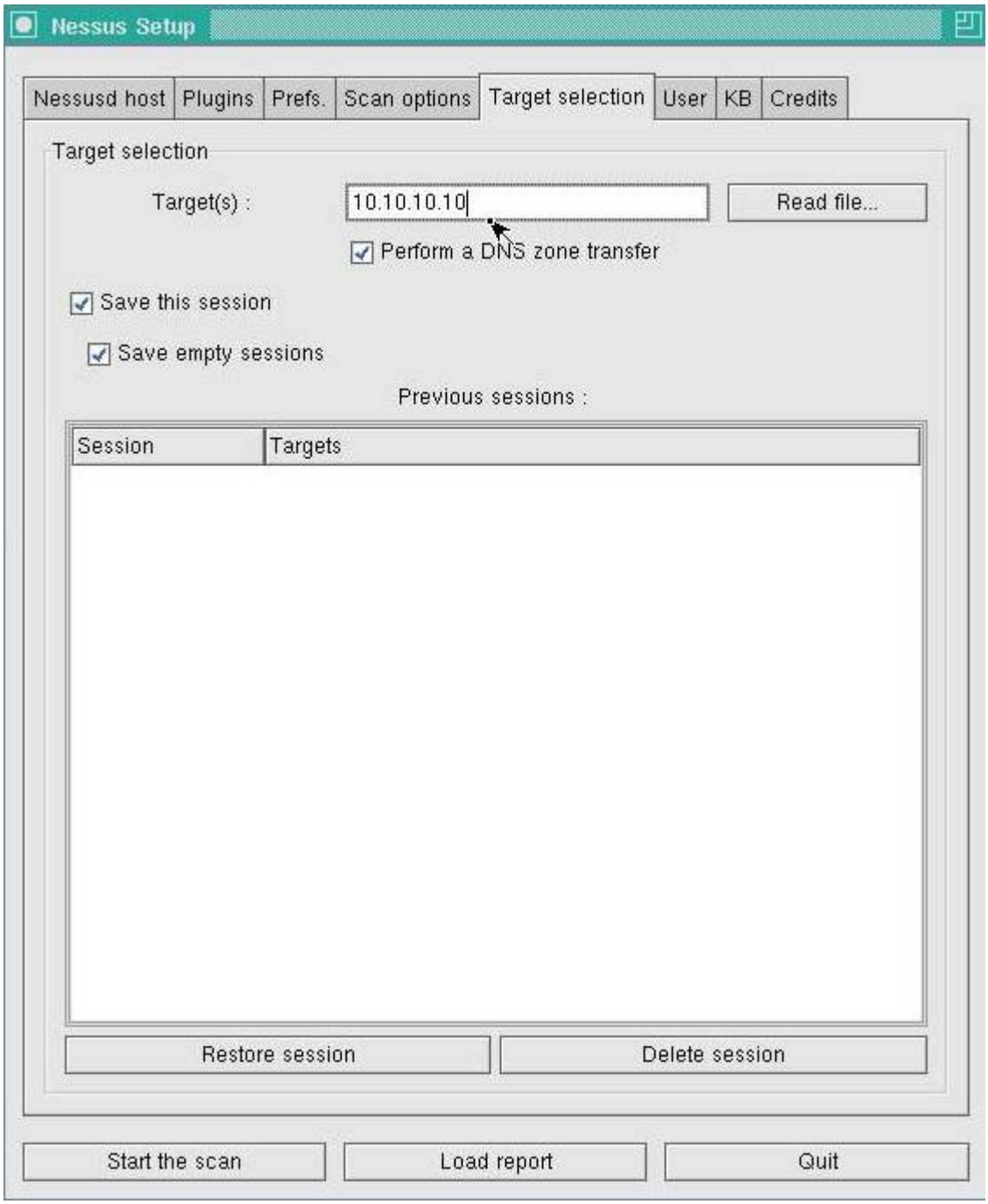


**Figure 10: Specifying Targets in the Unix GUI**

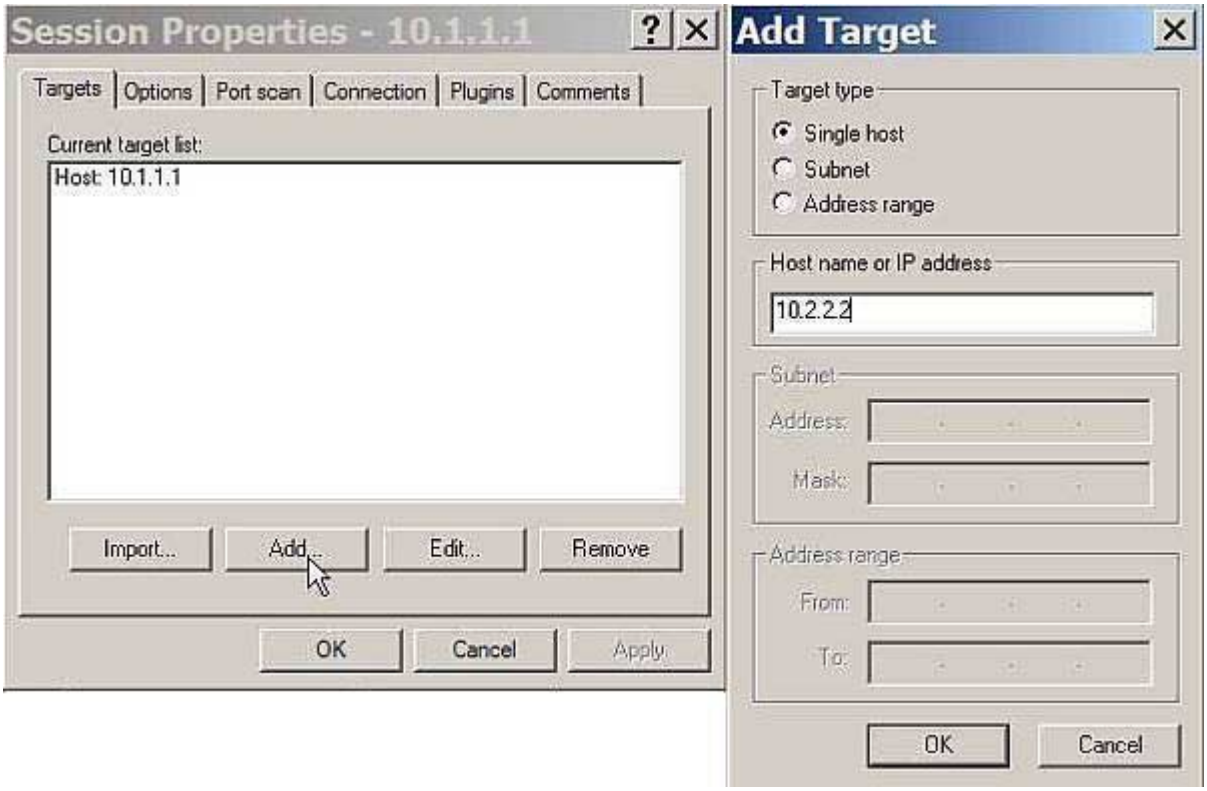


**Figure 11: Target Selection in NessusWX**

### 4.4 Start a scan

With your Nessus client and server in hand you are ready to scan systems. To start a scan in the Unix GUI just click "Start Scan" at the bottom of the window. In NessusWX, right click the desired session and select Execute. Properly used, Nessus can and will pinpoint problems and provide solutions. However, misused it can and will crash systems, cause the loss of data, and possibly cost you your job. As with anything powerful, there comes risk and responsibilities. Scanned systems sometimes will crash. Don't scan any system without permission. I suggest your first scan be against your own isolated test system. Future articles will lead you thorough a scan, sort out false positives and talk about stealth and firewall scanning. Figures 12, 13 and 14 show a scan using NessusWX.

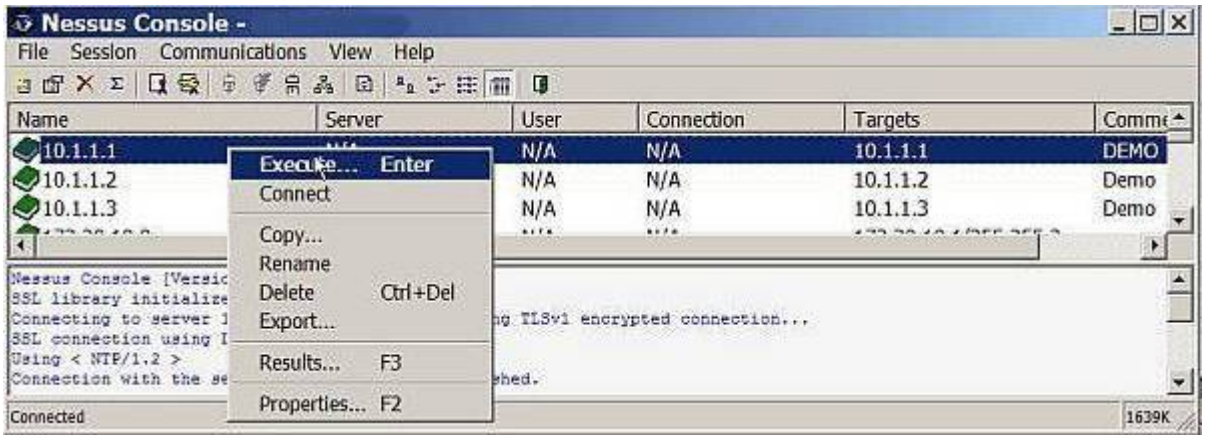


**Figure 12: Starting a scan in NessusWX**

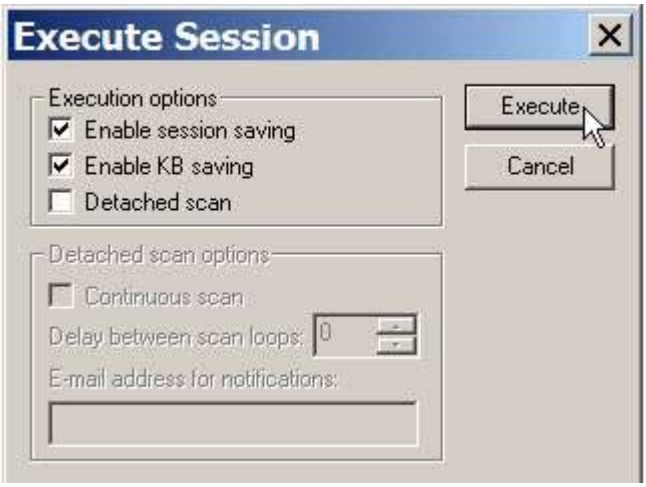


**Figure 13: Starting a scan in NessusWX**

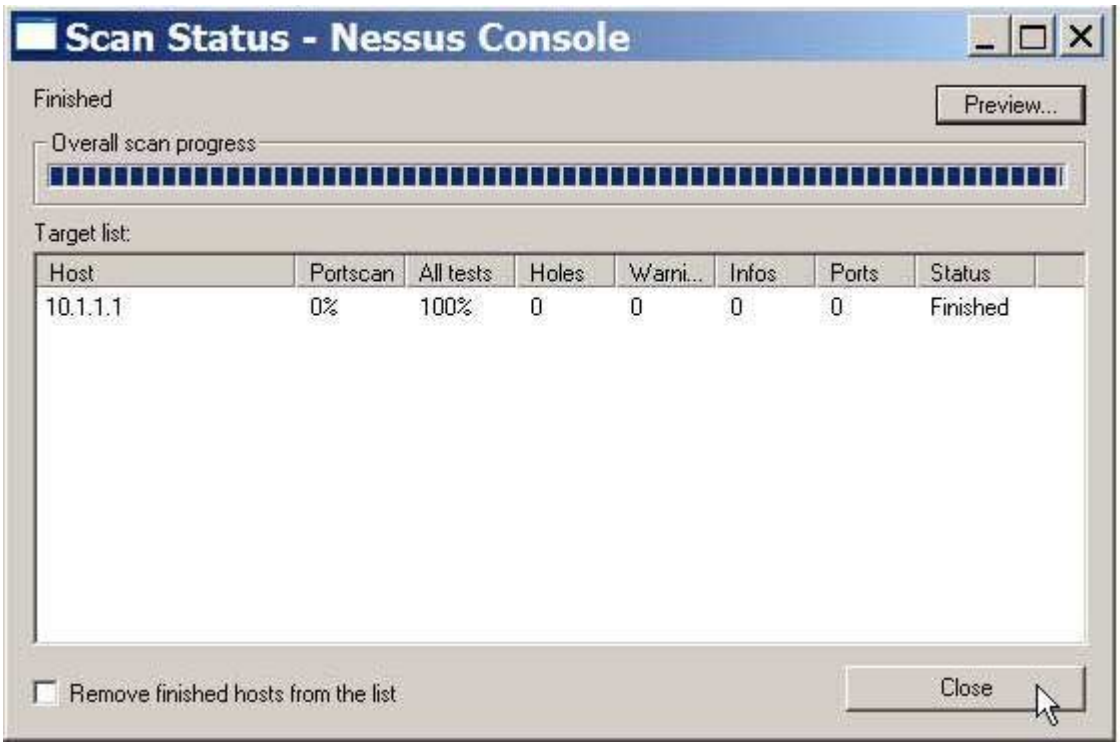


**Figure 14: NessusWX scan in Progress**

## 5.0 Conclusion

Nessus is an excellent tool that will greatly aid your ability to test and discover known security problems. As has been mentioned several times in this article, the power that Nessus gives you should be used wisely as it can render production systems unavailable with some of the more dangerous plus-ins. For more information on Nessus, visit the official Nessus site at www.nessus.org. Happy Scanning!

SecurityFocus™

**Infocus**

**< http://www.securityfocus.com/infocus/1753 >**

# Nessus, Part 2: Scanning

*by* Harry Anderson

last updated December 16, 2003

## 1.0 Introduction

Nessus is a vulnerability scanner, a program that looks for security bugs in software. There is a freely available open source version which runs on Unix. Tenable Security has also recently released a commercial version for Windows called Newt. Boasting over 1200 checks for individual security vulnerabilities, Nessus is a wonderful tool to help track down and eliminate security problems.

This article, the second in the series, will attempt to provide direction through the actual scanning process, general logic and rules of thumbs for parameter choices in different situations. If unfamiliar with Nessus, a reading of the first article will provide needed background information.

## 2.0 Data Gathering

As with many things, it is useful to initially identify the point of view. The scanner's view of the network will influence parameter choices. Possible points of view include a blind (no initial information) scan from the Internet, a somewhat knowledgeable attack from the Internet, a blind scan from a trusted network, or a scan using full system privileges from a trusted network. Each of these fulfills a different purpose, typically falling in to one of the following categories: a system administrator verifying hardening procedures, inventorying vulnerabilities, searching for worms/viruses, or a true penetration test determining if a determined real attack would be successful.

Initially a few pieces of initial administrative data must be obtained. Firstly obtaining the correct target IP address(es) is critical. Mistakenly scanning the wrong IP address is rude at best or at worst could have dire implications on one's personal freedom. A determination of a system's sensitivity to crash is also very important. For example, typically the fallout from a high profile e-commerce server crash would be greater than a crash of a worker-bee's desktop or a test server. For critical systems it is appropriate to obtain permission to scan a system, and written permission is always better then verbal.

Nessus employs a number of methods to reduce the chance of system crash and with care the likelihood of target problems is low. However, due to the enormous number of vulnerabilities tested for and the large number of possible software targets, crashes/ hangs etc can not be completely ruled out. Unexpected crashes usually are due to bad scanning techniques or software freaking out on a port scan. On the other hand since scanning isn't generally well understood, system managers of poorly maintained systems will sometimes try to use scanning as a scape goat for all sorts of problems. Hopefully this article will go a long way to enabling you to make wise parameter choices, producing excellent results with no side-effects and with the confidence to defend against unfair accusations.

| | |
|---|---|
| IP address(es) or subnets of targets | Required |
| Production/non-production | Required |
| Authorized time to perform scans | Required |
| Permission from system owner | Required |

**Table 1: Basic information needed to start a scan.**

## 3.0 Host Identification

When performing a blind scan across a subnet, it helps to know if an IP address is active or not. Traditionally this is done with a ICMP ping. Enabling pings can greatly reduce scanning times. The reduction in scanning time comes from the assumption that an IP which doesn't respond to pings has no system and therefore no further tests will be run against that IP. If pings are not chosen all IPs will be port scanned. The problem is that often, especially on Internet facing systems, traditional ICMP pings are filtered by firewalls or routers, specifically to make systems difficult to find. Also pinging all addresses in a subnet make the scan obvious in firewall and routers logs. Thus generally ICMP pings are only suitable for scanning from the same network as the target. That is, if it is known that there are no firewalls between the scanner and targets. There is also the option of doing TCP Pings. An attempt is made to connect to specific ports to determine if a system exists. This is a good option for quickly scanning an Internet facing firewalled subnet. Usually a common port (such as 80, 25 etc) is being used on all Internet facing systems. If these ports respond a host exists at that address. The options for enabling and configuring ping can be found under the "Port scan" Tab |Configure Button on NessusWX and under Prefs. Tab | Ping the remote host on the Nessus GUI.

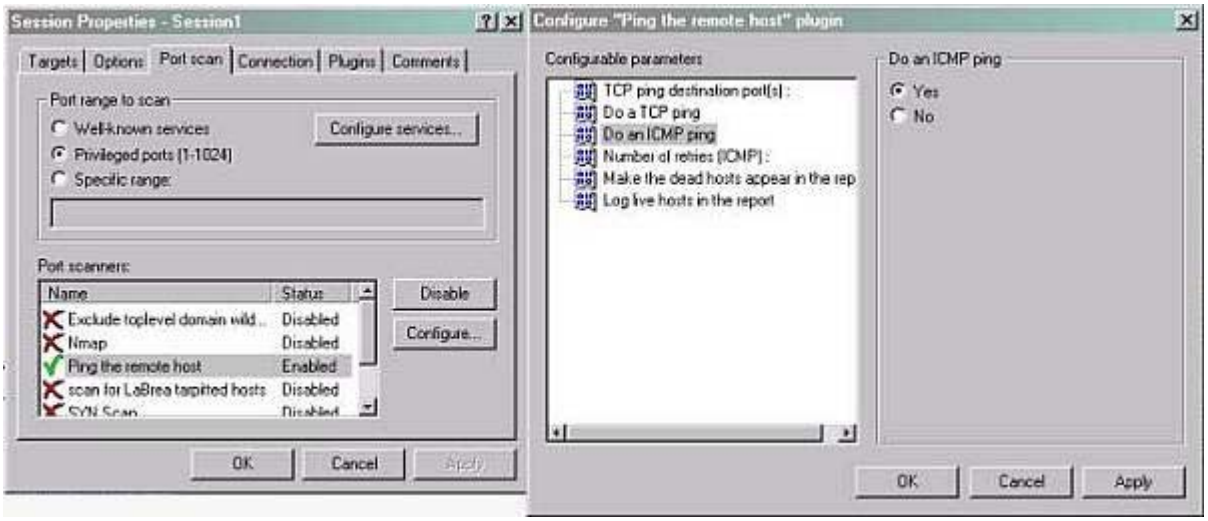


**Figure 1: Enabling Pings in NessusWX.**

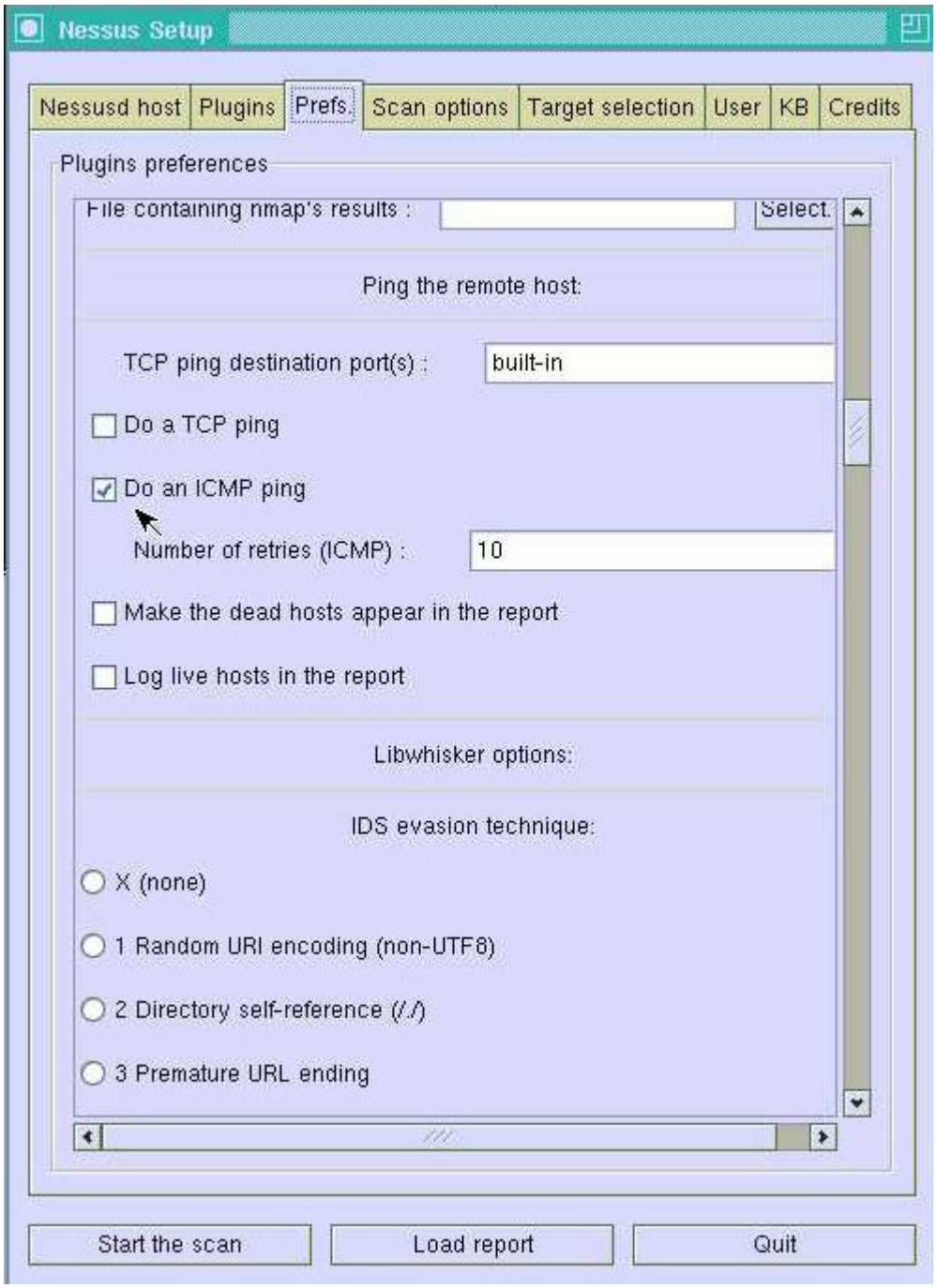


**Figure 1a: Enabling Pings in Nessus' Unix GUI.**

## 4.0 Port Scan

Next the active ports on each IP address must be enumerated. This is done by port scanning. Port scanning seems simple on the surface but is actually a very complex topic. One factor which makes port scanning difficult is the response system. When a port is closed you may receive "this port is unavailable message" in the form of a RST packet but from firewalled ports you may receive nothing.

Stealth, speed, and accuracy are the principal factors to balance when port scanning. The parameters which affect these are the type of scan, timeouts, and what ports to scan. The two most commonly used types of scans are the connect() and SYN scans. There are variations of both in Nessus and in the

optional NMAP component. The native scans have few configuration options and generally work quite well. The NMAP scans provide a wealth of options which provide valuable flexibility for some situations. A connect() scan is the most basic scan; it attempts to complete a connection to each scanned port. A connect() scan is somewhat less likely to crash targets since unlike other scans it tears down the connections it builds. It isn't very stealthy but is fast and accurate. A SYN scan is a bit more stealthy and harder to block since it doesn't complete the connection. SYN scan starts, but doesn't complete the TCP handshake sequence for each port selected. A SYN packet is sent. The port is marked open if an ACK packet is received within the specified timeout. It works well for direct scanning and often works well through firewalls. Another benefit stealth-wise is that a SYN scan looks like a failed connect attempt, thus it won't generate alarms on many IDS and firewalls. If detailed logging is enabled, the connect attempts may be logged but will tend to blend into the clutter.

Timing is an element of port-scanning that can catch one unaware. If scans are taking too long to complete or obvious ports are missing from the scan, various time parameters may need to be adjusted. The basic trade-off is between improving scan speed, dealing with slow networks and accuracy. Scanning firewalled or slow networks can be time consuming. Increasing port scan speed increases the possibility of missing slow responding ports, decreases stealth and increases the possibility of system crash so this should be done with care.

Generally Nessus's built-in port scans work well, but sometimes more control is needed. NMAP's scan can be controlled in fine detail. For convenience several canned timing options are provided. These range from insane to sneaky. Insane is generally too fast to be useful, Aggressive works well for scans across fast LANs. Normal is recommended for most uses. Polite is useful across slow WAN links or to hide the scan. Sneaky is very stealthy but requires some time. Paranoid requires a lot of time. There are a bunch of other scans types and parameters in the excellent tool NMAP which are very useful in specific situations. Fyodor, the author of NMAP, has published in-depth documentation on them at www.insecure.org. NMAP's timing parameters can be changed on the port scan tab | configure button| Timing policy in NessusWX. Select the Perfs Tab to change the NMAP timing policy in the Nessus' Unix GUI.

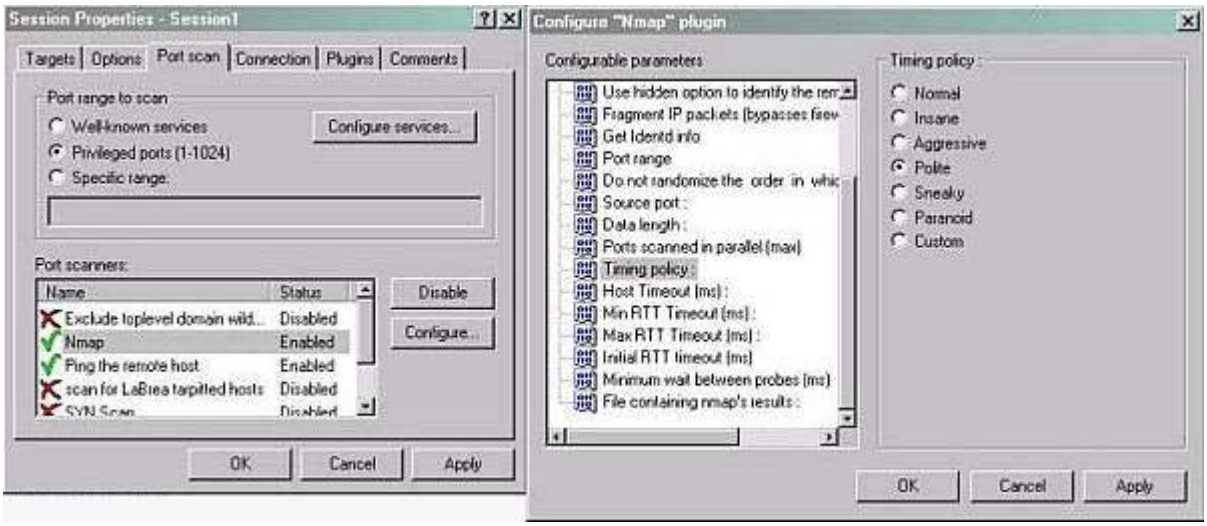


**Figure 2: Changing the port scanning timing in NessusWX.**

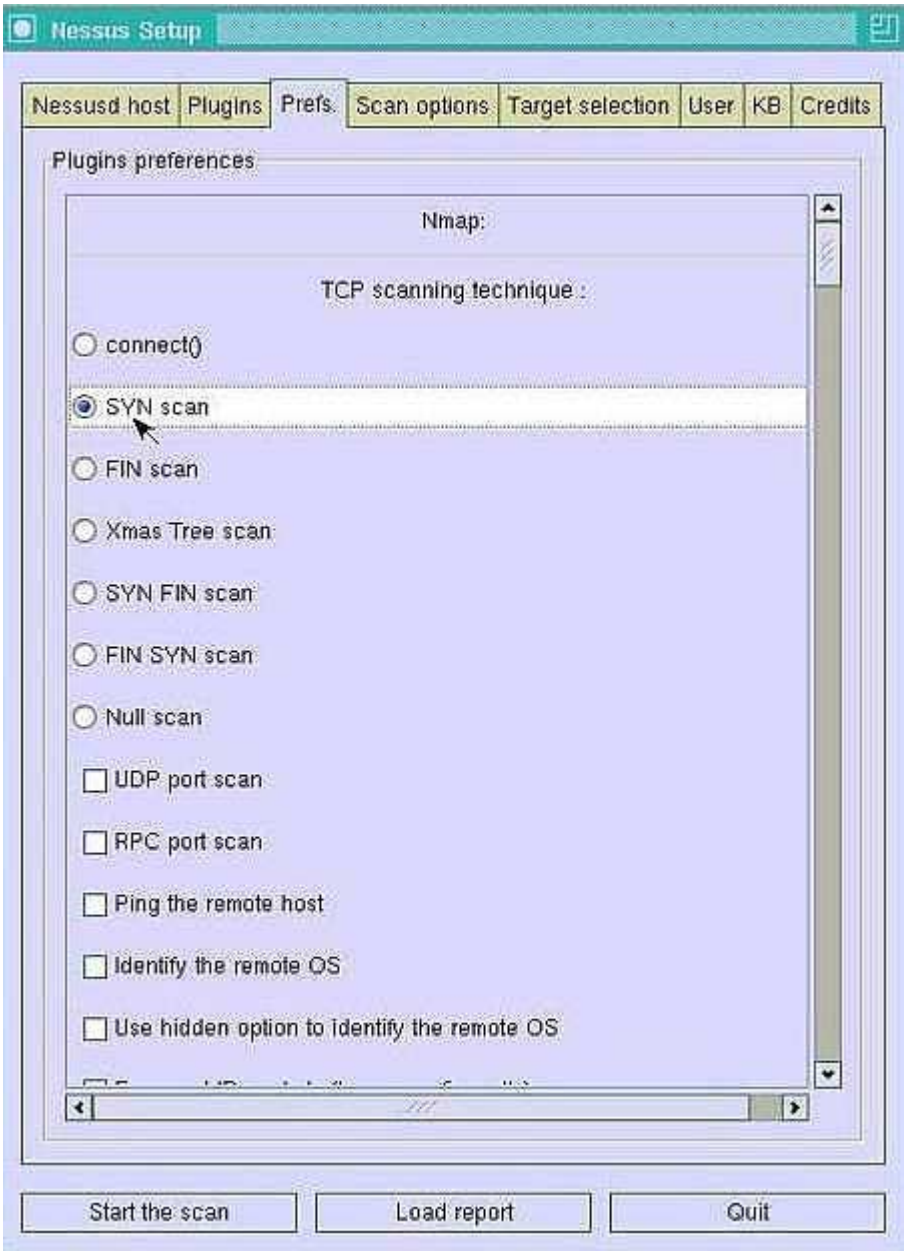


**Figure 2a: Choosing a SYN port scan in Nessus' Unix GUI.**

Since by default Nessus only runs vulnerabilities against ports found to be open, the choice of ports to scan is also a critical factor. Ideally one would scan for all ports but due to time constrains and desire for stealth, this can usually only be done when scanning a small number (< 20) of hosts. When scanning a large number of hosts a small selection of commonly used ports is typically scanned. The default approach is to scan all privileged ports IE TCP ports 1-1024 and the ports listed in the NMAP services file. This would cover all the well-known ports which applications "normally" use and ports that some Trojans use. This approach is a good compromise, only missing applications which use or have been changed to use ports greater then 1024. To speed up scans even more a subset of the well-known ports might be scanned. Scanning ports HTTP 80, 8080, Windows 135, 139,445, HTTPS 443, FTP 21, SMTP, SNMP 161 would identify ports that most often have vulnerabilities associated with them. This is a good option when time is critical and a large number of hosts must be analyzed. This has thus far assumed a broad based scan, if the scan is looking for a specific vulnerability (for example MS03-39 the Blaster vulnerability) only the ports associated with the vulnerability (for Blaster 132,139

and 445) would need to be scanned.

So far we have only talked about scanning for TCP ports, UDP scanning is possible as well. Fewer known vulnerabilities use UDP, UDP ports are difficult and time consuming to scan, thus currently UDP isn't commonly tested. NMAP does include a UDP scan and there are a number of vulnerabilities tested for that are associated with UDP ports.

| | Comments | Internal | External | Stealthness | Speed | Accuracy |
|---|---|---|---|---|---|---|
| ICMP Pings | Use if no firewall between server and target | Suggested | Not Suggested | Might be flagged by IDSes or honepots | Speeds up scans | Very accurate as long as pings are not blocked |
| TCP Pings | Use of firewalled systems if some known ports are being used. | Not Suggested | Suggested | Less obvious then ICMP Pings | Speeds up scans | Systems without chosen ports will be missed. |
| Sync Scan | Provides flexible timing options | Suggested | Suggested | Tends to blend into clutter on Logs | Dependant on speed settings | Very accurate |
| TCP Connect () | Less likey to crash systems | Works well | Works well | Obvious in IDS Logs | Slightly slower than SYN | Easier to block than SYN |

**Table 2: Port scanning rules of thumb.**

## 5.0 Plug-in selection

Plug-in selection is the second big feature in scanning. Due to the large number of options, the task of choosing plugins seems overwhelming at first but Nessus does a lot of the work for you. After Nessus performs the port scan it runs the services plug-in which identifies which server program is running on each found port. Based on the service plugin output Nessus chooses the appropriate plug-in subset to be run from the user-specified plug-in set. The services plug-in does a fantastic job of identifying the actual program bound to each port. Rarely is it ever wrong. This does not depend on the port being a well-known port; it actually analyzes responses from the host.

This identification step is necessary because often services are run on non-standard ports. IE often a web-server may be run on alternative ports such as 8080 or 4983 instead of the expected port 80. Or possibly even more sinister some Trojan is running on port 80 in order to bypass Firewall protections. The services plug-in will identify these and dynamically activate the plug-ins necessary to test that service.

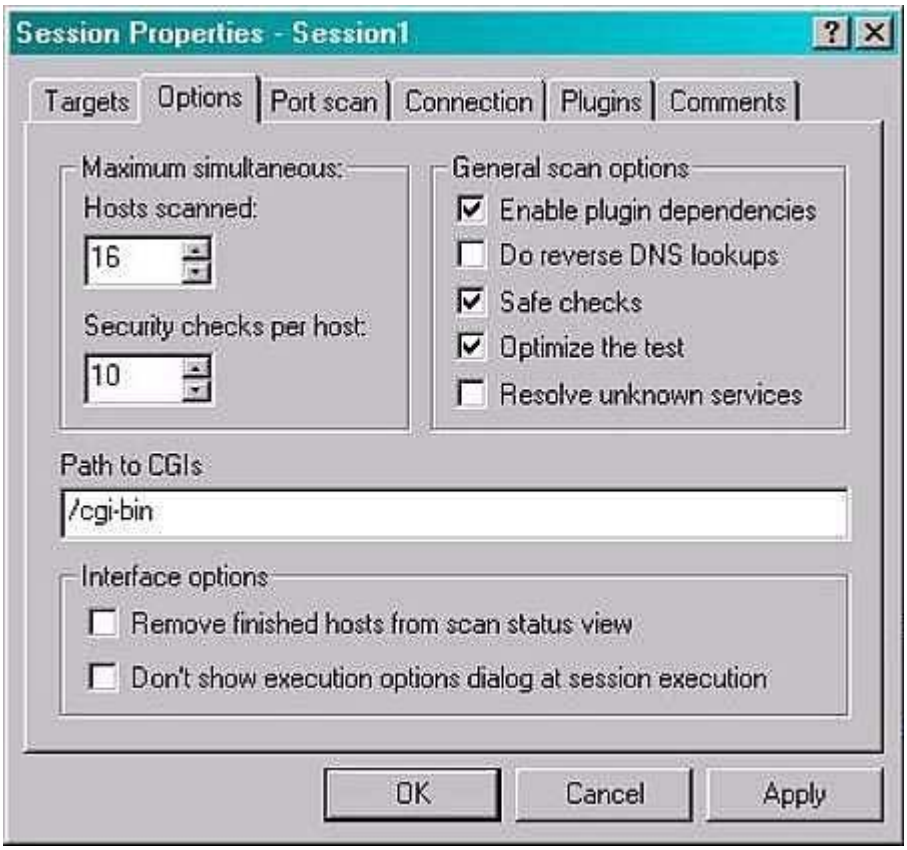


**Figure 3: Enabling plugin dependency in NessusWX.**

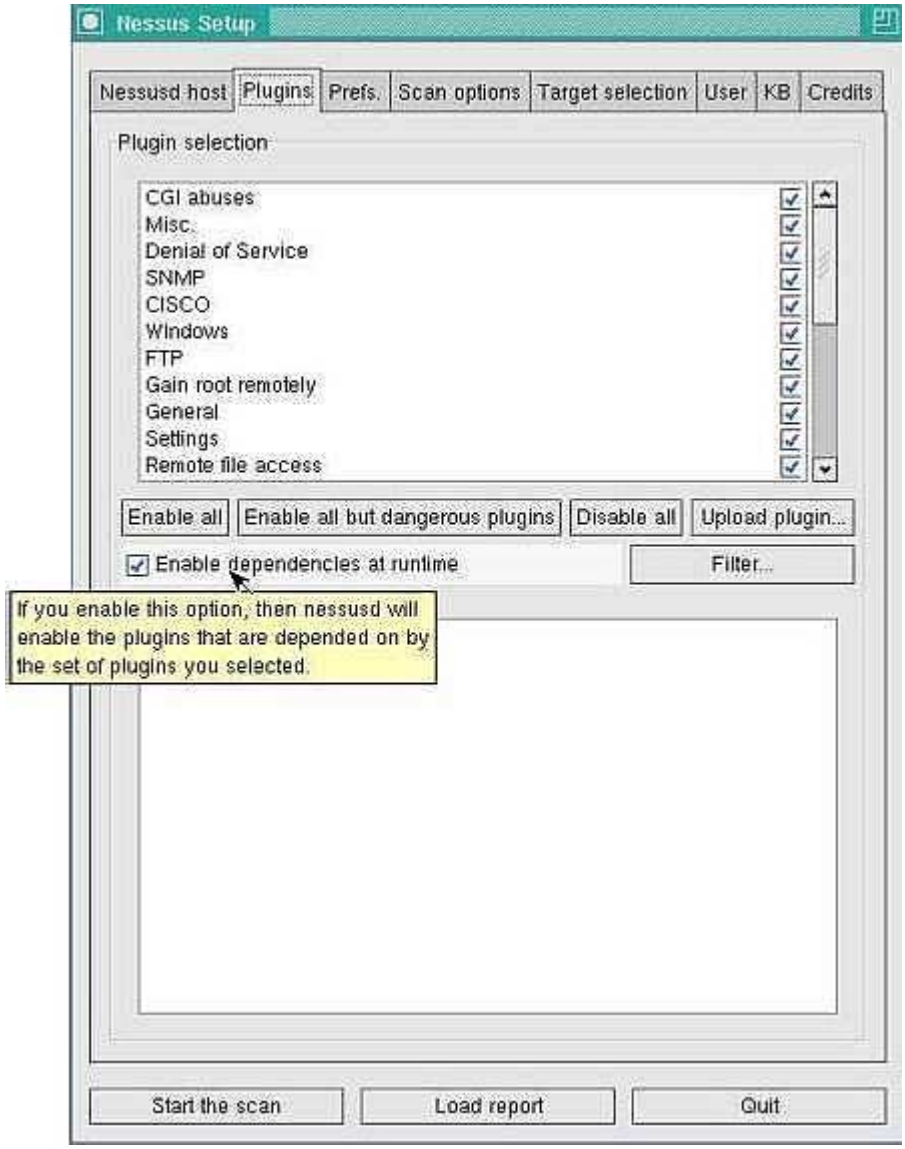


**Figure 3a: Enabling plugin dependency in Nessus' Unix GUI.**

The most obvious plug-in choice is dangerous plug-ins and safe checks. Dangerous plugins test for vulnerabilities by attempting to DOS (Denial of Service) the machine. Safe checks tests for DOS vulnerabilities through passively gathering info such as software version. Safe checks can generate false positives due to the varieties of possible patches and workarounds that may fix a particular vulnerability. Since Internet connected systems are prone to see almost any attack, they should be checked for DOS vulnerabilities using the dangerous plugins. Within an internal network the chance of a live DOS is lower, and the risk of running the dangerous plugins may outweigh any potential gain. Often the best approach is to run a safe-check enabled scan, investigate/fix all problems, then in a maintenance window scan using dangerous plug-ins and safe-checks disabled.

Safe-checks can be found on the scan options tabs in the Nessus' Unix GUI, and on the Options Tab in NessusWX. The dangerous-plug-ins choice can be found on the Plug-ins Tab | Select Plugins in NessusWX and on the Plugins Tab in the Nessus' Unix GUI.

Some Nessus plug-ins (especially some Windows ones) need administrator access to analyze the system more in-depth. The windows registry may be queried looking for virus infections or patch

levels, etc. For example the plug-in which detects the Blaster worm (plugin ID 11818) looks for the registry key that Blaster installs to identify infections. The administrator username and password should be entered in the Plugins TAB | Configure Plugins Button | SNB Password and SMB Account on NessusWX, and the toward the bottom of the page on the Perfs Tab in the Nessus' Unix GUI.

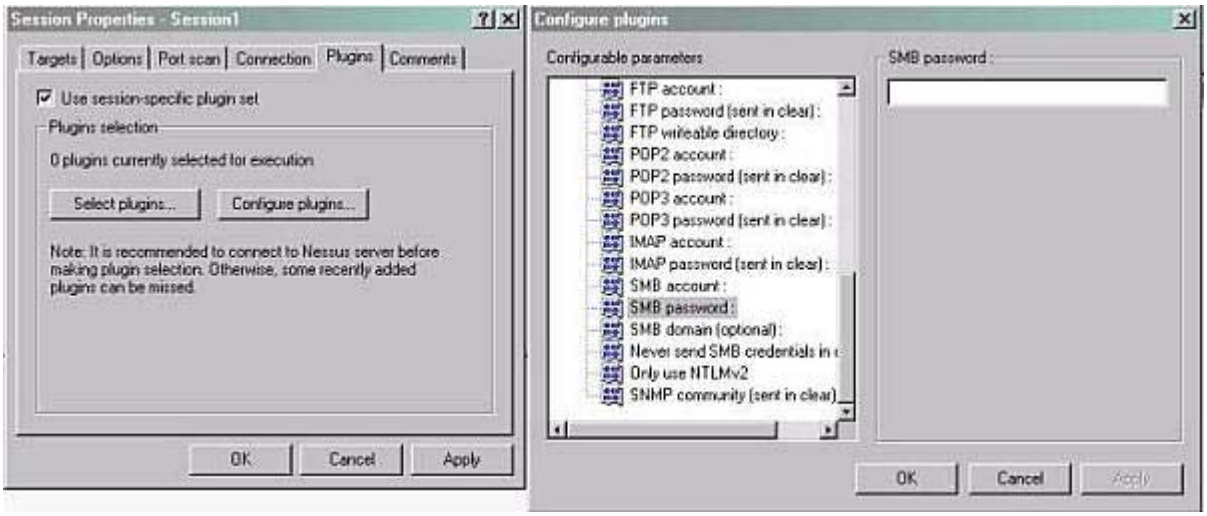


**Figure 4: Entering a Windows Username in NessusWX.**

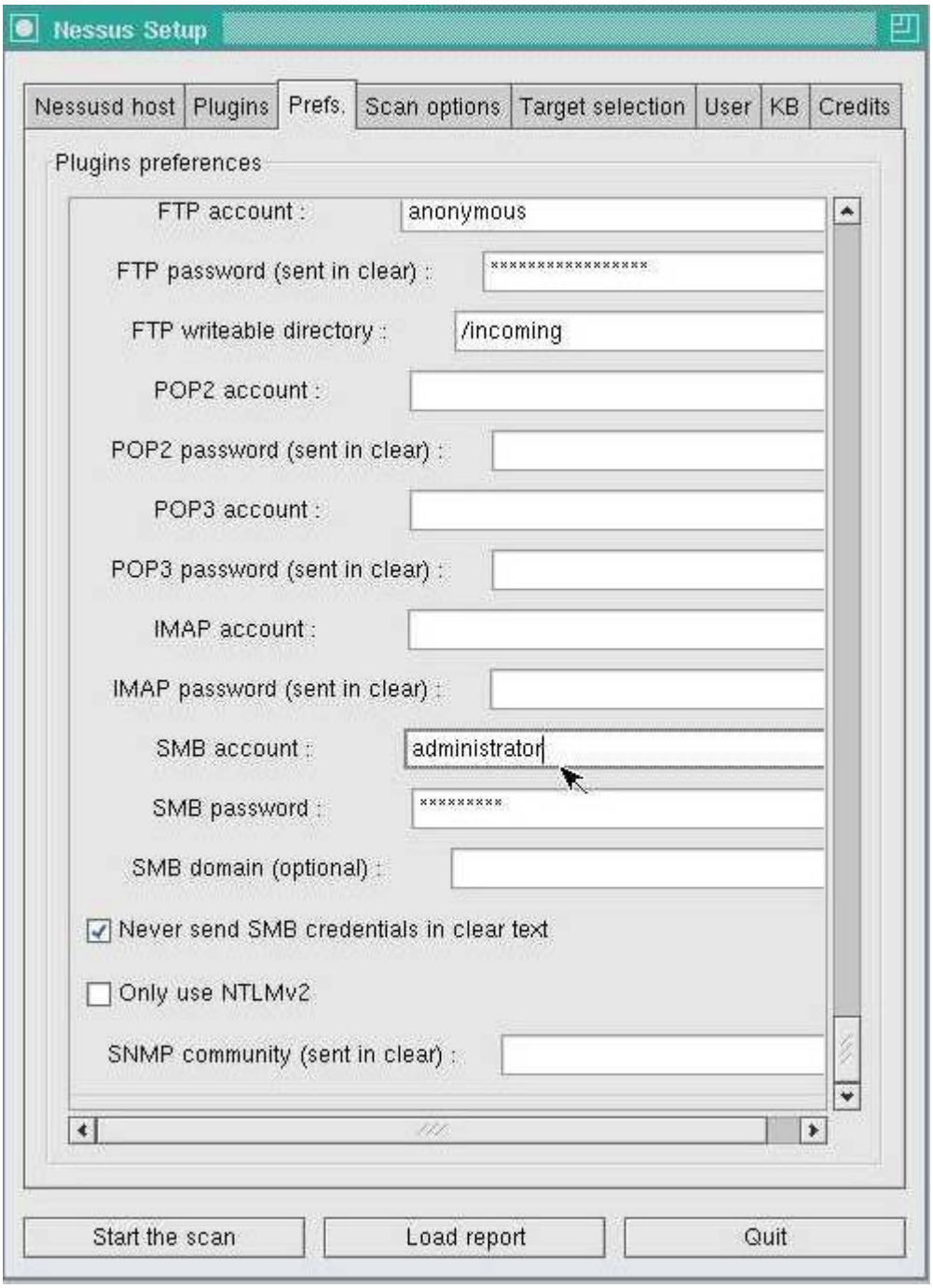


**Figure 4a: Entering a Windows Username in the Nessus' Unix GUI.**

It may be necessary to hand-pick specific plug-ins. Possibly you just need to check for systems with the Apache Chunked Vulnerability. In this case there is no reason to select 'run the plug-ins'. Or perhaps you run a Linux only shop so there is no reason to include the Windows plugins.

## 6.0 Other Issues

Although not common, honeypot deployments are increasing. A honeypot is a fake system designed to slow or confuse a scanner. For example, a Labrea tarpit hangs any attempt to connect to an unused IP address. It will add lots of false nodes, significantly slow a scan and/or cause it to fail. Although not a scan preventer, when properly deployed it is an effective damage control defense against worms and

network scans. Nessus has a scan that will identify Labrea systems. I suggest including it in any scans against unknown/blackbox networks. This will alert you to the cause if the scan is slow and producing strange results. The only real mechanism to workaround a honeypot is by segmenting large scans and manually weeding out the fake nodes. The Labrea scanner is choosen under the port scan tab under NessusWX and Under scan options | Port scanners on Nessus' Unix GUI.

One other aspect that should be considered is server placement. If testing is being done against an organization's Internet presence, an unencumbered (no firewall or router ACLs in between) connection to the Internet for the Nessus server is crucial. On the other hand if internal security is being tested a well-placed connection on the internal network is required. These placement issues may also profoundly alter the scanning choices already discussed.

How long should the scans take? This is a difficult question since there are so many factors involved, but an excellent general run-down on approximate scan times is included in the excellent open source vulnerability test manual.

### 7.0 Rules of thumb

Below is a rules of thumb chart for scanning. Rules of thumb can be handy because they quickly place you in the correct ball-park. However, they are general in nature and don't fit specific situations. Remember that the rules of thumb are just that. Some of them may overlap but generally are the best choice for the given situation, however don't just blindly accept them; use them as a guide and think for yourself.

| Situation: | Comment | Safe-Check | NON-Dangerous plugins | Ping | Type of Port Scan | Ports to Scan | Features |
|---|---|---|---|---|---|---|---|
| New system not in production | Best time to test, system can't be hurt | Disable | Run All Plugins | Yes | TCP SYN | 1-1024 and ports listed in NMAP services files | Comprehensive, slow, crash likely. |
| High-Visibility Production system | Get Maintenance window.<br><br>Follow-up with a Dangerous Plug-ins scan | Enable | Disable | Yes | TCP Connect | Predefined subset 80, 8080,135, 139,445, 443, 21, 161 are good choices | Crash unlikely. |
| Low-Visibility<br><br>Production system | Follow-up with Dangerous Plug-ins scan | Enabled | Disable | Yes | SYN scan | 1 -1024 and NMAP services ports | Crash unlikely, slow. |
| Internet | Likely | Depends | After initial | No | SYN scan | 1 -1024 | Slow depending |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| based Host | firewalled. | | safe checks scan | | Sometimes connect() scans work better with some firewalls. | and NMAP services ports | on firewall configuration. |
| Quick scan | | Enable | Disable | Yes | SYN scan | SYN Scan.<br><br>Ports: 80, 8080,135, 139,445, 443, 21, 161 | Quick, catches common problems. |
| Stealthy scan | | Enable | Disable | No | SYN scan<br><br>Choose a slower NMAP timing option | SYN Scan.<br><br>Ports: 80, 8080,135, 139,445, 443, 21, 161 | |
| Comp-rehensive report | | Disabled | Enabled | No | SYN scan | NMAP TCP and UDP scan | |

**Table 3: Scanning Rules of Thumb.**

## 8.0 Conclusion

After a scan is completed a report is generated which lists each found vulnerability. If the scan was well designed the report will be very complete and accurate. However, the sheer volume of data produced can be intimidating at first. In the last article in the series, we will discuss how to interpret the final report, eliminating any false positives and finding a solution for each vulnerability. Happy Scanning!

**Author Credit**

View more articles by Harry Anderson on SecurityFocus.

SecurityFocus™

**Infocus**

**< http://www.securityfocus.com/infocus/1759 >**

# Nessus, Part 3: Analysing Reports

*by* Harry Anderson

last updated February 3, 2004

## 1.0 Introduction

This article, the last in the series about Nessus, will endeavor to explain a Nessus report and how to analyze Nessus is a vulnerability scanner, a program that looks for security bugs in software. The first article explaine how to install Nessus and a basic overview of features. The second article gave general rules of thumb for various scanning situations. It is suggested that you review the first two articles before reading this one.

Understanding how and why vulnerabilities exist in software is important to being able to analyze the final sc report, and this article uses a number of examples to illustrate various important concepts. In some cases these may be older, more common examples instead of the "latest" day zero vulnerabilities. There are severa reasons for this. First, I don't currently know of any awe inspiring "zero day" vulnerabilities to wow the reade with. Second, the stated examples typically have a large body of information available for the reader to do further research, and I will list a few sites providing a good start for such research. Third, although some of these examples may be older, they are still quite common and unfortunately most readers will still encounter them when scanning.

## 2.0 Report generation

In the last article we covered the scan process. Once the scan has been completed the results need to be analyzed. These results can be directly viewed within a report generated with the client, either Nessus GUI (Unix) or NessusWx (Windows), or by having the data exported for analysis in an external program. If viewed from within a Nessus client, the results are arranged in a hierarchical fashion by the host. In NessusWX the scan results can be obtained by selecting the session and choosing Session | Manage Results. The results car then be viewed using the VIEW command, or written into a plain text (.txt), HTML, or Adobe Acrobat (.pdf) report. In the Unix GUI the results are automatically displayed upon completion of the scan, as shown below Figure 1. In NessusWX, sample results are shown in Figure 2.

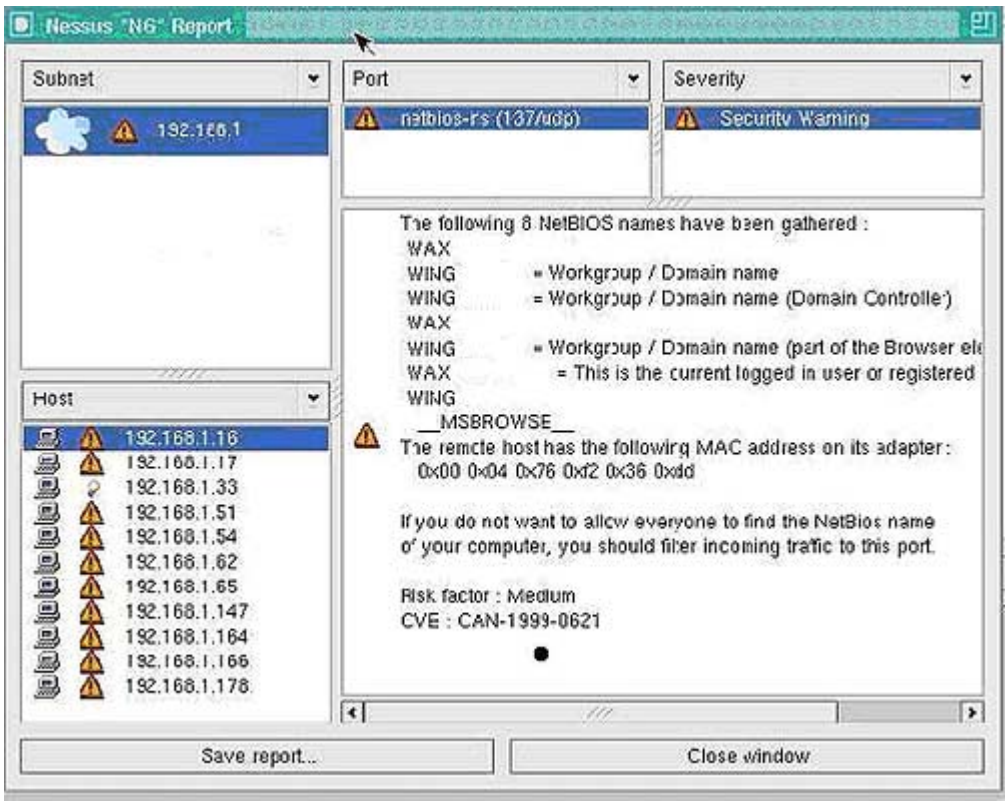


**Figure 1: Results are automatically displayed in Nessus GUI.**

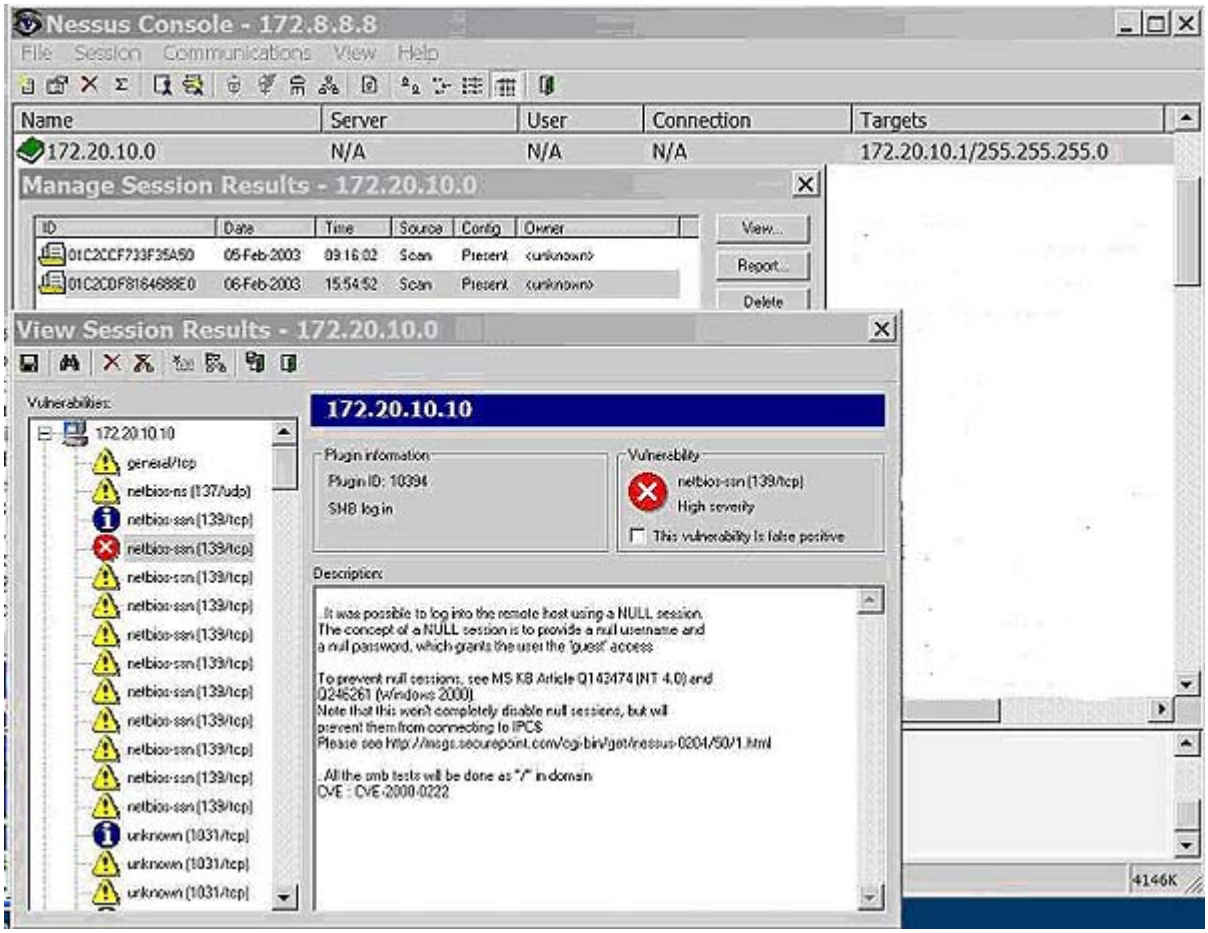


**Figure2: Viewing results in NessusWX.**

If a large number of hosts are being dealt with, the export functions can be very convenient. In NessusWX, results can be exported to CSV for importation to a spreadsheet, extended NSR, NSR and NBE for re-importation by any Nessus client, or into SQL files for importation into a SQL database (including MySQL). Each line item found will be exported as a single record. By importing the data into a spreadsheet or databas sorting and queries against various fields can be easily done. This gives one the ability to sort or group items by vulnerability, thus allowing one to deal with a single vulnerability at a time across many hosts. In NessusWX the scan results can be obtained by selecting the session and choosing Session | Export. In the Ur GUI, from the automatically displayed results screen, Save Report is chosen to export the data. The Unix GUI can save the data in a few more formats then NessusWX. The GUI saves in the depreciated NSR format (equivalent of the NSR format in NessusWX), as well as XML, HTML, NBE (equivalent to the extended NSR format in NessusWX), HTML, LaTeX, ASCII (similar to plain text in NessusWX) and HTML with Pies and Graph One difference to note between NessusWX and the Unix GUI is that NessusWX automatically saves the result when finished. The Unix GUI does not and the results will be lost upon exit if not saved.

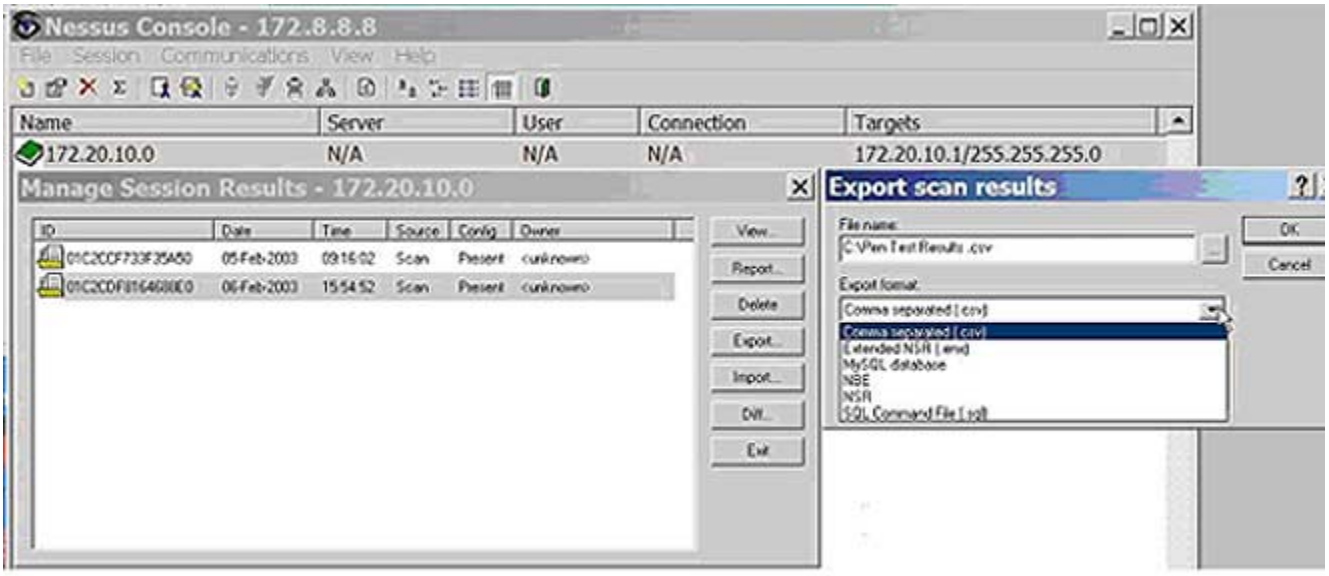


**Figure 3: Image of exporting results in NessusWX.**

## 3.0 Identifying false positives

Once the results are in a format you are prepared to deal with, the real work begins: analyzing the results an proposing solutions. In a perfect world this would be simple but due to the myriad combinations of software and configurations, remote black box vulnerability scanning is difficult and an inexact science.

One problem is that sometimes Nessus just gets it wrong. This is the case not very often but it does happen. appropriate techniques are used (many of which are discussed in the second article of this series) the accura can be increased, but for the foreseeable future a human will still need to pass final judgment. Sometimes vulnerability reports can be confusing due to the many possible combinations of software and configurations involved. Although every possible way that Nessus may throw a false positive or confuse the reader can't be enumerated, many of the reasons why these things happen and the signs to look for can be discussed. Nessu typically gives you a false positive due to one of two reasons: either the plug-in is only testing for a software version number or the results are unexpected but still valid.

Testing for a given piece of software's version number only may be desirable but can also create issues that must be reviewed. Nessus has a safe-check option which changes any dangerous (but safe-check enabled) plug-ins to just check for the version number. This is a good compromise for safety, but with the current stat of software configuration management, the safe check option sometimes raises false positives. There are several causes for this. First, the software may be reporting the version number of a vulnerable version but a patch or workaround has been applied. This is frequently the case with various Linux systems. The major distributions or manufacturers such as Red Hat, IBM, etc., may patch the code and not change the version number.

Version number checking may also be misleading if the vulnerability is present but not accessible due to a workaround or best practice. A good example of a best practice bypassing a vulnerability is the IIS Unicode (MS00-078, BID 1806) vulnerability. This vulnerability will execute system commands through specially formatted URLs. By following a commonly applied best practice of placing the OS and application software on separate disk partitions, this vulnerability is rendered useless. The IIS software itself would still be a

vulnerable version and ideally should be patched, but the system itself would not be vulnerable to the hole. Note that this example is not affected by the safe-check option in Nessus.

All safe-check enabled items bear a warning that they may give a false positive due to the fact that only version numbers are being checked. When this warning is displayed, often a bit of research is required to determine if the system is actually vulnerable. One approach is to check the manufacturer's release notes to see if they have fixed the problem in that version of software release. Possibly by researching the vulnerabilit it can be determined if the vulnerability is or isn't present on a particular OS version. In the end it simply ma not be possible to conclusively determine if a system is vulnerable without access to the system, or by using non-safe check enabled Nessus scan.

Sometimes false positives also occur due to unexpected but otherwise valid results being sent back. Nessus takes into account many possible responses but it is impossible to delineate all of them. A common example this is the large number of web vulnerabilities (usually port 80) that are sometimes falsely reported when scanning a web-server. The issue here is that Nessus isn't receiving what it expects to receive. A plug-in testing a web vulnerability will typically expect to receive an "HTTP 404" error response if the requested page isn't found. However, although it is technically against the HTTP standard, it is becoming increasingly commo for webmasters to use "custom 404 pages". These "custom 404 pages" aren't really "HTTP 404" error messages but typically a "pretty" page stating that the desired data was not found. Plug-in 10386, the "No 4( check" goes a long way to filter out these false positives by looking for common responses to page requests and correlating responses that are "custom 404s". Typically this works well. However if the web-server returr a custom page containing random or changing data, Nessus can not recognize these as a "custom 404". A telltale sign of such an issue is the large number of web-server vulnerabilities for both IIS and Apache (if the plug-ins for both web servers are being used). Verifying these is time-consuming but easy. Simply browse to the page reported to be a problem. You may have to reconstruct the problem URL from the vulnerability report. If a 404 type of custom page is displayed, it is a false positive. Otherwise the stated problem most likely does, in fact, exist.

Printers, UPSes and other hardware based devices occasionally flag a number of semi-false positives on any report. Stripped down versions of Unix/Linux and Apache are usually used for these devices and are often burned onto the device's read only memory (ROM). Of course these versions will still have whatever vulnerabilities that are later found in the software. However, since there is often little accessible memory and available resources, typically these vulnerabilities are unusable. Even if the vulnerability is exploitable, the ris may be low. Practically, how much damage can an intruder do compromising the typical printer? Printers can be identified by examining the OS discovery line item in the report.

### 4.0 Suspicious signs

Now that we know some of the reasons false positives may be generated, how do we recognize them? One suspicious sign is inconsistent results, especially with the reporting of a vulnerability for a software package that doesn't seem to reside on the target. Suppose a target is a identified as Windows based and yet an apparent Unix based vulnerability is found. This should raise suspicions, however the item should not be immediately written off. There could be three possible explanations: the item is false, the OS and software detection is wrong, or the problem software has been unexpectedly ported to this operating system.

While the OS and software detection in Nessus isn't perfect, it does quite well with common OSes. However, Nessus sometimes misses it when there is a variation from what is expected or when it finds an odd device. I addition, sometimes system administrators purposely change parameters specifically to confuse OS detection software. In today's computing environment more and more cross-platform development is being done, and many of today's software packages have been ported to a different OS or are used as part of other programs

Sometimes it is surprising how vulnerabilities can show up in ports across different platforms, so don't be hasty about writing off an item in Nessus' report without proper research. One example of such a surprising port is the recent system compromising OpenSSL vulnerability (BID 8732). Although usually referred to as a Unix based vulnerability, this software resides at such a low level in most packages it can also show up on Windows and Cisco environments as well. All that is required is a port of a program that uses the right versio of OpenSSL.

### 5.0 Areas of potential confusion

False positives aren't the only issues to deal with in the report, as sometimes the correct results themselves are confusing. Confusion may stem from the result of similar vulnerability notifications. In one instance, plug in 11424 advises against using the IIS WebDav component. This isn't a vulnerability but simply a suggested best practice. Plug-in 11412 tests for the much more famous and more dangerous, system compromising WebDav Overflow (MS03-007, BID 7116). Ironically this vulnerability has little to do with IIS and doesn't eve require WebDav. WebDav was just the first of several delivery mechanisms discovered for the hole. These tw items exist totally independent of each other, however in a quick reading they may be confused with each other as similar.

Another area of potential confusion is when the same named vulnerability produces different results in different environments. The Apache chunked encoding vulnerability (BID 5033) is a good example. Apache chunked encoding is actually two separate, highly related vulnerabilities. One is a DOS and one is a system compromise. The ancillary software on the target determines the result. If safe-checks are disabled, Nessus tests for this hole by sending enough junk to the server to trigger the problem. If the connection drops there a problem. If the system compromise aspect is present, the server will start executing the uploaded data which overloaded the buffer. If the DOS aspect is present the web-server thread, which was overflowed, will crash. Nessus assumes that if the server stops responding that the vulnerability is present but it does not report if the DOS or system compromise aspect is active. If safe-checks are enabled, only the version of apache is checked.

One other area of confusion worth mentioning is the set of a few vulnerabilities to which no exploit, patch or workaround exists. These can be identified by the Nessus solution that states, "Contact your vendor for a patch." Although these vulnerabilities truly exist they are general in nature and the risk factor is low. Also, since these vulnerabilities are generally not acknowledged by the software manufacturer there is normally litt that can be done to conclusively solve them

### 6.0 Informational items

Nessus also generates a number of informational items in its reports. Typically a report will include at least tv General/TCP informational messages per host. One will document the current traceroute information to a hos and the second will make an educated guess as to the OS of the remote host. By comparing the target's fingerprint of various network-oriented parameters against a library of known values, this determination is accurate, especially for common OSes. For less common OSes usually it identifies at least the manufacturer correctly. If the fingerprint gathered does not exist in the library and the user knows the OS, the program requests the user to submit the signature and OS type to os-signatures@nessus.org. These submissions help to improve Nessus's OS detection facility for future use.

### 7.0 Evaluating risk

Evaluating the risk that a vulnerability might occur is another matter. Since the risk level typically determines the attention given to a problem, risk level is a important quality in the report. Nessus classifies each

vulnerability as a Note, a Warning or a Hole. The risk is further identified in the plug-in summary as none, lov medium, High, Serious and Critical. Unfortunately, these classifications aren't well defined and have been subjectively applied over the years; therefore the definitions are somewhat inconsistent. There is a website available that lists some in depth statistics on Nessus's risk classifications.

If you are fixing the problems yourself, the built in classifications and risk level is probably sufficient. Howeve if you are passing the results on for correction by others a better defined scheme would be helpful. Unfortunately, unlike many other industries there are no accepted computer security definitions for rating risl The closest is the suggested definitions in the Open Source testing manual OSSTM (PDF document).

Another classification scheme that bears mention is the FBI/SAN's list of "The Twenty Most Critical Internet Security Vulnerabilities". This specification is valuable because it is fairly well known and can sometimes brin needed attention to a languishing problem. However, the list is fairly broad and it actually lists around 100 "Top" vulnerabilities and some of these are not as serious as ones which have been left off.

Most risk rating systems for vulnerabilities are based on risk relative to a target's operation and not the operation of the business. This simplifies the determining of risk but may be misleading to non-technical user since it is divorced from risk to business processes, which business types tend to better understand.

## 8.0 Finding a solution

Once there is a good list of found vulnerabilities, solving them is the ultimate goal. The person generating the audit report may also be the one in charge of remediation of problems, however this isn't always the case. Remediation is often passed off to others. Irregardless it is important for someone to know how to solve the problems that have been found. Some vulnerabilities are simple and obvious to solve, and the Nessus report will often include a link to a patch or a reference to a patch or workaround.

Sometimes a solution is a bit more elusive and doing some research becomes necessary. When available, Nessus provides some useful reference numbers. Chief among these are the Bugtraq ID (BID) and Common Vulnerability Exposure (CVE) numbers. The BID number is a reference number generated upon submission of vulnerability to the Bugtraq security list. These are also the reference numbers in the searchable SecurityFoc vulnerability database. This database lists discussions on why various vulnerabilities work, lists published exploits, vulnerable versions and solutions as well as a host of other data on specific vulnerabilities. This database is arguably the most complete collection of vulnerability data available.

The CVE number provides the key to the Common Vulnerability database run by Mitre. Although some vulnerability info is stored in the CVE database, its primary purpose is to differentiate and identify vulnerabilities. This is similar to a large vulnerability Rosetta stone, providing connection between various vulnerability databases. Although the CVE database is very useful, sometimes it is a bit frustrating. Many of the vulnerabilities are still in the Candidate stage, which means that their data isn't completely verified, even though they have been around for quite a while.

Once a solution has been found and applied, good practice dictates that the target be scanned again. Unfortunately sometimes patches do not update properly, workarounds do not work or open up other holes. Thus, a follow-up scan, although sometimes not possible, is certainly a good conclusion, verifying that the target is no longer vulnerable.

## 9.0 Conclusion

Analysis is always the tough part of any exercise. Pressing the buttons to start a scan is easy, but understanding the result is hard. Hopefully, now you understand some of the complexities of the scanning process and can start to be able to spot the significant problems out of the tangle of information Nessus can present. This the final article in the series of Nessus articles. Hopefully they have equipped you well for those long nights of scanning ahead. Happy scanning!

---