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.

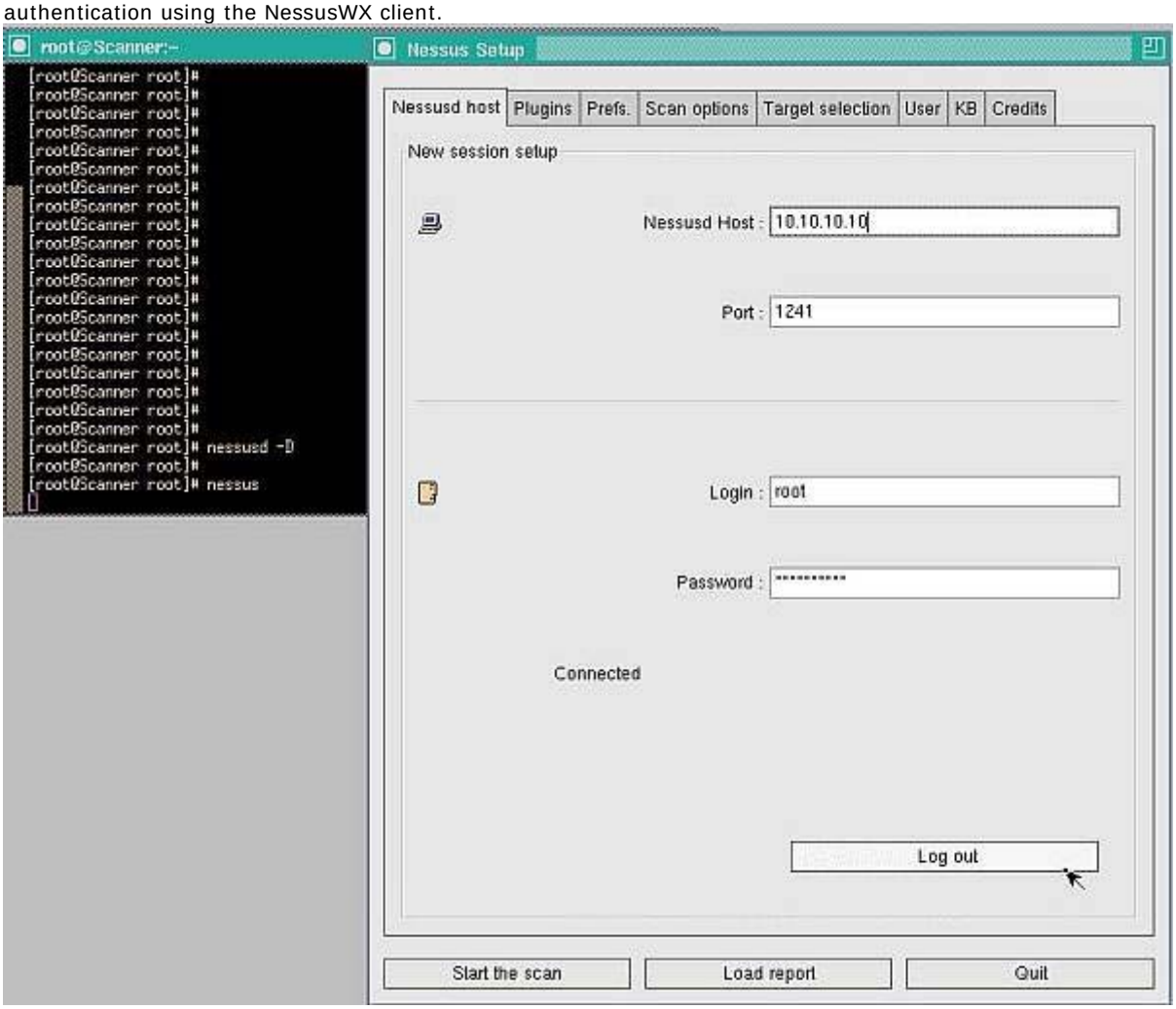

Figure 1: Starting the Nessus server and connecting with the Unix GUI

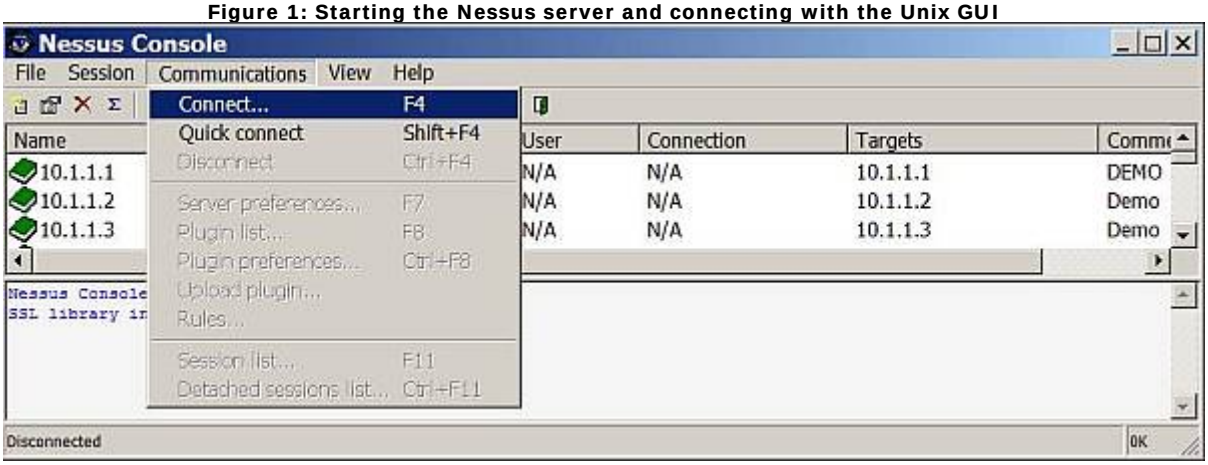

Figure 2: Connecting to the Nessus server with NessusWX (Windows Client)

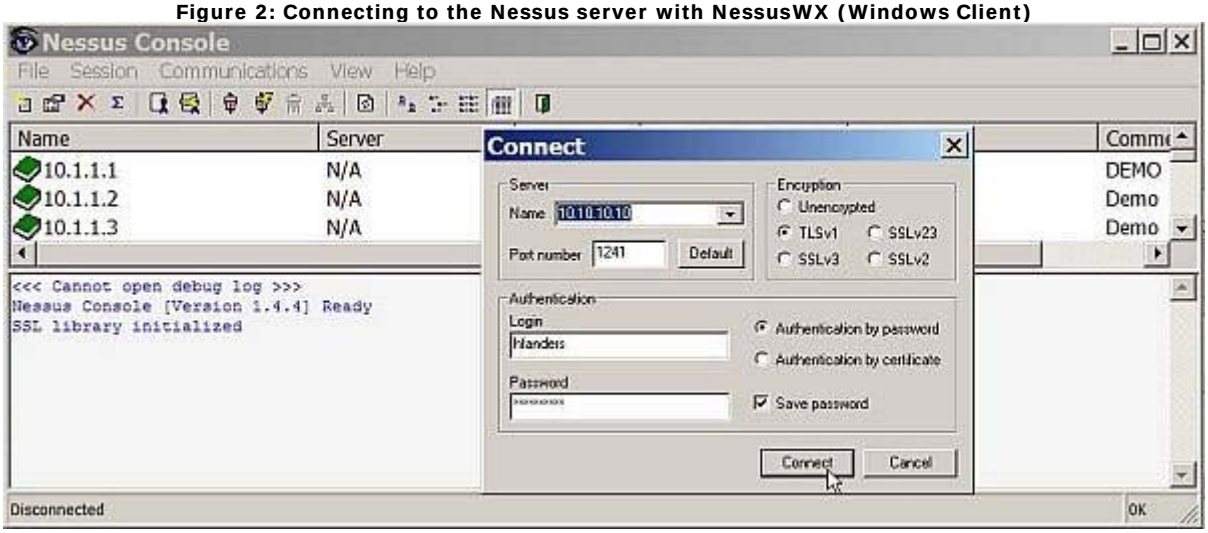

Figure 3: Enter in the server IP and the login and password setup with nessus_adduser

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:

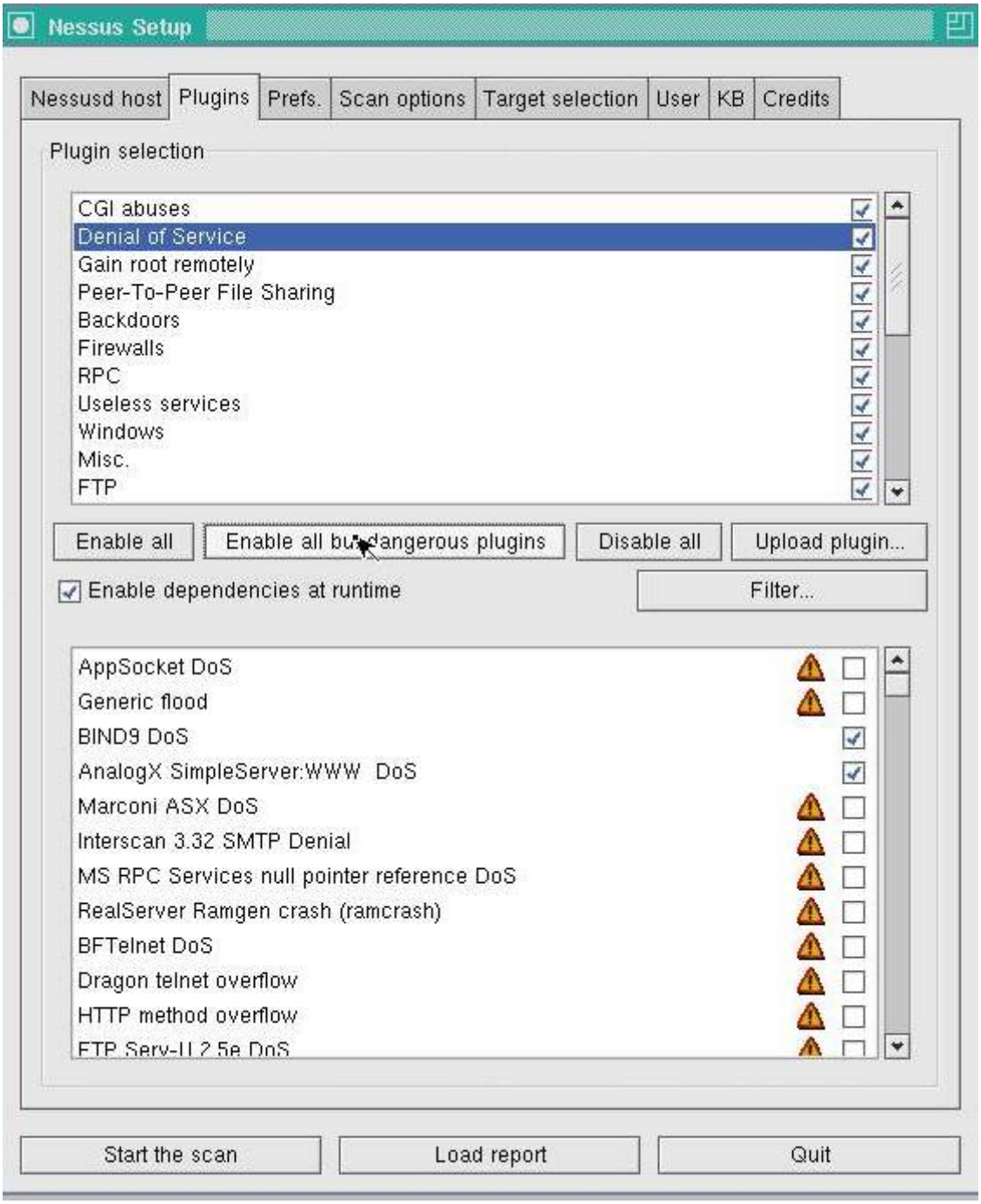

Figure 4: Enabling all but dangerous plugins with the Unix Nessus GUI

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

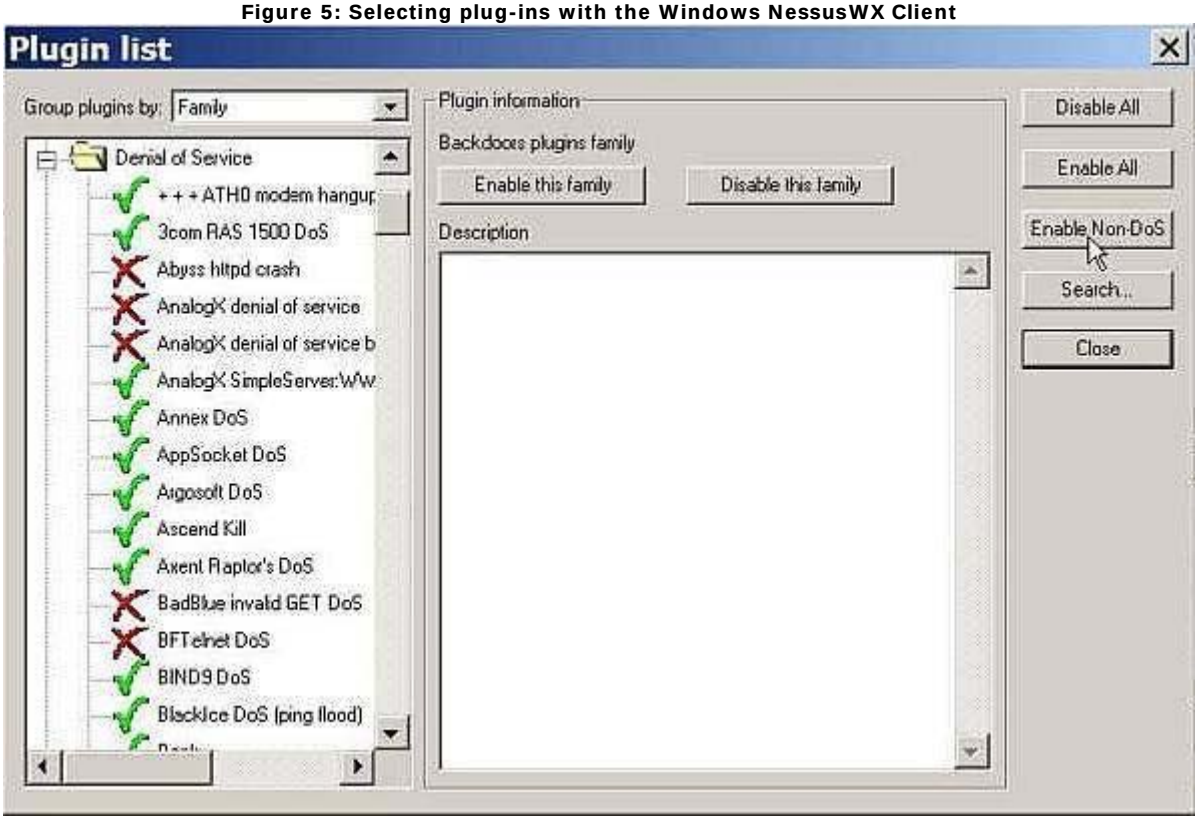

Figure 6: Enabling non-dangerous plug-ins with the Windows NessusWX Client

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:

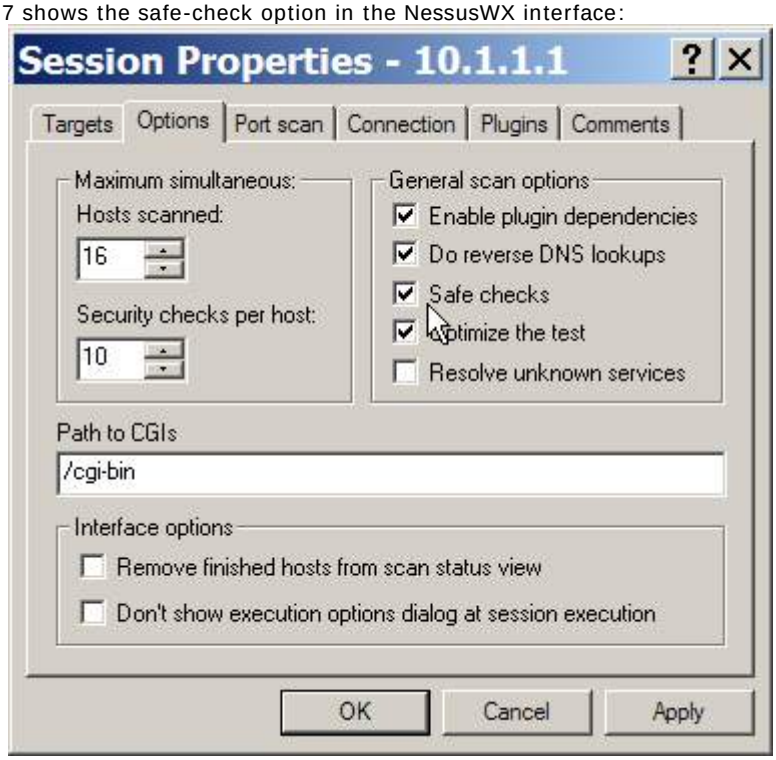

Figure 7: Choosing Safe Checks

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:

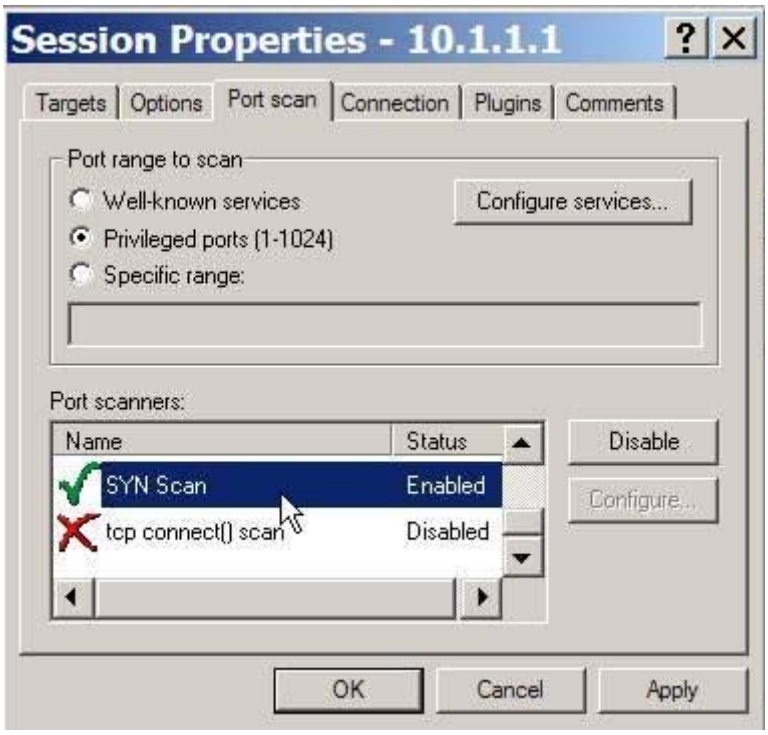

Figure 8: Configuring the internal SYN scan for a simple port scan on NessusWX

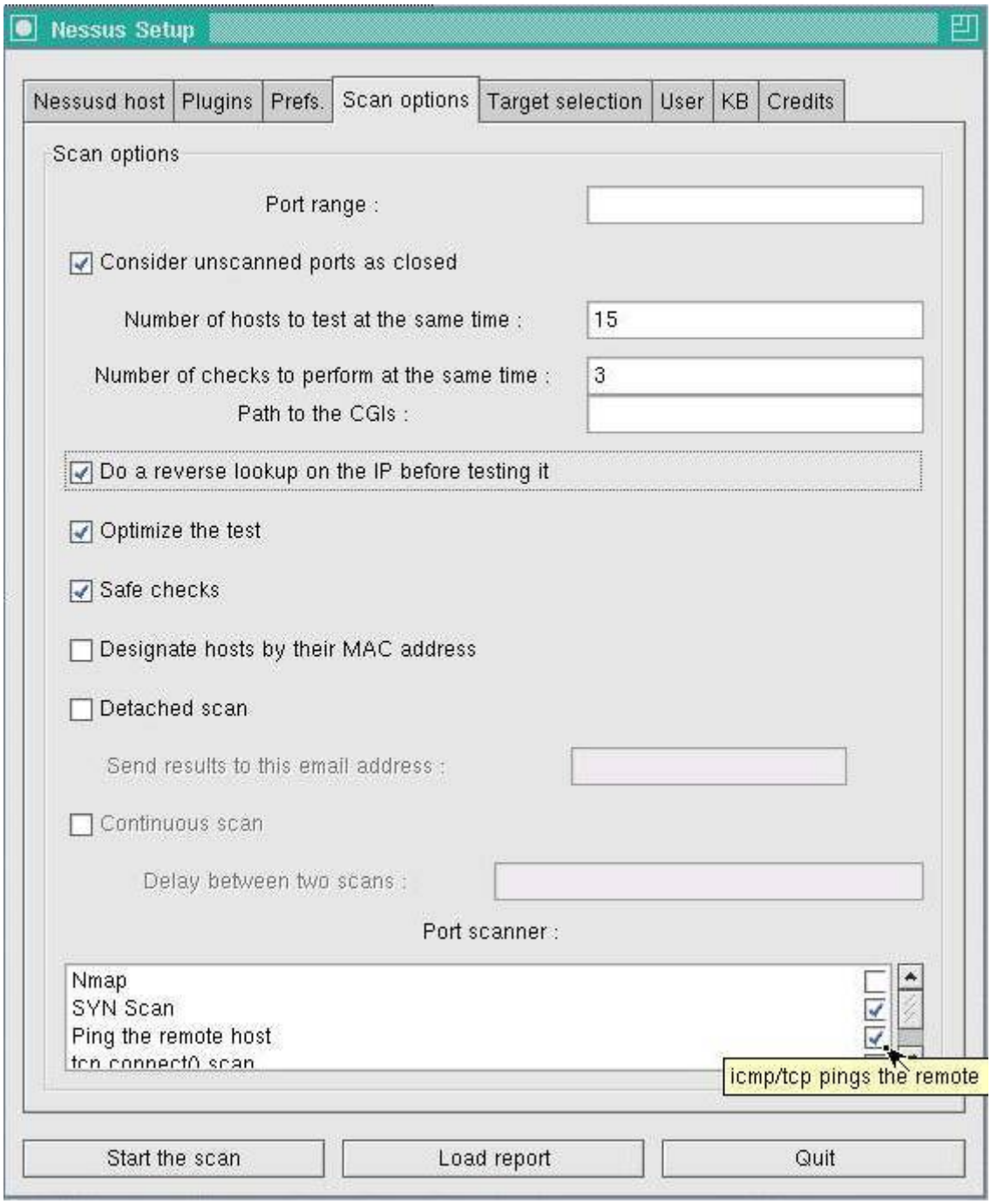

Figure 9: Configuring the internal SYN scan for a simple port scan on the Unix Client

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:

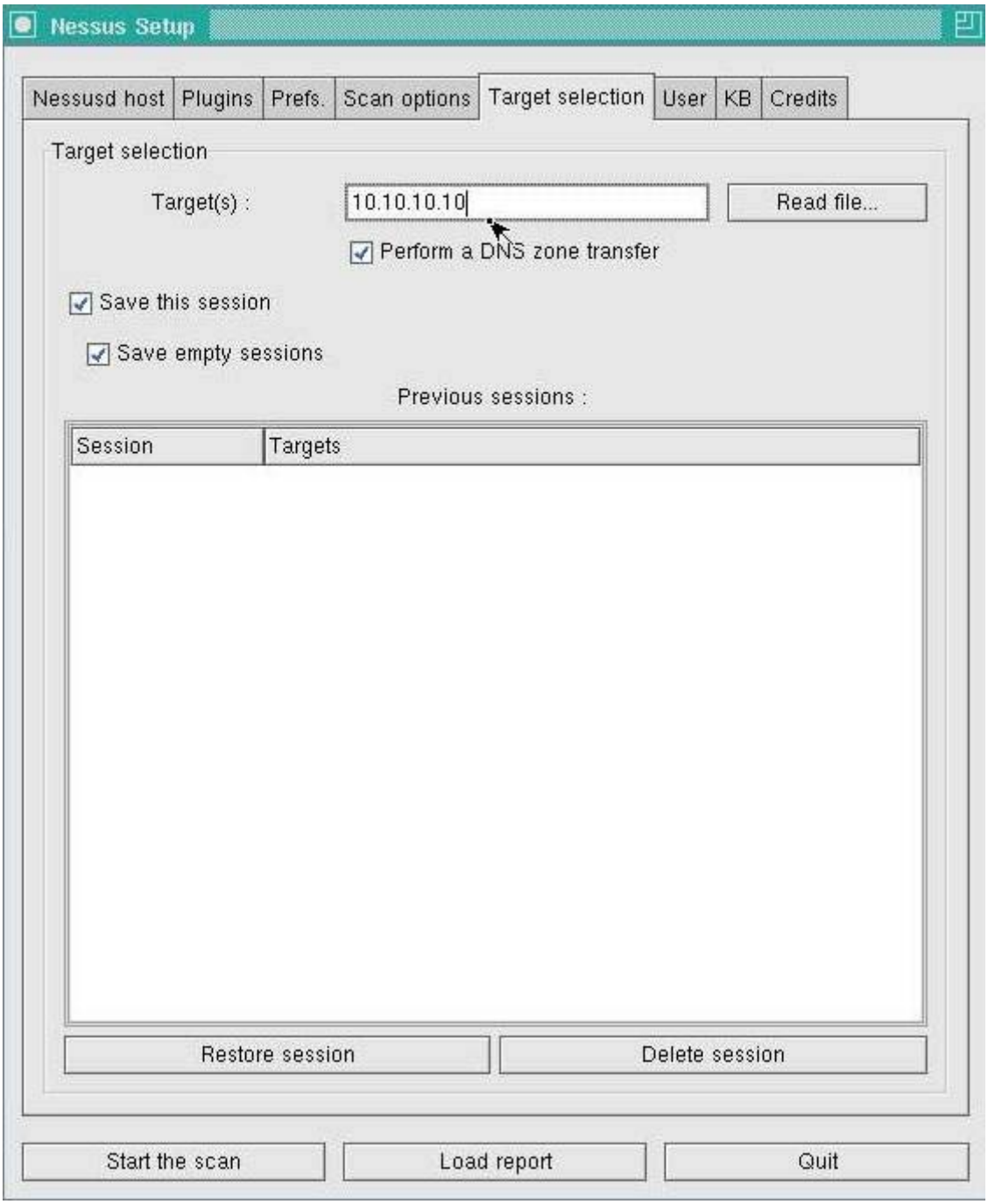

Figure 10: Specifying Targets in the Unix GUI

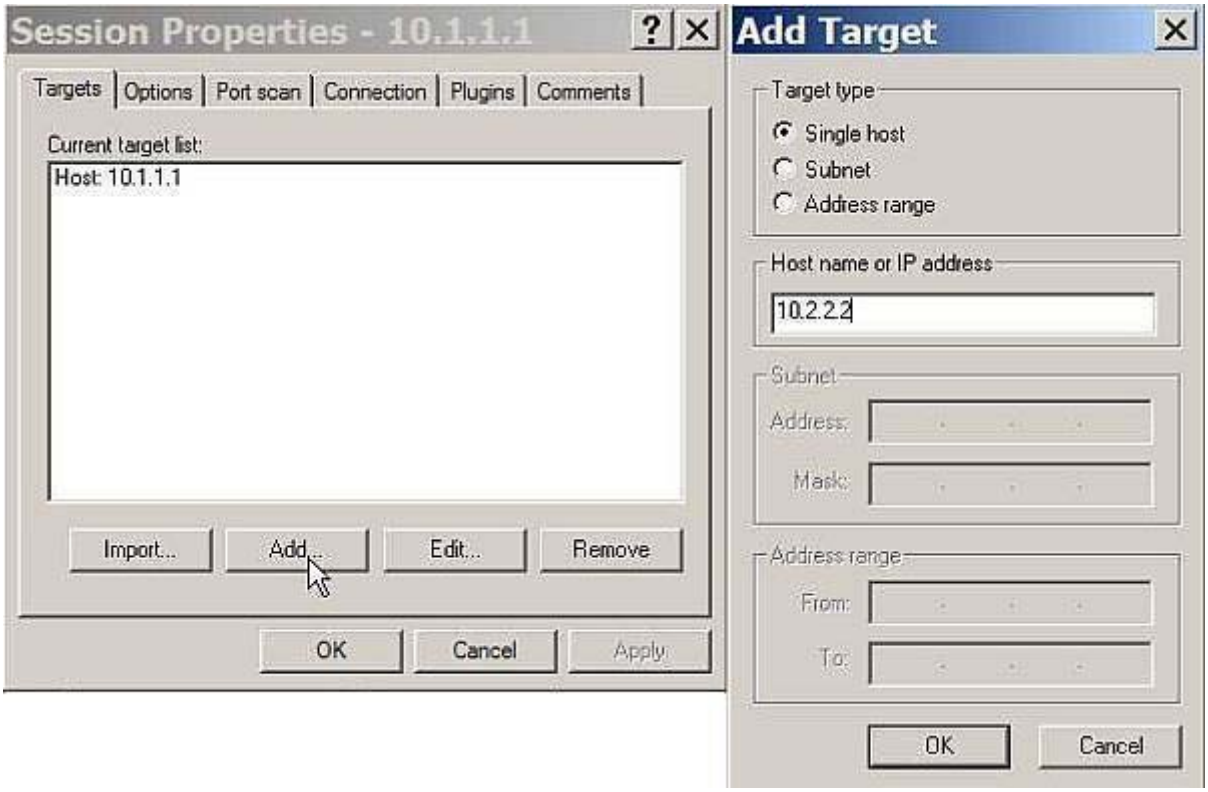

Figure 11: Target Selection in NessusWX

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.

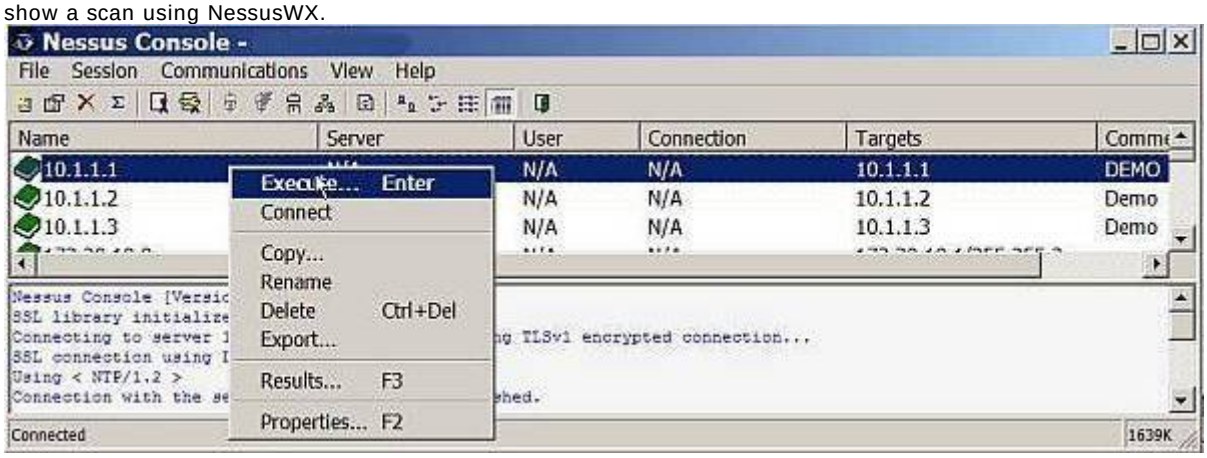

Figure 12: Starting a scan in NessusWX

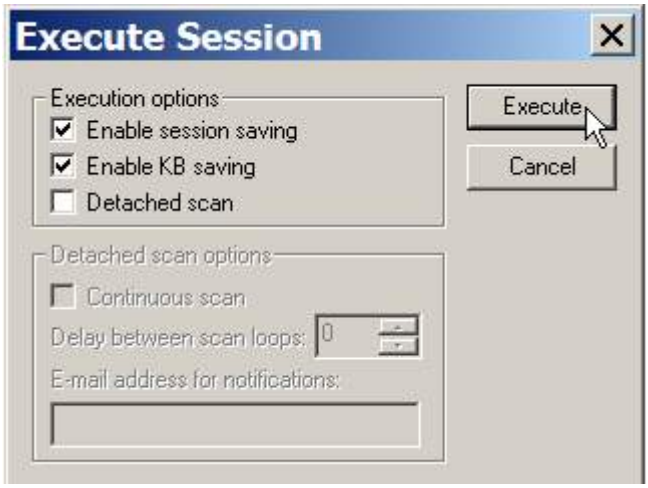

Figure 13: Starting a scan in NessusWX

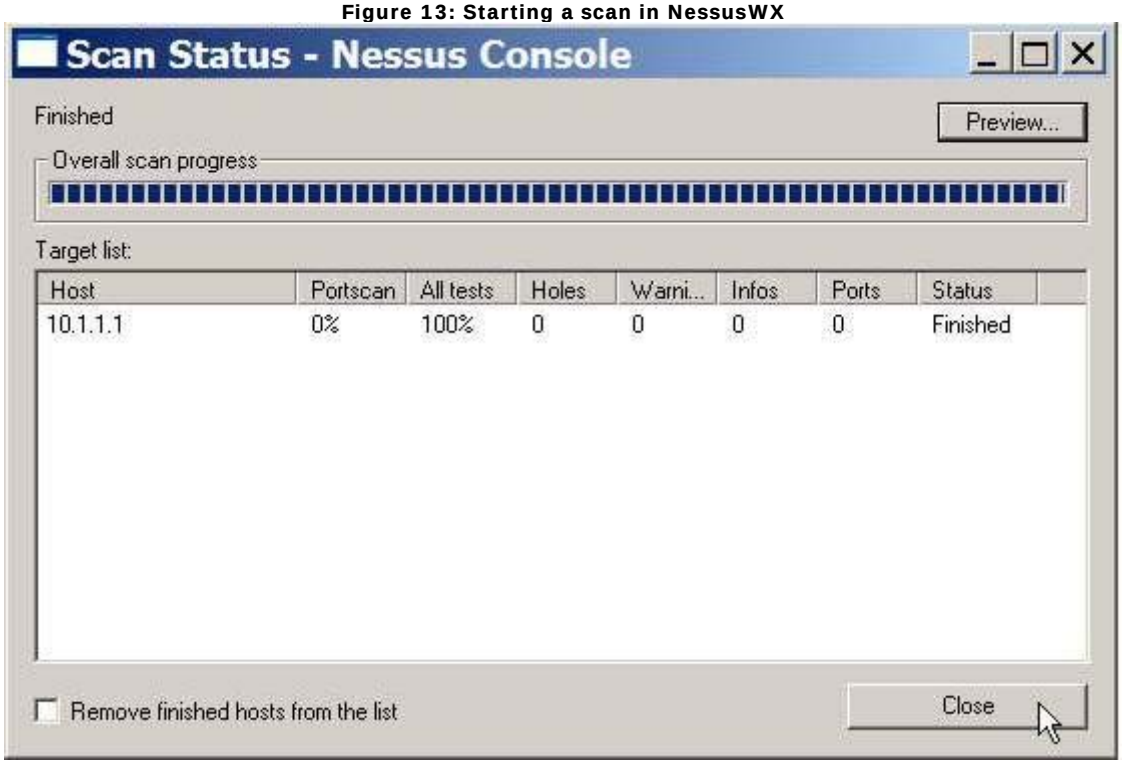

Figure 14: NessusWX scan in Progress

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!