Global Information Assurance Certification Paper

A Guide to Discovering Web Application Insecurities, Before Attackers Do

Don J Williams
GIAC Security Essentials Certification (GSEC) – 1.4b, Option 1
December 18, 2004

Abstract

It is all over the news: web based attacks are climbing, month over month, year over year. At the same time companies are attempting to combat such attacks, attackers are devising new methods to infiltrate systems. In the event you were on a reality show for the last few years and missed the latest news, just take a glance at these alarming statistics:

- "By exploiting a vulnerability in Microsoft's IIS web server product, over 250,000 web sites are thought to have been compromised by the 'Code Red' worm, in the course of a 9 hour period." (Danyliw)
- "When asked what types of losses their organizations experienced last year, over half of respondents (56%) report operational losses, 25% state financial loss and 12% declare other types of losses." (CERT)
- "In 1998, 50% of those surveyed reported no attack-related downtime whereas this year (2004), only 6% make such a claim." (Hume. p.54)
- "Nearly half of the fastest-growing U.S. companies have suffered security breaches, but most still aren't prepared to dedicate enough resources to address the problem, according to a study by PricewaterhouseCoopers." (PWC)

Web based attacks require attention today, and the consequences can be devastating for businesses who fail to take information security seriously. To protect against an attacker, you have to think defense. Truly defensive postures can always beat out offense.

For purposes of discussion, this paper will focus on discovery techniques companies can employ to uncover insecurities in web-based applications and system infrastructure. The following will provide valuable information on tools to determine if vulnerabilities are present and techniques application owners can deploy to mitigate potential attacks. While many of the tools showcased allow for multiple hosts to be assessed, this paper will demonstrate techniques based on a single host (application).

Disclaimer

At this point, it is crucial to inform the readers of the importance and consequences of obtaining “written permission”, otherwise known as get-out-of-jail card, by an authorized application owner *before* the teachings outlined below are performed on a site. Failure to do, so even in your own workplace, can lead serious legal consequences including potential loss of employment, financial penalties and/or jail time. Case in point, consider Randal Schwartz: While employed at Intel as a consultant, he used a password crack tool to demonstrate the company had weak passwords. Unfortunately, he did not get permission first, which led to three felony counts under Oregon’s Computer Crime Law, 5 years probation, 90 days jail time (deferred), $68,000 in fines and over $170,000 in court fees! (Lightlink)

Ed Skoudis, SANS instructor and author, has created a sample template permission document for your use. For a copy visit http://www.counterhack.net/permission_memo.html.

1.0 Profiling

During this initial phase, attackers probe systems to discover and collect information for later phases. Since this paper details how web application owners can protect systems and discover vulnerabilities before attackers do, I have included this phase as a window into the techniques attackers use. There are a plethora of methods which can be used to discover information about applications and the majority of them are free and publicly available.

This phase can be assimilated to military strikes on the enemy. Before forces perform missions, they need to plan the attack to be victorious. This same school of thought is used by successful web attacks. The following will demonstrate techniques which can be executed to harvest application information as a prerequisite to an attack.

1.1 WHOIS

WHOIS [webopedia.com/TERM/w/whois.html] is an Internet utility which returns information about a domain name and/or Internet Protocol (IP) address. The various WHOIS databases collect information about companies who register domain names (such as company.com or university.edu). The information is stored in the form of a record and resides in various publicly accessible databases. Information which can be discovered includes contact names, physical addresses, Internet addresses, phone numbers, and authoritative domain name servers (DNS). All of the above information can be used by attackers to infiltrate systems. For instance, contact names, phones and addresses can be used in social engineering attacks; IP addresses can be scanned; and technical email addresses can be used in spoofed requests for information. Application owners should review company WHOIS records to determine available information and validity of the data.

WHOIS data can be researched on the Internet via a number of websites. Additionally, many flavors of *NIX systems have the command line *whois* functionality built-in. If you are not sure if your system has the ability built-in, simply type whois and enter or perform a *find* on your *NIX system (**`find / -name whois –print`**).

Step 1:

A query should be performed at Internet Network Information Center [http://www.internic.net/whois.html] to determine high level data about the domain, such as the registrar of the domain. A query of your company's domain (such as Example.com) should be performed.

From the record returned, it is discovered that the registrar is "EBRANDSECURE, LLC".

```
Whois Server Version 1.3

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to
http://www.internic.net
for detailed information.

   Domain Name: EXAMPLE.COM
   Registrar: EBRANDSECURE, LLC
   Whois Server: whois.ebrandsecure.com
   Referral URL: http://www.ebrandsecure.com
   Name Server: DNS1.EXAMPLE.COM
   Name Server: DNS2.EXAMPLE.COM
   Status: ACTIVE
   Updated Date: 25-oct-2004
   Creation Date: 19-dec-1994
   Expiration Date: 18-dec-2005


>>> Last update of whois database: Fri, 12 Nov 2004 07:13:37 EST <<<
```

Step 2:

Now that we know the registrar and the WHOIS server to query, the server can be queried for more detailed company information. This can be performed on the Internet at number of locations, such as http://www.allwhois.com/, or, as mentioned earlier, using *NIX system one can issue the following command:

```
whois example.com@whois.ebrandsecure.com
```

The following WHOIS record excerpt (source: www.allwhois.com) details typical information which can be obtained:

```
EXAMPLE.COM
Language: ENG
Created: 10/19/2004 3:26:00 PM
Updated: 10/19/2004 3:26:00 PM
Expires: 12/18/2005 12:00:00 AM

Example Corporation (ASKXXXX)
Example Corporation
100 West Big Bear Rd
Sterling Heights, MI 48084
UNITED STATES
Phone: +1.248463xxxx Fax:
E-Mail: domainnames@example.com
Updated: 10/21/2004 9:01:00 AM
Created: 9/22/2004 1:09:00 PM

Administrative Contact:
Mary  Jane (ASKL4OHW7KHB0XXXX)
```

```
Example Corporation
100 West Big Bear Rd
Sterling Heights, MI 48084
UNITED STATES
Phone: +1.248463xxxx Fax:
E-Mail: ops2@ebrandsecure.com
Updated: 10/21/2004 9:01:00 AM
Created: 9/22/2004 1:09:00 PM

Billing Contact:
Billing  Department (8AMLEXXXX)
eBrandSecure LLC
550 Wilshure Blvd, Rm222
Los Angeles, CA 90010
UNITED STATES
Phone: +1.213387xxxx Fax: +1.213387xxxx
E-Mail: billing@ebrandsecure.com
Updated: 11/8/2004 6:16:00 AM
Created: 9/15/2004 4:12:00 PM

Technical Contact:
John  Doe (R185UXXXX)
Example Corporation
100 West Big Bear Rd
Sterling Heights, MI 48084
UNITED STATES
Phone: +1.248463xxxx Fax:
E-Mail: jdoe@example.com
Updated: 10/21/2004 9:01:00 AM
Created: 9/22/2004 1:09:00 PM

DNS Servers:
DNS1.EXAMPLE.COM (208.xx.xxx.1)
DNS2.EXAMPLE.COM (208.xx.xxx.2)
```


Step 3

Network discovery, that is, determining which IP ranges belong to a company, can be queried using American Registry for Internet Numbers [www.arin.net]. Data discovered in this step can be used in the upcoming scanning phase, or to determine if a system is owned by the target company or another entity, for instance an Internet Service Provider (ISP). Note: the data discovered here can be critical to validate written authorization has been provided by the owning company. To query, enter the company name (ie. Example) and hit enter. Sample edited result set is listed below.

Search results for: example
Example Corporation (EXAMPLEC)
EXAMPLE CORPORATION (EXAMPLE-1)
Example Fashions (EXAMPLEF)
Example International Headquarters (KIH)
Example Corporation (AS1350XX) EXAMPLECORPORATION 13507
Example Corporation EXAMPLEL (NET-12-47-XX-0-1) 148.xxx.0.0 - 148.xxx.255.255

Example Corporation EXAMPLE-CO146-65 (NET-12-47-XX-0-1) 12.XX.65.0 - 12.XX.65.255
…
ARIN WHOIS database, last updated 2004-08-20 19:10
Enter ? for additional hints on searching ARIN's WHOIS database.

The query reveals Example owns the 148.XXX.0.0/16 class B net block and the 12.XX.65.0/24 class C net block of IP addresses. Such information can be fed into port scanners, vulnerability scanners, and the like to determine potential vulnerabilities.

Profiling Defense Techniques

As you have just learned, there is an abundance of publicly available information which attackers can use in preparation to launch an assault. One may be wondering what can be done to protect a company from such a vulnerability. The answer is not an easy one, as the data is designed for public disclosure. The first step application owners should do is the look up their own domain names/IPs to review and validate the information.

Companies can implement safeguards to protect the organization. In many cases, administrative or technical contacts will leave the company and still be able to change the organization's domain information. Due to this, one approach is to use an email alias with a toll free number in place of company personnel and phone numbers. (Scambray, McClure, Kurtz, p.24-25) Adopting such an approach offers defense in two areas: first, real names and emails are obfuscated by the use of aliases, which can protect the company from social engineering; second, the use of toll free phone numbers thwart war-dial attacks.

It is important to state that using fake contacts and phone numbers may create more damage then protection, as in the event your application is compromised and used in an attack, others may not be able to advise you!

1.2 Application Detection

Advanced attackers, as opposed to 'script kiddies' [www.webopedia.com/TERM/S/script_kiddie.html], attempt to discover key pieces of information about applications in advance of an attack. Application detection, also known as footprinting, provides insight into the architecture behind web applications and can be used as a stepping-stone in an attacker's arsenal. Detection of Operating Systems (OS), web servers, application servers, mail servers and associated versions/service packs are of key interest to attackers. This section will demonstrate OS and web server detection. (For the other areas, a simple search on the Internet will reveal all that is needed.)

Have you ever reviewed your web server logs only to discover IIS attack

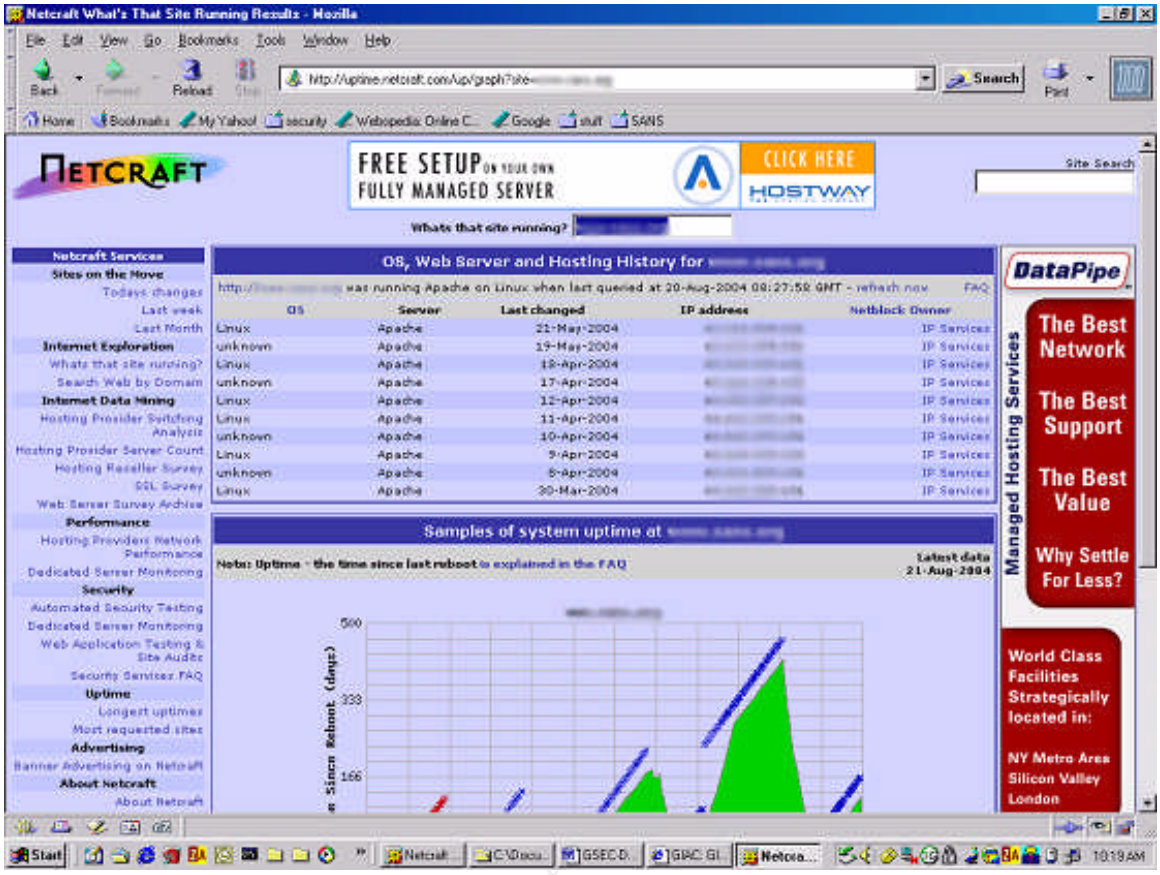
signatures (ie. **`GET /Null.printer HTTP/1.0`**) against your Apache web server? This is usually the sign of script kiddies who have not done their homework. Performing application detection allows attackers the ability to research and launch specific (and sometimes version/patch level specific) attacks, increasing their success ratio. There are many techniques and tools to discover such data. Let's explore some of the options you can use.

1.2.1 Operating System Detection

Discovering a remote OS version can be an extremely valuable network reconnaissance step, since it is possible to search for exploits down to the particular OS version and, if available, patch level (Fyodor). Once the attacker knows the specific version of the operating system, she can search for specific attacks to compromise the system or use the data as a basis for a social engineering attack.

So how does it work? OS detection ranges from simplistic approach to highly technical testing techniques. On the simple side, forcing a system to display a banner may reveal what you are seeking. The use of telnet or forcing a HTTP error (ie. access a website and add special characters to the end of the address) may reveal the OS. For more advanced checks, tools are available which perform several techniques such as sending bogus TCP flags, sending crafted packets with various TCP flags, and many more. Basically, you just look for things that differ among operating systems and write a probe for the difference. If you combine enough of these, you can narrow down the OS very tightly (Fyodor). In a nutshell, operating systems respond differently to various tests, which is the key as patterns are discovered and built on leading to a OS specific signature.

One of the quickest methods to determine what OS the target is running is to use the Internet tool "What is the site running" at www.netcraft.com. To utilize the tool, simply enter the domain name (ie. www.example.org) and enter. As displayed below, www.example.org is most likely running Linux OS and Apache web server.



Using an alternate OS detection tool, it may be prudent to validate the Netcraft findings, as the banner may have been changed. Examples include telnet, netcat [netcat.sourceforge.net], nmap [www.insecure.org/nmap], p0f (Passive OS Fingerprinting) [www.stearns.org/p0f], unicornscan [www.unicornscan.org] and queso [www.apostols.org/projectz/queso] . (Note: You may need to Google the queso site, as this URL seems to be down.)

Let's take a look at using a few of the tools mentioned to detect the OS running on a target server. First, we will use telnet, which is built into windows and *nix systems.

```
telnet 10.10.10.100

      SunOS 5.8


      WARNING:  UNAUTHORIZED PERSONS...  DO NOT PROCEED!

      login:
```

As you can see, the server responded with a verbose banner informing us that

the server is running Sun Solaris 5.8. Many times, servers on the internet do not have port 23 (telnet) open, so we must investigate alternate methods.

One of the most popular tools for OS detection is Fydor's nmap. Nmap is a versatile tool which provides many capabilities, including port scanning, OS detection, ping sweeps and more. This tool is a must for application owners. Our discussion will focus on nmap's OS detection, which is based on TCP/IP stack fingerprinting. The tool has a large collection of OS specific signatures and, unlike some of the tools available, it can provide OS *and* the version. The following command demonstrating the syntax and results:

```
nmap -sS -O -p 80 -v www.example.org
```

The -sS option issues a TCP SYN stealth port scan, -O option uses TCP/IP fingerprinting to guess remote operating system, -p 80 option specifies port 80 to scan, -v be verbose, and the IP/domain name of the target.

```
Starting nmap 3.75 ( http://www.insecure.org/nmap ) at 2004-11-24
11:05 Pacific Standard Time
Host mavy69.example.org(64.xx.xxx.13) appears to be up ... good.
Initiating SYN Stealth Scan against mavy69.example.org(64.xx.xxx.13)
at 11:05
Adding open port 80/tcp
The SYN Stealth Scan took 0 seconds to scan 1 ports.
Warning:  OS detection will be MUCH less reliable because we did not
find at least 1 open and 1 closed TCP port
For OSScan assuming that port 80 is open and port 30717 is closed and
neither are firewalled
For OSScan assuming that port 80 is open and port 36416 is closed and
neither are firewalled
For OSScan assuming that port 80 is open and port 39000 is closed and
neither are firewalled
Interesting ports on maverick31.example.org(64.xx.xxx.131):
PORT   STATE SERVICE
80/tcp open  http
Device type: general purpose
Running (JUST GUESSING) : Linux 2.4.X (90%)
Aggressive OS guesses: Linux 2.4.20-ac2 (90%)
No exact OS matches for host (test conditions non-ideal).
Uptime 18.850 days (since Fri Nov 05 14:41:17 2004)
TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
IPID Sequence Generation: Randomized

Nmap run completed -- 1 IP address (1 host up) scanned in 16.053
seconds
```

Next, we will look at passive OS fingerprinting (aka p0f). Command syntax and results are demonstrated. P0f is interesting as it has a thorough database of TCP/IP characteristics for both SYN initiation packets and SYN/ACK response packets. Unlike active OS detection, which includes sending crafted, abnormal

packets to the remote host analyzing the replies being returned from the remote host, p0f bases it response on a normal request. Different TCP stacks will provide different responses thereby allowing the fingerprinting tool to recognize a particular OS signature (Petersen). Passive OS detection does not contact the target and thus may not trigger Intrusion Detection Systems (IDS). Let's take a look.

```
p0f -l -A
```

The -l option uses single-line output, and the -A option enables SYN/ACK mode. Once the enter key is pressed, the tool is in listen mode. Now all that you have to do is to trigger the server to respond (Ack). For instance, in the reply below, I visited www.example-2.org (216.xx.xxx.11) and www.example-1.org (64.xx.xxx.22) websites, which generated an Ack form the web servers.

```
p0f - passive os fingerprinting utility, version 2.0.5
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN+ACK) on 'eth0', 57 sigs (1 generic), rule: 'all'.
216.xx.xxx.11:80 - UNKNOWN [65535:46:1:60:M1460,N,W2,N,N,T:AT:?:?]
(up: 10070 hrs) -> 10.20.30.100:32895 (link: ethernet/modem)
216.xx.xxx.11:80 - UNKNOWN [65535:46:1:60:M1460,N,W2,N,N,T:AT:?:?]
(up: 10070 hrs) -> 10.20.30.100:32896 (link: ethernet/modem)
216.xx.xxx.11:443 - UNKNOWN [65535:46:1:60:M1460,N,W2,N,N,T:AT:?:?]
(up: 10070 hrs) -> 10.20.30.100:32897 (link: ethernet/modem)
216.xx.xxx.11:443 - UNKNOWN [65535:46:1:60:M1460,N,W2,N,N,T:AT:?:?]
(up: 10070 hrs) -> 10.20.30.100:32898 (link: ethernet/modem)

64.xx.xxx.22:80 - Linux older 2.04 (up: 5619 hrs) ->
10.20.30.100:32899 (distance 17, link: GPRS, T1, FreeS/WAN)
64.xx.xxx.22:80 - Linux older 2.04 (up: 5619 hrs) ->
10.20.30.100:32900 (distance 17, link: GPRS, T1, FreeS/WAN)nmap -vv
-O -p 80 www.example-1.org
```

As detailed above, www.example-2.org (216.xx.xxx.11) the OS is obfuscated or "UNKNOWN" and www.example-1.org (64.xx.xxx.22) reports the OS is of type Linux.

1.2.2 Web Server Detection

Like OS detection, web server detection is valuable to an attacker in the planning of an assault. Knowing the web server used by the target allows an attacker to research specific exploits for the identified web server and potentially lead to a compromise. How is this accomplished? There are numerous methods which can be utilized to detect the type of web server the target is running. As seen earlier, the use of www.netcraft.com can reveal the web server based on the banner returned by the server. It is important to note, as in the case of OS banners, web server banners can be changed and should be validated using

more than one approach. Some alternate approaches include telnet, netcat, forcing an error while browsing the application, httprint, and more. The following command demonstrates how netcat (nc) can be used to detect a web server.

```
nc -vv www.target.com 80
```

The -vv option indicates that netcat is running in very verbose mode, followed by the target, which can be a domain or IP, and the default web server port (80). Once netcat connects, one must type in an HTTP command such as:

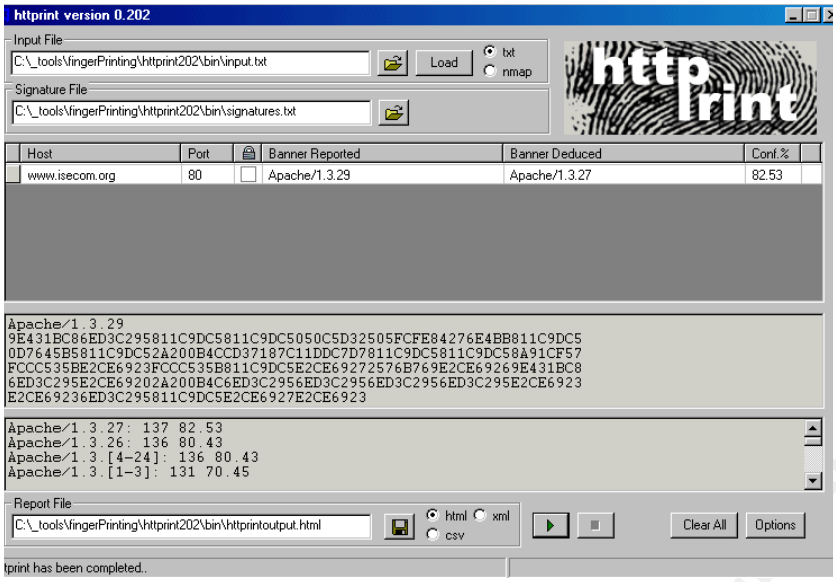
```
HEAD / HTTP/1.0
<enter>
<enter>
```

The reply should indicate what type of web server is running. An example of the output is listed below.

```
nc -vv www.example.com 80
DNS fwd/rev mismatch: www.target.com != 166-49.example.com
www.example.com [xxx.171.166.49] 80 (http) open
HEAD / HTTP/1.0

HTTP/1.1 302 Found
Date: Sun, 22 Aug 2004 18:09:21 GMT
Server: Stronghold/2.4.2 Apache/1.3.6 C2NetEU/2412 (Unix)
mod_fastcgi/2.2.12
Location: http://www.example.com/gp/home.html
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

What if the web master changed the default banner? Is there a method to determine what the real web server is? Yes! Enter the web server detection tool **httprint** [http://www.net-square.com/httprint/], which is based on web server text signature strings. Even if the banner has been obfuscated to depict an alternate type, each web server replies to requests with specific characteristics and this is what the Net-Square team implemented. Below is a sample output file using the Windows binary, checking for the web server used by www.example.org. The output reveals 82% confidence that the target is running Apache 1.3.29.



Application Detection Defense Techniques

How do you defend from application detection attempts? First, the techniques depicted above should be executed against the web application owners systems and should be saved for later comparison (baseline). Next, detected software and versions, for instance operating systems, web servers, application servers, mail server, proxy servers, and the like, should be researched to determine if they are at the latest vendor recommended version/patch level. To thwart potential attacks, all systems not at the recommended version should be upgraded. Failure to perform this initial step may lead to compromise and potentially your unemployment!

Once software products are at the vendor recommended version/patch level, one can investigate additional lines of defense. Since it has been proven that applications reveal vendor specific characteristics, as in human DNA sequencing, defense is not an easy task. According to Foundstone, operating systems can be obfuscated by changing the source code (before compilation) and/or altering system parameters to change the vendor specific OS fingerprint (McClure, Scambray, Kurtz, p.64).

Web server obfuscation, on the other hand, has more options to prevent disclosure. As for the most deployed web server, Apache (source: http://news.netcraft.com/archives/web_server_survey.html), web masters can edit the httpd.conf file and change ServerSignature directive from default setting “On” to “Off”.

```
ServerSignature Off
```

This will prevent the trailing verbose footer from displaying on server generated

responses (ie. web server generated errors), http://httpd.apache.org/docs/mod/core.html#serversignature. In addition, the ServerTokens directive can be modified from the default setting of "Full" to "Prod". http://httpd.apache.org/docs/mod/core.html#servertokens

```
ServerTokens Prod
```

By altering this parameter, information revealed from a probe (ie. netcat as described above) will be trimmed down to only display `Server: Apache` as opposed to the default setting (Full) `Server: Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2`.

Microsoft's IIS, the second most used web server as detailed by Netcraft (source: http://news.netcraft.com/archives/web_server_survey.html), has banner suppression methods which can be used to hide the identity. Specifically, one can create a custom ISAPI filter to protect the server identity, or configure the RemoveServerHeader feature in the URLScan security tool. For more information visit, http://www.microsoft.com/windows2000/community/centers/iis/iis6_faq.mspx and http://www.microsoft.com/technet/security/tools/urlscan.mspx

Finally, web masters can create custom error messages in lieu of default web server generated errors. This technique masks the server generated error and any trailing details which maybe included within the error. Most web servers allow the server generated errors, such as HTTP *404 Page Not Found*, to be configured to point to custom error page. This method suppresses the well known web server error page(s) and in addition to improving security, is much more user friendly.

2.0 Information Harvesting

2.1 Search Engines

Another area attackers may utilize to gain valuable information about your applications is via the publicly available search engines, newsgroups and job posting sites. Many times these avenues of information harvesting go undetected to companies; yet such information may be just what an attacker needs to social engineer a way into your most critical applications. How is this accomplished? Starting with the use of search engines, the following will demonstrate techniques utilizing one of the most popular search engines, Google [www.google.com]. Through the use of the advanced search directives, you can narrow a search down to a single domain.

Within the search area (form field), you can restrict the engine to your application with the use of the *site:* directive. Furthermore, one can build on the search to look for specific file types, using the *filetype:* directive. A few examples of files you can search for are: Word documents (*.doc), Adobe Portable Document Format (*.pdf), Excel files (*.xls), backup files (*.bak; *.bu, **.**old), configuration files (*.conf, *.cnf, *.ini), and more. As application owners, you are encouraged to be creative in your searches, as it is certain attackers will be. The following is a search string which will perform the search:

```
site:www.example.com filetype:pdf
```

Notice that there is no space after the directive, and there are no wildcards or periods needed. The use of these additional characters will nullify the search.

The following is a search of SANS web presence looking for Excel files:

```
site:www.example.org filetype:xls
```

As illustrated above, we have located an Excel file on the www.example.org web server. Take a look at such files as they could contain sensitive data.

Next we will build on the *site:* directive and add *intitle:* directive. This will allow one to search for a variety of resources, such as default installations of software, listing of sensitive directories, remote administration and more. To use the *intitle:* directive, combine it with the *site:* directive to stay focused on the domain. Alternatively, you can perform a global search on the Internet to see what is out there. You will be surprised!

Example search string for administration content:

```
site:www.example.com intitle:”Terminal Services Web Connection"
site:www.example.com intitle:”Remote Desktop Connection"
site:www.example.com intitle:admin intitle:login
```

Example search string for sensitive directories/files:

```
site:www.example.com intitle:”Index of etc”
site:www.example.com intitle:”This file generated by Nessus”
site:www.example.com intitle:”index of .bash_history”
site:www.example.com intitle:”index of .sh_history”
```

Example search string for default installations of software:

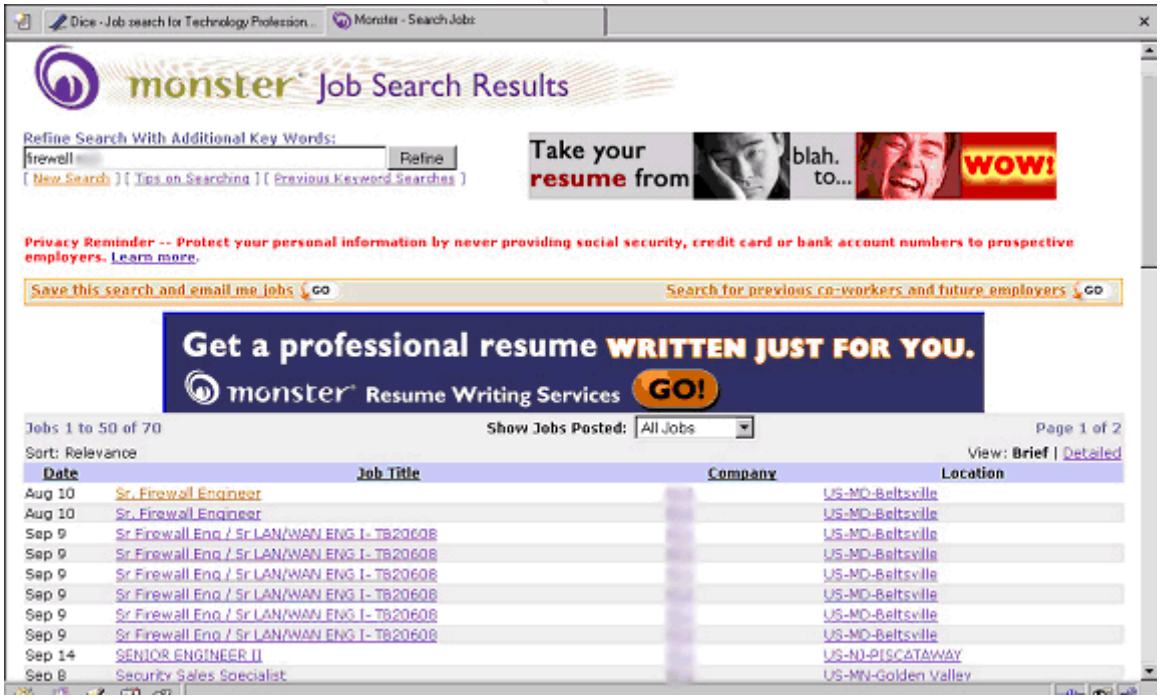
```
site:www.example.com intitle:"Terminal Services Web Connection"
site:www.example.com intitle:"Test page for Apache"
site:www.example.com intitle:"Welcome to IIS"
```

There are many more advanced search directives which can be used to locate information leakage on your applications. You are encouraged to perform these and other searches to uncover any potential risks you may have. For more information, see Google's help page [http://www.google.com/help/refinesearch.html].

2.2 Job Postings

Now that we have reviewed the use of search engine queries, let's focus on job postings. Unknown to most companies, job postings can serve as a serious risk to many organizations, as skills required many times reveal the technologies used by the requesting company. For instance, if there was a job posting on the Internet for a major financial institution, looking for a firewall administrator with specific knowledge of Netscreen firewalls, the company has most likely revealed the firewall used to protect the institution. To perform such queries for your company, you can visit several of the large job posting web sites to determine if information leakage is present. As mentioned earlier, attackers look to these career sites during their information gathering phases. The following two sites are recommended for technical positions, www.dice.com and www.monster.com. Open a browser of your choice and begin searching.

The following is a search for firewall positions at Example Company using the Monster application:



Digging deeper, select one of the job descriptions to determine if any useful information is revealed. When doing so, the qualifications section reveals the following: "…Candidates should also be proficient in operating and securing

Windows, UNIX(SUN), LINX (Red Hat), Firewalls (Checkpoint/Nokia), VPN (Cisco, NetScreen), and Cisco network devices..."

We have now learned that the operating systems used are Sun Solaris and Red Hat Linux; firewalls are Checkpoint and Nokia; and contact at the company is mary.jane@example.com.

Information Gathering Defense Techniques

Based on the techniques revealed above, there are safeguards companies can employ to protect sensitive data. To start, as with most areas of security, documented policies and education are keys to success. For new hires, build into orientation information security awareness training on these and other security related risks. For current employees, create awareness emails and short training seminars to demonstrate the simplicity and speed at which attackers can decipher information leakage avenues. In my opinion, once people see for themselves how information is gained, the retention factor is increased.

In dealing with the search engine risks, educate development staff of the dangers of storing non-essential files on productions systems. Production systems should only contain required code and, where possible, be stripped of all comments. Having a controlled production environment will decrease the potential of attacks. In addition, have a department (ie. information security) regularly search for information leakage and report findings to management. Ideally, on *NIX systems, a cron script can be used to search (find) non-essential code and move the code to a directory outside of the web root, or simply send an email reporting the files in violation.

Finally, with job postings, Human Resources need to be informed as to what types of information are not permitted to the general public. Companies may choose to hire third party firms to post and proxy all requests for open positions, thereby obfuscating the company name. Job postings looking for specific skills can be expanded to include other related technologies so as not to reveal the technology used.

3.0 Scanning

3.1 Ports Scan

To begin this section on port scanners, it is important to understand the output from port scanning tools. What does is it mean when a port scanner lists a Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) port as open, closed, or filtered? Sure, anyone can run a tool -- script kiddies are prime examples. Yet the results of tools need to be understood by security professionals in order to be effective. Port scanning tools simply send TCP and UDP requests to target machines and interpret the response from the target in the form of output. It is important to note that the results are not always valid and firewalls or screening routers could alter the response.

Systems can communicate using 65,535 TCP and UDP channels commonly referred to as "ports". Ports are used for such things as advertising the computers existence on a network, mail services, web services, and other services (SiteRecon). Once ports are discovered on a target system, they can be thought of a doors into the system. That is, ports are associated to services such as 23/telnet, 25/SMTP (mail), and 80/HTTP (web). Attackers use port scanners to discover what services are running on a target and attempt exploits for particular services in hopes of compromising the server. For a complete list port-to-service mapping, visit www.iana.org/assignments/port-numbers.

Essentially, for TCP port scans, there are two main methods: (1) a SYN or stealth scan, or (2) a full connect scan. What does this mean? For this we need to review the TCP three way handshake. During a normal TCP connection, an initiating computer sends a SYN request to the target, the target machine replies with a SYN/ACK, and then the initiating computer replies with a an ACK. Once this is accomplished, the handshake process is complete and the machines are ready to begin communicating. This method can be detected by firewalls or Intrusion Detection Systems (IDS), and is normally replaced for the use of a stealth scan, in which the initiating computer only sends a SYN packet to the target. With TCP, if the target replies with a:

SYN/ACK - the port is considered open.

RESET/ACK - the port is considered closed.

No Response/
ICMP Type3 from a different IP - the port is considered filtered (firewalled).

There are many port scanning tools available. A partial list includes nmap [http://www.insecure.org/nmap/], superscan (GUI) and scanline [www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent

=/resources/scanning.htm]; and unicornscan [http://www.dyadsecurity.com/s_unicornscan.html].

Introducing a new comer to the port scan space, we will look at unicornscan (version 0.4.2) created by the folks at Dyad Security. Unicornscan runs on most POSIX compliant systems including Linux, OS X, FreeBSD, NetBSD, OpenBSD, and Solaris . The interface is either command line or graphical through a web based front end. One of its main strengths is UPD scanning, as the tool has several UPD payloads which talks the protocol you are scanning (ie. DNS). The following is an example unicornscan TCP port scan command (run as root):

```
unicornscan -mT -pvr 250 10.10.10.100/32:q
```

The -mT option enables TCP scanning, which defaulst to having the SYN flag enabled. The -p option enables “no patients” mode to get results flowing to the screen as they are received. The -v option enables a more verbose output. The -r lets you specify the packet-per-second rate to send out the SYN packets. Unicornscan accepts CIDR [public.pacbell.net/dedicated/cidr.html] notation for the target. The /32 is used to represent a single host. The : separates the target from the port list. In this case q, or quick (291 defined default ports), is specified. Optionally, the -E flag can be used, which displays all responses, thus including closed ports. For more information on unicornscan, refer to the Getting Started Document: http://www.dyadsecurity.com/unicornscan/getting_started.txt.

```
unicornscan -mT -pvr 250 10.10.10.100/32:q
Scanning: 10.10.10.100 -> 10.10.10.100 : q from 0.0.0.0
[00:00:00:00:00:00] at
250 pps
Added     10.10.10.100 port 1025 ttl 128
Added     10.10.10.100 port 21 ttl 128
Added     10.10.10.100 port 25 ttl 128
Added     10.10.10.100 port 139 ttl 128
Added     10.10.10.100 port 563 ttl 128
Added     10.10.10.100 port 443 ttl 128
Added     10.10.10.100 port 80 ttl 128
Packets Sent: 291 Packet Errors: 0 Bytes Sent: 18624 took 0.139925
seconds
Packets recieved: 291 Packets Dropped: 11 Interface Drops: 0
Open          ftp[   21]         From 10.10.10.100        ttl 128
Open         smtp[   25]         From 10.10.10.100        ttl 128
Open         http[   80]         From 10.10.10.100        ttl 128
Open    netbios-ssn[  139]         From 10.10.10.100        ttl 128
Open        https[  443]         From 10.10.10.100        ttl 128
Open        nntps[  563]         From 10.10.10.100        ttl 128
Open     blackjack[ 1025]         From 10.10.10.100        ttl 128
```

Keep checking for the new releases (pre-release version 0.4.5) which provides additional functionality, such as `-e postgresql`, which takes all of the output and stores it to a postgresql database, and `-w file.pcap`, which writes the response packets to a pcap formatted file to view later with tools like tcpdump or

ethereal.

As with most things in life, we need to validate findings for false positives. The same holds true for scanning for open ports and services. A second port scanning tool to be demonstrated is scanline (sl), which is a command line tool for the Windows operating system, created by the team at Foundstone. Like nmap, scanline can perform TCP/UDP scans, ICMP queries, banner grabbing, and more.

The following is an example command issued:

```
sl -T -v -z -p 10.10.10.100 -o sl-1209-04.txt
```

The -T option instructs the tool to use the built in TCP port list; the –v option enables verbose mode; the –z option randomizes ports (stealthy); the –p option instructs the tool to not ping the host (stealthy); and the –o option write results to a file.

```
ScanLine (TM) 1.01
Copyright (c) Foundstone, Inc. 2002
http://www.foundstone.com

Adding IP 10.10.10.100
Scan of 1 IP started at Thu Dec 09 14:05:46 2004

-------------------------------------------------------------------------------
10.10.10.100
Responds with ICMP unreachable: No
TCP ports: 21 25 80 119 135 139 443 563 1025 1026 1027 2301
3372 3389 5631


-------------------------------------------------------------------------------

Scan finished at Thu Dec 09 14:05:52 2004

1 IP and 178 ports scanned in 0 hours 0 mins 6.03 secs
```

Note: For demonstration purposes, the built-in or default port list was used in the tools above. For a true test, either specify specific ports or scan all ports (1-65535).

Port Scanning Defense Techniques

Now that we have seen some powerful tools, which can be used to detect open ports and services, we should discuss what application owners can do to protect themselves from attackers. First and foremost, perform scans on your own systems as, most likely, an attacker has already done so. Determine what services are open and ensure only necessary ports are open.

It is a good idea to save your output and date all files for baseline data for future scans. Without baseline data, administrators may not know what new services have appeared since the last scan. Another area of concern is that some platforms (ie. Microsoft) start services by default on installation. Such services may be unknown to the owner (ie. FTP or SMTP) yet these are just the services/ports attackers scan for! It may be advisable to partner with company IT teams and information security team(s). Together, the teams can create secure build documents for new deployments of operation systems and servers (ie. web / application / mail servers). The build documents can have recommended parameters and security safeguards which must be implemented before production deployment.

Once you have a clear picture of the services running on each server and baseline data collected, you can now move to the daunting phase of determining if the services/ports available are needed. This is the area that requires careful attention and possibly an alternate environment (ie. pre-production environment which mirrors the production architecture) to test the results of disabling a running service/port. Be careful you are not the employee that management is looking for when a critical production application is down due to disabled service(s). For more information, SANS has created a list of recommended ports to block, which can be found in the Top 20 [www.sans.org/top20].

There are some well known clear text or unencrypted services which should be replaced with secure services to achieve maximum security. For instance, telnet (TCP 23) and FTP (TCP 20 & 21) should be eliminated in favor of SSH (TCP 22). Secure Shell [www.openssh.org] or SSH is a suite of programs which can provide secure authentication and encryption of data over public networks.

> “Many users of telnet, rlogin, ftp, and other such programs might not realize that their password is transmitted across the Internet unencrypted, but it is. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other network-level attacks. Additionally, OpenSSH provides a myriad of secure tunneling capabilities, as well as a variety of authentication methods.” (OpenSSH)

Now that you have scanned for open ports and eliminated unneeded services, we are left with required ports/services. How do we protect these services? At

this point, firewalls, gateway edge devises, and access control lists can be used to secure needed services.

3.2 Automated Vulnerability Scanning

Digging deeper into potential vulnerabilities, application owners can try to keep on top of each and every security weakness and visit a deluge of websites, which provide remedy information. For instance, let's assume you manage a site which is built on an Apache web server running on a Sun Solaris operating system. What can you do to ensure security of the deployment of the infrastructure? The first step would be to visit the vendor's site and research recommended patch levels and security deployment guides. When is the last time you have done this? If you are having a hard time remembering, it is time to refresh your assumptions.

Another plan of action could be to visit sites which focus on storing security issues and query for your product. An example that stands out is www.securiteam.com. Looking at the homepage of Securiteam, the data is organized into focus areas such as in the news, UNIX, Windows, Exploits and more; yet the search functionality is the greatest benefit to protecting your Internet presence. Simply enter "Apache" in the search box, select exploits as the focus, and review the potential risks you may have. A recent search resulted in 36 vulnerabilities. From here you can investigate each one and test for the existence of the potential insecurity within your infrastructure. Other sites which provide similar results include: www.cert.org; www.securityfocus.com; neworder.box.sk; searchsecurity.techtarget.com; and isc.sans.org.

However, most of us do not have time or manpower to individually lookup each potential risk for our heterogeneous environment(s). This is where vulnerability scanners may prove valuable. There are many types ranging from free to commercial solutions. In the commercial arena, the following should be considered based on your budget: SpiDynamics WebInspect [www.spidynamics.com], WatchFire/Sanctum AppScan [www.sanctuminc.com], Kavado ScanDo [www.kavado.com], and GFI Network Security Scanner [www.gfi.com]. If your budget is limited, there are no cost offerings (aka free) which provide great value, such as Nessus [www.nessus.com] and Security Auditor's Research Assistant (SARA) [www-arc.com/sara/]. These tools provide the ability to scour your applications for known vulnerabilities, thereby saving you valuable time.

So you may be asking, what are we checking for? There are a number of risks, ranging in severity, which can be uncovered via the use of a vulnerability scanner. Some common findings may include mis-configurations, default settings, default or blank passwords, sample applications or code, information gathering (ie. verbose errors, banners, internal IP addresses/domains, etc), extraneous services (ports), and more. According to Information Security

Magazine "47% of security breaches are caused by human error" (Aug 2004, pg. 23)

To gain the real advantage of using a vulnerability scanner, we will need to execute one. As mentioned earlier, there are a number of scanners from which to choose. I have selected one of the most widely used scanners on the market today, Nessus, which came into being in 1998. The particular build used in the screenshots below is Nessus version 2.1.3 running on Redhat Fedora Core1.

To begin you will need to download the source or binary code from the website [www.nessus.org/download.html]. It is important to note that the *free* server software does not run on Windows operating systems, yet it runs on the majority of Linux or UNIX platforms. For windows only shops, Tenable [www.tenablesecurity.com] has released a commercial Windows offering, namely Nevo. Once you have decided on source or binary distribution and have downloaded the code to the server, you will need to install the code.

For the purpose of this paper, the installation process will be assumed complete. A few areas to keep in mind when installing the nessus server: (1) Always check the PGP signature(s), especially when installing a binary distribution, as the code may have been modified to include a Trojan. For this very reason, source code option is recommended, as the potential of a third party altering the binary installation is minimized. (2) When running the three phases of the source code installation, ensure that *./configure* and *make* scripts are not run as root: only the *make install* script should be run as root.

Once installed, there are few configuration steps which are required prior to running the server. You will need to run the following commands and follow the prompts. First, *nessus_mkcert* is run to create a certificate which will be used for secure communication between the server and the client. Then *nessus_adduser* is run to create one or more users of the server (*nessusd*). For more advanced configuration settings, one can edit the *nessusd.conf* file, usually located in /etc/nessus. As with the installation phase, the configuration phase is left to the reader to research and perform.

At this point, we are ready to execute the server and begin using the tool. To start the server enter the following command *as root*:

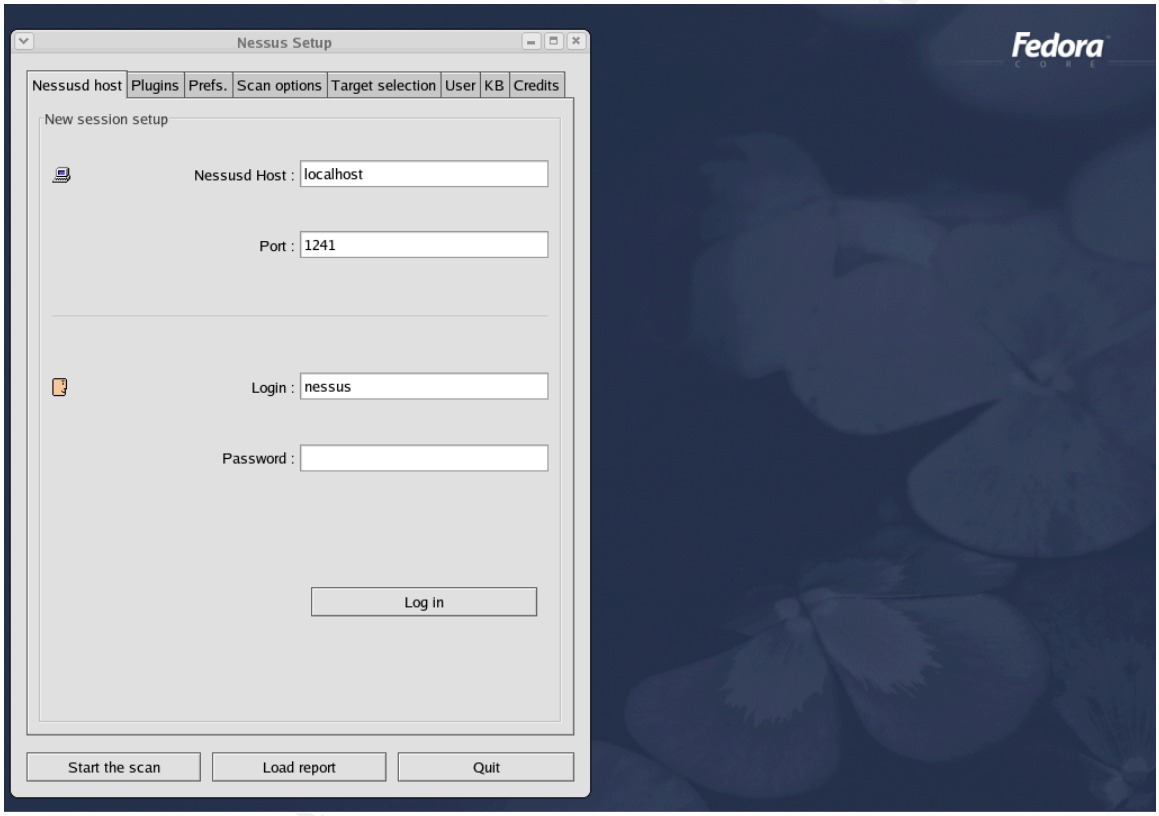
```
nessusd –D &
```

This will start the nessus server (nessusd) and run it in the background (&). By using the ampersand (&) at the end of the command, the terminal is freed up to be used for other tasks. Optionally, as a non-root user, you can enter **`su –c "nessusd –D &"`** which will prompt the user for the root password and issue a one-time command.

Now we need to start the client (front-end) or GUI to the server. To do this, enter the following command (root not mandatory):

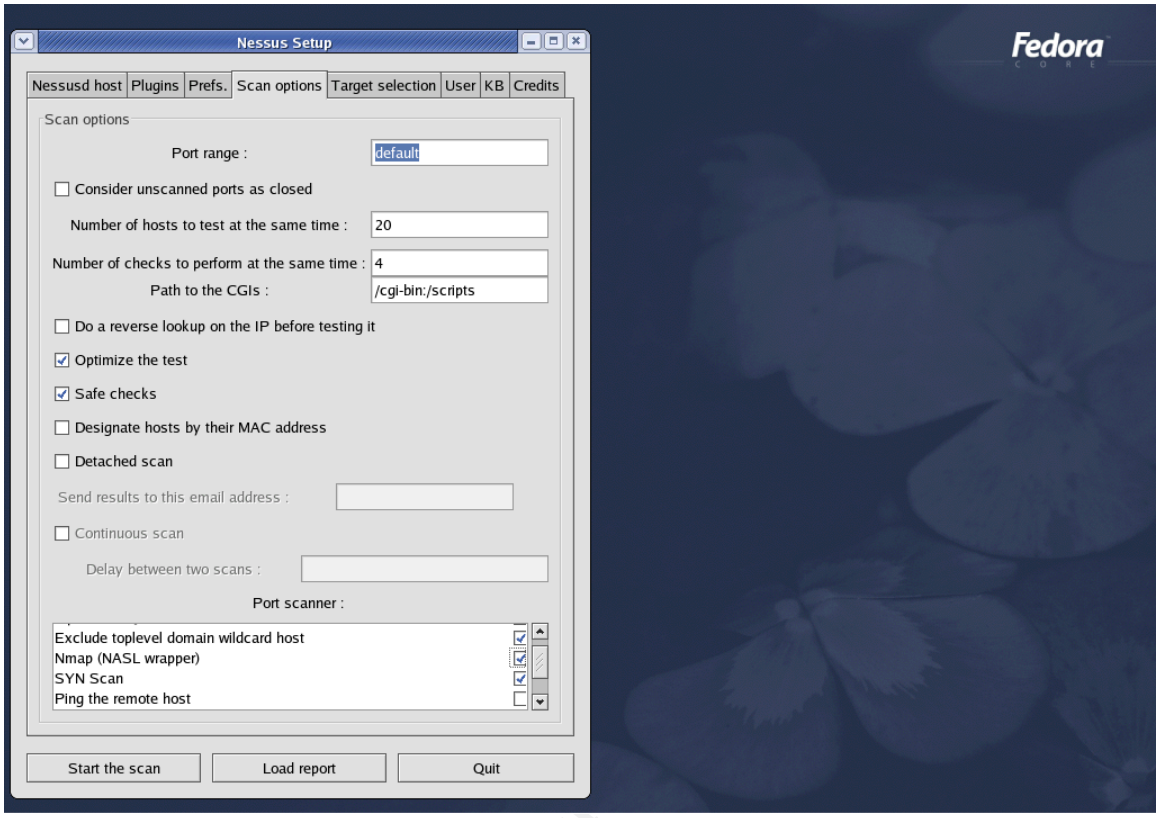
```
nessus &
```

This will start the GTK client (see below). From here, the user will have to provide a username and password to authenticate to the server. Note, there is a slick Windows client which can be used in place of the GTK client, namely nessusWX [nessuswx.nessus.org].



Once the GTK client is up, you will need to set the parameters of the scan. First, you will need to login to the application. Next, click on each of the available tabs and select the settings necessary for your requirements.

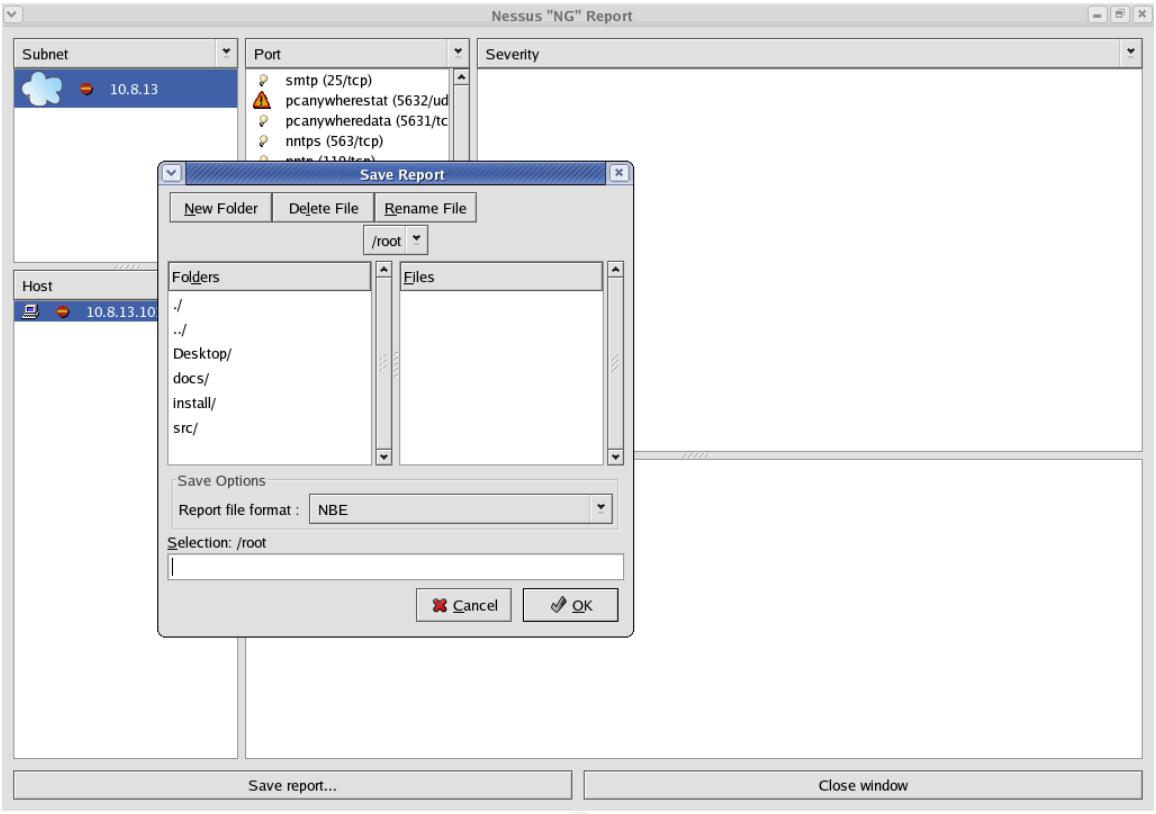
A few suggestions, on the Plugins tab, to minimize potential outages: select “enable all but dangerous plugins”; select “Enable dependencies at runtime”; and finally review the automatically selected plugins turning off unnecessary checks. There are a wide variety of options within the Prefs. Tab, so select the settings as needed. On the (Port) Scan Options tab it is recommended to select Nmap and SYN Scan. Finally, on the Target Selection tab, either enter a single IP address or an IP range, or read hosts from a file. Once you have the settings configured, you are ready to begin the scan by clicking, “Start the Scan” button at the bottom of the client.



Nessus configuration screen.

Nessus will begin issuing a variety of vulnerability checks against the target(s) entered. It maybe advantageous to run a packet sniffer, such as tcpdump/Ethereal, simultaneously to capture the traffic which can later be reviewed. By doing this, you will learn the inner mechanisms of the tool.

Once the scan completes, Nessus provides a built-in reporting engine to allow you to save the results. To create a report, simply click the Save Report button and choose the format of your desire. There are several formats to choose from, such as Nessus BackEnd (NBE), eXtensible Markup Language (XML), ASCII text, Hyper Text Markup Language (HTML), and others. Note: In order to import reports from different clients such as NessusWX into a GTK client, select NBE which formats the data as a pipe delimited text file.



Nessus report screen.

At this point, we have completed the scan and have generated a report, and it is time to analyze the Nessus Findings, to validate them and remove any potential false positives. In a sense, the report provides you with a starting point to begin accessing potential risks. Each discovered risk is categorized as a hole, warning, or open port. Each of the categories is further classified with severity levels (high, medium, low, or info). It is important to understand that Nessus is not always correct and, depending if you are using safe checks, the tool may only look at the version of the potential vulnerable service. For instance, Apache chunked encoding vulnerability, http://www.cert.org/advisories/CA-2002-17.html, if safe checks are not enabled, Nessus will test for the vulnerability by sending enough requests to the server to trigger the issue. On the other hand, if safe checks are enabled, only the Apache version is verified (Anderson).

A suggested course of action is to try to reproduce the findings to validate the potential threat. For instance, if the report states “The remote host is using a version of Apache which is older than 1.3.x.”, then use the banner grabbing techniques covered earlier to validate. It may be that the server is really running Microsoft IIS, which would be an example of a false positive. Maybe the report states directory browsing is enabled or that source code disclosure vulnerability is possible by appending a few special characters. Again you can easily validate these reported issues with the use of a browser. Sometimes, Nessus will report

findings such as buffer overflows and other unsafe vulnerabilities. Here, you must take caution before attempting to validate as attempting these types of issues may crash the server!

To assist you, Nessus includes references to informational sites which track vulnerabilities, such as Bugtraq ID (BID) and Common Vulnerability Exposure (CVE). “The BID number is a reference number generated upon submission of a vulnerability to the Bugtraq security list.” (Anderson) In the High Risk Example which follows, note the BID at the bottom of the risk is 4474. To research the finding, point your browser to www.securityfocus.com/bid/bugtraqid and enter the bid number (4474). The query returns useful data to help in your research, such as vulnerable products and versions, description of the problem, potential exploit code and solutions. It may be that your version is not listed, which may be all that is needed to rule out the finding.

Similar to the Bugtraq ID, the CVE reference number links the issue to the Common Vulnerability dictionary run by Mitre, www.cve.mitre.org (Anderson). The goal of the CVE database is to standardize names for vulnerabilities and to make it easier to share such data amongst the various databases available (Mitre.org). You are encouraged to use these links to your advantage to validate Nessus and other reported risks. In the High Risk Example below, Nessus even had a recommended solution with detailed steps within the body of the risk.

Nessus High Risk Example (Note: NessusWX report saved as HTML output)

80/tcp)	**High**	The IIS server appears to have the .HTR ISAPI filter mapped. At least one remote vulnerability has been discovered for the .HTR filter. This is detailed in Microsoft Advisory MS02-018, and gives remote SYSTEM level access to the web server. It is recommended that, even if you have patched this vulnerability, you unmap the .HTR extension and any other unused ISAPI extensions if they are not required for the operation of your site. Solution : To unmap the .HTR extension: 1.Open Internet Services Manager. 2.Right-click the Web server choose Properties from the context menu. 3.Master Properties 4.Select WWW Service -> Edit -> HomeDirectory -> Configuration and remove the reference to .htr from the list. In addition, you may wish to download and install URLSCAN from the Microsoft Technet Website. URLSCAN, by default, blocks all requests for .htr files. Risk factor : High CVE : CVE-2002-0071 BID : 4474 Other references : IAVA:2002-A-0002

In addition to high, medium, and low risks, Nessus includes informational data. This data should not be confused by the reader as a vulnerability; rather, the data is purely informational. A sample informational finding follows:

Nessus Informational Example

http (80/tcp)	**Info**	Port is open

Understanding how the tool functions is critical to your success. As recommended earlier, using a packet sniffer to capture the packets during a scan can be very educational. To further increase your knowledge, you are advised to review the scripts Nessus executes. When Nessus runs, it executes several individual scripts known as Nessus Attack Scripting Language (NASL) scripts which are created by the Nessus open community. It is these scripts which are the brains behind the tool. By default, NASL scripts are stored in /usr/local/lib/nessus/plugins directory and can be analyzed using your favorite editor. To validate the reported risks, you are encouraged to review the NASL scripts to learn how the risk was tested. It is beyond the scope of this paper to dig deeper into NASL scripts. For more information you may try the following links: http://www.nessus.org/doc/nasl.html and http://www.nessus.org/doc/nasl_doc/.

Vulnerability Scanning Defense Techniques

The first and foremost point is to understand the tool(s) and impact of the tool you select. You are advised to start by scanning test systems and/or non-critical systems initially to learn potential pitfalls of the selected options. As for Nessus specifically, “you might be tempted to enable a bunch of plugins and run Nessus against your entire network just to see what happens. This is not a good idea.” (Deraison, et al, p. 119) In my line of work, I have heard of such people and the majority of them have been terminated. Also, do not store Nessus reports in web accessible directories as attackers may find the reports providing them with sensitive information.

ALWAYS GET WRITTEN PERMISSION before assessing company developed applications and/or externally developed systems. Those who skip this step face the real possibility of litigation, termination and loss of certifications. Create a form which requestors complete and return to you signed via snail mail. Keep all such approval forms on file for later reference.

If you are scanning a production application, make sure you take the time to inform all parties who need to know. Most vulnerability scanners are noisy and will set off Intrusion Detection Systems, possibly creating unneeded havoc. An ideal approach for internal applications is to create a standing change control window for scanning applications. In addition, where possible, include the source (attacking) IP address and target IP addresses within the change entry.

4.0 Manual Review

As we have seen, there are several automated vulnerability scanning tools which may assist in discovering web application insecurities. Yet there is no substitute for manual testing with the human eye.

4.1 Detecting Vulnerabilities Using a Browser

A web browser can be an invaluable tool in locating potential application weaknesses.

4.1.1 Source Sifting

The practice, common amongst developers, of placing comments into HTML code is dangerous as the comments, in turn, get sent to the browser. Information discovered in comments can vary from developer names, phone numbers, usernames, and automated comments (ie. This page was generated

using Nessus), to logic functionality, debugging information, and 'hidden' functionality (Curphey, et al p.50). You can use a browser of your choice to view the source of most web pages. For the examples that follow, I used Firefox 1.0 available at www.mozilla.org. Note: it is assumed that the reader has some basic knowledge of HTTP protocol. For HTTP specifications and drafts visit: www.w3.org/Protocols/Specs.html.

To view HTML source code, right click the page on an area which is not an image or until you see a popup menu which has an option, such as View Page Source. Click this option and you will be presented with the HTML source of the page. Note: rarely, yet occasionally, developers disable the view source functionality, thus you may be forced to use an alternate method, such as netcat, to view source. From here you can begin by visually looking for HTML comments which are generally in the form of `<!--` (start) and `-->` (end) comment. Ideally, you can perform a find for the string `<!--` which will locate the beginning of each comment. Do not forget to view the source of linked files within the HTML such as javascript files (*.js) and CSS files (*.css).

Another test is to validate is the use of the HTML *hidden* tag, which is used to pass hidden parameters from one page to another and eventually to the server for processing. The hidden fields are not displayed in your browser and only can be seen be viewing the HTML source. Hidden parameters may be altered, which may lead to a number of issues including server errors, loss of revenue, or a system crash. Let's look at a hidden form field example. In your search, assume you found the following:

```
<input type="hidden" name="price" value="99.99">
```

You can save the page locally by clicking File > Save Page As. From here you can edit the HTML source to change the price from 99.99 to 9.99.

```
<input type="hidden" name="price" value="9.99">
```

Next search for the *form action* string and ensure the form action contains a fully qualified domain. The form action may be relative, such as below:

```
<form action="/estore/checkout.asp" method="post”>
```

If this is the case, add the fully qualified domain as detailed below:

```
<form action="http://www.example.com/estore/checkout.asp"
method="post”>
```

Then click the submit button to validate if the revised price was accepted. "This practice is still common on many sites, though to a lesser degree." (Melbourne, Jorm)

Source Sifting Defense techniques

According to OWASP Guide to Building Secure Web Applications, creating a script which filters comments prior to production release is all that is required (Curphey, et al p.50). In addition, quality control staff can be tasked to review source prior to production release, and policies should be in place to control comments in all source code.

In the case of the altered price example, applications should never use the hidden form tag to set prices. A product ID, which is cross referenced to a database for price, should be used. Thus, if the product ID is altered, either a new item and price will be displayed or an error will generate. Finally, it is critical to never rely on client side validation (ie. javascript). All input must be validated at the server.

4.1.2 Forceful Browsing

Forceful browsing is a configuration option in most web servers which allows the directory structure and associated files to be displayed to the browser. If this option is allowed, sensitive application information can be revealed, which may be used to compromise the system. To test if forceful browsing is enabled, simply delete the file name from the link and, if the directories are displayed, you have validated the risk is present. An example will help explain how this is done. Let's assume the web application link is as follows:

www.example.com/estore/products/ipod.asp

In your browser, you would delete "ipod.asp", which would leave:

www.example.com/estore/products/

If the directory structure is displayed in your browser and not an error (ie. 404 Page Not Found), then directory browsing is enabled. From here, you can visually review the directories for information, such as:

- text documents (*.txt, *.doc, *.xls, *.pdf)
- backup files (.bak, *.bu, *.old)
- compressed files (*.tar, *.jar, *.war)
- potential 'hidden' content.

This can prove to be detrimental to the security of an application. If backup files such as login.jsp.bak are discovered your browser will attempt to save the file locally. Now you can browse to the file and open it with an editor, which will contain the non-public source code of the jsp page.

Forceful Browsing Defense Techniques

Defending from such information leakage is simple: First, ensure directory browsing is not enabled on production web servers; and second ensure only necessary production code is allowed on the servers. On *NIX systems, cron scripts can be used to detect extensions listed above and either remove them and/or send the application owner an email of the findings.

4.1.3 Error Generation

Application errors can disclose sensitive information about web systems. Errors can be either server (ie. 403 Forbidden) or developer (ie. zip code must contain 5 numbers) generated. Using a browser, it is possible to force applications to cause errors. Skilled attackers are accustomed with the structure of server errors and most are able to determine what type of server/application generated a particular error. For instance, let's assume an attacker's goal is to gain access to an online financial account which is protected by a username and password. She could access the login page and enter a fake username and password. If the application returns an error such as "username invalid please try again", it may not be long before a valid username is discovered. At this point, the application may state "password invalid please try again". Now the attacker has used the application to reveal fifty percent of the authentication process, a valid username. From here she could run a password brute force tool using the valid username until success and potentially compromise the application.

Once an error is presented, take note of the contents and always view source of the errors for more information disclosure. For instance, viewing the source of a BEA Weblogic server generated error in version 5.x revealed the version and service pack of the server. Such information allows an attacker the ability to narrow down vulnerability searches, which may lead to breach of the application.

Error Generation Defense Techniques

To protect the application, always override server generated errors with custom error pages. Most servers have configuration options to replace server generated errors. Ensure custom errors are generic, especially during authentication. An example may be "authentication failed please try again".

4.1.4 Input Validation

> "Most of the common attacks on systems … can be prevented, or the threat of their occurring can be significantly reduced, by appropriate data

> validation. Data validation is one of the most important aspects of designing a secure web application." (Curphey, et al p.39)

In today's era, web sites are interactive and the majority of the web presences request input from the consumer. For instance, if you would like to purchase something online, join a newsgroup, or view financial information, all these applications require input. There are two main types of validation, client-side and server-side. Client-side validation, which is pushed to the browser and uses the client machine to process (validate) input, should only be used as a method to reduce unneeded calls to the server. That is, if an input field requires letters or a certain length of characters, client-side validation can be used to verify conditions are satisfied. Server-side validation, on the other hand, allows the data to be sent to the server for processing. Failure to ensure server-side validation can lead to a number of serious security vulnerabilities. Threats such as cross site scripting, SQL injection, and parameter manipulation are all possible if data validation checking is not done.

To verify if controls are active for your application, you can submit data which is known to be invalid. For instance, when attempting to perform a SQL injection, a database attack, special characters such as: ' (single quote); = (equals sign); and -- (double dash) are submitted to the server for processing. Using a browser, testing staff can attempt to submit such characters to determine how the application responds. Another example is cross site scripting (XSS), which is an attack where an application accepts a malicious script from a user. To perform such an breach, the site will need to accept characters such as < (less than sign), > (greater than sign), . (period) and / (forward slash). These characters can be submitted and tested. Note: It may be that client-side validation (ie. javascript) will trap such requests. Yet, as discussed earlier, anyone can save the web page locally and delete all instances of client-side validation then submit the request to the server. It is clear that the special characters described above rarely are required as valid input to applications.

Saving pages locally and determining which segment of code to delete can be a tedious effort and may require specific knowledge of scripting languages. It is important to note that the use of a proxy server can be an alternative approach. Proxy servers allow raw HTTP requests to be trapped (paused), analyzed, edited, and submitted. For instance, let's assume we are testing the quantity field of an online purchase which is displayed as a drop-down list box allowing the values of 1 through 5 only. With a proxy server, the raw HTTP data such as:

```
POST /checkout.jsp HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*
Referer: http://www.example.com/select.jsp
```

```
Accept-Language: en-us
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0
Host: www.example.com
Content-Length: 102
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: rememberID=; rememberPW=;
JSESSIONID=BKPn89L10wVYYgpSZF4TLrgrz3SsywFdGTyXbjT2GH;
Authorization: Basic dnp3dXNlcjpoYXVnbGFuZDk5

mode=buy&product=303747&desc=ipod&quantity=1&…
```

can be modified to test server-side validation. From here an attacker (or tester) can modify the quantity field to determine if negative quantities are accepted, essentially providing the attacker with a credit! To do this, the quantity value will be modified adding a negative sign as detailed below:

```
mode=buy&product=303747&desc=ipod&quantity=-1&…
```

Using this same approach, you can test the quantity boundaries (1-5) at the server.

```
mode=buy&product=303747&desc=ipod&quantity=100&…
```

As you can see, the use of a proxy server can prove very powerful. We used quantity as an example; yet the tool can be used to test any parameter captured.

There several proxy tools which can be used from free to commercial. The following are a few freely available proxy servers which can be used: Achilles [www.mavensecurity.com/Achilles], Spike [www.immunitysec.com/resources-freesoftware.shtml], and Paros [www.proofsecure.com/download.shtml]. The specific use of proxy servers is left to the reader to research.

Input Validation Defense Techniques

Proper input validation may be your best defense in web based vulnerabilities and are usually at the root of the majority of attacks. As discussed earlier, NERVER rely on client-side validation as the sole source of input validation.

To ensure maximum security, all applications should, at a minimum, perform server-side input validation. This is where the data is sent to the server and, before it is processed, it is validated to accept only known valid data, reject or strip bad data, and sanitize all input (Curphey, et al p.32). Development teams can create reusable input validation objects which must be called before processing of user supplied data. Ideally, allowing only known data is the

optimal defense. For instance, if the input field requests a name value, the validation process will only accept ASCII characters a-z and A-Z and length 20 characters. This approach leaves no room for error yet may involve more development time as opposed to the reusable code suggestion.

Summary

In summary, this composition was developed to serve as a primer for self-assessing web-based applications and showcase techniques to harden web based applications. By no means do the contents of this paper provide all the answers a web application owner needs to secure an Internet application. Information security is a dynamic field; thus, you are advised to expand your knowledge. Some further suggestions include: register for security email alerts (ie. http://www.sans.org/newsletters/), join security organizations (ie. Information System Security Association (ISSA)), attend training (SANS, Blackhat), and read security books and articles.

Reading should not be overlooked as there is a wealth of free information available. The following are recommended web application security reading Internet sources. Open Web Application Security Project (OWASP); OWASP Guide to Building Secure Web Applications http://www.owasp.org/documentation/guide/guide_about.html and http://unc.dl.sourceforge.net/sourceforge/owasp/OWASPGuideV1.1.1.pdf. SysAdmin, Audit, Network, Security (SANS); SANS' Information Security Reading Room http://www.sans.org/rr/ and The Twenty Most Critical Internet Security Vulnerabilities http://www.sans.org/top20/. Open Source Security Testing Methodology Manual **(**OSSTMM);Open Source Security Testing Methodology Manual http://www.isecom.org/osstmm/

Other areas to consider (candidates for future papers) include: use of proxy servers, development of reusable security code (libraries), Intrusion Detection Systems (IDS), Worm Detection Systems (WDS), firewalls and physical security. Do not overlook the importance of physical security because, if an attacker can gain physical access to your company and or servers, you may be doomed.

As application owners, you must strictly practice defense in-depth, by taking action. As detailed in this paper, to stay ahead of the statistics, you must regularly review your security status. If you do not, someone will do it for you and mostly it will lead to heartache, embarrassment, and possibly your termination! Case in point, consider the ISP Cloud-Nine which was in business for 6 years when it was forced to shut its doors due to Denial of Service (DoS) attacks (Geek). Take action now and become proactive as opposed to reactive!

Readers are encouraged to utilize and build on the contents of the document.

References

Danyliw, Roman; Householder, Allen. “CERT® Advisory CA-2001-23 Continued Threat of the "Code Red" Worm.” 26 July 2001. 17 Jan. 2002. <http://www.cert.org/advisories/CA-2001-23.html>

CERT, “2004 E-Crime Watch Survey Shows Significant Increase in Electronic Crimes.” 25 May 2004. <http://www.cert.org/about/ecrime.html>

Hulme, George. “Under Attack”. Information Week. 5 July 2004.

PWC, “Nearly Half of Fast Growth Companies Suffered Information Breaches Or Business Espionage Over Past 12-24 Months.” 24 November 2003. <http://www.pwc.com/extweb/ncpressrelease.nsf/DocID/031752489FF7C5C885256DE50070644C>

Lightlink, “Computer Crime?” <http://www.lightlink.com/spacenka/fors/>

Scambray, J., Stuart McClure, and George Kurtz. Hacking Exposed. 4nd Edition. Osborne/McGraw-Hill Co., 2001.

Fyodor, “Remote OS detection via TCP/IP Stack FingerPrinting” 18 Oct. 1998. 11 june 2002. <http://www.insecure.org/nmap/nmap-fingerprinting-article.html>

Petersen, Bente. “What is p0f and what does it do?” <http://www.sans.org/resources/idfaq/p0f.php>

SiteRecon, “Port Scanning.” <https://www.siterecon.com/PortScanning.aspx >

OpenSSH, “OpenSSH” 29 August 2004. <http://www.openssh.org/>

Information Security Magazine Aug 2004

Anderson, Harry. “Nessus, Part 3: Analyzing Reports.” 3 Feb. 2004 <http://www.securityfocus.com/infocus/1759>

Mitre.org, “About CVE.” 7 Sept. 2004. <http://www.cve.mitre.org/about/>

Deraison, Renaud, et al., Nessus Network Auditing. Rockland: Syngress, 2004.

Curphey, Mark, et al, “A Guide to Building Secure Web Applications” 11 Sept. 2002.
<http://www.owasp.org/documentation/guide/guide_about.html>

Melborne, Jody, Jorm, David, “Penetration Testing for Web Applications (Part One)” 16 June 2003.
<http://securityfocus.com/infocus/1704>

Geek, “ISP run out of business by DOS attacks” 10 May 2002.
<http://www.geek.com/news/geeknews/2002jan/gee20020123009914.htm>