Global Information Assurance Certification Paper

SANS Intrusion Detection In Depth

GCIA Practical Assignment

Version 3.2

Joseph Anthony B de los Santos

April 24,2002

Table of Contents

Describe the state of Intrusion Detection

Abstract

Bro (short for Big Brother) is described as a stand alone system for detecting network intruders in real time by passively monitoring a network link over which the intruders traffic passes. In this paper, what We wish to point out is that Bro is a promising alternative high-performance open-source NIDS. First we will discuss a brief overview of what basically an NIDS is, it's types, then discussing what Bro NIDS is, it's design and it's benefits, then it's structure and finally discussing on Bro's future.

With the explosion of e-commerce websites, online banking and other high-profile applications, it is understandable that organizations should want to avail themselves of the best possible protection against unauthorized entry. The threat of network intrusion hangs over any organization that possesses a network that is open to the outside world. In Essence, we open up the true dangers inherent in virtually every Internet protocol network, meaning every computer network today that are used in all sorts of industries.

Because the byword of every modern organization is connectivity, even those organizations that have no direct Internet presence remain vulnerable to network attacks and intrusions. Just because you don't have a Web Site or, equally, because your site doesn't feature any e-commerce capabilities, doesn't make you immune to the possibility of someone gaining unauthorized access to your network The fact is, that virtually every organization running anything other than a perfectly closed loop network is leaving itself open to possible intruder attack. The process of detecting such attacks is called Network Intrusion Detection. A good analogy of Network Intrusion Detection System is that of a well-trained guard dog. Now think of the rooms in your house represents your network and your fence the firewall. You need to gain access to the outside world and allow guests, visitors, gain access to your property. Naturally there will be some uninvited visitors who are not welcome in your property (such as burglars).

Because your guard dog has been trained to sniff out unwanted guests, it sounds a warning whenever it detects the presence of any unauthorized visitors coming in. Intrusion Detection systems can sit either in front of the firewall to see who's approaching or behind the firewall to warn of unauthorized entry into the network. The two types of Network Intrusion Detection systems that We wish to focus on is those that rely on audit information gathered by the hosts on the network (Host-Based IDS) and those that operate stand-alone (Network-Based IDS) by watching network traffic directly and passively using a packet filter. (There are other technologies aside from Host-based and Network-based as well). However, for the purposes of this discussion we will only focus on stand-alone Network-Based IDS sometimes called Monitors. Though monitors have limited information than systems with audit trails, the major advantage of the former is that they can be added to a network without requiring any changes to the hosts on the network. This advantage is very significant for monitoring a collection of hundreds or even thousands of hosts. Network Intrusion Detection Systems typically use two basic approaches in detecting an attack, Again, there may be others but for the purposes of this discussion we will only focus on two types of approaches: Anomaly-based and Signature-based. Anomaly-based involves the system learning what type of traffic is normal. Once a model of normal traffic behavior has been created, alerts are generated for any traffic not considered normal behavior. While these systems are theoretically able to catch any intrusion, in practice they are very difficult to implement. The distinction between normal traffic against anomalous traffic is ambiguous. A correct system would need to learn what is considered normal and not normal. It's like our guard dog

personally checking and interviewing everyone before letting anyone in. In network traffic terms, from all the headers of the IP packets it captures going towards the network, it filters out all known legal traffic. The main advantage of anomaly-based over signature-based is that an anomaly-based technique sees all the traffic running into the network, there are far fewer places to hide malicious code. However, the more popular approach is using signature-based, meaning they operate similar to a virus scanner by searching for a known signature for each specific intrusion event. Signature-based approach is based on a rule set or policy. This policy defines exactly what should be considered malicious traffic. A policy can be composed of simple string patterns which are known to appear in attacks, such as "/bin/sh" or they can be complete programming scripts written to analyze each packet and determine its intentions. The downside to a signature based or rule-based system is that it can rarely identify new attacks, but they are significantly easier to implement than anomaly-based.

Introduction to Bro

Bro is a powerful open-source signature-based network intrusion detection system developed by Vern Paxson of Lawrence Berkeley National Laboratory. Bro takes the traditional rule-based approach to intrusion detection, but adds a stateful programming interface to create policies for analyzing traffic. Bro is an automated network-based intrusion detection system that examines data as it passes through a network. Bro then analyzes this data looking for patterns of attack activity and network data can be recorded for a more comprehensive analysis. Bro is basically divided into two pieces that are independently written of each other, the event engine and policy. The engine is written in C++ and is considered the core of the system. It promiscuously monitors the traffic and prepares the data to be analyzed based on the directives of the policies. These policies provide flexibility in what kind of analysis can be done on the traffic. For example, the policy can search for known malicious strings such as /bin/sh, meaning trying to spawn a shell and so on. Actually, Bro contains a number of prewritten policies included in the distribution which can detect may known attack signatures. However, the policies should be carefully reviewed and customized to each specific network because including unnecessary policies put extra load on the monitor. An important feature of the policies is that they have their own dynamic memory allocations in which the state of a protocol can be stored. Thus enabling Bro to recognize syn floods, port scans and so on. These functions are not available in strictly pattern matching systems which make Bro highly flexible. Bro is different from other similar systems because of its flexibility and extensibility.

Bro was designed from scratch with a clear separation between the technical mechanisms that pick apart the network stream, and the policy that determines how to interpret and react to particular network events. This allows Bro to adapt quickly to the changes in the network environment because Bro searches for what is referred to as attack signatures. Since Bro is so flexible, it can be

configured to detect attack signatures that take many forms. Essentially, an attack signature can be any pattern of activity in network traffic that would likely appear as part of an attack or in accessing a compromised system. In most circumstances, Bro doesn't even need to look at a network connection to know there's something weird going on. It can tell just from summary information about a connection or series of connections. For instance: Bro is able to detect a horizontal scan, meaning when one host attempts connections to the same port on many other hosts over a very short period of time usually in seconds apart. Bro can immediately react to these since there's no legitimate reason for this kind of activity to occur. Multiple authentication failures to a particular service over a short period of time may also usually indicate an attack. Depending on what service and what systems are involved, Bro may take immediate action or simply alert security personnel.

In other circumstances, Bro needs to look deeper inside connections and examine the actual data passing through the network. For instance: There are certain commands that almost never appear in an interactive session unless the system has been or is being compromised. Since hackers rarely write their own tools, Bro can frequently identify telltale prompts or other text generated by well-known hacking tools. These are just a few examples of what Bro can detect. Unfortunately, much like virus signatures, the nature of attack signatures is rather hard to pin down since new attack and unknown signatures are found regularly. According to Mr. Craig Lant, Head of Computer Security at Berkeley, who uses Bro as their NIDS at the university, when Bro detects a possible attack it issues an alert to the security personnel. It may also begin collecting additional data for further analysis such as logs of interactive sessions. Furthermore, Bro can take independent action such as dropping the questionable connection or denying further connections from the attacking site. It's the responsibility of security personnel to follow up on these incidents by examining the data recorded by Bro as well as any other relevant data available. If it's clear that there was a successful attack, the departmental security contact for the affected computers will be contacted and advised as to what action is necessary. If it is determined that the affected computers pose a serious and immediate threat to other computers on the network, the affected computers may have their network access blocked. Once their systems have been secured or protected from further abuse, the site from which the attacks originated may be contacted along with other outside agencies like CERT.

Benefits of the Bro Network Intrusion Detection System

One major problem with Monitoring is that it usually comes hand in hand with potential privacy violations meaning confidential data may be collected and examined by the system. Since Bro is automated, no human ever sees the data examined by Bro. (Unless it’s data connected with some sort of suspicious activity).

Policies used in regards to privacy are different from one organization to the next. UC Berkeley for example, has its own privacy and confidentiality policy. (http://www.ucop.edu/ucophome/policies/ec/) Their policy requires that their campus annually report on data that are accessed without user's content and that the data never leaves the system where it's collected, which are physically and electronically secured. Policies are different because there is no right to privacy in the U.S. similar to right to free speech. In fact privacy isn't even mentioned in the constitution.

High-Load Monitoring- The ability to handle large amounts of data transfer rates and traffic without packet drops is extremely important because an intruder could defeat the NIDS by flooding the network with extraneous packets, thus avoiding detection.
Real-Time Notification- If an attack (attempted or successful) is detected quickly then timely response can be provided, such as try to trace back the attacker, minimizing damage, preventing further break-ins and initiate full recording of all of the attacker's network activity.
Decoupling mechanism from policy- Separating the data filtering, event identification and policy reactions to the events results in a cleaner software design, easier implementation, and more straightforward maintenance.
Extensibility- Because of the enormous number of different kinds of network attacks, together with the continuous discovery of new vulnerabilities waiting to be exploited, requires that Bro have the ability to rapidly add new attack scripts to its library. Consequently, Bro can be upgraded in small and easily debugged increments.
Ability to ward off attacks- Assumption that the monitor might likely be attacked, Sophisticated attackers will likely probe for weaknesses in intrusion detection systems themselves. Fortunately, Bro has to some degree the ability to ward off some of these attacks such as overload and crash.

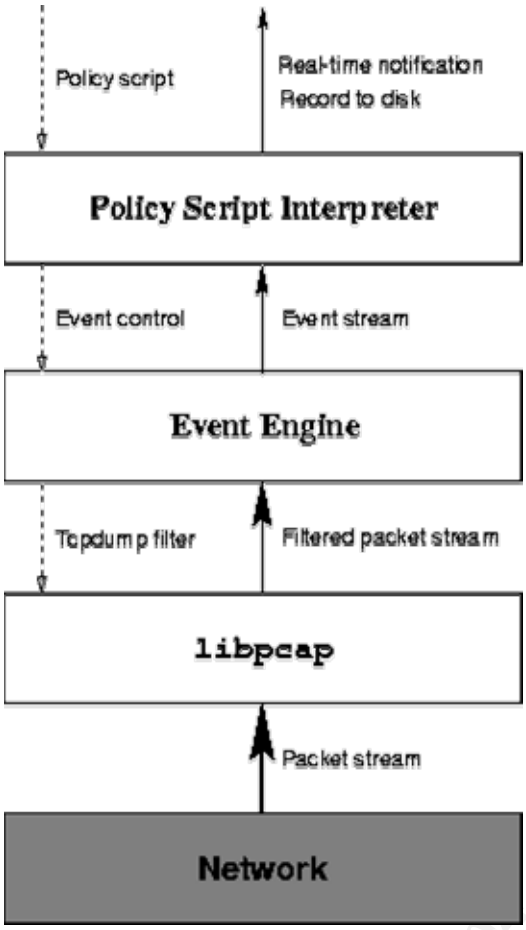
Overload- The goal of the attack is to overburden the monitor to the point where it fails to keep up with the traffic it must process. When it does fail the monitor will then start dropping packets and thus will be unable to detect attacks which normally it would detect. The result is an evasion of the Network Intrusion Detection System. To develop an overload attack against Bro, the attacker only needs to look at it's source code, especially looking at its structure (Below) to see how to increase the data flow. One way is to send packets that match the packet filter; another, packet streams that in turn generate events; and a third, events that lead to logging to disk. The first is easy since Bro's libpcap filter is fixed. The second and third is a bit more difficult to accomplish assuming that the attacker has no access to the policy scripts since the attacker does not know what events lead to logging. Also, to help defend against overload attacks, Bro's event engine generates a net_stats_update event. The value of this event gives some information on the number of packets received,dropped, etc Thus, Bro scripts at least have some basic information available to them to determine whether the monitor is becoming overloaded.

Crash- One type of crash attacks aim to knock the monitor out completely out of action by causing it to fault due to coding error. Effect can be immediate and violent but the end result is also an evasion of the NIDS. Bro provides two features to aid in defending this type of crash attacks. First, the event engine maintains a timer that expires every T seconds. (watchdog, not part of Bro but rather a Unix Alarm). Upon expiration, the watchdog handler checks to see whether the event engine has failed to finish processing the packet (and subsequent events) it was working on T seconds before. If so, watchdog timer logs this and then terminates the monitor process. This feature is coupled with another feature, the script that runs Bro also detects if it unduly exits, and if so, logs this fact and executes a copy of tcpdump that records the same traffic the monitor would have captured, meaning crash attacks are logged and it does not allow a subsequent intrusion attempt to go unrecorded.

Major Shortcoming of the Bro system

Whenever new attacks are discovered, policies must be written immediately for Bro to be able to detect it. Unfortunately, keeping up to date with this is very difficult to achieve. The best solution to this is to have well-written generalized, customized policies which can catch a multitude of suspicious behavior. For example, if there is a new buffer overflow attack for a web server, the policy might not recognize this new type of attack but it should detect an unusually large amount of binary data in the HTTP GET packet being received if the policy written was generalized enough.

Structure of Bro



At Bro's lowest level function, Bro uses libpcap, a utility to extract packets from the network. This decouples the main intrusion detection functionality of Bro from the networking details. This also allows a significant fraction of the packets entering the network to be rejected at a low level. Thus libpcap will capture all packets associated with the application protocols of which Bro is currently aware of such as portmapper, finger, ftp, telnet.
The next layer is the event engine that performs several integrity checks on the packet headers. If the header is mal-formed, an event identifying the problem is generated, and the header is discarded. The engine is written in C++ and promiscuously monitors network traffic and prepares it to be analyzed based on the directives of the policies. A check is then performed to determine if the full contents of the packet should be recorded (usually if the full packet was analyzed), if only the packet's header information should be recorded (usually if only the TCP flags were analyzed), or if nothing should be recorded (if no processing was done).

Events are generated from this process and placed on a queue (First In-First-Out) to be checked by the policy script interpreter located in the third layer. The policy script interpreter is written in a customized Bro language that uses strong typing to provide explicit support for packet header content such as port and domain and other constructs to support networking concepts. The interpreter

binds event values to the code for the event handler and then interprets the code. Executing the code may result in generating further events, logging real-time notifications, or recording data.

Essentially, when a packet arrives it begins the analysis of the engine by recombining the stream of fragments if necessary and evaluating special characters such as backspace key, delete and so on. Once the stream has been rebuilt, Bro then attempts to parse the packets based on the protocols it can recognize. Each protocol is defined with a set of functions which are used to parse the data then Bro passes control from the engine to the policies where intrusion analysis takes place. All applicable policies are passed various specific parameters and executed. The functions in the policies analyses the traffic and then creates alerts and log files of suspicious traffic.

To add new capability to Bro, one needs to identify the events associated with the protocols of the application, and write corresponding event handlers to extend the functionality of the policy script interpreter. According to Mr. Vern Paxson, this decoupling of events from their handlers improves Bro's extensibility.

Future Directions of the Bro NIDS

Bro is still in alpha stage which means that it is still under development, but according to the author, “Bro has operated continuously since April 1996 as an integral part of our site's security system. It initially included only general TCP/IP analysis; as time permitted, we added the additional modules discussed in § 6, and we plan to add many more. Presently, the implementation is about 22,000 lines of C++ and another 1,900 lines of Bro (about 1,650 lines of which are ``boilerplate'' not specific to our site). It runs under Digital Unix, FreeBSD, IRIX, SunOS, and Solaris operating systems. It generates about 40 MB of connection summaries each day, and an average of 20 real-time notifications, though this figure varies greatly. While most of the notifications are innocuous (and if we were not also developers of the system, we would suppress these), we not infrequently also detect break-in attempts. Operation of the system has resulted so far in 4,000 email messages, 85 incident reports filed with CIAC and CERT, a number of accounts deactivated by other sites, and a couple incidents involving law enforcement.

In addition to developing more application analysis modules, we see a number of avenues for future work. As discussed above, compiling Bro scripts and devising mechanisms to distribute monitoring across multiple CPUs have high priority. We are also very interested in extending BPF to better support monitoring, such as adding lookup tables and variable-length snapshots. Another interesting direction is to add some ``teeth'' to the monitoring in the form of actively terminating misbehaving connections by sending RST packets to their endpoints, or communicating with intermediary routers. This form of ``reactive firewall'' might fit

particularly well to environments like ours that need to strike a balance between security and openness."[1]

At present Bro monitors four applications: finger, ftp, portmapper, and telnet. Adding new applications to Bro is according to the author, "quite straightforward, a matter of deriving a C++ class to analyze each connection's traffic, and devising a set of events corresponding to the significant elements of the application [R31]."[2] Bro runs under several UNIX variants and is used as part of the lab's security system. As of 1998, Bro's operation had resulted in the filing of 85 CIAC and CERT/CC incident reports. Bro experiences no packet loss on a FDDI network carrying 25mbs traffic with analysis loads of peaking at about 200 packets/second.

Final Thoughts on Bro

Bro is publicly available in source-code form (see http://www-nrg.ee.lbl.gov/bro-info.html for release information). In the hopes that it will both benefit the community and in turn benefit from community efforts to enhance it.
Bro is an extremely well written, well-thought out network intrusion detection system. It has the functionality of most major NIDS, as well as a unique programmability which few others possess.

Since it is still in development, it means that it needs to have a lot more policies to catch current attacks but these policies will come with time. It is expected that they will be completed before it's official release.

References

http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028chap02.html

http://istpub.berkeley.edu:4201/bcc/Winter2001/sec.bro.html

http://istpub.berkeley.edu:4201/bcc/Nov_Dec2000/sec.whatsnext.html

http://www.itsecurity.com/papers/proseq1.htm

http://www.cc.gatech.edu/~zhangtao/mini3.pdf

http://www.usenix.org/publications/library/proceedings/sec98/paxson.html

http://www.icir.org/vern/bro-alpha-doc.html

http://www.cc.gatech.edu/people/home/scarlata/papers/scarlata.7001.ids.ps.gz.

[1] Bro: A system for detecting Network Intruders in real-time, Vern Paxson bro-usenix98-revised.ps
[2] http://www.itsecurity.com/papers/proseq1.htm

http://www.cpsr.org/program/privacy/p3p-faq.html#why

http://www.aclu.org/students/slprivacy.html

Assignment 2 Network Detects

Network Detect # 1 DNS named version request

23:11:04.884488 217.131.173.220.2839 > 226.185.188.10.domain: 4660 [b2&3=0x80] TXT CHAOS)? version.bind. (30)

=+

05/26-23:11:04.884488 217.131.173.220:2839 -> 226.185.188.10:53
UDP TTL:48 TOS:0x0 ID:41677 IpLen:20 DgmLen:58
Len: 38
12 34 00 80 00 01 00 00 00 00 00 00 07 76 65 72 .4...........ver
73 69 6F 6E 04 62 69 6E 64 00 00 10 00 03 sion.bind.....

=+

1.Source of Trace:

This trace came from http://www.incidents.org/logs/Raw/2002.4.26

2.Detect Generated By:

Looks like it was captured by snort in tcpdump binary format. If snort, the alert that could have triggered this would look something like
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS BIND Version Request"; content:"VERSION"; nocase;)

3.Probability that the source address was spoofed:

Source address is probably not spoofed or is very low since the attacker is performing reconnaissance by requesting the version of DNS being used. If the source address were spoofed then the attacker will not be able to get any useful information since the return traffic will not go back to him. However, it can still be spoofed if the return traffic can somehow be sniffed by the attacker (sort of like man-in-the-middle)

Victim----Attacker----Spoofed_host

This would only be useful during recon. The reason being to try and divert the attention of the security personnel by making them go after the spoofed host and therefore hide their real IP's (for the time being anyway, until they launch the follow-up attack) because after recon the attacker would likely launch a follow-up actual attack and at that time spoofing will no longer be available since he would need a legitimate connection in order to complete the attack like a buffer overflow (root access) for example that spawns a shellcode.

Seems to look like attacker comes from superonline in turkey.
Whois 217.131.173.220
% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html
inetnum: 217.131.0.0 - 217.131.255.255
netname: TR-SUPERONLINE-980319
descr: Provider Local Registry
country: TR
admin-c: MSO14-RIPE
admin-c: GB3469-RIPE
admin-c: OK955-RIPE
tech-c: GB3469-RIPE
tech-c: OK955-RIPE
tech-c: MSO14-RIPE
tech-c: ABK13-RIPE
status: ALLOCATED PA
notify: hostmaster@superonline.net
changed: hostmaster@ripe.net 20010122
source: RIPE
mnt-by: RIPE-NCC-HM-MNT
changed: hostmaster@ripe.net 20010528
changed: hostmaster@ripe.net 20010601
route: 217.131.128.0/17
descr: SUPERONLINE-AS
origin: AS6822
notify: muratoz@superonline.net
mnt-by: SOL-NET
changed: muratoz@superonline.net 20011220
source: RIPE
person: Murat Selahattin Oz
address: Buyukdere Cad.Yapi Kredi Plaza
address: A Blok 80620
address: Yeni Levent

address: Istanbul - Turkey
phone: +90 212 2700890
fax-no: +90 212 2702231
e-mail: muratoz@superonline.net
nic-hdl: MSO14-RIPE
changed: muratoz@superonline.net 20000301
source: RIPE
person: Gokhan Borat
address: Superonline
address: Buyukdere Cad. Yapi Kredi Plaza
address: A Blok 80620
address: Yeni Levent
address: Istanbul - Turkey
phone: +90 212 270 0890
fax-no: +90 212 270 2231
e-mail: gborat@superonline.net
nic-hdl: GB3469-RIPE
changed: muratoz@superonline.net 20010123
source: RIPE
person: Osman Kazdal
address: OK
phone: +00 000 000 0000
fax-no: +00 000 000 0000
e-mail: oska53@hotmail.com
nic-hdl: OK955-RIPE
changed: oska53@hotmail.com 20020327
source: RIPE
person: Abidin Kahraman
address: Superonline
address: Buyukdere Cad. Yapi Kredi Plaza
address: A Blok 80620
address: Yeni Levent
address: Istanbul - Turkey
phone: +90 212 270 0890
fax-no: +90 212 270 2231
e-mail: kahramana@superonline.net
e-mail: abidin.kahraman@superonline.net
nic-hdl: ABK13-RIPE
changed: kahramana@superonline.net 20001021
source: RIPE

4.Description of the attack:

DNS or Domain Name Server is a distributed database that is used by applications to map between hostnames and IP addresses and to provide email routing information. DNS can use either TCP or UDP procotol for its messages.

The most commonly used implementation of the DNS is called BIND (Berkeley Internet Name Domain server). BIND is used on the majority of name serving machines on the Internet and has a resolver library that provides the standard APIs for translation between domain names and Internet addresses. Older versions of BIND have a number of vulnerabilities in them that could easily be compromised that could allow attackers to gain root. In our logs we see attacker requesting the DNS version number from the intended victim. This is actually a reconnaissance and not an attack but this type of activity would see leading to an attack. From the logs, there was one more reconnaissance attempt to get the DNS version number by the same attacker to a different host 34 minutes after the first reconnaissance attempt. A search of "bind" at cve.mitre.org yields the following vulnerabilities that could be exploited but the most notable ones are the buffer overflows which allows execution of arbitrary code that can gain root access.
There are 24 CVE entries or candidates that match your search.
CVE version: 20020625

Name	Description
CVE-1999-0009	Inverse query buffer overflow in BIND 4.9 and BIND 8 Releases.
CVE-1999-0010	Denial of Service vulnerability in BIND 8 Releases via maliciously formatted DNS messages.
CVE-1999-0011	Denial of Service vulnerabilities in BIND 4.9 and BIND 8 Releases via CNAME record and zone transfer.
CVE-1999-0024	DNS cache poisoning via BIND, by predictable query IDs.
CVE-1999-0184	When compiled with the -DALLOW_UPDATES option, bind allows dynamic updates to the DNS server, allowing for malicious modification of DNS records.
CVE-1999-0385	The LDAP bind function in Exchange 5.5 has a buffer overflow that allows a remote attacker to conduct a denial of service or execute commands.
CVE-1999-0833	Buffer overflow in BIND 8.2 via NXT records.
CVE-1999-0835	Denial of service in BIND named via malformed SIG records.
CVE-	Denial of service in BIND by improperly closing TCP sessions via

1999-0837	so_linger.
CVE-1999-0848	Denial of service in BIND named via consuming more than "fdmax" file descriptors.
CVE-1999-0849	Denial of service in BIND named via maxdname.
CVE-1999-0851	Denial of service in BIND named via naptr.
CVE-2000-0887	named in BIND 8.2 through 8.2.2-P6 allows remote attackers to cause a denial of service by making a compressed zone transfer (ZXFR) request and performing a name service query on an authoritative record that is not cached, aka the "zxfr bug."
CVE-2000-0888	named in BIND 8.2 through 8.2.2-P6 allows remote attackers to cause a denial of service by sending an SRV record to the server, aka the "srv bug."
CVE-2000-1169	OpenSSH SSH client before 2.3.0 does not properly disable X11 or agent forwarding, which could allow a malicious SSH server to gain access to the X11 display and sniff X11 events, or gain access to the ssh-agent.
CVE-2001-0010	Buffer overflow in transaction signature (TSIG) handling code in BIND 8 allows remote attackers to gain root privileges.
CVE-2001-0011	Buffer overflow in nslookupComplain function in BIND 4 allows remote attackers to gain root privileges.
CVE-2001-0012	BIND 4 and BIND 8 allow remote attackers to access sensitive information such as environment variables.
CVE-2001-0013	Format string vulnerability in nslookupComplain function in BIND 4 allows remote attackers to gain root privileges.
CVE-2001-0497	dnskeygen in BIND 8.2.4 and earlier, and dnssec-keygen in BIND 9.1.2 and earlier, set insecure permissions for a HMAC-MD5 shared secret key file used for DNS Transactional Signatures (TSIG), which allows attackers to obtain the keys and perform dynamic DNS updates.
CVE-2002-0007	CGI.pl in Bugzilla before 2.14.1, when using LDAP, allows remote attackers to obtain an anonymous bind to the LDAP server via a request that does not include a password, which causes a null password to be

	sent to the LDAP server.
CAN-1999-1499	** CANDIDATE (under review) ** named in ISC BIND 4.9 and 8.1 allows local users to destroy files via a symlink attack on (1) named_dump.db when root kills the process with a SIGINT, or (2) named.stats when SIGIOT is used.
CAN-2002-0400	** CANDIDATE (under review) ** ISC BIND 9 before 9.2.1 allows remote attackers to cause a denial of service (shutdown) via a malformed DNS packet that triggers an error condition that is not properly handled when the rdataset parameter to the dns_message_findtype() function in message.c is not NULL.
CAN-2002-0651	** CANDIDATE (under review) ** Buffer overflow in the DNS resolver code in libc and libbind, as derived from ISC BIND, allows remote malicious DNS servers to cause a denial of service and possibly execute arbitrary code via the stub resolvers.

5. Attack Mechanism:

The attack worked by sending a UDP packet with the DNS message flag setting of query set and waiting for a response back to the query. If this type of query was allowed and the version was returned to the attacker, he can now research what vulnerabilities that version contains and exploit them. BIND has a feature where its database contains a CHAOS/TXT resource record with the name "Version.Bind". If somebody queries this record, the version of the BIND software will be returned. From our logs, we see:

23:11:04.884488 217.131.173.220.2839 > 226.185.188.10.domain: 4660 [b2&3=0x80] TXT CHAOS)? version.bind. (30)

The attacker requesting the BIND version from the intended victim, if the victim was running BIND and the query was allowed to go through, there would normally be a response like 22:34:44.573408 226.185.188.10.53 > 217.131:173.220 14049*- 1/0/0 CHAOS) TXT 9.1.1 (48) (DF)
saying there was one answer and version was 9.1.1 but in our case, there was none but this doesn't mean that there were no replies because snort has a preprocessor ignore-hosts $DNS_SERVERS that lets snort ignore traffic from these servers treating them like legitimate traffic, of course you can always write a custom snort rule to catch this. 4660 is the identification number to match response to request, b2&3=0x80 looks like a filter for bytes 2 & 3 that contains a hex value of 80 (1000 0000 binary). Bytes 2 and 3 of the DNS header contain the flags field. Since we are looking for a value of 1000 0000, only the QR field is set meaning the message is a query. Todd Beardsley asks, "Why does the attacker go to all this trouble figuring out the version number and why not just try to throw all exploits available at the target and hope one works?" The answer is it's too

noisy and it's easier to map vulnerable machines via application fingerprinting and then sending specific exploits.

6.Correlations:

A search on Google for this particular IP address yielded nothing. However, Jeff Holland reported a similar DNS Named Version Request in his practical but his was followed by the TSIG buffer overflow attempt. (because his DNS server responded, ours did not) Our logs however, only show the DNS named version request by the attacker. Also, DNS version requests are quite common.

There is also a new worm called Lion Worm (Thanks to Paul Asadoorian for this information) that tries to exploit BIND TSIG vulnerability. This worm uses randb to generate class B addresses and pscan to scan random class B internet address space and sets up to listen to port 27374 and feeds it to a webpage and sends out outgoing email to huckit@china.com with the /etc/password and /etc/shadow files. More information is at http://www.incidents.org/react/lion.html.

A Snort rule to detect lion is alert UDP $EXTERNAL any -> $INTERNAL 53 (msg:"IDS482/named-exploit-tsig-infoleak"; content: "|AB CD 09 80 00 00 00 01 00 00 00 00 00 00 01 00 01 20 20 20 20 02 61|";)

We do not appear to be infected with this worm since the packet dump does not have the content: "|AB CD 09 80 00 00 00 01 00 00 00 00 00 00 01 00 01 20 20 20 20 02 61|";) which would indicate a lion signature.

7.Evidence of active targeting:

The fact that there were only two hosts (226.185.188.10 and 226.185.227.124) that were targeted by the attacker in the log files instead of a range of address suggests that these two hosts were actively targeted as opposed to doing a general scanning of an entire network looking for open DNS ports.

8.Severity:

Criticality- Since DNS servers are critical machines criticality is set to 5.

Lethality- Reconnaissance activity is not considered lethal because this is just an information-gathering attempt. Lethality is set to 2.

System Countermeasures- Looking at the logs, it is hard to say what kind of system countermeasures are in place. System Countermeasures is set to 1.

Network Countermeasures- DNS servers are usually located outside an organization's firewall. Assuming target did not respond (However, if DNS servers were included in the snort preprocessor ignore-hosts $DNS_SERVERS,

then it will be most likely that snort would not show the replies. Network Countermeasures is set to 3.

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
(5 + 2) – (1 + 3) = 3

9. Defensive Recommendation:

Make sure that all patches are up to date. Always upgrading to the latest BIND version is always good but that is easier said than done. Configuring the BIND software to not return the version when queried to do so is a good start.

Add line: options { version "Operation Not Permitted";} to named.conf to hide the real version of bind Or you could use djbdns instead. Djbdns can be snagged at: http://cr.yp.to/djbdns.html

Also contacting Superonline and asking, reporting the offending IP address is also a good idea.

10. Multiple Choice Questions:
In this trace

23:11:04.884488 217.131.173.220.2839 > 226.185.188.10.domain: 4660 [b2&3=0x80] TXT CHAOS)? version.bind. (30)
What does the hex value “0x80” convert to in binary?
A.1111 0000
B. 1000 1000
C. 1000 0000
D. 0101 1000

ANSWER: C- First we change each hex digits into 4 bits each. Hex 8 is equal to 1000 in binary and hex 0 is equal to binary 0000.

Top Three Questions:

1. Donald Smith-“Imagine you have a dns box behind your ids. Do you log every packet it sends? Not normally. Would your IDS system (snort)
have logged a packet for a dns reply? Would it ever log a dns reply if your dns box was listed in the snort.conf file?”

Answer- if your dns box was listed in
the snort.conf file under preprocessor ignore-hosts (Essentially to reduce the number of false alerts of portscans generated by legitimate traffic from your dns servers) then my answer is no. Snort would not log any traffic from these servers

because it would think all traffic were valid not unless you create a custom snort rule to log specific dns replies.

2. Tod Beardsley- “Why does the attacker go to all this trouble figuring out versionnumbers in the first place? Wouldn't it be easier to just throw every BIND exploit under the sun at the target and hope one works?

Answer- Because its to “noisy”. It’s generally easier to use specific exploits if the machine is vulnerable rather than wasting time trying it on non-vulnerable systems. Also, a failed buffer overflow attempt could cause a segfault meaning the daemon will die until its restarted so if you blindly use a wrong exploit, you might just cause the process to quit so if a hacker wants to get into your machine, it is in his best interest that he use the right exploit first.

3.Allen Witt- “Was there any evidence of similar activity from other hosts in in the data? An aspect not covered in the SANS training that I'm seeing on the nets that I monitor is distributed recon from multiple hosts using the same methodology. The hosts can be from the same logical network or multiple unrelated networks. A few smart hackers are doing this with 'bots to reduce the chance that they'll be detected, and to avoid identification when detected.”

Answer: Yes. There was one other coming from the same attacker but directed to a different host 34 minutes after the first reconaissance attempt. You are correct that there are a lot of hackers out there that try to be as "noisy" as possible in the hope of swamping the ids with lots of alerts so that the security personnel would either be overwhelmed or would pay less attention to these and therefore be able to evade being detected when they do actually launch an attack. There was one hacking group that I read (forgot their name) that automates their scanning of target network for weeks and essentially try to be as noisy as possible via cron jobs before launching an attack. The poor security personnel became lax and didnt realize it until it was too late. However it is highly unlikely that the attacker in this log is using this same methodology since as I said there was only two different detects logged (34 minutes apart) but they can also use this methodology to be “stealthy” such as using multiple unrelated IP addresses with the same exploit and same tool with consistent timings for the relationships.

Network Detect # 2 Linuxconf Remote Buffer Overflow Scan

```
Mar 31 19:09:35 hosth snort[75541]:
spp_portscan: PORTSCAN DETECTED from 203.85.30.129
Mar 31 19:09:41 hosth snort[75541]: spp_portscan:
portscan status from 203.85.30.129: 14 connections
across 14 hosts: TCP(14), UDP(0)
Mar 31 19:09:47 hosth snort[75541]: spp_portscan:
```

End of portscan from 203.85.30.129

Mar 31 19:09:34 203.85.30.129:1542 -> A.B.C.30:98 SYN **S*****
Mar 31 19:09:34 203.85.30.129:1545 -> A.B.C.33:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1710 -> A.B.C.197:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1714 -> A.B.C.201:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1717 -> A.B.C.204:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1720 -> A.B.C.207:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1727 -> A.B.C.214:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1728 -> A.B.C.215:98 SYN **S*****
Mar 31 19:09:38 203.85.30.129:1731 -> A.B.C.218:98 SYN **S*****
Mar 31 19:09:36 203.85.30.129:1748 -> A.B.C.235:98 SYN **S*****

1. Source of trace:

This trace came from http://www.incidents.org/archives/y2k/040200.htm

2. Detect was generated by:

Probably Snort. The initial messages in the logs come from the Snort portscan preprocessor. This preprocessor logs the start and end of portscans from a single source IP to the portscan.log file. As we can see, this includes the ports scanned, how many connection attempts, the number of hosts scanned as well as the type of scan. Snort also has an alerts file (snort.alert) which contains the alert message, date and time stamp, source IP and port, Dest IP and port, protocol and IP flags. The snort portscan preprocessor logs detected portscans to the snort.alert file. The portscan.log file which is different from the alerts file contains a more detailed log of the portscans which tells which ports were hit and the IP flags used. For example,

Mar 31 19:09:35 hosth snort[75541]:
spp_portscan: PORTSCAN DETECTED from 203.85.30.129
This is a portscan logged by the snort portscan preprocessor to the snort alerts file (above). If you wanted to see more detailed explanation about this portscan, you would look at the portscan.log file which would contain a more detailed entry for this. (see below)

Mar 31 19:09:34 203.85.30.129:1542 -> A.B.C.30:98 SYN **S*****

The signature that could have fired in the alerts file should be similar to:
alert TCP $EXTERNAL_NET any -> $HOME_NET 98 (msg: " Possible Linuxconf Scan"; flags: S;)

3. Probability that the source address was spoofed:

It is highly unlikely that the source address is spoofed because the attacker is trying to gain some useful insight, there is no advantage in spoofing the source address else there is a risk of not gathering any useful information. It could *probably* still be spoofed if the return traffic can somehow be sniffed by the attacker (see description on DNS Named Version request above # 3 for more details on how this could work)

4. Description of attack:

This scan is directed at TCP port 98 which is associated with Linuxconf. Linuxconf is a sophisticated administration tool used for system configuration by various Linux distributions most notably RedHat. Linuxconf is important because it is a configuration utility, meaning it is capable of doing configuration tasks such as adding users, deleting users, etc and it is an activator, meaning it can activate/deactive services that are running on your machine. Linuxconf is basically involved at different points in the operation of your machine but if it can be exploited to gain root, then that is the most important thing to an attacker. A search of the cve website produces some interesting results.

There are 5 CVE entries or candidates that match your search.
CVE version: 20020309

Name	Description
CVE-1999-1327	Buffer overflow in linuxconf 1.11r11-rh2 on Red Hat Linux 5.1 allows local users to gain root privileges via a long LANG environmental variable.
CVE-1999-1328	linuxconf before 1.11.r11-rh3 on Red Hat Linux 5.1 allows local users to overwrite arbitrary files and gain root access via a symlink attack.
CVE-2001-0143	vpop3d program in linuxconf 1.23r and earlier allows local users to overwrite arbitrary files via a symlink attack.
CAN-1999-1348	** CANDIDATE (under review) ** Linuxconf on Red Hat Linux 6.0 and earlier does not properly disable PAM-based access to the shutdown command, which could allow local users to cause a denial of service.
CAN-2000-0017	** CANDIDATE (under review) ** Buffer overflow in Linux linuxconf package allows remote attackers to gain root privileges via a long parameter.

The most dangerous one is the remote buffer overflow attacks (CAN-2000-0017) and (CVE-1999-1327) where remote attackers can gain root via execution of arbitrary code.

5. Attack Mechanism:

The attack works by sending a packet with the SYN flag on to try and elicit a response if any hosts are listening on port 98 on the A.B.C.0/24 subnet. Note that

the reconnaissance scanning was done pretty fast for the 14 hosts so this was probably from an automated tool. This is a stimulus by the hostile IP looking for open TCP port 98. The packets do not appear to be crafted since the source ports increment and a legitimate flag is set (SYN), normal behavior for an initial TCP connection. There are a number of vulnerabilities that exist for linuxconf according to CVE but the most dangerous one is the remote buffer overflow vulnerability allowing for a remote creation of a shell on the target machine. According to Alepht1, "The vulnerability may be in the program's handling of HTTP headers."[3]. From what I have read there is an exploit code available somewhere where it uses a http post statement to tcp port 98 to overflow the buffer, unfortunately none so far in the security community had been able to use this exploit code successfully.

6. Correlations:

Comments on cve.mitre.org regarding Linuxconf and bugtraq mailing list have consistently reported that scans have took place for TCP port 98 for several months such as http://www.sans.org/y2k/032601.htm and http://www.sans.org/y2k/060100-1400.htm

7. Evidence of active targeting:

The attacker is searching for open Linuxconf port 98 on the A.B.C.0/24 subnet. Since the A.B.C.0/24 subnet has been targeted means that there is a strong evidence of active targeting of the network.

8. Severity:

Criticality- The importance of the host is not known (for all we know it could be also be webserver) but there has been no evidence that the exploit works, Criticality is set to 4.

Lethality- Reconnaissance attempts are not lethal. It just gives some useful insight that can be used later. Lethality is set to 2.

System Countermeasures- Looking at the logs, it is hard to say what kind of system countermeasures are in place. System Countermeasures is set to 1.

Network Countermeasures- Assuming that there is a firewall in place blocking the scans and the IDS caught the traffic. Also, perhaps only the stimuli was shown on the logs meaning logs had probably been previously sanitized since this may be normal procedure to most organizations. Network Countermeasures is set to 3.

[3] http://marc.theaimsgroup.com/?l=bugtraq&m=94580196627059&w=2

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
(4+ 2)– (1 + 3)= 2

9. Defensive Recommendation:

Checking machines that run linuxconf and determining if linuxconf is required to be running on them be patched up to date else disable linuxconf. Denying traffic coming from the outside towards port 98 is also a nice solution since we do not see any good reason for system configurations to be done remotely. Locally is probably acceptable but I am strongly against it and highly dangerous since linuxconf runs as root and has a web interface that allows anyone to login via http://yourmachine:98, so anybody can use that and try to guess the root password.

10. Multiple Choice Questions:

Buffer Overflows are due to:
A). It's a type of port scanning by sending a packet with the SYN/FIN flag set on
B). Not enough memory
C). Program or process tries to store more data in a buffer than it was intended to hold, normally because input is not checked.
D). When the Window size is 0x404

Answer: C

Network Detect # 3 Fast Distributed FTP Scan

10/26-06:59:12.559492 208.184.11.192:53290 -> www.xxx.yyy.4:21
TCP TTL:118 TOS:0x0 ID:629 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCCFD871B Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

10/26-06:59:12.572921 216.132.188.188:4275 -> www.xxx.yyy.10:21
TCP TTL:48 TOS:0x0 ID:8363 IpLen:20 DgmLen:44
******S* Seq: 0x2B9F9F0 Ack: 0x0 Win: 0x1C84 TcpLen: 24
TCP Options (1) => MSS: 1460

10/26-06:59:12.653053 208.184.11.192:53291 -> www.xxx.yyy.14:21
TCP TTL:118 TOS:0x0 ID:632 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCCFE7532 Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

10/26-06:59:12.735977 216.161.238.129:2740 -> www.xxx.yyy.12:21

TCP TTL:115 TOS:0x0 ID:35245 IpLen:20 DgmLen:44 DF
******S* Seq: 0x98C0740C Ack: 0x0 Win: 0x2000 TcpLen: 24
TCP Options (1) => MSS: 1460

10/26-06:59:12.793004 204.101.16.163:2754 -> www.xxx.yyy.16:21
TCP TTL:49 TOS:0x0 ID:44376 IpLen:20 DgmLen:44
******S* Seq: 0x114C54B0 Ack: 0x0 Win: 0x1C84 TcpLen: 24
TCP Options (1) => MSS: 1460

10/26-06:59:12.835849 216.158.34.123:1422 -> www.xxx.yyy.9:21
TCP TTL:117 TOS:0x0 ID:34740 IpLen:20 DgmLen:44 DF
******S* Seq: 0x513CFB0 Ack: 0x0 Win: 0x2000 TcpLen: 24
TCP Options (1) => MSS: 1460

10/26-06:59:15.059068 216.145.95.3:1171 -> www.xxx.yyy.8:21
TCP TTL:56 TOS:0x0 ID:21880 IpLen:20 DgmLen:44
******S* Seq: 0xA1DE0310 Ack: 0x0 Win: 0x1C84 TcpLen: 24
TCP Options (1) => MSS: 1460

10/26-06:59:17.129919 208.184.11.192:53298 -> www.xxx.yyy.6:21
TCP TTL:118 TOS:0x0 ID:667 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCD1628E9 Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK

10/26-06:59:17.163196 216.124.39.10:3749 -> www.xxx.yyy.2:21
TCP TTL:113 TOS:0x0 ID:24128 IpLen:20 DgmLen:44 DF
******S* Seq: 0x118ADB90 Ack: 0x0 Win: 0x2000 TcpLen: 24
TCP Options (1) => MSS: 1460

1. Source of Trace:

This trace came from http://www.incidents.org/archives/intrusions/msg01549.html

2. Detect was generated by:

Detect looks like it was captured with snort. If snort, the default snort portscan preprocessor should be able to capture this else it would have to be tweaked a little such as (Maybe this'll work but I never tried it):

```
# portscan: detect a variety of portscans
# ---------------------------------------
# portscan preprocessor by Patrick Mullen <p_mullen@linuxrc.net>
# This preprocessor detects UDP packets or TCP SYN packets going to
# one port in less than two seconds. "Stealth" TCP
# packets are always detected, regardless of these settings.

preprocessor portscan: $HOME_NET 1 2 portscan.log
```

3.Probability that the source address was spoofed:

The probability that the source address was spoofed is highly unlikely because the attacker is trying to gain some useful insight. Scan data would require return packet to verify that the socket is open. It seems like there are probably several unique source IP's by looking at Time to live fields, TCP options and Identification fields. For example, looking at 216.132.188.188 it has a ttl value of 48 (upper limit on the number of routers through which a datagram can pass which is initialized by the sender to some value), 216.158.34.123 has a different ttl of 117 and so is 216.145.95.3 which has a ttl value of 56, and so on. Also IP ID numbers are significantly apart meaning different hosts has sent these datagrams (because ID field is normally incremented by one each time a datagram is sent). 216.132.188.188 has an ID number of 8363, 216.158.34.123 has a greater number of ID of 34740 while 216.145.95.3 has an ID of 21880 and so on. Finally we check if they do come from different network blocks via Arin whois database. 208.184.11.192 comes from abovenet communications, 216.132.188.188 is choice one communications, 204.101.16.163 is worldlinx telecommunications, 216.158.34.123 belongs to DCA.net (Consult Dynamics, Inc), and so on.

4. Description of attack:

Network scan are in a range of IP's in the www.xxx.yyy.0/24 subnet and are milliseconds apart using 24 and 28 bytes. The logs show SYN attempts to TCP port 21 associated with ftp (File Transfer Protocol), the Internet standard for file transfer. There are a number of possible matches (174 cve entries) at cve.mitre.org when the string 'ftp' is searched but the most likely attacks following this reconnaissance are site exec attacks that can gain root access, buffer overflow attacks which allows execution of arbitrary code that can also gain root access. Below are some of FTP vulnerabilities that can be exploited via site exec commands and buffer overflows as seen from http://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=ftp.

CVE-1999-0219
CVE-1999-0349
CVE-1999-0368
CVE-1999-0080
CVE-1999-0955
CVE-1999-0097
CVE-2000-0573
CVE-2001-0318

5. Attack Mechanism:

The attack works by sending a packet with the SYN flag on to try and elicit a response if any hosts are listening on port 21 on the www.xxx.yyy.0/24 subnet.

Note that the scanning was done also pretty fast probably by an automated tool. This is a stimulus by the hostile IP's looking for open TCP port 21. The packets do not appear to be crafted since the source ports increment for the same source IP and a legitimate flag is set (SYN), normal behavior for an initial TCP connection.

6. Correlations:

Chris Calabrese discussed a similar detect in his GCIA practical and according to him, he was able to find plenty of other attacks that come from these same net blocks and that these net blocks are often used by attackers. However, He mentioned that the "packets we're seeing are initial login attempts" and payloads of 24 and 28 bytes" which We do not agree to since these packets are SYN packets, meaning the initial login attempts will even be tried until the three-way handshake has been completed. We do not understand what he meant by 24 and 28 bytes of payload since TCP will not have any payload since the packets only contain IP and TCP headers.

7. Evidence of active targeting:

The scans were fairly accurate, destined for port 21 on the www.xxx.yyy.0/24 subnet. Since the www.xxx.yyy.0/24 subnet has been targeted means that there is a strong evidence of active targeting of the network but not a specific host on that network.

8. Severity:

Criticality- FTP servers are fairly critical systems so criticality would probably be high. Criticality is set to 4.

Lethality- Reconnaissance gathering is not lethal. Lethality is set to 2.

System Countermeasures- Looking at the logs, it is hard to say what kind of system countermeasures are in place. System Countermeasures is set to 1.

Network Countermeasures- Assuming that there is a firewall in place blocking the scans and the IDS caught the traffic. Also, perhaps only the stimuli was shown on the logs meaning logs had probably been previously sanitized since this may be normal procedure to most organizations. Network Countermeasures is set to 3.

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures) = Severity
(4+ 2)– (1 + 3)= 2

9. Defensive Recommendation:

Auditing for systems that are running ftp and keeping them patched up to date and putting the offending IP's in the watchlist to monitor would be a good start.

10. Multiple Choice Test Question:

In the trace below, how many TCP options are enabled?

10/26-06:59:17.129919 208.184.11.192:53298 -> www.xxx.yyy.6:21
TCP TTL:118 TOS:0x0 ID:667 IpLen:20 DgmLen:48 DF
******S* Seq: 0xCD1628E9 Ack: 0x0 Win: 0x4000 TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
A. 48
B. 28
C. 4
D. 1460

Answer: C- There are 4 TCP options in this trace namely Maximum Segment Size of 1460 bytes, No Operation, No Operation and Selected Acknowledgement.

Assignment # 3 Analyze This

Introduction:

We have been asked to perform a security audit for GIAC University. GIAC has provided us with data from a Snort Intrusion Detection System that uses a fairly standard ruleset. With this data, we must analyze and evaluate GIAC University's information security. GIAC also has required us for inclusion in the report:

A list of the files chosen for analysis. At least five days' worth of "scans", "alerts", and "oos" (Out of Spec) files shall be used.
An overview or executive summary of our analysis.
A list of detects prioritized either by severity or number of occurrences and a brief description of these.
A "top talkers" list: The top ten talkers, in terms of detects.
A list of five external source addresses and registration information about these addresses, explained why we chose these.
Correlations from previous analysts' reports (numbered 209 and above).
A link graph and analysis of OOS files.
Any insights into internal machines such as compromise or possible dangerous or anomalous activity.
Defensive recommendations.

A description of our analysis process, the steps used when performing the analysis.
Meaningful analysis identifying relationships between the different computers that generated logs

Executive Summary:

We have finished conducting our analysis of the data supplied by GIAC University and has determined that there are some problems that need to be addressed, potentially compromised hosts, violations of sound security policies, and firewall and IDS ruleset configurations. This report will detail the investigation of the top 10 alerts reported during the data capture period, and defensive recommendations will also be given.

We feel that our ability to analyze GIAC University's security posture was handicapped because we were not provided with some important information such as network topology, the actual rulesets and Giac University's Security policies.

This report aims to provide the University with a picture of all intrusion attempts that occurred between March 24-28 2002 over a five-day period. The goal is to identify systems that exhibit signs of system compromise, identify successful and unsuccessful intrusion attempts. The following log files were used for the analysis.

Alerts	OOS	Scans
alert.020323	oos_Mar.24.2002	Scans.020323
alert.020324	oos_Mar.25.2002	Scans.020324
alert.020325	oos_Mar.26.2002	Scans.020325
alert.020326	oos_Mar.27.2002	Scans.020326
alert.020327	oos_Mar.28.2002	Scans.020327

List of top 10 Detects shown using SnortSnarf prioritized by number of occurrences:

Priority	Signature (click for sig info)	# Alerts	# Sources	# Dests	Detail link
N/A	SMB Name Wildcard	57280	131	115	Summary
N/A	SNMP public access	40021	21	146	Summary
N/A	connect to 515 from inside	34435	55	4	Summary
N/A	ICMP Echo Request L3retriever Ping	28323	90	12	Summary

N/A	MISC Large UDP Packet	22187	14	7	Summary
N/A	spp_http_decode: IIS Unicode attack detected	8555	78	427	Summary
N/A	INFO MSN IM Chat data	4462	64	64	Summary
N/A	INFO Inbound GNUTella Connect request	4248	3485	8	Summary
N/A	ICMP Echo Request Nmap or HPING2	3717	61	5	Summary
N/A	Watchlist 000220 IL-ISDNNET-990517	2915	13	6	Summary

Brief Description of the TOP 10 list of detects

1. SMB name wildcard- This is a standard netbios name retrieval query. These alerts are triggered when Windows machines send often exchange these queries as part of filesharing protocol to determine netbios names when only IPS are known. This type of query can gather important information such as workstation name, domain, and a list of currently logged users. TCP and UDP port 137 is the port used for Netbios name service that could also reveal unprotected windows shares.

Since the SMB name wildcard alerts were triggered from sources coming from MY.NET and destined for MY.NET, these are highly possible to be legitimate traffic and are due to the windows machines browsing the network. Changing the rule to reflect the source address as being !$HOME (or whatever variable for the internal network) but still be able to log those sources outside of MY.NET would make a lot of false positives go away. Also performing an audit into what these machines are actually sharing and who is accessing it is also a good idea.
Here are some of the logs captured:

03/23-00:00:16.241347 [**] SMB Name Wildcard [**] 192.170.11.6:137 -> 192.170.152.14:137
03/23-00:01:22.027366 [**] SMB Name Wildcard [**] 192.170.11.6:137 -> 192.170.152.11:137

2. SNMP public access- Depending on the nature of these machines providing SNMP data, this could be legitimate traffic. As best practice, the use of “public” community string should be avoided and filtering out SNMP at the border router since there were no recorded probes from outside of MY.NET is a good idea.
Here are some of the logs:

03/23-00:16:45.653951 [**] SNMP public access [**] 192.170.70.177:1068 -> 192.170.5.31:161
03/23-00:16:45.666947 [**] SNMP public access [**] 192.170.70.177:1068 -> 192.170.5.31:161

TCP and UDP port 161 is the port used for SNMP. However, SNMP has been historically plagued with a lot of vulnerabilities such as:

Name	Description
CVE-1999-0294	All records in a WINS database can be deleted through SNMP for a denial of service.
CVE-1999-0472	The SNMP default community name "public" is not properly removed in NetApps C630 Netcache, even if the administrator tries to disable it.
CVE-1999-0815	Memory leak in SNMP agent in Windows NT 4.0 before SP5 allows remote attackers to conduct a denial of service (memory exhaustion) via a large number of queries.
CVE-1999-1335	snmpd server in cmu-snmp SNMP package before 3.3-1 in Red Hat Linux 4.0 is configured to allow remote attackers to read and write sensitive information.
CVE-2000-0221	The Nautica Marlin bridge allows remote attackers to cause a denial of service via a zero length UDP packet to the SNMP port.
CVE-2000-0379	The Netopia R9100 router does not prevent authenticated users from modifying SNMP tables, even if the administrator has configured it to do so.
CVE-2000-0515	The snmpd.conf configuration file for the SNMP daemon (snmpd) in HP-UX 11.0 is world writable, which allows local users to modify SNMP configuration or gain privileges.
CVE-2000-1058	Buffer overflow in OverView5 CGI program in HP OpenView Network Node Manager (NNM) 6.1 and earlier allows remote attackers to cause a denial of service, and possibly execute arbitrary commands, in the SNMP service (snmp.exe), aka the "Java SNMP MIB Browser Object ID parsing problem."
CVE-2001-0236	Buffer overflow in Solaris snmpXdmid SNMP to DMI mapper daemon allows remote attackers to execute arbitrary commands via a long "indication" event.
CVE-2001-0487	AIX SNMP server snmpd allows remote attackers to cause a denial of service via a RST during the TCP connection.

In summary, an audit of machines running SNMP is advisable to make sure they are patched up to date.

3. Connect to 515 from inside- The connect to 515 from inside alert is triggered when a host from MY.NET connects to port 515 to MY.NET also. Port 515 for both TCP and UDP is associated with printer spooler or lpr service which basically queus print jobs. We can see that source and destinations are within MY.NET and could be legitimate print servers since most of the alerts for these destinations were connect to 515 from inside (A few had SMB name wildcard and L3retriver ping aside from connect to 515 from inside, see above and below for further discussions on these). However, there are a number of vulnerabilities that exist for the printer service such as:

Name	Description
CVE-1999-0032	Buffer overflow in BSD-based lpr package allows local users to gain root privileges.
CVE-1999-0335	Buffer overflow in BSD and linux lpr command allows local users to execute commands as root through the classification option.
CVE-1999-1102	lpr on SunOS 4.1.1, BSD 4.3, A/UX 2.0.1, and other BSD-based operating systems allows local users to create or overwrite arbitrary files via a symlink attack that is triggered after invoking lpr 1000 times.
CVE-2001-0906	teTeX filter before 1.0.7 allows local users to gain privileges via a symlink attack on temporary files that are produced when printing .dvi files using lpr.
CAN-1999-0020	** CANDIDATE (under review) ** ** REJECT ** Duplicate of CVE-1999-0032 ** REJECT ** Buffer overflow in Linux lpr command gives root access.

Here's a sample of the logs:

03/23-15:16:12.408761 [**] connect to 515 from inside [**] 192.170.153.125:3880 -> 192.170.150.198:515

03/23-15:16:12.424077 [**] connect to 515 from inside [**] 192.170.153.125:3880 -> 192.170.150.198:515

Changing the rule to reflect the source address as being !$HOME (or whatever variable for the internal network) would make the false positives go away. Also performing an audit if these machines are authorized spoolers is not a bad idea and making sure they are patched up to date.

4. ICMP Echo Request L3retriever Ping-Varied information I found on this type of ping could be caused by the following:

A. - L3 Networks security scanner called Retriever 1.5, now owned by Symantec uses this type of characteristic for ping. This legitimate security tool is used by security professionals for scanning their own network (hopefully) for authorized security assessment. However, Hackers can also use this tool to scan other networks maliciously. From the logs, there were two destination IPS not belonging to MY.NET, one was:

03/25-14:53:44.946399 [**] ICMP Echo Request L3retriever Ping [**] 192.170.153.163 -> 129.22.134.36
03/25-14:53:47.379004 [**] ICMP Echo Request L3retriever Ping [**] 192.170.153.163 -> 129.22.134.36

Whois 129.22.134.36
Case Western Reserve University
(NET-CWRUNET)
Campus Communications Network - Network Services
Crawford Hall, Room 426
Cleveland, OH 44106
US

Netname: CWRUNET
Netblock: 129.22.0.0 - 129.22.255.255

Coordinator:
Gumpf, Jeffrey A (JAG3-ARIN) Gumpf@INS.CWRU.EDU
(216) 368-2982

Domain System inverse mapping provided by:

NS.CWRU.EDU	129.22.4.1
NS2.CWRU.EDU	129.22.4.3
NCNOC.NCREN.NET	192.101.21.1

Record last updated on 22-Oct-1999.
Database last updated on 14-Jul-2002 19:59:49 EDT.

B. It could also mean that the L3retriever ping was caused by another reason, such as this type of ping can be caused by plain w2k host talking to w2k domain controllers. This was correlated by Joshua Wright and John Berkers at the snort-users mailing list. (http://www.mcabee.org/lists/snort-users/Aug-01/msg01234.html) Here are some more of the logs:

03/23-00:00:02.008798 [**] ICMP Echo Request L3retriever Ping [**] 192.170.152.158 -> 192.170.11.6
03/23-00:00:16.239659 [**] ICMP Echo Request L3retriever Ping [**] 192.170.152.14 -> 192.170.11.6

5. MISC Large UDP Packet- This type of alert could be caused by the quotes around the rule for example,

alert udp !$HOME_NET any -> $HOME_NET any (msg:"IDS247 - MISC - Large UDP Packet"; dsize: ">800";)

Removing the quotes around the >800 should make the false positives go away. However, The new rulesets have already been updated to fix this problem. The thread on this discussion could be found at http://archives.neohapsis.com/archives/snort/2000-06/0330.html
Perhaps GIAC university is still using an old ruleset that does not yet correctly address this problem. Also, according to Debrata Dash, large UDP traffic can also be associated with streaming media. (http://www.incidents.org/archives/intrusions/msg03704.html)
Here are some of the logs,

03/26-12:15:05.906469 [**] MISC Large UDP Packet [**] 66.28.104.154:1608 -> 192.170.153.153:3783
03/25-15:10:14.730738 [**] MISC Large UDP Packet [**] 140.142.8.72:2031 -> 192.170.153.157:2876
03/25-15:33:44.995952 [**] MISC Large UDP Packet [**] 202.101.232.110:1354 -> 192.170.153.159:1304

All of the source IP's were external to MY.NET and it would be wise to find out if MY.NET destinations (MY.NET.153.153,MY.NET.153.157 and MY.NET.153.159) have any connections to IP addresses 66.28.104.154 (cogent communications),140.142.8.72 (NorthWestNet Network Operations Center) and 202.101.232.110 (CHINANET Jiangxi province network Data Communication Division China Telecom) respectively. These can provide us insight into what could these Misc Large UDP packets be. Denying traffic from these sources would be a good preventive measure until things can be sorted out that these are indeed legitimate traffic.

6. spp_http_decode: IIS Unicode attack detected- There were a lot of alerts for IIS Unicode attack detected such as:

03/26-08:33:49.665025 [**] spp_http_decode: IIS Unicode attack detected [**] 192.170.153.197:1137 -> 207.68.162.250:80
03/26-08:36:31.193540 [**] spp_http_decode: IIS Unicode attack detected [**] 192.170.153.197:1245 -> 211.32.117.27:80

03/26-08:36:32.741165 [**] spp_http_decode: IIS Unicode attack detected [**] 192.170.153.197:1256 -> 211.233.29.233:80

This type of alert tries to identify hostile traffic by interpreting unicode data as an attempt to obfuscate an attack. Based on the varying distribution of source and destination addresses these alerts appear to be false positives. These alerts were probably triggered by visiting sites that uses multibyte characters such as traditional Chinese, simplified Chinese, Korean, Japanese. Making sure that your signatures are up to date and using –unicode with the http_decode preprocessor may reduce the number of false positives. The URL/URI's will still be normalised and directory traversal signatures and others will still be able pick up actual attacks. More correlation on this common false alert can be found at: (http://archives.neohapsis.com/archives/snort/2001-08/0528.html)

However, It could also indicate that a host is infected with code red worm. A lot of sources for these alerts come from MY.NET and might indicate that these sources are infected with code red as correlated by Philip Stadler in his practical. It is suggested that these MY.NET hosts be checked for code red worm and cleaned if necessary.

192.170.153.197	192.170.153.177
192.170.153.127	192.170.153.115
192.170.152.19	192.170.153.111
192.170.153.154	192.170.153.107
192.170.88.162	192.170.153.114

7. INFO MSN IM Chat data- Messenger Service used by Hotmail members. These can be considered miscellaneous activity. The logs show that these alerts were generated by MY.NET hosts communicating with hotmail chat servers and vice versa.

03/27-14:06:48.942286 [**] INFO MSN IM Chat data [**] 192.170.153.177:4945 -> 64.4.12.156:1863
03/27-14:07:10.138600 [**] INFO MSN IM Chat data [**] 192.170.153.177:4945 -> 64.4.12.156:1863
03/27-14:07:14.044689 [**] INFO MSN IM Chat data [**] 192.170.153.177:4945 -> 64.4.12.156:1863
03/25-11:49:52.762014 [**] INFO MSN IM Chat data [**] 64.4.12.158:1863 -> 192.170.150.242:1209
03/25-11:50:13.692981 [**] INFO MSN IM Chat data [**] 64.4.12.158:1863 -> 192.170.150.241:1273

Whois 64.4.12.156

Search results for: 64.4.12.158

MS Hotmail (NETBLK-HOTMAIL)
1065 La Avenida
Mountain View, CA 94043
US

Netname: HOTMAIL
Netblock: 64.4.0.0 - 64.4.63.255

Coordinator:
Myers, Michael (MM520-ARIN) icon@HOTMAIL.COM
650-693-7072

Domain System inverse mapping provided by:

NS1.HOTMAIL.COM	216.200.206.140
NS3.HOTMAIL.COM	209.185.130.68
NS2.HOTMAIL.COM	216.200.206.139
NS4.HOTMAIL.COM	64.4.29.24

Record last updated on 15-Jul-2002.
Database last updated on 15-Jul-2002 20:00:47 EDT.

GIAC University should decide on a policy whether to allow this type of activity to occur or if not, deny traffic to port 1863.

8. INFO Inbound GNUTella Connect request- This alert is associated with "GNUTella" protocol allowing file sharing on port 6346 of any host connected to the GNUTella network. GNUtella allows you to share any files you want to everyone connected to the GNUTella network or download or search for files you want. Basically, it is a mini search engine and file serving system in one. The GNUTella protocol works different from a client-server type in that clients become servers and servers become clients all at once, accomplished by creating a sort of distributed environment. Like, acting as a server to people who want the files on your machine, and you act as a client to access files on other people's machines. Logs show that a couple of MY.NET hosts are participating in sharing their files to the GNUTella Network namely:

192.170.153.45	192.170.153.178
192.170.153.159	192.170.150.209
192.170.153.196	192.170.152.21
192.170.153.191	192.170.88.194

IT would depend on GIAC University if this type of file sharing is accepted. If Not, Denying traffic to port 6346 would be a start.

9. ICMP Echo Request Nmap or HPING2- ICMP echo requests are normally used for mapping networks to get additional information of destination networks. A lot of implementations of the PING program create unique echo request packets that can be associated with the signature of the Nmap or HPING2 programs (Among others). These two programs can be used to scan for open ports on hosts by permitting protocol parameters to be set as command line options. Here are some of the logs:

03/23-00:00:19.632134 [**] ICMP Echo Request Nmap or HPING2 [**]
192.170.152.21 -> 192.170.11.6
03/23-01:56:22.656324 [**] ICMP Echo Request Nmap or HPING2 [**]
192.170.152.21 -> 192.170.11.6
03/23-00:45:00.210690 [**] ICMP Echo Request Nmap or HPING2 [**]
192.170.152.169 -> 192.170.11.7
03/27-14:56:45.672717 [**] ICMP Echo Request Nmap or HPING2 [**]
192.170.88.212 -> 207.46.131.30
03/27-14:57:06.080930 [**] ICMP Echo Request Nmap or HPING2 [**]
192.170.88.212 -> 207.46.131.30

The logs show that all source addresses were from MY.NET destined to most hosts in the MY.NET network and three external destination hosts namely: Microsoft, Absolute software and Hewlett Packard. Microsoft for example, does not let icmp echo requests or responses through their firewall and the logs that were captured were probably from nmap used to portscan Microsoft with the –P0 or –PT80 option. Note that source address can be spoofed since both nmap and hping2 programs have an option to spoof source address but if source IP's were spoofed one reason could be to fill up the IDS logs to try and hide the real attacks.

10. Watchlist 000220 IL-ISDNNET-990517- Watchlist alerts are generated by all traffic originating from 212.179.X.X destined for MY.NET. This means that these hosts placed on the watchlist for surveillance possibly because of suspicious activity. Below are some of the sample log files captured

03/25-15:26:45.192070 [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.118:80 -> 192.170.153.181:2531
03/25-15:26:45.193368 [**] Watchlist 000220 IL-ISDNNET-990517 [**]
212.179.35.118:80 -> 192.170.153.181:2531

The netblock 212.179.0.0 through 212.179.255.255 belongs to the IL-ISDNNET-990517 for ISDN Net, Ltd in Israel:

% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.

% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 212.179.0.0 - 212.179.255.255
netname: IL-ISDNNET-990517
descr: PROVIDER
country: IL
admin-c: NP469-RIPE
tech-c: TP1233-RIPE
tech-c: ZV140-RIPE
tech-c: ES4966-RIPE
status: ALLOCATED PA
mnt-by: RIPE-NCC-HM-MNT
changed: hostmaster@ripe.net 19990517
changed: hostmaster@ripe.net 20000406
changed: hostmaster@ripe.net 20010402
source: RIPE

route: 212.179.0.0/17
descr: ISDN Net Ltd.
origin: AS8551
notify: hostmaster@isdn.net.il
mnt-by: AS8551-MNT
changed: hostmaster@isdn.net.il 19990610
source: RIPE

person: Nati Pinko
address: Bezeq International
address: 40 Hashacham St.
address: Petach Tikvah Israel
phone: +972 3 9257761
e-mail: hostmaster@isdn.net.il
nic-hdl: NP469-RIPE
changed: registrar@ns.il 19990902
source: RIPE

Most of the traffic were destined for ports 2561,2587,1434,2144,2531,2750. Of all the ports the most known for a number of vulnerabilities is port 1434, associated with MS-SQL. Here are some of the vulnerabilities of MS-SQL

There are 70 CVE entries or candidates that match your search.
CVE version: 20020625

CAN-2002-0567	** CANDIDATE (under review) ** Oracle 8i and 9i with PL/SQL package for External Procedures (EXTPROC) allows remote attackers to bypass authentication and execute arbitrary functions by using the TNS Listener to directly connect to the EXTPROC process.

CAN-2002-0571	** CANDIDATE (under review) ** Oracle Oracle9i database server 9.0.1.x allows local users to access restricted data via a SQL query using ANSI outer join syntax.
CAN-2002-0581	** CANDIDATE (under review) ** WorkforceROI Xpede 4.1 allows remote attackers to execute arbitrary SQL commands and read, modify, or steal credentials from the database via the Qry parameter in the sprc.asp script.

MY.NET.153.120 should be audited if MS-SQL should be running and make sure that it is patched up to date.

Criteria Used for Alerts, Scans and OOS - To be considered a “TopTalker”, large number of alerts must be triggered with the prospective host as the source address. Note: MY.NET is replaced with “192.170” since snortsnarf would fail to process if the default “MY.NET” is used.

A. Alert Analysis

Top 10 Talkers

Count	Src IP Address	Top Alerts
20905	192.170.70.177	SNMP Public Access
17756	192.170.11.6	SMB Name Wildcard
10805	66.28.104.154	Misc Large UDP Packet
9663	192.170.11.7	SMB Name Wildcard
6203	140.142.8.72	Misc Large UDP Packet
5604	192.170.153.125	Connect to 515 from Inside
5238	192.170.150.198	SNMP Public Access
3229	192.170.153.197	Spp_http_decode: IIS Unicode Attack Detected
3148	192.170.153.115	Connect to 515 from Inside
3000	192.170.153.120	Connect to 515 from Inside

We will focus on 4 interesting IP address in the top talkers list above (See Bolded), two that is external to MY.NET and two hosts from MY.NET.

1. Whois information for 66.28.104.154:
Cogent Communications (NETBLK-COGENT-NB-0000)
1015 31st Street, NW
Washington, DC 20007
US

Netname: COGENT-NB-0000

Netblock: 66.28.0.0 - 66.28.255.255
Maintainer: COGC

Coordinator:
Cogent Communications (ZC108-ARIN) noc@cogentco.com
+1-877-875-4311

Domain System inverse mapping provided by:

AUTH1.DNS.COGENTCO.COM 66.28.0.14
AUTH2.DNS.COGENTCO.COM 66.28.0.30

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
Reassignment information for this block can be found at
rwhois.cogentco.com 4321

All queries from this IP address account for 10805 counts of "MISC Large UDP Packet" alerts. Source ports and destination ports constantly change with no distinguishable pattern but all of them are above the 1024 reserved port numbers. Large UDP packets can be associated with streaming media, denial of service or use of a covert channel. Since, Cogent Communications is an ISP for service providers and commercial end users, I am guessing it's streaming media of some sort. See article below:
(http://cert.uni-stuttgart.de/archive/intrusions/2002/03/msg00071.html)

2. Whois information for 140.142.8.72
NorthWestNet Network Operations Center (NET-UW-SEA)
Academic Computing Center
3737 Brooklyn NE
Seattle, WA 98105
US

Netname: UW-SEA
Netblock: 140.142.0.0 - 140.142.255.255
Maintainer: UWND

Coordinator:
University, Of Washington (OWU2-ARIN) noc@CAC.WASHINGTON.EDU
206-543-5128

Domain System inverse mapping provided by:

HANNA.CAC.WASHINGTON.EDU 140.142.5.5
MARGE.CAC.WASHINGTON.EDU 140.142.5.13
NS.UNET.UMN.EDU 128.101.101.101

Record last updated on 17-Mar-2000.
Database last updated on 14-Jun-2002 20:01:02 EDT.

Traffic is similar to the above except the ephemeral source port of 2031 change and destination port of 2876 does not change. Port 2876 is associated with sps-tunnel. However, a search of sps-tunnel did not yield any interesting results.

3. MY.NET.153.125

This source is generating 5470 alerts on connect to port 515 from inside. Port 515 of the Unix Printer service has a tradition of high severity vulnerabilities (root). However, the destination host looks like it's a valid print server. This host is also generating alerts on spp_http_decode: IIS Unicode attack detected (137) and spp_http_decode: CGI null byte attack detected (2) which I believe are false positives. Why false positives? For example, this traffic

03/25-14:32:43.006310 [**] spp_http_decode: IIS Unicode attack detected [**] 192.170.153.125:4984 -> 211.233.28.70:80

whois 211.233.28.70

remarks: This IP address space has been allocated to KRNIC.
remarks: For more information, using KRNIC Whois Database
remarks: whois -h whois.nic.or.kr
remarks: This information has been partially mirrored by APNIC from
remarks: KRNIC. To obtain more specific information, please use the
remarks: KRNIC whois server at whois.krnic.net.
mnt-by: MNT-KRNIC-AP
changed: hostmaster@nic.or.kr 20020610
source: KRNIC

person: Hanju Kim
country: KR
phone: +82-2-6446-6407
fax-no: +82-2-6446-6499
e-mail: hankim@daumcorp.com
nic-hdl: HK4035-KR
remarks: This information has been partially mirrored by APNIC from
remarks: KRNIC. To obtain more specific information, please use the
remarks: KRNIC whois server at whois.krnic.net.
mnt-by: MNT-KRNIC-AP
changed: hostmaster@nic.or.kr 20020610
source: KRNIC

We see its registered from Korea. Unicode attacks are mostly just false positives or could also indicate code red worm activity. There is a more lengthy

explanation on this subject and defensive recommendations in my brief description of detects above. (See # 6 on brief description of top ten detects)

4. MY.NET.153.197

Again a lot of alerts (2260) are generated from spp_http_decode: IIS Unicode attack detected. A whois on one of the destination IP's yield:

Korea Internet Information Service V1.0 (created by KRNIC, 2001.6)

2001³â 7¿ù 2ÀÏºÎÅÍ´Â °³¼±µÈ Whois ¼ºñ½º¸¦ Àû¿ëÇÏ°í ÀÖ½À´Ï´Ù.

query: 211.32.117.27

ENGLISH

KRNIC is not ISP but National Internet Registry similar with APNIC.
Please see the following end-user contacts for IP address information.

IP Address : 211.32.116.0-211.32.119.255
Network Name : DACOM-KIDC
Connect ISP Name : BORANET
Connect Date : 19991015
Registration Date : 20000601

[Organization Information]
Orgnization ID : ORG105718
Org Name : DACOM
State : SEOUL
Address : 261-1 Nonhyun-dong Kangnam-gu

As we see, it's basically the same thing as what triggered the alert above (either code red worm or just false positive (see explanation above or # 6 on brief description of top ten detects).

Top ten Alert Destination IPS

Count	Dest IPs	Top Alerts
37834	192.170.11.6	SMB Name Wildcard and ICMP L3Retriever Ping
34416	192.170.150.198	Connect to 515 from Inside
20718	192.170.11.7	SMB Name Wildcard and ICMP L3Retriever

		Ping
10878	192.170.153.153	Misc Large UDP Packet
7236	192.170.150.195	SNMP Public Access
6204	192.170.153.157	Misc Large UDP Packet
4751	192.170.153.159	Misc Large UDP Packet
4152	192.170.5.248	SNMP Public Access
3510	192.170.152.109	SNMP Public Access
3474	192.170.5.96	SNMP Public Access

Top 10 Alert Destination Ports

Count	Port #	Purpose
57280	137	Netbios
40021	161	Snmp
34435	515	Printer
20088	192.170.11.6	See explanation
11070	192.170.11.7	See explanation
10494	80	Http, www
6234	2876	Sps-tunnel
4805	6346	Gnutella
2225	1863	Msn
2176	1304	Boomerang

Explanation: These are return messages caused ICMP. No ports existed for these types of messages. There were a lot of ICMP echo request (NMAP or HPING2 and L3Retriever) pings all coming from internal MY.NET machines. According to the Whitehats website, L3Retriever pings are associated W2k Machines talking to W2k domain controllers. The rest of the services seem normal except that I could not find any meaningful information (search on Google) on SPS-Tunneling and Boomerang on what they are for.

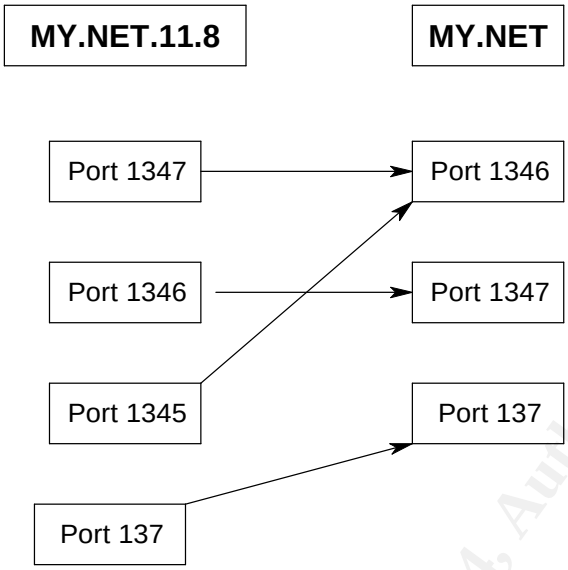
B. Scans Analysis

TOP TEN Talkers

Count	Source IP Address
691153	MY.NET.11.8
411714	MY.NET.60.43
156547	MY.NET.150.113
44966	MY.NET.150.143
24696	MY.NET.6.52
21698	MY.NET.152.21
20955	MY.NET.6.50

18407	MY.NET.6.49
18194	64.124.157.32
16541	MY.NET.6.45

MY.NET.11.8 is the number 1 top talker in the list and indicates mass scans to various hosts in the MY.NET network destined to ports 1346, 1347 and 137 using UDP. With that mind, we have decided to illustrate the scope of MY.NET.11.8's scanning activity on MY.NET network using a simple Link graph.



Here are some of the logs:

MY.NET.11.8 Port 1347 TO MY.NET port 1346
Mar 23 00:00:04 000.111.11.8:1347 -> 000.111.152.46:1346 UDP
Mar 23 00:00:05 000.111.11.8:1347 -> 000.111.152.215:1346 UDP
Mar 23 00:00:06 000.111.11.8:1347 -> 000.111.152.15:1346 UDP

MY.NET.11.8 Port 1346 TO MY.NET Port 1347
Mar 24 16:41:37 000.111.152.179:1346 -> 000.111.11.8:1347 UDP
Mar 24 16:41:42 000.111.152.247:1346 -> 000.111.11.8:1347 UDP

MY.NET.11.8 Port 1345 TO MY.NET Port 1346
Mar 26 15:03:22 000.111.11.8:1345 -> 000.111.152.16:1346 UDP
Mar 26 15:49:23 000.111.11.8:1345 -> 000.111.152.171:1346 UDP
Mar 26 16:22:26 000.111.11.8:1345 -> 000.111.152.16:1346 UDP

MY.NET.11.8 Port 137 To MY.NET.5.50 Port 137
Mar 27 15:55:10 000.111.11.8:137 -> 000.111.5.50:137 UDP
Mar 27 15:55:13 000.111.11.8:137 -> 000.111.5.50:137 UDP
Mar 27 16:07:17 000.111.11.8:137 -> 000.111.5.50:137 UDP

MY.NET.11.8 source ports are 1347,1346,1345 and 137. The first three are all over 1024 and therefore can be considered “emphemeral”. However, destination port 1347 is associated with bbn-mmc (Multi-Media conferencing), 1346 (Alta Analytics License Manager), (btw, 1345 VPJP has something to do with Multi-Media. See http://archives.neohapsis.com/archives/incidents/2000-11/0033.html for more details) Port 137 is associated with Netbios name server and it looks like MY.NET.11.8 is just querying a Netbios nameserver (MY.NET.5.50). We can see that packets destined for ports 1346 and 1347 are milliseconds apart and uses UDP suggesting that traffic may be some sort of syncronization Conferencing. Using “grep” to count the number of occurances, there was around 691143 counts for scanning of port 1346, 9942 counts for port 137 and 10 counts for port 137.

On the other hand, MY.NET.60.43 seems to be an NTP server (network Time protocol) since the source port is 123, commonly associated with NTP. Here are some of the logs:

Mar 23 01:04:41 000.111.60.43:123 -> 000.111.153.207:1526 UDP
Mar 23 01:04:43 000.111.60.43:123 -> 000.111.153.161:1624 UDP
Mar 23 01:04:43 000.111.60.43:123 -> 000.111.153.184:1785 UDP
Mar 23 01:04:45 000.111.60.43:123 -> 000.111.153.149:1496 UDP
Mar 23 01:04:45 000.111.60.43:123 -> 000.111.153.193:1489 UDP
Mar 23 01:04:52 000.111.60.43:123 -> 000.111.153.197:1506 UDP
Mar 23 01:04:52 000.111.60.43:123 -> 000.111.153.200:1644 UDP
Mar 23 01:04:53 000.111.60.43:123 -> 000.111.153.140:1755 UDP
Mar 23 01:04:53 000.111.60.43:123 -> 000.111.153.177:1453 UDP

NTP is a UDP based protocol to syncronize clocks of networked computers. NTP has a buffer overflow vulnerability wherein an attacker can execute shellcode. (**http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0414**)

Name	CVE-2001-0414
Description	Buffer overflow in ntpd ntp daemon 4.0.99k and earlier (aka xntpd and xntp3) allows remote attackers to cause a denial of service and possibly execute arbitrary commands via a long readvar argument.

Miika Turkia discussed a very well written white paper on the subject of ntpd exploit in his practical. It is very hard to tell if these logs were actual attacks or false positives on MY.NET.60.43 because we do not see the packet dump if it contains the sequence number 2 and data length of zero which are used to query for the internal variables of the NTP server with corresponding values that can execute shellcode. To detect an actual attack on the NTP server, one way is to check for NOP machine instruction in the command part of the NTP control

message since normal NTP traffic do not have these values. A custom snort rule that could be used is:

alert udp any any -> $HOME_NET 123 (msg: "ntpd exploit attempt"; content:"|16|";offset:0;depth:1;content:"|90|"; offset:12;depth:32;)

This rule checks that it is an NTP control message and if command contains NOP. Updating to the latest version of NTP and/or keeping up to date with patches is also a good idea. If there are no patches available, adding “restrict default ignore” to ntp.conf and “restrict MY.NET.60.43 nomodify” can be a quick fix. (this setting denies queries from the NTP server but syncing is still possible, since source address can be spoofed making it look like the attack came from the server itself)

MY.NET.150.113 and MY.NET.150.143 are scanning for hosts running the edonkey2000 P2P file sharing system[4] which listens to incoming TCP connections at port 4662 and incoming UDP connections on port 4665. Here’s a small excerpt from the scans logs that shows what I mean:

Mar 27 23:06:21 MY.NET.150.143:3665 -> 61.216.201.133:4662 SYN******S*
Mar 27 23:06:23 MY.NET.150.143:3667 -> 61.224.10.228:4662 SYN ******S*
Mar 27 23:06:34 MY.NET.150.143:1057 -> 140.122.197.182:4665 UDP
Mar 27 23:06:34 MY.NET.150.143:1057 -> 150.162.165.8:4665 UDP
Mar 27 23:06:35 MY.NET.150.143:1057 -> 129.125.148.105:4665 UDP
Mar 27 23:06:35 MY.NET.150.143:1057 -> 129.194.101.117:4665 UDP
Mar 27 23:06:35 MY.NET.150.143:1057 -> 195.112.128.222:4665 UDP

Mar 23 00:01:03 000.111.150.113:1257 -> 194.109.18.201:4665 UDP
Mar 23 00:01:04 000.111.150.113:1257 -> 62.226.96.110:4665 UDP
Mar 23 00:01:05 000.111.150.113:1257 -> 80.130.249.225:4665 UDP
Mar 23 00:01:11 000.111.150.113:1989 -> 80.3.202.112:4662 SYN ******S*
Mar 23 00:01:11 000.111.150.113:1990 -> 141.156.237.157:4662 SYN ******S*

While MY.NET.6.52, MY.NET.6.50, MY.NET.49 and MY.NET.6.45 seems to be mounting remote AFS mounts on various university systems, MY.NET.152.21 seems to be scanning for GNUTella services (port 6346) on various hosts outside the MY.NET network and 64.124.157.32 seems to be scanning only one host MY.NET.153.46 for open ports using UDP. Whois 64.124.157.32

Abovenet Communications, Inc. (NETBLK-ABOVENET)
50 W. San Fernando Street, Suite 1010
San Jose, CA 95113
US

Netname: ABOVENET

[4] http://homepage.ntlworld.com/robin.d.h.walker/cmtips/p2p.html

Netblock: 64.124.0.0 - 64.125.255.255
Maintainer: ABVE

Coordinator:
Metromedia Fiber Networks/AboveNet (NOC41-ORG-ARIN)
noc@ABOVE.NET
408-367-6666
Fax- 408-367-6688

Domain System inverse mapping provided by:

NS.ABOVE.NET 207.126.96.162
NS3.ABOVE.NET 207.126.105.146

ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE

Record last updated on 27-Apr-2001.
Database last updated on 19-Jun-2002 20:01:03 EDT.

Note: SPP_portscans in the alerts file also matched the top 10 talkers in the scans files.

C. OOS ANALYSIS

Top Ten Talkers

Count	Source IP Address
29	80.133.124.114
4	213.169.245.41
2	128.97.84.53
1	80.145.117.134
1	80.144.189.160
1	62.31.125.225
1	*61.216.83.124*
1	61.198.200.52
1	217.82.123.75
1	213.3.191.240

There were 29 counts of out of spec packets sent by source IP 80.133.124.114 destined for MY.NET.150.113 port 1214.

80.133.124.114 comes from an ISP in Germany. It's interested in MY.NET.150.113 at port 1214 because MY.NET.150.113 seems to be offering filesharing service called KAZAA. It looks like legitimate traffic except that this machine sets the reserved bits: The EOL is a delimiter and may be used as a padding for NOP. With TTLs at 39 and random TCP sequence numbers but otherwise well-formed packets, this would seem to indicate the use of ECN

(explicit congestion notification) bits from the source rather than as a malicious crafted packet from queso mapping attempt.

=+

03/28-06:44:37.146162 80.133.124.114:3687 -> MY.NET.150.113:1214
TCP TTL:39 TOS:0x0 ID:19815 DF
21S***** Seq: 0x96BB59AE Ack: 0x0 Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 5321 0 EOL EOL EOL EOL

=+

03/28-06:47:29.683553 80.133.124.114:3837 -> MY.NET.150.113:1214
TCP TTL:39 TOS:0x0 ID:125 DF
21S***** Seq: 0xA09B58E0 Ack: 0x0 Win: 0x16B0
TCP Options => MSS: 1412 SackOK TS: 22685 0 EOL EOL EOL EOL

=+

The next top talker with only 4 counts is a machine seems to be sending illegal packets (see flags) to only one host, MY.NET.152.21 at port 6346. Port 6346 is associated with another filesharing service called GNUTella but as we can see while the syn bit is set (among others) acknowledgement number is greater than zero so there is evidence of mangling of packet headers. There is a correlation to this with the demon net router incident were a router in demon.net was performing a mangling of packet headers possibly due to stress in heavy traffic loading. We also see the big "jumps" of ID number in a small amount of time (see below, from ID 408 to ID being 5784 to 8618) suggesting this machine was pretty busy. Given the pattern of these packets, this would seem to be more likely reason than malicious packet crafting as the cause.

03/27-15:24:28.649944 213.169.245.41:3800 -> MY.NET.152.21:6346
TCP TTL:110 TOS:0x0 ID:408 DF
2*SF*P*U Seq: 0x3F7473 Ack: 0x20736D61 Win: 0x6564
68 65 61 64 20 77 69 74 68 20 head with

=+

03/27-15:24:29.845479 213.169.245.41:3800 -> MY.NET.152.21:6346
TCP TTL:110 TOS:0x0 ID:5784 DF
2*SF*P*U Seq: 0x3F7473 Ack: 0x20736D61 Win: 0x6564
68 65 61 64 20 77 69 74 68 20 head with

=+=
+=+=+

03/27-15:28:06.137307 213.169.245.41:3800 -> MY.NET.152.21:6346
TCP TTL:110 TOS:0x0 ID:8618 DF
2*SF*PA* Seq: 0x4143B2 Ack: 0xF3660ABC Win: 0x970A
TCP Options => Opt 17

=+=
+=+=+

03/27-15:29:17.651596 213.169.245.41:3800 -> MY.NET.152.21:6346
TCP TTL:110 TOS:0x0 ID:23472 DF
21SF*P** Seq: 0x410005 Ack: 0x2549DEE5 Win: 0x4A9D
34 CB 4A 9D 1C 41 A5 88 BA 8F 76 80 01 06 1D 00 4.J..A....v.....
00 00 00 00 6E 37 ....n7

On March 25,2002 128.97.84.53 was caught in the oos files attempting a SYN/FIN scan on destination host MY.NET.150.113:4662. Port 4662 is associated with a lesser known P2P filsharing application called edonkey2000. There were no reported incidents that edonkey2000 is vulnerable to an exploit but I am not ruling out that possibility. Furthermore, What is interesting to note is that there had been no reports on edonkey2000 by the other students in the GIAC program and/or in any security mailing lists. Putting 128.97.84.53 on the watchlist or blocking it and verifying the security of MY.NET.150.113 would be good idea.

=+=
+=+=+
03/25-20:19:25.415397 128.97.84.53:4242 -> MY.NET.150.113:4662
TCP TTL:47 TOS:0x0 ID:28183 DF
2*SF**** Seq: 0xD7266F63 Ack: 0x2E688172 Win: 0xD388
54 5A E0 9D 09 37 86 81 1A 89 84 BD TZ...7......
=+=
+=+=+

Whois 128.97.84.53

University of California, Los Angeles (NET-UCLANET)
741 Circle Dr South
Los Angeles, CA 90095-1363
US

Netname: UCLANET
Netblock: 128.97.0.0 - 128.97.255.255

Coordinator:
University of California, Los Angeles (NO102-ORG-ARIN)
noc@NOC.UCLA.EDU
+1 310 206 5345

Domain System inverse mapping provided by:

DNS.UCLA.EDU 164.67.128.1
DNS2.UCLA.EDU 164.67.128.2
DNS3.UCLA.EDU 164.67.128.3
NS2.BERKELEY.EDU 128.32.136.12 128.32.206.12

Record last updated on 31-Oct-2000.
Database last updated on 17-Jul-2002 20:01:06 EDT.

While on march 28, another host performing seeming malicious activity was caught in the oos files initiating a SYN connection to MY.NET.153.210 at port 113 (ident) with the reserved bits on. However, this also looks like of ecn variety rather than of crafted packet. (see above, explanation on 80.133.124.114)

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+=+
03/28-14:03:11.864168 128.97.84.53:2075 -> MY.NET.153.210:113
TCP TTL:50 TOS:0x0 ID:58086  DF
21S***** Seq: 0xF0B15061   Ack: 0x0   Win: 0x16D0
TCP Options => MSS: 1460 SackOK TS: 110473863 0 EOL EOL EOL EOL

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
+=+=+
```

This particular host seems to be also doing a reconnaissance on MY.NET.150.113 since it has bogus TCP flags set also (SYN,FIN,RST, A, Urgent) all at the same time. As I've mentioned before, Port 1214 is associated with KAZAA. Acknowledgement number is a non-zero value and the seemingly inverted use of the End of Option List preceding the options could indicate a crafted packet. IT seems to be a source host coming from a cable company in Belgium, Germany.

```
03/27-18:10:02.788706 213.132.137.149:3504 -> MY.NET.150.113:1214
TCP TTL:105 TOS:0x0 ID:10760  DF
**SFR*AU Seq: 0x24977AE   Ack: 0xE4EA88   Win: 0x5010
TCP Options => EOL EOL EOL EOL
00 00
```

These additional detects would seem to indicate hostile intent by 213.132.137.149.

03/27-18:21:07.456878 [**] spp_portscan: PORTSCAN DETECTED from 213.132.137.149
(STEALTH) [**]
03/27-18:21:09.155735 [**] spp_portscan: portscan status from 213.132.137.149:
1 connections across 1 hosts: TCP(1), UDP(0) STEALTH [**]
03/27-18:21:11.202681 [**] spp_portscan: End of portscan from 213.132.137.149:
TOTAL time(0s) hosts(1) TCP(1) UDP(0) STEALTH [**]

Whois 213.132.137.149

% This is the RIPE Whois server.
% The objects are in RPSL format.
% Please visit http://www.ripe.net/rpsl for more information.
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-services/db/copyright.html

inetnum: 213.132.128.0 - 213.132.143.255
netname: TVD-INTERNET
descr: TVD Internet - UPC Belgium - Chello
descr: ISP - CATV operator Brussels/Leuven
descr: Customer DHCP pools
country: BE
admin-c: TVD-RIPE
tech-c: TVD-RIPE
rev-srv: bruns00.chello.com
rev-srv: bruns01.chello.be
rev-srv: ns1.chello.at
status: ASSIGNED PA
remarks: If you suspect unusual activity originating from this network:
remarks: send relevant logfiles, IP adresses, protocol and port numbers,
remarks: UTC timestamps and other useful information to abuse@chello.be
remarks: in plain ascII text. Do not send HTML, proprietary encodings,
remarks: graphical formats or mime attachments. Improper use of the
remarks: changed attribute is not tolerated.
notify: ripe-notify@tvd.be
mnt-by: AS8733-MNT
changed: steven@tvd.be 20011112
source: RIPE

route: 213.132.128.0/20

```
descr:       UPC Belgium - Chello Belgium
descr:       Cable Internet Provider
origin:      AS8733
mnt-by:      AS8733-MNT
remarks:     If you suspect unusual activity originating from this network:
remarks:     send relevant logfiles, IP adresses, protocol and port numbers,
remarks:     UTC timestamps and other useful information to abuse@chello.be
remarks:     in plain ascII text. Do not send HTML, proprietary encodings,
remarks:     graphical formats or mime attachments. Improper use of the
remarks:     changed attribute is not tolerated.
changed:     steven@tvd.be 20011101
source:      RIPE

role:        TVD Internet
address:     UPC Belgium - Chello Belgium - Radio Public
address:     Chazallaan 140
address:     1030 Schaarbeek
address:     BELGIUM
phone:       +32 2 2400800
fax-no:      +32 2 2400800
e-mail:      ripe-role@tvd.be
trouble:     trouble@tvd.be or +32 2 2400899 (08:00-18:00 UTC)
admin-c:     SVS4-RIPE
admin-c:     SB9000-RIPE
tech-c:      PVE8-RIPE
nic-hdl:     TVD-RIPE
notify:      ripe-notify@tvd.be
mnt-by:      AS8733-MNT
changed:     steven@tvd.be 19990105
changed:     steven@tvd.be 20020301
source:      RIPE
```

It is suggested that the offending IP is reported to the ISP as indicated on their remarks together with the relevant information so that they make take action and putting the source IP in the watchlist could be a good idea.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+
03/28-15:48:28.582159 0.192.5.106:19169 -> MY.NET.153.191:33376
TCP TTL:112 TOS:0x0 ID:932  DF
2*SF***U Seq: 0x41DCEE70   Ack: 0x1021A743   Win: 0xEBCB
30 08 1A D5 57 B8 6F 70 75 70 6B 69 6E 67 2E 64  0...W.opupking.d
65 0A                                            e.
```

Source address is spoofed, illegal flag combinations and acknowledgement number is non-zero would indicate a malicious crafted packet. Verifying the security of MY.NET.153.191 is advised.

Lastly, 61.216.83.124 has sent an xmas packet to MY.NET.150.220 with a non zero acknowledgement number would indicate a malicious crafted packet also. Auditing MY.NET.150.220 and checking whatever service may be running on 4662 is a good idea and putting source IP 61.216.83.124 in the watchlist. IP address resolves to an ADSL subscriber in Taiwan.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+
03/28-23:09:52.462171 61.216.83.124:64835 -> MY.NET.150.220:4662
TCP TTL:107 TOS:0x0 ID:14647  DF
2*SFRPAU Seq: 0xBAB12DF0   Ack: 0xA4B9   Win: 0xACB3
BA B1 2D F0 00 00 A4 B9 13 7F AC B3 14 69 1C A2  ..-.........i..
0D F3 26 AB 3F A5 13 C9                          ..&.?...

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
+=+=+
```

Whois 61.216.83.124

```
% Rights restricted by copyright. See http://www.apnic.net/db/dbcopyright.html
% (whois6.apnic.net)

inetnum:     61.216.0.0 - 61.219.255.255
netname:     HINET-TW
descr:       CHTD, Chunghwa Telecom Co.,Ltd.
descr:       Data-Bldg.6F, No.21, Sec.21, Hsin-Yi Rd.
descr:       Taipei Taiwan 100
country:     TW
admin-c:     HN27-AP
tech-c:      HN28-AP
remarks:     Delegated to HiNet for ADSL subscriber.
remarks:     This information has been partially mirrored by APNIC from
remarks:     TWNIC. To obtain more specific information, please use the
remarks:     TWNIC whois server at whois.twnic.net.
mnt-by:      TWNIC-AP
changed:     hostmaster@twnic.net 20010117
source:      APNIC

person:      HINET Network-Adm
address:     CHTD, Chunghwa Telecom Co., Ltd.
address:     Data-Bldg. 6F,  No. 21, Sec. 21, Hsin-Yi Rd.,
address:     Taipei Taiwan 100
```

country: TW
phone: +886 2 2322 3495
phone: +886 2 2322 3442
phone: +886 2 2344 3007
fax-no: +886 2 2344 2513
fax-no: +886 2 2395 5671
e-mail: network-adm@hinet.net
nic-hdl: HN27-AP
remarks: same as TWNIC nic-handle HN184-TW
mnt-by: TWNIC-AP
changed: hostmaster@twnic.net 20000721
source: APNIC

person: HINET Network-Center
address: CHTD, Chunghwa Telecom Co., Ltd.
address: Data-Bldg. 6F, No. 21, Sec. 21, Hsin-Yi Rd.,
address: Taipei Taiwan 100
country: TW
phone: +886 2 2322 3495
phone: +886 2 2322 3442
phone: +886 2 2344 3007
fax-no: +886 2 2344 2513
fax-no: +886 2 2395 5671
e-mail: network-center@hinet.net
nic-hdl: HN28-AP
remarks: same as TWNIC nic-handle HN185-TW
mnt-by: TWNIC-AP
changed: hostmaster@twnic.net 20000721
source: APNIC

inetnum: 61.216.0.0 - 61.217.255.255
netname: HINET-NET
descr: Chunghwa Telecom Data communication Business Group
descr: No.21, Hsin-Yi Rd., sec. 1
descr: Taipei Taiwan
country: TW
admin-c: CYK-TW
tech-c: CYK-TW
remarks: This information has been partially mirrored by APNIC from
remarks: TWNIC. To obtain more specific information, please use the
remarks: TWNIC whois server at whois.twnic.net.
mnt-by: TWNIC-AP
changed: network-adm@hinet.net 20010925
source: TWNIC

person: Chung Yung Kang

address: Chunghwa Telecom Data communication Business Group
address: No.21, Hsin-Yi Rd., sec. 1
address: Taipei Taiwan
country: TW
phone: +886-2-2322-3442
fax-no: +886-2-2344-2513
e-mail: cykang@ms1.hinet.net
nic-hdl: CYK-TW
remarks: This information has been partially mirrored by APNIC from
remarks: TWNIC. To obtain more specific information, please use the
remarks: TWNIC whois server at whois.twnic.net.
changed: hostmaster@twnic.net 19990924
source: TWNIC

Top Ten Destination Ports

Count	Port
30	1214
10	6346
3	4662
1	80
1	33376
1	23
1	1320
1	113

Top Ten Destination IPS

31	MY.NET.150.113
5	MY.NET.152.21
2	MY.NET.153.45
2	MY.NET.153.210
2	MY.NET.153.191
2	MY.NET.150.220
1	MY.NET.5.79
1	MY.NET.153.196
1	MY.NET.153.159
1	MY.NET.150.226

Meaningful analysis identifying relationships between the different computers that generated logs

In order to determine the threats to each host a host profile was built which formed the basis for the analysis of all data. It identifies hosts that could be considered prime targets. This profile was built by inferring the purpose of each host from the attacks that were logged against it. To minimize false positives, additional analysis was performed on the questionable data.

The TOP 20 alert destination MY.NET hosts and the purposes of each are shown below. It is recommended that the University review this list to verify the services running on each host.

DESTINATION	PURPOSE
192.170.11.6	NETBIOS
192.170.150.198	PRINTER
192.170.11.7	NETBIOS
192.170.153.153	PROXY
192.170.150.195	SNMP
192.170.153.157	UNKNOWN
192.170.153.159	GNUTELLA
192.170.5.248	SNMP
192.170.152.109	SNMP
192.170.5.96	SNMP,HTTP
192.170.5.137	SNMP
192.170.5.143	SNMP
192.170.5.31	SNMP
192.170.5.97	SNMP
192.170.5.127	SNMP
192.170.151.114	SNMP FTP
192.170.113.202	SNMP
192.170.5.244	SNMP
192.170.153.45	GNUTELLA
192.170.153.191	FTP,GNUTELLA
192.170.5.50	NETBIOS

Insights into internal machines that could be compromised:

There are a lot of hosts in the alert logs that were scanning other hosts, mostly in the MY.NET network. The hosts in the table below show those who have had at least more than 1000 counts of being guilty of “portscanning” in the five-day period of logs. It is our guess that these machines are compromised/dangerous. The counts given are for the number of hosts these machines have scanned. It is our recommendation that this list be used as a starting point in identifying the correct hosts in the MY.NET that may be compromised.

Count	IP Address
106728	192.170.11.8:
95841	192.170.60.43:
11296	192.170.150.113:
5259	192.170.88.212:
4872	192.170.150.143:
4397	192.170.60.43
4339	192.170.150.71:

4278	192.170.150.113
3490	192.170.152.21:
3206	192.170.151.105:
2819	192.170.6.45:
2562	192.170.151.70:
2525	192.170.11.6:
2371	192.170.153.196:
2295	192.170.6.60:
2149	192.170.153.178:
2094	192.170.151.85:
2090	192.170.149.19:
1974	192.170.11.6
1943	192.170.153.159:
1931	192.170.149.11:
1922	192.170.149.43:
1664	192.170.6.53:
1538	192.170.150.143
1510	192.170.6.52:
1491	192.170.153.186:
1431	192.170.88.207:
1418	192.170.153.197:
1401	192.170.11.7:
1367	192.170.153.181:
1363	192.170.149.12:
1298	192.170.149.56:
1201	192.170.153.157:
1183	192.170.11.7
1154	192.170.6.48:
1115	192.170.152.171:
1107	192.170.6.49:
1049	192.170.88.203:
1029	192.170.152.174:
1012	192.170.88.183:

Defensive Recommendation

The following list above should be taken offline and thoroughly examined for signs of compromise. If they are compromised, they must be cleaned and before they are returned to the network, First, their system countermeasures must be strengthened via up to date patches, installing a host-based IDS and using remote syslogging can help prevent the hosts succumbing to the same exploits used and at least correctly detect hosts that become compromised. Installation and maintenance of a more restrictive firewall policy, usually a “deny-all” approach and only allow specific services that are necessary in and out of the

network. Furthermore, bogus packets should be instantly dropped by the firewall (or router) such as:

- Incoming packets outside the firewall that claim to come from the internal network
- Outgoing packets from inside the firewall that claim to come from outside the network
- All incoming packets with illegal flags set
- Incoming packets outside the firewall to printer port 515
- Incoming packets outside the firewall to the Netbios ports (port 137,138,139 and 445)

We also recommend that if GIAC University use a stateful firewall (If it's not using one already). Traditional firewalls only look at individual packets not recognizing that they are part of a larger whole, such as a connection. Stateful firewalls can do this, thus being able to check packets if they belong to an established connection or not. This is important so that incoming TCP packets with the ACK flag set but do not belong to an established connection can be dropped.

Specific Recommendations would be to refining the rules for large number of alerts on legitimate traffic such file sharing and instant messaging (if these are permitted to prevent excessive alerts generated on legitimate traffic but there should also be user education on network security since these types of applications offers a conduit where malicious software can enter your network), SMB name wildcard (SMB alerts and L3Retriever Ping alerts should be triggered if sources are external to MY.NET destined to MY.NET), SNMP public access public string should be avoided. We also realize that all traffic from internal users cannot be assumed not malicious but making sure systems are patched up to date can offer some form of protection against these malicious internal users.

Description of the Analysis Process

First I downloaded all the necessary files from incidents.org/logs. A total of 15 files were gathered. Five from each of the alerts, scans and oos logs. I chose those logs that were "complete", meaning those that had consecutive and continuous dates for five days and those that were pretty recent. I narrowed it to log files of March 24-28 2002. Since I was using windows 98 I had to download additional applications in order to complete my analysis. I wanted to use snortsnarf and I also figured I'd need to use certain unix tools such as sort, uniq, grep, etc that were not *unfortunately* available in windows 98. Incidentally, those unix tools were the main utilities that were used for the bulk of this analysis. So I downloaded activeperl from activestate's website activestate.com and cygwin from cygwin.com. After I set this up, I proceeded to download snortsnarf from http://www.silicondefense.com/software/snortsnarf/index.htm After I untarred snortsnarf and downloaded the necessary log files, I concatenated all of the log files into three large files: alert_final, scan_final and

oos_final. I then removed duplicate entries. Since snortsnarf “barks” at MY.NET. entries, I changed them to something that was not used. I then ran snortsnarf on the alerts file. The full snortsnarf output is at appendix A.

Some examples of Commands that were used in the analysis process

Alerts

```
1. $ grep "\[\*\*\]" alert_final | grep -v spp_portscan | grep -v Tiny\ Fragments
| grep -v ICMP\ SRC | cut -d '>' -f 2 | cut -d ' ' -f 2 | cut -d ':' -f 2 | tr
-d ' ' | sort | sort | uniq -c | sort -nr > alert_dstports-test.log

2. $ grep "\[\*\*\]" alert_all | grep -v spp_portscan | cut -d '>' -f 2 | cut -d '
 ' -f 2 | cut -d ':' -f 1 | tr -d ' ' | sort | sort | uniq -c | sort -nr > aler
t.destips.tester.log

grep "\[\*\*\]" alerts.txt | grep -v spp_portscan | cut -d \] -f 3 | cut -d \- -f 1 | cut -d : -
f 1 | sed s/\ //g >> alerts.srcips.log.unsorted
grep PORTSCAN alerts.txt | cut -d \] -f 2 | cut -d \  -f 6 | sed s/\ //g >>
alerts.srcips.log.unsorted
cat alerts.srcips.log.unsorted | sort | uniq -c | sort -nr > alerts.srcips.log
rm alerts.srcips.log.unsorted
```

SCANS

```
#tally number of src ips
$ grep '..:..:' scan_final | cut -d '>' -f 1 | cut -d ' ' -f 4 | cut -d '
:' -f 1 | tr -d ' ' | sort | uniq -c | sort -nr > scan_src_ips_test.log

#tally number of dst ips
$ grep '..:..:' scans_final| cut -d '>' -f 2 | cut -d ':' -f 1  | tr -d '
 ' | sort | uniq -c | sort -nr > scan_dst_ips_test.log

#tally number of dst ports
$ grep '..:..:' scan_final | cut -d '>' -f 2 | cut -d ':' -f 2 | cut -d '
 ' -f 1 |  tr -d ' ' | sort | uniq -c | sort -nr > scan.dst.ports.log
```

OOS

Tally of dst ips:

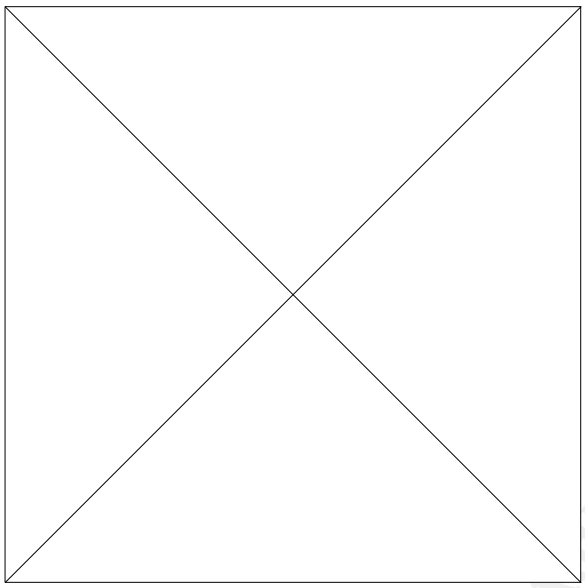
```
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c |
sort -nr > oos.dstips.log
```

Tally of dst ports:

```
grep "..\/..\-..\:..\:"  oos.txt | cut -d \> -f 2 | cut -d \: -f 2 | sed s/\ //g | sort | uniq -c |
sort -nr > oos.dstports.log
```

Tally of src ips:
grep "..\/..\-..\:..\:" oos.txt | cut -d \> -f 1 | cut -d \ -f 2 | cut -d \: -f 1 | sed s/\ //g | sort | uniq -c | sort -nr > oos.srcips.log

Appendix A:



SnortSnarf start page
All Snort signatures
SnortSnarf v020516.1

Signature section (217068) | Top 20 source IPs | Top 20 dest IPs

217068 alerts found using input module SnortFileInput, with sources:
alert_final
Earliest alert at 00:00:02.008798 *on 03/23/2002*
Latest alert at 23:54:52.240002 *on 03/27/2002*

Priority	Signature (click for sig info)	# Alerts	# Sources	# Dests	Detail link
N/A	SMB Name Wildcard	57280	131	115	Summary
N/A	SNMP public access	40021	21	146	Summary
N/A	connect to 515 from inside	34435	55	4	Summary
N/A	ICMP Echo Request L3retriever Ping	28323	90	12	Summary
N/A	MISC Large UDP Packet	22187	14	7	Summary
N/A	spp_http_decode: IIS Unicode attack detected	8555	78	427	Summary
N/A	INFO MSN IM Chat data	4462	64	64	Summary

N/A	INFO Inbound GNUTella Connect request	4248	3485	8	Summary
N/A	ICMP Echo Request Nmap or HPING2	3717	61	5	Summary
N/A	Watchlist 000220 IL-ISDNNET-990517	2915	13	6	Summary
N/A	ICMP Fragment Reassembly Time Exceeded	2197	24	55	Summary
N/A	High port 65535 udp - possible Red Worm - traffic	1594	68	106	Summary
N/A	FTP DoS ftpd globbing	1499	15	2	Summary
N/A	INFO Outbound GNUTella Connect request	1448	8	977	Summary
N/A	WEB-IIS view source via translate header	797	34	1	Summary
N/A	ICMP Router Selection	760	93	1	Summary
N/A	WEB-MISC Attempt to execute cmd	755	19	28	Summary
N/A	INFO FTP anonymous FTP	297	7	22	Summary
N/A	WEB-FRONTPAGE _vti_rpc access	177	71	2	Summary
N/A	WEB-IIS _vti_inf access	176	71	2	Summary
N/A	Port 55850 tcp - Possible myserver activity - ref. 010313-1	151	6	6	Summary
N/A	Watchlist 000222 NET-NCFC	146	2	2	Summary
N/A	SCAN Proxy attempt	140	21	39	Summary
N/A	Null scan!	138	27	11	Summary
N/A	Possible trojan server activity	109	10	10	Summary
N/A	ICMP Destination Unreachable (Communication Administratively Prohibited)	74	1	1	Summary

N/A	ICMP Echo Request Windows	66	17	5	Summary
N/A	INFO - Possible Squid Scan	47	13	8	Summary
N/A	NMAP TCP ping!	45	14	6	Summary
N/A	INFO Possible IRC Access	43	11	15	Summary
N/A	WEB-CGI scriptalias access	34	3	1	Summary
N/A	Incomplete Packet Fragments Discarded	27	4	4	Summary
N/A	EXPLOIT x86 NOOP	26	15	16	Summary
N/A	INFO Napster Client Data	24	2	18	Summary
N/A	WEB-MISC 403 Forbidden	18	3	6	Summary
N/A	ICMP traceroute	15	8	2	Summary
N/A	SCAN Synscan Portscan ID 19104	14	14	7	Summary
N/A	Queso fingerprint	10	9	6	Summary
N/A	ICMP Destination Unreachable (Protocol Unreachable)	9	1	1	Summary
N/A	suspicious host traffic	8	6	2	Summary
N/A	EXPLOIT x86 setuid 0	7	7	6	Summary
N/A	spp_http_decode: CGI Null Byte attack detected	6	5	3	Summary
N/A	WEB-IIS Unauthorized IP Access Attempt	6	2	4	Summary
N/A	TCP SRC and DST outside network	5	1	1	Summary
N/A	EXPLOIT x86 setgid 0	5	5	4	Summary
N/A	BACKDOOR NetMetro Incoming Traffic	5	1	1	Summary
N/A	WEB-MISC http directory traversal	4	1	1	Summary
N/A	INFO Inbound GNUTella	4	3	3	Summary

	Connect accept				
N/A	EXPLOIT NTPDX buffer overflow	4	2	2	Summary
N/A	MISC traceroute	4	1	1	Summary
N/A	Attempted Sun RPC high port access	4	2	2	Summary
N/A	WEB-MISC compaq nsight directory traversal	3	3	3	Summary
N/A	ICMP Echo Request CyberKit 2.2 Windows	2	1	2	Summary
N/A	ICMP Echo Request BSDtype	2	1	2	Summary
N/A	MISC PCAnywhere Startup	2	2	2	Summary
N/A	Back Orifice	2	2	2	Summary
N/A	RFB - Possible WinVNC - 010708-1	2	2	2	Summary
N/A	Port 55850 udp - Possible myserver activity - ref. 010313-1	2	2	2	Summary
N/A	EXPLOIT x86 stealth noop	2	2	2	Summary
N/A	IDS50/trojan_trojan-active-subseven [arachNIDS]	1	1	1	Summary
N/A	x86 NOOP - unicode BUFFER OVERFLOW ATTACK	1	1	1	Summary
N/A	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	1	1	1	Summary
N/A	TFTP - Internal UDP connection to external tftp server	1	1	1	Summary
N/A	WEB-MISC prefix-get //	1	1	1	Summary
N/A	WEB-MISC webdav search access	1	1	1	Summary
N/A	WEB-IIS encoding access	1	1	1	Summary
N/A	SYN-FIN scan!	1	1	1	Summary

N/A	TFTP - External UDP connection to internal tftp server	1	1	1	Summary
N/A	ICMP Echo Request Sun Solaris	1	1	1	Summary

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Sun Jun 9 13:22:52 2002

References:

http://www.eecis.udel.edu/~ntp/

http://dshield.org

http://www.activestate.com

Chris Baker GCIA Practical

Chris Calabrese GCIA Practical

http://www.cert.org

http://www.cve.mitre.org

http://www.cygwin.com

http://www.giac.org/GCIA.php

http://www.iana.org/assignments/port-numbers

http://www.silicondefense.com/software/snortsnarf/index.htm

http://www.snort.org/snort-db/all.html

Kyle Haugsness GCIA Practical

Lorraine Weaver GCIA Practical

Stevens, W. Richard. TCP/IP Illustrated, Volume 1

http://www.redhat.com/support/manuals/RHL-6.2-Manual/ref-guide/ch-sysconfig.html

http://www.solucorp.qc.ca/linuxconf/

http://www.xc.org/jonathan/linuxconf-rh5.1-faq.html

http://www.isc.org/products/BIND/

http://advice.networkice.com/Advice/Intrusions/2000417/default.htm

http://www.rixsoft.com/Knowbuddy/gnutellafaq.html#resgnutella

http://www.sans.org/y2k/ecn.htm

http://cr.yp.to/djbdns.html

http://www.incidents.org/react/lion.html

http://www.portsdb.org

http://rr.sans.org

http://www.rff.com

http://www.whitehats.com

http://www.snort.org

http://enterprisesecurity.symantec.com/pdf/retreiv.pdf?PID=na&EID=1

http://www.insecure.org/nmap/

http://www.eaglenet.org/antirez/hping2.html

http://homepage.ntlworld.com/robin.d.h.walker/cmtips/p2p.html

http://marc.theaimsgroup.com/?l=bugtraq&m=94580196627059&w=2

Last Updated: July 23rd, 2017

Upcoming Training

SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201709,	Sep 11, 2017 - Oct 18, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS Scottsdale SEC503	Scottsdale, AZ	Oct 02, 2017 - Oct 07, 2017	Community SANS
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced