Global Information Assurance Certification Paper

GCIA Intrusion in Depth - Assignment v3.3

Holger van Lengerich

Submitted: May 05, 2003

Part 1: Passive Fingerprinting Using Timestamps

Abstract:

In this paper I will show, that in some cases timestamps which are logged without sending a stimulus can be correlated to a schedule of a specific protocol implementation. As proof of concept I will show how timestamps correspond to consecutive retries of a TCP-Connection request (TCP-SYN) can be used to fingerprint the TCP implementation at the source.

Introduction

For the Detect #1 of part 2 of this practical I wanted to correlate the timing pattern of consecutive RPC retries to a specific client implementation but could not find any reference data on the net. It seems to me that sets of recorded timestamps are not being used for passive fingerprinting. So I wrote this paper, because I think correlating sets of timestamps to implementation dependent schedules can be a usefull addition to conventional passive fingerprinting methods.

Passive Fingerprinting

I would describe passive fingerprinting as the art of making educated guesses about the OS or Software running on a remote host by only evaluating network traffic, which is only observed passively and by no means actively stimulated. It is a powerfull tool for intrusion analysts to get information about the capabilities of their adversaries, without letting the latter know that someone got suspicious and is now watching for them.

Previous papers (e.g. [2],[4]) describe passive fingerprinting by correlating protocol parameters found in payload captured from network with default values, which are known to be used by distinct protocol implementations. E.g. to fingerprint an IP implementation parameters like Time-To-Live (TTL), Identification (ID) and "Don't Fragment-flag" (DF) can be used. In the references section you find papers describing some of these techniques, which I do not reiterate here, because I assume they are common knownledge among intrusion analysts.

Although techniques for passive fingerprinting are already very sophisticated, there are cases, when additional criteria to passively assess a protocol implementation running on a remote host might come in handy, e.g.:

- **no high fidelity log data is available**

  Filtering devices like firewalls and routers most commonly only log a timestamp, source and destination IP-address, IP-Subprotocol (e.g. ICMP, UDP, TCP) and if applicable source and destination port. As none of the logged protocol parameters can be set by choice of a specific protocol implementation, known passive fingerprinting techniques cannot be applied to such data.
- **need of more correlation**

  As passive fingerprinting is always guesswork, deducted results can only be hold up with a certain probability. If an analyst is in doubt about his results or wants more confidence about his results, he needs to harden his evidence by evaluating more criteria.
- **passive fingerprinting evasion**

  While passive fingerprinting techniques were evolving in the past, techniques to evade passive fingerprinting did also. The easiest way to evade passive fingerprinting is to configure the source host via the designated interfaces of the

corresponding OS as shown in [2] (e.g. modifying values through the proc-filesystem of Linux).

A more sophisticated approach is implemented by a public available fingerprinting evasion tool for GNU/Linux called IP Personality, which was developed by Gaël Roualland and Jean-Marc Saffroy [3]. As IP Personality is hooked via ip-tables directly into the Linux network-layer, it is able to modify the protocol data of any packet which traverses the TCP/IP-Stack of the host it runs on. As GNU/Linux has the capability to act as router, IP Personality can also be used to decoy a TCP/IP-Stack running on a remote host.

Previous Work On Using Timestamps For Fingerprinting

In April 2002 a research team consisting of Franck Verysset, Olivier Courteay and Olivier Heen from Intranode Software Technologies released an article [6] which publicized how consecutive timestamps received as responses to an actively sent out stimulus can be used for remote fingerprinting of TCP implementions. A linux based proof of concept tool called Ring which can be downloaded from http://www.intranode.com/en/site/techno/techno_articles.htm was also released. A nice evaluation of this tool can be found in the GCIA practical[7] of Tod Beardsley.

What Protocol-Schedules Can Be Fingerprinted?

As comparing timestamp schedules with reference data can be applied to any protocol, where application specific timing can be measured, most promising results are archieved , when following conditions are met:

- The protocol implementation which is to be fingerprinted, sends packets due to it's own implemented schedule.

  Timestamps cannot be used to fingerprint protocols which have no own timing behaviour to be recognized. So it is unlike that connectionless protocols like IP, UDP or syslog have recognizable timing patterns themself. Timestamps of IP packets which carry TCP or UDP packets which embed DNS protocol information, may be correlated to a specific TCP or DNS implementation if some retransmission schedule is triggered because no response is received.

- the source must not get any response which affect their behaviour

  E.g. if a TCP stack gets the awaited response, it has no need to send the retries which are necessary to measure the characteristic timing. If the TCP stack gets any failure messages, like "ICMP unreachable" or TCP "reset", it will most likely abort the connection attempt.

  This is the most limitating factor of fingerprinting on timestamps. But as dropped packets are often logged by filter-devices like firewalls, these logs are well suited for passive fingerprinting on timestamps. As stated above, they often don't contain any other information than timestamps which could be used for fingerprinting.

- the network-latency between source to the location of observation should be fairly constant

  The ideal environment for recognizing the schedule used at the source host, would be a network where all packets will need a constant time to get from source to the point where the timestamps are recorded.

Proof of Concept

Searching for a protocol suited to proof that implementation defined schedules can be used for remote fingerprinting, I choose TCP for the following reasons:

- TCP is used by many applications
- TCP connection attempts are logged by a broad variety of devices
- TCP is known to ignore "ICMP host unreachable" errors. (s. [1], p.318)

As firewall logs seem to me like an appealing target for passive remote OS detection, I choose to fingerprint the retransmission schedules used by TCP implementations for the initial packet (SYN) of a connection attempt. TCP is a normally a central managed resource of an OS and is normally implemented at the kernel or system layer. So by correlating

the timestamps of TCP protocol data to reference data, assumptions about the senders OS can be made.

As the command "`telnet <hostname>`" is available on all platforms I want to test, it was chosen as connection initiator. As telnet just calls OS functions to initiate a TCP connection, the results still will apply to the TCP layer of the corresponding OS and are not connected to the telnet application.

building a reference database

To observe timestamps I use a Intel Pentium II PC with Debian GNU/Linux (unstable branch) installed. This host had an already running firewall set up and TCP to port 23 (Telnet) amongst other traffic is blocked at the network layer. So no response will be created to answer incoming traffic on the telnet port. The observation hosts and the sample hosts used for generating reference data are physically connected to the same fast ethernet switch. To record a set of timestamps I proceed with the following steps:

1. Starting tethereal on the observation host:
   ```
   # tethereal -f port 23
   ```
   As other hosts were connected to the switch used in this experiment, tethereal was called with a capture filter (`-f port 23`).
2. Starting telnet on the sample host:
   ```
   # telnet observation-host
   ```
3. Waiting until telnet times out the connection request on the sample host.
4. Terminating tethereal on the observation host and collect output. (Figure 1.1 shows a sample tethereal output.)

```
# tethereal -i eth1  -f port 23
Capturing on eth1
 0.000000 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
 2.995703 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
 8.995882 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
20.995155 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
44.995719 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
92.996863 sample-suse -> observation-host TCP 24322 > telnet [SYN] Seq=2010522157 Ack=0 Win=5840 Len=0
```

Figure 1.1: tethereal records timestamps from a source running SuSE Linux 8.1

For each sample host I repeat this procedure least 3 times.

Linux and Solaris show up with constant results each time. Windows 2000, Windows XP and Mac OS X seem to choose an slightly variable first retransmission time out (RTO). Further RTOs are allmost constant in each generated trace. (Most probably this odd timing of the 1st retry is related to a BSD derived TCP implementation, which is described in [1], p.235f. So we most likely have found a fingerprint of BSD TCP implementation in the TCP/IP Stacks of Windows and Mac OS X.) To get a corridor where estimated timestamps have to fit in, 10 more traces were generated for each of this 3 OS. The rounded results are presented in table 1.1.

Another anomaly of MAC OS X's fingerprint which is worthy of mention is, that the first 5 Packets were all sent with a constant RTO of 3s. After the 5th timeout the RTO is doubling and exponential backoff, which is recommended by [5] is used.

Operating system	Linux 2.4.19-4GB (SuSE8.1)	Solaris 8 (Sparc)	Windows 2000 Professional	Windows XP Professional	Mac OS X 10.2.5
recorded timestamps (in ms)	0.00 3.00 9.00 21.00 45.00 93.00	0.00 3.36 10.11 23.62 50.61 104.61 164.61	0.00 3.18 + x 9.75 + x 0 < x < 0.068	0.00 + x 2.87 + x 8.88 + x 0 < x < 0.098	0.00 2.52 + x 5.52 + x 8.52 + x 11.52 + x 14.52 + x 20.52 + x 32.52 + x 56.52 + x

					0 < x < 0.405
# recorded timestamps	6	7	3	3	9

Table 1.1: recorded timestamps

These results show, that only counting all retries is sufficient to get a clue about the distinct TCP implementation. The timestamps are also distinct and can be used for fingerprinting.

Real World Scenario

For a test under real life conditions I choose following setup:

From two hosts, which are both connected to a german universtity campus, I connect to a firewalled telnet port of a host which is located in a production environment of my employer. As traceroute tells, the logging device is 14 hops away from the source host and traffic from source to destination travels through the backbone of a german science network towards the central german peering DeCIX. From DeCIX packets are routed through the backbone of my employer towards a Netscreen 10 firewall. Both hosts are connected to switched fast ethernets and both sites have gigabit uplinks uplinks to the corresponding backbones.

The Netscreen 10 Firewall located in a data center of my employer is configured to drop and log packets sent to the telnet port via syslog to a remote loghost. The resolution of the timestamps in the recorded logdata is 1 second.

original	**normalized**
12:23:20	0s
12:23:23	3s
12:23:29	9s
12:23:41	21s
12:24:05	45s
12:24:53	93s

Table 1.2: 1st set of timestamps recorded by Netscreen 10

The first set of timestamps recorded in the Netscreen logs can be found in table 1.2. As the count of the timestamps is 6, the source may be a host running Linux 2.4. After subtracting the 1st timestamp from all timestamps in this set, we get a normalized list of timestamps, printed in the 2nd column of table 1.2. This list exactly maches the recorded reference data for Linux 2.4. As a matter of fact the tested host is running Redhat 7.3 at with an self-compiled Linux-Kernel 2.4.20 and the the TCP implementation is correctly identified.

original	**normalized**
11:32:46	0s
11:32:50	4s
11:32:57	11s
11:33:10	24s
11:33:37	51s
11:34:31	105s
11:35:32	164s

Table 1.3: 2nd set of timestamps recorded by Netscreen 10

The second set of recorded timestamps can be found in table 1.3. As the count of the timestamps is 7 it seems to be a Solaris 8 host. As we subtract the 1st timestamp from all timestamps, we get the following normalized list: 0s,4s,11s,24s,51s,105s,164s, which maches the recorded reference data for Solaris 8. As this host was running Solaris 9, the TCP implementation could be correctly identified and it seems that Sun has not changed this behaviour in the more recent Solaris 9.

Conclusion & Perspective

It was shown, that a set of timestamps from an unsuccessful initiation af a TCP connection can be used to fingerprint the TCP implementation used at the source host. This is, because it seems that each TCP implementation of a protocol has its own characteristic behaviour about when and how much retries will be sent, if an awaited response is missing.

I assume other protocols and even other parts of the TCP implementation offer timing behaviours which can be used for passive implementation recognition. The method of collecting sets of timestamps for fingerprinting is generic to any protocol as far as its implementations show distinct and measurable timing schedules. I expect that following protocols also may have implementations with distinct inherent timing schedules usable for passive fingerprinting:

- SunRPC
- DNS
- SMB/Netbios
- Keyexchange protocols for IPSec
- *to be continued ...*

An analyst recognizing protocol implementation specific timing patterns on encrypted channels may also guess what is transmitted over tunneling protocols like IPSec, PPTP, etc.

As I only want to show, that remote os guessing is possible based on recorded sets of timestamps, I did not cover any methods to deal with errors of measurements. Sources of such errors include missing accuracy of timestamps, interfering network traffic or packet loss. Applying statistics to the method described in this paper will improve its usefulness method as well as the quality of the derived results.

As timing behaviour may be easily changed through OS or application supplied interfaces, using timestamps for passive fingerprinting should always be employed with care and if possible in concert with other methods.

References

[1] W. Richard Stevens. "TCP/IP Illustrated Vol. 1 - The Protocols". Addison Wesley. 1994
[2] Craig Smith, Peter Gundl. "Know Your Enemy: Passive Fingerprinting - Identifying remote hosts, without them knowing". 2002/03/04 http://project.honeynet.org/papers/finger/
[3] Gaël Roualland and Jean-Marc Saffroy. "IP Personality - Home", Website. http://ippersonality.sourceforge.net
[4] Hee So. "SANS Intrusion Detection FAQ: How can passive techniques be used to audit and discover network vulnerability?". http://www.sans.org/resources/idfaq/passive_vuln.php
[5] Robert Braden. RFC1122 "Requirements for Internet Hosts -- Communication Layers". http://www.faqs.org/rfcs/rfc1122.html
[6] Franck Veysset, Olivier Courtay, Olivier Heen. "New Tool And Technique For Remote Operating System Fingerprinting -Full Paper-", Intranode Research Team Articles. April 2002. http://www.intranode.com/en/pdf/techno/ring-full-paper.pdf
[7] Tod Beardsley. "Intrusion Detection and Analysis: Theory, Techniques and Tools" GCIA Practical. 2002. www.giac.org/practical/Tod_Beardsley_GCIA.doc

Part 2 - Detect #1:
RPC portmap request mountd (snort)

Trace

In this part I will discuss the alert shown in figure 2.1.

```
[1:579:2] RPC portmap request mountd [**]
[Classification: Decode of an RPC Query] [Priority: 2]
11/18-04:40:58.696507 153.33.24.3:965 -> 170.129.113.233:111
UDP TTL:113 TOS:0x0 ID:18078 IpLen:20 DgmLen:84 Len: 56
[Xref => http://www.whitehats.com/info/IDS13]
```

Figure 2.1: Snort alert

The tethereal output in the appendix shows a decoded protocol tree and a complete hexdump of the packet, which triggered the alert above.

1. Source Of Trace

I obtained all logs from [1], which are listed to be uploaded on Dec, 2nd 2002. (beginning with "2002.9.9", ending with "2002.10.18"). If not stated otherwise packets discussed below were found in the file named "2002.10.18".

According to the README [1] these packets were captured by a "Snort instance running in binary logging mode". So these files are in PCAP-format (also known as tcpdump-format), which is understood by many network-analysis tools an a wide variety of operation systems. Documentation on libpcap and tcpdump can be found at [2]. Information about the rules were used as trigger to capture the packets have not been given.

With the same methods André Cormier used in his post [5], I was able to confirm that the snort-sensor which recorded the logs originally was most likely hooked on a direct link between 2 Cisco devices. In case of the file 2002.10.18 the internal net is modified to Class B 170.129/16.

2. Detect was generated by

The following software was used to analyze the data and generate alerts from it:

- Snort 2.0.0rc2
  - obtained as source-tarball from http://www.snort.org
  - compiled with support for MySQL
  - configured with all attack-signatures distributed with same source-tarball
  - configured with preprocessors stream4 and stream4_reassemble disabled
  - ACID v0.9.6b21, part of snort-tarball
- Apache, MySQL (taken from original packages from Redhat GNU/Linux 7.3)

To setup these components I used some hints from [4]. All components were installed on a single Host with GNU/Linux-Distribution Redhat 7.3 .

To do an in depth analysis of the events I further used tethereal, which is part of ethereal[7] and is part of Redhat 7.3.

The rule shown in figure 2.2 triggered this alert. (I reformatted the rule for better readability.)

```
alert udp $EXTERNAL_NET any -> $HOME_NET 111
   (msg:"RPC portmap request mountd";
    content:"|01 86 A5 00 00|"; offset:40;depth:8;
    reference:arachnids,13; classtype:rpc-portmap-decode;
    sid:579; rev:2;)
```

Figure 2.2: snort-rule which triggered discussed alert

This rule will trigger on any packet, which is targeted to any host's UDP Port 111 (rpcbind aka. portmap) and asks for the UDP/TCP-Ports for mountd, addressed by RPC procedure number 10005(186a5hex).

3. Probability the source address was spoofed

As UDP is used, the source address could have been spoofed easily. Though I assume this is highly unlikely, as the only possible intend of these packets I can think of is information gathering, the source cannot be disguised because the answer has to find it's way back.

4. Description of attack

The packet is destined to udp-port 111, the well-known port of the portmap-demon (aka. rpcbind). The payload is call to the remote procedure GETPORT of the RPC portmapper, which will tell the UDP and TCP portnumber number on which mountd listens, if this service is running on the target host. The payload show no signs of sever malicious intent and should be regarded as information gathering.

There is no real CVE entry corresponding to this detect, but CVE candidate CAN-1999-0632 is matching fairly.

5. Attack mechanism

The portmap-demon acts as a directory service for remote applications. A client can get information about the UDP- and TCP-ports of a specific RPC-service. Examples for RPC-services are NFS, YP and RWALL.

On Unix hosts RPC-services are listed with the corresponding service-numbers in `/etc/rpc`. Detailed explanation of Sun-RPC can be found in [6] pp. 461ff. as well as in [7].

The service asked for in this detect is mountd. mountd is part of the NFS-protocol-suite which implements sharing filesystems across networks. As NFS and RPC were proposed by Sun Microsystems, it is most likely being used on hosts which are running Unix-like operating systems. There are implementations for non-Unix-Systems too.

As this is a valid RPC getport-request, I consider this "attack" as reconnaisance activity.

6. Correlations

The Source IP belongs to the Class B 153.33/16, which is listed at whois.arin.net to be used by LTX Corp. from San Jose, CA, US. A quick glance showed up they are a tech company. According to their Website[9] they are working in the field testing of integrated circuits.

The IP is not listed to have been logged at incidents.org's Internet Storm Center [11]. In fact there have been only 3 logentrys from 1 IP adress from the hole class B. Also Google [12] didn't reveal anything about the source IP.

This detect was the 1st of 16 "RPC portmap request mountd" alerts on the same date. Figure 2.3 shows a listing of the triggering packets, which is generated by tethereal:

```
# tethereal -r 2002.10.18  udp
 57 9617.400000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b5
 58 9618.220000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b5 dup XI
 59 9619.830000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b5 dup XI
 60 9623.030000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b5 dup XI
 61 9652.410000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b6
 62 9653.220000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b6 dup XI
 63 9654.830000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b6 dup XI
 64 9658.030000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b6 dup XI
 65 9722.070000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b7
 66 9722.890000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b7 dup XI
 67 9724.500000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b7 dup XI
 68 9727.700000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b7 dup XI
 69 9757.070000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b8
 70 9757.890000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b8 dup XI
```

```
 71 9759.490000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b8 dup XI
 72 9762.700000  153.33.24.3 -> 170.129.113.233 Portmap V2 GETPORT Call XID 0x48c805b8 dup XI
```

Figure 2.3: recorded timestamps from a sample host running SuSE 8.1

The timestamps show following pattern:

- initial request with XID (=transaction ID) 0x48c805b5
- pause of ~0.8 seconds
- retry of XID 0x48c805b5
- pause of ~1.6 seconds
- retry of XID 0x48c805b5
- pause of ~3.2 seconds
- retry of XID 0x48c805b5

- pause of ~30 seconds

- new request with XID 0x48c805b6
- 3 retries of 0x48c805b6 (pauses: 0.8, 1.6 and 3.2 seconds)

- pause of ~64 seconds

- new request with XID 0x48c805b7
- 3 retries of 0x48c805b7 (pauses: 0.8, 1.6 and 3.2 seconds)

- pause of ~30 seconds

- new request with XID 0x48c805b8
- 3 retries of 0x48c805b8 (pauses: 0.8, 1.6 and 3.2 seconds)

The pattern of increasing pauses in exponential manner is, is known as exponential backoff algorithm and is most probably the typical behaviour of a normal NFS client, which gets no answer from the asked server.

For comparison I generated the same events from a Linux and a Solaris Client, I could observe the following behaviour:

1. Red Hat 7.3 Linux

   Trying to enumerate NFS-exports with the "showmount", the RPC -request is repeated every 5 seconds. After 12 consecutive requests a new transaction id was chosen.

   Using the command "mount" the client, tried to connect to TCP, which was blocked on my test equipment.

2. Sun Solaris 8 (Sparc)

   "showmount" on solaris only send one RPC-request via udp and timed out, after it got no reply.

   By using "mount", I saw 3 requests with the same transaction ID. The 1st pause was 15 seconds log, the second pause was 30 seconds. After 15 seconds this pattern repeated with a new transaction ID.

So far it seems that neither standard os utilities from Solaris 8 nor Redhat 7.3 were used to approach the victim. Which is not a surprise as the ttl of 113 (see alert at top) suggests, that a the source was a windows host.

Mark E. Donaldson analyzed RPC mountd requests in earlier logfiles. [8] As these requests show a different behaviour (only 1 retry, pcnfsd is queried also) I assume, that the detects, he found, are not related to the ones I analyze here.

7. Evidence of active targeting

As mentioned in the correlation section, I cannot find any evidence, that any other hosts within the internal net or within internet have been probed or attacked from this source. This leads me to the conclusion that the destination may be the only target the user at the source host is interested in.

I can think of two probable reasons for this:

1. Misconfiguration at the source

   Someone wanted to mount NFS and miswrote the IP address of the original intended fileserver and caused the alerts by accident. In this case someone has just called a wrong number.

2. Roaming hardware

   Hardware like laptops and demo equipment is used in one place one day and at a different place the next. If configuration is not tidied up carefully, the information about one network travels with the hardware from one location to the other. In this case, the source machine could have been located in the destination network with legal access to the NFS server. Still being configured to mount data from 170.129.113.233 is linked to the network of LTX, it is now calling home. As the IP of a probably valued server is told in the corresponding packet, a malicious sniffer may get a clue where the pot of gold can be found.

8. Severity

According to the assignment[13] the severity of the detect has to be assessed with the following formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

The values for criticality, lethality, system countermeasures and network countermeasures have to be rated from 1 to 5.

Though I don't know if there is an NFS Server listening at the target IP, I will base my ratings on the defensive assumption that there is one.

criticality: 5

NFS is used to provide access to information. As I don't know what kind of data is stored on the targeted NFS-Server and being defensive, I assume mission critical information. So I give the highest score for criticality.

lethality: 2

If configured out of the box, NFS provides no security on many operating systems because it completely relys on the client for proper authentication of the user. Meaning that administrator of NFS clients are able to impersonate any user which is listed to have access to an NFS share.

There are known vulnerabilities that may lead to root compromise for almost any of the services in the SUN-RPC suite. For rpc.mountd as well as for portmap/rpcbind there are many exploits in the wild (e.g. [10]).

As the recorded packets don't carry any sign's of hostile intent, I presume at least non intruding recon, which I rate with lowest lethality. As this probe seems to be directed solely to the target, suggesting one is specially interested in this one host, I increase this rating by 1.

network countermeasures: 4

As stated above, there seems to be no response from the destination host, I assume that none of the packets reached the intended target. If there is an host, which offers the RPC-portmap service, I presume that packets have been blocked properly. As I cannot be sure, I decrement maximum by 1 point.

system countermeasures: 1

Because nothing of the target system is known - in fact I don't know if target system exists - I have nothing to assess. I chose the worst possible value for system countermeasures until more information about the target is provided.

severity = (5 + 2) - (4 + 1) = 2

With the proposed formula severity is measured on a range from -8 to +8, on which +2 is fairly medium.

9. Defensive recommendation

As network countermeasures seem to be in place no active reaction is required. Because the requests seem specially directed to the target, I would ask the contact listed in whois for more information.

If the theory of "roaming hardware" can be confirmed, I would recommend to establish a policy that any hardware leaving the institution must be cleaned from any configuration files which contain information about my home network. Shouting the IP address of my valued NFS server to the internet is bad habit in my opinion.

10. Multiple choice test question

To get the tcp/udp-port of an remote mountd, the client calls via RPC

a. RPC program number 10000, procedure number 1
b. RPC program number 10005, procedure number 1
c. RPC program number 10000, procedure number 3
d. RPC program number 10005, procedure number 3

Correct answer: c.

As the portmapper is asked, the correct RPC program number is 10000. The GETPORT function is addressed by procedure number 3. (s. [6] p. 465)

Question and Answers

The detect was sent to the mailing list incidents@intrusions.org. I received the following questions regarding my exploit. I received the following questions:

- Andrew Rucker Jones:

  Q: *What about the possibility that it's an attack?*

  A: The packet content is a valid RPC information query and constitutes no attack itself. The exponential backoff suggests a normal client crafted the packets captured and no attack tool is used. However, the RPC query may have preceded a mount attempt, which I would regard as attack. But there is no evidence....

  Q: *Still, the question remains: what is it? Is it an attack? Is it roaming hardware? Is it a wrong number? Make an educated guess.*

  A: Roaming hardware. (only guessed!)

- Anton Chuvakin:

  Q:Hmm, I am curious how would you generate those on Windows? Have you done any more checking (beyond the TTL), such as with p0f that the src is in fact a Win box?

A: No I was interested in, if the packets were part of mount (accessing files) or something like showmount (only getting port information). As I don't have any other NFS clients for testing available here, I didn't bother to further fingerprinting the NFS.

BTW: p0f will only work on TCP syn packets. I analyzed UDP packets in this detect.

Q: *I wonder how it combines with the fact that you say that the likely reason is misconfig/roaming hardware? Should it be closer to 'low' than to 'medium' in this case?*

And, in the end, a more open ended question. What kind of information would clarify the intent of the above packet being sent?

A: As soon as I get information about the destination IP, e.g. if there is an NFS Server and which system countermeasures are implemented, the severity will most likely drop instantly. As said all values are defensive. As Mike Poor said at SANS Amsterdam 2002: "An intrusion analyst always needs more information".

Appendix: Verbose output of tethereal

The following fragment is the part of the output from "tethereal -xVr orig/2002.10.18 ip.src == 153.33.24.3" which shows the packet analyzed in this detect.

```
Frame 57 (98 bytes on wire, 98 bytes captured)
    Arrival Time: Nov 18, 2002 03:40:58.696507000
    Time delta from previous packet: 238.800000000 seconds
    Time relative to first packet: 9617.400000000 seconds
    Frame Number: 57
    Packet Length: 98 bytes
    Capture Length: 98 bytes
Ethernet II, Src: 00:03:e3:d9:26:c0, Dst: 00:00:0c:04:b2:33
    Destination: 00:00:0c:04:b2:33 (Cisco_04:b2:33)
    Source: 00:03:e3:d9:26:c0 (Cisco_d9:26:c0)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 153.33.24.3 (153.33.24.3), Dst Addr: 170.129.113.233 (170.129.113.233)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 84
    Identification: 0x469e
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 113
    Protocol: UDP (0x11)
    Header checksum: 0x356c (correct)
    Source: 153.33.24.3 (153.33.24.3)
    Destination: 170.129.113.233 (170.129.113.233)
User Datagram Protocol, Src Port: 965 (965), Dst Port: sunrpc (111)
    Source port: 965 (965)
    Destination port: sunrpc (111)
    Length: 64
    Checksum: 0xd1ca (correct)
Remote Procedure Call
    XID: 0x48c805b5 (1221068213)
    Message Type: Call (0)
    RPC Version: 2
    Program: Portmap (100000)
    Program Version: 2
    Procedure: GETPORT (3)
    Credentials
        Flavor: AUTH_NULL (0)
        Length: 0
    Verifier
        Flavor: AUTH_NULL (0)
        Length: 0
Portmap
    Program Version: 2
    V2 Procedure: GETPORT (3)
    Program: MOUNT (100005)
    Version: 3
    Proto: UDP (17)
```

```
    Port: 0

0000  00 00 0c 04 b2 33 00 03 e3 d9 26 c0 08 00 45 00   .....3....&...E.
0010  00 54 46 9e 00 00 71 11 35 6c 99 21 18 03 aa 81   .TF...q.5l.!....
0020  71 e9 03 c5 00 6f 00 40 d1 ca 48 c8 05 b5 00 00   q....o.@..H.....
0030  00 00 00 00 00 02 00 01 86 a0 00 00 00 02 00 00   ................
0040  00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 01 86 a5 00 00 00 03 00 00 00 11 00 00   ................
0060  00 00                                             ..
```

Figure 2.3: Verbose tethereal output of the packet discussed

Bibliography

[1] Anonymous. "GIAC Certification Practical Logs", Website. http://www.incidents.org/logs/Raw
[2] Anonymous. "TCPDUMP public repository", Website. http://www.tcpdump.org
[3] Anonymous. "Snort.org", Website. http://www.snort.org
[4] Scott, Steven J. "Snort Enterprise Implementation - Snort, MySQL, Acid and SnortCenter on Redhat 7.3". http://www.superhac.com/docs/snort_enterprise.pdf
[5] André Cormier, "LOGS: GIAC GCIA Version 3.3 Practical Detect(s) (Andre Cormier)", post on intrusions@incidents.org . 2003/01/20 http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00162.html
[6] Stevens, W. Richard. "TCP/IP Illustrated, Vol. 1 - The Protocols", Addisson-Wesley, 1994
[7] Anonymous. "The Ethereal Network Analyzer", Website. http://www.ethereal.com/
[8] Mark E. Donaldson. "LOGS: GIAC GCIA Version 3.3 Practical Detect", post on intrusions@incidents.org . 2003/01/21. http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00209.html
[9] LTX Corporation. "LTX Home", Website. http://ww.ltx.com
[10] Hudin Lucian. "rpc.mountd exploit", post on bugtraq@securityfocus.com . 1998/09/29 http://lists.insecure.org/lists/bugtraq/1998/Sep/0242.html
[11] Anonymous. "Internet Storm Center", Website. http://isc.incidents.org/source_report.html?order=&subnet=153.033
[12] Google. http://www.google.com, Website. http://www.google.com
[13] GIAC, GIAC Certified Intrusion Analyst (GCIA) Practical Assignment http://www.giac.org/GCIA_assignment_print.php

---

Part 2 - Detect #2:
Virus - SnowWhite Trojan Incoming (snort)

Trace:

In this part I will discuss the alert shown in figure 2.4.

```
[**] [1:720:3] Virus - SnowWhite Trojan Incoming [**]
[Classification: Misc activity] [Priority: 3]
10/18-10:02:37.236507 167.206.112.6:110 -> 32.245.166.236:63677
TCP TTL:58 TOS:0x0 ID:47029 IpLen:20 DgmLen:1500
***A**** Seq: 0xACB90F8C  Ack: 0x97556  Win: 0xFAF0  TcpLen: 20
```

Figure 2.4: Snort alert

The tethereal output in the appendix shows a decoded protocol tree and a complete hexdump of the packet, which triggered the alert above.

1. Source Of Trace:

I obtained all logs from [1], which are listed to be uploaded on Dec, 2nd 2002. (beginning with "2002.9.9", ending with "2002.10.18"). If not stated otherwise packets discussed below were found in the file named "2002.10.18".

According to the README [1] these packets were captured by a "Snort instance running in binary logging mode". So these

files are in PCAP-format (also known as tcpdump-format), which is understood by many network-analysis tools an a wide variety of operation systems. Documentation on libpcap and tcpdump can be found at [2]. Information about the rules were used as trigger to capture the packets have not been given.

With the same methods André Cormier used in his post [5], I was able to confirm that the snort-sensor which recorded the logs originally was most likely hooked on a direct link between 2 Cisco devices. In case of the file 2002.9.18 the internal net is modified to Class B 32.245/16.

2. Detect was generated by:

The following software was used to analyze the data and generate alerts from it:

- Snort 2.0.0rc2
  - obtained as source-tarball from http://www.snort.org
  - compiled with support for MySQL
  - configured with all attack-signatures distributed with same source-tarball
  - configured with preprocessors stream4 and stream4_reassemble disabled
  - ACID v0.9.6b21, part of snort-tarball
- Apache, MySQL (taken from original packages from Redhat GNU/Linux 7.3)

To setup these components I used some hints from [4]. All components were installed on a single Host with GNU/Linux-Distribution Redhat 7.3 .

To do an in depth analysis of the events I further used tethereal, which is part of ethereal[7] and is part of Redhat 7.3.

The rule shown in figure 2.5 triggered this alert. (I reformatted the rule for better readability.)

```
alert tcp any 110 -> any any
    (msg:"Virus - SnowWhite Trojan Incoming";
     content:"Suddlently"; sid:720;
     classtype:misc-activity; rev:3;)
```

Figure 2.5: snort rule "Virus - SnowWhite Trojan Incoming"

This rule will trigger, when a client is fetches an email from a POP3 server (110/tcp) which contains the string "Suddlently". "Suddlently" is a typo typical for some incarnations of the so called "Hybris" worm.

3. Probability the source address was spoofed:

It is always very difficult to spoof IP addresses, when TCP is used. Also the IP source address in this attack is an external POP3 server contacted from an inside client. So it is highly unlikely that the source IP in the alert was spoofed.

As [10] states the source email address is always forged.

4. Description of attack

Hybris is a typical mail worm, relying on a user to execute attached malicious executable code.

5. Attack Mechanism

A good description of Hybris can be found at [10]. The following summary is based on the information wich is given there:

A mail containing an entertaining story, which can occur in different languages, is received. The reader is leaded by the story to execute an attachment sent with this mail. This attachment contains malicious code which is only executable on the win32-platform.

Once executed the worm will install a backdoor and looks for updates on a special newsgroup. Hybris will propagate itself via SMTP to email addresses found in the data it captures from the network layer of the host it runs on.

6. Correlations:

As typical for worms and viruses no CVE number is given. According to [10] Hybris is also named: IWorm_Hybris, I-Worm.Hybris, Snow White, SnowWhite or SnoWhite.

I found Hybris aka. SnowWhite covered in two previous GCIA practicals ([11] and [12]) In both practicals the mail server is located on the internal network.

In the alert analyzed in this paper the mail server is external and seems to have no anti virus protection implemented, as Hybris is delivered to one of the internal hosts.

Source Address

The source address 167.206.112.6 resolves via to s1.optonline.net aka. mail.optonline.net. According to [13] OptimumOnline is a broadband internet access product of Cablevision Systems Corporated. Email is an included service and the source host in this detect is the mail server for their customers.

Signature	Date	Source Address	Destination Address
Virus - Possible scr Worm	2002-10-14	167.206.112.6	32.245.166.236
Virus - SnowWhite Trojan Incoming	2002-10-18	167.206.112.6	32.245.166.236
Virus - Possible scr Worm	2002-10-21	167.206.112.6	32.245.166.236
Virus - Possible scr Worm	2002-10-21	167.206.112.6	32.245.166.236
Virus - Possible MyRomeo Worm	2002-10-21	167.206.112.6	32.245.166.236
Virus - Possible MyRomeo Worm	2002-10-21	167.206.112.6	32.245.166.236
Virus - Possible pif Worm	2002-10-23	167.206.112.6	32.245.166.236

Table 2.1: Alerts Regarding IP 167.206.112.6

Looking at my ACID console I found a total count of 7 alerts where this host is listed as source (s. Table 2.1). All these alerts show the same destination IP and are related to malicious content transferred via POP3.

destination host

Looking at my ACID console again I found no evidence that the destination host has developed viral activity. Even if this host had developed such activity, I doubt the the snort sensor, which generated the logs in the first place, would have logged them, because the Snort standard ruleset include no rules to detect viral activity over SMTP.

Instead I found this host occuring as source of a total of 3179 alerts (9 unique) and as destination of another 2877 alerts (18 unique). After eye balling them and looking at a chosen sample of alerts, they can be summed up as false alerts, incoming portscans and questionable user activity like peer to peer file sharing.

As I drill down the outgoing alerts, I recognized a TTL of 124. This leeds me to the conclusion, that the destination host runs a Windows operating system.

7. Evidence of active targeting:

Hybris attacks every email-address it gets hold off. I do not consider this activity as directed.

8. Severity:

According to the assignment[9] the severity of the detect has to be assessed with the following formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

The values for criticality, lethality, system countermeasures and network countermeasures have to be rated from 1 to 5.

Criticality = 5

Apart from seeing normal end user related traffic, I have no information about the criticality of this system. As this host may be used by an adminitrator who has access to core network components or by a person who has access to highly sensitive information, I will use the maximum as fail safe default. If more information becomes available this value needs to be adjusted.

Lethality = 5

Hybris installs a backdoor inside the perimeter and new code to be executed is regularly downloaded via NNTP. So this hosts is a willing slave to anybody who knows howto write "plugins" for Hybris.

Network Countermeasures = 1

As the packet is most likely a part of an established TCP connection I assume that no device has blocked the traffic on its way towards the POP3 client. If the mail client would have been configured to use SSL or just plaintext IMAP, snort had not issued any alert.

System Countermeasures = 2

There is no evidence of outgoing hostile activity from host 32.245.166.236 in all logs available to my ACID console. I tend to assume that at least an anti virus software installed on the target host. As I have doubts and the host's operating system most probably matches the target architecture of hybris, maximum is decremented by 3.

Severity = 5 + 5 - 1 - 2 = 7

With the provided formula severity is rated between -8 and +8. Rated with a severity of +7 the event needs immediate attention.

9. Defensive recommendation:

A trojan may have been installed inside of the organizations network and an immediate reaction is required. I recommend to locate the suspicious host with the IP 32.245.166.236. Once found it should be checked for any signs of malicious infestation. Evaluating logs from companys' firewalls and mailserver for evidence of malicious activity should also be considered, especially when the host cannot be located straightaway.

If the host is infected I would recommend a complete reinstallation from scratch, because it will be hard to tell which plugins have been installed via the NNTP update mechanism. If recovering from backups is considered, these should be also analyzed for viral infections and trojans, because ACID logs shows that this host could have been infected more than once.

Defense in depth

A more general approach should be considered, as a known worm should not reach one of our internal hosts. It is more than likely that 1 out of 100 workstation has no, an old or a disabled antivirus software installed and may pose as a willing stage

for viral activity.

ACID console shows no evidence, that there is a problem with virus protection on internal mail servers, so my recommendations will only reflect issues raised by use of external mail servers.

application level firewalls

All HTTP, POP3 and IMAP traffic directed to external mail and web servers should be screened by an application level firewall.

Drawback: If these protocols are used in concert with SSL an application level firewall may not be able to read the transmitted content or will break the certificate based authentication of SSL.

security software on workstations

Workstations should be equipped with up to date anti-virus software. I assume this is standard in most organizations today. Personal firewall software may prevent that malicous code is executed. Blocking unallowed outgoing network connections, it may also counteract the virus' propagation.

On the other hand a virus may also have methods to deal with personal firewall software.

surf PC's

A surf PC's dedicated to surfing and private email accounts. Surf pc's can be set up with no access to internal resources. As needed an employee may get it's own surf PC or surf PC's can be deployed as shared resources. As surf PC's are deployed direct internet access from "work"stations can be blocked.

If a problem occurs a surf PC can be switched off and reinitalized from an backup without disrupting an employees ability to work. However, the ability to receive and send email directly is a common prerequisite today for many worklplaces.

policy

I recommend to establish a policy which forbids the use of external mail servers from internal hosts. If external servers are known to implement state of the art anti virus countermeasures, these hosts may be put on a whitelist.

If an organization want its employees to be able to receive personal email at work, it may explicitly allow private mail addresses to be forwarded to business email accounts. Forwarded mail will then be send through employers infrastucture and implemented anti virus protection will cover private mail before it reaches the client inside the perimeter.

user education

Even if all possible technical countermeasures are in place, a small possibility remains, that an email with malicous content will not be blocked before it reaches the user. It would be a good idea to provide him with background knowledge and the awareness to not execute unknown content.

If I were responsible for the internal network, I would immediately start to educate the user(s) of the POP3 client. ;-)

10. Multiple choice test question:

Given that a new email virus is on the loose which is not yet known by your antivirus software, what will be your last line of defense?

a. application level firewall
b. heuristic antivirus software installed on mail server
c. personal firewall installed on all workstations

d. educated end user

Correct answer: d

Defensive countermeasures listed in a, b or c are not foolproof. An unrecognized email virus may still reach a workstation. The last one who decides if an attachment gets executed is the end user.

Appendix: Verbose output of tethereal

The following fragment is the part of the output from tethereal which shows the packet analyzed in this detect. The hexdump of the packet has been shortened, because the full ASCII representation of the transmitted email is also included the decoded protocol tree.

Also the 1st part of the recipient address has been replaced by JOHNDOE.

```
Frame 596 (1514 bytes on wire, 1514 bytes captured)
    Arrival Time: Oct 18, 2002 10:02:37.236507000
    Time delta from previous packet: 18.770000000 seconds
    Time relative to first packet: 28526.260000000 seconds
    Frame Number: 596
    Packet Length: 1514 bytes
    Capture Length: 1514 bytes
Ethernet II, Src: 00:03:e3:d9:26:c0, Dst: 00:00:0c:04:b2:33
    Destination: 00:00:0c:04:b2:33 (Cisco_04:b2:33)
    Source: 00:03:e3:d9:26:c0 (Cisco_d9:26:c0)
    Type: IP (0x0800)
Internet Protocol
    Src Addr: 167.206.112.6 (167.206.112.6),
    Dst Addr: 32.245.166.236 (32.245.166.236)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1500
    Identification: 0xb7b5
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 58
    Protocol: TCP (0x06)
    Header checksum: 0xce98 (incorrect, should be 0xe3b0)
    Source: 167.206.112.6 (167.206.112.6)
    Destination: 32.245.166.236 (32.245.166.236)
Transmission Control Protocol, Src Port: pop3 (110),
    Dst Port: 63677 (63677), Seq: 2897809292, Ack: 619862, Len: 1460
    Source port: pop3 (110)
    Destination port: 63677 (63677)
    Sequence number: 2897809292
    Next sequence number: 2897810752
    Acknowledgement number: 619862
    Header length: 20 bytes
    Flags: 0x0010 (ACK)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...1 .... = Acknowledgment: Set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..0. = Syn: Not set
        .... ...0 = Fin: Not set
    Window size: 64240
    Checksum: 0x9479 (incorrect, should be 0xa991)
Post Office Protocol
    Response: +OK
    Response Arg: 31062 octets
    Return-path: <>\r\n
    Received: from mta4.srv.hcvlny.cv.net (mta4.srv.hcvlny.cv.net [167.206.5.10])\r\n
     by mstr2.srv.hcvlny.cv.net\r\n
     (iPlanet Messaging Server 5.2 HotFix 0.8 (built Jul 12 2002))\r\n
     with ESMTP id <0H4500CHXDPGXW@mstr2.srv.hcvlny.cv.net> for\r\n
     JOHNDOE@ims-ms-daemon (ORCPT JOHNDOE@optonline.net); Thu,\r\n
     17 Oct 2002 18:41:40 -0400 (EDT)\r\n
    Received: from u5e7v4 (adsl-63-246-82.mia.bellsouth.net [208.63.246.82])\r\n
     by mta4.srv.hcvlny.cv.net\r\n
     (iPlanet Messaging Server 5.2 HotFix 0.9 (built Jul 29 2002))\r\n
```

```
     with SMTP id <0H45001FBDPHTN@mta4.srv.hcvlny.cv.net> for JOHNDOE@optonline.net\r\n
     (ORCPT JOHNDOE@optonline.net); Thu, 17 Oct 2002 18:41:43 -0400 (EDT)\r\n
    Date: Thu, 17 Oct 2002 18:41:41 -0400 (EDT)\r\n
    Date-warning: Date header was inserted by mta4.srv.hcvlny.cv.net\r\n
    From: Hahaha <hahaha@sexyfun.net>\r\n
    Subject: Snowhite and the Seven Dwarfs - The REAL story!\r\n
    Message-id: <0H45001FEDPHTN@mta4.srv.hcvlny.cv.net>\r\n
    MIME-version: 1.0\r\n
    Content-type: multipart/mixed; boundary="Boundary_(ID_UNG2TmtgBnUz4OFvQ0xFZw)"\r\n
    \r\n
    \r\n
    --Boundary_(ID_UNG2TmtgBnUz4OFvQ0xFZw)\r\n
    Content-type: text/plain; charset=us-ascii\r\n
    Content-transfer-encoding: 7BIT\r\n
    \r\n
    Today, Snowhite was turning 18. The 7 Dwarfs always where very educated and\r\n
    polite with Snowhite. When they go out work at mornign, they promissed a \r\n
    *huge* surprise. Snowhite was anxious. Suddlently, the door open, and the Seven\r\n
    Dwarfs enter...\r\n
    \r\n
    \r\n
    --Boundary_(ID_UNG2TmtgBnUz4OFvQ0xFZw)\r\n
    Conten

0000  00 00 0c 04 b2 33 00 03 e3 d9 26 c0 08 00 45 00   .....3....&...E.
0010  05 dc b7 b5 00 00 3a 06 ce 98 a7 ce 70 06 20 f5   ......:.....p. .
0020  a6 ec 00 6e f8 bd ac b9 0f 8c 00 09 75 56 50 10   ...n........uVP.
0030  fa f0 94 79 00 00 2b 4f 4b 20 33 31 30 36 32 20   ...y..+OK 31062
0040  6f 63 74 65 74 73 0d 0a 52 65 74 75 72 6e 2d 70   octets..Return-p
0050  61 74 68 3a 20 3c 3e 0d 0a 52 65 63 65 69 76 65   ath: <>..Receive
0060  64 3a 20 66 72 6f 6d 20 6d 74 61 34 2e 73 72 76   d: from mta4.srv
0070  2e 68 63 76 6c 6e 79 2e 63 76 2e 6e 65 74 20 28   .hcvlny.cv.net (
0080  6d 74 61 34 2e 73 72 76 2e 68 63 76 6c 6e 79 2e   mta4.srv.hcvlny.
0090  63 76 2e 6e 65 74 20 5b 31 36 37 2e 32 30 36 2e   cv.net [167.206.
00a0  35 2e 31 30 5d 29 0d 0a 20 62 79 20 6d 73 74 72   5.10]).. by mstr
00b0  32 2e 73 72 76 2e 68 63 76 6c 6e 79 2e 63 76 2e   2.srv.hcvlny.cv.
00c0  6e 65 74 0d 0a 20 28 69 50 6c 61 6e 65 74 20 4d   net.. (iPlanet M
00d0  65 73 73 61 67 69 6e 67 20 53 65 72 76 65 72 20   essaging Server
00e0  35 2e 32 20 48 6f 74 46 69 78 20 30 2e 38 20 28   5.2 HotFix 0.8 (
00f0  62 75 69 6c 74 20 4a 75 6c 20 31 32 20 32 30 30   built Jul 12 200
0100  32 29 29 0d 0a 20 77 69 74 68 20 45 53 4d 54 50   2)).. with ESMTP
:
:
05b0  74 65 72 2e 2e 2e 0d 0a 0d 0a 0d 0a 2d 2d 42 6f   ter..........--Bo
05c0  75 6e 64 61 72 79 5f 28 49 44 5f 55 4e 47 32 54   undary_(ID_UNG2T
05d0  6d 74 67 42 6e 55 7a 34 4f 46 76 51 30 78 46 5a   mtgBnUz4OFvQ0xFZ
05e0  77 29 0d 0a 43 6f 6e 74 65 6e                     w)..Conten
```

Figure 2.6: Verbose tethereal output of the packet discussed

Bibliography

[1] Anonymous. "GIAC Certification Practical Logs", Website. http://www.incidents.org/logs/Raw
[2] Anonymous. "TCPDUMP public repository", Website. http://www.tcpdump.org
[3] Anonymous. "Snort.org", Website. http://www.snort.org
[4] Scott, Steven J. "Snort Enterprise Implementation - Snort, MySQL, Acid and SnortCenter on Redhat 7.3". http://www.superhac.com/docs/snort_enterprise.pdf
[5] André Cormier, "LOGS: GIAC GCIA Version 3.3 Practical Detect(s) (Andre Cormier)", post on intrusions@incidents.org . 2003/01/20 http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00162.html
[6] Stevens, W. Richard. "TCP/IP Illustrated, Vol. 1 - The Protocols", Addisson-Wesley, 1994
[7] Anonymous. "The Ethereal Network Analyzer", Website. http://www.ethereal.com/
[8] Google. "Google", Website. http://www.google.com
[9] GIAC, GIAC Certified Intrusion Analyst (GCIA) Practical Assignment http://www.giac.org/GCIA_assignment_print.php
[10] F-Secure Corporation. "F-Secure Computer Virus Information Pages: Hybris" http://www.f-secure.com/v-descs/hybris.shtml
[11] Mike Poor. "Intrusion Detection in Depth", GCIA Practical Assignment. http://www.giac.org/practical/Mike_Poor_GCIA.doc
[12] Scott Baird. "Intrusion Detection In Depth", GCIA Practical Assignment. http://www.giac.org/practical/Scott_Baird_GCIA.doc
[13] CSC Holdings Inc. "High speed internet service and cable modem provider - Optimum Online", Website.http://www.optimumonline.com

Part 2 - Detect #3:
SQL Slammer inside(tcpdump)

Trace:

In this part I will discuss the alert shown in figure 2.7. The destination address has been sanitzed.

```
20:45:14.283282 10.20.15.41.1299 >  x.x.x.x.1434:  udp 376
```

Figure 2.7: Packet logged by tcpdump

1. Source of Trace

On Saturday, January 25th my colleague Cord Beermann sent this trace by email to the security team of my employer, a major german ISP. After reading about a new MS SQL worm being on the loose, he wanted to know if this worm was hitting his GNU/Linux desktop at home and started tcpdump. His desktop was connected via an ISDN-DialUp to a CISCO Router located in a DMZ of my employers network. Incoming traffic from external sources to the DMZ is blocked by a firewall. Access from DMZ to internal sources of my employer is also restricted.

2. Detect was generated by

My colleague used following command to observe worm related traffic:

```
tcpdump -i ippp0 port 1434
```

##This packet was captured shortly after the outcome of Slammer and matches the characteristic size of the worm.

3. Probability the source address was spoofed

As UDP is used as transport the source address may have been spoofed easily. Slammer has no mechanism for spoofing source IP address. So the probability of spoofing is minimal. However, source address is taken from RFC1918 address space. This space is designated for private use and the IP 10.20.15.41 is used more than once worldwide. So the source address cannot be matched directly with a distinct host.

4. Description of attack

Slammer consists of a single UDP packet, which is sent to port 1434/udp. Apart from its propagation code the payload contains a buffer overflow attack which is directed against MS-SQL Server.

Slammer is also called Sapphire, SQLSlammer or W32.Slammer. The buffer overflow used by slammer is referenced by CVE Canditate CAN-2002-00649 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649").

5. Attack Mechanism

According to [10] this worm uses a stack overflow against the SQL Server Resolution Service (SRSS) of Microsoft SQL Server 7.0/2000 and the derived Products Microsoft Data Engine (MSDE) 1.0 and Microsoft Desktop Engine (MSDE) 2000. If successful, this attack leads to privelege elevation, giving the attacker full control of the attacked system.

If a target is vulnerable Slammer starts propagation to random destinations immediately after the buffer overflow. The whole worm consists of only one UDP packet of 376 bytes, copies of the worm will are sent out at wirespeed and all available network bandwidth is consumed. Besides propagation, no harmfull code is executed at the target. Slammer remains in

volatile memory only and will not touch any data on persistent storage. ##

Patches for the used buffer overflow vunerability have been available since August 2002 [8]. As Microsoft regards neither SQL Server nor MSDE as part of the Windows operating system the corresponding patches are not covered by the automatic update mechanism of newer windows variants and must be installed manually. There is also a variety of software products that include MSDE as part of their installation. The most complete list of such software I found is provided by Chip Andrews [9].

6. Correlations

My research revealed no prior detect of Slammer as part of GCIA practical. However, there are various resources on the net, which are covering the spread of Slammer. The a very good coverage is a joint effort of David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford and Nicholas Weaver [11]. Also usefull summaries I found are [13] and [14]. A disassembled version of the Slammer code with annotations was made available by eEye Digital security [15].

To get an idea where this Slammer packet came from, I asked our firewall administrators whether they can find any proof of slammer activity in the corresponding subnet. They reported that the firewall logs prove that there is evident Slammer activity on the DMZ interface. All traffic came from private address space as defined in RfC1918 [6].

To spot the responsible device, I asked the firewall administrators for the source MAC address. The hardware address revealed that the packets came from another Cisco router in the same subnet which terminates a GRE tunnel. This tunnel is used to provide a contractor with limited access to internal resources.

7. Evidence Of Active Targeting

As Slammer chooses its targets randomly, I do not consider its activity targeted in any way.

8. Severity

According to the assignment[9] the severity of the detect has to be assessed with the following formula:

severity = (criticality + lethality) - (system countermeasures + network countermeasures)

The values for criticality, lethality, system countermeasures and network countermeasures have to be rated from 1 to 5.

Criticality = 5

The destination host is an end user system. Normally I wouldn't need to care about and rate criticality with 1. However, purpose of the dialin targeted is to provide administrative access to all sorts of mission critical components for our standby. Among the reachable components is a Voice over IP device, which manages all telephone calls in the company. This device has a MS SQL Server installed by default and is reachable from the destination host.

Lethality = 3

The only damage done by slammer is consuming network bandwith and processor time. No data is corrupted and recovering from a Slammer infection can be done by simple reboot. In my opinion Slammer's impact is similar to effects of a successfull denial of service attack.

Network Countermeasures = 4

As stated the SQL worm has been stopped by one of our firewalls. No Slammer infestations have been reported for any device within the adminstrative domain of our organization. However, Slammer related activity is captured in a security zone, where it is not supposed to be. I reduce the maximum value by 1, because existing network countermeasures work reliable, but don't seem to be foolproof.

System Countermeasures = 5

The system targeted is a GNU/Linux box, which is not vulnerable to MS SQL vulnerabilities.

Severity = 5 + 3 - 4 - 5 = -1

With the provided formula severity is rated between -8 and +8. Rated with a severity of -1, severity of this detect is medium.

However, another colleague using windows with an accidently installed MSDE could have been reachable under the same destination address. In this case system countermeasures would have been rated with 1 and overall severity climb to +3 which puts this event in the category "must follow up".

9. Defensive recommendation

The administrators responsible for the infected hosts were contacted as soon as we got aware of the real destination of Slammer related activity. At the same time an IP access list denying all traffic related to 1434/udp was installed on both Cisco routers in the DMZ.

To get aware of any Slammer related activity on other subnets, I deployed tcpdump as mini IDS on several hosts in our LAN and data centers. But as filters were in place on almost all network devices no Slammer activity was captured any more and so the tcpdumps were canceled after a week.

To get an advanced protection against similar incidents, we developed more restrictive access lists on both routers and a modified firewall policy.

10. Multiple choice test question

Which packets should be dropped at border routers?

a. inbound packet with destination IP 172.31.3.37
b. inbound packets with source IP 127.4.66.2
c. outbound packets with source IP 10.0.12.34
d. all of them

Correct answer: d

All mentioned IP's should not be transmitted over public internet backbones, as they are reserved for special use only. The IPs 172.31.3.37 and 10.0.12.34 belong to private address space listed in RFC1918[6]. The IP 127.4.66.2 is reserved as part of the net 127/8 and should be used as loopback address inside a host only (s. RFC3330[7]).

Bibliography

[1] Anonymous. "GIAC Certification Practical Logs", Website. http://www.incidents.org/logs/Raw
[2] Anonymous. "TCPDUMP public repository", Website. http://www.tcpdump.org
[3] Stevens, W. Richard. "TCP/IP Illustrated, Vol. 1 - The Protocols", Addisson-Wesley, 1994
[4] Google. "Google", Website. http://www.google.com
[5] GIAC. GIAC Certified Intrusion Analyst (GCIA) Practical Assignment http://www.giac.org/GCIA_assignment_print.php
[6] Yakov Rekhter, Robert G Moskowitz, Daniel Karrenberg, Geert Jan de Groot, Eliot Lear. "Address Allocation for Private Internets", Internet RfC 1918. February 1996. ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt [7] IANA. "Special-Use IPv4 Addresses" Internet RfC 3330. September 2002 ftp://ftp.rfc-editor.org/in-notes/rfc3330.txt [8] Microsoft. "Finding and Fixing Slammer Vulnerabilities". February 2003. http://www.microsoft.com/security/slammer.asp
[9] Chip Andrews. "SQLSecurity.com :SQL Server/MSDE-Based Applications". 2003. http://www.sqlsecurity.com/DesktopDefault.aspx?tabindex=10&tabid=13 [10] CERT/CC. "CERT(r) Advisory CA-2003-04 MS-SQL Server Worm". January 2003. http://www.cert.org/advisories/CA-2003-04.html
[11] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, Nicholas Weaver. "The Spread of the

Sapphire/Slammer Worm" http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html [12] CERT/CC. "Vulnerability Note VU#484891:Microsoft SQL Server 2000 contains stack buffer overflow in SQL Server Resolution Service". July 2002 http://www.kb.cert.org/vuls/id/484891 [13] Internet Stormcenter. "Analysis: Port 1434 MS-SQL Worm". January 2003. http://isc.incidents.org/analysis.html?id=180
[14] Robert Graham. "Advisory: SQL Slammer". January 2003. http://www.robertgraham.com/journal/030126-sqlslammer.html
[15] eEye Digital Security. "SAPPHIRE WORM CODE DISASSEMBLED". January 2003. http://www.eeye.com/html/Research/Flash/sapphire.txt

---

Part 3: Analyze this!

Executive Summary

The task of this part of the practical is to analyze log data generated by a snort sensor in an unversity environment over a period of 5 days. The analysis should cover the universities overall information security. Throughout the analysis system breakins are found and the infection of systems with viral code is spotted. At the end of the analysis process recommendations to improve security will be given.

Analyzed Data

Table 3.1 shows a list of the files obtained from http://www.incidents.org/logs used for this analysis. Along with name and size, a cryptographic checksum (SHA1) is given. The checksums are generated with `sha1sum` which is part of Redhat GNU/Linux.

Filename	Size	SHA1
alert.030315.gz	2116498	18893f7862c49630d432fc33d2ffdec1e92f7386
alert.030316.gz	727652	e149418c0b49da0c3a1dac58e02b2e31f846bfd6
alert.030317.gz	1399848	f47d11207579110f751fb8406f8300540ba59020
alert.030318.gz	761529	6687dc6a579901e263a407b80e989cfb18a97f5b
alert.030319.gz	833610	93880de2f3c38c6af814f9aac9586a0eefa9d993
scans.030315.gz	500311	ce0337d360b6569b092c6c98d813617f5fc5a0be
scans.030316.gz	356731	db0229e2827d892958afa628c754f78fc7e1ae6e
scans.030317.gz	206148	73b290c431be450616c459e4e94645d1acbdc3ae
scans.030318.gz	295416	ec4fe7c2149f4f18344b421cf270e56e635fb574
scans.030319.gz	231174	91463c0c0a17d2f998f9eee9dbaae16fd16b89dd
OOS_Report_2003_03_16_10675	849923	92004d0bb8429922ec2a519c880c26621210c170
OOS_Report_2003_03_17_27088	578563	9925efd51f2f7ae532fdc06456047010bf1663f5
OOS_Report_2003_03_18_28243	501763	9da015d743bc7672559f2e499f0a42182086a730
OOS_Report_2003_03_19_8418	670723	659a7b302d5747758611944db15fd7df08175310
OOS_Report_2003_03_20_31998	1126403	15c3b8a4493aded37bca0e2ff1a9e0d6a320a1b3

Table 3.1: Files used for this analysis

Before I start to process any logged data, I always compute cryptographic checksums and print them. Any tampering afterwards can be detected by recomputing these hashes and comparing with the original printout. If a reader wants to validate my conclusions on the raw data, he may also use the hashes to verify if he has obtained the same files.

How the analysis is performed

The steps to carry out this analysis are:

- decide which tools will be used
- normalize data
- get an overview over alerts and list them
- pick alerts, worth of further investigatiion
- do detailed analysis on chosen alerts

All data is analyzed on a host with Intel based architecture and GNU/Linux distribution Redhat 7.3 installed. Having previous knowledge of writing small efficient shell scripts and as none of the tools described in previous analyses matched my needs, I decide to use self written shell scripts.

All three types of files (alerts, scans and out of specification) have a distinct format. In order to be able to use the same tool on all data I use customized scripts to normalize each format. For my analysis I use a simple but efficient script based query tool.

Apart from logging portscan related packets to the scan logfiles, the snort spp_portscan preprocessor logs summaries in the alert file as well. As more detailed information about portscans is available in the scan logs I regard the scan summaries as redundant and drop them in the shell script used for normalizing the alert logs. In the same shell script I also discard any line that seems to be incomplete or damaged.

All scripts used for the following analysis are included in the appendix.

As the alert logs already have too much information to be covered in this analysis, I use the scan and out of specification logs merely for correlation with already suspicious events.

Lists of Alerts

After normalizing the five alert logfiles contain 420829 alerts in total. Instead of giving one list with with all unique alerts, I choose to split them into the four categories inside (0 alerts), outbound (52231 alerts, 12,41%, s. table 3.2) , inbound (195479 alerts, 46,45%, table 3.3) and outside (195479 alerts, 46,45%. s. Table 3.4) activity.

# Alerts	Description
28411	**Tiny Fragments - Possible Hostile Activity**
11703	spp_http_decode IIS Unicode attack detected
3942	High port 65535 tcp - possible Red Worm - traffic
2907	spp_http_decode CGI Null Byte attack detected
2021	TFTP - Internal TCP connection to external tftp server
1470	**Incomplete Packet Fragments Discarded**
827	High port 65535 udp - possible Red Worm - traffic
552	Port 55850 tcp - Possible myserver activity - ref. 010313-1

118	**IRC evil - running XDCC**
110	**IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize**
78	**NIMDA - Attempt to execute cmd from campus host**
71	Possible trojan server activity
9	TFTP - Internal UDP connection to external tftp server
7	TFTP - External TCP connection to internal tftp server
3	Port 55850 udp - Possible myserver activity - ref. 010313-1
2	RFB - Possible WinVNC - 010708-1
52231	**Total (12,41%)**

Table 3.2: List of outbound alerts

If internal hosts can be isolated as source of malicious activity, we are able to spot intrusions or malicous users and insight in ongoing activities is given. Looking at the list of outbound alerts in table 3.2, I consider following alerts as indicator for malicious activity:

- any kind of fragmentation

  Fragmentation may indicate reconaissance and may also be used as a shield to evade intrusion detection of an actual attack. If not with malicious intent, fragmentation can be an indicator for a network problem or a broken device.

- any kind of alert which is substantially related to a real attack

  For example "IIS Unicode attack" is prone to false positives because it is general and indicates a mechanism, which indicates that someone has used a Unicode sequence escape the document root directory of IIS Web server. Apart from worms like Nimda, Unicode is used by many harmless web applications in a way that this alert is triggered.

  On the other hand, "NIMDA - Attempt to cmd from campus" seems to be specially designed to detect NIMDA activity inside the campus network. Also the alert "IDS553/web-iis_IIS ISAPI Overflow ida INTERNAL nosize" is specific to a real attack and known from experience to be very authentic.

- protocols indicating hacker activity

  Since my first incident I have been called to analyze years ago, installing IRC bots is a recurring motive for hacking. I rate any unexpected IRC traffic as one top sign for something evil going on. So "IRC evil - running XDCC" is certainly on the list to be reviewed.

In table 3.2, alerts that fit in one of these categories are bold. I am going to analyze these in depth. Besides from the error prone spp_http_decode processor I do not consider any event which is based on a simple port or IP address range. However I do not throw them away and use them for correlation as needed.

# Alerts	Description
104312	SMB Name Wildcard
38519	Watchlist 000220 IL-ISDNNET-990517
8109	CS WEBSERVER - external web traffic
5068	High port 65535 tcp - possible Red Worm - traffic
2748	SUNRPC highport access!
2331	MY.NET.30.4 activity
2198	Null scan!

2158	TFTP - Internal TCP connection to external tftp server
1037	High port 65535 udp - possible Red Worm - traffic
932	Queso fingerprint
932	IDS552/web-iis_IIS ISAPI Overflow ida nosize
816	Watchlist 000222 NET-NCFC
757	External RPC call
669	spp_http_decode IIS Unicode attack detected
409	MY.NET.30.3 activity
312	CS WEBSERVER - external ftp traffic
267	Incomplete Packet Fragments Discarded
224	FTP DoS ftpd globbing
217	Port 55850 tcp - Possible myserver activity - ref. 010313-1
214	SNMP public access
197	EXPLOIT x86 NOOP
124	Possible trojan server activity
111	NMAP TCP ping!
105	Tiny Fragments - Possible Hostile Activity
93	EXPLOIT x86 setuid 0
44	EXPLOIT x86 setgid 0
41	EXPLOIT x86 stealth noop
39	spp_http_decode CGI Null Byte attack detected
32	Back Orifice
22	DDOS shaft client to handler
13	Probable NMAP fingerprint attempt
13	Notify Brian B. 3.56 tcp
11	TFTP - Internal UDP connection to external tftp server
10	SMB C access
10	Notify Brian B. 3.54 tcp
9	FTP passwd attempt
8	TFTP - External TCP connection to internal tftp server
2	RFB - Possible WinVNC - 010708-1
2	EXPLOIT NTPDX buffer overflow
1	connect to 515 from outside
1	TCP SMTP Source Port traffic
1	SYN-FIN scan!
1	NETBIOS NT NULL session
173119	**Total (41,13%)**

Table 3.3: List of inbound alerts

Table 3.3 shows a summary of all unique alerts regarding inbound traffic.

Brief description of inbound alerts

I will group these alerts and cover them shortly

- SMB Name Wildcard

  Reconaissance activity regarding SMB protocol used by windows family, may precede attack or infection with network.vbs worm. (s. [5]) - Due to worm activity this event is very noisy.

- Watchlist 000220 IL-ISDNNET-990517
- Watchlist 000222 NET-NCFC
- MY.NET.30.3 activity
- MY.NET.30.4 activity
- CS WEBSERVER - external web traffic
- CS WEBSERVER - external ftp traffic
- Notify Brian B. 3.54 tcp
- Notify Brian B. 3.56 tcp

  Alerts used on campus to monitor special hosts or nets. Events are not related to special attack and very noisy.

- spp_http_decode IIS Unicode attack detected
- spp_http_decode CGI Null Byte attack detected

  Alerts issued by Snort spp_http_decode plugin indicating possible malicious access to webserves. Very noisy because mostly triggered by legitimate web requests (s. [5],[6])

- High port 65535 tcp - possible Red Worm - traffic
- High port 65535 udp - possible Red Worm - traffic
- Port 55850 tcp - Possible myserver activity - ref. 010313-1
- Port 55850 udp - Possible myserver activity - ref. 010313-1
- SUNRPC highport access!
- RFB - Possible WinVNC - 010708-1
- Possible trojan server activity
- Back Orifice
- DDOS shaft client to handler

  Based on looking at the alerts: All these rules trigger if specific ephemeral ports (UDP and/or TCP) are used. Events are not related to a special attack. Rules are very noisy, because rules will often trigger on P2P filesharing protocols.

- External RPC call
- TFTP - Internal TCP connection to external tftp server
- TFTP - Internal UDP connection to external tftp server
- TFTP - Internal UDP connection to external tftp server
- TFTP - Internal UDP connection to external tftp server
- connect to 515 from outside
- TCP SMTP Source Port traffic

  Based on looking at the alerts: All these rules trigger if specific low ports are used. Events are not related to a special attack are noisy, because they are triggered by scan activity.

- Null scan!
- Queso fingerprint
- NMAP TCP ping!
- SYN-FIN scan
- Probable NMAP fingerprint attempt

  Scans/Reconnaissance, may precede attack

- EXPLOIT x86 setuid 0
- EXPLOIT x86 getuid 0
- EXPLOIT x86 NOOP
- EXPLOIT x86 stealth noop

Events triggered by code used in exploits, from my experience prone to false positives, because is likely to be triggered on multi-media data like pictures, movies or sound which embeds same pattern.

- FTP DoS globbing

  May be related wu-ftpd 2.6.1 (CVE-2001-0550). If successfull this may result in system compromise.

- SNMP public access

  May indicate severe information leak on network devices and servers. Configuration data may be publicly accessible.

# Alerts	Description
195479	**TCP SRC and DST outside network**
195479	**Total (46,45%)**

Table 3.4: List of outside alerts

Table 3.4 shows 195479 events, indicating traffic from external sources to external destination. As these events should not occur and sum up to 46.45% of overall alerts I start detailed analysis straight away with this alerts.

Detailed analysis of specific alerts

TCP SRC and DST outside network (195479 Alerts)

The analyzed files list 192541 distinct sources and 111 distinct destinations for this of event. Generating a list of top destinations I see that 194996 (99,75%) of these alerts are related with only 5 destinations. To see if these events are equaly distributed or show any peaks, I evaluate only the timestamps of these entries. The result of this evaluation is table 3.5

# packets	date	start	end	Source IP range	Destination IP:Port
39609	Mar 15	14:45:37	14:47:37	222.115.0.6 - 222.134.47.112	208.253.114.222:80
43423	Mar 15	15:05:38	15:07:37	222.115.0.174 - 222.134.47.111	208.253.114.222:80
33095	Mar 15	16:36:22	16:37:22	39.254.124.38 - 40.11.122.5	131.118.254.39:80
29997	Mar 17	01:22:15	01:23:33	120.90.232.56 -120.106.240.121	208.225.90.120:80
22288	Mar 17	14:20:30	14:22:30	222.115.2.115 - 222.140.253.239	208.253.114.222:80
18530	Mar 17	16:00:01	16:01:14	75.95.1.98 - 75.105.245.161	65.116.88.75:6667
8054	Mar 19	08:10:25	08:10:45	146.196.253.10 - 146.201.83.29	64.251.196.146:80

Table 3.5: alerts triggered by TCP source address spoofing

This leads me to the following questions:

- Do these packets come from an external or an internal source?

  The provider of the university must not have configured a route for any of these destinations towards campus network. If coming from outside, the last router on behalf of the universities' provider would have to route this packet in the opposite direction. My conclusion is that these packets are generated inside of our network.

- Are these attacks stimulus or response?

  Stimulus. Each packet is certainly the result of a packet crafting tool and I am not aware of any normal situation where such a packet would be the response to a stimulus.

- Is the same tool used for each of this attacks?

  Certainly the same packet crafting tool was used, as the same behaviour is shown in all attacks: Table 3.6 shows the beginning of the first attack against 208.253.114.222.

time	source	destination
14:45:37.446150	222.115.0.6:1020	208.253.114.222:80
14:45:37.446178	222.115.0.7:1208	208.253.114.222:80
14:45:37.446191	222.115.0.8:1358	208.253.114.222:80
14:45:37.446273	222.115.0.9:1114	208.253.114.222:80
14:45:37.446406	222.115.0.10:1322	208.253.114.222:80
14:45:37.448833	222.115.0.44:1626	208.253.114.222:80
14:45:37.455447	222.115.0.62:1366	208.253.114.222:80
14:45:37.456825	222.115.0.81:1364	208.253.114.222:80

Table 3.6: Beginning of IP spoofing attack

  All traces begin with a source IP address related to the reverse destination IP address plus an offset and all source ports seem to be randomly between 1000 to 2000. (208.253.**114.222** -> **222.114**.253.208)

- Possible motives of the attacker

  - Denial of Service by SYN-Flooding

    The number of half open TCP connections a host can handle is limited. Each packet lead to a half open TCP connection on the target host. After the target host reaches the limit of (half) open connections he can handle, further legitimate connection requests can no more be handled.

    TCP SYN Flooding and IP Spoofing attacks are covered by a CERT Advisory [1]

  - IDS evasion

    It is also possible that these attacks are carried out to hide an more malicious paket from IDS detection. I will not try to directly find evidence for this possibility, because I consider it to be somewhat unlikely.

The sources of the remaining outside events show sources from RFC1918 address space (275 events), 0.0.0.0 (171) and public address space (27 events).

Conclusion: There seems to be a lack in defenses against IP spoofing.

Tiny Fragments - Possible Hostile Activity (28411 Outbound Alerts)

The first thing I noticed about these alerts was, that all 28411 outbound alerts came from the same source IP

MY.NET.239.202. To get an idea I printed a list of all outbount events regarding this host.

# events	description
28411	Tiny Fragments - Possible Hostile Activity
4361	Scan - NULL
3120	Scan - NOACK
803	Scan - INVALIDACK
423	Scan - VECNA
334	Scan - UNKNOWN
194	Scan - XMAS
98	Scan - NMAPID
86	Scan - SYN
75	Scan - FULLXMAS
51	Scan - UDP
47	Scan - FIN
46	Scan - SYNFIN
36	Scan - SPAU
1	High port 65535 tcp - possible Red Worm - traffic

Table 3.7: Outbound events (alerts, scan & out of specification) regarding MY.NET.239.202

The 1st event was observed at March, 15th at 0:01, right at the beginning of the 1st analyzed log data, the last on occured on March 19th at 16:06. The last host scanned was 81.51.86.125. This first event for this destination host was at 15:45.

date/time	source	destination	description
03/15 08:22:59	65.33.11.251:1555	MY.NET.239.202:137	SMB Name Wildcard
03/15 08:36:51	211.93.36.102:2838	MY.NET.239.202:80	IDS552/web-iis_IIS ISAPI ...
03/15 10:37:05	213.249.2.212:137	MY.NET.239.202:137	SMB Name Wildcard
03/15 13:17:38	207.69.92.112:137	MY.NET.239.202:137	SMB Name Wildcard
03/15 13:21:24	192.193.195.178:0	MY.NET.239.202:0	Scan - NOACK
03/16 01:15:42	64.175.110.159:137	MY.NET.239.202:137	SMB Name Wildcard
03/16 08:30:12	203.232.225.78:3031	MY.NET.239.202:445	Scan - SYN
03/16 08:48:12	128.211.221.175:3036	MY.NET.239.202:137	SMB Name Wildcard
03/16 10:13:50	212.68.236.167:3737	MY.NET.239.202:3814	TCP Out Of Specification
03/16 13:19:08	66.125.152.227:137	MY.NET.239.202:137	SMB Name Wildcard
03/16 13:20:28	218.31.234.9:1026	MY.NET.239.202:137	SMB Name Wildcard

03/16 15:01:11	128.61.33.97:1512	MY.NET.239.202:137	SMB Name Wildcard
03/16 17:15:40	68.113.65.41:1032	MY.NET.239.202:137	SMB Name Wildcard
03/17 23:16:23	148.235.130.62:19225	MY.NET.239.202:137	SMB Name Wildcard
03/18 05:19:11	35.11.244.136:1488	MY.NET.239.202:137	SMB Name Wildcard
03/18 21:20:15	211.24.87.225:1024	MY.NET.239.202:137	SMB Name Wildcard
03/19 15:37:21	80.15.121.137:137	MY.NET.239.202:137	SMB Name Wildcard
03/19 15:47:24	81.51.86.125:137	MY.NET.239.202:137	SMB Name Wildcard
03/19 15:58:07	81.51.86.125:137	MY.NET.239.202:137	SMB Name Wildcard

Table 3.8: Inbound events (alerts, scan & out of specification) regarding MY.NET.239.202

My next step is to match inbound events with this incident. Table 3.8 shows all inbound events regarding MY.NET.239.202. All entries look like normal noise. -- All but the last two entries! The source IP corresponds to the last destination of the scan alerts.

The following plot seems reasonable to me: someone feeling responsible for the host 81.51.86.125 noticed that there was a scan from MY.NET.239.202. As the scan didn't stop after 20 minutes he may have hacked the host MY.NET.239.202 and terminated the scan.

Even if terminating this scan is in the interest of the university, it remains hacking and so full registration info for 81.51.86.125 (obtained via whois) is given here:

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

inetnum:      81.51.86.0 - 81.51.86.255
netname:      IP2000-ADSL-BAS
descr:        BSAUB105 Aubervilliers Bloc1
country:      FR
admin-c:      WITR1-RIPE
tech-c:       WITR1-RIPE
status:       ASSIGNED PA
remarks:      for hacking, spamming or security problems send mail to
remarks:      postmaster@wanadoo.fr AND abuse@wanadoo.fr
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.com 20030120
changed:      gestionip.ft@francetelecom.com 20030318
source:       RIPE

route:        81.51.0.0/16
descr:        France Telecom
descr:        Wanadoo Interactive
origin:       AS3215
remarks:      ------------------------------------------
remarks:      For Hacking, Spamming or Security problems
remarks:      send mail ONLY to      abuse@wanadoo.fr
remarks:      ------------------------------------------
notify:       addr-reg@rain.fr
mnt-by:       RAIN-TRANSPAC
changed:      karim@rain.fr 20021126
source:       RIPE

role:         Wanadoo Interactive Technical Role
```

```
address:      WANADOO INTERACTIVE
address:      48 rue Camille Desmoulins
address:      92791 ISSY LES MOULINEAUX CEDEX 9
address:      FR
phone:        +33 1 58 88 50 00
e-mail:       abuse@wanadoo.fr
e-mail:       technical.contact@wanadoo.com
admin-c:      WITR1-RIPE
tech-c:       WITR1-RIPE
nic-hdl:      WITR1-RIPE
mnt-by:       FT-BRX
changed:      gestionip.ft@francetelecom.com 20010504
changed:      gestionip.ft@francetelecom.com 20010912
changed:      gestionip.ft@francetelecom.com 20011204
changed:      gestionip.ft@francetelecom.com 20030428
source:       RIPE
```

Conclusion: The host MY.NET.239.202 is source of agressive outbound portscanning and should be inspected immediately.

Just for the records: The alert regarding Red Worm occures in the middle of a scan of the corresponding destination host and I don't believe it constitutes any evidence of worm activity.

Incomplete Packet fragments discarded(1470 Outbound Alerts)

These packets seem to indicate a problem with fragmentation of IP datagrams. Table 3.9 shows a summary of alerts regarding this rule.

# Alerts	source	destinations
895	MY.NET.194.125	195.92.255.169
555	MY.NET.194.125	209.126.191.143
14	MY.NET.212.158	66.36.97.144
2	MY.NET.203.10	81.27.36.54
2	MY.NET.203.10	207.44.203.188
1	MY.NET.203.10	161.184.138.246
1	MY.NET.60.11	195.94.94.111

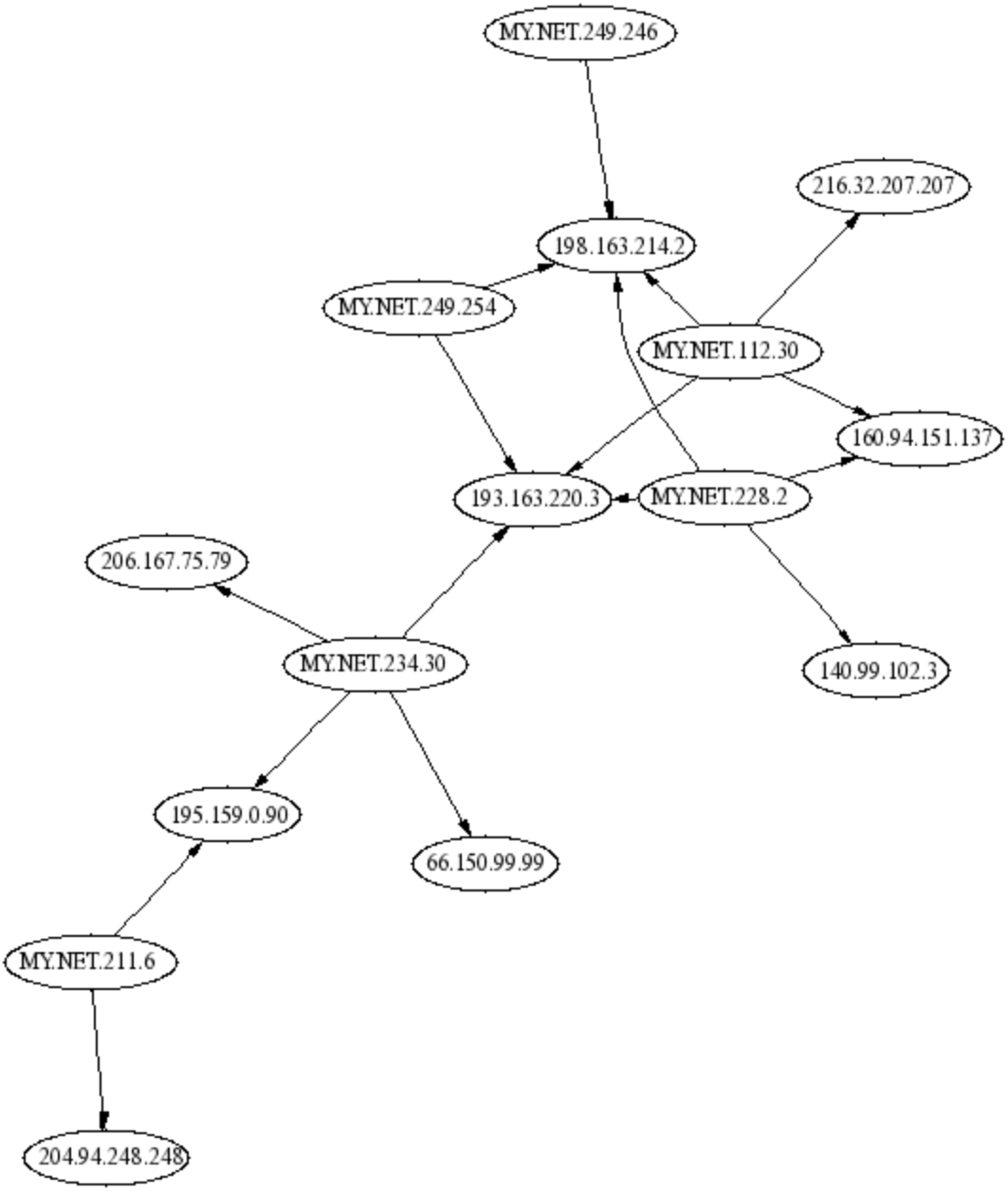
Table 3.9: Summary of "Incomplete Packet fragments discarded"

As no other harmfull event can be correlated, I assume that there is some problem with MY.NET.194.125 and recommend to investigate this host.

IRC evil - running XDCC(118 Outbound Alerts)

According to the generated alerts, I assume this rule triggers on any traffic which is destined to port 6666-7000. A possible predecessor of this rule can be found in a posting from Christopher E Cramer [3].

These alerts cannot be correlated to other harmful events in all analyzed logfiles. As I found two interesting sets of links between source and destination IPs, I choose to generate link graphs (s. Figure 3.1) for this event type.


MY.NET.249.246
216.32.207.207
198.163.214.2
MY.NET.249.254
MY.NET.112.30
160.94.151.137
193.163.220.3
MY.NET.228.2
206.167.75.79
MY.NET.234.30
140.99.102.3
195.159.0.90
66.150.99.99
MY.NET.211.6
204.94.248.248

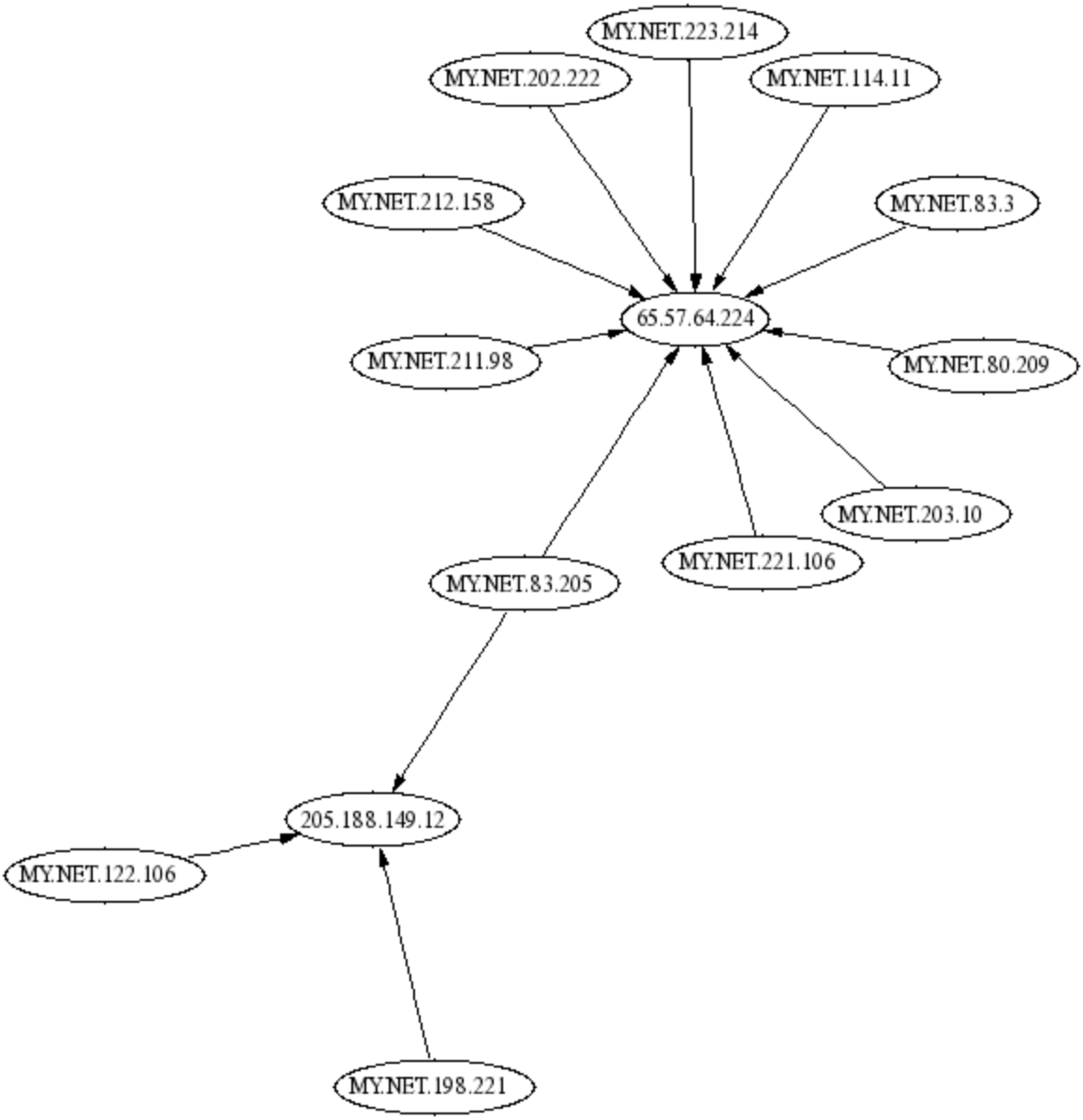


Figure 3.1: Link graphs regarding event "IRC evil - running XDCC"

The external hosts with most events regarding this alert is 66.57.64.224. The whois info regarding this host is given below:

```
OrgName:    Road Runner
OrgID:      RRMA
Address:    13241 Woodland Park Road
City:       Herndon
StateProv:  VA
PostalCode: 20171
Country:    US

NetRange:   66.56.96.0 - 66.57.255.255
CIDR:       66.56.96.0/19, 66.56.128.0/17, 66.57.0.0/16
NetName:    ROADRUNNER-2-MIDSOUTH
NetHandle:  NET-66-56-96-0-1
Parent:     NET-66-0-0-0-0
NetType:    Direct Allocation
NameServer: DNS1.RR.COM
NameServer: DNS2.RR.COM
```

```
NameServer: DNS3.RR.COM
NameServer: DNS4.RR.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-12-18
Updated:    2002-08-14

TechHandle: ZS30-ARIN
TechName:   ServiceCo LLC
TechPhone:  +1-703-345-3416
TechEmail:  abuse@rr.com

OrgAbuseHandle: ABUSE10-ARIN
OrgAbuseName:   Abuse
OrgAbusePhone:  +1-703-345-3416
OrgAbuseEmail:  abuse@rr.com

OrgTechHandle: IPTEC-ARIN
OrgTechName:   IP Tech
OrgTechPhone:  +1-703-345-3416
OrgTechEmail:  abuse@rr.com

# ARIN WHOIS database, last updated 2003-05-04 20:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
NIMDA - Attempt to execute cmd from campus host

As both kind of alerts are related I discuss them in one section. NIMDA is a worm, which utilizes multiple propagation vectors. A full analysis is given in [5]. The discussed alerts trigger on worm attacks via HTTP port 80/tcp.

# alerts	source	description
82	MY.NET.97.222	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
62	MY.NET.97.222	NIMDA - Attempt to execute cmd from campus host
21	MY.NET.97.72	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
12	MY.NET.97.72	NIMDA - Attempt to execute cmd from campus host
7	MY.NET.97.43	IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize
3	MY.NET.97.43	NIMDA - Attempt to execute cmd from campus host
1	MY.NET.195.157	NIMDA - Attempt to execute cmd from campus host

Table 3.10: sources for nimda related events.

Analysis for each host:

MY.NET.97.72

The first NIMDA related alert was seen on March 16 at 03:23:12 and logged as "TFTP - Internal UDP connection to external tftp server" and is indicating that admin.dll is downloaded from 130.130.26.110. As it seems appropriate to inform someone about the rampant NIMDA, the full whois record to this address is given:

```
OrgName:    University of Wollongong
OrgID:      UNIVER-14
Address:    Illawarra Technology Corporation Ltd.
Address:     P.O. Box 1144
Address:     Wollongong
```

```
Address:     N.S.W., 2500
City:
StateProv:
PostalCode:
Country:    AU

NetRange:   130.130.0.0 - 130.130.255.255
CIDR:       130.130.0.0/16
NetName:    UOWNET
NetHandle:  NET-130-130-0-0-1
Parent:     NET-130-0-0-0-0
NetType:    Direct Assignment
NameServer: DNS.UOW.EDU.AU
NameServer: DNS1.UOW.EDU.AU
Comment:
RegDate:    1988-08-11
Updated:    2001-05-01

TechHandle: ZI57-ARIN
TechName:   Information Technology ServicesUniversity of Wollo
TechPhone:  +61 2 4221 3775
TechEmail:  dns-admin@uow.edu.au

# ARIN WHOIS database, last updated 2003-05-04 20:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

First sign of own worm activity in the alert logs can be seen at 03:23:12. The last entry regarding its activity is seen at 03/16-03:52:07. Beginning at 03:51:47 a portscan to various address with destination port 80 can be seen. The portscan stops at 3:55:18. These scans may be the result of non responding hosts attacked by Nimda or sign of a system compromise.

On March 18th at 00:20:30 a new portscan with target 137 UDP starts. This portscan ends at 3:03:53. The sytem MY.NET.97.72 should be regarded as compromised.

A summary of all outbound alerts regarding this IP is given in table 3.11.

# alerts	Dst. Port	description
743	137	Scan - UDP
453	80	Scan - SYN
21	80	IDS552/web-iis_IIS ISAPI Overflow ida ...
12	80	NIMDA - Attempt to execute cmd from campus ...
3	139	Scan - SYN
1	69	TFTP - Internal UDP connection to external tftp server

Table 3.11: Summary of outbound alerts regarding MY.NET.97.72

MY.NET.97.43

On March 16th at 22:38:12 an inbound alert "IDS552/web-iis_IIS ISAPI Overflow ida >...<" regarding MY.NET.97.43 occurs. Source IP of this event is 218.93.14.90 and the whois data is given:

```
% [whois.apnic.net node-1]
% How to use this server        http://www.apnic.net/db/
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:      218.90.0.0 - 218.94.255.255
netname:      CHINANET-JS
```

```
descr:          CHINANET jiangsu province network
descr:          China Telecom
descr:          A12,Xin-Jie-Kou-Wai Street
descr:          Beijing 100088
country:        CN
admin-c:        CH93-AP
tech-c:         CJ186-AP
mnt-by:         MAINT-CHINANET
mnt-lower:      MAINT-CHINANET-JS
mnt-routes:     maint-chinanet-js
changed:        hostmaster@ns.chinanet.cn.net 20020209
changed:        hostmaster@ns.chinanet.cn.net 20030306
status:         ALLOCATED non-PORTABLE
source:         APNIC

route:          218.93.0.0/16
descr:          CHINANET jiangsu province network
country:        CN
origin:         AS23650
mnt-by:         MAINT-CHINANET-JS
changed:        ip@jsinfo.net 20030414
source:         APNIC

role:           CHINANET JIANGSU
address:        No.268,Hanzhong Road,Nanjing 210029
country:        CN
phone:          +86-25-6588783
fax-no:         +86-25-6588740
e-mail:         ip@jsinfo.net
trouble:        send anti-spam reports to spam@jsinfo.net
trouble:        send abuse reports to abuse@jsinfo.net
trouble:        times in GMT+8
admin-c:        CH360-AP
tech-c:         CS306-AP
tech-c:         CN142-AP
nic-hdl:        CJ186-AP
remarks:        www.jsinfo.net
notify:         ip@jsinfo.net
mnt-by:         MAINT-CHINANET-JS
changed:        dns@ptt.js.cn 20020530
changed:        ip@jsinfo.net 20021213
source:         APNIC

person:         Chinanet Hostmaster
address:        No.31 ,jingrong street,beijing
address:        100032
country:        CN
phone:          +86-10-66027112
fax-no:         +86-10-66027334
e-mail:         hostmaster@ns.chinanet.cn.net
e-mail:         anti-spam@ns.chinanet.cn.net
nic-hdl:        CH93-AP
mnt-by:         MAINT-CHINANET
changed:        hostmaster@ns.chinanet.cn.net 20021016
source:         APNIC
```

12 seconds later MY.NET.97.43 connected to only 10 external hosts showing the typical NIMDA related alerts. After a pause of 7 minutes a portscan destined to various hosts port 80/tcp starts.

Beginning on march 18th at 03:03:27 and ending at 4:24:27 a series of "CGI Null Byte attack detected" is detected. After looking up corresponding DNS addresses, these alerts look to me like someone is looking at auctions at eBay.

Beginning on March 19th at 08:45:50 and ending at 8:51:17 a series of "IIS Unicode attack detected" is logged. Target of

these events are the IPs 211.233.28.47, 211.233.29.59 and 211.233.28.247. Looking at the whois info:

```
Query: 211.233.29.59

# 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         : 211.233.28.0-211.233.31.255
Network Name       : KIDC-INFRA-SERVERROOM-DAUM
Connect ISP Name   : KIDC
Connect Date       : 20001213
Registration Date  : 20011115

[ Organization Information ]
Orgnization ID     : ORG231919
Org Name           : Daum Communication
State              : SEOUL
Address            : Gangnam-gu, Yeoksam-dong, DACOM B/D 12F. 706-1
Zip Code           : 135-987

[ Admin Contact Information]
Name               : Hanju Kim
Org Name           : Daum Communication
State              : SEOUL
Address            : Gangnam-gu, Yeoksam-dong, DACOM B/D 12F. 706-1
Zip Code           : 135-987
Phone              : +82-2-6446-6407
Fax                : +82-2-6446-6499
E-Mail             : hankim@daumcorp.com

[ Technical Contact Information ]
Name               : youngchul Lee
Org Name           : Daum Communication
State              : SEOUL
Address            : Gangnam-gu, Yeoksam-dong, DACOM B/D 12F. 706-1
Zip Code           : 135-987
Phone              : +82-2-6446-6407
Fax                : +82-2-6446-6499
E-Mail             : uniace@daumcorp.com

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

If the above contacts are not rechable, please see the following ISP contacts
for relevant information or network abuse complaints.

[ ISP IP Admin Contact Information ]
Name               : Jang Sang Gyu
Phone              : +82-2-6440-2920
Fax                : +82-2-6440-2909
E-Mail             : support@kidc.net

[ ISP IP Tech Contact Information ]
Name               : Lee Yun mi
Phone              : +82-2-6440-2925
Fax                : +82-2-6440-2909
E-Mail             : ip@kidc.net

[ ISP Network Abuse Contact Information ]
Name               : Lee Min Sik
Phone              : +82-2-6440-2936
Fax                : +82-2-6440-2909
E-Mail             : security@kidc.net

# KOREAN
```

```
> skipped <
```

As these addresses belong to a korean server, I assume that Unicode is heavily utilized to transfer the korean language and these series of events do not represent an actual attack.

MY.NET.97.222

First sign of worm activity in the alert logs can be seen at March 18th, 00:00:02. The last entry regarding this host about nine minutes later. All events regarding this host indicate NIMDA or CodeRed activity. It seems that someone noticed the worm and stopped it.

Overall Top Talkers

The following list represents the top talkers regarding to outgoing requests along with a brief description:

1. 194996 Events: Anonymous

   The host which caused most overall events in terms of outbound alerts, scans and out of specfication logs remains anomymous, due to the fact that the source address was spoofed. See detailed discussed detect regarding "TCP SRC & DST outside Network".

2. 38086 Events: MY.NET.239.202

   covered in detailed discussed detect regarding "Tiny Fragments - Possible Hostile Activity"

3. 9931 Events: MY.NET.70.176

   source of 9512 scan events regarding port 6257/udp
   6257/UDP is normally used by WinMX Filesharing application
   -> P2P filesharing

4. 8484 MY.NET.196.179

   source of 5500 scan events regarding port 22321/udp
   source of 2930 scan events regarding port 7674/udp Khan Rohail has reported the use of this port combination in February 2003 [8]. I find it likely to be a new filesharing tool.
   -> unknown intent/possible P2P filesharing

5. 7802 MY.NET.88.134

   source of 4437 scan events regarding port 7674/udp
   source of 3160 scan events regarding port 22321/udp
   -> unknown intent/possible P2P filesharing (see above)

6. 7417 MY.NET.1.3

   source of 7416 scan events regarding port 53
   -> Nameserver, s. practical of Hee So [7]

7. 5393 MY.NET.88.180

   source of 4113 scan events regarding port 22321/udp
   source of 1234 scan events regarding port 7674/udp
   -> unknown intent/possible P2P filesharing (see above)

8. 4312 MY.NET.88.193

   source of 1576 scan events regarding port 135/tcp
   source of 376 scan events regarding port 445/tcp
   source of 289 scan events regarding port 137/udp
   source of 258 scan events regarding port 139/tcp
   source of 98 scan events regarding port 524/tcp
   -> **malicious activity/possible system compromise**

9. 3413 MY.NET.168.82

   source of 1974 scan events regarding port 22321/udp
   source of 1422 scan events regarding port 7674/udp
   -> unknown intent/possible P2P filesharing (see above)

10. 3364 MY.NET.97.21

    source of 3261 scan events regarding port 137/udp
    -> **malicious activity/possible system compromise**

Defensive Recommendations

Based on my analysis I recommend:

- Deal with compromised / infected hosts

  The following internal IPs have been identified as source of malicious activity and are threatening the security of the campus network as well as any host in the internet:
  - MY.NET.239.202(System compromised / Intense scanning)
  - MY.NET.97.72 (System compromised / NIMDA)
  - MY.NET.97.43 (System compromised / NIMDA)

  I recommend to reinstall these hosts from scratch and apply all available security patches. As an additional precaution these hosts should be equipped with host based filtering of network traffic. As these hosts have proven to be vulnerable the actual compromise there may have been earlier malicious activities. Therefore I would not recommend to restore these systems from backups.

- Perform a thorough virus scan of MY.NET.97.0/24 and hosts related to this net.

  Nimda is known to propagate via network shares. All network shares available to this net should be checked for worm related content.

- Deal with suspicious hosts

  The following host showed an abnormal behaviour and should be looked at:
  - MY.NET.194.125 "Incomplete Packet Fragments Discarded"
  - MY.NET.97.222 (possible NIMDA)
  - MY.NET.195.157 (possible NIMDA)
  - MY.NET.88.193 (source of scans)
  - MY.NET.97.21 (source of scans)

- Implement more restrictive firewall policy and router ACLs

  About 46% of all alerts analyzed are the result of IP spoofing, probably used to disable someone elses communication. If not already in place, a minimal ingress and egress filtering as described in [2] should be implemented immediately to prevent that the university gets sued for mounting denial of service attacks. [2] is

specific to CISCO but may give an overview what should be possible with your hardware.

Furthermore I recommend to employ more restrive ingress and egress filtering. At least a restrictive filtering policy regarding the ports should be implemented:

- 25/tcp (smtp)
- 69/udp&tcp (TFTP)
- 135-139,445/udp&tcp (Windows SMB)
- 80,443/tcp (HTTP,HTTPS)
- 1433-1434/udp&tcp (MS SQL)

These ports are used as propagation vectors by actual worms. Closing these ports in both directions with exceptions needed would result in a dramatically improved protection for the campus network. Besides the probability that universities hosts spread worms to the internet (including hosts I am responsible for) is also minimized. External access to internal web- & mailservers should only be allowed if absolutely necessary.

- Forbid filesharing

  As the main use of filesharing is distributing illegal content, P2P protocols should be banned generally from the network of the campus. If the ban is in place, implement some means of control. This may carried out as part of the IDS ruleset. Doing so will not only result in more security for your equipment, but may also help to protect your students and employees from getting into disagreeable disputes with the lawyers of MPAA and RIAA.

- Quality of alert log

  There are 194 mangled lines in the alert logfiles, which could not be used for this analysis. There may be a software problem. In order to not risk that a grave alert gets lost, steps to solve this problem should be taken. A software update should be obtained or if this problem occures because of concurrent use of this logfiles by more than one process at the same time, it should be considered to use an own log file for each process.

- Sanitizing data before publishing

  The scan logs were not sanitized before releasing them on incidents.org. As I assume that this was not by intention, I suggest more care when giving away log data.

Appendix - Scripts Used

normalize_alerts.sh

```
#!/bin/bash

SCRIPT_HOME=`pwd`
LOG_ORIG=${SCRIPT_HOME}/original
LOG_NORM=${SCRIPT_HOME}/normalized

#
# Remove old files
# Create needed ${LOG_NORM}
#

if [ -d ${LOG_NORM} ]; then
  rm ${LOG_NORM}/tmp*
  rm ${LOG_NORM}/discard*
  rm ${LOG_NORM}/portscan*
  rm ${LOG_NORM}/alert*
else
  mkdir ${LOG_NORM}
fi
```

```
##
## process alert-files
##

for FILE_ORIG in ${LOG_ORIG}/alert.*; do
  FILEDATE=`basename ${FILE_ORIG} | awk -F. '{ print $2 }'`
  printf "\nprocessing alert.$FILEDATE\n##########################\n\n"
  FILEYEAR=`echo ${FILEDATE} | cut -c1-2`
  FILEMONTH=`echo ${FILEDATE} | cut -c3-4`
  FILEDAY=`echo ${FILEDATE} | cut -c5-6`
  FILE=${LOG_NORM}/tmp.${FILEDATE}
  ##
  ##  o uncompressing
  ##
  cp ${FILE_ORIG} ${LOG_NORM}/tmp.${FILEDATE}.gz
  gzip -d ${LOG_NORM}/tmp.${FILEDATE}.gz
  ##
  ## o discarding corrupt entries
  ##   - lines with more than 1 alert  {3 or more occurrences of "[**]}
  ##   - lines not beginning with a date
  ##
  egrep  "\[\*\*\].*\[\*\*\].*\[\*\*\]" ${FILE} > ${LOG_NORM}/discarded.${FILEDATE}
  egrep  -v "\[\*\*\].*\[\*\*\].*\[\*\*\]" ${FILE} > ${LOG_NORM}/tmp.0.${FILEDATE}

  grep  -v ^${FILEMONTH}/${FILEDAY} ${LOG_NORM}/tmp.0.${FILEDATE} \
            > ${LOG_NORM}/discarded.${FILEDATE}
  grep  ^${FILEMONTH}/${FILEDAY} ${LOG_NORM}/tmp.0.${FILEDATE} \
            > ${LOG_NORM}/tmp.1.${FILEDATE}
  ##
  ## STEP 2 - Splitting between portscan and "normal" alerts
  ##
  grep "\[\*\*\] spp_portscan: " ${LOG_NORM}/tmp.1.${FILEDATE} \
              | sed -r 's/ *\[\*\*\] spp_portscan: PORTSCAN DETECTED from /#detected#/g' \
              | sed -r 's/ *\[\*\*\] spp_portscan: portscan status from /#status#/g' \
              | sed -r 's/ *\[\*\*\] spp_portscan: End of portscan from /#end#/g' \
              | sed 's/: /#/g' | sed  's/ (/#/g' \
             > ${LOG_NORM}/portscan.${FILEDATE}
  grep -v "\[\*\*\] spp_portscan: " ${LOG_NORM}/tmp.1.${FILEDATE} \
              | sed -r 's/ *\[\*\*\] */#/g' \
              | sed -r 's/spp_http_decode:/spp_http_decode/g' \
              | sed "s=^\(../..-..:..:.........\)#\(.*\)#\(.*\):\([0-9]*\) -> \(.*\):\([0-9].
              | sed "s=^\(../..-..:..:.........\)#\(.*\)#\(.*\) -> \(.*\)=\1#ALERT#\2#SRC_IP#
               > ${LOG_NORM}/alert.${FILEDATE}
  rm ${LOG_NORM}/tmp*
done
```

normalize_scans.sh

```
#!/bin/bash

SCRIPT_HOME=`pwd`
LOG_ORIG=${SCRIPT_HOME}/original
LOG_NORM=${SCRIPT_HOME}/normalized

#
# Create needed directories
# Remove old files
#
if [  -d ${LOG_NORM} ]; then
  rm ${LOG_NORM}/scans*
else
  mkdir ${LOG_NORM}
fi

if [ -d ${LOG_NORM} ]; then
```

```
  rm ${LOG_NORM}/tmp*
  rm ${LOG_NORM}/scan*
else
  mkdir ${LOG_NORM}
fi

##
## process alert-files
##

for FILE_ORIG in ${LOG_ORIG}/scans.*; do
  FILEDATE=`basename ${FILE_ORIG} | awk -F. '{ print $2 }'`
  printf "\nprocessing scan.$FILEDATE\n##########################\n\n"
  FILEYEAR=`echo ${FILEDATE} | cut -c1-2`
  FILEMONTH=`echo ${FILEDATE} | cut -c3-4`
  FILEDAY=`echo ${FILEDATE} | cut -c5-6`
  FILE=${LOG_NORM}/tmp.${FILEDATE}
  ##
  ## - uncompressing
  ##
  cp ${FILE_ORIG} ${LOG_NORM}/tmp.${FILEDATE}.gz
  gzip -d ${LOG_NORM}/tmp.${FILEDATE}.gz
  ##
  ## - reformat file
  ##
  cat ${FILE} \
    | awk '{ print "03/" $2 "-" $3 "#" $7 "#" $4 "#" $6 "#" $8 " " $9 }' \
    | awk -F: '{ print $1 ":" $2 ":" $3 "#" $4 "#" $5 }' \
    | awk -F\# '{ print $1 "#ALERT#Scan - " $2 "#SRC_IP#" $3 "#SRC_PO#" $4 "#DST_IP#" $5 "#DST
    | sed 's/#SRC_IP#130.85/#SRC_IP#MY.NET/g' \
    | sed 's/#DST_IP#130.85/#DST_IP#MY.NET/g' \
    > ${LOG_NORM}/scans.${FILEDATE}
  rm ${FILE}
done
```

normalize_oos.sh

```
#!/bin/bash

SCRIPT_HOME=`pwd`
LOG_ORIG=${SCRIPT_HOME}/original
LOG_NORM=${SCRIPT_HOME}/normalized

#
# Create needed directories
# Remove old files
#
if [ -d ${LOG_NORM} ]; then
  rm ${LOG_NORM}/oos*
else
  mkdir ${LOG_NORM}
fi

##
## process alert-files
##


for FILE_ORIG in ${LOG_ORIG}/OOS_Report_*; do
  FILEDATE=`basename ${FILE_ORIG} | awk -F_ '{ print "3" $4 $5 }'`
  FILEDATE=0`expr $FILEDATE - 1`
  printf "\nprocessing oos."${FILEDATE}"\n######################\n\n"
  FILEYEAR=`echo ${FILEDATE} | cut -c1-2`
  FILEMONTH=`echo ${FILEDATE} | cut -c3-4`
  FILEDAY=`echo ${FILEDATE} | cut -c5-6`
```

```
  ##
  ## - reformat file
  ##
  cat ${FILE_ORIG} \
    | grep -v  "^.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..  ."\
    | sed "s=^\(../..-..:..:.........\) \(.*\):\([0-9]*\) -> \(.*\):\([0-9].*\)=\1#ALERT#TCP (
    | sed "s=^\(../..-..:..:.........\) \(.*\) -> \(.*\)=\1#ALERT#TCP Out Of Specification#SR(
    | sed "s/^TCP TTL:\(.*\) TOS:\(0x.*\) ID:\([0-9]*\) IpLen:\([0-9]*\) DgmLen:\([0-9]*\) \(
    | sed "s/^TCP TTL:\(.*\) TOS:\(0x.*\) ID:\([0-9]*\) IpLen:\([0-9]*\) DgmLen:\([0-9]*\)$/#1
    | sed "s=^\(........\) Seq: \(0x.*\)  Ack: \(0x.*\)  Win: \(0x.*\)  TcpLen: \([0-9]*\)=#T(
    | sed "s/^TCP Options ... => /#TCP_OPT#/g" \
    | sed "s/^=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+$/\&/g
    | tr -d "\n"  \
    | tr "&" "\n" \
    | tr \* _ \
    > ${LOG_NORM}/oos.${FILEDATE}
done
```

log2summary.sh

```
#!/bin/bash
#
# Usage: log2summary <TYPE> <FIELDS> <LENGTH> <FILTER_INC> <FILTER_EXC>
#
# TYPE: "a" "s" "o"  or any combination
# FIELDS: columns of log include, separate by space
# LENGTH: max. length of generated lists
# FILTER_INC: regexp to include
# FILTER_EXC: regexp to exclude
#
SCRIPT_HOME=/data/gcia/assign3
LOG_NORM=${SCRIPT_HOME}/normalized

if [ -z "$1" ]; then
  TYPE=alert
  TYPEO=alert
else
  TYPEO="$1"
fi
TYPE="[""$TYPEO""]"
echo "$TYPE"

if [ -z "$2" ]; then
  FIELDS='$3'
else
  for FIELD in $2 ; do
  if [ -z "${FIELDS}" ]; then
     FIELDS="$"${FIELD}
  else
     FIELDS="${FIELDS}"' "  | "  $'${FIELD}
  fi
  done
fi


if [ -z "$3" ]; then
  LENGTH=100
else
  LENGTH=$3
fi

if [ -z "$4" ]; then
  FILTER_INC=""
else
  FILTER_INC="$4"
```

```
fi

if [ -z "$5" ]; then
  FILTER_EXC="^\$"
else
  FILTER_EXC="$5"
fi
#
# Remove old files
# Create needed ${LOG_NORM}
#

##
## LIST OF ALERTS
##


echo
echo "TOP ALERTS: INSIDE -> INSIDE($TYPEO)"
echo

cat $LOG_NORM/${TYPE}* \
  | grep "#SRC_IP#MY.NET" \
  | grep "#DST_IP#MY.NET" \
  | egrep "$FILTER_INC" \
  | egrep -v "$FILTER_EXC" \
  | awk -F\# '{ print "#" '"${FIELDS}"' }' \
  | sort | uniq -c | sort -r \
  | head -"${LENGTH}" \
  | awk -F\# '{ printf "%s | %s \n" ,  $1 , $2 }'

echo
echo "TOP ALERTS: INSIDE -> OUTSIDE($TYPEO)"
echo

cat $LOG_NORM/${TYPE}* \
  | grep "#SRC_IP#MY.NET" \
  | grep -v "#DST_IP#MY.NET" \
  | egrep "$FILTER_INC" \
  | egrep -v "$FILTER_EXC" \
  | awk -F\# '{ print  "#" '"${FIELDS}"' }' \
  | sort | uniq -c | sort -r \
  | head -"${LENGTH}" \
  | awk -F\# '{ printf "%s | %s \n" ,  $1 , $2 }'

echo
echo "TOP ALERTS: OUTSIDE -> INSIDE($TYPEO)"
echo

cat $LOG_NORM/${TYPE}* \
  | grep -v "#SRC_IP#MY.NET" \
  | grep "#DST_IP#MY.NET" \
  | egrep "$FILTER_INC" \
  | egrep -v "$FILTER_EXC" \
  | awk -F\# '{ print "#" '"${FIELDS}"' }' \
  | sort | uniq -c | sort -r \
  | head -"${LENGTH}" \
  | awk -F\# '{ printf "%s | %s \n" ,  $1 , $2 }'

echo
echo "TOP ALERTS: OUTSIDE -> OUTSIDE($TYPEO)"
echo

cat $LOG_NORM/${TYPE}* \
  | grep -v "#SRC_IP#MY.NET" \
  | grep -v "#DST_IP#MY.NET" \
  | egrep "$FILTER_INC" \
```

```
| egrep -v "$FILTER_EXC" \
| awk -F\# '{ print "#" '"${FIELDS}"' }' \
| sort | uniq -c | sort -r \
| head -"${LENGTH}" \
| awk -F\# '{ printf "%s | %s \n" ,  $1 , $2 }'
```

References

[1] CERT/CC. " CERT(r) Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks". September 1996 http://www.cert.org/advisories/CA-1996-21.html

[2] Jan Philippo. "SANS Intrusion Detection FAQ: Preventing SYN Flooding with Cisco Routers". September 1996 http://www.sans.org/resources/idfaq/syn_flood.php

[3] Christopher E. Cramer. "Re: [unisog] RE: ATTENTION MORE COMPROMISED COMPUTERS FOUND!", Post on unisog@sans.org. May 2002. http://www.theorygroup.com/Archive/Unisog/2002/msg00677.html

[4] Andrew Mackie, Jensenne Roculan, Ryan Russel and Mario Van Velzen. "Nimda Worm Analysis - Incident Analysis Report Version 2". September 2001. http://www.securityfocus.org/alerts/nimda/010919-Analysis-Nimda.pdf

[5] Tod Beardsley. "Intrusion Detection and Analysis: Theory, Thechniques, and Tools". GCIA Practical. May 2002. http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

[6] Michael Wisener. "Intrusion Detection In Depth: Finding The Needle", GCIA Practical. January 2002. http://www.giac.org/practical/GCIA/Michael_Wisener_GCIA.pdf

[7] Hee So. "GIAC Intrusion Detection In Depth", GCIA Practical. February 2002. http://www.giac.org/practical/Hee_So_GCIA.doc

[8] Khan Rohail. "UDP Traffic on port 22321 AND 7674" Post on security-basics@securityfocus.com. 16 February 2003 http://www.securityfocus.com/archive/105/311770/2003-02-08/2003-02-14/2

Last Updated: May 26th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
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 Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 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**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
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 Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CA**	**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**
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 OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**