GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Network Monitoring and Threat Detection In-Depth (Security 503)"
at http://www.giac.org/registration/gcia

GCIA practical assignment
Version 3.3
Hammersmith, London 23-28 June 2003

Bent Mathiesen
Date submitted: 5 December 2003

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Table of Contents

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, Author retains full rights.

Introduction

What is this report about?

This report is a practical work as part of the GCIA certification. See www.giac.org

The report has 3 parts:

1. Describe the state of intrusion detection [IDS].
   In this part I am looking at some of the latest features in some firewall products and how these features relate to an Intrusion Detection system.
2. Network analyzes
   In this part there are 3 different network traces that get analyzed.
3. "Analyze this"
   In this part 5 days of network logs get analyzed. These logs come from a network at an University.

Typography used in report

Text

Quote: *"Text from other source"*
Command: `command argument1 argument2 (Courier New 10)`
Result output: `This is a computer text in Courier New 9 or 10.`

IP addresses

Lookup of IP addresses have been done with `nslookup` and `whois`.
The format used in the report is (parent range optional):

IP: A,B,C,D	Official name (nslookup)
Name:	Name
Range:	IP range
Parent Range:	IP range
Contact:	Contact Info
Last change:	Date
Notes:	Notes

References:

The references used have the form [id, Name]. In the end of each section, there is a list of references with the form: [id, name]: organizational (name), topic, Url.
Like: [19,Snort]: Snort organization, Snort IDS, www.snort.org

Wordlist:

At the end of the report, there is a small list of URL's to definition of subjects.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Section 1: Describe the state of Intrusion Detection

Use of inline IDS technologies

In my paper I have decided to look at the packet inspection options and IDS integration in some well-known Firewall products. More specific I am looking at Checkpoint SmartDefence, OpenBSD FP and Linux IPTables – with focus on packet crafting detection and logging options.

One of the primary reasons for this analyze is the report from Gartner, who claims that IDS will be dead in 2005. A summary can be found in Web Host Industry Review [16, Thewir]: http://thewhir.com/marketwatch/gar061103.cfm

Recently there have been a new (fast growing) marked for IPS (Intrusion Preventing System). It is evolving very fast. However, I have chosen not to go into this, but to concentrate on some of the (top) firewall players that have been around for some while, to see what trends there have been.

This analyze involve the commercial product SmartDefence (Checkpoint), the free alternative PF (OpenBSD) and Netfilter (Linux). I have chosen these products as they are well known products and they give indication of where firewalls are heading.

At the end there is a compiled table of IDS techniques can be used in the commercial and open source version for an IDS solution.

Inline IDS and Gartner report

A definition of Inline IDS from the Snort Inline Project [8, Inline]:

"Inline IDSs (IIDSs) forward packets after having analyzed the packets for intrusions. Those systems are sometimes also referred to as Gateway IDS (GIDS)".

Comment on Gartner report from Avi Chesla [14, AC]:

"It is not clear however, what Gartner means by saying that firewalls (with more security functionalities – network and applications protections) will replace the IDS products. Intrusion analysis is an entirely different technology than the technology which is associated with current firewalls products. Integrating both technologies in the same product is possible and it seems that this is what Gartner was intending to explain, unsuccessfully."

© SANS Institute 2004, Author retains full rights.

Checkpoint SmartDefence

Checkpoint firewall has, for a long period, offered stateful inspection, antispoofing features and syn flood protection, as the basic, in order to provide network security. However Checkpoint SmartDefence is a relative new part in their Checkpoint firewall product series. This new product, provide more than the basic.

IDS and logging

Inspection

SmartDefence offer the user a number of features related to packet inspection and pattern matching, and based on the result, accept, reject, drop and/or log the packet. For a detailed list of the packet inspection features, take a look at the white paper [1, Checkpoint] and the compiled table [table: “Related features”] later in this section.

The more interesting new features are the inspection of the IP, ICMP inspection. For the TCP packets, the check for type, header and flags, as well as the sequence verification. Attacks that bend the inspected protocols will be logged and or blocked depending on the setup and options.

Logging

The new logging features offer details to what part of the packet inspection that cause the SmartDefence to react to it. While the log still is in the same log file as the normal firewall log, it is treated different as it has an entry “origin” from “SmartDefence”. The log is actually an integrated IDS log.

A typical log entry in SmartDefence, have the form:

```
"Time" "Product" "Origin" "Source" "Destination" "Protocol" "Attack
Name" "Information"
```

Like:

```
"23:59:26" "SmartDefense" "nodename" "A.B.C.D" "E.F.G.H" "icmp"  "Large
ping ""Attack Info: Echo request too long; icmp-type: 8; icmp-code: 0;"
```

A close inspection of the documentation shows, that though it is possible to log some info, like attack type, attack info like protocol, it is not possible to log the actual packet for later inspection.

© SANS Institute 2004, Author retains full rights.

Protection

The Fingerprint/Scrambling features are also interesting. While it can be seen as IPS (Intrusion Prevention System) methods, it is a way to deal with scans [Scan] that happens on every day basis. We also see similar features in the PF and IPTables.

OpenBSD PF

The PF firewall [5, PF], that comes along with OpenBSD, have some nice features for network protocol analyze and actions, reaction as well as scrubbing and normalization feature. Currently OpenBSD is in version 3.4 and the packet filter has got some latest twists and features.

IDS and logging

Inspection

TCP-Flags

The firewall is capable of combining rules with state handling and individual TCP-flags.

Options, TCP-flags:

```
 F: FIN – Finish; end of session
 S: SYN – Synchronize; indicates request to start session
 R: RST – Reset; drop a connection
 P: PUSH – Push; packet is sent immediately
 A: ACK – Acknowledgement
 U: URG – Urgent
 E: ECE - Explicit Congestion Notification Echo
 W: CWR - Congestion Window Reduced
```

Now, with these features (TCP-flags and an API to userspace (see later)), the PF firewall give a range of options to use it for an inline IDS. It is to some extend possible to create specific rules who deal with attacks that violate/misuse the protocols defined and log these incidents for later inspection.

Logging

The PF provide an API (device) to communicate with userspace processes. An example of a process can be a tcpdump [18,Tcpdump] process or a snort sensor that listen on this device.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Protection:

From the document the following have be extracted:

The Scrub feature

This feature provides a way to do traffic normalization. Normalization provide a way to ensure that dangerous network traffic get filtered out from the desired traffic. It can be viewed as advanced power appliance that cut peaks away from the power, in order to secure the delicate electrical equipment.

The manual from PF list the following feature (shortened version):

no-df: Clears the don't fragment bit from the IP packet header
Random-id: Replaces the IP identification field of outbound packets with random values to compensate for operating systems that use predictable values
Min-ttl num: Enforces a minimum Time To Live (TTL) in IP packet headers.
Max-mss num: Enforces a Maximum Segment Size (MSS) in TCP packet headers.
Fragment reassemble: Buffers incoming packet fragments and reassembles them into a complete packet before passing them to the filter engine
Fragment Crop: Causes duplicate fragments to be dropped and any overlaps to be cropped.
Fragment drop-ovl: Similar to fragment crop except that all duplicate or overlapping fragments will be dropped as well as any further corresponding fragments.

Netfilter/IPTables

Netfilter/IPTables is used on the Linux platform and is a part of the kernel 2.4. Netfilter is the kernel part, where IPTables (module) is the part used by netfilter. The firewall provides the normal standard features like stateful inspection and antispoofing. However, Netfilter/IPTables also have a number of also packet mangling and extensions that give a wide range of inspection and logging features.

IDS and logging:

Logging

The LOG option for IPTables takes arguments for logging with level, prefix, TCP-sequence, TCP and IP options to the syslog. While this is useful, there are some more interesting features when you look at the extensions that exist to IPTables.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Ulog [17,Ulog] is not a part of the standard packet, but extensions that allow packet to be send to a *netlink socket*. In short, the packet can be sent from kernel mode to a number (32) of *netlink groups*, where processes using in usermode can process the packet.

TCP-Flags

TCP-Flags: TCP-Flags can be used in the ruleset to control what kind of packets, potential harmful, should be inspected and based on the inspection take an action.

The Flags are: SYN ACK FIN RST URG PSH ALL NONE.

Also there exist a number of extensions, like “Patch-O-Matrix” that provide options for control and log packets that is out of sequence and with ack problems.

There is an option *unclean* that match when a packet is malformed. However, this feature is beta and undocumented. In its current state the documentation say “performing random check”.

Finally there are some „user land states“ (New, Established, Related, Invalid) that allow creating rules based on what status the current connection have.

If we summarize these features, it is possible to inspect the packets for a number of packet crafting options and log them with a prefix (could be attack name) or send to a user land for inspection.

Example of rule, for IDS integration:

```
iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK -j DENY -j
ULOG --ulog-nlgroup 3
```

If one of the flags plus syn and ack is set, it will trigger the rule. The packet will be dropped/blocked and logged to userspace netlink group 3.

Conclusion

What is the state of IDS?

What we can see as a resent trend in use of IDS techniques is the use of IDS as an inline part in a number of old and new products. These products are firewalls, IPS, IDS with API to firewalls and traditional IDS. A conclusion is, instead of merely an IDS that is excellent for IDS purpose, a new use have surfaced in these new product lines.

© SANS Institute 2004, Author retains full rights.

Firewalls have traditional been filtering the traffic based on a few parameters like source, destination and ports. Later on stateful inspection have been added. Now we see that more and more features for inspecting the individual parameters of the packets are being added and with options to log or send the output to further processing.

With Checkpoint we see that they have announced SmartDefence. What is interesting here is the approach to address a number of typical attacks, which you can't with a normal ruleset. They provide some parameterization and log options, along with the possibility to integrate with www.dshield.org.

With Netfilter/IPTables and PF, we see that it is possible to setup these firewall products, not just as a firewall or an IPS, but also to be a part of an IDS solution, by logging detailed information (ranging from textual log to raw packets). With these extensions it is possible to build a ruleset with (pre) categorized traffic with the option to send the same traffic to dedicated processes in userspace for a deeper analyze.

An example for inline IDS (Netfilter and PF) could be a layered approach ("divide and conquer"):

Step:

1. Bad traffic, like no flags or invalid combination, spoofed addresses: drop and log to userspace, where an IDS tuned for bad traffic do further analyze and alarming.
2. Dos traffic, like sudden burst: limit the bandwidth and log to userspace (IDS tuned for Dos, taking action, alarming)
3. Normal legal traffic, log to userspace, where some IDS, with limited ruleset for the actual known services, do further inspection.
4. Drop all the rest, log to userspace where an IDS tuned for other stuff than part 1-3 – like scans. The Snort should log all packets if possible.

(Inline) IDS is not plug and play

A phrase that is often heard is that IDS is not "plug and play". True, you can easily download/buy an IDS and in 1 day have it running in your network. But from the initial setup to the point where the IDS is useful and not just generating false alarms, take some planning, tuning and knowledge.

This also applies to firewalls with inline IDS features. We do not have the perfect firewall/ids combination where the firewall do packet analyzes (as an IPS or inline IDS). No! There are still problems with false positives due to protocols wrongly implemented or used, legal traffic that contain certain patterns used in IDS, etc.

© SANS Institute 2004, Author retains full rights.

A search on the Internet show that some people using Checkpoint SmartDefence have had some problems with the fixed IDS/IPS features, related to FTP, DNS, ICMP. Also for OpenBSD, the features like scrub tend to give some problems and the state handling can cause problems for email. While it can be nice (and useful) to have automatic logging and response to attacks, it must also be possible to see the traffic in details and disable the mechanism when it leads to a failure.

What we see here, is that well known parts of IDS, have been analyses and understood, classified and documented, now find a way into Firewalls and offer a way to do some kind of IDS or provide an extension to existing IDS solutions.

Where firewall does connect more network segments there is a natural place to do IDS. All traffic go through these points and there is less confusions to what the firewall see and what the IDS see if one part control the other. It might also be very possible to tune one part with the other, instead of seeing them as different parts.

Jim Hurst [20, Hurst] has been writing a nice paper on the design options for network IDS related to performance and options.

Comments on Gartner's statement:

I find that the Gartner report do put lights on a number of problems with the standard approach with IDS. However, without these IDS and the analyses from these, it would not be possible today to put IDS techniques into use, like IPS, inline IDS, packet analyzers in firewalls, etc.

An IDS product, like Snort is a fundament to get an understanding of the wild network traffic and problems with protocols and applications that use them. Snort is excellent for playing around to get understanding and look at network traffic in every way. This cannot be done if a product is fixed and only provide limited logging and on/off features.

And what I have failed to see in the firewall products is a working *deep packet inspection*. From the number of discussion on the Internet, it seems clear that in its current state, the inspection of packets with focus on headers is complex, but if including payloads and protocols it becomes a very complex thing.

While IDS technologies have taken a long time to develop, and usually is expensive and questionable as a passive system, a product like Snort is highly configurable, and easily deployed where needed.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

I believe that IDS like many other products have come to a point where the use will become more transparent and widespread. The state of IDS is changing very fast. Now, it is used actively – as more than just for academic analyses and ghost hunting.

Links and references:

[1, Checkpoint]: Checkpoint, Checkpoint, Firewall product/inline IDS, http://www.checkpoint.com/products/downloads/smartdefense_whitepaper.pdf
[2, Checkpoint]: Checkpoint, Application Intelligence/Inline IDS http://www.checkpoint.com/products/downloads/applicationintelligence_whitepaper.pdf
[3, OpenBSD]: Openbsd, Operating System, http://www.openbsd.org
[4, OpenBSD: Software Coordinating Center, OpenBSD and syn flood, http://www.kb.cert.org/vuls/id/JPLA-5AVQAP
[5, PF]: Openbsd, PF FAQ, http://www.openbsd.org/faq/pf/
[6, NF]: Netfilter, firewall, http://www.netfilter.org/
[7, Inline]: Snort Inline, project, http://sourceforge.net/projects/snort-inline/
[8, Inline], Snort org, ids criteria, http://www.snort.org/docs/IDS_criteria.pdf
[9, KJ]: Kobus Jooste, Snort inline and hogwash, http://www.giac.org/practical/GSEC/Kobus_Jooste_GSEC.pdf
[10,Dshield]: Dshield.org, Distributed Intrusion Detection System, http://secure.dshield.org/
[11, Joe], Joe McAlerney, Snort reset discussion, http://www.mcabee.org/lists/snort-users/Jun-01/msg00490.html
[12, LA]: Luiz Arruda, false positive, http://www.giac.org/practical/GSEC/Luiz_Arruda_GSEC.pdf
[13, Neohapis]: Note on Gartner and inline IDS http://archives.neohapsis.com/archives/sf/ids/2003-q2/0313.html
[14, AC]: Avi Chesla, Comment on Gartner's statement, http://lists.insecure.org/lists/focus-ids/2003/Jun/0171.html
[15, PvO]: Peter van Oosterom, Note to recommendations, http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00176.html
[16, Thewir] Web Host Industry Review, Summary of Gartner's conclusion, http://thewhir.com/marketwatch/gar061103.cfm
[17, Ulog], Gnumonks, Ulog project, http://gnumonks.org/projects/ulogd
[18, Tcpdump], Tcpdump, tcpdump project, http://www.tcpdump.org/
[19, Snort], Snort, The Snort IDS, http://www.snort.org/
[20, Hurst]: Jim Hurst, Emerging IDS, http://www.sans.org/resources/idfaq/emerg_nids.php

A feature list of the IDS related technique and features

Explanation to the table: The compiled table is for a comparison/summary of Checkpoints Smartdefence, OpenBSD PF and Linux Netfilter.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The purpose of the table is to see what features currently exist (directly or as an add on) for logging (along with optional blocking) and detecting traffic that relate to known intrusion methods. The table is by no means complete, as there are some add on for both SmartDefence (ApplicationIntelligence) and for PF and Netfilter in form of many user projects

The data is taken from manuals and white papers from the products web site, Gnumonks [17, Ulog], PF Faq [5, PF].

Comment about problems come from search on the net about use of the individual features.

Table: [Related features]

Feature	Checkpoint SD	OpenBSD PF	Linux Netfilter/IPTables
DOS/DDOS			
TearDrop	Block connection with overlap – cannot be disabled	Provide normalization and resolve conflicts. Can this be logged?	*No?*
Ping of Death	Block connection More than max size (64 KB) – cannot be disabled	Rules with limits can be made	Rules with limits can be made
LAND	Block connection cannot be disabled	*No?*	*No?*
IP, ICMP			
Packet Sanity	Check for crafted packets. *How do it inspect and when is it dropped?*	Scrub+TCP options. Provide normalization. Possible to use TCP flags on rules.	Possible to use TCP options and stateful options to create some check.
Max Ping size	It is dropped.	Rules with limits can be made	Rules with limits can be made
IP Fragments	Check for consistency of all packets. Options to block fragments or how fragments are handled.	Set timeout. Set limit (number of fragments) “no-df” “random”. Fragment assemble Fragment crop. Fragment drop-ovl	*No?*
Network Quota	Possible to put limit on single IP’s or single service (to prevent DOS/DDOS.	Queuing. Possible to make definition of classes and quota to these.	Option “iplimit” / “limit” Possible to use option limit.
TCP	Check for type, header and flags are correct.	“Scrub” + tcp flag combinations. *Is it logged ?*	“clean” flag. Still in beta and undocumented
SYN Flood	It can be disabled, enabled (passive or gateway)	PF in OpenBSD 3.4 have a syn proxy, that allow syn protection.	Option “iplimit” / “limit” Possible to use option limit.
Small PMTU	Configuration of size possible. *Can it be*	*“Set limit”. ?*	*No?*

	disabled ? (yes). *Can cause problems.*		
Sequence Verification	If out of sequence, dropped or stripped for data.	*No?*	(Extension) Patch: patch-o-matrix allow sequence and ack control, drop and logging
Fingerprint/Scrambling			
ISN Spoofing	Mask the clients ISN. Default?	Possible to replace all ISN with random sequence	No?
TTL	Possible to mask the TTL of the client. OS. *Not clear if this is default.*	Option Min-ttl	”-t mangle” + ”ttl --ttl-set” can set ttl value. Also option “—ttl-dec” and “ttl-inc” can decrease, raise the ttl.
IP ID	Possible to Mask the client behind	Random-id on outbound packets with modular state.	*Is this possible?*
Dynamic ports	It is possible to block predefined ports from being allocated dynamically	*Can this be configured? Option related?*	*Can this be configured? Option related?*
Web security			
HTTP worm	Possible to configure signatures for worms to be dropped. Support for reassembly and decode of content.	*No?*	*No?*
Cross site Scripting	Default reject of POST commands. Options of tags influence are configurable. Header and payload is inspected.	*No?*	*No?*
HTTP protocol inspection	Option for ASCII header only – to prevent buffer overflows. P2P traffic inspection and possibly blocking.	*No?*	*No?*
Mail security			
SMTP Content	Number of options for masking, mail amount control and security of attachment	*No?*	*No?*
FTP			
FTP Bounce	Options for checking attack.	*No?*	*No?*
FTP security server	Restriction on PUT/GET names and content. Logging of commands done. Option to block for connection made to	*No?*	*No?*

© SANS Institute 2004, Author retains full rights.

	known ports.		
Microsoft related			
File and print sharing	Possible to configure worm signatures for blocking	*No?*	*No?*
DNS	Option to enforce RFC1035 compliance. *Known problems with this feature.*	*No?*	*No?*
IDS integration			
Storm Center integration (www.dshield.org)	It is possible to configure SD management module to send logs to www.shield.org and download block lists.	Possible via applications that go through the logs. Source can be downloaded from www.dshield.org	Plugin can be downloaded from www.dschield.org
Logging and Audit	Logs generated by SmartDefence can be separate extracted and statistic analyzed. Only in Checkpoint format.	Provide a dedicated interface to allow other application to do audit in Tcpdump format. Also possible to log via syslog or packet directly in tcpdump format in a separate file. Additional options in Tcpdump OpenBSD version.	Ulog: Extension for logging to userspace applications.). Ulog PCAP provides compatibility for tcpdump. Log (to kernel, with prefix and different packet header information)
Attack reference and info	Yes, via link in Console to Checkpoint site.	While the firewall do not list attacks by names themselves, the packet that get logged can be send to a combination like Snort+Acid for easy management.	Same as for OpenBSD (further analyze by sending packets to an IDS with a frontend gui)
Successive Events	Setup by changeable parameters. Spoofing Port Scans Number of alerts Multiple connections	*No?*	Limit match

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Section 2, Network Detects

This section contains 3 network detects.

Detect 1: Altered or fragmented Code-Red ?

The snort alert:

```
[**] [1:1322:5] BAD-TRAFFIC bad frag bits [**]
[Classification: Misc activity] [Priority: 3]
11/08-01:53:36.436507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5CA
213.106.126.8 -> 207.166.213.221 TCP TTL:111 TOS:0x0 ID:47694 IpLen:20
DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x0014
```

The actual packet that triggered the alarm:

```
tcpdump -X -n -r 2002.10.8 host 213.106.126.8

01:53:36.436507 213.106.126.8.3010 > 207.166.213.221.80: P
763151695:763153123(1428) ack 3171228475 win 17520 (frag 47694:1448@0+)
0x0000   4500 05bc ba4e 6000 6f06 7d40 d56a 7e08        E....N`.o.}@.j~.
0x0010   cfa6 d5dd 0bc2 0050 2d7c c54f bd05 1b3b        .......P-|.O...;
0x0020   5018 4470 4531 0000 4745 5420 2f64 6566        P.DpE1..GET./def
0x0030   6175 6c74 2e69 6461 3f4e 4e4e 4e4e 4e4e        ault.ida?NNNNNNN
[identical pattern showing "NNNNNNNNNNNNNNNN" deleted for space]
0x0110   4e4e 4e4e 4e25 7539 3039 3025 7536 3835        NNNNN%u9090%u685
0x0120   3825 7563 6264 3325 7537 3830 3125 7539        8%ucbd3%u7801%u9
0x0130   3039 3025 7536 3835 3825 7563 6264 3325        090%u6858%ucbd3%
0x0140   7537 3830 3125 7539 3039 3025 7536 3835        u7801%u9090%u685
0x0150   3825 7563 6264 3325 7537 3830 3125 7539        8%ucbd3%u7801%u9
0x0160   3039 3025 7539 3039 3025 7538 3139 3025        090%u9090%u8190%
0x0170   7530 3063 3325 7530 3030 3325 7538 6230        u00c3%u0003%u8b0
0x0180   3025 7535 3331 6225 7535 3366 6625 7530        0%u531b%u53ff%u0
0x0190   3037 3825 7530 3030 3025 7530 303d 6120        078%u0000%u00=a.
0x01a0   2048 5454 502f 312e 300d 0a43 6f6e 7465        .HTTP/1.0..Conte
0x01b0   6e74 2d74 7970 653a 2074 6578 742f 786d        nt-type:.text/xm
0x01c0   6c0a 484f 5354 3a77 7777 2e77 6f72 6d2e        l.HOST:www.worm.
0x01d0   636f 6d0a 2041 6363 6570 743a 202a 2f2a        com..Accept:.*/*
0x01e0   0a43 6f6e 7465 6e74 2d6c 656e 6774 683a        .Content-length:
0x01f0   2033 3536 3920 0d0a 0d0a 558b ec81 ec18        .3569.....U.....
0x0200   0200 0053 5657 8dbd e8fd ffff b986 0000        ...SVW..........
[data continues, rest deleted]
```

Source of trace

This trace is from the file 2002.10.8 from http://www.incidents.org/logs/Raw/.

Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The document “README” in the directory state:

- The data come from a Snort censor, running in binary mode (Note: Only packets, that triggering rules get logged).
- The data have been sanitized (mugged), including the checksum.
- File with timestamp from the same day have been sanitized the same way.

Detect generated by

Snort version 2.0.2 with the ruleset that was included with the source

Getting background information:

I looked at the raw output from 2002.10.8 with tcpdump using option `“-n –r FILE”`. As a result of this, I saw packets with sync to/from various network and a b-class network 207.106.0.0/16.

This information was used to setup a configuration file for a snort 2.0.2.

Tcpdump, with option `“-e”` for link layer, shows 2 Mac addresses:

0:3:e3:d9:26:c0, 0:0:c:4:b2:33.
Inside, 207.106.0.0/16 have the Mac address: 0:0:c:4:b2:33
Outside, 207.106.0.0/16, have the Mac address: 0:3:e3:d9:26:c0

By using http://coffer.com/mac_find/, we see that the two Mac addresses belong to Cisco Systems:

```
0003E3        Cisco Systems, Inc
00000C        Cisco Systems, Inc
```

Thereby, we know that the network traffic is from a network between two Cisco devices – and what Mac address and IP addresses belong to the inside and outside. The actual location of the sensor can be on a spanning port, hub or even a bridging server/firewall between the 2 Cisco devices.

Snort setup

I took the snort rules file and set it up to use all the included rules and preprocessors as a first start. The inside network was defined as the b-class network from earlier, while the outside was the negation of the inside.

```
var HOME_NET 207.166.0.0/16
var EXTERNAL_NET !$HOME_NET
```

© SANS Institute 2004, Author retains full rights.

Missing alarms

After that I ran the first scan, with the snort 2.0.2 using the newly created snort configuration. To my surprise, I got only 1 alarm. I did a further test with the older snort versions 1.8.4 and 1.9.1 and those generated many alarms.

A closer look on the manual to snort, showed that it was necessary to use the option “-k none” to make snort accept and analyze data with false checksums. A closer look at snort version 2.0.2 also showed that there had been some patches between 2.0.1 and 2.0.2 for mugged data.

I altered the command to run snort and ran it again. The result now was 384 alarms.

After that I used the command that Johnny Wong [22, Johnny] listed in his detect, to make a simple statistic of the generated alert:

```
cat alert | grep '\[\*\*\]' | sort | uniq -c | sort -rn | cat >
alert.stats
```

The output was this:

```
    108 [**] [1:1390:3] SHELLCODE x86 inc ebx NOOP [**]
     51 [**] [1:648:5] SHELLCODE x86 NOOP [**]
     37 [**] [1:618:4] SCAN Squid Proxy attempt [**]
     34 [**] [1:527:4] BAD-TRAFFIC same SRC/DST [**]
     34 [**] [1:184:3] BACKDOOR Q access [**]
     30 [**] [1:628:2] SCAN nmap TCP [**]
     19 [**] [1:1394:3] SHELLCODE x86 NOOP [**]
     11 [**] [1:1322:5] BAD-TRAFFIC bad frag bits [**]
      7 [**] [1:620:3] SCAN Proxy (8080) attempt [**]
      5 [**] [1:615:4] SCAN SOCKS Proxy attempt [**]
      3 [**] [116:46:1] (snort_decoder) WARNING: TCP Data Offset is
less than 5! [**]
      1 [**] [1:653:5] SHELLCODE x86 unicode NOOP [**]
      1 [**] [1:523:4] BAD-TRAFFIC ip reserved bit set [**]
      1 [**] [1:522:1] MISC Tiny Fragments [**]
```

However, I was not satisfied with this initial result. The snort I had been running gave a binary log file, that had far less packets than the original binary file.

A closer look at the binary log from the snort I ran, I had only traffic flowing from Mac 0:3:e3:d9:26:c0 to 0:0:c:4:b2:33 (inbound traffic).

A search via Google, gave the link: http://www.wesi.ch/itsecurity/detect1.html where Daniel Wesemann [23, Wesemann], concludes that IGMP packets where going in “wrong” direction:

© SANS Institute 2004, Author retains full rights.

> *It seems as if only the IGMP packets are violating the direction of the flow on the MAC layer. Consequently, these packets only APPEAR to come from the inside, but in fact are originating from the outside.*

I filtered out this traffic, by using tcpdump options for TCP traffic:

```
tcpdump -n -e -r ../../logs.binaries/2002.10.8 'ether src host 0:0:c:4:b2:33' and 'tcp[13] & 0x0f = 0x08'
```

Bingo, it was outbound traffic on port 80 that was missing. And beside the traffic on port 80 I saw 4 other outbound packets on another port to a destination 64.4.12.179. The IP turned out to be a Hotmail server when I did a check.

Why I did not catch all packets with the standard snort rule set I did not know. I had no further information about the snort setup from the snort that originated the packets. I assumed that the snort was an older version (from 2002) and with a different ruleset, than the one I used, and that those other rules generated more alerts.

However, I was still not satisfied. I wanted to do an analyze of the packets with the latest snort, however, asking myself how to do so without being able to see alerts for all the packets.

I started up the network tool "ethereal" on the original file containing the binary packets. Doing this I saw several inbound packets that did not give alert by my snort setup. One was an SMB packet. A search on the ethereal output on Google I found that Daniel Wesemann [24, Wesemann] do the conclusion that only, web traffic and SMB traffic failed to trigger the rules in snort on the binary log 2002.10.16.

This observation from Daniel Wesemann vent along with the observation I had done.

Selecting a packet for analyze

From ethereal I made a sort of the inbound traffic and spotted a packet that looked familiar in some ways. The source IP was 213.106.126.8 and a part of ethereal listed:

```
0030  44 70 45 31 00 00 47 45  54 20 2f 64 65 66 61 75   DpE1..GE T /defau
0040  6c 74 2e 69 64 61 3f 4e  4e 4e 4e 4e 4e 4e 4e 4e   lt.ida?N NNNNNNNN
```

I wanted to know if this indeed was an attack and therefore searched in the alert file for the IP 213.106.126.8 and saw:

```
[**] [1:1322:5] BAD-TRAFFIC bad frag bits [**]
[Classification: Misc activity] [Priority: 3]
```

```
11/08-01:53:36.436507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5CA
213.106.126.8 -> 207.166.213.221 TCP TTL:111 TOS:0x0 ID:47694 IpLen:20
DgmLen:1468 DF MF
Frag Offset: 0x0000   Frag Size: 0x0014
```

I was not contented with this. The payload of the packet looked to me as being web traffic or a virus. So I vent on Google (again) and did a search on “default.ida”. The result resolved to as a pattern from the virus *code red*.

Why did I see an alarm as `“bad traffic”` and not `“code red”`?

I vent through the snort rules and found only one rule, regarding code red. However, this was not the rule that had triggered the alarm!

The rule for code red was for code red version 2 and looked like this:

```
From web-iis.rules:
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
CodeRed v2 root.exe access"; flow:to_server,established;
uricontent:"/root.exe"; nocase; classtype:web-application-attack;
reference:url,www.cert.org/advisories/CA-2001-19.html; sid:1256;
rev:7;)
```

While the rule that triggered the alarm was:

```
From bad-traffic.rules:
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"BAD-TRAFFIC bad frag
bits"; fragbits:MD; sid:1322; classtype:misc-activity; rev:5;)
```

The fragment bits in the packet, “Don’t fragment and More fragments” (MD) trigger the alarm (marked red in the packet listed in start of detect). This is network traffic that should not normally be seen and therefore raise an alarm.

For testing purpose I ran the snort again, but without the file bad-traffic.rules included. And this time I got no alarm at all for the IP above!

As the alarm from snort didn’t say “code red”, I did a search for rules for code red, but not version 2.

I found a rule for Code Red v1 from John Berkers[28, Berkers].
http://archives.neohapsis.com/archives/snort/2001-08/0165.html

The rule were:

```
alert tcp $EXTERNAL_NET any <> $HTTP_SERVERS 80 (msg: "LOCAL Code Red
v1 IDA Overflow"; dsize: >239; flags: A+; content:"|2F646566 61756C74
2E696461 3F4E4E4E|";)
```

© SANS Institute 2004, Author retains full rights.

After including this rule and running snort again I got the same alarm on the same IP, however **2 other packets** with other IP’s for this type of alarm:

```
[**] [1:0:0] LOCAL Code Red v1 IDA Overflow [**]
[Priority: 0]
11/08-22:14:03.946507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800
len:0x5EA
213.19.167.99:1554 -> 207.166.87.40:80 TCP TTL:240 TOS:0x10 ID:0
IpLen:20 DgmLen:1500
***AP*** Seq: 0xA40C5718  Ack: 0x1F4D802E  Win: 0x7D78  TcpLen: 20
```

The 3 packets that contained the *“get default.ida”* http command and had all the other signatures for code red. However, only 2 of them would fire the code red alarm. A look for the content showed that this was clearly imbedded in the payload.

I vent on Google and did a search for “snort rules code red” and found, related to this, that Corey Merchant [21, Merchant] had posted a practical detect, including a similar trace. This detect can be found at: http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html

The 3 packets that had the “get default.ida” string imbedded, had all the other signatures as well. However, only 2 of them would generate an alarm.

McClure Gammon [30,McClure] propose an answer to this:

> *“I’ve ran into similar issues, which will be the basis for my practical - IMHO snort isn't being deceived by the DF MF flags. We're being deceived because the operation of Snort changed radically between version 1.8.7 and 1.9.0 with the advent of the "flow" keyword and the deprecation of "flags: A+;" If you look at 1.9.0 rules, you'll find that for CodeRed to trigger an alert, the condition "flow: to_server,established;" has to be met. What that translates to is that snort has to see traffic from both directions - client and server to make the "established condition true. Since the logs that we are working with only include alerts logged by snort, we'll never see the response, only the stimulus.”*

As I do not know if there have been any 3 way handshake, I cannot say if it the flow was established. The downloaded file only had this packet. Depending on the snort versions and configuration, the alert result and logged packets will vary.

I did try to disable the preprocessor and reassembler in the Snort 2.0.2 configuration. However, I was still not able to generate a “Code Red v1” alarm for this particular packet. An inspection of the 3 different packets that had the *“default.ida”* string in the payload, showed the identical content:"|2F646566 61756C74 2E696461 3F4E4E4E|"; However, the result from Snort was different. I also did try out the rule for Code red with snort 1.8.7, but was still not able to get the wanted alarm.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Probability the Source Address was spoofed:

The normal way for a Code Red v1 to infect a vulnerable webserver is by establishing a network connection. Based on this, it would make only little sense to use spoofed sources addresses.

Also, in the earlier tcpdump we see that the packet is coming from the inbound interface. Therefore the traffic come from outside (assuming the Internet). Therefore I see no indication of an internal attack.

A check of the IP shows:

213.106.126.8	cust8.manc.broadband.ntl.com
Name:	NTL BIA - Wardley Cable Modem DHCP Pool
Range:	213.106.112.0 - 213.106.127.255
Parent Range:	???
Contact:	For technical issues/questions please - email : nmc@ntli.net For abuse notifications please – abuse@ntlworld.com
Last change:	20030707
Notes:	Ripe

Result: Probably not spoofed.

Description of Attack:

The machine on the Internet is trying to infect a Microsoft IIS server. An open question is if the TCP options are a result of network problems or crafted packets. If crafted (as I suspect), the attack is to infect the IIS server by slipping though the network perimeter by sending packets without establishing the normal connection.

The attack here is a code red worm (it matches clearly the pattern), however, it is not detected by the code-red rules I have found. The main difference between the 2 other packets that did trick the code red rule is:

IP	Trigger	Flags	Other	Differenced service
213.106.126.8	Bad traffic	DF MF AP	TTL 111, (frag 47694:1448@0+)	0x00
213.19.167.99	Code red	AP	TTL 240	0x04
202.130.104.233	Code red	AP	TTL 240	0x04

A closer look on the TTL:

© SANS Institute 2004, Author retains full rights.

Windows 2000 and NT start normally with a TTL of 128. Here the TTL is 111. 128-11=17. 17 are a reasonable distance. I would assume that it is an infected Windows machine that did send the package.

What is special about the traffic seen from 213.106.126.8 is the combination of flags and fragments, plus the virus pattern in the payload. It looks like an attempt to get through a firewall with a code red virus.

Attack Mechanism:

I will not go into lengthy details of the worm mechanism. In short the attack use a buffer overflow and reproduce itself on target system before it launch a target against other systems.

Detailed explanations can be found on the Internet. Microsoft [25, Microsoft] list a description on their website:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/virus/newred.asp
For a deep analyze go to eEye Digital Security [26, eEye]:
http://www.eeye.com/html/Research/Advisories/AL20010717.html
And CVE number: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500

Correlations:

Paul M Young [29, Young] have made analyzes of Worm activities that can go undetected through a Snort Sensor. As he specifically includes the Code Red worm as example, I would recommend his paper for further investigation.

Also Corey Merchant [21, Merchant] had posted a practical detect, including a similar trace. See http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html

Evidence of Active Targeting:

I did a tcpdump –n –r logs.binaries/2002.10.8 host 207.166.213.221, but did not found more traffic from 207.166.213.221.

Furthermore I checked some days before and after using the IP range from the providers modem pool. I did a check with:
tcpdump -n -r logs.binaries/2002.10.8 net 213.106.112.0/20, but found no more traffic.

© SANS Institute 2004, Author retains full rights.

I do not think it was deliberate target toward the IP 207.166.213.221 – and I saw no other signs of traffic. I assume it was random targeting. Also the Code Red Worm does use a random targeting mechanism.

Severity:

I use the following formula for calculating the severity:

Severity = (criticality + lethality) - (system countermeasures + network countermeasures), with 1 as the lowest and 5 as the highest.

Criticality: 3. This is a known exploit. If this does get to a vulnerable webserver, it will take some effort get it back to normal.

Lethality: 4. There is a possible loss of web services and public face.

System countermeasures: 4. As I do not see any code red traffic going out from the network from this target, I will assume the system do not have a vulnerable IIS server installed.

Network countermeasures: 4. I do not know the setup of the network, however, there is an IDS and the system caught the bad traffic and did not show signs of outbound traffic related to this packet.

Severity=(3 + 4) – (4 + 4) = -1

Defense Recommendations:

Do keep your systems patched to fairly new level and keep services on system to a minimum.

If possible, do not make a Webserver directly reachable from the Internet.
An option is to put proxies (reverse proxies for outbound traffic) as a first layer of security. Keep this proxy in shielded network [def, DMZ]. Also, you can place a network device to catch malicious web traffic on the boundary of the network.

Multiple Choice Question:

A tcpdump command list only one packet from a snort alert file, with src host 213.106.126.8:

```
tcpdump -x -t -n -r raws/2002.10.8 host 213.106.126.8

213.106.126.8.3010 > 207.166.213.221.80: P 763151695:763153123(1428)
ack 3171228475 win 17520 (frag 47694:1448@0+)
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
                              4500 05bc ba4e 6000 6f06 7d40 d56a 7e08
                              cfa6 d5dd 0bc2 0050 2d7c c54f bd05 1b3b
                              5018 4470 4531 0000 4745 5420 2f64 6566
                              6175 6c74 2e69 6461 3f4e 4e4e 4e4e 4e4e
[The rest deleted]
```

What do this packet tell us?

a) It is a Code Red worm trying to infect a Microsoft IIS.
b) The traffic comes from a busy network.
c) This is a download of information about code red.
d) This is bad traffic due to the TCP flags used.

Answer: a+d. The analyze of the content of the packet will show a code red pattern. The TCP option DF and MF will likely trigger a “Bad Traffic” alarm when using snort. Is it not a download, as the bad traffic is going to the server (port 80) and not the client.

Feedback from mailing list:

As part of the certification process, this analyze was posted to the mailing list given in the assignment. This was done 23 Nov 2003. Unfortunate there were no feedback on this posting

Link and references:

[21,Merchant]:Corey Merchant, http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html
[22, Wong]: Johnny Wong, Giac practical detect, http://cert.uni-stuttgart.de/archive/intrusions/2003/06/msg00125.html
[23, Wesemann], Daniel Wesemann, Giac practical detect 1, http://www.wesi.ch/itsecurity/detect1.html
[24, Wesemann], Daniel Wesemann, Giac practical detect 2, http://www.wesi.ch/itsecurity/detect2.html
[25, Microsoft]: Microsoft, code red, http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/topics/virus/newred.asp
[26, eEye]: eEye Digital Security, “.ida "Code Red" Worm”, http://www.eeye.com/html/Research/Advisories/AL20010717.html
[27, CVE]: CVE List, Common Vulnerabilities and Exposure, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0500
[28, Berkers]: John Berkers, Code red rules, http://archives.neohapsis.com/archives/snort/2001-08/0165.html
[29, Young]: Paul M Young, GCIA Practical, http://www.giac.org/practical/GCIA/Paul_Young_GCIA.pdf

© SANS Institute 2004, Author retains full rights.

[30,McClure]: McClure Gammon, Snort, http://cert.uni-stuttgart.de/archive/intrusions/2003/02/msg00049.html
[31,Merchant]: Corey Merchant, Fragmented Code Red, http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00237.html

© SANS Institute 2004, Author retains full rights.

Detect 2: Scanning for Web-CGI vulnerabilities

The following trace is based on the following snort alarms:

```
[**] [1:2152:1] WEB-PHP test.php access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
09/23-18:13:44.492215 217.83.197.108:1028 -> YYY.ZZZ.B.20:80
TCP TTL:118 TOS:0x0 ID:2774 IpLen:20 DgmLen:118 DF
***AP*** Seq: 0x5485B285  Ack: 0xFC6E63E6  Win: 0xFF3C  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11617]

[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-18:13:44.496392 YYY.ZZZ.B.20:80 -> 217.83.197.108:1028
TCP TTL:63 TOS:0x0 ID:30735 IpLen:20 DgmLen:521 DF
***AP*** Seq: 0xFC6E63E6  Ack: 0x5485B2D3  Win: 0x16D0  TcpLen: 20

[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
09/23-18:13:44.504394 YYY.ZZZ.B.20:80 -> 217.83.197.108:1029
TCP TTL:63 TOS:0x0 ID:65226 IpLen:20 DgmLen:523 DF
***AP*** Seq: 0xFC71E520  Ack: 0x548690AD  Win: 0x16D0  TcpLen: 20
```

Source of trace

These alarms come from a snort sensor placed on a commercial network. The sensor was placed outside the company firewalls and the network traffic was taken over the week, 16-23 September 2003.

Detect generated by

The sensor was a Snort version 2.0.0 with the standard ruleset from version 2.0.1. Only alarms and packets triggering the alarm were logged, not all the network traffic. Packets were logged with full packet length. The network sensor was set to run for around a week in order to gather enough data to look for slow scans, scans followed by direct attack, etc.

The data in this analyze have been sanitized to hide confidential information. Mac addresses, IP and names have been altered.

Analyze of detect reason

The analyze was performed using the following method:

© SANS Institute 2004, Author retains full rights.

First the Alarms from snort where organized for statistic purpose – and to get a list of the type of alarms Snort had generated. As the amount of alarms where above 700.000, the list of types of alarms became very long. The list of alarms is not listed here in order to respect the company policies.

I quickly decided not to look at the top generators, as it resembled the same type alerts I had seen from my sensor on my private ADSL line. Instead I concentrated on the alerts related to information leaking and related to DNS, mail and web traffic.

At the inspection, most of the alarms turned out to be traffic between internal network and customer related network. I could categorize this as false alarms, based on the content and IP's in the dumps.

Whenever I found a high priority alarm in the Alert file, and the IP listed was an internal, or a suspicious external IP, system with an application related to the alarm I did first an extract for the related traffic with tcpdump and a closer look with ethereal.

```
tcpdump –X –n –r binary.logfile –w bin.A.B.C.D host A.B.C.D
```

And then had a closer look by:

```
ethereal –n –r bin.A.B.C.D
```

Selecting a packet for analyze

I came to the following alarm:

```
[**] [1:2152:1] WEB-PHP test.php access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
09/23-18:13:44.492215 217.83.197.108:1028 -> YYY.ZZZ.B.20:80
TCP TTL:118 TOS:0x0 ID:2774 IpLen:20 DgmLen:118 DF
***AP*** Seq: 0x5485B285  Ack: 0xFC6E63E6  Win: 0xFF3C  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11617]
```

My first reaction was, that it was possible that an external developer had tried to access a test application. However, normally test would be done on test systems and not production, over secure lines, and not from the Internet. Based on this I though it was worth some investigation.

A tcpdump command list the following (Note: [info cut] is to hide company information, and x used in part of url's):

```
tcpdump -X -n -r snort.log.1063710628 host 217.83.197.108
```

© SANS Institute 2004, Author retains full rights.

```
17:13:42.891975 YYY.ZZZ.B.20.80 > 217.83.197.108.2885: P
4227010041:4227010514(473) ack 1408103254 win 5840 (DF)
0x0000   4500 0201 16ec 4000 3f06 f278 8abe 0714        E.....@.?..x....
0x0010   d953 c56c 0050 0b45 fbf3 0df9 53ed f356        .S.l.P.E....S..V
0x0020   5018 16d0 2879 0000 4854 5450 2f31 2e31        P...(y..HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3620        p.2003.16:13:06.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f0a 6f6e 2074 6869 7320        ccess./.on.this.
0x0190   7365 7276 6572 2e3c 503e 0a3c 4852 3e0a        server.<P>.<HR>.
[info cut]
0x01e0   506f 7274 2038 303c 2f41 4444 5245 5353        Port.80</ADDRESS
0x01f0   3e0a 3c2f 424f 4459 3e3c 2f48 544d 4c3e        >.</BODY></HTML>
0x0200   0a                                             .

17:13:44.460526 YYY.ZZZ.B.20.80 > 217.83.197.108.1025: P
4220099230:4220099712(482) ack 1417885748 win 5840 (DF)
0x0000   4500 020a 817c 4000 3f06 87df 8abe 0714        E....|@.?.......
0x0010   d953 c56c 0050 0401 fb89 9a9e 5483 3834        .S.l.P......T.84
0x0020   5018 16d0 8e41 0000 4854 5450 2f31 2e31        P....A..HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3820        p.2003.16:13:08.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f69 6e64 6578 2e70 6870        ccess./index.php
0x0190   0a6f 6e20 7468 6973 2073 6572 7665 722e        .on.this.server.
0x01a0   3c50 3e0a 3c48 523e 0a3c 4144 4452 4553        <P>.<HR>.<ADDRES
```

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

```
[info cut]
0x01f0   3c2f 4144 4452 4553 533e 0a3c 2f42 4f44        </ADDRESS>.</BOD
0x0200   593e 3c2f 4854 4d4c 3e0a                       Y></HTML>.
17:13:44.464527 YYY.ZZZ.B.20.80 > 217.83.197.108.1026: P
4220754089:4220754570(481) ack 1417928211 win 5840 (DF)
0x0000   4500 0209 834d 4000 3f06 860f 8abe 0714        E....M@.?.......
0x0010   d953 c56c 0050 0402 fb93 98a9 5483 de13        .S.l.P......T...
0x0020   5018 16d0 07a3 0000 4854 5450 2f31 2e31        P.......HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3820        p.2003.16:13:08.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f6d 6169 6e2e 7068 700a        ccess./main.php.
0x0190   6f6e 2074 6869 7320 7365 7276 6572 2e3c        on.this.server.<
[info cut]
0x01e0   xxxx xx2e 6e65 7420 506f 7274 2038 303c        xxx.net.Port.80<
0x01f0   2f41 4444 5245 5353 3e0a 3c2f 424f 4459        /ADDRESS>.</BODY
0x0200   3e3c 2f48 544d 4c3e 0a                         ></HTML>.

17:13:44.489365 YYY.ZZZ.B.20.80 > 217.83.197.108.1027: P
4234507017:4234507501(484) ack 1417988600 win 5840 (DF)
0x0000   4500 020c 68f0 4000 3f06 a069 8abe 0714        E...h.@.?..i....
0x0010   d953 c56c 0050 0403 fc65 7309 5484 c9f8        .S.l.P...es.T...
0x0020   5018 16d0 beb8 0000 4854 5450 2f31 2e31        P.......HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3820        p.2003.16:13:08.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f70 6870 696e 666f 2e70        ccess./phpinfo.p
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x0190   6870 0a6f 6e20 7468 6973 2073 6572 7665        hp.on.this.serve
0x01a0   722e 3c50 3e0a 3c48 523e 0a3c 4144 4452        r.<P>.<HR>.<ADDR
[info cut]
0x01f0   3830 3c2f 4144 4452 4553 533e 0a3c 2f42        80</ADDRESS>.</B
0x0200   4f44 593e 3c2f 4854 4d4c 3e0a                  ODY></HTML>.

17:13:44.492215 217.83.197.108.1028 > YYY.ZZZ.B.20.80: P
1418048133:1418048211(78) ack 4235092966 win 65340 (DF)
0x0000   4500 0076 0ad6 4000 7606 c919 d953 c56c        E..v..@.v....S.l
0x0010   8abe 0714 0404 0050 5485 b285 fc6e 63e6        .......PT....nc.
0x0020   5018 ff3c 6f96 0000 4745 5420 2f74 6573        P..<o...GET./tes
0x0030   742e 7068 7020 4854 5450 2f31 2e30 0d0a        t.php.HTTP/1.0..
0x0040   486f 7374 3a20 xxxx xxxx xxxx xxxx xxxx        Host:.YYY.ZZZ.B.
0x0050   3230 0d0a 4163 6365 7074 3a20 2a2f 2a0d        20..Accept:.*/*.
0x0060   0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f        .Connection:.clo
0x0070   7365 0d0a 0d0a                                 se....

17:13:44.496392 YYY.ZZZ.B.20.80 > 217.83.197.108.1028: P 1:482(481) ack 78 win
5840 (DF)
0x0000   4500 0209 780f 4000 3f06 914d 8abe 0714        E...x.@.?..M....
0x0010   d953 c56c 0050 0404 fc6e 63e6 5485 b2d3        .S.l.P...nc.T...
0x0020   5018 16d0 5cb6 0000 4854 5450 2f31 2e31        P...\...HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3820        p.2003.16:13:08.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f74 6573 742e 7068 700a        ccess./test.php.
0x0190   6f6e 2074 6869 7320 7365 7276 6572 2e3c        on.this.server.<
0x01a0   503e 0a3c 4852 3e0a 3c41 4444 5245 5353        P>.<HR>.<ADDRESS
[info cut]
0x01f0   2f41 4444 5245 5353 3e0a 3c2f 424f 4459        /ADDRESS>.</BODY
0x0200   3e3c 2f48 544d 4c3e 0a                         ></HTML>.

17:13:44.504394 YYY.ZZZ.B.20.80 > 217.83.197.108.1029: P
4235322656:4235323139(483) ack 1418105005 win 5840 (DF)
0x0000   4500 020b feca 4000 3f06 0a90 8abe 0714        E.....@.?.......
0x0010   d953 c56c 0050 0405 fc71 e520 5486 90ad        .S.l.P...q..T...
0x0020   5018 16d0 19f3 0000 4854 5450 2f31 2e31        P.......HTTP/1.1
0x0030   2034 3033 2046 6f72 6269 6464 656e 0d0a        .403.Forbidden..
0x0040   4461 7465 3a20 5475 652c 2032 3320 5365        Date:.Tue,.23.Se
0x0050   7020 3230 3033 2031 363a 3133 3a30 3820        p.2003.16:13:08.
0x0060   474d 540d 0a53 6572 7665 723a 2041 7061        GMT..Server:.Apa
0x0070   6368 652f 312e 332e 3236 2028 556e 6978        che/1.3.26.(Unix
0x0080   2920 4465 6269 616e 2047 4e55 2f4c 696e        ).Debian.GNU/Lin
0x0090   7578 206d 6f64 5f70 6572 6c2f 312e 3236        ux.mod_perl/1.26
0x00a0   0d0a 436f 6e6e 6563 7469 6f6e 3a20 636c        ..Connection:.cl
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

```
0x00b0   6f73 650d 0a43 6f6e 7465 6e74 2d54 7970        ose..Content-Typ
0x00c0   653a 2074 6578 742f 6874 6d6c 3b20 6368        e:.text/html;.ch
0x00d0   6172 7365 743d 6973 6f2d 3838 3539 2d31        arset=iso-8859-1
0x00e0   0d0a 0d0a 3c21 444f 4354 5950 4520 4854        ....<!DOCTYPE.HT
0x00f0   4d4c 2050 5542 4c49 4320 222d 2f2f 4945        ML.PUBLIC."-//IE
0x0100   5446 2f2f 4454 4420 4854 4d4c 2032 2e30        TF//DTD.HTML.2.0
0x0110   2f2f 454e 223e 0a3c 4854 4d4c 3e3c 4845        //EN">.<HTML><HE
0x0120   4144 3e0a 3c54 4954 4c45 3e34 3033 2046        AD>.<TITLE>403.F
0x0130   6f72 6269 6464 656e 3c2f 5449 544c 453e        orbidden</TITLE>
0x0140   0a3c 2f48 4541 443e 3c42 4f44 593e 0a3c        .</HEAD><BODY>.<
0x0150   4831 3e46 6f72 6269 6464 656e 3c2f 4831        H1>Forbidden</H1
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f69 6e64 6578 2e70 6870        ccess./index.php
0x0190   330a 6f6e 2074 6869 7320 7365 7276 6572        3.on.this.server
0x01a0   2e3c 503e 0a3c 4852 3e0a 3c41 4444 5245        .<P>.<HR>.<ADDRE
[info cut]
0x01f0   303c 2f41 4444 5245 5353 3e0a 3c2f 424f        0</ADDRESS>.</BO
0x0200   4459 3e3c 2f48 544d 4c3e 0a                    DY></HTML>.
```

By the tcpdump on the host IP, I found that there were 7 packets coming to or from the IP, not just the single alarm above.

By looking into the alert file, I found 2 types of alarms:

For the inbound traffic:

```
[**] [1:2152:1] WEB-PHP test.php access [**]
[Classification: access to a potentially vulnerable web application]
[Priority: 2]
```

For the outbound traffic:

```
[**] [1:1201:7] ATTACK-RESPONSES 403 Forbidden [**]
[Classification: Attempted Information Leak] [Priority: 2]
```

As there where 6 outbound packets and only 1 inbound – there were more into this – and it looked very much like someone was trying to access an application or information that was not supposed to be accessible.

By looking up the IP. I found it to belong to a dial up pool to Deutches Telecom in Germany.

IP: 217.83.197.108	pD953C56C.dip.t-dialin.net
Name:	Deutsche Telekom AG
Range:	217.80.0.0 - 217.89.31.255
Parent range:	
Contact:	Person: Security Team address: Deutsche Telekom AG address: Germany phone: +49 180 5334332 fax-no: +49 180 5334252

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

	e-mail: abuse@t-ipnet.de
	nic-hdl: DTST
	mnt-by: DTAG-NIC
	changed: abuse@t-ipnet.de
Last change:	20030210
Notes:	

As the logs from snort showed traffic between the client on the Internet and the server, I vent on to have a look at the logs from the firewalls on the inside.

On the firewall I started to check the log from the day and time of the alert: 09/23-18:13:44.

The firewall contained the following:

```
"23Sep2003"  "18:12:53"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.A.128"
"23Sep2003"  "18:12:53"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.A.129"
[entries of drop http to YYY.ZZZ.A.130 – YYY.ZZZ.A.254 deleted]
"23Sep2003"  "18:12:54"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.A.255"
"23Sep2003"  "18:12:59"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
"23Sep2003"  "18:12:59"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.21"
"23Sep2003"  "18:12:59"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.22"
"23Sep2003"  "18:12:59"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.23"
"23Sep2003"  "18:12:59"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.24"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.25"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.26"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.27"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.28"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.29"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.30"
"23Sep2003"  "18:12:59"  "drop"    "http"  "217.83.197.108"  "YYY.ZZZ.B.31"
"23Sep2003"  "18:13:01"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
"23Sep2003"  "18:13:01"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
"23Sep2003"  "18:13:01"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
"23Sep2003"  "18:13:01"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
"23Sep2003"  "18:13:01"  "accept"  "http"  "217.83.197.108"  "YYY.ZZZ.B.20"
```

The interpretation of the firewall log:

It turned out that the client on the Internet, had been sending http request to all the IP's that where reachable on the network. YYY.ZZZ.A.128/25 network had been scanned and resulted in drops – as the firewall did stop this traffic.

After that the scan continued to another network segment. Also here the YYY.ZZZ.B.16/28 subnet was scanned for http servers. As this was a hosting zone for servers with port 80 open, reachable from the outside, the scan finally were able get a response from a server.

From the log we see that servers YYY.ZZZ.B.20-24 have been given the http request. However, from this firewall log, we do not know if the servers have replied, and if they did, what the reply was.

© SANS Institute 2004, Author retains full rights.

We also see that only server YYY.ZZZ.B.20 have been sending more than one http request.

To investigate this further, I did some http request to these servers, similar to the one in the network trace, to see what the response where.

During my test, all servers, except YYY.ZZZ.B.20 answered with the error “404, Not found”. Server YYY.ZZZ.B.20 answered with “403, Access denied”.

“403, Access denied” was also the trigger for the alarms that Snort generated on the outbound traffic.

After this, I used the reference in Snort, http://cgi.nessus.org/plugins/dump.php3?id=11617 [31,Nessus], to check the related info for the inbound traffic.

The reference says:

> *The remote server is running Horde and/or IMP with test scripts available from the outside. The scripts may leak server-side information that is valuable to an attacker.*
>
> *Solution: test.php and imp/test.php should be deleted,*
> *or they should be made unreadable by the web server.*
> *Risk factor : Medium*

I did not have the logs from the servers or information about the application running. However, it did look to me, that if there were a Horde or IMP on the servers, it had already been hardened by making the files unreachable from outside.

To check if there were any signs of damages I continued to look though the logs of the firewalls. Specifically I looked for entries of incoming traffic from the range, 217.80.0.0 - 217.89.31.255 (The range of the dynamic pool from Deutche Telecom) and for outbound traffic from the server YYY.ZZZ.B.20. I checked several days and there were no signs of network activity. It looked like the client that had done the scan for Webservers, had not returned later with any new IP from the range that was listed. Neither was there any (initiated) outbound traffic from the server YYY.ZZZ.B.20.

Probability the Source Address was spoofed:

© SANS Institute 2004, Author retains full rights.

From the firewall logs it was clear that the traffic to the web server was in fact coming from the Internet interface. A check of the network traffic at layer 2 (Mac addresses) confirmed this.

Also the client did successfully do a TCP 3 way handshake on port 80 – the traffic did get back to the client. If not, the packet would be been dropped by the firewall as “Packet out of state”.

Therefore the address is reachable, and likely not spoofed.

Description of Attack:

The attack was a search for a vulnerable http server that could leak further information. It was successful in its search to find a server who could give a wanted response. After that, further http request was send to the server.

The attack where done in 2 phases.

First phase:

A search on port 80 (http) was done from a client. This search was looking for a certain response from a web server. Those with response “404 – Not found” where ignored, those with “403 – Access denied” where investigated further.

Second phase:

After a web server, with the right response pattern was found, the attacking client where doing some specific web attack.

It is a reconnaissance attempt.

Attack Mechanism:

The first part of the attack is the information gathering (reconnaissance attempt). As there where no information leaking and further packets, we cannot say directly what the next stage of the attack would have been.

The reconnaissance attempt, is working by doing a “get /test.php” like this:

```
tcpdump -n -X -r bin.217.83.197.108 src 217.83.197.108
17:13:44.492215 217.83.197.108.1028 > YYY.ZZZ.B.20.80: P
1418048133:1418048211(78) ack 4235092966 win 65340 (DF)
0x0000   4500 0076 0ad6 4000 7606 c919 d953 c56c        E..v..@.v....S.l
0x0010   8abe 0714 0404 0050 5485 b285 fc6e 63e6        .......PT....nc.
0x0020   5018 ff3c 6f96 0000 4745 5420 2f74 6573        P..<o...GET./tes
0x0030   742e 7068 7020 4854 5450 2f31 2e30 0d0a        t.php.HTTP/1.0..
```

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

© SANS Institute 2004, Author retains full rights.

```
0x0040   486f 7374 3a20 xxxx xxxx xxxx xxxx xxxx        Host:.YYY.ZZZ.B.
0x0050   xxxx 0d0a 4163 6365 7074 3a20 2a2f 2a0d        20..Accept:.*/*.
0x0060   0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f        .Connection:.clo
0x0070   7365 0d0a 0d0a                                 se....
```

Based on the timestamps on the firewall and the snort logs, it looks like the process is automated both for the search and the direct inquiry.

An extract of the dumps of the outbound packets captured:

```
0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f0a 6f6e 2074 6869 7320        ccess./.on.this.
0x0190   7365 7276 6572 2e3c 503e 0a3c 4852 3e0a        server.<P>.<HR>.

0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f69 6e64 6578 2e70 6870        ccess./index.php
0x0190   0a6f 6e20 7468 6973 2073 6572 7665 722e        .on.this.server.
0x01a0   3c50 3e0a 3c48 523e 0a3c 4144 4452 4553        <P>.<HR>.<ADDRES

0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f6d 6169 6e2e 7068 700a        ccess./main.php.
0x0190   6f6e 2074 6869 7320 7365 7276 6572 2e3c        on.this.server.<
0x01a0   503e 0a3c 4852 3e0a 3c41 4444 5245 5353        P>.<HR>.<ADDRESS

0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f70 6870 696e 666f 2e70        ccess./phpinfo.p
0x0190   6870 0a6f 6e20 7468 6973 2073 6572 7665        hp.on.this.serve
0x01a0   722e 3c50 3e0a 3c48 523e 0a3c 4144 4452        r.<P>.<HR>.<ADDR

0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f74 6573 742e 7068 700a        ccess./test.php.
0x0190   6f6e 2074 6869 7320 7365 7276 6572 2e3c        on.this.server.<

0x0160   3e0a 596f 7520 646f 6e27 7420 6861 7665        >.You.don't.have
0x0170   2070 6572 6d69 7373 696f 6e20 746f 2061        .permission.to.a
0x0180   6363 6573 7320 2f69 6e64 6578 2e70 6870        ccess./index.php
0x0190   330a 6f6e 2074 6869 7320 7365 7276 6572        3.on.this.server
```

Correlations:

I have not found correlations for these specific http requests. However, a search on Google did result in a number of sites that list statistics of web request that have failed. A number of these contained the same strings as seen above in bold.

The Nessus reference: http://cgi.nessus.org/plugins/dump.php3?id=11617 [31,Nessus].

© SANS Institute 2004, Author retains full rights.

Evidence of Active Targeting:

It is a scan for vulnerable webservers containing certain applications. While the initial scan itself do not target directly, the result of this do result direct targeting.

Severity:

Severity = (criticality + lethality) - (system countermeasures + network countermeasures), with 1 as the lowest and 5 as the highest.

Critically: 3 (medium). This is a reconnaissance attempt to find a vulnerable server.

Lethality: 3. If a vulnerable server is found, important information about network and servers can leak out.

System countermeasures: 4. Only 1 server gave a wanted feedback, however, further penetration did not give any response.

Network countermeasures: 5: The servers are protected in zones that will allow cgi exploits to provide further use for internal or external attack and information gathering.

Severity = (3 + 3) – (4 + 5) = -3

Defense Recommendations:

In general:

I would recommend avoiding test systems in production zone. All applications must be cleanly installed on a production system with minimized base, in order to prevent unnecessary application and services to live on a public server.

The systems in production zones should only have the necessary access to the outside world. When so set up, and a new vulnerability with the application running on the server gets known, it will still be a target, but would have limited risk to launch new attacks towards other targets, if infected.

And for known attacks against test applications. If the firewalls have a possibility to use an inline IDS, or application inspections, it would be possible to stop some of the attacks directed against a default setup with known vulnerabilities.

Specifically:

For any Horde and/or IMP application, delete the test.php and imp/test.php files.

© SANS Institute 2004, Author retains full rights.

Multiple Choice Question:

You have a tcpdump that shows:

```
tcpdump -n -X -r bin.217.83.197.108 src 217.83.197.108

17:13:44.492215 217.83.197.108.1028 > YYY.ZZZ.B.20.80: P
1418048133:1418048211(78) ack 4235092966 win 65340 (DF)
0x0000   4500 0076 0ad6 4000 7606 c919 d953 c56c        E..v..@.v....S.l
0x0010   8abe 0714 0404 0050 5485 b285 fc6e 63e6        .......PT....nc.
0x0020   5018 ff3c 6f96 0000 4745 5420 2f74 6573        P..<o...GET./tes
0x0030   742e 7068 7020 4854 5450 2f31 2e30 0d0a        t.php.HTTP/1.0..
0x0040   486f 7374 3a20 xxxx xxxx xxxx xxxx xxxx        Host:.YYY.ZZZ.B.
0x0050   xxxx 0d0a 4163 6365 7074 3a20 2a2f 2a0d        20..Accept:.*/*.
0x0060   0a43 6f6e 6e65 6374 696f 6e3a 2063 6c6f        .Connection:.clo
0x0070   7365 0d0a 0d0a                                 se....
```

What do this information tell us?

1: It is a scan for open ports on a server
2: It is a test for finding a vulnerable web server using HTTP/1.0
3: It is a http request to find a developer application
4: It is Dos attempt to make a webserver fill itself with request

1: While it would make sense to use a GET command for testing on port 80, it is not normal that scan for open port on a single server happens this way.
2: The packet shows no sign of crafting for buffer overflows or potential dangerous http options.
3: The packet does try to run a test.php application with options.
4: While it could have an interesting effect to ask an application with options to the server itself, the client here seem to complete a 3 way TCP handshake and would have problem to send many request if the server get loaded – and do not seem to use a spoofed IP either.

Answer 3: The packet is a test trying to find a test application on web servers. While web servers can have many vulnerabilities in the web server itself, often a default installation have test application for showing the web servers setup. A successful test can reveal some amount of information.

Link and references

[31,Nessus]: Nessus, Scanner,
http://cgi.nessus.org/plugins/dump.php3?id=11617

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Detect 3: A new Ring Zero/Scan for proxies?

The following tcpdump output:

```
16:17:06.831053 200.152.97.160.48626 > 81.62.27.208.1080: S 666666:666666(0)
win 16384 [tos 0x10]
                         4510 0028 8798 0000 f106 aae0 c898 61a0
                         513e 1bd0 bdf2 0438 000a 2c2a 0000 0000
                         5002 4000 ea3c 0000
16:17:06.836861 200.152.97.160.51006 > 81.62.27.208.3128: S 666666:666666(0)
win 16384 [tos 0x10]
                         4510 0028 4ff6 0000 f106 e282 c898 61a0
                         513e 1bd0 c73e 0c38 000a 2c2a 0000 0000
                         5002 4000 d8f0 0000
16:17:06.838538 200.152.97.160.59807 > 81.62.27.208.8080: S 666666:666666(0)
win 16384 [tos 0x10]
                         4510 0028 7fd6 0000 f106 b2a2 c898 61a0
                         513e 1bd0 e99f 1f90 000a 2c2a 0000 0000
                         5002 4000 a337 0000
```

Source of trace

This trace was taken from my home network. For a longer period I have had a personal firewall (IPTables) on a Linux box and running snort 2.0.1. As part of the certification process I wanted to get a closer look on the network traffic coming to my computers via the ADSL.

Detect generated by

The sensor is a snort Version 2.0.1 (Build 88), placed directly in my primary computer. This computer is connected directly to the Internet via ADSL.

The HOME network is 192.168.0.0/16, where all other physical and virtual computer are, and the Internet/ADSL interface is ppp0.

In the ruleset the dynamic assigned interface for the ADSL + the home network is
`HOME_NET [$ppp0_ADDRESS,192.168.0.0/24]`

The detect: Active scan for proxies

```
[**] [1:615:4] SCAN SOCKS Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/15-16:17:06.831053 200.152.97.160:48626 -> 81.62.27.208:1080
TCP TTL:241 TOS:0x10 ID:34712 IpLen:20 DgmLen:40
******S* Seq: 0xA2C2A  Ack: 0x0  Win: 0x4000  TcpLen: 20
[Xref => http://help.undernet.org/proxyscan/]

[**] [1:618:4] SCAN Squid Proxy attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/15-16:17:06.836861 200.152.97.160:51006 -> 81.62.27.208:3128
```

© SANS Institute 2004, Author retains full rights.

```
TCP TTL:241 TOS:0x10 ID:20470 IpLen:20 DgmLen:40
******S* Seq: 0xA2C2A  Ack: 0x0  Win: 0x4000  TcpLen: 20

[**] [1:620:3] SCAN Proxy (8080) attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/15-16:17:06.838538 200.152.97.160:59807 -> 81.62.27.208:8080
TCP TTL:241 TOS:0x10 ID:32726 IpLen:20 DgmLen:40
******S* Seq: 0xA2C2A  Ack: 0x0  Win: 0x4000  TcpLen: 20
```

The 3 rules that triggered the alarms where:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS Proxy
attempt"; flags:S,12; reference:url,help.undernet.org/proxyscan/;
classtype:attempted-recon; sid:615; rev:4;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy
attempt"; flags:S,12; classtype:attempted-recon; sid:618; rev:4;)
alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy \(8080\)
attempt"; flags:S,12; classtype:attempted-recon; sid:620; rev:3;)
```

As I start looking at this scan, it proved to be interesting. It tried to scan for 3 different proxy ports – and the scan was seen over more days. And as I am running a squid proxy myself, this was a natural choice of detect to look at.

Analyze

The packages

The first I noticed was that all packages had the same sequence number of 666666. This gave a very strong indication of crafted packages.

As my Linux machine have a firewall setup with IPTables, dropping all incoming traffic, I started to check the logs from there.

The firewall logs

The logs showed more ports being scanned:

Log 1: (First time scan pattern seen):

```
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=34712
PROTO=TCP SPT=48626 DPT=1080 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=56013
PROTO=TCP SPT=52291 DPT=1075 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=30205
PROTO=TCP SPT=37568 DPT=4588 WINDOW=16384 RES=0x00 SYN URGP=0
```

© SANS Institute 2004, Author retains full rights.

```
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=64217
PROTO=TCP SPT=11257 DPT=6588 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=20470
PROTO=TCP SPT=51006 DPT=3128 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 15 17:17:06 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.27.208 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=32726
PROTO=TCP SPT=59807 DPT=8080 WINDOW=16384 RES=0x00 SYN URGP=0
```

Log 2: (Second time scan pattern seen):

```
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=39222
PROTO=TCP SPT=56220 DPT=1080 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=42023
PROTO=TCP SPT=14757 DPT=1075 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=61161
PROTO=TCP SPT=378 DPT=4588 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=50209
PROTO=TCP SPT=64811 DPT=6588 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=30733
PROTO=TCP SPT=5512 DPT=3128 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 16 17:15:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.31.232 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=65027
PROTO=TCP SPT=52001 DPT=8080 WINDOW=16384 RES=0x00 SYN URGP=0
```

Log 3 (Third and last day scan pattern seen):

```
Aug 17 16:17:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.152.134 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=39469
PROTO=TCP SPT=11392 DPT=1080 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 17 16:17:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.152.134 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=47962
PROTO=TCP SPT=8889 DPT=1075 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 17 16:17:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.152.134 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=20638
PROTO=TCP SPT=22270 DPT=4588 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 17 16:17:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.152.134 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=59185
PROTO=TCP SPT=40543 DPT=6588 WINDOW=16384 RES=0x00 SYN URGP=0
Aug 17 16:17:43 linux kernel: SuSE-FW-DROP-DEFAULT IN=ppp0 OUT= MAC=
SRC=200.152.97.160 DST=81.62.152.134 LEN=40 TOS=0x10 PREC=0x00 TTL=241 ID=31308
PROTO=TCP SPT=37036 DPT=3128 WINDOW=16384 RES=0x00 SYN URGP=0
```

Probability the Source Address was spoofed:

I did a check on the IP of the source of these scans.
It did belong to a small network: 200.152.97.160/29

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

200.152.97.160	For.more.information.check.proxyblocker.com.br
Name:	reydson telecom ltda
Range:	200.152.97.160/29
Parent Range:	
Contact:	Security issues should also be addressed to nbso@nic.br, http://www.nic.br/nbso.html
Last change:	20030626
Notes:	

It seem that the source IP is being used for proxying (or blocking proxies?) according to the official name.
I was wondering if it were an infected proxy who was being used for scanning for other proxies?

As the idea behind the scan is to seek out vulnerable proxy servers, it would have no purpose to use spoofed addresses. The TTL of 241 – assuming it started with 255, matched the observation I did by trying traceroute. I was not able to do a traceroute the whole way, however, I got close with some 15 hop.

From the source port increments, it seem that this system is doing a lot of network traffic. That would make sense if it is a proxy and/or is scanning networks.

There is also the possibility that this is a Dos attack, targeting systems on the small network behind the source IP, using normal http traffic to generate the stimuli.

The IP is likely not spoofed.

Description of Attack:

The host is trying to find proxies or web services with an insecure setup. The scan have been run for a longer period (several days) and scans for a number of fixed ports.

Attack Mechanism:

A number of sync packets are send to the target host on ports where one could find a proxy. This was information gathering.

As there was no replies to the packets send, we cannot say what specific attack mechanism would have been.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Correlations:

From the list:

Simon Ktung [42,Ktung], have made similar analyze in his practical detect #2.

Also Zela [43,Zela] have made a similar analyze in his practical detect #1.

Compguruman [44, Compguruman] list on security basis mailing list, the following:

> *I've been getting port scans from the same IP address for 3 days.*
> *[…]*
> *It scans to see if ports **1075, 3128, 4588, 6588, and 8080** are open. I ran retina against the machine and its running a default install of Apache without much anything configured. The Sequence # of the packets are always **666666** and all have **the SYN flag set**.*

It matches closely the scans I saw on my machine.

A follow up from John Choe [45, John], says:

> *"The port 6588 scan is for AnalogX Proxy server:*
> *http://archive.develooper.com/qpsmtpd@perl.org/msg00390.html*
> *They may be looking for insecure proxies to bounce spam off of"*

Evidence of Active Targeting:

The network traffic shows signs of scans for systems, gathering information about proxy servers and to find vulnerable cgi applications.

However the scan is not directly targeting.

Severity:

Using the formula from network detect 1.

Severity = (criticality + lethality) - (system countermeasures + network countermeasures), with 1 as the lowest and 5 as the highest.

Criticality: 4. If a proxy or a web server gets infected, it can be used to send malicious traffic to other targets.

Lethality: 4. If a proxy server get infected it can be misused.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

System countermeasures: 5. Here the system is configured only to allow private range internal IP addressed to use the proxy.

Network countermeasures: 5. The firewall only allows outbound connections to be established. All incoming traffic not part of an existing connection will be dropped.

Severity = (4 + 4) - (5 + 5) = -2

Defense Recommendations:

It is important to have the right configuration for traffic flow in a proxy server. Do set up a firewall to prevent malicious traffic to go in/out. Do you antispoofing for traffic control. Keep the systems and proxies updated with patches – and minimize other services running.

In particular, a proxy server should not be allowed any other traffic for going out, than is necessary for it’s function.

Multiple Choice Question:

A tcpdump command shows this packet:

```
16:17:06.831053 200.152.97.160.48626 > 81.62.27.208.1080: S
666666:666666(0) win 16384 [tos 0x10]
                         4510 0028 8798 0000 f106 aae0 c898 61a0
                         513e 1bd0 bdf2 0438 000a 2c2a 0000 0000
                         5002 4000 ea3c 0000
```

What can we say about the nature of the traffic?

1: This is a scan for open ports.
2: This is an attempt to contact a Trojan on port 1080.
3: It is normal traffic, just the sequence number happens to be 666666.
4: This is an attempt to look for proxy servers.

Answer 1,4.
The traffic is an attempt to look for an open port where a proxy server is running. It is clear indication of crafted packets, because the sequence number is 666666, however this is not used for a backdoor. If looking into the IDS log, there will likely be seen more proxy server ports (3128, 8080, etc) scanned, with the same type packets.

© SANS Institute 2004, Author retains full rights.

Links and References:

[40, Kovacevich]: Susan Kovacevich, Giac practical, http://www.giac.org/practical/GCIA/Susan_Kovacevich_GCIA.pdf
[41, Zero]: Sans, Ring Zero, http://www.sans.org/newlook/resources/ringzero.ppt
[42, Ktung]: Simon Ktung, Detect #2, Scanning for open proxy , http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00030.html
[43, Zela]: Detect 1, scanning for Proxy, http://cert.uni-stuttgart.de/archive/intrusions/2003/07/msg00132.html
[44, compguruman], security basis mail list, http://lists.insecure.org/lists/security-basics/2003/Jun/0877.html
[45, John], John Choe, security basis mail list , http://seclists.org/lists/security-basics/2003/Jun/0904.html

© SANS Institute 2004, Author retains full rights.

Section 3: Analyze this

This section contains an analyze network traffic from an University. The analyze go into details of 3 kinds of logfiles gathered over a period of 5 days. From this analyze there is a summary.

Summary

This analyze showed that the University were having a larger amount of unwanted activities on their network.

The most obvious seen is a larger amount of scans running on the internal network, as well as from external net. While these scans might not be successful due to prevention mechanism like firewalls, those seen on the internal network are certainly unwanted. While there are good reasons to have scans of your own network from inside security staff, there should not be non-legitimate scans from inside to the inside or Internet/external net. The exact reason to the scan activity seen must be cleared with an inspection of the machines.

Beside the scans, there is some direct communication (or attempt to) between internal and external sources. These being KaZaa (peer to peer) traffic, attempt to get to Trojans horses and Adore traffic. This is mostly worrying, as any success in getting to Trojans can lead to bigger problems in term of leaking information and loss of productivity. The peer-to-peer traffic is also a problem, as it raises an issue of lack of network control and chances of malicious software (with backdoors or viruses) getting directly through the network borders.

Also abnormal traffic is seen, with external addresses on internal network. While this might be a problem, it still leads back to the network borders. A University does have communication with the whole world and direct network between more physical locations.

It seem like the University do have security policies in place, as they have an IDS and log of these. What we do not see is how the network is protected – except where we see answers to bad traffic – and we do not see the policies for communication. The security policies covering the network structure and protection should be expanded to covering the border and standards for what communication can go in and out of the network and how the different locations are connected.

The Analyze Process

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

The network data log used in this specific analyze is taken from http://www.incidents.org/logs/, where files from period 031019 to 031023 have been used. There were alerts, scans and oos files.

Method used:

I started out to look for tools, related to Snort and data drilling. However, a short search and tests, showed, that the tools around needed tailoring or the data needed twisting and cleaning up in order to make them work together. The closest fit I found to the alerts was snortalog, however, it would not work correctly with all the data. A test with own IDS alert files, there were no problems (and a test with a file from earlier in Oct 2003 was ok). So far I could see and judge it was due to some wrong formatted records of the fast mode format.

I ended up using a few scripts from earlier student Chris Baker [50, Chris] – as well as programming some shell, awk and perl scripts myself to sort, extract and make statistic analyze, as well as look for patterns (fit, abnormalities) in the provided data.

The primary reason for me, not to use a database, was the extra time (overhead) for setup and configuration plus test. I wanted to keep the setup simple and efficient, keeping records in files directly. And with a PC and 1.5 GB memory, more files could be in memory at the same time – making a full scan (no index) – possible.

I would not recommend this approach for production purpose, only the scripts for sorting and additional scripts for loading into a database.

Organizing The Data:

OOS files:
OOS files records were folded into one-line records in order to be able to seek and sort at file level. In was not imported into a DB, but searched directly with grep/egrep, awk, sort, perl. The result was split into files of IP and IP+port. The records that contained packets with additional payload were concatenated in a file for later inspection.

Alert files:
Alert files were concatenated into a single file. The file format was keep.

Later this file was split into 2, one for portscan and one for other snort alarms. For external and internal IP addresses, I made a sort list of the top alert generators.

A compiled list of alarms was made to use for correlation and statistic.

© SANS Institute 2004, Author retains full rights.

For easy lookup, the type of alarms were later sorted in src IP directly and src IP with src/dst port.

For reference to rules, the latest ruleset to snort (2.0.2) was stored locally.

Scan files:
Scan files was unzipped and concatenated. After that perl scripts was used to split the records into src, dst, port on file level – in order to see what scan activities had been from which sources at different levels.

Special ports and network traffic:

From files all port 0 activity was filtered out for further investigation.

IP's for network traffic with src and dst both marked as inside or outside were extracted.

Special numbers/signatures in OOS was checked by using `sort`, `awk`, `uniq`.

© SANS Institute 2004, Author retains full rights.

Analyzes:

Alerts and distribution

This table contains data about type of alarm, number of internal and external IP's as src or dst in the alarm. As index there is an [ID, Reference group] like: 9, Virus.

Explanation of colors used:
Where only internal src addresses are seen.
Where only external src addresses are seen.
Where both internal and external src addresses are seen.
Where only external src and dst addresses are seen.

ID, Ref	Name of alarm	Alarms	Src		Dst	
			My Net	Ext	Ny Net	Ext
1, Win	SMB_Name_Wildcard	187366	875	0	2	130094
2, Win	SMB_C_access	28541	0	611	959	0
3, Net	MY.NET.30.4_activity	14002	0	351	1	0

4, Win	EXPLOIT_x86_NOOP	11430	0	1370	935	1
5,Unix	connect_to_515_from_inside	6523	1	0	0	1
6, Net	TCP_SRC_and_DST_outside_network	4468	0	22	0	86
7, Net	MY.NET.30.3_activity	4373	0	95	1	0
8, Net	External_RPC_call	3269	0	4	1831	0
9, Virus	High_port_65535_tcp_-_possible_Red_Worm_-_traffic	3119	37	47	39	57
10, Virus	Possible_trojan_server_activity	1856	24	51	264	45
11, Net	ICMP_SRC_and_DST_outside_network	1548	0	84	0	1358
12, Net	NMAP_TCP_ping!	610	0	132	52	0
13, Port	SUNRPC_highport_access!	473	0	19	24	0
14, Net	Null_scan!	426	0	54	52	0
15, Virus	High_port_65535_udp_-_possible_Red_Worm_-_traffic	313	11	48	36	25
16, IRC	[UMBC_NIDS_IRC_Alert]_XDCC_client_detected_attempting_to_IRC	182	4	0	0	3
17, NET	FTP_passwd_attempt	99	0	31	6	0
18, NET	[UMBC_NIDS]_External_MiMail_alert	88	0	45	1	0
19, Virus	Back_Orifice	84	0	2	84	0
20, TFTP	TFTP_-_Internal_UDP_connection_to_external_tftp_server	81	6	11	11	15
21, NET	Incomplete_Packet_Fragments_Discarded	69	0	52	44	0
22, NET	Tiny_Fragments_-_Possible_Hostile_Activity	59	1	36	23	1
23, IRC	[UMBC_NIDS_IRC_Alert]_Possible_sdbot_floodnet_detected_attempting_to_IRC	55	7	0	0	2
24, WIN	NETBIOS_NT_NULL_session	51	0	3	12	0
25, WIN	EXPLOIT_x86_stealth_noop	46	0	9	6	0

© SANS Institute 2004, Author retains full rights.

26, NET	DDOS_shaft_client_to_handler	38	0	5	2	0
27, IRC	[UMBC_NIDS_IRC_Alert]_Possible_drone_command_detected.	37	0	2	6	0
28, WIN	EXPLOIT_x86_setgid_0	24	0	20	20	0
29, NET	EXPLOIT_NTPDX_buffer_overflow	22	0	8	12	0
30, WIN	EXPLOIT_x86_setuid_0	18	0	17	12	0
31, IRC	[UMBC_NIDS_IRC_Alert]_Possible_Incoming_XDCC_Send_Request_Detected.	12	0	2	1	0
32, TFPT	TFTP_-_External_UDP_connection_to_internal_tftp_server	11	0	7	7	0
33, NET	FTP_DoS_ftpd_globbing	10	0	3	1	0
34, NET	Attempted_Sun_RPC_high_port_access	9	0	3	4	0
35, WIN	RFB_-_Possible_WinVNC_-_010708-1	8	3	3	3	2
36, Help	HelpDesk_MY.NET.70.49_to_External_FTP	5	1	0	0	3
37, IRC	[UMBC_NIDS_IRC_Alert]_K\:line'd_user_detected,_possible_trojan.	4	0	2	2	0
38, Virus	NIMDA_-_Attempt_to_execute_cmd_from_campus_host	4	4	0	0	3
39, Virus	[UMBC_NIDS]_Internal_MSBlast_Infection_Request	3	2	0	0	3
40, TFTP	TFTP_-_Internal_TCP_connection_to_external_tftp_server	2	0	1	1	0
41, TFTP	TFTP_-_External_TCP_connection_to_internal_tftp_server	2	1	1	1	1
42, Help	External_FTP_to_HelpDesk_MY.NET.70.50	2	0	2	1	0
43, Unix	connect_to_515_from_outside	2	0	1	2	0
44, Virus	[UMBC_NIDS_IRC_Alert]_Possible_trojaned_box_detected_attempting_to_IRC	1	1	0	0	1
45, Help	External_FTP_to_HelpDesk_MY.NET.70.49	1	0	1	1	0
46, Help	External_FTP_to_HelpDesk_MY.NET.53.29	1	0	1	1	0
47, Net	DDOS_mstream_client_to_handler	1	0	1	1	0

© SANS Institute 2004, Author retains full rights.

48, IRC	IRC_evil_-_running_XDCC	1	1	0	0	1
49, Net	Probable_NMAP_fingerprint_attempt	1	0	1	1	0
50, Virus	Bugbear@MM_virus_in_SMTP	1	0	1	1	0

Top Scans MY.NET and external sources

This table shows the top scan alarms.

Explanation of colors used:
A number of alarms are distributed on a large number of IPs.
A large number of alarms are distributed on a few Ips
Some of the alarms come from traffic to/from port 0

Explanation of terms:
* port: Evenly distributed (or close to) a number of ports.

ID, Ref	IP	Alarms	Dst	Src Ports(N)	Dst Port(N)

1, scan	MY.NET.1.3	2166939	85813	62206(2155488)
				123(8785)
					53 UDP(2155488)
					53 SYN(40)
					123 UDP(26)
					* UDP

2, scan	MY.NET.70.154	1294189	285691	SYN on * port 137(16)	135 SYN(1102408) 80 SYN(191763)
3, scan	MY.NET.163.107	966595	966532	SYN on * port	135 SYN(966568) 80 SYN(18)
4, scan	MY.NET.84.194	888186	884153	SYN on * port	135 SYN(886352) 8000 UDP(1175)
5, scan	MY.NET.163.249	669973	586885	SYN on * port + udp traffic	135 SYN(591537) 445 SYN(76973)
6, scan	MY.NET.42.1	273705	99301	SYN on * port	135 SYN(243093) 80 SYN(30593)
7, scan	MY.NET.70.129	213577	85732	SYN on * port	135 SYN(184556) 80 SYN(29020)

8, scan	MY.NET.1.5	211571	21215	38361(210445)
				123(632)
					53 UDP(210445)
					123 UDP(509)

9, scan	MY.NET.80.149	175960	92534	SYN on * port	135 SYN(93240) 445 SYN(61875) 80 SYN(20818)
10, scan	MY.NET.111.72	171526	171525	SYN on * port	135 SYN(171514) 80 SYN(12)
11, scan	218.94.41.98	74607	9901	SYN on * port	SYN on * port, equal distributed (9 ports)

12, scan	213.180.193.68	30239	1	50458(16533)
				50457(13706)
					SYN on * port

13, scan	63.250.195.10	27734	23	0(10994)
				4736(1953)
					0 UDP(10941)

14, scan	193.114.70.169	19164	1322	137(903) SYN on * port	(16471 139 SYN (1058 1433 SYN 137 UDP(902)

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

					111 SYN(732)

15, scan	68.85.216.188	18246	1	42188(17989)
				42189(184)
					SYN on * port

16, scan	213.51.194.191	16244	12081	SYN on * port	SYN on 3389 port
17, scan	200.168.78.213	15688	2	SYN on * port	443 SYN(15593) 80 SYN(95)
18, scan	219.121.66.87	14827	10016	SYN on * port	445 SYN(14827)
19, scan	217.158.99.5	14548	1	SYN on * port	1433 SYN(14548)
20, scan	165.123.179.206	14452	1	SYN on * port	80 SYN(14452)

Analyzes

OOS traffic analyze

Most of the data from the OOS files showed only few details. I did a breakdown of the data into source, destination and what protocols was used – along with the TTL and Flags seen. There was only a little or no difference in the flags used in all of the files.

IP	Destinations	Dst Port Protocol	TTL?	Flags
217.174.98.145	MY.NET.111.52	25,TCP	39-40	DF,12****S*
195.111.1.93	MY.NET.100.165 MY.NET.162.87 MY.NET.24.34 MY.NET.24.35 MY.NET.24.44 MY.NET.6.7 MY.NET.60.14	80,TCP	50	DF,12****S*
158.196.149.61	MY.NET.111.52	25,TCP	50,52	DF,12****S*
212.16.0.33	MY.NET.111.52 MY.NET.12.6	25,TCP	42,44,47	DF,12****S*
194.67.62.194	MY.NET.111.52	25,TCP	44-45	DF,12****S*
82.82.64.209	MY.NET.69.181	8887,TCP	44,50	DF,12****S*
213.23.46.99	MY.NET.69.181	8887,TCP	46,49,50	DF,12****S*
66.225.198.20	MY.NET.12.6	25,TCP	48,51	DF,12****S*
195.208.238.143	MY.NET.25.71 MY.NET.111.52	113,TCP 25,TCP	48	DF,12****S*
195.14.47.202	MY.NET.111.52	25,TCP	43,44	DF,12****S*

It is interesting, that port 25 (smtp) get listed several times. When a mail system uses the ECN bit, it has been known to give problems with firewalls that do not respect the ECN bits. Also there have been known issues with Linux kernel 2.4 and access to web servers on networks who were blocking ECN: See Linux kernel mailing list [55, Lkml], http://lkml.org/faq/lkmlfaq-14.html. A check of the source addresses from the table showed that some was mail systems or gateways. I assume that most of this is false alarms from the IDS.

From the 5 days, there were a few packets logged with data. From one of these packets we see:

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Day 23 October: KaZaa Traffic:

The following network dump was found in the OOS file from 23 October.

```
10/23-21:41:22.868546 148.63.207.124:4597 -> MY.NET.83.109:3264
TCP TTL:114 TOS:0x0 ID:50681 IpLen:20 DgmLen:443 DF
****P*** Seq: 0x10B3680A  Ack: 0x0  Win: 0x2000  TcpLen: 20
47 45 54 20 2F 2E 68 61 73 68 3D 36 39 31 61 66  GET /.hash=691af
63 35 61 30 36 33 37 32 36 38 32 35 61 37 32 31  c5a063726825a721
37 37 62 63 32 36 30 66 36 30 30 61 33 36 66 66  77bc260f600a36ff
30 34 33 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F  043 HTTP/1.1..Ho
73 74 3A 20 31 33 30 2E 38 35 2E 38 33 2E 31 30  st: MY.NET.83.10
39 3A 33 32 36 34 0D 0A 55 73 65 72 41 67 65 6E  9:3264..UserAgen
74 3A 20 4B 61 7A 61 61 43 6C 69 65 6E 74 20 4A  t: KazaaClient J
75 6C 20 20 36 20 32 30 30 33 20 31 37 3A 35 34  ul  6 2003 17:54
3A 31 33 0D 0A 58 2D 4B 61 7A 61 61 2D 55 73 65  :13..X-Kazaa-Use
72 6E 61 6D 65 3A 20 XX XX XX XX XX XX XX 0D 0A  rname: xxxxxxx..
58 2D 4B 61 7A 61 61 2D 4E 65 74 77 6F 72 6B 3A  X-Kazaa-Network:
20 4B 61 5A 61 41 0D 0A 58 2D 4B 61 7A 61 61 2D   KaZaA..X-Kazaa-
49 50 3A 20 31 34 38 2E 36 33 2E 32 30 37 2E 31  IP: 148.63.207.1
32 34 3A 33 30 33 30 0D 0A 58 2D 4B 61 7A 61 61  24:3030..X-Kazaa
2D 53 75 70 65 72 6E 6F 64 65 49 50 3A 20 31 34  -SupernodeIP: 14
30 2E 31 39 32 2E 31 37 37 2E 31 33 36 3A 33 33  0.192.177.136:33
33 34 0D 0A 52 61 6E 67 65 3A 20 62 79 74 65 73  34..Range: bytes
3D 33 38 37 30 35 39 37 31 32 2D 34 30 32 36 35  =387059712-40265
33 31 38 33 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E  3183..Connection
3A 20 63 6C 6F 73 65 0D 0A 58 2D 4B 61 7A 61 61  : close..X-Kazaa
2D 58 66 65 72 49 64 3A 20 34 32 36 35 34 32 31  -XferId: 4265421
0D 0A 58 2D 4B 61 7A 61 61 2D 58 66 65 72 55 69  ..X-Kazaa-XferUi
64 3A 20 71 67 49 39 33 4D 61 6C 77 56 6E 2B 6D  d: qgI93MalwVn+m
37 36 36 49 6E 41 68 44 38 65 47 6C 35 32 4F 69  766InAhD8eGl52Oi
48 4C 30 43 63 35 6E 43 74 33 63 78 72 67 3D 0D  HL0Cc5nCt3cxrg=.
0A 0D 0A                                         ...
```

This is Look like KaZaa (peer to peer) traffic information exchange. To allow this kind of traffic flowing directly between machines in Internet and machines at the University is a risk.

Correlation:

Tyler Hudak [56, Hudak], in the GCIA practical assignment makes observations of how KaZaa traffic can be a target of an attack (detect #1) and as part of the University network traffic ("Analyze this"). See: http://www.giac.org/practical/GCIA/Tyler_Hudak_GCIA.pdf

Analyze of Alerts

Alert: 1	Alarms: 187366	SMB_Name_Wildcard	Priority: Medium

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Origin: This type alarm is seen when there is a query for a machines NetBios name table.

Analyze

The main part of the alarms was generated from

IP	Number of Alarms	To IP's	Ports
MY.NET.80.51	115761	115627	From 1035 and 1036
MY.NET.150.133	61719	11515	From several ports
MY.NET.29.2	3101	2147	From 137
MY.NET.150.198	458	227	From 4 ports

Alarms:

MY.NET.80.51: The number of IP and Alarms is almost 1 to 1 and the src port is limited to 1035 and 1036. This amount of alarms is from only 1 day, the 23 October. I assume that MY.NET.80.51 is doing scanning.

MY.NET.150.133: This source shows a similar pattern, however is using more ports to communicated more than once with each destination. The scan had taken place during all 5 days.

MY.NET.29.2 also have a similar pattern, but less active. However, like MY.NET.80.51, the alarms are from only the 23 October until morning, and then a single alarm in the afternoon.

From the timestamp, it looks like MY.NET.80.51 “took over” when MY.NET.29.2 stopped the activity. Interesting.

MY.NET.150.198 generated the 458 alarms over all 5 days. If this is a scan, it looks like a slow scan to me.

Correlation: None

Recommendation

I would recommend an investigation of the activity on these systems. If infected with a virus, clean it – if used from outside via backdoors, close them and follow up on how this started and make updates to the security policies in place.

Notes

Alert: 2	Alarms: 28541	SMB_C_access	Priority: Medium

© SANS Institute 2004, Author retains full rights.

Origin: Snort have some standard rules to look for access to drive C$ and other default share name son Windows machines. See: http://www.snort.org/snort-db/sid.html?sid=533

Analyze
The top address were: 80.50.168.42
The destinations were for all the top IP's spread over a number of C segments, typical MY.NET.A/24 to MY.NET.A+(3-4)/24.
The destination from the top 10 IP's were to MY.NET.69-112+150-152+180-153. I assume thereby that the upper end 69-255 of the MY.NET subnets contains Windows PC and servers. A close look at the other Alerts showed lower subnets as well.
Correlation: None
Recommendation
Close all unnecessary access on outer and inner boundary. Access to services should be on a need to use basis. I cannot see from the alerts if the traffic do get through from outside or if it only is noise from the outer sensor.
Notes

Alert: 3	Alarms: 14002	MY.NET.30.4_activity	Priority: Low

Origin: This is a type of Alarm from the IDS that is machine specific. Simple: there is some kind of activity, that needs to be checked.

Analyze
A closer look the traffic from outside to MY.NET.30.4, showed traffic on mainly 4 ports: 80,443,524,51443 – fairly equal distributed. Top external IP was: 68.55.85.180, with 68.54.91.147 close behind.
The port 80,443 is likely to be http and https traffic.
From the Internet a search showed that port 524 is a know port for NCP (Novell) and 51443 is a secure port for Novell secure web server.

Therefore it is likely MY.NET.30.4 is a Novell server for external users. The analyze traffic didn't show any abnormalities.

IP: 68.55.85.180	*pcp243940pcs.howard01.md.comcast.net*
Name:	*Comcast Cable Communications, In. Baltimore*
Range:	*68.55.0.0 - 68.55.255.255*
Contact:	*?*
Notes:	*The other IP also showed to be from the same company*

I would assume that there are some strong relations between the University and the IP's that use the Novell server and it secure web services.
Correlation: I did not find any.
Recommendation: Configure the IDS to ignore the traffic.

© SANS Institute 2004, Author retains full rights

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Notes http://novell2.net-burg.com/Programms/Novell/EGuide/2.0/

Alert: 4	Alarms: 11430	EXPLOIT_x86_NOOP	Priority: Low

Origin: This type alarm come when a pattern for certain x86 processor code (NOPs) is seen. See: http://www.snort.org/snort-db/sid.html?sid=648. The rules often generate many false alarms.

Analyze
The Analyze showed that most alarms where generated my traffic to port 135, 445, 80 in that order. All the traffic where from external sources. The trigger to this alarm is certain hex codes (x86 code), normally used for buffer overflow. Unfortunate if you look at all traffic, this rule will generate many false alarms, therefore priority: low.

The Snort configuration files I have seen, usually have set SHELLCODE to !80 (not http), however, it seem that the University have configured their snort sensor(s) to look on all ports.
Correlation: Terry McDonals [51,Terry], make similar conclusion regarding the setup of the IDS at the University (3 months back). He also mentions that there are plans (or have been) for an improved preprocessor “spp_fnord” to Snort
Recommendation: None
Notes: The changelog from Snort mention the spp_fnord preprocessor at 2002-03-11.

Alert: 5	Alarms: 6523	connect_to_515_from_inside	Priority: High

© SANS Institute 2004, Author retains full rights.

Origin: Port 515 is often used by the Unix printer demon. If there is an exploit, command can be executed through the demon. See: http://www.snort.org/snort-db/sid.html?sid=301

Analyze
All the alarms showed the following, [**] connect to 515 from inside [**] MY.NET.162.41:721 -> 128.183.110.242:515. The traffic continued all through the period of the 5 days.

An analyze of the destination shows:

IP:128.183.110.242	tek924.gsfc.nasa.gov
Name:	National Aeronautics and Space Administration
Range:	128.183.0.0 - 128.183.255.255
Contact:	TechPhone: +1-256-544-5623 TechEmail: dns.support@nasa.gov OrgAbusePhone: +1-800-762-7472 OrgAbuseEmail: abuse@nasa.gov
Notes:	

Here it looks like a lot of traffic is going to this port. It might be legal, as in print, or a service that just happen to run on that port.
Correlation: None. I did find a number of analyzes for traffic from outside to inside on port 515, but not the other way around.
Recommendation
Investigate the traffic that goes to this destination. If legal traffic, configure the sensors to ignore this traffic.
Notes

Alert: 6	Alarms: 4468	TCP_SRC_and_DST_outside_network	Priority: Medium

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Origin: This type alarm is an indication of invalid network traffic, which should not be seen. It can be a configuration error – or malicious traffic.

Analyze
During this analyze, I came across some traffic marked “TCP_SRC_and_DST_outside_network” [6,net].

While there were a number of IP’s that did give this alarm, the most frequently seen alarm was:

```
10/19-14:12:17.141125  [**] TCP SRC and DST outside network [**]
169.254.244.56:2678 -> 218.16.124.131:21
10/19-14:12:20.368298  [**] TCP SRC and DST outside network [**]
169.254.244.56:2679 -> 211.91.144.72:996
```

Each of these alarms could be seen many times, just with different time and src port.
Access to port 21 was seen 2852 times. Access to port 996 was seen 1420 times.

IP	IP Range	Description
169.254.244.56	169.254.0.0 - 169.254.255.255	Internet Assigned Numbers Authority
218.16.124.131	218.13.0.0 - 218.18.255.255	CHINANET Guangdong province network
211.91.144.72	211.90.0.0 - 211.91.255.255	China United Telecommunications Corporation

I did not succeed in finding anything useful about traffic to port 996. The official port said Xtree license server. Port 21 list to FTP.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Beside the above I found:

```
10/23-22:45:08.789634  [**] TCP SRC and DST outside network [**]
192.168.0.4:3003 -> 209.10.203.102:6301
10/22-18:50:37.241960  [**] TCP SRC and DST outside network [**]
10.0.1.12:49289 -> 68.55.61.253:143
```

To me it looked like one of the NIDS sensors was placed on a network there private address ranges could be seen. A reason to this is a sensor placed on a network segment behind a NAT device – or – in front of a NAT device where the device pass non-natted addresses due to a misconfiguration. The network addresses 192.168/16 and 10/8 get seen as external because these networks are not included in the configuration as internal in the configuration of the sensor.

Based on the observation seen, I assume that the IDS at the University have more sensors, 1 or more on the boundary of the network, and one or more placed on internal network segments.
A search in the OOS and scan files did not show additional info about traffic from/to these IP's. There was some scan traffic from an IP in the same range as 211.91.144.72. The traffic was:

```
Oct 21 15:55:31 211.91.255.148:7342 -> MY.NET.190.212:135 SYN ******S*
Oct 21 15:55:31 211.91.255.148:7343 -> MY.NET.190.213:135 SYN ******S*
Oct 21 15:55:31 211.91.255.148:7344 -> MY.NET.190.214:135 SYN ******S*
```

SYN to a number of IP on the same Network segment. However, I could not see any connection to the traffic above.
Correlation: Terry McDonalds [51, Terry], do the analyze that a rule like that might be an old rule from the past for seeing address spoofing – and now it generate false alarms.
Recommendation: Investigate why this traffic is seen! I would keep the rule as it will trigger if something change in the network and Snort is not updated. It might be optimized for amount of alarms is needed.
Notes

Alert: 7	Alarms: 4373	MY.NET.30.3_activity	Priority: Low

Origin: This is a type of Alarm from the IDS that is machine specific. It states that there is some kind of network activity, which needs to be checked.

Analyze
Most of the traffic seen to this IP was to port 524.
Furthermore, from the 95 external IP addresses, 50 of these were also seen to MY.NET.30.4. Based on analyze 3, and the close IP sequence between the two machines, I will assume this is a Novell server. However, only with service running on port 524.
Correlation: I did not find any.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Recommendation: Configure the sensors to ignore this traffic.
Notes

Alert: 8	Alarms: 3269	External_RPC_call	Priority: Low

Origin: There exist a number of exploits for services related to RPC call. It look to me that the IDS here have a general rule for logging

Analyze
IP: 193.114.70.169 generated 2840 alerts and was trying to reach 1595 IP's.

The alarms looks like:

```
10/23-04:36:22.581114  [**] External RPC call [**] 193.114.70.169:2927
-> MY.NET.1.0:111
10/23-04:36:23.124143  [**] External RPC call [**] 193.114.70.169:2933
-> MY.NET.1.5:111
10/23-04:36:23.455707  [**] External RPC call [**] 193.114.70.169:2930
-> MY.NET.1.2:111
```

And were going through a number of subnets in MY.NET.

IP check on 193.114.70.169

IP:193.114.70.169	No official name found
Name:	FIRST PROCUREMENT ASSOCIATES LIMITED
Range:	193.114.70.160 - 193.114.70.191
Contact:	sysadmin@uk.psi.com
Notes:	

Correlation: None
Recommendation: If possible, block the port on network border
Notes

Alert: 9	Alarms: 3119	High_port_65535_tcp_-_possible_Red_Worm_-_traffic	Priority: High

© SANS Institute 2004, Author retains full rights.

Origin: This type alarm is generated when traffic to/from port 65535 is seen and can be a sign or Red Worm or Adora traffic. See [52, F-secure].

Analyze

Most of the alarms seemed to be false. However, 2 IP's from MY.NET stood out.
MY.NET.80.105:3951 send 1114 packet to 200.96.13.157:65535 for around 1 hour.
MY.NET.153.141:2071 send 309 packets on port 2071 to 66.66.71.92:65535 for around 15 minutes.

A close look for the first IP in the alarms log shows:

```
10/23-09:42:42.055370  [**] High port 65535 udp - possible Red Worm -
traffic [**] 66.66.71.92:65535 -> MY.NET.80.105:2740
10/23-09:42:42.055644  [**] High port 65535 udp - possible Red Worm -
traffic [**] MY.NET.80.105:2740 -> 66.66.71.92:65535
10/23-09:53:00.291182  [**] NMAP TCP ping! [**] 194.78.59.253:80 ->
MY.NET.80.105:2740
10/23-11:05:35.082654  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 200.96.13.157:65535 -> MY.NET.80.105:3951
10/23-11:05:35.082834  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.80.105:3951 -> 200.96.13.157:65535
```

The OOS showed the following:

```
10/23-16:15:04.633102 80.143.11.192:53335 -> MY.NET.80.105:2740
10/23-16:18:58.961766 212.21.255.78:55191 -> MY.NET.80.105:2740
10/23-16:20:40.761366 217.230.14.202:3704 -> MY.NET.80.105:2740
10/23-18:33:27.420929 81.56.214.240:36474 -> MY.NET.80.105:2740
10/23-20:41:58.338297 24.43.50.231:36521 -> MY.NET.80.105:2740
10/23-21:33:07.432159 203.218.221.69:33456 -> MY.NET.80.105:2740
```

It looks like the system on MY.NET.80.105 is in a compromised state. During the first days log we see only outbound scans, but later on at 23 October, active traffic on high port 65535. And later on a number of systems start to have a communication going to this machine.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

For the other IP we see the following from the log on October 23:

```
10/22-18:55:23.800403  [**] EXPLOIT x86 NOOP [**] 212.81.218.50:1077 ->
MY.NET.153.141:135
10/22-22:24:16.827253  [**] SMB C access [**] 66.72.208.69:3285 ->
MY.NET.153.141:139
10/23-09:00:36.025540  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.153.141:2071 -> 66.66.71.92:65535
```

And after a while, a portscan start:

```
10/23-10:01:58.498699  [**] spp_portscan: PORTSCAN DETECTED from
MY.NET.153.141 (THRESHOLD 12 connections exceeded in 0 seconds) [**]
```

There are actually many records, these here are just the pattern seen before the first high port alert.

Here the alert come around 42 minutes before the alert for MY.NET.80.135.

The Alert log shows a number of alerts for high port traffic during the whole period; however, these two IP's was standing out, as they had the highest number of records in form of port scans and high port traffic!
Correlation: Susan Kovacevich [40, Kovacevich], do a detailed analyze of how Adora traffic get spread through the University network.
Recommendation: Inspect the machine and take action!! Keep logging traffic on port 65535 as more machines might be infected.
Notes: See also Sans [53, Sans] for information about Adore worm

Alert: 10	Alarms: 1856	Possible_trojan_server_activity	Priority: High
Origin: Some of the related Trojans to port 27374 is Subseven 2.X or Ramen worm. See: [54, Trojans], http://www.snort.org/snort-db/sid.html?sid=103, http://www.snort.org/snort-db/sid.html?sid=514			

© SANS Institute 2004, Author retains full rights.

Analyze:

Active response:

554 alarms where seen from 200.163.61.175, early night 23 October to 1 internal IP. The alarms were like:

```
10/23-01:45:02.037407  [**] Possible trojan server activity [**]
200.163.61.175:27374 -> MY.NET.163.249:6667
```

And in the same period, we see

```
10/23-01:45:11.036163  [**] Possible trojan server activity [**]
MY.NET.163.249:6667 -> 200.163.61.175:27374
```

And a little after we see:

```
10/23-05:42:55.278029  [**] Possible trojan server activity [**]
MY.NET.163.249:6667 -> 200.203.68.71:27374
```

Scans:

From more IP's (like 66.169.146.100 and 212.95.105.31) we see, the following scan to the whole MY.NET.190/24

```
10/19-03:16:17.264221  [**] Possible trojan server activity [**]
66.169.146.100:3660 -> MY.NET.190.1:27374
10/19-19:38:01.604153  [**] Possible trojan server activity [**]
212.95.105.31:3960 -> MY.NET.190.2:27374
```

IP: 200.163.61.175	200-163-061-175.cbabm7004.e.brasiltelecom.net.br
Name:	Brasil Telecom S.A.
Range:	200.163/16
Contact:	abuse@NOC.BRASILTELECOM.NET.BR
Notes:	

Correlation: For a related view on Subseven and Trojan activity see Tu Niem[57, Niem], http://www.giac.org/practical/GCIA/Tu_Niem_GCIA.pdf

Recommendation: Check the machines for Trojans. Check those IP's in the alert log, as well as other machines in subnet MY.NET.190/24. Depend on the result, plan and keep an eye on further related activity. Make sure Windows machines have anti-virus software

Notes see also: http://www.sans.org/resources/idfaq/subseven.php

© SANS Institute 2004, Author retains full rights.

Analyze of scan traffic

Scan: 1	Alarms: 2166939	MY.NET.1.3	Priority: Low

Analyze

Most alarms looked like this:

```
Oct 19 00:07:34 MY.NET.1.3:62206 -> 210.49.20.208:53 UDP
```

The amount of alarms was 3-7/sec. The sources IP’s were 85813.

A check of the destination IP’s, showed that these (all the ones I tested) were name servers.

I assume that this Is legal traffic, however many DNS queries coming from the same port (port 62206)?

www.dshield.org shows no special activity around this date, however, more report with activity to/from port 62206 have been reported the latest dates (Nov 13-20)

Correlation: None

Recommendation

The legal DNS traffic should not generate alarm. A proper configuration of the IDS systems that generate the scan alarms.

Notes

Scan: 2	Alarms: 1294189	MY.NET.70.154	Priority: Medium

Analyze

The destination ports were: 191763 destinations with port 80 SYN and 1102408 destinations with port 135 SYN

The total numbers of destinations were 285691.

```
Typical alarms was like:
Oct 19 07:10:16 MY.NET.70.154:1929 -> 220.124.70.43:80 SYN ******S*
Oct 19 07:10:16 MY.NET.70.154:1937 -> 202.63.102.76:80 SYN ******S*
```

And

```
Oct 19 07:49:38 MY.NET.70.154:2754 -> 130.87.133.82:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2755 -> 130.87.133.84:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2756 -> 130.87.133.88:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2757 -> 130.87.133.87:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2689 -> 130.87.132.75:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2690 -> 130.87.132.83:135 SYN ******S*
Oct 19 07:49:38 MY.NET.70.154:2691 -> 130.87.132.84:135 SYN ******S*
```

It is a scan to find services on port 80 and 135.

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Correlation: None
Recommendation
Find the machines that do the scans. Check it for virus and Trojans.
Notes

Scan: 3	Alarms: 966595	MY.NET.163.107	Priority: Medium

Analyze
The destination ports were 135 with 966532 destinations. It is a straight forward scan to find machines with something running on port 135, like Microsoft services.

Typical alarms was like:

```
Oct 19 00:24:14 MY.NET.163.107:2486 -> 131.109.33.223:135 SYN ******S*
Oct 19 00:24:14 MY.NET.163.107:2487 -> 131.109.33.224:135 SYN ******S*
Oct 19 00:24:14 MY.NET.163.107:2488 -> 131.109.33.225:135 SYN ******S*
```

Correlation: None
Recommendation
Find the machines that do the scans. Check it for virus and Trojans.
Notes:
Scans seen from whole period Oct 19-23.

For scan 4,5,6,7,9,10 we see similar scans: therefore combined.

Scan:	Alarms:		Priority: Medium
4	888186	MY.NET.84.194	
5	669973	MY.NET.163.249	
6	273705	MY.NET.42.1	
7	213577	MY.NET.70.129	
9	175960	MY.NET.80.149	
10	171526	MY.NET.111.72	

Is this scanning for services on Microsoft machines?

All of the IP's are having the same scan pattern for port 135 as the IP in scan 3. Additional, scan 9 did scan for port 445 as well.
Correlation: None
Recommendation
Find the machines that do the scan. Check it for virus how it got there.
Notes
Scan 4 started: Oct 20 19:45:26 (End not seen)
Scan 5 started: Oct 20 19:45:26 (End not seen)
Scan 6 started: Oct 22 19:08, Ended Oct 23 10:22
Scan 7 started: Oct 19 00:08, Ended Oct 23 23:47 (?)
Scan 9 started: Oct 22 19:37 (End not seen)
Scan 10 started: Oct 20 14:37, Ended Oct 21 12:08

Scan:11	Alarms: 74607	218.94.41.98	Priority: Medium

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Analyze
The IP 218.94.41.98 make a syn on different ports.
One of the syn sequences look like:

```
Oct 22 06:10:56 218.94.41.98:4017 -> MY.NET.1.19:9090 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4018 -> MY.NET.1.19:8888 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4019 -> MY.NET.1.19:2000 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4020 -> MY.NET.1.19:80 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4021 -> MY.NET.1.19:8000 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4022 -> MY.NET.1.19:8080 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4023 -> MY.NET.1.19:3128 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4024 -> MY.NET.1.19:4128 SYN ******S*
Oct 22 06:10:56 218.94.41.98:4025 -> MY.NET.1.19:5128 SYN ******S*
```

The same take place against other destinations. With these 9 destination ports or less.
The ports look like web, proxy or Trojan related scan.
On http://isc.incidents.org/port_details.html?port=2000, for TCP, port 2000 list to a number of Trojans.
Port 8888 is likely a scan for Solaris Answerbook.
Port 3128 is to find a Squid proxy.
Correlation: None
Recommendation: Make sure the network at the border, block unwanted inbound traffic.
Notes
See [54, Trojans]http://www.simovits.com/trojans/trojans.html for a list of Trojans.

Scan:12	Alarms: 30239	213.180.193.68	Priority: Medium

© SANS Institute 2004, Author retains full rights.

Analyze
The traffic seen is only to one IP, starting on Oct 23 in the night and running for a short while:
First entry:

```
Oct 23 01:30:04 213.180.193.68:50458 -> MY.NET.15.27:50926 SYN ******S*
```

Last entry:

```
Oct 23 02:41:42 213.180.193.68:50458 -> MY.NET.15.27:15003 SYN ******S*
```

And src port only 50457 and 50458.

The OOS files did not show any entry for these IP's.

IP:213.180.193.68	*proxychecker.yandex.net*
Name:	*COMPTEK-NET1*
Range:	*213.180.192.0 - 213.180.193.255*
Parent Range:	
Contact:	CompTek International *3, Gubkina str., Moscow, 117809* abuse@yandex.ru
Last change:	*Date*
Notes:	*Notes*

I did not find any information on the port.
Correlation: None
Recommendation: Check this machine (MY.NET.15.27) for suspicious activity.
Notes

Scan:13	Alarms: 27734	63.250.195.10	Priority: Medium

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

Analyze
This network traffic pattern is unique, as it go through a number of IP's, with a number of various UDP port. A closer look on the ports, showed the following top 0 port:

```
   1227 MY.NET.153.153:0
   1196 MY.NET.153.30:0
   1072 MY.NET.152.175:0
```

A sample of the traffic to the first IP shows:

```
Oct 19 20:50:36 63.250.195.10:0 -> MY.NET.153.153:0 UDP
Oct 19 20:50:35 63.250.195.10:40218 -> MY.NET.153.153:1711 UDP
Oct 19 20:50:33 63.250.195.10:4736 -> MY.NET.153.153:2461 UDP
Oct 19 20:50:34 63.250.195.10:4698 -> MY.NET.153.153:41044 UDP
Oct 19 20:50:34 63.250.195.10:24574 -> MY.NET.153.153:61870 UDP
Oct 19 20:50:34 63.250.195.10:28897 -> MY.NET.153.153:17984 UDP
Oct 19 20:50:35 63.250.195.10:52877 -> MY.NET.153.153:56626 UDP
Oct 19 20:50:37 63.250.195.10:40218 -> MY.NET.153.153:1711 UDP
Oct 19 20:50:37 63.250.195.10:0 -> MY.NET.153.153:0 UDP
Oct 19 20:50:37 63.250.195.10:4736 -> MY.NET.153.153:2461 UDP
Oct 19 20:50:39 63.250.195.10:0 -> MY.NET.153.153:0 UDP
Oct 19 20:50:39 63.250.195.10:40218 -> MY.NET.153.153:1711 UDP
Oct 19 20:50:38 63.250.195.10:23401 -> MY.NET.153.153:33375 UDP
Oct 19 20:50:39 63.250.195.10:41112 -> MY.NET.153.153:53696 UDP
Oct 19 20:50:41 63.250.195.10:4736 -> MY.NET.153.153:2461 UDP
Oct 19 20:50:41 63.250.195.10:0 -> MY.NET.153.153:0 UDP
```

The info about the IP list:

IP:63.250.195.10	*l8.cache.vip.dal.yahoo.com*
Name:	*Yahoo! Broadcast Services, Inc*
Range:	*63.250.192.0 - 63.250.223.255*
Parent Range:	*NET-63-0-0-0-0*
Contact:	*netblockadmin@yahoo-inc.com*
Last change:	*2003-05-06*
Notes:	*Notes*

From the above and the official name of the IP, it looks like this is some kind of broadcast service via UDP – and not a scan. I did try to find further information about their broadcast service, but ran info some non-working links.
Correlation: Did not find any.
Recommendation: If legal traffic, configure the IDS to ignore the traffic.
Notes

Scan:14	Alarms: 19164	193.114.70.169	Priority: Medium

© SANS Institute 2004, Author retains full rights.

Analyze

The IP, 193.114.70.169, was scanning a number of IP.
A sample of the log shows:

```
Oct 23 04:40:34 193.114.70.169:137 -> MY.NET.5.25:137 UDP
Oct 23 04:40:34 193.114.70.169:137 -> MY.NET.5.23:137 UDP
Oct 23 04:40:35 193.114.70.169:137 -> MY.NET.5.95:137 UDP
Oct 23 04:40:35 193.114.70.169:4243 -> MY.NET.5.17:139 SYN ******S*
Oct 23 04:40:35 193.114.70.169:4244 -> MY.NET.5.26:139 SYN ******S*
Oct 23 04:40:36 193.114.70.169:4252 -> MY.NET.5.17:139 SYN ******S*
Oct 23 04:40:37 193.114.70.169:4264 -> MY.NET.5.26:139 SYN ******S*
Oct 23 04:40:36 193.114.70.169:4255 -> MY.NET.5.24:139 SYN ******S*
Oct 23 04:40:36 193.114.70.169:4256 -> MY.NET.5.34:139 SYN ******S*
Oct 23 04:40:36 193.114.70.169:137 -> MY.NET.5.95:137 UDP
```

The scan ran from Oct 23 04:40:00 to Oct 23 10:15:02. Looking into the details, there are not any specific IP that get a lot more hit than others. However, 97 syn or udp packets were send to MY.NET.80.148 in just 4 minutes.
Correlation: None
Recommendation: Make sure the network at the border block unwanted traffic.
Notes

Scan:15	Alarms: 18246	68.85.216.188	Priority: Medium

Analyze

This scan was directed toward one IP. A sample of the packets look like:

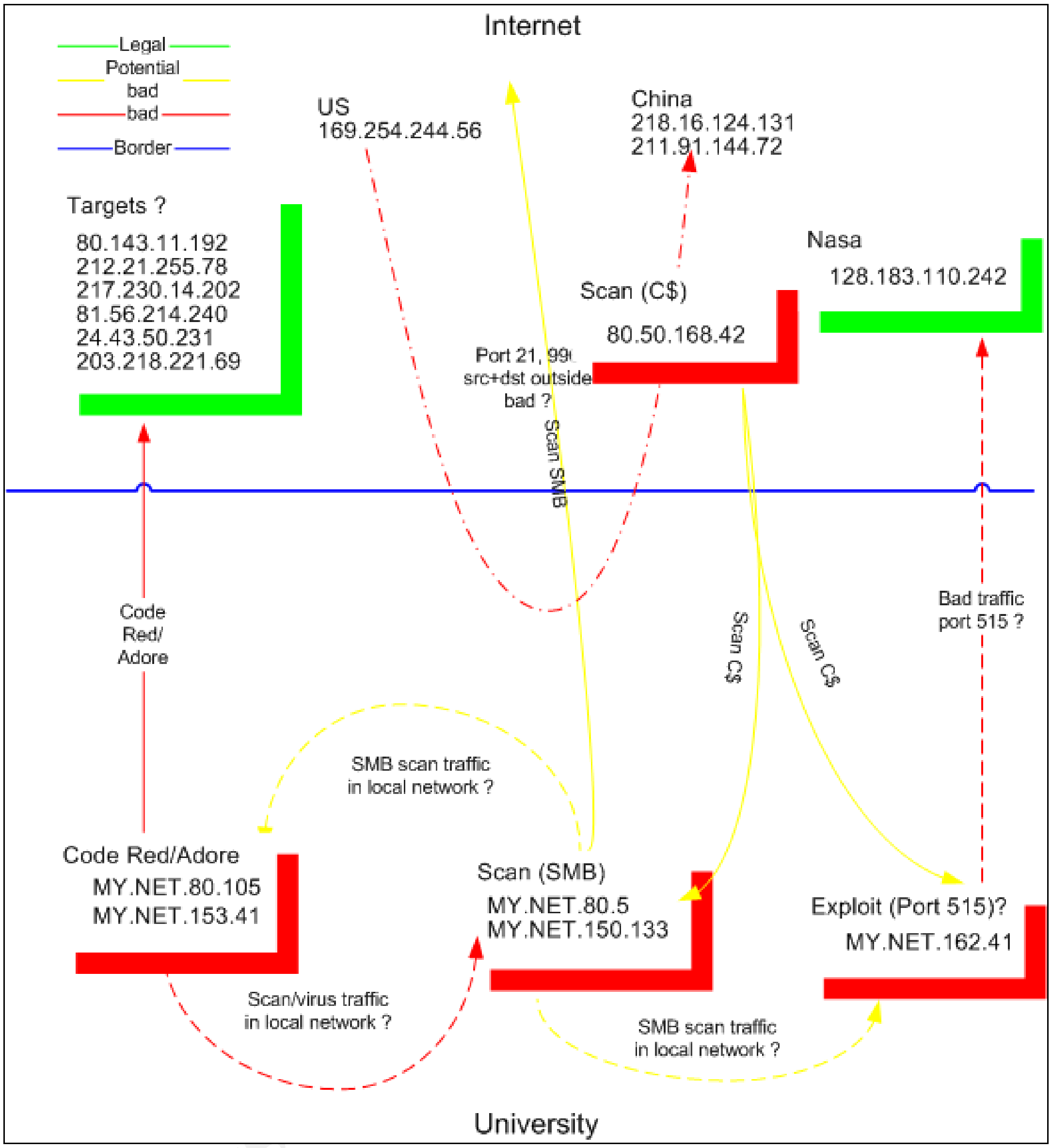
```
Oct 20 23:20:21 68.85.216.188:42188 -> MY.NET.24.34:22730 SYN ******S*
Oct 20 23:20:21 68.85.216.188:42188 -> MY.NET.24.34:22157 SYN ******S*
Oct 20 23:20:21 68.85.216.188:42188 -> MY.NET.24.34:40551 SYN ******S*
Oct 20 23:20:21 68.85.216.188:42188 -> MY.NET.24.34:54130 SYN ******S*
Oct 20 23:20:21 68.85.216.188:42188 -> MY.NET.24.34:64494 SYN ******S*
```

All with same src port but directed to a number of dst ports.
Correlation: None
Recommendation: Make sure the network at the border block unwanted traffic.
Notes

Link Graph

The following graph shows some of the top activity and talkers from analyzes. What is done is to categorize the traffic by colors, direction and potential dangers. Note that based on the logs, it is we do not actually know what inbound and outbound traffic actually get through the border.

© SANS Institute 2004, Author retains full rights.



© SANS

Defensive recommendations

This is a University network. There are obvious a lot of communication with external sources. For specific recommendations on the incidents seen, see the individual analyzes. Here are the general recommendations:

Use firewalls and DMZ's:

With the right use of DMZ's, it is possible to make multi level network security. Each DMZ should be controlled for inbound and outbound traffic by use of

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.

firewalls. There should be firewalls on the border to other networks and the Internet. The firewalls should be configured as "only necessary traffic allowed".

Use Proxies:

Proxies + reverse proxies: For web application, mail, etc – there are a number of benefits in directing this traffic through a proxy (application firewall). With right configuration a number of security problems can be addressed directly there. The network should be configured not to allow direct access from inside to out and vice versa – avoiding malicious traffic directly to servers and machines.

Shared information networks:

Use Virtual Private Network [VPN] tunnels for critical network sharing between the University and partners. When used, whole network can be isolated from the Internet – or at least be transmitted encrypted over the Internet. Also here, use a dedicated firewall where the tunnel begin and terminate.

Tune the IDS:

The University already has an IDS. If configured/tuned along with firewall policies, it should be possible to address a part of the false positives seen.

Enforce policies on personal machines:

Make sure that virus software is installed and active on machines. Do not allow peer to peer traffic from inside to outside and reverse. Students and staff should be aware about these policies and how they are enforced. Specifically, there should be a close look on machines in MY.NET.80/24 subnet, as many alarms are related to this.

Links and references:

[50, Chris]: Chris Baker, GCIA practical assignment, http://www.giac.org/practical/Chris_Baker_GCIA.zip
[51, Terry]:Terry MacDonald, GCIA practical assignment, http://www.giac.org/practical/GCIA/Terry_MacDonald_GCIA.pdf
[52, F-secure]: F-secure, Red worm/Adore, http://www.europe.f-secure.com/v-descs/adore.shtml
[53, Sans]: Sans, Adore worm, http://www.sans.org/y2k/adore.htm
[54, Trojans]: Simovits, Trojan list, http://www.simovits.com/trojans/trojans.html
[55, Lkml]: Linux Kernel Mailing List, ECN issue, http://lkml.org/faq/lkmlfaq-14.html
[56, Hudak]: Tyler Hudak, GCIA Practical Assignment, http://www.giac.org/practical/GCIA/Tyler_Hudak_GCIA.pdf
[57, Niem], Tu Niem, GCIA Practical Assignment, http://www.giac.org/practical/GCIA/Tu_Niem_GCIA.pdf

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004,   As part of GIAC practical repository.   Author retains full rights.

Words and definitions:

[DMZ]: “demilitarized zone”, http://www.webopedia.com/TERM/D/DMZ.html
[VPN]: “Virtual Private network], http://www.webopedia.com/TERM/V/VPN.html
[IDS]: “Intrusion Detection System”,
http://www.webopedia.com/TERM/I/intrusion_detection_system.html
[Scan]; “Network Scanning”,
http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci802800,00.html

© SANS Institute 2004, Author retains full rights.

© SANS Institute 2004, As part of GIAC practical repository. Author retains full rights.