Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst Practical Assignment (re-take)

Tillman Hodgson

GCIA Assignment Version 3.4

Submitted 10 June 2004

Table of Contents

List of Tables

Practical Abstract 1

The GCIA practical assignment version 3.4 is split into five parts. The first part is a paper discussing a topic related to intrusion detection. The second through fourth parts are individual and in-depth analysis of specific network detects. The fifth part is an analysis of five days worth of Snort logs for a University network.

Measuring Success at Intrusion Detection 2

Abstract

With the increasing use of intrusion detection tools comes a corresponding increasing demand for appropriate reporting models, an often diffi cult task that intrusion detection system (IDS) implementors face. Assessment is a critical element in the design, initiation, and ongoing management of successful intrusion detection systems, and one that is often overlooked. This paper examines a variety of methods to assess and measure success for intrusion detection and security practioners.

2.1 Introduction

Organizations that have deployed intrusion detection tools often face the difficult task of finding suitable metrics to use to measure their ongoing success. Measuring success periodically (often as part of a periodic reporting cycle) is important because it ties assessment into the ongoing management and evolution of your intrusion detection practices.

> Metrics for security are very difficult. [. . .] selling security is like selling insurance. If you can produce some form of metrics that makes sense, describing at some level how the security team is performing and what value they are giving the company for its money, then it will be easier to convince management to fund security infrastructure projects.[4]

Finding suitable metrics, however, can prove difficult.

2.2 What makes a good metric?

An IETF BoF[1] discussed metrics and came up with a simple list[2], summarized here:

- It should be a true metric, providing concrete, repeatable measurements without subjective interpretation (e.g. a tape measure).

[1] Internet Engineering Task Force “Birds of a Feather” conference discussion

- It should be non-discriminatory in the sense that it should not imply quality. The metric should support providers who are selling services that are ”inexpensive yet good enough” as well as ”the best possible”.
- It should be useful for publication in data sheets. This means that it is repeatable by anyone. It should have a capability to measure one provider independent of all others.
- It must be fair for homogenous (uniform) technology. It should be as fair as possible for heterogenous (non-uniform) technology. Attempting to properly account for intrinsic differences in technology is difficult.
- It should be useful for diagnostics. It should be able to sectionalize a multi-provider path.
- It should be useful for procurement, contracts and RFPs[2].
- It must be owned ”by the community”, or more specifically, not encumbered by some company or individual.

As this paper continues to discuss suitable metrics for measuring success at intrusion detection these rules will be followed.

2.3 Attempting to prove a negative?

In Gary Golomb’s response to a Gartner report on IDS[3], he said:

> Intrusion Detections systems are used for one reason. It’s your last chance to be notified about a potential break-in; a virtual safety net. Once an organization [has set up] “PROTECTIVE” technologies such as (but not limited to) firewalls, encryption, authentication, proxies, gateways, PKI, VPN, access control, virus detection/removal, etc. . . The IDS serves the single purpose of sitting back and watching over everything to see if people are still getting though. And here’s a curveball for you: After all the protective technologies just described, attackers (both automatic like worms/viruses and live people) were/are STILL getting through! Whether it’s because of vulnerabilities in network designs, application vulnerabilities, or unknowingly misconfigured devices, they do get through. And this is why IDS’s were invented. . .

Because intrusion detection systems are themselves fallible, this has some serious implications when designing metrics to measure intrusion detection. Rather than

[2]Request for Proposals, normally part of a procurement process

demonstrating that something *is* happening, effective metrics must demonstrate that something is *not* happening. This is very difficult to do by direct measurement of primary effects.

An analogy is crime statistics. Consider a scenario where a a decrease in crime is noticed: does that imply that a specific pro-active anti-crime program was successful or does it imply that an unrelated decrease in crime *reporting* took place that month?

This is supported by Bruce Schneier, who has said that[1] "...security must be evaluated not based on how it works, but on how it fails". While intrusion detection is used to detect the failure of other security mechanisms, it itself is prone to failure. Working around this quandary would involve careful selection of meaningful metrics, and may mean monitoring secondary effects.

Avoiding this quandary is an important step in designing suitable metrics for measuring success at intrusion detection.

2.4 What's the defi nition of success?

> "Good metrics should help management and other nonsecurity people within the company understand at some level what you are doing and how well you are doing it. Good metrics help build confidence in the security team for the rest of the company."[4]

What is the business goal that IDS fulfills? While this can be dependent on an organizations particular technology management goals, there are some things that most organizations will have in common for IDS[7]:

- Detecting problems that are not prevented by other security measures
- Preventing some security incidents by increasing the perceived risk of discovery and punishment of attackers
- Detecting the preambles to attacks
- Documenting the existing threat to an organization
- Quality control for security design and administration
- Providing useful information about actual intrusions

Taking time to understand the goals that intrusion detection fulfills for an organization is necessary to measuring progress ("success") along the path to that goal. Understanding the goals, then, is a prerequisite of being able to demonstrate the effectiveness of the organization's intrusion detection efforts.

2.5 So what can we measure?

Taking into consideration the previous sections, what are some metrics that we can measure on? While finding useful primary effects to measure can be difficult, there are quite a few secondary effects that suit our purposes to varying degrees of usefulness. In other fields, these might be called “measurable outcomes”.

2.5.1 Length of vulnerability gap

The time from a new exploit being announced and a signature (whether custom or vendor-supplied) being available is easily measured. Focusing on reducing this vulnerability gap, the time when your sensors are “blind” to the new type of traffic, has a positive effect on your security stance. A goal to reduce this measurement may result in a greater emphasis being placed on custom signatures and participation in the security exploit research community.

Pre-defined goals could be set, such that a gap of several hours is “green”, several hours to a day is “yellow”, greater than 24 hours is “orange”, and so forth. Exemptions for situations such as exploits that have absolutely no application in your environment would also need to be designed.

2.5.2 Cross reference your results

If your intrusion detection infrastructure allows you cross-reference results across different types of IDS, you can combine and correlate the results. This allows you to directly measure primary effects such as the types of vulnerabilities that are being targeted. With redundant sensors from different vendors using different analysis engines or signature sets you can achieve a fair degree of confidence in the results, excluding newly discovered vulnerabilities.

This redundant IDS infrastructure can be expensive to deploy and maintain, but provides more accurate primary effects to measure directly.

Examples include ensuring that trending patterns and counts for specific exploits are consistent between the IDS platforms.

2.5.3 Confi rm proper operation of other security tools

Related to the previous topic and suggested by Limoncelli and Hogan[4], you can use IDS to get the answers that you “know” you should be getting. For example, you could put sensors before and after your firewall filters traffic. Since the rules of your firewall are known, the results of the filtering are similarly known. The metric becomes, simply, any deviation from expected behaviour. This can also be used to confirm policy compliance.

This has the benefits of providing defense in depth, a solid and easily understood metric, and proves the value that other aspects of the security infrastructure are providing.

2.5.4 Time to incident resolution

The time it takes to resolve a security incident directly affects the impact and cost of security incidents. This is a secondary effect of your collective security tools and practices. It's difficult to measure without an IDS, however, as the IDS alert provides the "start time" for the incident.

Because the initial detection and alerting by an IDS system occurs very rapidly (and thus gives a reliable "start time"), this can be used to help provide provable costs for incident resolution. Being able to prove the costs of current incident resolution practices allows improvements (or regressions!) to the process to be measured, making it easier to obtain management support for security initiatives.

Tying this to an incident response procedure that includes timed points ("within 15 minutes of detection", etc) can often allow an organization to fine tune individual steps within it's process. For example, a manager may be contacted and briefed for possible escalation if an incident does not meet it's resolution milestones within a certain period of time.

Specific metric examples include:

- Minutes to first response
- Minutes to completed response (not including post-incident analysis and reporting)
- Percentage of security staff time spent in reactive rather than pro-active security (by week, month or quarter)
- Total employee time (in person-hours) spent in resolving any given incident, and
- Total lost employee time (in person-hours) to other areas of the organization as a result of the incident

Each of those items can be tracked over time to show incident resolution trends and highlight where initiatives might have the most effect.

Custom variants of this metric are possible, covering situations such as:

- After-hours coverage (which may have different response requirements)
- Different types of incident, different scope and impact of incidents

2.5.5 Alerts requiring response

The number of alerts generated by an IDS system is a topic that many organizations considering deployment are concerned about. The real measurement here is the number of alerts that need to be *responded* to. While it needs to be emphasized that this number is not under the direct control of the security team (as the attackers are generally outside of their influence), the trending information can be useful. For example, in spite of the generally upward trend of attempted attacks per month, a sudden down-swing in the response count after a concerted alert tuning effort provides positive feedback.

Examples of ways to measure this include a simple count of alerts that were paged per day, alerts that were paged and required some level of investigation, and alerts that were paged and required some level of incident response. Tracking this metrics over time can provide valuable trending information and a business case tp justify time spent in tuning alerts.

2.5.6 Financial impact

Monitoring an intrusion detection system carries an ongoing operational cost. This is a secondary effect that can be measured to show increased efficiency. The time spent monitoring, as a percentage of an FTE[3], can be tracked to measure the impact of adding sensors and the positive effects of time spent tuning alerts. It is also a component of the overall cost of computer security to an organization.

This can be tracked using a combination of metrics, such as the sum of the cost for on-call time, alert investigation, alert response, incident response, and post-incident analysis. Other potential variables include cost of lost staff time due to an incident and cost of loss of data integrity.

2.5.7 User impact

Intrusion detection may have a user-visible impact to an organization's user community. An example of this secondary effect is automated responses in the event of a false positive. Other examples include an incident response resulting in a host or network being temporarily disconnected, an in-line intrusion detection sensor failing and interrupting the traffic on the link it was monitoring, and performance impact for an on-host sensor.

If an organization has a goal of providing low impact to their user community, this impact can be tracked. Efforts to reduce the user-visible impact, such as using copies of traffic for network sensors rather than placing them in-line, can then have their effectiveness determined. One way to do this is to add an "IDS" item to your

[3] "Full-time equivalent"

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

help desk ticket system and change control system. A network outage caused by a bad sensor, for example, can then have it's impact tracked. This might provide the business case for a different architecture that reduces dependencies and increases fault tolerance.

Related to this is having well-developed service monitoring and data integrity monitoring[5]. While not strictly an intrusion detection metric, operational metrics and intrusion detection metrics fundamentally lead into each other. Certainly the impact of lost or untrustworthy data or services is visible to the users.

2.5.8 Compare your organization to others

Being involved in a geographically local or industry security organization (such as a regional CERT/CIRT or an industry association with a security committee) is a good security practice. It also allows an organization to compare itself to similar organizations (or at least organizations in similar IP space).

This can be considered a higher-level view of the same issues that this paper addresses: What are good intra-organizational metrics that can be used to compare disparate security infrastructures? Developing those within security communities that you take part in will provide a set of useful comparison metrics that can be jointly defined and evolved.

Examples include percentage of desktops affected by the latest worm, security as a percentage of the IT budget (which can demonstrate cost effectiveness when combined with other information), and alerts requiring response (per month).

Developing the "political" climate that will allow this sort of information sharing a is large task with little published art[6].

2.5.9 Tin Soldiers: Tabletop exercises

Running incident response exercises or drills at a defined cyclic period (such as quarterly) allows one to measure a host of metrics in a controlled environment. In general this may be said to relate more to the human component of intrusion detection, and leads towards metrics for incident response. This is not necessarily a bad thing: success at incident response aligns well with success at intrusion detection.

The tabletop exercise could have a set of goals that are being scored against, specific to that exercise, as well as a standard set of milestones that need to be reached in the incident response process.

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

2.5.10 Penetration tests

External auditing of the general security infrastructure can be a useful source of metrics. Often, holes identified by the audit can be shown by the security team to have at least partial coverage through the application of intrusion detection and the incident response process. Other identified problems can become the “to do” list for the evolution of the intrusion detection architecture.

Similarly, penetration tests can be used to measure the complete incident response process in a fairly controlled environment (in the sense that both sides of the story are known). Needless to say, if a penetration test is not detected the intrusion detection systems have not demonstrated success.

2.6 Conclusion

This paper examined a variety of methods to assess and measure success for intrusion detection and security practioners. While often overlooked in relatively new field that is intrusion detection, an organization looking to “raise the bar” for professional intrusion detection management must carefully consider how to go about measuring their success.

Building effective reporting and self-assessment models for intrusion detection can be difficult because its direct effects are not necessarily what you want to measure. Defining the goals for IDS in your organization and choosing appropriate metrics to measure progress towards those goals is a critical element of a successful intrusion detection strategy.

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

Bibliography

[1] Bruce Schneier, *Crypto-Gram Newsletter*, April 15 2004

[2] *IP Provider Metrics BoF, 32nd IETF – Danvers, Mass.*, http://www.psc.edu/ mathis/ippm/meet01.danvers32/minutes.html

[3] Gary Golomb, *IDS v. IPS Commentary*, June 16 2003, posted to the focus-ids@securityfocus.com and isn@c4i.org mailing lists, archived at http://www.linuxsecurity.com/articles/forums_article-7476.html

[4] Thomas A. Limoncelli and Christine Hogan, *The Practice of System and Network Administration*, Addison-Wesley, 2002, pg 157–158

[5] Simson Garfinkel and Gene Spafford, *Practical Unix & Internet Security (2nd edition)*, O'Reilly, 1996, pg 277–287

[6] Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, *Building Internet Firewalls*, O'Reilly, 2000, pg 734–741

[7] Tillman Hodgson, *Current State of the Art* deliverable (client deliverable not publicly released), Dec 2001

```
The first two signatures always occurred in matching pairs, and are
directly related:

[**] [111:2:1] (spp_stream4) possible EVASIVE RST detection [**]
06/29-18:14:32.934488 255.255.255.255:31337 -> 46.5.215.12:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
(Also IPs 46.5.215.40, 46.5.162.228, 46.5.205.134, 46.5.244.39,
 46.5.195.252, 46.5.61.22, 46.5.68.148, 46.5.64.14, 46.5.40.247,
 46.5.87.177, 46.5.8.229, 46.5.7.18, 46.5.40.166, 46.5.207.168,
 46.5.64.85, 46.5.24.93, 46.5.140.171, 46.5.241.69, 46.5.34.176,
 46.5.105.30, 46.5.85.248, 46.5.229.33, 46.5.60.19, 46.5.241.219,
 46.5.138.210, 46.5.102.14, 46.5.175.8, 46.5.72.144, 46.5.22.227,
 46.5.155.153, 46.5.61.230, 46.5.15.14, 46.5.61.77, 46.5.0.8,
 46.5.222.89, 46.5.71.167, 46.5.205.220, 46.5.144.236, 46.5.24.235,
 46.5.85.63, 46.5.183.212, 46.5.69.246, 46.5.211.245, 46.5.136.133)

[**] [1:184:4] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
06/29-18:14:32.934488 255.255.255.255:31337 -> 46.5.215.12:515
TCP TTL:14 TOS:0x0 ID:0 IpLen:20 DgmLen:43
***A*R** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS203]
(Also IPs 46.5.215.40, 46.5.162.228, 46.5.205.134, 46.5.244.39,
 46.5.195.252, 46.5.61.22, 46.5.68.148, 46.5.64.14, 46.5.40.247,
 46.5.87.177, 46.5.8.229, 46.5.7.18, 46.5.40.166, 46.5.207.168,
 46.5.64.85, 46.5.24.93, 46.5.140.171, 46.5.241.69, 46.5.34.176,
 46.5.105.30, 46.5.85.248, 46.5.229.33, 46.5.60.19, 46.5.241.219,
 46.5.138.210, 46.5.102.14, 46.5.175.8, 46.5.72.144, 46.5.22.227,
 46.5.155.153, 46.5.61.230, 46.5.15.14, 46.5.61.77, 46.5.0.8,
 46.5.222.89, 46.5.71.167, 46.5.205.220, 46.5.144.236, 46.5.24.235,
 46.5.85.63, 46.5.183.212, 46.5.69.246, 46.5.211.245, 46.5.136.133)
```

3.1 Source of the network trace log

The network trace was obtained from `http://isc.sans.org/logs/Raw/2002.5.30`, part of the logs provided specifically for use in a GCIA practical. As such, not much is known about the network in question – the IP addresses were obfuscated, for example. It appears that the network is a `/16`, represented with the IPs `46.5.0.0`.

3.2 Detect was generated by Snort

Summarizing the `README`, the log files are the result of a Snort instance running in binary logging mode and only the packets that violate the rule set will appear in the log. All ICMP, DNS, SMTP and Web traffic has also been removed.

Snort 2.1.2, using a default FreeBSD -STABLE port installation, was used to generate the detects shown above. The command used was:

```
snort -c /usr/local/etc/snort.conf -N -A full -l snort -r 2002.5.30
-S HOME_NET=46.5.180.0/16 -k none.
```

Note that the `-k none` is required when running sanitized libpcap-recorded network traces through snort because the checksum modifications will otherwise interfere.

The output shown eliminates duplicates for brevity, electing instead to simply list additional IPs. If a signature matched exactly except for the destination IP address, it was counted as a duplicate.

A number of other potentially malicious packets were detected, which will not be covered in this Network Detect section. For reference purposes: 13 unique IPs came up for “DNS named version attempt”; “Oversize chunk encoding”, “Non-RFC HTTP delimiter”, “TCP checksum changed on retransmission”, “Scan SOCKS Proxy attempt” and “Bare byte unicode encoding” signatures were also seen. Additionally, there were a number of web server scans for unsecured `formmail.pl` installations and IIS exploits `cmd.exe` that Snort did not alert on.

3.3 Probability the source address was spoofed

The packets with a source IP address of `255.255.255.255` are certainly spoofed. This is a limited broadcast address, described by RFC 919 section 7[1]. As such there are some contraints on it’s behavior:

- It denotes a broadcast on the local hardware network, which must not be forwarded. This address is typically used by hosts that do not their IP and are broadcasting for a DHCP server to provide it.
- It would only be seen legitimately as a destination address. Supporting this is the fact that only TCP Reset packets are seen, meaning that an established stateful connection does not exist.

The port in question (LPD) certianly does not involve DHCP-like behaviour. It’s interesting that `255.255.255.255` was used as the *source* address. This could perhaps be used to obscure the real source while still allowing the destination to be reached (as routing decisions noramlly occur on destination, and not source, IP addresses).

[1]Also described by W. Richard Stevens, *TCP/IP Illustrated, Volume 1*, pg 171

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

3.4 Description of attack

Self-described as “Remote access and redirection services with strong encryption”, in the same vein as NetBus and Back Oriface. In some ways it can be thought of a secure version of `netcat`. There is a slim possibility that the administrator is using Q purposely, though there isn't enough information to know that. Written by “Mixter”[2], Q has advanced features like challenge-response authentication, strong encryption, and connection bouncing (typically used for IRC). CVE candidate CAN-1999-0660, a generic entry covering a “hacker utility, back door, or Trojan Horse”, is the closest fit for this signature.

This server must be installed by someone, and is therefore typically a sign of a generally compromised host. The default name of the executable is `qd` and it must run as the root UID, but it includes compile-time options to rename its process to `klogd` and to change the UID it runs under.

3.5 Attack Mechanism

This does not appear to be a direct attack, but rather a querying scan.

Creative use of `wc -l` and `uniq` while grep'ing the snort logs for the “BACKDOOR Q access” signature shows that there were 44 packets detected. The destination IP was unique in each instance, as listed in the initial packet detect at the beginning of this section. This implies that ongoing “conversations” are not occurring, but rather that it is a stimulus to query Q trojans previously installed.

This analysis supported by the fact that all the packets are mostly identical (as shown by `tcpdump -v -X -r 2002.5.30`). Typical packet contents look like this:

```
0x0000   4500 002b 0000 0000 0e06 acc4 ffff ffff        E..+............
0x0010   2e05 d70c 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 60ec 0000 636b 6f00 0000             P.......cko...
```

Variations include bytes 11-12, 19–20, and 37–38 (which sometimes contain data). `cko` always appears in bytes 41 through 43.

Port 515, reserved for the `lpd` printer service, was likely used because it has a greater possibility of being allowed through loosely-configured firewalls (as compared to a random port, though not as compared to common ports such as 80) and of possibly being considered “normal” traffic by an inexperienced network administrator. Both the RESET and the ACK bit are set in the TCP packet: either may help the packet pass through less-capable firewalls. Note that it is possible to make Q use different ports and flags, these are merely the defaults.

[2] Source code available at http://mixter.void.ru/progs.html, the README is particularly instructive

3.6 Correlations

Q was written by “Mixter”, and the source code is available at `http://mixter.void.ru/progs.html`. It was widely reported on security mailing lists starting in May of 2001. CVE candidate CAN-1999-0660, a generic entry covering a “hacker utility, back door, or Trojan Horse”, is the closest fit for this signature.

Whitehats.ca published a detailed analysis on Q[3].

3.7 Evidence of active targeting

The IPs appear to be randomly chosen, part of a scan for already planted Q trojans. No direct evidence for active targeting exists. Grep’ing throug the log file for the targeted IPs does not show them to be subsequently involved in any activity, which supports the hypothesis that the Q activity was an unsuccessful scan.

3.8 Severity

Severity =
(Criticality & Lethality) – (System Countermeasures & Network Countermeasures)

Without the network and host configuration details being available, the criticality of the targeted system is unknown. Backdoor Q can be installed on any kind of host, from workstations through to highly critical servers (on both Unix variants as well as Windows), and so the criticality must be estimated as at least moderate to be safe. *Criticality = 3*.

Because Q, if it is indeed installed, involves root-level access, the lethality of an attacker with control of an active Q installation on the target network can be extremely high because the root access is also carried within a stealthed and encrypted channel. Note that Q can also set up session redirection on a compromised target and thus turn into a platform for further attacks. *Lethality = 5*.

The actual state of system countermeasures are unknown. However, Q must first be installed with root-level access before it can be used. This means than an initial root-level intrusion must have taken place. Note that Q is able to disguise its process-names using standard root-kit techniques. Assuming standard security measures have been taken, this is a difficult task. *System countermeasures = 4*.

The actual state of network countermeasures are unknown, except that snort was running to capture the attack (which is a good sign!). Q can be configured to use a variety of ports and flags rather than just the default ones seen in this particular

[3]Available at http://www.whitehats.ca/main/publications/external_pubs/Q-analysis/Q-analysis.html

set of logs, making detection and thus countermeasures difficult. *Network countermeasures = 3*.

The overall severity of these detected signatures is 1.

3.9 Defensive recommendations

Effective ingress filtering (and egress filtering at the ISP level) on source IPs (instead of just the more common destination IP access controls) would eliminate the IP spoofing. Firewalling the default port for Q (515) would be effective because `lpd` is a rarely a service that should be offered over the Internet. Host-based integrity checking would detect the installation of Q. Network intrusion detection sensors can warn about scanning – if a host responds, immediate intrusion response can be undertaken. The Whitehats.ca paper referenced in the Correlations section lists a variety of lab-tested signatures that would be useful for this situation.

3.10 Multiple choice question

Given the following trace:

```
18:14:32.934488 255.255.255.255.31337 > 46.5.215.12.515: R 0:3(3) ack 0 win 0 [RST cko]
(ttl 14, id 0, l
en 43, bad cksum acc4!)
0x0000   4500 002b 0000 0000 0e06 acc4 ffff ffff        E..+............
0x0010   2e05 d70c 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 60ec 0000 636b 6f00 0000              P...‘...cko...
```

And assuming no previous packets (such as a TCP handshake) have been exchanged, what is the most likely purpose of this packet?

1. OS fingerprinting
2. Access to a backdoor/trojan
3. Denial of service attack
4. General LPD service exploit

Answer: “Access to a backdoor/trojan”. “General LPD service exploit” is a red herring.

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

3.11 Posting to intrusions@incidents.org

This network detect was posted to the intrusions@incidents.org mailing list May 04, 2004. However, a “not allowed to post” reply was received. After confirming that my subscription was indeed still valid, a detailed request for help was sent to the list owner. No reply was received.

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

```
(Bad checksum information has been removed from this listing)

20:29:48.956507 66.28.100.206.53247 > 32.245.157.117.8080: S
    330352269:330352269(0) win 5840 <mss 1460,sackOK,timestamp
    4653786 0,nop,wscale 0> (DF) (ttl 43, id 50263, len 60)
20:29:51.976507 66.28.100.206.53247 > 32.245.157.117.8080: S
    330352269:330352269(0) win 5840 <mss 1460,sackOK,timestamp
    4654086 0,nop,wscale 0> (DF) (ttl 43, id 50264, len 60)
20:29:51.976507 66.28.100.206.55301 > 32.245.157.117.3128: S
    332210663:332210663(0) win 5840 <mss 1460,sackOK,timestamp
    4654086 0,nop,wscale 0> (DF) (ttl 43, id 9697, len 60)
20:29:54.956507 66.28.100.206.55301 > 32.245.157.117.3128: S
    332210663:332210663(0) win 5840 <mss 1460,sackOK,timestamp
    4654386 0,nop,wscale 0> (DF) (ttl 43, id 9698, len 60)
20:29:57.956507 66.28.100.206.59422 > 32.245.157.118.8080: S
    349984409:349984409(0) win 5840 <mss 1460,sackOK,timestamp
    4654686 0,nop,wscale 0> (DF) (ttl 43, id 20679, len 60)
20:30:00.956507 66.28.100.206.59422 > 32.245.157.118.8080: S
    349984409:349984409(0) win 5840 <mss 1460,sackOK,timestamp
    4654986 0,nop,wscale 0> (DF) (ttl 43, id 20680, len 60)
20:30:00.956507 66.28.100.206.33267 > 32.245.157.118.3128: S
    344086866:344086866(0) win 5840 <mss 1460,sackOK,timestamp
    4654986 0,nop,wscale 0> (DF) (ttl 43, id 24280, len 60)
20:30:03.956507 66.28.100.206.33267 > 32.245.157.118.3128: S
    344086866:344086866(0) win 5840 <mss 1460,sackOK,timestamp
    4655286 0,nop,wscale 0> (DF) (ttl 43, id 24281, len 60)
20:30:06.956507 66.28.100.206.37368 > 32.245.157.119.8080: S
    360908682:360908682(0) win 5840 <mss 1460,sackOK,timestamp
    4655586 0,nop,wscale 0> (DF) (ttl 43, id 3688, len 60)
20:30:09.966507 66.28.100.206.37368 > 32.245.157.119.8080: S
    360908682:360908682(0) win 5840 <mss 1460,sackOK,timestamp
    4655886 0,nop,wscale 0> (DF) (ttl 43, id 3689, len 60)
20:30:09.966507 66.28.100.206.39422 > 32.245.157.119.3128: S
    357870743:357870743(0) win 5840 <mss 1460,sackOK,timestamp
    4655886 0,nop,wscale 0> (DF) (ttl 43, id 42905, len 60)
20:30:12.956507 66.28.100.206.39422 > 32.245.157.119.3128: S
    357870743:357870743(0) win 5840 <mss 1460,sackOK,timestamp
    4656186 0,nop,wscale 0> (DF) (ttl 43, id 42906, len 60)
etc ...
```

```
(The corresponding sample Snort alerts)
[**] [1:620:6] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
10/25-20:29:48.956507 66.28.100.206:53247 -> 32.245.157.117:8080
TCP TTL:43 TOS:0x0 ID:50263 IpLen:20 DgmLen:60 DF
******S* Seq: 0x13B0C68D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
```

```
7  TCP Options (5) => MSS: 1460 SackOK TS: 4653786 0 NOP WS: 0
8
9  [**] [1:620:6] SCAN Proxy Port 8080 attempt [**]
10 [Classification: Attempted Information Leak] [Priority: 2]
11 10/25-20:29:51.976507 66.28.100.206:53247 -> 32.245.157.117:8080
12 TCP TTL:43 TOS:0x0 ID:50264 IpLen:20 DgmLen:60 DF
13 ******S* Seq: 0x13B0C68D  Ack: 0x0  Win: 0x16D0  TcpLen: 40
14 TCP Options (5) => MSS: 1460 SackOK TS: 4654086 0 NOP WS: 0
15
16 [**] [1:618:5] SCAN Squid Proxy attempt [**]
17 [Classification: Attempted Information Leak] [Priority: 2]
18 10/25-20:29:51.976507 66.28.100.206:55301 -> 32.245.157.117:3128
19 TCP TTL:43 TOS:0x0 ID:9697 IpLen:20 DgmLen:60 DF
20 ******S* Seq: 0x13CD21E7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
21 TCP Options (5) => MSS: 1460 SackOK TS: 4654086 0 NOP WS: 0
22
23 [**] [1:618:5] SCAN Squid Proxy attempt [**]
24 [Classification: Attempted Information Leak] [Priority: 2]
25 10/25-20:29:54.956507 66.28.100.206:55301 -> 32.245.157.117:3128
26 TCP TTL:43 TOS:0x0 ID:9698 IpLen:20 DgmLen:60 DF
27 ******S* Seq: 0x13CD21E7  Ack: 0x0  Win: 0x16D0  TcpLen: 40
28 TCP Options (5) => MSS: 1460 SackOK TS: 4654386 0 NOP WS: 0
```

4.1 Source of the network trace log

The network trace was obtained from `http://isc.sans.org/logs/Raw/2002.9.26`, part of the logs provided specifically for use in a GCIA practical. As such, not much is known about the network in question – the IP addresses were obfuscated, for example. It appears that the network is a `/24`, represented with the IPs `32.245.157.0`. Every IP from `.1` through to `.253` was represented.

4.2 Detect was generated by Snort

Summarizing the `README`, the log files are the result of a Snort instance running in binary logging mode and only the packets that violate the rule set will appear in the log. All ICMP, DNS, SMTP and Web traffic has also been removed.

Tcpdump 3.7.2 and Snort 2.1.2, using the default FreeBSD -STABLE port installations, was used to generate the detects shown above. The Snort command used was:

```
snort -c /usr/local/etc/snort.conf -N -A full -l snort -r 2002.9.26
-S HOME_NET=32.245.157.0/24 -k none.
```

Note that the `-k none` is required when running sanitized libpcap-recorded network traces through snort because the checksum modifications will otherwise interfere.

The output shown above has been stripped down to representative samples for brevity.

4.3 Probability the source address was spoofed

The source of the scan, `66.28.100.206`, is likely legitimate. SYN scans require the ability to receive a response in order to perform their scan. If the attacker was local to the subnet and was able to sniff the resulting traffic it's possible that the IP was spoofed. However, if an intruder had the capability to sniff traffic then it follows that they could passively sniff for traffic to the ports they were interested in in order to discover local proxies and thus avoid the chance of their scan being detected and reacted to.

I also noted that the source ports appear to be reasonable and change between each SYN attempt in a manner which tends to indicate a single host as being the source.

4.4 Description of attack

There were 1016 snort alerts for `SCAN Squid Proxy attempt` which corresponds with 1016 alerts for `SCAN Proxy Port 8080 attempt`. After reviewing the alerts in question, This correlates to a scan of the entire Class C target network. The scan connects to each IP in turn, beginning with `.117` and wrapping around until it completes the network range. The scan is a simple SYN scan which sends two attempts to each of port 3128 and 8080. These ports are commonly used by web proxies such as Squid. The source of all the alerts for ports 8080 and 3128 (the proxy scan) is the IP address `66.28.100.206`.

There are proxy vulnerabilities that this scan could be looking for. Because the ports do not correlate to a specific proxy product it is more likely that this scan is merely looking for *any* web proxy. Proxies are generally useful to an intruder, serving purposes such as obscuring the real origin of outgoing attacks and perhaps providing a method to bypass filtering mechanisms.

4.5 Attack Mechanism

This did not appear to be a direct attack, but was rather a simple and straight-forward SYN scan (i.e. not using any stealth techniques) for proxy servers.

The scan simply sent a SYN packet (twice) to each of TCP port 8080 and 3128. The timing is rather interesting: an initial packet is sent to 8080, and three seconds later a packet is sent to 8080 again and immediately afterwards a packet is sent to 3128. Three seconds after that one is sent, a second packet is sent to 3128.

Testing with both a Linux-based and a BSD-based TCP/IP stack show that this is normal behaviour for the SYN retransmit interval[1] Because each port is tried only twice, the 3 second value appears to be normal TCP behaviour and not usable as a fingerprint to help determine the scanning tool.

If an open proxy had been found, the scanner would have been able to use that as a stepping stone to perform possibly malicious network actions (such as web-based attacks) that would appear to be coming from the proxy and not from the originating IP address. Indeed, it's possible the source of the scan is itself an open general (not just HTTP) proxy.

4.6 Correlations

Becuase there are many web proxy products[2] there are correspondingly many CVE numbers that might apply.

Because port 3128 was included in addition to 8080, it seems most likely a generic HTTP proxy was being sought rather than only Squid. Because proxies are so convenient, it's unlikely that the attacker intended to exploit the proxy itself but rather intended to use it for it's intended (albeit unauthorized) purpose.

Based on those assumptions, I wasn't able to locate a CVE for a generic open HTTP proxy scan. There were several Squid specific CVE entries that might apply for the next stage of what the scanner was likely trying to do:

- CVE-1999-1481: Squid 2.2.STABLE5 and below, when using external authentication, allows attackers to bypass access controls via a newline in the user/password pair. This would be very useful for using even user authenticated proxies.
- CVE-2001-1030: Squid before 2.3STABLE5 in HTTP accelerator mode does not enable access control lists (ACLs) when the httpd_accel_host and http_accel_with_proxy off settings are used, which allows attackers to bypass the ACLs and conduct unauthorized activities such as port scanning. Again, this would be useful for using proxies even when ACLs were supposed to be in effect.
- CVE-2002-0067: Squid 2.4 STABLE3 and earlier does not properly disable HTCP, even when "htcp_port 0" is specified in squid.conf, which could allow remote attackers to bypass intended access restrictions.
- CAN-1999-1273: Squid Internet Object Cache 1.1.20 allows users to bypass access control lists (ACLs) by encoding the URL with hexadecimal escape sequences.

[1]The values "backed off" via simple doubling: 3 seconds, 6 seconds, 12 seconds, 24 seconds, etc.
[2]Including Squid, Microsoft Proxy and ISA server, and others

The Ring Zero scan (http://www.sans.org/resources/idfaq/ring_zero.php) was considered as a correlation for this detect but could not be confirmed as a possibility because because the analyzed scan does not include port 80 (one of the signatures of Ring Zero) or port 8000 (one of the occasionally-occuring signatures of Ring Zero). Examination of the source machine for the identified trojan files would be required to either confirm or eliminate this as a correlation.

4.7 Evidence of active targeting

There is no evidence of active targeting. In fact, as this is a sequential-by-IP scan, there is strong evidence that that it is a general scan of the entire network (though the size of the entire network is not known).

4.8 Severity

Severity =
(Criticality & Lethality) – (System Countermeasures & Network Countermeasures)

Without the network and host configuration details being available (such as whether or not there were any HTTP proxies), the criticality of the targeted system is unknown. However, the scan was intended for an HTTP proxy, which generally permits Internet access and can lead to a second stage of malicious activity. Thus the criticality must be estimated as high to be safe. *Criticality = 4*.

The lethality of the version scan itself is quite low, basically unauthorized HTTP access. There is rarely trust relationships involving HTTP proxies, which also minimizes the lethality. If the HTTP service was disrupted, it might cause a fair amount of inconvenience however. HTTP access to the Internet is assumed to be either not business critical, or business critical but mitigated with fault tolerant network infrastructure. *Lethality = 2*.

The actual state of system countermeasures are unknown. However there are known ways to restrict access to an HTTP proxy by IP and by robust user authentication. Verbose logging and automated log checking can also be helpful. Assuming standard security measures have been taken, this implies a reasonable level of available system countermeasures. *System countermeasures = 3*.

The actual state of network countermeasures are unknown, except that snort was running to capture the attack (which is a good sign!). Firewalls are often ineffective with this scan because a HTTP proxy generally operates on the perimeter of the network and may not be protected by a firewall. The HTTP proxy itself can (and should) implement IP-based restrictions. Detection of this signature via an IDS system is an important step because a network-wide scan takes a fair amount of time, giving the responders an opportunity to black-list the originating IP. *Network countermeasures = 3*.

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

The overall severity of these detected signatures is zero.

4.9 Defensive recommendations

All proxies should be secured, preferably with both IP-based access control lists as well as with robust user authentication mechanisms. Authenticating the user is important not only for restricting access but also to properly identify the source of traffic in the case of multi-user machines.

Access control lists on the firewalls and/or the proxy itself should control ingress/egress carefully. Proxying should generally only occur from the internal network to the external network, any other combination is likely malicious in nature. Ideally, proxy servers would also be located in seperate network, hung off the firewall, that could be more carefully controlled and monitored.

Proxies should also be configured to verbosely log what they are proxying. In the event a proxy is used for malicious purposes, this will aid in backtracking the origin of the traffic.

A honey-pot could detect this scan and potentially tie it up in slow responses, perhaps logging more obviously malicious as well.

Quick response to an IDS alert for this sort of scanning (or even automated response if your organization is comfortable with it) could black-list the scanning IP address until the system can be investigated.

4.10 Multiple choice question

Given a network trace that shows a single IP connecting to hosts (sequentially, by IP) on TCP ports 8080 and 3128, what would you expect the next action to be?

1. An Apache exploit
2. Access to a backdoor/trojan
3. A squid exploit
4. Unauthorized use of a squid proxy

Answer: “Unauthorized usse of a squid proxy”. “A squid exploit” is a red herring.

```
Snort reports 3 interesting (and similar) packets:

[**] [111:5:1] (spp_stream4) DATA ON SYN detection [**]
08/25-13:56:08.454488 209.67.29.9:2100 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:46381 IpLen:20 DgmLen:64
******S* Seq: 0x3176F88A  Ack: 0x0  Win: 0x800  TcpLen: 20

[**] [1:526:7] BAD-TRAFFIC data in TCP SYN packet [**]
[Classification: Misc activity] [Priority: 3]
08/25-13:56:08.454488 209.67.29.9:2100 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:46381 IpLen:20 DgmLen:64
******S* Seq: 0x3176F88A  Ack: 0x0  Win: 0x800  TcpLen: 20
[Xref => http://www.cert.org/incident_notes/IN-99-07.html]

[**] [111:5:1] (spp_stream4) DATA ON SYN detection [**]
08/25-13:56:08.454488 209.67.29.9:2101 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:19148 IpLen:20 DgmLen:64
******S* Seq: 0x6D66C06  Ack: 0x0  Win: 0x800  TcpLen: 20

[**] [1:526:7] BAD-TRAFFIC data in TCP SYN packet [**]
[Classification: Misc activity] [Priority: 3]
08/25-13:56:08.454488 209.67.29.9:2101 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:19148 IpLen:20 DgmLen:64
******S* Seq: 0x6D66C06  Ack: 0x0  Win: 0x800  TcpLen: 20
[Xref => http://www.cert.org/incident_notes/IN-99-07.html]

[**] [111:5:1] (spp_stream4) DATA ON SYN detection [**]
08/25-13:56:08.454488 209.67.29.9:2102 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:48055 IpLen:20 DgmLen:64
******S* Seq: 0x77802862  Ack: 0x0  Win: 0x800  TcpLen: 20

[**] [1:526:7] BAD-TRAFFIC data in TCP SYN packet [**]
[Classification: Misc activity] [Priority: 3]
08/25-13:56:08.454488 209.67.29.9:2102 -> 138.97.18.88:53
TCP TTL:241 TOS:0x0 ID:48055 IpLen:20 DgmLen:64
******S* Seq: 0x77802862  Ack: 0x0  Win: 0x800  TcpLen: 20
[Xref => http://www.cert.org/incident_notes/IN-99-07.html]


tcpdump shows this as:

13:56:08.454488 209.67.29.9.2100 > 138.97.18.88.53: S [bad tcp cksum 4040!]
829880458:829880482(24) win 2048 0 [0q] (22) (ttl 241, id 46381, len 64, bad cksum ff44!)
0x0000   4500 0040 b52d 0000 f106 ff44 d143 1d09        E..@.-.....D.C..
0x0010   8a61 1258 0834 0035 3176 f88a 0000 0000        .a.X.4.51v......
0x0020   5002 0800 aa1a 0000 0000 0000 0000 0000        P...............
```

```
0x0030   0000 0000 0000 0000 0000 0000 0000 0000        ................
13:56:08.454488 209.67.29.9.2101 > 138.97.18.88.53: S [bad tcp cksum 4040!]
114715654:114715678(24) win 2048 0 [0q] (22) (ttl 241, id 19148, len 64, bad cksum ffa5!)
0x0000   4500 0040 4acc 0000 f106 ffa5 d143 1d09        E..@J........C..
0x0010   8a61 1258 0835 0035 06d6 6c06 0000 0000        .a.X.5.5..l.....
0x0020   5002 0800 613e 0000 0000 0000 0000 0000        P...a>..........
0x0030   0000 0000 0000 0000 0000 0000 0000 0000        ................
13:56:08.454488 209.67.29.9.2102 > 138.97.18.88.53: S [bad tcp cksum 4040!]
2004887650:2004887674(24) win 2048 0 [0q] (22) (ttl 241, id 48055, len 64, bad cksum ffba!)
0x0000   4500 0040 bbb7 0000 f106 ffba d143 1d09        E..@.........C..
0x0010   8a61 1258 0836 0035 7780 2862 0000 0000        .a.X.6.5w.(b....
0x0020   5002 0800 3437 0000 0000 0000 0000 0000        P...47..........
0x0030   0000 0000 0000 0000 0000 0000 0000 0000        ................
```

5.1 Source of the network trace log

The network trace was obtained from `http://isc.sans.org/logs/Raw/2002.7.25`, part of the logs provided specifically for use in a GCIA practical. As such, not much is known about the target network in question – the IP addresses were obfuscated, for example.

Despite the name of the log file, the packets were from 08/25, not 07/25.

5.2 Detect was generated by Snort

Summarizing the `README`, the log files are the result of a Snort instance running in binary logging mode and only the packets that violate the rule set will appear in the log. All ICMP, DNS, SMTP and Web traffic has also been removed.

Snort 2.1.2, using a default FreeBSD -STABLE port installation, was used to generate the detects shown above. The command used was:

```
snort -c /usr/local/etc/snort.conf -N -A full -l snort -r 2002.7.25 -k none.
```

Note that the `-k none` is required when running sanitized libpcap-recorded network traces through snort because the checksum modifications will otherwise interfere. The `tcpdump` network dump was generated with:

```
tcpdump -n -X -v -r 2002.7.25
```

The output shown is only a small portion of the complete network trace contained in the log file. The interesting packets shown will be analyzed.

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

5.3 Probability the source address was spoofed

Very unlikely. The rest of this analysis will show that this is likely annoying but legitimate traffic—a false positive. Legitimate traffic is not normally spoofed.

This hypothesis is supported by the fact that the service that F5 provides relies on the source IP being reachable in order for replies (used to determine network path information) to be received. Being "stealthy" doesn't serve the purposes of F5 at all: it would simply raise the amount of abuse reports caused by false positives that their customers would have to handle.

5.4 Description of attack

Extracting the rule information for `BAD-TRAFFIC data in TCP SYN packet` from the Snort rules database, we find:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any
 (msg:"BAD-TRAFFIC data in TCP SYN packet";
  flags:S,12;
  dsize:>6;
  stateless;
  reference:url,www.cert.org/incident_notes/IN-99-07.html;
  sid:526;
  classtype:misc-activity;
  rev:7;)
```

This implies that any TCP SYN packet coming into our network with a data payload greater than six bytes[1] will generate this alert. In this case, the extra data is always 24 bytes of zeros.

RFC 793 section 3.4 appears to allow allow data in SYN packets, saying "this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state)". In fact, it appears to be the only time that data is permitted in the packet without the ACK bit also being set. This can conceivably be used for benign purposes as it can reduce the latency of short-lived TCP connections. Thus is not anomolous traffic . . . it may be rare, but it is a legimately crafted packet.

The second signature, which matched on the same packets, was `DATA ON SYN detection`. It is generated by the Snort pre-processor which notes: "detect_state_problems: turns on alerts for stream events of note, such as evasive RST packets, *data on the SYN packet* [emphasis mine], and out of window sequence numbers". The

[1] 6 bytes or greater rather than 1 byte or greater because of the minimum frame size for ethernet

assumption appears to be that data on SYN is unusual enough to note, but not inherently malicious. Again, this rare but plausibly legimate.

As I discuss in the next section, the traffic is anomolous but rather is a false positive.

5.5 Attack Mechanism

I wasn't able to find a signature that matched this behavior of data in a SYN packet to TCP port 53 (DNS, typically). However, a posting by Laurie Zirkle to the Snort-users mailing list[2] implies that it's a benign-though-annoying DNS probe:

> "This system is a 3DNS box from F5. It performs data center load balancing by trying to determine which data center you are closest to and routes you there. It does this through some pretty strange and intrusive ways and it looks like this box was not brought up in one of the approved configurations. Pounding port 53 is one of the intrusive things the product is know to do. I've passed it on to the folks who created that approved configuration and police the misconfigured boxes."

Additionally, it appears that Microsoft's `windowsupdate.com` does the same thing. A post to the Incidents@securityfocus.com mailing list by Ryan W. Maple[3] says that:

"It's a global traffic director location probe thing. They want to figure out which server(s) are closest to you. When you or one of your users does a DNS request to them, it will had back an answer that is supposed to be the best performer for you.

¿ Remote operating system guess: F5labs Big/IP HA TCP/IP Load Balancer (BSDI kernel/x86)

..One brand of that type of product is the Big IP from F5."

It's possible that the original IP address (before obfuscation) resolved to that domain. It's also possible that Microsoft it using whatever

This is supported by other posts the same mailing list, including one by Matt Kettler[4] that sums up my own conclusion about these packets:

> "Since the syn packet contains nothing but null bytes I'm not too worried about them. It strikes me as something more likely generated by an broken, misbehaved, or strangely written DNS package than any kind of attack or trojan."

[2] Archived at http://archives.neohapsis.com/archives/snort/2002-01/0355.html
[3] Archived at http://cert.uni-stuttgart.de/archive/incidents/2001/02/msg00341.html
[4] Archived at http://www.mcabee.org/lists/snort-users/Dec-01/msg00613.html

5.6 Correlations

- The Snort-users mailing list thread with the subject line “BAD TRAFFIC data in TCP SYN packet” begun by Lars Jorgensen on Monday, January 14, 2002 (00:39:17 CST). In this thread it was pointed out that the packets, which match the signature analyzed here, are coming from `*.windowsupdate.com` (a known F5 user). Dewey Paciaffi referred the discussion to a CERT organization (`cert.uni-stuttgart.de`) to explain how the traffic is a way for a network (like Microsoft) to determine the best server to use to service a request. Unfortunately, the link is now dead. The last mail in the thread was from Laurie Zirkle and provided an explanation of the 3DNS box from F5 and how it produces this signature in the course of it’s normal operation.
- The Incidents@securityfocus.com mailing list thread with the subject line “Probes from Microsoft” begun by Ryan W. Maple on Friday, February 23, 2001 (19:53:53 -0500). Responses included one from Ryan Russell who reported that “a global traffic director location probe” and “one brand of that type of product is the Big IP from F5”.
- The Snort-users mailing list thread with the subject line “Re: [Snort-users] Incident Identification (data in TCP syn packet)”[5] begun by Matt Kettler on Wednesday, December 26, 2001 (16:46:55 -0500).

5.7 Evidence of active targeting

Following the analysis that that this is not an active attack but is rather the result of some Internet load balancing technology.

5.8 Severity

Severity =
(Criticality & Lethality) – (System Countermeasures & Network Countermeasures)

The scan is intended for a DNS server and DNS servers are highly critical pieces of a network infrastructure. Trust relationships often exist based on the resolution of hostnames, such as the location of a Kerberos authentication server. Man-in-the-middle attacks are made much easier with a compromised DNS server. ISC BIND is listed as the top Unix vulnerability on the SANS Top 20 list[6]. Thus the criticality must be estimated as high to be safe. *Criticality = 4*.

The lethality of the version scan itself is quite low, as it is a false positive. It can consume some incident response time while being investigated. *Lethality = 1*.

[5] As it was a subject change on an existing thread, the “Re:” really is the beginning of this thread.
[6] http://www.sans.org/top20/

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

The actual state of system countermeasures are unknown. However DNS servers can be secured through techniques like jails and are usually redundant (a form of fault tolerance). Assuming standard security measures have been taken, this implies a reasonable level of available countermeasures. *System countermeasures = 4*.

The actual state of network countermeasures are unknown, except that snort was running to capture the attack (which is a good sign!). Firewalls are effective with this type of traffic because restricting TCP SYN access to port 53 (the signature in question) on a DNS server doesn't affect name resolving for the majority of cases. Assuming that DNS is served from a highly restricted subnet hung off the firewall, this gives the ability to completely eliminate this type of traffic from the view of the DNS server (false positive or not). In addition, being able to detect the signature with snort provides the ability to blacklist scanning hosts (or whitelist known F5 users) if desired. *Network countermeasures = 5*.

The overall severity of these detected signatures is –4.

At worst, it's an unusual way to scan for DNS servers (though it would be unusual for an attacker to pick a method that is *more* like to trigger an IDS). This detect is considered a false positive.

5.9 Defensive recommendations

Many sites will not allow TCP requests to their DNS servers (excepting specific DNS servers that they exchange zones with), a practice which may interfere with this load balancing technology and cause this sort of false positive. Tuning the alerts to eliminate known F5 users will allow this rule to still alert on other data-in-SYN packets, which will still require individual investigation.

The SANS Top 20 list contains a good discussion of securing an ISC BIND server. At the host level, Bind should only be run on designated DNS hosts and should be maintained & patched vigilantly. Seperate DNS servers for internal and external networks can help limti exposure. Bind 9 should be preferred to Bind 8. Jailing (where available) or chrooting Bind can help contain and limit an intruder.

At the network level, DNS service can be run from a highly-restricted secure subnet off the firewall. The version string can be hidden or faked in the configuration file, and zone transfers can be restricted to designated secondaries.

5.10 Multiple choice question

Given a TCP SYN packet being sent to your DNS server on port 53, from a large and well-known Internet site, that carries a data payload of 24 null bytes, what is the most likely explanation for the packet?

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

1. A DNS zone transfer attempt
2. A misconfigured DNS server: small requests should use UDP
3. A DNS probe or exploit involving null bytes
4. A load-balancing mechanism is trying to decide what server is closest to you

Answer: “A load-balancing mechanism is trying to decide what server is closest to you”. The second and third options are plausible red herrings.

Analyze This 6

6.1 Overview

These are the results of a SANS/GCIA intrusion detection system audit of your University network. While hands-on access to the network would provide more detailed results, some interesting discoveries and defensive recommendations can be made. Your investment in running a Snort intrusion detection system enabled this audit and your organization is to be commended on that. While universities thrive on open information, it is unfortunately true that an open infrastructure is vulnerable to abuse.

> Intrusion detection is needed in today's IT environment because it is impossible to keep pace with current and potential threats and vulnerabilities in IT systems. The IT environment is constantly evolving and changing fueled by new technology and the Internet. To make matters worse, threats and vulnerabilities in this environment are also constantly evolving. Every new technology, product, or system brings with it a new generation of bugs and unintended conflicts or flaws. [1]

The following (summarized) discoveries were made:

- IRC seems to be a major issue: It has a custom rules written for it which are triggered often. However, the network does not have an IRC server, just clients. Section 6.5 shows the custom alerts related to IRC that were discovered.
- There are some compromised Linux hosts, affected by the "Red Worm" backdoor. The list of affected hosts can be found in section 6.5.1.
- There are internal services that are available to the Internet. The most critical one is an open LPD server, discussed in section 6.4.10.

The following (summarized) defensive recommendations were made:

- Perform a full audit of logs, policy and infrastructure design, with "hands-on" capabilities given to the auditor(s).

[1] *Planning Concerns, Considerations, and Tips for IDS in Federal IT Systems*, March 30/2001, http://www.sans.org/infosecFAQ/intrusion/fed_IT.htm

- *Huge* amounts of data are being produced by your intrusion detection system. Efforts to reduce alerts by eliminating false positives combined with efforts to automate the analysis and trending of the data would simplify the efforts of your intrusion analysts.
- Running an IRC server (for MY.NET clients only) that can be properly secured and controlled by the University might be one way of mitigating the risk.
- Clean up the compromised Linux hosts
- Restrict access to all internal services that are currently available to the Internet
- Installing and running something like Argus[2] would make traffic analysis much easier for your security staff.
- policy enforcement and incident response (automated, where possible) should be key goals going forward.

6.2 Analyzed Files

Files were obtained from `http://isc.sans.org/logs/`. The date range chosen was March 25 through to 29, 2004.

Three log formats are represented, all generated by Snort. One contains Alerts, the next contains port scans, and the last contains out-of-spec packets. The uncompressed form of the files looks like this:

```
1  -rw-r--r--  1 tillman  tillman        12M Mar 29 04:03 alert.040325
2  -rw-r--r--  1 tillman  tillman        24M Mar 30 04:03 alert.040326
3  -rw-r--r--  1 tillman  tillman        33M Mar 31 04:02 alert.040327
4  -rw-r--r--  1 tillman  tillman        30M Apr  1 04:02 alert.040328
5  -rw-r--r--  1 tillman  tillman        14M Apr  2 04:01 alert.040329
6  -rw-r--r--  1 tillman  tillman         1M Mar 29 04:04 oos_report_040325
7  -rw-r--r--  1 tillman  tillman         1M Mar 30 04:03 oos_report_040326
8  -rw-r--r--  1 tillman  tillman         1M Mar 31 04:03 oos_report_040327
9  -rw-r--r--  1 tillman  tillman       952K Apr  1 04:03 oos_report_040328
10 -rw-r--r--  1 tillman  tillman       240K Apr  2 04:01 oos_report_040329
11 -rw-r--r--  1 tillman  tillman        99M Mar 29 04:04 scans.040325
12 -rw-r--r--  1 tillman  tillman       185M Mar 30 04:03 scans.040326
13 -rw-r--r--  1 tillman  tillman       367M Mar 31 04:03 scans.040327
14 -rw-r--r--  1 tillman  tillman       542M Apr  1 04:03 scans.040328
15 -rw-r--r--  1 tillman  tillman       167M Apr  2 04:01 scans.040329
```

[2] http://www.qosient.com/argus/

The logs files consume a total of 1.4GB of disk space, averaging 296MB per day(!).

These files were combined into several large files named `alerts`. `scans` and `oos` in order facilitate working on them collectively.

6.3 Traffi c and Network Analysis

Discovering what services were *supposed* to be running on the monitored network was an important first step as it provides a baseline for what normal traffic should look like. It can also identify which alerts might be false positives.

The subnet that occurred the most frequently was `MY.NET.0.0/?`. Combining this with the custom Snort alerts (which have “UMBC” pre-pended to their name) and the hint that this is a University, this likely corresponds to the following summarized WHOIS information:

```
1 OrgName:     University of Maryland Baltimore County
2 NetRange:   130.85.0.0 - 130.85.255.255
3 CIDR:       130.85.0.0/16
4 NameServer: UMBC5.UMBC.EDU
5 NameServer: UMBC4.UMBC.EDU
6 NameServer: UMBC3.UMBC.EDU
```

Many of the IPs in the `scans` file also correspond to that IP range, supporting this belief.

6.4 Computer Relationships

Much information about the network infrastructure can be gleaned from the logs provided.

6.4.1 IRC servers

In spite of many alerts for the IRC service being generated, MY.NET does not appear to contain an IRC server as all the alerts point to outside IP addresses.

6.4.2 Databases

Ports for Oracle, MySQL, Postgres and MS SQL (1521, 3306, 5432 and 1433/1434 respectively) were checked in the logs. While scans were seen, none appeared to be successful. There is no conclusive evidence of any database servers in MY.NET.

68.48.90.101 appears to be an external Oracle server that is alerted on 10 times, always to MY.NET.30.4:51443.

68.48.90.101 appears to be an external MS SQL server that is alerted on 10 times, always to MY.NET.30.4:51443.

MY.NET.30.3 possibly contains a MySQL server but since the origin is an outside IP address and there isn't any evidence that it was a successful connection, the evidence is weak:

```
alert.040329:03/29-06:02:39.309269  [**] MY.NET.30.3 activity [**]
  217.162.121.111:4191 -> MY.NET.30.3:3306
alert.040329:03/29-06:02:39.315777  [**] MY.NET.30.4 activity [**]
  217.162.121.111:4192 -> MY.NET.30.4:3306
alert.040329:03/29-06:02:39.795945  [**] MY.NET.30.4 activity [**]
  217.162.121.111:4192 -> MY.NET.30.4:3306
```

6.4.3 Routers

traceroute or CDP from one of the routers would be easiest way to map out the routers, so it's unfortunate that information like that wasn't available. Routers are typically associated with TFTP and often have an IP at either end of a subnet. Assuming that the /16 has been further split into /24s, .1 and .254 are likely locations for a router. The "External TCP connection to internal tftp server" alert can also help with this because it shows where others have possibly expected to find routers acting as a TFTP server. Combining that information yields: MY.NET.12.1, MY.NET.30.1, and MY.NET.34.1.

These results are very tentative and obviously not comprehensive, but would be a good place to look first for routers if "hands-on" access to the University network becomes available. Having tcpdump traces that include MAC addresses would have been very helpful, as routers can be much more accurately identified that way.

6.4.4 Firewalls

No firewall appears to be in place: All IPs are from public ranges and connections don't appear to be shunted-off to a single chokepoint. It's possible that a transparent bridging firewall is in use, but such a firewall isn't well suited to high-bandwidth networks like a University. There is also not much evidence that obviously bad traffic is being blocked.

6.4.5 TFTP servers

One way to look for TFTP servers is to look for TCP connections to the TFTP server. Unlike UDP, these require a connected state and so eliminate the most

common forms of false positive

```
# grep "External TCP connection to internal tftp server" alerts |
  grep "> MY.NET" | awk '{print \$15}' | awk -F # : '{print \$1}' |
  sort | uniq | sort
MY.NET.111.34
MY.NET.12.1
MY.NET.12.2
MY.NET.12.3
MY.NET.12.4
MY.NET.12.6
MY.NET.12.7
MY.NET.12.9
MY.NET.153.149
MY.NET.190.97
MY.NET.24.15
MY.NET.25.10
MY.NET.25.68
MY.NET.30.1
MY.NET.30.10
MY.NET.30.2
MY.NET.30.5
MY.NET.30.6
MY.NET.30.7
MY.NET.30.8
MY.NET.30.9
MY.NET.34.1
MY.NET.34.10
MY.NET.34.11
MY.NET.34.12
MY.NET.34.13
MY.NET.34.14
MY.NET.34.15
MY.NET.34.16
MY.NET.34.17
MY.NET.34.19
MY.NET.34.2
MY.NET.34.20
MY.NET.34.21
MY.NET.34.22
MY.NET.34.23
MY.NET.34.24
MY.NET.34.25
MY.NET.34.26
MY.NET.34.3
MY.NET.34.4
MY.NET.34.5
MY.NET.34.6
MY.NET.34.7
MY.NET.34.8
MY.NET.53.225
MY.NET.82.117
MY.NET.82.15
```

```
52 MY.NET.84.194
53 MY.NET.97.35
```

6.4.6 DNS Servers

In the various `alert.*` logs there were 492 instances of port 53 being the destination (as shown by `grep :53\$ alert* | wc -l`). 381 one these show MY.NET.3.1 as the destination, 66 show MY.NET.4.1, and 33 show MY.NET.5.1. It is likely that these IP addresses represent the DNS servers for the MY.NET network.

Incredibly, the various `scans.*` logs show 4,181,771 packets where 130.85.1.3:53 UDP was the destination. Only a mere 74 packets were destined for 130.85.1.4:53 UDP, an 67 for 130.85.1.5:53 UDP. Most of source IPs are from outside MY.NET. These facts strengthen the belief that 130.85.1.3 is the primary external DNS server, and 130.85.1.4 and .130.85.1.4 are low-load secondaries.

6.4.7 SMTP servers

58953 instances of port 25 being the destination (as shown by `grep ":25 SYN" scans.040325* | wc -l`).

IP addresses of likely SMTP servers (as well as the number of times they were seen in the logs) include 130.85.34.14 (25373), 130.85.25.69 (9789), 130.85.25.70 (8909), 130.85.25.71 (6302), 130.85.25.66 (2036), 130.85.25.67 (1873), 130.85.25.73 (1826), 130.85.25.68 (1126), 130.85.12.6 (991) and 130.85.97.73 (367).

6.4.8 Web servers

Any of the rules in the `web-*.rules` rulesets would help identify web servers. Because HTTP is TCP based, these signatures are less prone to the common false positive scenarios. All the rules contain “WEB” as part of their name, so that short word is worth searching for.

While many external web servers were identified by the “IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize” signature, using `grep -i web alerts | grep 130.85` did not identify any web server in MY.NET address space. Checking the combined `scans` also did not reveal anything beyond SYN packets. This is unusual: a University network would be expected to have web servers. It’s possible that Snort has been tuned in a way that this information is not available.

Looking through the `alert.*` logs, the top 10 port 80 destinations in the MY.NET network can be isolated:

```
1 # grep ":80 SYN" scans.040325* | awk '{print \$6}' | grep 130.85 | sort | uniq -c | sort -r | head -
2   60 130.85.24.44:80
```


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

```
  24 130.85.34.11:80
  22 130.85.29.3:80
  14 130.85.97.72:80
  13 130.85.6.7:80
   7 130.85.97.29:80
   5 130.85.100.165:80
   4 130.85.97.87:80
   4 130.85.60.14:80
   4 130.85.29.13:80
```

The numbers are quite low and do not provide a lot of confidence that any of the IPs can be firmly identified as web servers

6.4.9 FTP servers

Any of the rules in `ftp.rules` ruleset would help identify FTP servers. Because FTP is TCP based, these signatures are less prone to the common false positive scenarios. All the rules contain "FTP" as part of their name, so that short word is worth searching for.

And, indeed, several FTP servers turn out prominently: MY.NET.153.81, MY.NET.24.47, MY.NET.53.29, MY.NET.70.5, MY.NET.24.27, and MY.NET.70.49.

Three custom signatures appear to specify FTP servers even further: "External FTP to HelpDesk MY.NET.53.29", "External FTP to HelpDesk MY.NET.70.5" and "External FTP to HelpDesk MY.NET.70.49".

6.4.10 LPD servers (print hosts)

One alert, "connect to 515 from outside [**] 68.32.127.158:797 -> MY.NET.24.15:515" occurs 13328 times. This host is definitely an LPD server, one that is used extremely often from `68.32.127.158`. That IP corresponds to host in the `comcast.net` network and could either be a home user accessing the printer at the University, or (more likely due to the extreme frequency) could be an attack of some kind. The LPD service, which is inherently insecure, is normally not allowed to external hosts.

6.4.11 Windows Domain Controllers

There were no alerts involving port 88 (Kerberos) or 389 (LDAP), which indicates that Windows 2000 domains are likely not in use. However, there are many many alerts involving port 137 (NT domains). The 6 heaviest sources of port 137 traffic are possibly domain controllers:

```
# grep ":137 " alerts | awk '{print \$7}' | sort | uniq -c
  | sort -r | head -8
2078 MY.NET.11.7:137
 609 MY.NET.75.13:137
 299 MY.NET.190.92:137
 216 MY.NET.5.34:137
 195 MY.NET.29.30:137
 117 MY.NET.111.228:137
 108 MY.NET.150.44:137
  99 MY.NET.150.198:137
```

6.4.12 DHCP servers

No alerts on ports 67 or 68 were found. This is unusual: a University network would be expected to have web servers. It's possible that Snort has been tuned in a way that this information is not available, or that there were simply no attacks for this service.

6.5 Categorized List of Detects

Over the five day period a total of 108233 detects were seen in the "alerts" files. They are shown below sorted by category and listed in decreasing order of occurrence[3]:

Snort Alert	Number
High port 65535 tcp - possible Red Worm - traffic	13806
High port 65535 udp - possible Red Worm - traffic	1114
Possible trojan server activity	373
NIMDA - Attempt to execute cmd from campus host	56
DDOS shaft client to handler	30
DDOS mstream client to handler	7
NIMDA - Attempt to execute root from campus host	2

Table 6.2: Worms, DDoS, trojans

[3]snort_sort.pl was used to generate much of the raw information that went into this table.

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

Snort Alert	Number
MY.NET.30.3 activity	29841
MY.NET.30.4 activity	21077
(UMBC NIDS IRC Alert) IRC user /kill detected, possible trojan.	593
(UMBC NIDS IRC Alert) Possible sdbot floodnet detected attempting to IRC	459
(UMBC NIDS) External MiMail alert	283
(UMBC NIDS IRC Alert) Possible drone command detected	17
(UMBC NIDS IRC Alert) Possible Incoming XDCC Send Request Detected.	9
(UMBC NIDS IRC Alert) User joining XDCC channel detected. Possible XDCC bot	3
External FTP to HelpDesk MY.NET.53.29	6
External FTP to HelpDesk MY.NET.70.50	5
External FTP to HelpDesk MY.NET.70.49	2
(UMBC NIDS) Internal MiMail alert	1

Table 6.4: Custom UMBC alerts

Snort Alert	Number
FTP DoS ftpd globbing	358
FTP passwd attempt	129
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize	88
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	3
DOS Real Server template.html	1

Table 6.6: FTP and Web related

Snort Alert	Number
ICMP SRC and DST outside network	90
TFTP - Internal UDP connection to external tftp server	8
External RPC call	6

Table 6.8: Stuff heading outside that shouldn't

Snort Alert	Number
connect to 515 from outside	13758
TFTP - External TCP connection to internal tftp server	172

Table 6.10: Stuff attempting to come inside that shouldn't be

Snort Alert	Number
Null scan!	2219
NMAP TCP ping!	898
SYN-FIN scan!	24
Probable NMAP fingerprint attempt	13

Table 6.12: Scans and probes

Snort Alert	Number
EXPLOIT x86 NOOP	10170
EXPLOIT x86 NOPS	192
EXPLOIT x86 setuid 0	34
EXPLOIT x86 setgid 0	27
EXPLOIT x86 stealth noop	10
EXPLOIT NTPDX buffer overflow	9

Table 6.14: Shellcode containing exploit

Snort Alert	Number
Incomplete Packet Fragments Discarded	5391
SUNRPC highport access!	397
Tiny Fragments - Possible Hostile Activity	280
TCP SRC and DST outside network	136
TCP SMTP Source Port traffic	74
Attempted Sun RPC high port access	33
Traffic from port 53 to port 123	3

Table 6.16: Stuff that shouldn't happen (suspicious)

Snort Alert	Number
SMB Name Wildcard	5772
SMB C access	125
IRC evil - running XDCC	56
RFB - Possible WinVNC - 010708-1	48

Table 6.18: Misc

The tables above shows the alerts produced over the five day period, categorized by the auditor.

6.5.1 Prominent alerts

Items that stand out from the crowd include:

Red Worm

Alerts for "possible Red Worm - traffic" figure prominently, with 14920 alerts between UDP and TCP.

This is not the similarly named Code Red worm but rather is a Linux-specific backdoor with listens on port 65535. It is also known as the Adore worm, and is similar to the Ramen and Lion Linux worms. It operates by scanning hosts for vulnerabilities in LPRng, rpc-statd, wu-ftpd and BIND then exploiting them and installing a backdoor that provides remote access. Because the worm scans hosts and attacks them in order to spread, the impact of a compromised host can be high. SANS has an information page at http://www.sans.org/y2k/adore.htm which discusses the worm in detail.

Hosts matching this signature are almost certainly compromised.

Adore appears to have started spreading on April 1 2001, making this an old worm. Compromised hosts are likely older and unmaintaiend Linux installations. SANS provides a tool, Adorefind, that can be used to clean an affected system.

Hosts internal to MY.NET were seen connecting *outwards* to port 65535, which may indicate that some internal users are breaking “acceptable use” policies. The list of such accesses, including a count, follows:

```
# grep "Red Worm" alerts | awk '{print \$14 \$15 \$16}'
  | grep \^MY | grep 65535\$ | sort | uniq -c | sort -r
5022 MY.NET.97.82:1122->80.181.112.186:65535
 847 MY.NET.53.111:3658->66.118.165.120:65535
 487 MY.NET.110.72:12203->64.112.193.187:65535
  80 MY.NET.60.17:110->68.55.62.110:65535
  21 MY.NET.24.44:80->167.102.229.26:65535
  14 MY.NET.150.83:80->66.151.181.4:65535
  13 MY.NET.29.3:80->68.165.246.13:65535
  13 MY.NET.24.74:443->64.139.69.43:65535
  10 MY.NET.84.235:4672->213.37.240.225:65535
  10 MY.NET.24.34:80->128.231.88.5:65535
  10 MY.NET.24.20:25->62.253.164.43:65535
  10 MY.NET.12.6:25->64.134.101.78:65535
   9 MY.NET.84.235:5877->213.37.240.225:65535
   9 MY.NET.84.235:5677->81.182.63.210:65535
   9 MY.NET.24.58:443->68.32.54.10:65535
   8 MY.NET.70.164:4662->62.243.101.59:65535
   8 MY.NET.42.3:6257->220.43.148.149:65535
   7 MY.NET.24.74:443->151.196.12.50:65535
   6 MY.NET.6.62:65535->69.140.137.209:65535
   6 MY.NET.24.34:80->67.72.26.151:65535
   6 MY.NET.12.6:25->64.12.138.18:65535
   5 MY.NET.84.235:5877->81.67.233.192:65535
   5 MY.NET.6.7:80->158.103.0.1:65535
   5 MY.NET.24.34:80->68.55.194.137:65535
   5 MY.NET.24.34:80->24.199.246.106:65535
   4 MY.NET.5.20:80->66.58.241.175:65535
   4 MY.NET.42.3:6257->219.31.84.156:65535
   4 MY.NET.34.11:80->64.68.82.144:65535
   4 MY.NET.112.192:11028->82.64.166.117:65535
   3 MY.NET.84.235:5877->80.25.230.157:65535
   3 MY.NET.84.235:5877->80.24.159.82:65535
   3 MY.NET.29.3:80->68.33.51.13:65535
   3 MY.NET.24.48:443->68.32.54.10:65535
   2 MY.NET.97.52:4672->82.48.41.145:65535
   2 MY.NET.97.52:4672->81.50.2.123:65535
   2 MY.NET.84.235:4672->82.65.155.157:65535
   2 MY.NET.84.235:4672->80.24.159.82:65535
   2 MY.NET.70.207:12203->208.187.67.142:65535
   2 MY.NET.5.20:80->209.165.168.2:65535
   2 MY.NET.42.3:6257->171.75.116.134:65535
```

```
   2 MY.NET.12.7:443->68.55.192.115:65535
   2 MY.NET.12.4:143->68.55.192.221:65535
   2 MY.NET.11.4:20->65.88.98.1:65535
   1 MY.NET.97.55:6112->209.102.157.22:65535
   1 MY.NET.97.52:4672->82.65.174.122:65535
   1 MY.NET.97.52:4672->80.138.81.83:65535
   1 MY.NET.97.52:4672->207.248.43.87:65535
   1 MY.NET.97.168:4672->81.185.253.77:65535
   1 MY.NET.84.235:4672->81.49.191.102:65535
   1 MY.NET.70.229:2612->81.10.33.243:65535
   1 MY.NET.70.229:2612->64.53.153.88:65535
   1 MY.NET.70.207:12300->62.203.139.98:65535
   1 MY.NET.60.16:80->199.188.42.65:65535
   1 MY.NET.6.7:80->64.68.82.44:65535
   1 MY.NET.42.3:6257->82.48.94.152:65535
   1 MY.NET.24.74:443->69.139.236.8:65535
   1 MY.NET.24.34:80->68.48.109.217:65535
   1 MY.NET.24.20:113->154.33.63.98:65535
   1 MY.NET.153.81:94->143.205.246.112:65535
   1 MY.NET.12.6:25->65.208.170.18:65535
   1 MY.NET.110.72:12300->4.64.10.46:65535
```

The following are the hosts in MY.NET that are likely compromised with Red Worm, along with a count of accesses to them:

```
# grep "Red Worm" alerts | awk '{print \$16}' | grep MY
  | grep 65535 | sort | uniq -c | sort -r
  47 MY.NET.25.12:65535
  30 MY.NET.34.14:65535
  26 MY.NET.25.67:65535
  17 MY.NET.25.69:65535
  17 MY.NET.25.68:65535
  14 MY.NET.25.70:65535
  12 MY.NET.25.10:65535
   9 MY.NET.25.71:65535
   6 MY.NET.153.149:65535
   4 MY.NET.24.20:65535
   1 MY.NET.97.50:65535
   1 MY.NET.34.5:65535
   1 MY.NET.25.66:65535
   1 MY.NET.190.92:65535
   1 MY.NET.147.248:65535
   1 MY.NET.12.6:65535
```

There doesn't appear to be any MY.NET to MY.NET Red Worm traffic.

connect to 515 from outside

The SANS Top 20 list, available at http://www.sans.org/top20/, lists LPD as one of the services that should be blocked at the perimeter in Appendix A. Unlike remote

access to applications, remote access to hardware (like a printer) is rarely justified. Combined with the primitive IP address based security model in the LPD service and a history of remotely exploitable vulnerabilities[4], this service should not be considered safe to expose to the Internet.

The alert "connect to 515 from outside [**] 68.32.127.158:797 -> MY.NET.24.15:515" occurs 13328 times over this 5 day period. While no alerts specific to LPD itself were generated, this is suspicious behavior.

shellcode exploits

The shellcode alerts are fired when packets that contain executable code designed to spawn a shell as their payload are detected. Thus, this is typically the payload for a buffer overflow exploit.

The top ten source IPs for this traffic are:

```
# grep x86 alerts | awk '{print \$7}' | sed 's/:[0-9]*//' | sort | \
    uniq -c | sort -r | head
 614 141.157.60.104
 334 211.96.243.1
 227 130.13.111.49
 183 130.13.154.54
 176 61.175.223.60
 161 130.234.200.141
 148 66.161.196.103
 136 130.88.177.47
 126 130.240.193.238
 124 130.13.128.195
```

And the top ten destination IPs were:

```
# grep x86 alerts | awk '{print \$9}' | sed 's/:[0-9]*//' | sort | \
    uniq -c | sort -r | head
 795 MY.NET.17.4
 388 MY.NET.17.3
 252 MY.NET.53.84
 216 MY.NET.15.227
 188 MY.NET.32.165
 140 MY.NET.83.98
 137 MY.NET.150.44
 119 MY.NET.24.8
 118 MY.NET.17.2
 116 MY.NET.190.95
```

[4]CVE Candidate CAN-1999-0061 and CERT Advisory CA-2001-30 are some examples

It is noteworthy that all shellcode exploits (10433 of them) are for x86 class computers. This implies that other architectures are not being exploited, not being covered by snort signatures, or do not exist.

MY.NET.30.3 and 30.4 "activity"

MY.NET.30.3 and MY.NET.30.4 activity: These custom alerts generated very high counts (50918 between the two), but don't specify what they're alerting on beyond "activity". These may be simple honeypots, where *any* activity to them is considered suspicious. It's also possible that this is a custom signature for statistical purposes.

130.85.30.3 and 130.85.30.4 appear extensively in the `scans` log. Listing the top destination ports that incoming connections to those IPs used doesn't show a pattern that would indicate a "real" production host:

```
# grep 130.85.30.3: scans | awk '{print \$6}' | sed 's/:/ /' | \
     awk '{print \$2}' | sort | uniq -c | sort -r | head
  20 6129
  11 4899
  11 20168
   6 4000
   6 21
   4 80
   4 5900
   4 554
   3 666
   3 6112
# grep 130.85.30.4: scans | awk '{print \$6}' | sed 's/:/ /' | \
     awk '{print \$2}' | sort | uniq -c | sort -r | head
  20 6129
  11 4899
  11 20168
   8 21
   7 4000
   5 80
   5 5900
   4 554
   3 666
   3 6112
```

These ports tend to correspond to remote administration tools. For example, the top ports listed correspond to Dameware remote administration, Radmin, and the W32/Lovegate backdoor worm. This supports the hypothesis that this is a honeypot box.

IRC user /kill detected

A typical alert looks like this:

```
03/25-00:59:18.404467  [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.
 [**] 64.85.20.76:6667 -> MY.NET.97.66:2853
```

Looking a little closer into the IRC alerts, we find that a small number of hosts is sending data that is causing all the fuss:

```
# grep "IRC user" alerts | awk '{print \$14}' | sort | uniq -c | sort -r | head -9
 457 139.165.206.128:6666
  26 209.126.201.103:6667
  13 61.6.39.100:6667
  12 205.177.13.100:7000
   9 199.184.165.133:6667
   7 206.252.192.194:6667
   6 209.123.78.250:7000
   5 195.140.143.117:6667
   4 216.109.195.222:6667
```

Looking at destinations is also interesting:

```
# grep "IRC user" alerts | awk '{print \$16}' | sort | uniq -c | sort -r | head -9
   7 MY.NET.97.159:3589
   6 MY.NET.97.170:4244
   6 MY.NET.42.4:3099
   5 MY.NET.42.4:3100
   4 MY.NET.97.235:2312
   4 MY.NET.97.209:3065
   3 MY.NET.97.92:3678
   3 MY.NET.97.185:3468
   3 MY.NET.97.102:2672
```

This implies that the IRC server is the source of the alerts, which makes sense. This is a University network without it's own IRC server, so while students will be connecting to IRC servers all over the Internet they're unlikely to have "oper" privileges on them. As the `kill` command requires oper status, they will only be seen to be coming from the IRC servers. It is very interesting that a custom snort rule was written to look for this. . . it's possible that it's purpose is to identify users that have a high potential to be causing outgoing problems in the near future (because they're just be `kill`ed, likely for bad behavior on IRC).

Looking for where the clients are coming from with `grep "IRC user" alerts | awk 'print $16' | grep -v MY.NET` shows that they are all from MY.NET, whereas none of the servers are. This information was used when reconstructing the network layout.

6.6 Top Ten Talkers

Note that these lists are the top ten talkers that generated alerts. For a list of top ten talkers that is not dependent on suspicious behavior, a tool like `argus`[5] is much better suited than Snort.

Looking at the log of port scans, we'll find some very talkative IPs (the scanners). This is natural as scanning a large network is a very talkative process. Note that even on a fast computer that parsing this much data takes a significant amount of time. If a real-time list of the top ten talkers category were ever desired, a more efficient way of obtaining it would have to be used.

6.6.1 Scanners – TCP

```
# grep -v UDP scans > talkers.tcp
# cat talkers.tcp | awk '{print \$4}' | awk -F : '{print \$\1}'
  | sort | uniq -c | sort -r | head
9890  130.85.81.64
9877  221.237.160.164
9869  130.85.97.126
9658  130.85.97.159
9477  62.179.205.188
9390  128.210.108.195
8895  202.152.11.196
8751  211.72.48.251
8575  130.85.97.87
8515  35.10.39.90
```

Four of top ten positions are from MY.NET.

Looking at the top talker (`130.85.81.64`) was doing in the `alerts` log, we see:

```
 372 spp_portscan: portscan status from MY.NET.81.64: ...
  23 spp_portscan: PORTSCAN DETECTED from MY.NET.81.64
     (THRESHOLD 12 connections exceeded in 1 seconds)
  22 spp_portscan: End of portscan
   2 NMAP TCP ping! [**]
   2 Incomplete Packet Fragments Discarded
   1 EXPLOIT x86 NOOP [**]
```

6.6.2 Scanners – UDP

```
# grep UDP scans > talkers.udp
# cat talkers.udp | awk '{print \$4}' | awk -F : '{print \$1}'
```

[5] http://www.qosient.com/argus/

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

```
  | sort | uniq -c | sort -r | head
976304 130.85.1.4
93078  130.85.70.207
9559   130.85.97.23
7224   130.85.83.91
7099   130.85.97.71
6096   130.85.112.192
5834   130.85.112.152
5407   130.85.153.31
4934   130.85.97.214
4769   130.85.97.63
```

All the top UDP talkers were from MY.NET. `130.85.1.4` is interesting because it appears to be doing widespread sweeps of hosts all over the Internet, exclusively for DNS and NTP. Additionally, several Internet hosts appear to be scanning it back. This host should be immediately disconnected from the network.

Looking at the top talker (`130.85.81.64`) was doing in the `alerts` log, we see:

```
2503 spp_portscan: PORTSCAN DETECTED from MY.NET.1.4: ...
2216 spp_portscan: End of portscan
123344 spp_portscan: portscan status from from MY.NET.1.4: ...
  70 NMAP TCP ping! [**]
  14 High port 65535 tcp
   2 TCP SRC and DST outside network [**] 192.168.1.47:3181 -> 216.109.127.60:443
   1 connect to 515 from from outside
   1 [UMBC NIDS] External MiMail
   1 MY.NET.30.3 activity [**]
   1 High port 65535 udp 65535 udp - possible Red Worm - traffic
   1 FTP passwd attempt [**]
   1 EXPLOIT x86 NOOP [**]
```

This host is likely compromised with the Red Worm/Adore trojan and is being used by an intruder to perform a port scan. This is a high priority problem to be resolved– see sectin 6.5.1 for more details.

6.7 Five External Suspicious Hosts

6.7.1 68.32.127.158

`68.32.127.158` is suspicious because of it's numerous connections to `MY.NET.24.15:515`, the LPD server. It resolves to pcp01823879pcs.howard01.md.comcast.net. WHOIS confirms that it is a Comcast Cable Communications IP.

WHOIS information:

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

```
CustName:   Comcast Cable Communications, Inc.
Address:    3 Executive Campus
Address:    5th Floor
City:       Cherry Hill
StateProv:  NJ
PostalCode: 08002
Country:    US
RegDate:    2003-03-18
Updated:    2003-03-18

NetRange:   68.32.112.0 - 68.32.127.255
CIDR:       68.32.112.0/20
NetName:    BALTIMORE-A-2
NetHandle:  NET-68-32-112-0-1
Parent:     NET-68-32-0-0-1
NetType:    Reassigned
Comment:    NONE
RegDate:    2003-03-18
Updated:    2003-03-18

TechHandle: IC161-ARIN
TechName:   Comcast Cable Communications Inc
TechPhone:  +1-856-317-7200
TechEmail:  cips_ip-registration@cable.comcast.com

OrgAbuseHandle: NAPO-ARIN
OrgAbuseName:   Network Abuse and Policy Observance
OrgAbusePhone:  +1-856-317-7272
OrgAbuseEmail:  abuse@comcast.net

OrgTechHandle: IC161-ARIN
OrgTechName:   Comcast Cable Communications Inc
OrgTechPhone:  +1-856-317-7200
OrgTechEmail:  cips_ip-registration@cable.comcast.com
```

6.7.2 221.237.160.164

The top external talker, `221.237.160.164` is suspicious for it's widespread scanning. All it does is a simple SYN scan for web servers on port 80, but it does so almost ten thousand times. A sample follows:

```
Mar 29 08:45:06 221.237.160.164:57342 -> 130.85.4.36:80 SYN ******S*
Mar 29 08:45:06 221.237.160.164:57346 -> 130.85.4.40:80 SYN ******S*
Mar 29 08:45:06 221.237.160.164:57344 -> 130.85.4.38:80 SYN ******S*
Mar 29 08:45:06 221.237.160.164:52967 -> 130.85.4.42:80 SYN ******S*
Mar 29 08:45:06 221.237.160.164:57345 -> 130.85.4.39:80 SYN ******S*
Mar 29 08:45:06 221.237.160.164:57343 -> 130.85.4.37:80 SYN ******S*
etc ...
```

221.237.160.164 does not resolve, it appears to not have a reverse DNS entry. WHOIS reports the IP as being from “CHINANET Sichuan province network”, a telecom. An APNIC IP with no reverse is usually suspicious in it’s own right, and follow-up with the IP owner is usually quite difficult.

WHOIS information:

```
inetnum:       221.236.0.0 - 221.237.255.255
netname:       CHINANET-SC
descr:         CHINANET Sichuan province network
descr:         China Telecom
descr:         A12,Xin-Jie-Kou-Wai Street
descr:         Beijing 100088
country:       CN
admin-c:       CH93-AP
tech-c:        CS408-AP
mnt-by:        APNIC-HM
mnt-lower:     MAINT-CHINANET-SC
mnt-routes:    MAINT-CHINANET-SC
remarks:       This object can only modify by APNIC hostmaster
remarks:       If you wish to modify this object details please
remarks:       send email to hostmaster@apnic.net with your
remarks:       organisation account name in the subject line.
changed:       hm-changed@apnic.net 20030910
status:        ALLOCATED PORTABLE
source:        APNIC

role:          CHINANET SICHUAN
address:       No.72,Wen Miao Qian Str Chengdu SiChuan PR China
country:       CN
phone:         +86-28-86190657
fax-no:        +86-25-86190641
e-mail:        ipadmin@my-public.sc.cninfo.net
trouble:       send anti-spam reports to anti-spam@mail.sc.cninfo.net
trouble:       send abuse reports to security@mail.sc.cninfo.net
trouble:       times in GMT+8
admin-c:       YZ43-AP
tech-c:        RL357-AP
tech-c:        XS16-AP
nic-hdl:       CS408-AP
remarks:       noc.cd.sc.cn
notify:        ipadmin@my-public.sc.cninfo.net
mnt-by:        MAINT-CHINANET-SC
changed:       zhangys@mail.sc.cninfo.net 20030318
source:        APNIC

person:        Chinanet Hostmaster
address:       No.31 ,jingrong street,beijing
address:       100032
country:       CN
phone:         +86-10-66027112
fax-no:        +86-10-58501144
```

```
e-mail:       hostmaster@ns.chinanet.cn.net
e-mail:       anti-spam@ns.chinanet.cn.net
nic-hdl:      CH93-AP
mnt-by:       MAINT-CHINANET
changed:      hostmaster@ns.chinanet.cn.net 20021016
remarks:      hostmaster is not for spam complaint,please send spam complaint to anti-spam@ns.chinan
source:       APNIC
```

6.7.3 62.179.205.188

Number five on the top TCP talkers list, `62.179.205.188` resolves to vs85p81.cm.chello.no. WHOIS reports that it is owned by HAUGESUND-CUSTOMERS-CABLE. It made the list of suspicious hosts for SYN scanning for port 6129:

```
Mar 28 22:46:02 62.179.205.188:3940 -> 130.85.1.2:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3941 -> 130.85.1.3:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3942 -> 130.85.1.4:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3944 -> 130.85.1.6:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3945 -> 130.85.1.7:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3946 -> 130.85.1.8:6129 SYN ******S*
Mar 28 22:46:02 62.179.205.188:3947 -> 130.85.1.9:6129 SYN ******S*
etc ...
```

Port 6129 TCP is used by the Dameware remote administration software. There is a vulnerability in older versions which allow unauthorized login. Dameware even was installed by some viruses for the purpose of “remote administration” of the infected system.

WHOIS information:

```
inetnum:      62.179.200.0 - 62.179.215.255
netname:      HAUGESUND-CUSTOMERS-CABLE
descr:        UPC Norway
descr:        Customers Haugesund
country:      NO
admin-c:      NKH4-RIPE
tech-c:       HMCB1-RIPE
status:       ASSIGNED PA
remarks:      Contact abuse@chello.no concerning criminal
remarks:      activities like spam, hacks, portscans
notify:       hostmaster@chello.at
mnt-by:       CHELLO-MNT
changed:      hostmaster@chello.at 20040406
source:       RIPE

route:        62.179.128.0/17
descr:        AT-TELEKABEL-20000918
```

```
descr:          Chello Norway
origin:         AS6830
mnt-by:         CHELLO-MNT
changed:        hostmaster@chello.at 20011217
source:         RIPE

role:           Hostmaster Chello Broadband
address:        UPC Technology
address:        Internet Services
address:        Erlachplatz 116
address:        A-1100 Vienna
address:        Austria
phone:          +43 1 96068 5000
fax-no:         +43 1 96068 5666
e-mail:         hostmaster@chello.at
admin-c:        AK991-RIPE
tech-c:         SB666-RIPE
tech-c:         MS2509-RIPE
tech-c:         AK991-RIPE
nic-hdl:        HMCB1-RIPE
notify:         hostmaster@chello.at
mnt-by:         CHELLO-MNT
changed:        hostmaster@chello.at 20040204
source:         RIPE

person:         Norbert K Hinna
address:        chello broadband as
address:        maridalsveien 323
address:        N-0872 OSLO
address:        Norway
phone:          +47 21 90 62 00
fax-no:         +47 21 90 00 01
e-mail:         norbert@chello.no
nic-hdl:        NKH4-RIPE
changed:        norbert@chello.no 20020206
source:         RIPE
```

6.7.4 128.210.108.195

Number six on the top TCP talkers list, `128.210.108.195` resolves to treelab1.fnr.purdue.edu. That's notable as it appears to be an administration maintained host at another University, likely compromised. WHOIS confirms that it is an IP owned by Purdue (PURDUE-CCNET). It made the list of suspicious hosts for SYN scanning for port 6129 (just as the previous entry was).

```
Mar 28 19:16:54 128.210.108.195:2135 -> 130.85.1.1:6129 SYN ******S*
Mar 28 19:16:54 128.210.108.195:2136 -> 130.85.1.2:6129 SYN ******S*
Mar 28 19:16:55 128.210.108.195:2138 -> 130.85.1.4:6129 SYN ******S*
Mar 28 19:16:55 128.210.108.195:2140 -> 130.85.1.6:6129 SYN ******S*
Mar 28 19:16:55 128.210.108.195:2141 -> 130.85.1.7:6129 SYN ******S*
```

```
Mar 28 19:16:55 128.210.108.195:2142 -> 130.85.1.8:6129 SYN ******S*
Mar 28 19:16:55 128.210.108.195:2143 -> 130.85.1.9:6129 SYN ******S*
etc ...
```

WHOIS information:

```
OrgName:    Purdue University
OrgID:      PURDUE
Address:    Information Technology
Address:    150 N. University Street
City:       West Lafayette
StateProv:  IN
PostalCode: 47907
Country:    US

NetRange:   128.210.0.0 - 128.210.255.255
CIDR:       128.210.0.0/16
NetName:    PURDUE-CCNET
NetHandle:  NET-128-210-0-0-1
Parent:     NET-128-0-0-0-0
NetType:    Direct Assignment
NameServer: NS.PURDUE.EDU
NameServer: PENDRAGON.CS.PURDUE.EDU
NameServer: HARBOR.ECN.PURDUE.EDU
Comment:
RegDate:
Updated:    2003-01-13

AbuseHandle: PUISP-ARIN
AbuseName:   Purdue University IT Security and Policy
AbusePhone:  +1-765-496-8289
AbuseEmail:  abuse@purdue.edu

TechHandle: SMB17-ARIN
TechName:   Ballew, Scott M.
TechPhone:  +1-765-496-8232
TechEmail:  smb@purdue.edu

OrgAbuseHandle: PUISP-ARIN
OrgAbuseName:   Purdue University IT Security and Policy
OrgAbusePhone:  +1-765-496-8289
OrgAbuseEmail:  abuse@purdue.edu

OrgNOCHandle: PNOC-ARIN
OrgNOCName:   Purdue Network Operations Center
OrgNOCPhone:  +1-765-496-6200
OrgNOCEmail:  noc@purdue.edu

OrgTechHandle: KFR9-ARIN
OrgTechName:   Rice, Kenneth F
OrgTechPhone:  +1-765-496-8320
OrgTechEmail:  rice@purdue.edu
```

6.7.5 202.152.11.196

Number seven on top TCP talkers list, `202.152.11.196` doesn't have a reverse DNS entry. WHOIS reports that it is an APNIC IP, assigned to "PT Lintasarta bagian Sistem Informasi" in Jakarta. It's suspicious behavior was a SYN scan for port 21, FTP servers.

```
Mar 26 04:30:37 202.152.11.196:28629 -> 130.85.1.0:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28630 -> 130.85.1.1:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28683 -> 130.85.1.2:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28632 -> 130.85.1.3:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28633 -> 130.85.1.4:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28634 -> 130.85.1.5:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28635 -> 130.85.1.6:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28636 -> 130.85.1.7:21 SYN ******S*
Mar 26 04:30:37 202.152.11.196:28637 -> 130.85.1.8:21 SYN ******S*
etc ...
```

Interestingly, it included the `.0` IPs, which the other scanners mentioned above did not. This might imply that they had not done any groundwork to determine likely network boundaries.

WHOIS information:

```
inetnum:      202.152.11.192 - 202.152.11.197
netname:      LA-SI
descr:        PT Lintasarta bagian Sistem Informasi
descr:        Jakarta
country:      ID
admin-c:      LA60-AP
tech-c:       LA60-AP
mnt-by:       MAINT-LINTASARTA
changed:      hostmaster@idola.net.id 20030428
status:       ASSIGNED NON-PORTABLE
remarks:      spam and abuse report : abuse@idola.net.id
source:       APNIC

role:         LINTASARTA ADMINISTRATOR
address:      PT Aplikanusa Lintasarta
address:      MH Thamrin Kav 3
address:      Menara Thamrin Bulding 12th Floor
address:      Jakarta 10250
country:      ID
phone:        +62-21-2302345
fax-no:       +62-21-2303883
e-mail:       parman@idola.net.id
trouble:      spam and abuse report : abuse@idola.net.id
trouble:      technical and routing : support@idola.net.id
trouble:      hostmasters : hostmaster@idola.net.id
```

```
admin-c:      YA1-AP
tech-c:       PS174-AP
nic-hdl:      LA60-AP
remarks:      LINTASARTA administrators role object
notify:       parman@idola.net.id
mnt-by:       MAINT-LINTASARTA
changed:      hostmaster@idola.net.id 20030307
source:       APNIC
```

6.8 Correlations from other Practicals

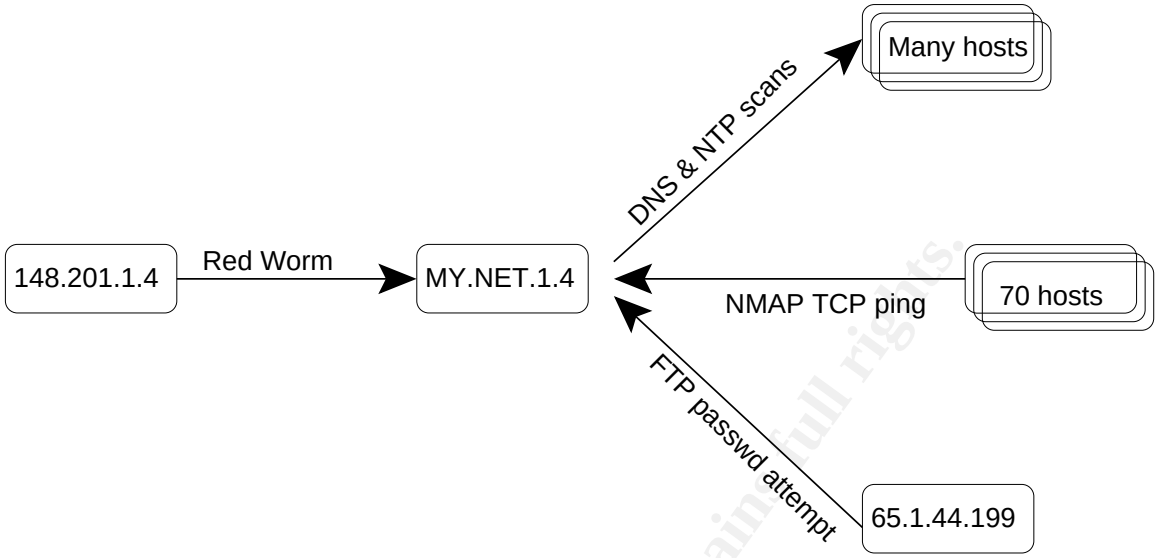
- *Erik Montcalm*, GCIA version 3.3, submitted 2003-11-12. While Erik covered a different date range than I did, his ideas for the MY.NET.30.3 and .4 activity gave me a new track to follow when I was analyzing those alerts. His notes on the custom IRC alerts correlated with the pattern I'd seen.
- *Glenn Larrat*. GCIA version 3.0. Glenn covered Red Worm/Adore and a link to the SANS Adore worm analysis which helped direct my investigation on this signature.
- *Pete Storm*, GCIA version 3.3, submitted Nov 15th 2003. Pete's practical received the Honors designation, and so I used his for ideas on how to structure my practical.

I wasn't able to locate a practical which discussed the external access to LPD issue.

6.9 Link Graph

The link graph presented is for the host `130.85.1.4`. This host was chosen because it was performing widespread sweeps over of the Internet for DNS and NTP. At the same time, it appears to be the victim of a Red Worm/Adore intrusion.

6.9.1 Graph



6.9.2 Description

The Red Worm/Adore traffic:

```
03/28-23:08:42.678343  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535
03/28-23:08:42.678602  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:42.679071  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:42.968418  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:43.051152  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535
03/28-23:08:43.128811  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535
03/28-23:08:43.158807  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:48.810263  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535
03/28-23:08:48.810426  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535
03/28-23:08:48.810466  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:53.817451  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   MY.NET.25.71:65535 -> 148.201.1.4:25
03/28-23:08:53.888099  [**] High port 65535 tcp - possible Red Worm - traffic [**]
   148.201.1.4:25 -> MY.NET.25.71:65535

And a single UDP attempt (likely a generic Red Worm scan):

```

```
03/27-16:46:35.131573  [**] High port 65535 udp - possible Red Worm - traffic [**]
    81.52.250.37:65535 -> MY.NET.1.4:53
```

Hosts that performed an “NMAP TCP ping” on 130.85.1.4 (with count of number of attempts):

```
# grep "NMAP TCP" linkgraph | awk '{print \$7}' | sed 's/:[0-9]*//' | sort | uniq -c | sort -r
  26 64.152.70.68
  16 63.211.17.228
   5 216.5.176.162
   5 205.244.232.133
   2 207.236.181.130
   2 199.34.6.3
   1 4.17.130.252
   1 221.4.176.14
   1 213.11.160.2
   1 212.145.140.210
   1 211.156.183.167
   1 210.77.96.194
   1 208.28.24.252
   1 202.168.194.182
   1 199.34.4.3
   1 194.7.63.155
   1 194.250.176.194
   1 194.244.78.232
   1 194.205.219.4
   1 159.237.4.2
```

Interestingly, none of these IP addresses are targets of the scans for DNS and NTP that 130.85.1.4 was performing.

A single host (65.1.44.199) also performed a “FTP passwd attempt” on 130.85.1.4. This host did not generate any other alerts and did not interact with any other hosts.

6.10 Defensive Recommendations

- This audit, working only from snort logs, could not possibility cover all bases. Perform a full audit of logs, policy and infrastructure design, with “hands-on” capabilities given to the auditor(s).
- *Huge* amounts of data are being produced by your intrusion detection system. Efforts to reduce alerts by eliminating false positives combined with efforts to automate the analysis and trending of the data would simplify the efforts of your intrusion analysts.

- The hosts compromised with Red Worm need to be cleaned up, and standardized patching procedures for Linux hosts implemented. Port 65535 should be blocked.
- Deny outside connections to services that should really be internal only. LDP and TFTP are two prominent examples.
- Running an IRC server (for MY.NET clients only) that can be properly secured and controlled by the University might be one way of mitigating the risk.
- Since a University environment usually won't permit the use of a stringent firewall, and the use of public IP addresses everywhere makes firewalling difficult in any case, extra precautions should be taken:
  - Ensure that an acceptable use policy is both developed and known to all—police it actively to demonstrate that compliance is required
  - Use transparent bridging firewalls for vulnerable services
  - Use host-based IDS products on University servers
  - Consider treating incident response as a higher priority than might normally be the case. Diligent development of rapid response procedures combined with remote switch port control can prevent the spread of new problems by catching it early.
- Because student machines are often not under administrative control of the University, consider tying together your intrusion detection system with your network switches so that certain classes of alerts automatically disconnect a student machine from the network.

6.11 Analysis Process Used

The log files were combined into three large files named `alerts`, `scans` and `oos` to facilitate analyzing the time period as a whole.

The analysis host used is a Celeron 900 running FreeBSD -STABLE and all the standard Unix utilities. Many security-related ports were installed, including `tcpdump` and `snort`.

The logs were run through `snort_sort.pl` (a perl script that sorts a snort alert file by alert type) to help me select what areas to concentrate on. The logs were massaged as needed with `grep`, `awk`, `sed`, `sort` , `wc`, `head` and `uniq`. Examples of the command lines used are shown with the results throughout the practical.

It was quickly discovered that `sort` requires a large amount of temporary file space to work with datasets as large as these 1.4GB logs: the `-F` switch was used to point to a different volume as `/tmp` was not large enough.

This document was prepared with LaTeX2e and `xfig` and coverted to PDF with `ps2pdf`. The ability to incorporate raw log information as an “include” made generating revisions much easier.

References 7

- Bruce Schneier, *Crypto-Gram Newsletter*, April 15 2004
- *IP Provider Metrics BoF, 32nd IETF – Danvers, Mass.*, http://www.psc.edu/ mathis/ippm/meet01.danvers32/minutes.html
- Gary Golomb, *IDS v. IPS Commentary*, June 16 2003, posted to the focus-ids@securityfocus.com and isn@c4i.org mailing lists, archived at http://www.linuxsecurity.com/articles/forums_article-7476.html
- Thomas A. Limoncelli and Christine Hogan, *The Practice of System and Network Administration*, Addison-Wesley, 2002, pg 157–158
- Simson Garfinkel and Gene Spafford, *Practical Unix & Internet Security (2nd edition)*, O'Reilly, 1996, pg 277–287
- Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, *Building Internet Firewalls*, O'Reilly, 2000, pg 734–741
- Tillman Hodgson, *Current State of the Art* deliverable (client deliverable not publicly released), Dec 2001
- W. Richard Stevens, *TCP/IP Illustrated, Volume 1*, pg 171
- W. Richard Stevens, *TCP/IP Illustrated, Volume 1*, pg 235–236
- Laurie Zirkle, *posting to Snort-users mailing list*, archived at http://archives.neohapsis.com/archives/snort/2002-01/0355.html
- Ryan W. Maple, *posting Incidents@securityfocus.com mailing list*, archived at http://cert.uni-stuttgart.de/archive/incidents/2001/02/msg00341.html
- Matt Kettler, *posting to Snort-users mailing list*, archived at http://www.mcabee.org/lists/snort-users/Dec-01/msg00613.html
- *;login: The Magazine of USENIX & SAGE*, Vol 26 Number 7, page 95
- *Planning Concerns, Considerations, and Tips for IDS in Federal IT Systems*, March 30/2001, http://www.sans.org/infosecFAQ/intrusion/fed_IT.htm
- *Erik Montcalm*, GCIA version 3.3, submitted 2003-11-12
- *Pete Storm*, GCIA version 3.3, submitted Nov 15th 2003

Last Updated: October 24th, 2017

Upcoming Training

SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Munich December 2017	Munich, Germany	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
Las Vegas 2018 - SEC503: Intrusion Detection In-Depth	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	vLive
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS London February 2018	London, United Kingdom	Feb 05, 2018 - Feb 10, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS Northern VA Spring - Tysons 2018	McLean, VA	Mar 17, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Baltimore Spring 2018	Baltimore, MD	Apr 21, 2018 - Apr 28, 2018	Live Event
SANS vLive - SEC503: Intrusion Detection In-Depth	SEC503 - 201805,	May 02, 2018 - Jun 07, 2018	vLive
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
Community SANS Columbia SEC503	Columbia, MD	Aug 13, 2018 - Aug 18, 2018	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced