SANS GIAC Level Two:
Intrusion Detection in Depth

GCIA Practical Assignment 3.2

SANS Colorado Springs, CO Conference

June 14, 2002

Jason B. Anderson

<back to TOC>

ASSIGNMENT #1: Describe the State of Intrusion Detection

Reducing False Negatives in Conventional Fail-Open Network Intrusion Detection Systems with Fail-Closed Rule-Set Augmentation

1 Background and Intent of Paper

The advent of network intrusion detection systems (NIDS) have enabled system and network administrators to increase the security and monitoring capability for multiple systems across multiple networks unobtrusively, efficiently, and cost-effectively. The majority of NIDS operate on a 'fail-open' disposition methodology, where traffic is compared against a finite collection of known hostile activity signatures. If hostile traffic does not meet the criteria specified in the known hostile activity signature set, the traffic is presumed to be benign, and the NIDS will 'fail-open.'

The effectiveness of a fail-open NIDS approach continuously degrades with time and must be regularly updated for reasons including:

1. Discovery of new security flaws in existing and future applications/protocols.
2. Discovery of new NIDS evasion and insertion techniques
3. Development of new attack tools for which signature detection rule-sets have yet to be developed based upon the tool's characteristics.
4. Modification of (often trivial) existing attack tools to evade commonly deployed NIDS detection rule-sets.

The purpose of this paper will be to describe and demonstrate the advantages, disadvantages, and implementation considerations of utilizing a fail-closed NIDS design. Specifically, an acceptable-use fail-closed rule-set augmentation method will be discussed for increasing the effectiveness of a conventional fail-open NIDS by augmenting publicly available 'fail-open' attack signature rule-sets with 'fail-closed' rule-set configurations. The goal of such a system is to detect not only well known attacks but to additionally log traffic not explicitly defined as acceptable. The end result of such a system is that traffic should be compared first against a fail-open rule-set, and then secondly verified via an acceptable-use rule-set to validate legitimacy. A simple scenario will be presented using Snort at the end of the paper to illustrate the concepts as they might be employed in a real world scenario.

2 The Goal of Intrusion Detection

A goal of network intrusion detection is to provide a consistent, reliable, and correct disposition of all types of network activity as an acceptable or unacceptable use of network resources; where correctness should be defined in the context of organization-defined security policies. 'Unacceptable use' and 'misuse' will be used synonymously throughout the remainder of this paper.

Even without considering the complications in interpretation/application of organizational security policies in the context of every possible scenario of network activity, the task of delivering consistent, complete, and accurate detection of system misuse becomes increasingly difficult as the sophistication of the adversary's methods for evading common NIDS filter sets continue to improve. Sasha and Beetle summarized the challenges facing security analysts with the following statement:

"The goal of IDS technology - to detect misuse, must be considered a genuinely 'hard problem', and indeed there exists several areas of difficulty associated with implementing an NIDS (network-based IDS) such that the results it generates are genuinely useful, and can also be trusted. "(Sasha, Beetle)

2.1 The Role of Network Intrusion Detection in a Site Security Plan and Approaches to Misuse Response

Network based intrusion detection can be regarded as an approach to ensuring the three basic tenets of information security, namely: data availability, data confidentiality, and data integrity. Intrusion detection systems support this goal in an approach that can generally be classified as either active or passive; NIDS systems can be configured to either 1) actively respond to data perceived as anomalous through transmission of traffic intended to terminate a questionable transaction between the attacker and victim, or 2) by passively logging traffic for later investigation and response from an analyst. *In both cases, the ability of the network intrusion detection system to identify 'misuse' is the first step in either actively or passively responding to attacks.* This paper will attempt to propose a method to increase the resolution of network misuse, by reducing false negative NIDS traffic disposition (missing legitimate attacks).

2.2 Methods by which Intrusion Detection Analysts Can Define Misuse

In practical application, many real world NIDS are configured to monitor only for well known attacks by implementation of a 'generic' rule-set. Such rule sets are generally either publicly available, or provided by NIDS vendors. In either case, studious adversaries can arm themselves with valuable insight into a majority of NIDS configurations by merely acquiring and studying these easily accessible rule sets.

By solely depending on an easily accessible set of fail-open well-known attack signature rules, the analyst contributes to allowing the 'arms-race' between the adversary and analyst to perpetuate indefinitely.

A method of supplementing this strictly fail-open methodology can be identified by quantifying the intent of an organizational security policy so that the terms of this policy are effective and enforceable. Security policies generally boil down to: "Use of certain systems in an unauthorized manor, or by unauthorized personnel is strictly prohibited." It is by these two general classifications of system misuse that the intrusion detection analyst can disposition all layers of network traffic on her/his monitored networks. The unfortunate reality of such an assessment is that its vagueness allows for a wide spectrum of interpretation. In terms of defining a NIDS rule set, an analyst could interpret misuse as all usages which match an explicitly defined set of well-known misuse signatures. Another less commonly implemented interpretation of misuse is all traffic falling outside the explicitly defined set of acceptable-use signatures. The former's primary benefit is easy enforcement, the former, while the latter's benefit is comprehensive coverage.

2.3 An Alternate Approach to Network Traffic Classification: Fail Closed vs. Fail Open NIDS Rule-Sets.

The conventional approach to network activity disposition has been to consolidate a list of well-known misuse signatures and compare each packet on monitored networks against this list. This is a 'fail open' system, where all traffic is dispositioned as acceptable, unless specifically determined to be anomalous based upon the fail-open misuse rule-set. This approach has the advantage of covering a large majority of common attacks, but leaves itself open to false negatives for trivial manipulations of standard attacks, zero-day attacks (attacks not yet known to the security community), and attempts at network mapping and reconnaissance activity.

An alternative approach to traffic disposition, described by (Sasha, Beetle), is to identify a set of 'acceptable use' signatures, whereas traffic that fails to match upon a list of known 'acceptable-use' patterns is dispositioned as anomalous, this approach can be summarized as a 'fail-closed' strategy.

2.4 Advantages and Disadvantages of the 'Fail-Closed' and 'Fail-Open' Traffic Disposition Strategies

Inherent in the nature of a fail-closed strategy is an entirely different set of advantages and disadvantages as contrasted to those associated with the fail-open strategy. The primary advantage of a fail-closed strategy is the assurance that potentially hostile traffic not explicitly identified as anomalous according to a finite set of well-known attack signatures should still be detected by the NIDS system, i.e., more true-positives should be detected by a well tune fail-closed methodology than by a well tuned fail-open methodology, at least in cases where the acceptable-use space is well defined. This reduces the NIDS' propensity to produce false-negative dispositions; as false negatives are inversely correlated with true positives. A disadvantage of the fail-closed system is that a insufficiently defined fail-closed rule-set produces a larger number of false-positives for the analyst to review than a fail-open rule-set, thereby potentially reducing the effectiveness of the analyst.

Inconclusive fail-open rule-sets will tend to produce too many false negatives; while inconclusive fail-closed rule-sets will tend to produce too many false-positives. A 'conclusive' fail-closed rule-set would, in theory, account for *all and only* the legitimate scenarios associated with a given type of traffic. In practice, it is not practical to produce such an exhaustive set. To do so would require intimate familiarity with protocol definitions, as defined in protocol RFC's, and with site network and system topology, among other things.

Both false positives and false negatives reduce the effectiveness of the NIDS and its analyst. Optimum NIDS effectiveness exists in the state where both false negatives and false positives are minimized. A summary of this discussion is presented in **Table 1**.

	Disadvantages	Advantages
Fail Open Disposition Strategy	1. A difficult task to comprehensively account for all variations of inherently bad traffic to minimize the false-negatives at the rate in which new attack methodologies are developed. 2. Rules require constant update to keep up with a changing global security environment.	1. Maintaining a 'pretty good' list of well-known attack signatures is fairly easy 2. downloading signature sets from centralized locations can sometimes be at least partially automated
Fail Closed Disposition Strategy	1. An overwhelmingly difficult task to exhaustively account for all variations of good traffic, while including no others, to minimize the false-positives	1. A comprehensive filter set would detect a larger set of true-positives 2. Rules stay up to date as long as protocol definitions and site policy remains unchanged.

Table 1: Advantages/Disadvantages of the Fail-Open and Fail-Closed Disposition Strategies

2.5 A Comparison of Simple Fail-Open vs. Fail-Closed Disposition Strategies

The following explanation assumes that all traffic transmitted across a network can be classified as either good or bad in accordance with a site security policy.

In **Figure1**, we see a logical diagram of a fail-closed and fail-open strategy for disposition of 'bad traffic,' where the definition of a 'bad traffic' is defined within the context of a site security policy.

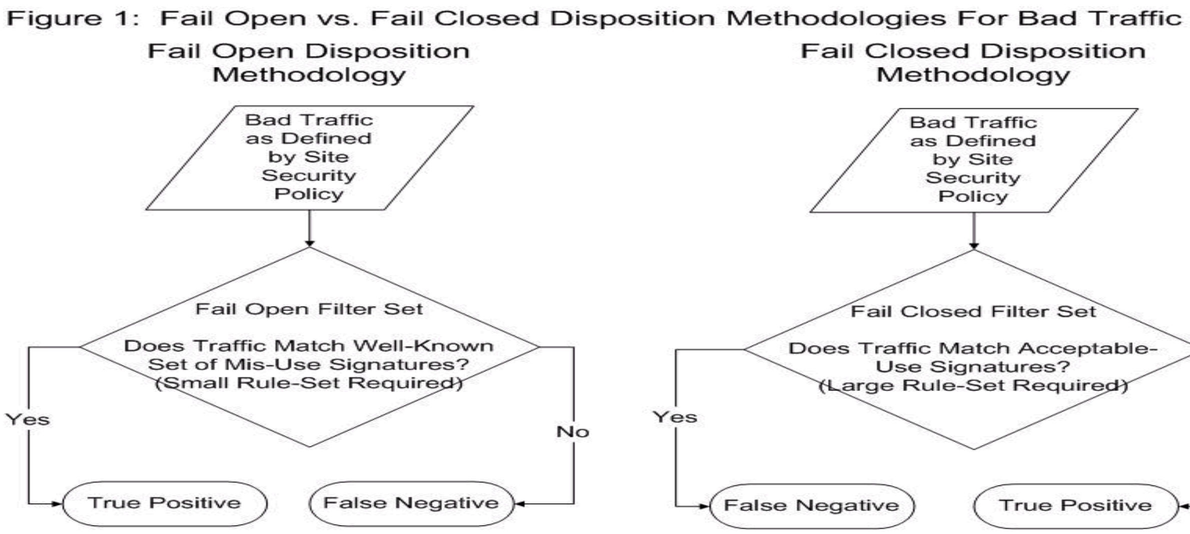


Figure 1: Fail Open vs. Fail Closed Disposition Methodologies For Bad Traffic
Fail Open Disposition Methodology
Bad Traffic as Defined by Site Security Policy
Fail Open Filter Set
Does Traffic Match Well-Known Set of Mis-Use Signatures? (Small Rule-Set Required)
Yes
No
True Positive
False Negative
Fail Closed Disposition Methodology
Bad Traffic as Defined by Site Security Policy
Fail Closed Filter Set
Does Traffic Match Acceptable-Use Signatures? (Large Rule-Set Required)
Yes
No
False Negative
True Positive

Figure 1: Fail open vs Fail Closed Disposition Methodologies for Bad Packets

In Figure2, we see a logical diagram of a fail-closed and fail-open strategy for disposition of ‘good traffic,’ where the definition of a ‘good traffic’ is also defined within the context of a site security policy.

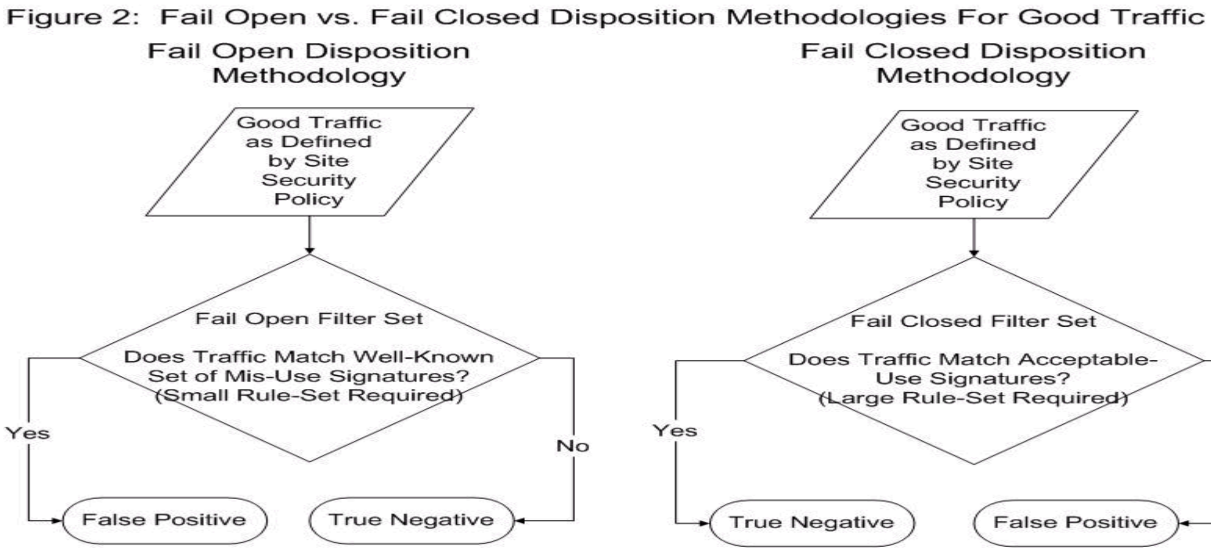


Figure 2: Fail Open vs. Fail Closed Disposition Methodologies For Good Traffic
Fail Open Disposition Methodology
Good Traffic as Defined by Site Security Policy
Fail Open Filter Set
Does Traffic Match Well-Known Set of Mis-Use Signatures? (Small Rule-Set Required)
Yes
No
False Positive
True Negative
Fail Closed Disposition Methodology
Good Traffic as Defined by Site Security Policy
Fail Closed Filter Set
Does Traffic Match Acceptable-Use Signatures? (Large Rule-Set Required)
Yes
No
True Negative
False Positive

Figure 2: Fail open vs. Fail Closed Disposition Methodologies for Good Packets

Inherent to a stagnant fail-open policy is that the ratio of true positives to false negatives will degrade over time, as new attack methodologies are developed by the Black-Hat community (causing false negatives to increase). The fail-closed strategy demands that a relatively large number of rules are developed upfront to accurately define the acceptable use space to ensure that the ratio of false positives to true negatives is satisfactory. Once this acceptable-use rule set is identified, it need only be updated when a new variation of traffic is deemed as acceptable-use according to site policy. It degrades over time only if modifications arise within the realm of acceptable-use network activity, since such an acceptable use policy should exclude the entire set of unacceptable uses. Because the assimilation of an exhaustive acceptable-use rule-set would have to include not only all possible acceptable variations in each of the TCP/IP protocol fields, it would also have to account for all possible acceptable variations in the data payload in all of the acceptable TCP/IP traffic, yet account for no more than what is strictly acceptable. In practice, a perfect fail-closed rule-set would be nearly impossible to construct.

3. Simultaneously minimizing both false positives and false negatives in a NIDS Environment.

While it is very difficult to construct either an exhaustive well defined acceptable-use rule-set for a fail-closed disposition methodology, or maintaining a comprehensive fail-open rule-set incorporating all known instances and variations of known misuse signatures at the rate in which they are developed, it is relatively easy to build a *pretty good* rule-set for either the fail-open or fail-closed methodologies. A majority of the NIDS in use today likely fall into the category of *pretty good* fail-open disposition based, presuming they are well tuned (such that false positives are minimized) and regularly updated. Such a rule-set is commonly available either as an open-source collection of rules, such as the regularly updated Snort rule-set, or from an NIDS vendor. Setting up a *pretty good* fail open NIDS is as simple as downloading the commonly available fail-open rule-set and performing tweaks to integrate it for one's monitored environment to minimize false positives on a regular basis.

3.1 Constructing a *pretty good* fail-closed rule-set for a simple network

Despite the fact that it is difficult to construct a well defined fail-closed rule-set capable of accounting for all variations in both TCP/IP protocol data and application cargo data, it should be relatively easy to narrow the types of TCP/IP protocol data that should be expected on any given network. As an example, a network populated by only UNIX systems should only see UNIX related traffic, possibly as little as just NIS/NIS+, SSH, RSH, HTTP, RPC, and DNS traffic. Further, traffic direction will depend on network and system topologies specific to site implementations, and should thus be limited between certain hosts. As a theoretical example, consider a network populated by only Linux systems who, according to site security policy, are expected to only communicate via the SSH protocol. The following acceptable use table (**Table 2**)could be used to specify a *pretty good* rule-set of acceptable SSH traffic based upon the IP, and TCP protocols, as seen in Table 1. Here we strive to define *pretty good* as a maximized subset of the acceptable-use space, minimizing the inclusion of unacceptable-use space. **Diagram 1** illustrates this point.

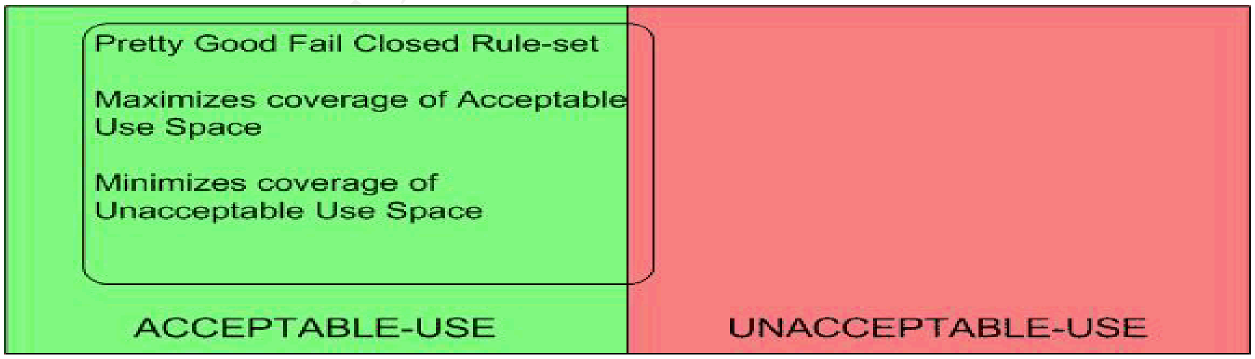

Pretty Good Fail Closed Rule-set
Maximizes coverage of Acceptable Use Space
Minimizes coverage of Unacceptable Use Space
ACCEPTABLE-USE
UNACCEPTABLE-USE

Diagram 1

SSH (TCP Port 22 Acceptable Use Table) (All SSH traffic falling outside these criteria should be dispositioned as anomalous)

Protocol Type	Protocol Field	Snort 1.8 features Available to explicitly quantify acceptable use?	Acceptable-use criteria
IP	Version	No	ipv4 is only acceptable IP version
IP	Header length	No	20 Bytes, assuming no options will be acceptable
IP	Type of Service	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.4	Should be expected to be 0, unless site networks employ TOS network optimizations, otherwise, see (Stevens, [1]) for details.
IP	Total Packet Length	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.8	>40 Bytes (20 Bytes for IP header, 20 for TCP Header) also, according to [14] the SSH_CMSG_ MAX_PACKET_SIZE variable can be modified in the source code of openSSH such that packets larger than this value can be identified
IP	IP ID Number	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.5	Any IPID should be acceptable
IP	Flags	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.7	Any Flag combination should be acceptable
IP	Fragment Offset	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.37	Any offset should be acceptable
IP	Time to Live	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.3	If only Linux SSH clients TTL should be less than 64 (See [13]) and greater than (64 – acceptable hop count)
IP	Protocol Type	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.32	TCP (by definition)
IP	Checksum	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.1.3	Any valid checksum should be acceptable
IP	Source Address	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.2.3	Only from acceptable SSH clients
IP	Destination Address	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.2.3	Only to acceptable SSH servers
IP	Options	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.6	Assume no options will exist, otherwise, see (Stevens, [1]) for details.
TCP	Source Port	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.2.4	1024:65535 (presuming .rhost based authentication is considered invalid) (Ylonen, [6])
TCP	Destination Port	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.2.4	22 (By definition)
TCP	Sequence Number	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.14	Any Sequence number should be acceptable

TCP	**Acknowledgement Number**	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.15	Any Acknowledgement number should be acceptable
TCP	**Header Length**	No	20 Bytes, assuming no options, otherwise, see (Stevens, [1]) for details.
TCP	**Reserved Bits**	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.13	No reserved bits is acceptable, unless ECN (explicit congestion notification) is in use.
TCP	**URG, ACK, PSH RST, SYN, FIN Bits**	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.3.13	Any valid combination is see (Stevens, [1]) for details on valid combinations
TCP	**Window Size**	No	Any Window Size is acceptable
TCP	**TCP checksum**	http://www.snort.org/docs/writing_rules-1.9.0/chap2.html#tth_sEc2.1.3	Any TCP checksum is acceptable
TCP	**Urgent Pointer**	No	No urgent pointers are acceptable
TCP	**Options**	No	No options are acceptable

Table 2: An assessment of acceptable-use criteria for TCP/22 (SSH) traffic, and the Snort NIDS capability of monitoring for it.

While most networks typically have legitimate usage of more than one type of traffic, the example acceptable-use set in Table 1 can be replicated for other acceptable protocol traffic in preparation of an acceptable-use rule-set. By using a process such as the one exemplified in acceptable-use template of Table 1, an analyst can assimilate a *pretty good* fail-closed rule-set for any network transmitting a reasonable amount of different traffic types.

A note of caution, while it is important to quantify a sufficiently wide range of acceptable traffic to account for all of the acceptable traffic scenarios, as to minimize the false positives illustrated in **Figure 2**, it is *critical* that great care is taken to ensure acceptable-use rule-sets provide very little room for scenarios falling outside of acceptable use, else, the false negatives, as seen in **Figure 1.,** will become unacceptably high. **Diagram 1** captures the spirit of how a good fail-closed rule-set should span acceptable and unacceptable-use space. The best way to quantify a traffic type's exact acceptable use space would be to carefully study the associated RFC, as done with the above attempt at quantification of the SSH acceptable-use table (see [6] Ylonen). Additionally, a strict review of site implementation and application of the given protocol should be valuable in minimizing the acceptable-use rule-set.

3.2 Integrating a *pretty good* fail-open rule-set with a *pretty good* fail-closed rule-set to produce a *better* comprehensive rule-set

While it is possible to deploy a NIDS with either a fail-open or fail-close based disposition rule-set, there is nothing stopping a sufficiently paranoid analyst from integrating *both* techniques into a common rule-set. **Figure 3** and **Figure 4** outline a logical disposition flow for such a fail-open, followed by a fail-closed rule-set.

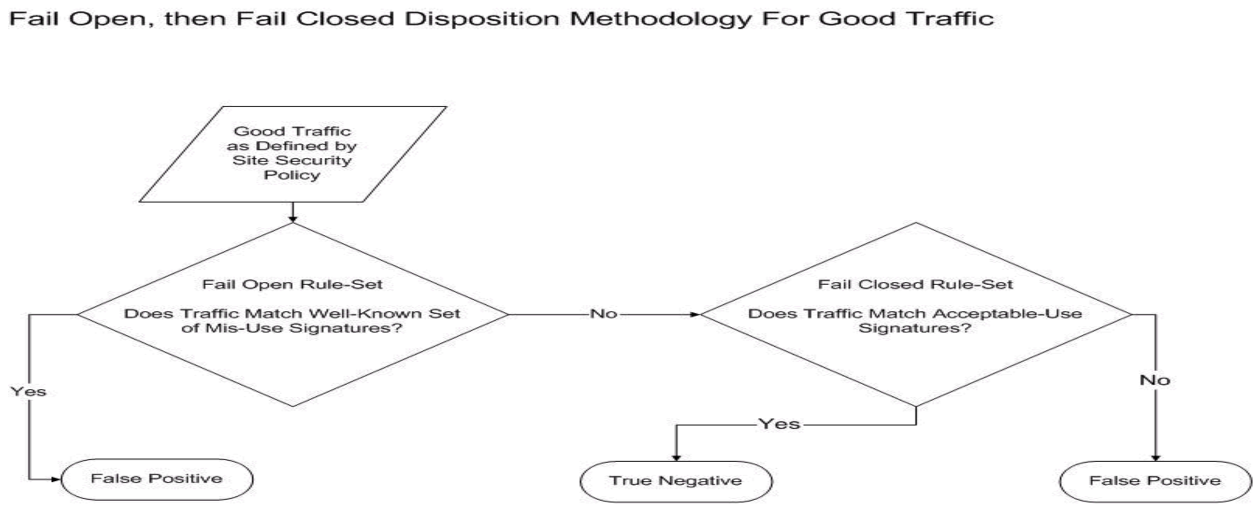

Fail Open, then Fail Closed Disposition Methodology For Good Traffic
Good Traffic as Defined by Site Security Policy
Fail Open Rule-Set
Does Traffic Match Well-Known Set of Mis-Use Signatures?
No
Fail Closed Rule-Set
Does Traffic Match Acceptable-Use Signatures?
Yes
No
Yes
False Positive
True Negative
False Positive

Figure 3: Fail Open, then Fail Closed Disposition for Good Traffic

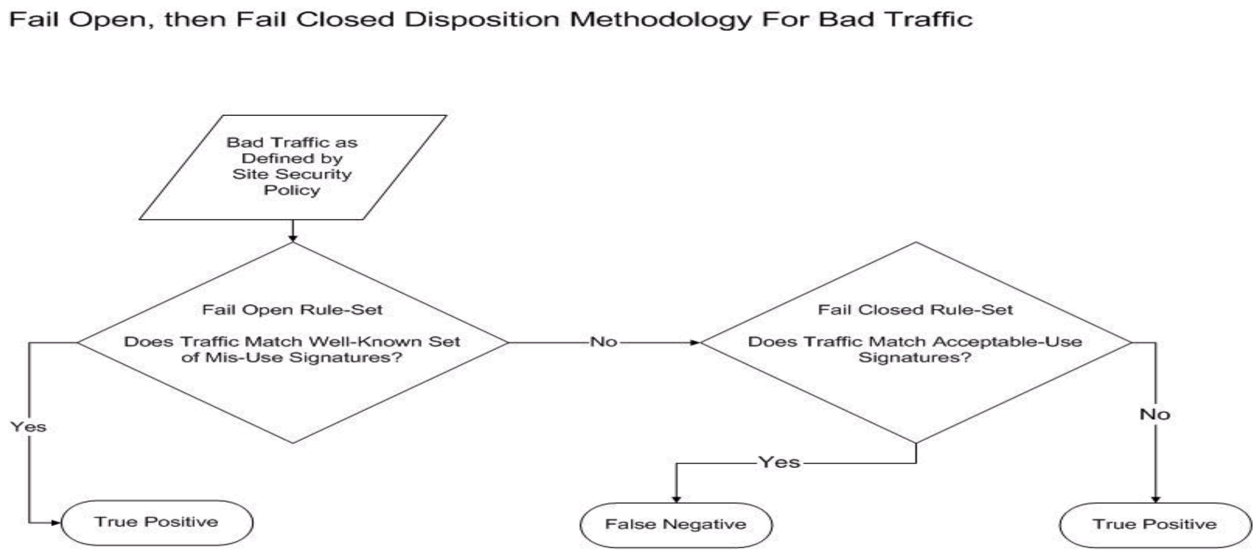

Fail Open, then Fail Closed Disposition Methodology For Bad Traffic
Bad Traffic as Defined by Site Security Policy
Fail Open Rule-Set
Does Traffic Match Well-Known Set of Mis-Use Signatures?
No
Fail Closed Rule-Set
Does Traffic Match Acceptable-Use Signatures?
Yes
No
Yes
True Positive
False Negative
True Positive

Figure 4: Fail Open, then Fail Closed Disposition for Bad Traffic

The benefit of augmenting the conventional fail-open disposition rule-set with a fail-closed rule-set is that bad traffic managing to pass through the fail-open filters will still be compared to the acceptable-use filter-set of the fail-closed disposition rule-set. If the fail-open validated traffic matches the acceptable use criteria, it is considered legitimate, otherwise, it is considered potentially illegitimate and is logged appropriately.
The additional benefit of passing bad traffic through a fail-closed acceptable-use rule-set to detect a greater number of true positives is only effective if the acceptable-use filter-set is sufficiently specific and well-defined.

4. An example of a fail-closed rule-set augmented fail-open NIDS dispositioning methodology

To summarize the paper, an attempt will be made to apply the fail-closed augmented fail-open disposition approach by employing the Snort lightweight, open source NIDS to define a rule-set that should be capable of dispositioning all TCP/Port 22 traffic as it should exist on the theoretical network topology mentioned above, populated only by Linux systems communicating only via the SSH protocol.

4.1 Assumptions for fail-closed augmented fail-open disposition methodology

Assumptions for the following, *pretty good* SSH fail-closed acceptable-use rule-set, as quantified in **Table 4**, include:

- The Snort -1.8,6 NIDS or newer is used to parse and employ the rules-set (although other NIDS could use a similar strategy, if rule-sets are reconfigured to match the specific NIDS syntax)
- The variable VALID_SSH_CLIENTS contains the list of only valid SSH clients
- The variable VALID_SSH_SERVERS contains the list of only valid SSH servers
- The rule order of Snort has must be set as default, i.e., the following line has been included in the snort configuration file:

  “config order: alert pass log”

Additionally, **Table 4**, represents an example subset of the acceptable use criteria as defined in **Table 1,** and conceptually illustrated in **Diagram 1** above..

Fail-open rule-set from http://www.snort.org/dl/signatures/snortrules-stable.tar.gz Relevant to SSH known exploits (Color Coded in accordance with Figure 5 below)
alert TCP $EXTERNAL_NET any -> $HOME_NET 22 (msg:"**EXPLOIT SSH CRC32 overflow /bin/sh**"; flags:A+; content:"/bin/sh"; reference:bugtraq,2347; reference:cve,CVE-2001-0144; classtype:shellcode-detect; sid:1324; rev:3;)
alert TCP $EXTERNAL_NET any -> $HOME_NET 22 (msg:"**EXPLOIT SSH CRC32 overflow filler**"; flags:A+; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"; reference:bugtraq,2347; reference:cve,CVE-2001-0144; classtype:shellcode-detect; sid:1325; rev:3;)
alert TCP $EXTERNAL_NET any -> $HOME_NET 22 (msg:"**EXPLOIT SSH CRC32 overflow NOOP**"; flags:A+; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; reference:bugtraq,2347; reference:cve,CVE-2001-0144; classtype:shellcode-detect; sid:1326; rev:3;)
alert TCP $EXTERNAL_NET any -> $HOME_NET 22 (msg:"**EXPLOIT SSH CRC32 overflow**"; flags:A+; content:"|00 01 57 00 00 00 18|"; offset:0; depth:7; content:"|FF FF FF FF 00 00|"; offset:8; depth:14; reference:bugtraq,2347; reference:cve,CVE-2001-0144; classtype:shellcode-detect; sid:1327; rev:3;)
alert TCP $EXTERNAL_NET any -> $HOME_NET 22 (msg:"**SCAN SSH-research-scanner**"; flags:A+; content:"|00 00 00 60 00 00 00 00 00 00 00 00 01 00 00 00|"; classtype:attempted-recon; sid:617; rev:2;)

Table 3. SSH known misuse signatures for fail-open ruleset.

Fail-closed Acceptable-Use Rule-Set derived from Table 1. (Color Coded in accordance with Figure 5 below)	
Snort Rule-set	**Purpose**
pass TCP $VALID_SSH_CLIENTS 1024:65535 -> $VALID_SSH_SERVERS 22 (tos:0; flags: A+; dsize:>40;)	This rule represents the acceptable use critieria for port 22 destined TCP traffic

pass TCP $VALID_SSH_SERVERS 22 -> $VALID_SSH_CLIENTS 1024:65535 (tos:0:; flags:A+; dsize:>40;)	This rule represents the acceptable use criteria for port 22 sourced traffic

Table 4. Fail closed ruleset

Fail-closed Catch-All rule-set derived from Table 1. (Color Coded in accordance with Figure 5 below)	
Snort Rule-set	**Purpose**
log TCP any any -> any 22 (msg:" **TCP port 22 destination traffic outside acceptable-use criteria**";)	This rule catches and logs all incoming TCP port 22 traffic caught neither be fail-open or acceptable use criteria
log TCP any 22 -> any any (msg": **TCP port 22 source traffic outside acceptable-use criteria**";)	This rule catches and logs all outgoing TCP port 22 traffic caught neither be fail-open or acceptable use criteria

Table 5.

4.2 Fail-Open, then Fail-Closed Disposition methodology in terms of Snort RuleTreeNodes and OptTreeNodes Rule-Set Implementation

According to (Ruiu, [7] section 3.13), Snort rule-sets are parsed into data structures based upon:

1) the rule order specification (default is alert->pass->log),
2) the order upon which the rules are parsed upon snort process start up.

While no automated tool is known by the author to parse snort rules into visual diagrams equivalent to the data structures as produced in the Snort engine, a representation can be made of the SSH protocol as seen in **Figure 5**.

Figure 5 shows that all TCP/Port 22 (SSH) traffic is compared first against the conventional fail-open rule-sets pertaining to the SSH protocol as found on October 4th, 2002 at www.snort.org. After being compared to the known misuse signatures quantified in **Table 3**, the TCP/Port 22 traffic that does not match one of the well known misuse signatures is passed to **Table 4**, while **Table 4** is not exhaustively comprehensive, nor perfectly specific, it *narrows the range of TCP/22 traffic to a smaller range* than covered by the fail-open rule-set specified in **Table 3**. If the TCP/Port 22 traffic does not match an acceptable-use snort 'pass' rule, it is logged in the fail-closed 'catch-all' rule listed in **Table 5**. The end result is that *all* traffic related to TCP/Port 22 is *dispositioned according to one of the three disposition scenarios* illustrated in **Figure 5.**

5. Summary

While the fail-closed augmented fail-open implementation illustrated above does not provide a comprehensive real-world example of all of the necessary rules that would be required to deploy a solution to significantly decrease the number of false-negatives for all protocols typically seen on a typical network, it provides a framework for building acceptable-use quantified fail-closed rule-sets that can be used to identify a greater number of potentially hostile, possibly true positive signatures. Such a configuration could be considered in environments were maximum monitor capability is required from a NIDS.

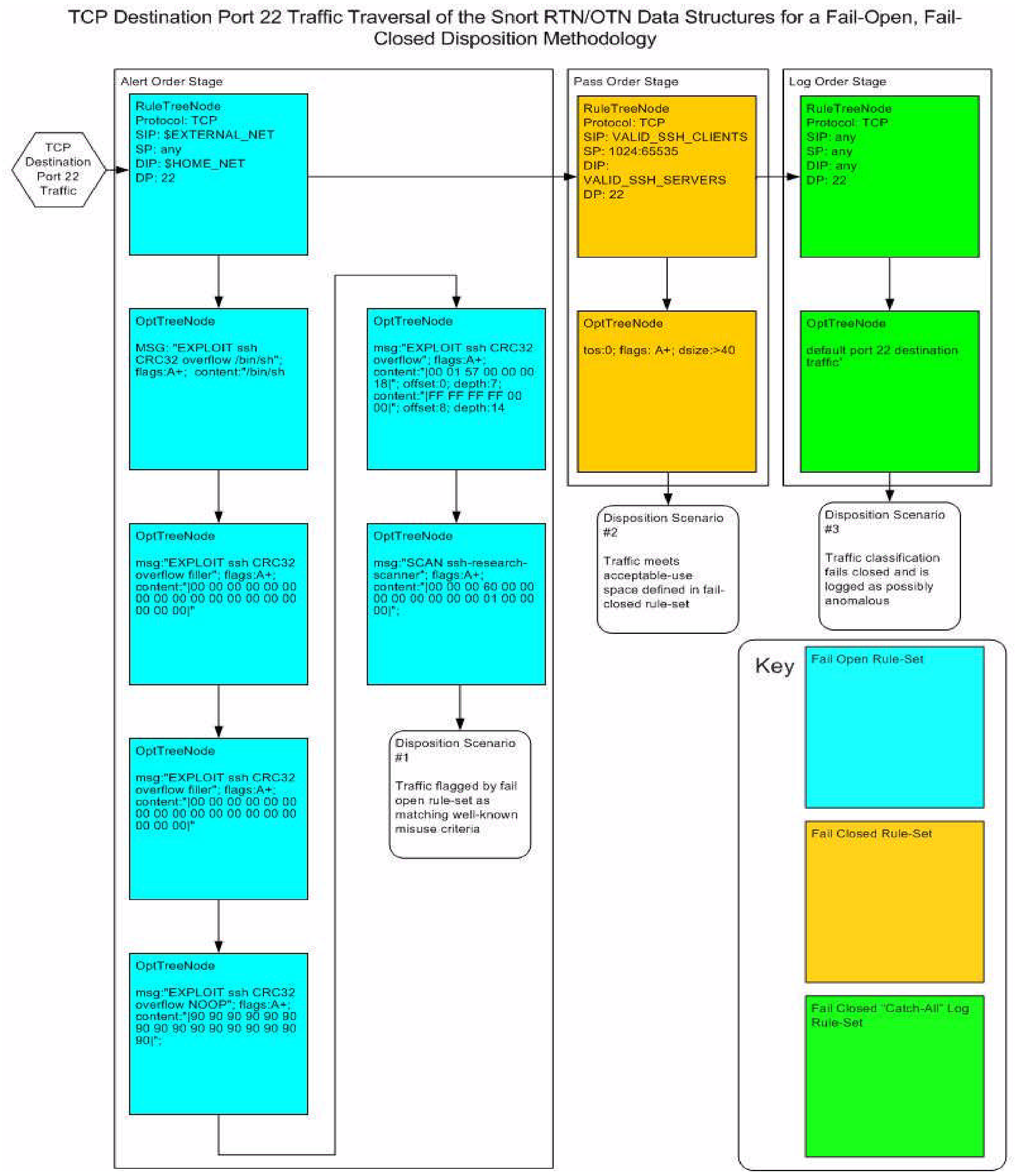

TCP Destination Port 22 Traffic Traversal of the Snort RTN/OTN Data Structures for a Fail-Open, Fail-Closed Disposition Methodology
TCP Destination Port 22 Traffic
Alert Order Stage
RuleTreeNode
Protocol: TCP
SIP: $EXTERNAL_NET
SP: any
DIP: $HOME_NET
DP: 22
OptTreeNode
MSG: "EXPLOIT ssh CRC32 overflow /bin/sh"; flags:A+; content:"/bin/sh
OptTreeNode
msg:"EXPLOIT ssh CRC32 overflow filler"; flags:A+; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"
OptTreeNode
msg:"EXPLOIT ssh CRC32 overflow filler"; flags:A+; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"
OptTreeNode
msg:"EXPLOIT ssh CRC32 overflow NOOP"; flags:A+; content:"|90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90|";
OptTreeNode
msg:"EXPLOIT ssh CRC32 overflow"; flags:A+; content:"|00 01 57 00 00 00 18|"; offset:0; depth:7; content:"|FF FF FF FF 00 00|"; offset:8; depth:14
OptTreeNode
msg:"SCAN ssh-research-scanner"; flags:A+; content:"|00 00 00 60 00 00 00 00 00 00 00 00 01 00 00 00|";
Disposition Scenario #1
Traffic flagged by fail open rule-set as matching well-known misuse criteria
Pass Order Stage
RuleTreeNode
Protocol: TCP
SIP: VALID_SSH_CLIENTS
SP: 1024:65535
DIP: VALID_SSH_SERVERS
DP: 22
OptTreeNode
tos:0; flags: A+; dsize:>40
Disposition Scenario #2
Traffic meets acceptable-use space defined in fail-closed rule-set
Log Order Stage
RuleTreeNode
Protocol: TCP
SIP: any
SP: any
DIP: any
DP: 22
OptTreeNode
default port 22 destination traffic"
Disposition Scenario #3
Traffic classification fails closed and is logged as possibly anomalous
Key
Fail Open Rule-Set
Fail Closed Rule-Set
Fail Closed "Catch-All" Log Rule-Set

Figure 5.

6.0 References

[1] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994
[2] Northcutt, Stephen; Novak, Judy. Network Intrusion Detection: An Analyst's Handbook, Second Edition. Indianapolis: New Riders Publishing, Inc, 2000

[3] Beetle; Sasha. "A STRICT ANOMOLY DETECTION MODEL FOR IDS." Phrack 56. 1 May 2000. URL: http://www.phrack.com/show.php?p=56&a=11 (September 21, 2002)
[4] Roesch, Martin. Chris Green "Snort Users Manual Release: 1.9.x: How to Write Snort Rules and keep your sanity." URL: http://www.snort.org/docs/writing_rules/chap2.html#tth_chAp2 (September 21, 2002)
[5] Unknown, "Default TTL values in TCP/IP." URL:http://www.map.ethz.ch/ftp-probleme.htm , SWITCH: Swiss Academic & Research Network (October 04, 2002)
[6] Tatu Ylonen, "The SSH (Secure Shell) Remote Login Protocol." 15 November, 1995. URL: http://www.free.lp.se/fish/rfc.txt (October 05, 2002)
[7] Roesch, Martin, Dragos Ruiu. "Snort FAQ." (5 March, 2002), URL: http://www.snort.org/docs/faq.html, (October 05, 2002)

<Back to TOC>

Assignment 2: Analysis of Anomalous Network Detects

In the following assignment, the 3 traces represent 2 investigations of logs from the www.incidents.org log repository, and one from a network I have access to. The first represents what could be evidence of a sophisticated idle scanning technique, the second what appears to be a successful DNS zone transfer attempt, and the third represents a detect which yielded a relatively small amount of correlations from the NIDS community, and seems to be a rare catch which will hopefully be helpful to future analysts dealing with similar traffic.

Network Detect 1: TCP Destination Port 0 Scan

1. **Source of Trace:**
   The following trace was obtained from the www.incidents.org/logs/Raw/ repository. The specific file including the trace can be found at www.incidents.org/logs/Raw/2002.6.11
2. **Detect was generated by:**
   The detect was generated by the Snort Intrusion Detection System Version 1.8.6 (Build 105). The rule-set used was a standard snortrules-stable.tar.gz set downloaded from http://www.snort.org/dl/signatures/ on 6/10/02 19:03 PM. All rule-sets are being utilized. The specific rule responsible for detecting the anomalous network behavior was extracted from the bad-traffic.rules file included with the standard signature set, and is listed as follows:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp port 0 traffic"; sid:524; classtype:misc-activity; rev:3;)
```

Figure 2-1 Snort Rule

It should be noted that the rule used to capture the packet for the original log may have been a custom rule.

The $HOME_NET variable in the above rule has been specified to represent the the x.y.0.0/16 class B subnet. $HOME_NET represents all traffic local to the monitored systems on the x.y.0.0/16 network.

The standard Snort 1.8.6 (Build 105) TCP full format as produced by the '-X' argument is described below in **Figure 2-2**. (The number of bits per field is in parentheses)

<The message from the rule that matched the packet>									
MM/DD-HH:MM:SS.xxxxxx	Source MAC address	->	Dest MAC addres	Ether Type	Ether Length				
Source IP: TCP Source Port	->	Dest IP: TCP Dest Port	IP Proto	IP Type of Service	IP TOS	IP Frag ID	IP Header Length	IP datagram Length	Fragment Flags
IP Flags UAPRSF	Sequence Number	Acknowledge number	TCP Window Size	TCP header length					
TCP Options(n) => Opt1 Opt2 Opt3 … Opt(n)									
0x0000	Dest 48 bit Ethernet Address(48)	Source 48 bit Ethernet Address(48)	Ether Type (16_	IP Ver IpLen TOS (16_					ASCII rep. of 32 Byte Stanza
0x0010	IP DgmLen (16)	IP ID (16)	IP Frag Offset/ID (16)	IP TTL (8)	IP prot (8)	IP Header chksum (16)	IP Source Addr (32)	IP Dest Addr (1-16 of 32)	ASCII rep. of 32 Byte Stanza
0x0020	IP Dest addr (17-32 of 32)	TCP Src port(16)	TCP Dest Port(16)	Sequence Number (32)	Acknowledgement Number(32)	tcp Hdr en(4) Res Bits(6) TCP Flags(6)			ASCII rep. of 32 Byte Stanza
0x0030	Window Size (16)	TCP Chksum (16)	Urgent Pointer (16)	Options (1-80 of 96)					ASCII rep. of 32 Byte Stanza
0x0040	Options (81-96 of 96)								ASCII rep. of 32 Byte Stanza

Figure 2-2 Snort TCP Packet Key

The purple cells represent the Ethernet protocol data, the green cells represent the IP protocol data, and the yellow cells represent the TCP protocol data.

3. Probability the source address was spoofed:

The following data will support the conclusion that the data was probably not spoofed, to support this analysis, data from both sides will be presented and weighed against each other.

3.1 Data supporting probability that source address was spoofed:

- **Invalid TCP destination port**

  It is safe to assume that no TCP connection was made. Because port 0 is not a valid TCP destination port, no host should allow for a TCP connection to this port. Had a TCP connection have been made, it would be reasonable to assume that spoofing of the source IP address was not done; conversely, since it is reasonably assumed that no TCP connection was made, spoofing remains a possibility.

- **Frequency of Stimuli**

  As can be seen in **Figure 2-4**, the detect has been broken into 3 sets of 4 packets. The logic supporting this breakdown can be seen by observing the delay between SYN packets with like source port numbers. The delay follows the conventional SYN packet decaying retransmit frequency of 3, then 6, then 12 seconds. However, most TCP stacks do not 'give up' after just 4 attempts. It is typical to also see another packet after 24 seconds. This missing last attempt suggests that the attackers system's TCP stack was possibly not involved in producing this TCP attempt.

- **IP Fragment Number is 0**

  As can be seen in **Figure 2-5**, the IP ID number is 0, which is not a valid number for IP Fragment ID's, which are legally limited to between 1 and 65535. This data suggests that the attacker isn't using a conventional IP stack to generate the IP packet.

  It follows that if the attacker has resources allowing him to modify the IP fragment ID within the IP protocol header, that she also has the ability of easily manipulating the Source IP address.

  If we assume that the above conclusion regarding the packet being crafted outside of the constraints of an unmodified TCP stack, then it follows that efforts may have been made to craft protocol headers in the packet. Because source IP address is one such packet protocol parameter, we can assume that the attacker is in possession of utilities at least capable of spoofing.

3.2 Data supporting probability that source address was not spoofed:

- **IP Protocol Type: TCP**

  Unlike UDP, TCP is a connection oriented protocol, delivering the following services to applications: virtual circuits, application i/o management, network i/o management, flow control and reliability.[Hall, pg 270-271] These services requires correspondence between the source and destination endpoints, specifically, flow control and reliability employ the tracking of sequence and acknowledgement number logging to account for packets on both sides of the full duplex connection. Sequence number prediction is a requirement of a successful TCP spoof, and can be fairly difficult to attain on modern operating systems with non-trivial sequence number incrimination algorithms. Because of this difficulty, established TCP connections are very difficult to spoof.
- **Apparently legitimate incrementing TCP Source Ports**

  If we assume that the packet was crafted by a tool, then it follows that the tool used to craft the series of packets would not remember what source port it had used in prior instantiations. However, the source ports seem to be randomly incrementing as seen in **Figure 2-4** (40069, 40410, 40771, 41173) (with a delta of 341, 361, 402, respectively)
- **TTL consistent with Active OS fingerprinting analysis**

  http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html shows that the default TCP TTL for a Linux system is 64. The 211.47.255.21 system is thus expected to be a Linux system. Active OS fingerprinting techniques (nmap –O), and banner solicitation, confirms that 211.47.255.21 is indeed a Linux system. If the source is not spoofed, then the hop distance between the attacker and target is 17 (64-47) hops. A traceroute from the analysts machine to the attacker yielded 23 hops. Given the current 'diameter' of the internet, 17 hops seems reasonable.

3.3 Conclusion: The Source IP Address was probably not spoofed

The only motivation for spoofing the traffic would be to employ a DOS attack, or to communicate a stream of info anonymously to the host. Because the traffic is relatively infrequent, and because less than 10 packets were seen in 2 minutes, it is improbable that the intent of the transmission was to employ a DoS attack (However, the information reaching the destination host would warrant anonymizing one's source address). As will be seen further in the analysis, it appears that the packet signature could have been produced by the hping2 utility, for possibly either recon (which would infer that the packet was not spoofed, assuming the attacker wanted to receive the recon info herself), or for a more sophisticated idle scan attack.

To illustrate further evidence that the data was not spoofed, the incrementing TCP source port data can be countered by considering the consistency in the delay between what appears to be sub-instantiations in the decaying SYN packet retransmission rate, which hardly vary from ~11 seconds between sub-instantiations. Maybe the tool is capable of repeating itself? Maybe the attacker wrote a wrapper script and fed the tool believable source port info? In any case, the consistency between decaying SYN retransmissions implies a possible false statefulness in the tool that suggest the ability to replicate a valid TCP stacks tendency to randomly increment source ports, or possibly the NIDS is configured in such a way that it is considering the retransmit packets as part of a 'stream' and has timed out prior to the last few retransmit attempts.

4. Description of attack:

The following log summary represents the packet frequency characteristics:

```
07/11-16:35:37.044488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40069 -> x.y.90.10:0
07/11-16:35:40.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40069 -> x.y.90.10:0
07/11-16:35:46.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40069 -> x.y.90.10:0
07/11-16:35:58.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40069 -> x.y.90.10:0

07/11-16:36:09.044488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40410 -> x.y.90.10:0
07/11-16:36:12.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40410 -> x.y.90.10:0
07/11-16:36:18.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40410 -> x.y.90.10:0
07/11-16:36:30.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40410 -> x.y.90.10:0

07/11-16:36:41.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40771 -> x.y.90.10:0
07/11-16:36:44.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40771 -> x.y.90.10:0
07/11-16:36:50.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40771 -> x.y.90.10:0
07/11-16:37:02.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:40771 -> x.y.90.10:0

07/11-16:37:13.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:41173 -> x.y.90.10:0
07/11-16:37:16.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:41173 -> x.y.90.10:0
07/11-16:37:22.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:41173 -> x.y.90.10:0
07/11-16:37:34.034488 BAD TRAFFIC tcp port 0  {TCP} 211.47.255.21:41173 -> x.y.90.10:0
```

Figure 2-4 Summary of TCP Destination Port 0 Detects (each box represents sub-instantiation, with 11 second interval between sub-instantiations)

The following packet layout represents the data seen in each of the 12 packets listed in **Figure 2-4**. Each of the 12 packets was analyzed at the byte level, and because **only** expected variations in: TCP source port, TCP sequencing number, and TCP checksums were observed, only the first packet in **Figure 2-4** has been broken down for analysis, as listed below in **Figure 2-5**.

[**] BAD TRAFFIC tcp port 0 traffic [**]																	
07/11-16:35:37.044488	0:3:E3:D9:26:C0	->	0:0:C:4:B2:33	type:0x800	len:0x42												
211.47.255.21:40069	->	x.y.90.10:0	TCP	TTL:47	TOS:0x0	ID:0	IpLen:20	DgmLen:52	DF								
******S*	Seq: 0x5CAAA0AB	Ack: 0x0	Win: 0x16D0	TcpLen: 32													
Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0																	
0x0000	00	00	0C	04	B2	33	00	03	E3	D9	26	C0	08	00	45	00	.....3....&...E.
0x0010	00	34	00	00	40	00	2F	06	F8	76	D3	2F	FF	15	2E	05	.4..@./..v./....
0x0020	5A	0A	9C	85	00	00	5C	AA	A0	AB	00	00	00	00	80	02	Z.....\.........
0x0030	16	D0	6B	1F	00	00	02	04	05	B4	01	01	04	02	01	03	..k.............
0x0040	03	00															..

Figure 2-5 Packet Layout of each of the detects listed in Figure 2-4 (only variations for other packets are: TCP source port, TCP checksum, Sequence number)

5. Attack mechanism:

It is suspected that, at the least, the attacker is gaining reconnaissance info around x.y.90.10's network connectivity. Port 0 'pings' are characteristic of hping default port usage, and are often used for idle scanning techniques. Because the destination port is 0, it is suspected that the Attacker may be exploiting the fact that an idle host will send RST/ACK's while providing a streaming list of constant IP ID increments. If x.y.90.10 is a MS Windows based host, then it will increment it's IP ID's predictably (in increments of 256). The attacker can then port scan yet another system, spoofing the source address to be x.y.90.10, and correlate the spurious IP ID activity returned in first scan to the time that a particular spoofed source target port was scanned. This procedure is well described in the http://rr.sans.org/audit/hping2.php idle host scanning paper, with relevant hping command line syntax that could create the observed phenomena.

Additionally, as discussed in Ronald Clark's GCIA detect submission for a similar port 0 detect from the 211.47.255.21 host, it is speculated that the attacker is using the hping tool (www.hping.org) as the method to create the stimuli. Investigation of the hping tool suggested that it would not have the ability to retransmit SYN scans at the decayed frequency as seen with this scan alone (hping sends probes at 1 second intervals). However, a wrapper script could have been written around hping to simulate the observed behaviors. It is conceivable that the technique listed in http://rr.sans.org/audit/hping2.php has been scripted in such a way that the effects on the idle system look as benign as possible; this could be accomplished by mimicking the SYN packet retransmission decay frequency of commonly implemented TCP stacks as has apparently been done in **Figure 2-4**. Each 'sub-instantiation' may be used to monitor the returning RST/ACK IP ID for a spoofed 3rd party scan. If so, then the 3 'scan's listed in **Figure 2-4** may represent 3 target system scans.

Nmap was also investigated as a possible stimuli source, but in it's un-altered form, is incapable of directing TCP port scans at port 0, and also defaults to sending probes at 3 second intervals, without means to limit scans to 4, as was seen in **Figure 2-4**

It should also be noted that Check Point's Firewall-1 product is known to have DoS vulnerabilities to UDP/port 0 traffic. It is conceivable that that such a vulnerability has inspired the BlackHat community to increase experimentation other variations on port 0 attacks.

6. Correlations:

An analysis was done on all binary logs present during the time of download at www.incidents.org/logs/Raw, no other traffic patterns beyond the IP ID 0 / TCP Destination Port 0 signature were found. A query to www.dshield.org showed a significant number of records for port 80 queries as late as 7/29/02 from the 211.47.255.21 host

- http://cert.uni-stuttgart.de/archive/intrusions/2002/08/msg00082.html
- http://www.iss.net/security_center/static/3233.php
- http://rr.sans.org/audit/hping2.php

7. Evidence of active targeting:

The Scan was seen to query the following IP addresses exactly 16 times each:

x.y.104.69, x.y.106.99, x.y.148.130, x.y.169.123, x.y.174.171, x.y.214.181, x.y.227.153, x.y.76.25, x.y.91.253

This possibly suggests active targeting by the attacker. Possibly, the attacker has determined these to be relatively idle hosts, and thus useful for indirect scanning of other target hosts. While such specific targeting implies that the attacker may be carefully selecting targets for use in sophisticated indirect idle scanning methods, it may also be possible that the detects are merely a result of an attacker is scanning the entire x.y.0.0/16 network space; where we are only see traffic on the above subnets because that is where the NIDS is monitoring. More information is needed to eliminate doubt.

8. **Severity:**

Category	Explanation	Score (0 to 5)
Target Criticality	considered non-critical as no other attacks were made upon listed targets from source addresses	2
Attack Lethality	attack would have only elicited TCP RESET/ACK packets with no other victim consequence	1
System Countermeasures	no TCP RESET/ACK's were noticed in the investigated logs, possibly due to personal system firewalls such as Black Ice or TCP Wrappers	2
Network Countermeasures	internal firewall/ filter my be blocking outbound TCP RESET/ACK's prior to reaching monitored networks	2
Severity (TC + AL – SC –NC)		-1

9. **Defensive recommendation:**
TCP/UDP Destination/Source Port 0 traffic should be dropped at perimeter firewall. The firewall should preferably be stateful and be ran with a default deny configuration, allowing for only traffic explicitly deemed acceptable.
10. **Multiple choice test question:**
For which incrementing IP Protocol field on an idle host would an attacker monitor while spoofing packets to the intended victim in a conventional 'idle host scan' scenario?
a. IP Time to live
b. IP Identification number
c. Fragment offset
d. Type of Service

Correct Answer: b. A monitored IP ID number on an idle host would increase irregularly if it's IP stack had to send a RST/ACK in response to the fooled victim's SYN/ACK response to an open port request spoofed by the attacker.

Network Detect 2: Successful DNS Zone Transfer

1. **Source of Trace:**
The following trace was obtained from the www.incidents.org/logs/Raw/ repository. The specific files including the trace can be found at www.incidents.org/logs/Raw/ under the directories respective to the dates in which the packets were collected.
2. **Detect was generated by:**
The detect was generated by the Snort Intrusion Detection System Version 1.8.6 (Build 105). The rule-set used was a standard snortrules-stable.tar.gz set downloaded

from http://www.snort.org/dl/signatures/ on 6/10/02 19:03 PM. All rule-sets are being monitored. The specific rule responsible for detecting the anomalous network behavior was extracted from the dns.rules file included with the standard signature set, and is listed as follows:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS zone transfer"; flags:A+;  content: "|00 00
FC|"; offset:13; reference:cve,CAN-1999-0532; reference:arachnids,212; classtype:attempted-recon; sid:255;
rev:5;)
```

Figure 2-1 Snort Rule for DNS Zone Transfer

The $HOME_NET variable in the above rule has been specified to represent the x.y.0.0/16 class B subnet. $HOME_NET represents all traffic local to the monitored systems on the x.y.0.0/16 network.

The standard Snort 1.8.6 (Build 105) TCP full format as produced by the '-X' argument is described below in **Figure 2-2**. (The number of bits per field is in parentheses)

<The message from the rule that matched the packet>									
MM/DD-HH:MM:SS.xxxxxx	Source MAC address	->	Dest MAC addres	Ether Type	Ether Length				
Source IP: TCP Source Port	->	Dest IP: TCP Dest Port	IP Proto	IP Type of Service	IP TOS	IP Frag ID	IP Header Length	IP datagram Length	Fragment Flags
IP Flags UAPRSF	Sequence Number	Acknowledge number	TCP Window Size	TCP header length					
TCP Options(n) => Opt1 Opt2 Opt3 … Opt(n)									

0x0000	Ethernet Destination Address(48)	Ethernet Source Address(48)	Ether Type (16_	IP Ver IpLen TOS (16_	ASCII rep. of 32 Byte Stanza				
0x0010	IP DgmLen (16)	IP ID (16)	IP Frag Offset/ ID (16)	IP TTL (8)	IP prot (8)	IP Header chksum (16)	IP Source Addr (32)	IP Dest Addr (1-16 of 32)	ASCII rep. of 32 Byte Stanza
0x0020	IP Dest addr (17-32 of 32)	TCP Src port (16)	TCP Dest Port(16)	TCP Sequence Number (32)	TCP Acknowledgement Number(32)	TCP Hdr len(4) Res Bits(6) Flags(6)	ASCII rep. of 32 Byte Stanza		
0x0030	TCP Window Size (16)	TCP Chksum (16)	TCP Urgent Pointer (16)	TCP Options (1-80 of 96)	ASCII rep. of 32 Byte Stanza				
0x0040	TCP Options (81-96 of 96)	DNS ID (16)	DNS Flags (16)	DNS # of questions (16)	DNS # of Answer RR's (16)	DNS Questions (variable length)	ASCII rep. of 32 Byte Stanza		
0x0050	DNS Answers (variable length)	DNS Authority (variable length)							
0x0060	DNS Additional Info (variable length)								

Figure 2-2 Snort TCP Packet Key

The purple cells represent the Ethernet protocol data, the green cells represent the IP protocol data, and the yellow cells represent the TCP protocol data. DNS data is represented in BLUE and will be further broken down in Figure 2-5 through 2-8.

3. **Probability the source address was spoofed:**

The following data will support the fact that the trace is **not** spoofed:

- **TCP Timestamp options**

In Karen Fredrick's description of TCP options at http://online.securityfocus.com/infocus/1223 , she presents an excellent description of the TCP Timestamp options documented in RFC1323. TCP TS options enable the two endpoints to compensate for line latency in the acknowledgement timer. Because both endpoints must agree to use this TCP feature, it is reasonable to assume that the captured packets represent a valid TCP stream.

- **TCP Ack/Seq reasonability**

  Sequence and Acknowledgement fields, as seen in **Figure-2.5.0** and **Figures-2.5.1** contain valid values and thus suggest the existence of a valid TCP zone transfer connection.
- **IP TCP TTL reasonability**

  The IP TTL as seen in **Figure-2.5.0** and **Figures-2.5.1** appear to be legitimately decremented from a probable value of 64. A 14 hop distance between the attacker and target is reasonable considering the current 'diameter' of the internet.

4. Description of attack:

The following data in **Figure 2-4** shows the summary DNS zone transfer detects between 6/03 and 7/03. In **Figure 2-5**, the full packet breakout for the first log is displayed.

06/03-14:20:33.934488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1116 -> x.y.180.250:53
06/07-09:33:24.594488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1093 -> x.y.180.250:53
06/07-10:05:42.754488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1098 -> x.y.180.250:53
06/11-09:15:26.564488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1095 -> x.y.180.250:53
06/11-12:18:46.814488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1110 -> x.y.180.250:53
06/24-11:06:48.224488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1100 -> x.y.180.250:53
06/28-07:55:56.414488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1084 -> x.y.180.250:53
07/02-08:28:21.544488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1088 -> x.y.180.250:53
07/03-10:29:02.774488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1099 -> x.y.180.250:53
07/03-10:52:27.734488 [**] "DNS zone transfer [**] {TCP} 216.30.135.34:1107 -> x.y.180.250:53

Figure 2-4

Figures 2-5.0/1 illustrate 2 examples of the DNS TCP traffic. In these diagrams we see that the Ethernet/IP/TCP data has been specified in addition to a protocol header breakdown of the DNS protocol cargo. A description of the DNS protocol is provided in **Figure 2-6.**

[**] "DNS zone transfer [**]									
06/03-14:20:33.934488	0:3:E3:D9:26:C0	->	0:0:C:4:B2:33	type:0x800	len:0x5E				
216.30.135.34:1116	->	x.y.180.250:53	TCP	TTL:50	TOS:0x0	ID:2191 9	IpLen:2 0	DgmLen:80	DF
AP	Seq: 0x7F8DE42E	Ack: 0xCB8DE4D F	Win: 0x7D78	TcpLen: 32					
TCP Options (3) => NOP NOP TS: 6321811 322642702									

0x0000	00	00	0C	04	B2	33	00	03	E3	D9	26	C0	08	00	45	00	.....3....&...E.
0x0010	00	50	55	9F	40	00	32	06	B6	CE			87	22			.PU.@.2......"...
0x0020	B4	FA	04	5C	00	35	7F	8D	E4	2E	CB	8D	E4	DF	80	18	...\.5..........
0x0030	7D	78	66	C9	00	00	01	01	08	0A	00	60	76	93	13	3B	}xf........`v..;
0x0040	23	0E	00	1A	E9	B3	00	00	00	01	00	00	00	00	00	00	#..............
0x0050	04	58	58	58	58	03	63	6F	6D	00	00	FC	00	01			.XXXX.com.....

Figure 2-5.0

[**] "DNS zone transfer [**]																
06/07-10:05:42.754488	0:3:E3:D9:26:C0	->	0:0:C:4:B2:33	type:0x800	len:0x5E											
216.30.135.34:1098	->	x.y.180.250:53	TCP	TTL:50	TOS:0x0	ID:49696	IpLen:20	DgmLen:80	DF							
AP	Seq: 0xB4DCDD56	Ack: 0x20E7CFD	Win: 0x7D78	TcpLen: 32												
TCP Options (3) => NOP NOP TS: 4793722 355676055																
0x0000	00	00	0C	04	B2	33	00	03	E3	D9	26	C0	08	00	45	00	.....3....&...E.
0x0010	00	50	C2	20	40	00	32	06	4A	4D			87	22			.P. @.2.JM..."..
0x0020	B4	FA	04	4A	00	35	B4	DC	DD	56	02	0E	7C	FD	80	18	...J.5...V..|...
0x0030	7D	78	B4	12	00	00	01	01	08	0A	00	49	25	7A	15	33	}x.........I%z.3
0x0040	2F	97	00	1A	E2	16	00	00	00	01	00	00	00	00	00	00	/...............
0x0050	04	58	58	58	58	03	63	6F	6D	00	00	FC	00	01			.XXXX.com.....

Figure 2-5.1

(Stevens, pg 191-194) describes the DNS protocol fields relative to the above packet as such:

Location	Width(bits)	Description
0x0000 – 0x0041		Ethernet/IP/TCP header
0x0042 – 0x0043	16	DNS ID: enables client to match responses with requests
0x0044 - 0x0045	16	DNS Flags See **Figure 2-7**
0x0046 - 0x0047	16	DNS Number of Questions:
0x0048 – 0x0049	16	DNS Number of Answer Resource Records
0x004A – 0x004B	16	DNS Number of Authority Resource Records
0x004C – 0x004D	16	DNS Number of Additional Resource Records
🟩	Variable	Questions
🟩	Variable	Answers
🟩	Variable	Authority
🟩	Variable	Additional Information

Figure 2-6

From (Stevens, page 192, Figure 14.4) The numbers in parenthesis represent bits dedicated to protocol field

QR(1)	Opcode(4)	AA(1)	TC(1)	RD(1)	RA(1)	(zero)(3)	Rcode(4)

Figure 2-7

QR	0 indicates that message is a query, 1 indicates that it is a response
Opcode	0=standard query, 1=inverse query, 2=server status request
AA	1 indicates that server's answer is authoritative
TC	1 indicates that response was truncated, prompts client to resubmit request with TCP
RD	1 indicates request for server recursion, basically saves trouble of client from having to query multiple DNS servers (root DNS servers typically don't recurse)
RA	1 indicates a response from server that it is willing to perform recursion
Zero	3 zero bits specified as filler
Rcode	Return code: 0 indicates no error, 3 indicates that requested domain name doesn't exist.

Figure 2-8

Using **Figure 2-7** and **2-8**, we can infer the purpose of the detected packets **in Figure 2-9.**

Date	0x044-0x045	QR	O1	O2	O3	O4	AA	TC	RD	RA	Z1	Z2	Z3	R1	R2	R3	R4
06/03-14:20:33	E9 B3	1	1	1	0	1	0	0	1	1	0	1	1	0	0	1	1
06/07-09:33:24	EB 1C	1	1	1	0	1	0	1	1	0	0	0	1	1	1	0	0
06/07-10:05:42	E2 16	1	1	1	0	0	0	1	0	0	0	0	1	0	1	1	0
06/11-09:15:26	08 1D	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0	1
06/11-12:18:46	73 59	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	1
06/24-11:06:48	6C 1F	0	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1
06/28-07:55:56	4A 09	0	1	0	0	1	0	1	0	0	0	0	0	1	0	0	1
07/02-08:28:21	F0 0F	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
07/03-10:29:02	11 05	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1
07/03-10:52:27	31 97	0	0	1	1	0	0	0	1	1	0	0	1	0	1	1	1

Figure 2-9

Bits listed in orange are '1's, when according to (Stevens, pg 190), they should always be zero. This suggests a possible inappropriate use of the DNS protocol.

5. **Attack mechanism:**
An attacker's motivation for conducting zone transfers is that highly valuable reconnaissance data is available by collecting information on entire DNS zones, that could be used to plan and develop attacks on machines within that zone.
Assuming that what we are seeing is indeed a zone transfer, the question most relevant to this analysis becomes whether or not the zone transfer is legitimate, or a reconnaissance attempt. (Stevens, pg 190) tells us that legitimate zone transfers are expected to only be initiated by a pre-determined secondary DNS slave server to it's primary(master) server. Under no other cases should a zone transfer occur.
To infer upon the potential legitimacy of the traffic, let us assume that the traffic is legitimate, and evaluate the implications. Assuming that the zone transfer is legitimate, then 216.30.135.34 is a legitimate slave to the x.y.180.250 master. By considering the TTL of 50 for each of the captured packets, and assuming that the default TTL is 64 (see http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html), then we see that there is an approximate **14 hop gap between the Master and Slave**. Typical network configurations would place a slave and master closer than 14 hops away. Additionally, we see that each zone transfer at least included information for xxxx.com. This assumes default TTL's were not modified on the source host.
Inconsistencies in the above scenario suggest that the traffic is not legitimate. It would appear that if the packets were not modified to obfuscate some other domain name with xxxx.com (which is a legitimate DNS name resolving to 216.131.110.169, and hence a bad obfuscation name), then xxxx.com is within the zone of the 216.30.135.34 (Austin.smsc.com) and of x.y.180.250. Without comprehensive understanding of the purposes of the x.y.180.250 node, it is not possible that Austin.smsc.com could be the dns server for a different zone than it's own (i.e., xxxx.com is not within smsc.com, as it would have to be xxxx.smsc.com to be within Austin.smsc.com's delegated zone)
If xxxx.com is an obfuscation of smsc.com, then it is conceivable that 216.30.135.34 is a legitimate slave looking for zone information for the smcs.com domain. Still, it is suspicious that the DNS Master and Slave are 14 hops removed.
6. **Correlations:**
The attacker IP address 216.30.135.34 was search for in all logs available during the time of analysis within www.incidents.org/logs/Raw , no detects other than DNS TCP were found to originate from this source. Additionally, no correlations were found at

http://www.dshield.org for this address. It is conceivable that the DNS administrator at the remote site is exploiting the target as a misconfigured DNS server.

- http://www.giac.org/practical/Joseph_Rach.html

7. **Evidence of active targeting:**

   All detects from the attacker IP address 216.30.135.34 were directed towards only on victim, x.y.180.250. This is indicative of active targeting. The large period of time between the scans is suggestive that the attacker has gained knowledge of the x.y.180.250 target in a node list and is occasionally scanning it to identify whether any DNS domain data is available.

8. **Severity:**

Category	Explanation	Score (0 to 5)
Target Criticality	From using Snortsnarf to analyze all data in www.incidents.org/logs/Raw repository, it appears that this victim is a commonly targeted machine, inferring that it is a possibly critical or open machine as perceived by attackers	4
Attack Lethality	The attack was a network reconnaissance probe.)	1
System Countermeasures	It would appear that the DNS server at this site allows for zone transfers to any interested parties	0
Network Countermeasures	internal firewall/ filter my be blocking outbound further zone transfer information, as packet transmission frequency is very low	2
Severity (TC + AL – SC –NC)		4

9. **Defensive recommendation:**

   Block all TCP port 53 traffic initiating from sources external to the home network. On the DNS machine itself, block inbound TCP connections to all hosts except known secondary servers.

   (Liu and Albitz, Section 10.11.3) describes a method of preventing unauthorized zone transfers in BIND8 by specifying the ‘zone-transfer’ switch in the zone statement to only legitimate secondary slave servers.

   From (Liu and Albitz, Section 10.11.3):

```
zone "acmebw.com" {
                type master;
                file "db.acmebw";
                allow-transfer { 192.168.0.1; 192.168.1.1; };
};
```

Could be used to limit TCP DNS zone transfers from the 192.168.0.1, and 192.168.1.1 slave servers, respectively. Additionally, because BIND 8 by default allows for world access to zone transfers, it is suggested that zone transfers be disabled on the slave systems, as DNS slaves can’t have DNS slaves themselves, respectively.

10. **Multiple choice test question:**

   What datagram size below would be the clearest indication of illegitimate DNS TCP

traffic?
a. A DNS TCP packet with datagram size of 350 bytes?
b. A DNS TCP packet with datagram size of 530 bytes?
c. A DNS TCP packet with datagram size of 800 bytes?
d. A DNS TCP packet with datagram size of 514 bytes?

Correct Answer: a. DNS TCP datagrams are used to respond to queries where the answer is more than 512 bytes. A TCP datagram size of 350 contains a sufficiently small amount of information that would be transferred with the UDP protocol.

Network Detect 3: Low and Slow 41530 UDP Broadcast Strangeness

1. **Source of Trace:**

The UDP broadcast packets were captured on a network described in **Figure 3-1** below. The corporate WAN as described in the diagram connects approximately 120 thousand systems and spans multiple continents. The NIDS sensors are connected to high-importance stub networks supported by 10 Megabit Ethernet Hubs. The NIDS sensors clocks are sustained by a stratum 4 NTP server. The sensors are 550 MHz Pentium3 workstations running a stripped down RedHat Linux 7.2 OS with a dedicated non configured NIC dedicated to snort monitoring.

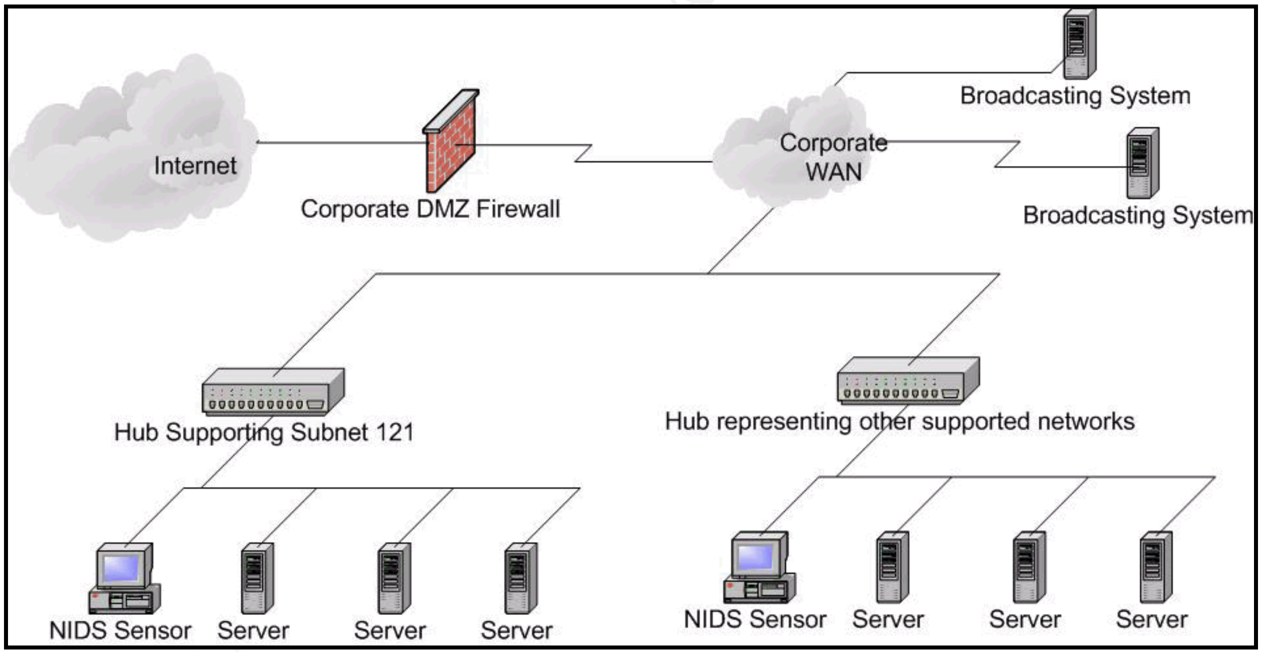


Figure 3-1: Source of 41530 UDP Broadcast Traffic

2. **Detect was generated by:**

The detect was generated by the Snort Intrusion Detection System Version 1.8.7 (Build 128). The rule-set used was a standard snortrules-stable.tar.gz set downloaded from http://www.snort.org/dl/signatures/ on 9/02/02 10:35 AM. All rule-sets are being monitored in addition to a local.rules modified rule-set including the following rule, which detected the packets of interest.

```
alert udp !$INTERNAL_NET any -> $INTERNAL_NET 41530 (msg: "Inbound 41530 udp traffic""; )
```

The $INTERNAL_NET variable in the above rule has been specified to represent the the aaa.bbb.121.0/24 class C subnet. $INTERNAL_NET represents all traffic local to the monitored systems on the aaa.bbb.121.0/24 network, and was selected based upon the business requirements of the monitored network.

Because the monitored subnet exists on an Intranet behind a firewall, a lower frequency of snort rule-set detects enables the rule-set to be augmented by generic rules such as the one used to detect the network traffic of interest without an excessive number of NIDS detects.

The Snort Sensor produced the following type of packet every ~3 hours(note the approximation) over a 24 hour period based upon the rule above:

[**] Inbound 41530 udp traffic [**]																	
09/09-03:33:16.193328	xxx.yyy.129.84:1029								->			aaa.bbb.121.255:41530					
UDP	TTL:116			TOS: 0x0				ID:33800			IpLen:20			DgmLen: 127			Len: 107
0x0000	FF	FF	FF	FF	FF	FF	00	03	31	B7	D0	80	08	00	45	00	........1.....E.
0x0010	00	7F	84	08	00	00	74	11	10	E5			81	54			......t......T..
0x0020	79	FF	04	05	A2	3A	00	6B	F1	4A	9C	4A	50	53	53	51	y....:.k.J.JPSSQ
0x0030	4C	30	30	34	00	00	00	00	00	00	00	25	00	00	00	AC	L004.......%....
0x0040	1D	81	54	4A	50	52	44	30	31	00	00	00	00	00	00	00	..TJPRD01.......
0x0050	00	00	00	01	00	68	02	A2	94	00	00	00	25	00	00	00	.....h......%...
0x0060	00	00	00	00	00	02	3C	00	00	00	00	00	05	00	00	00	......<.........
0x0070	32	32	32	32	32	32	32	32	00	B4	EA	0E	11	00	00	00	22222222........
0x0080	00	00	00	00	00	00	00	00	00	00	00	00	00				………….

Figure 3-2.

The above packet represents the standard Snort 1.8.7 (Build 128) format as produced by the '-X' argument, used to dump raw packet data including the link layer. Interpretation of the above output can be made from the following key (where B=1 Byte = 8 bits = 0.25 Words)

The message from the rule that matched the packet																	
MM/DD-HH:MM:SS.xxxx			Source IP:Source Port								->		Destination IP:Destionation Port				
IP Protocol	IP Time To Live			IP Type of Service				IP (Fragment) ID				IP Header Length		IP datagram Length			UDP Length
0x0000	Dest Ethernet Address(48)						Source 48 bit Ethernet Address(48)						Ether Type (16)	IP Ver IpLen TOS(16)			ASCII rep. of 32 Byte Stanza
0x0010	IP DgmLen (16)		IP ID (16)		IP Frag (3) Offset/ID (13)		IP ttl (8)	IP proto (8)	IP Hdr chksum (16)		IP Source Addr (32)				IP Dest Addr (32)		ASCII rep. of 32 Byte Stanza
0x0020	IP Dest addr(32)		UDP Src Port(16)		UDP Dest Port (16)		UDP Length (16)		UDP Cksum (16)		B B	B B	B B	B B	B B	BB	ASCII rep. of 32 Byte Stanza
0x0030	BB	BB	B B	B B	BB	BB	BB	BB	BB	BB	B B	B B	B B	B B	B B	BB	ASCII rep. of 32 Byte Stanza

0x0040	BB	BB	B B	B B	BB	BB	BB	BB	BB	BB	B B	B B	B B	B B	B B	BB	ASCII rep. of 32 Byte Stanza
0x0050	BB	BB	B B	B B	BB	BB	BB	BB	BB	BB	B B	B B	B B	B B	B B	BB	ASCII rep. of 32 Byte Stanza
32 Byte Stanza	BB	BB	B B	B B	BB	BB	BB	BB	BB	BB	B B	B B	B B	B B	B B	BB	ASCII rep. of 32 Byte Stanza
32 Byte Stanza	BB	BB	B B	B B	BB	BB	BB	BB	BB	BB	B B	B B	B B				ASCII rep. of 32 Byte Stanza

Figure 3-3

The purple cells represent the Ethernet protocol data, the green cells represent the IP protocol data, and the yellow cells represent the UDP protocol data.

The dark blue cells compose the UDP packet cargo, and will be broken down appropriately in section 4. Description of Attack, below.

3. **Probability the source address was spoofed:** The following analysis is intended to support the conclusion that the traffic is *not* spoofed. The traffic will be broken down and presented with arguments supporting both contentions prior to reaching this conclusion.
   Data supporting probability that source address was spoofed:
   - **IP Protocol Type: UDP**
     Because the IP protocol is UDP, the transmission is considered to be connection-less. Unlike UDP, TCP is a connection oriented protocol, delivering the following services to applications: virtual circuits, application i/o management, network i/o management, flow control and reliability.[Hall, pg 270-271] These services requires correspondence between the source and destination endpoints, specifically, flow control and reliability employ the tracking of sequence and acknowledgement number logging to account for packets on both sides of the full duplex connection. Sequence number prediction is a requirement of a successful TCP spoof, and can be fairly difficult to attain on modern operating systems with non-trivial sequence number incrimination algorithms. Because of this difficulty, established TCP connections are very difficult to spoof.
     Unlike TCP, UDP serves as a lightweight protocol alternative where reliability is forsaken for a more lightweight, efficient delivery mechanism. The advantages of UDP can be leveraged for applications which are tolerant to packet loss, or deal with it via their own mechanisims.. Because the application is expected to manage such services built into the TCP protocol, the simplified UDP protocol contains no data fields that would complicate spoofing a 'virtual' connection. Because an attacker could easily craft a data stream to a target, and encapsulate it in UDP packets with bogus source addresses and still be relatively confident that the packets will arrive at the intended destination, an analyst cannot be entirely sure that UDP one way spoofed connections are not being processed in undesirable ways at the destination host/port.
   - **IP Address Destination: Broadcast**
     Because the IP destination address, aaa.bbb.121.255 serves as a broadcast address to the aaa.bbb.121.0/24 class C network, each host configured between aaa.bbb.121.1-254 with a 255.255.255.0 netmask will accept the destination broadcast IP address and pass it up to the (in this case) UDP layer for response. If the UDP destination port specified in the packet reaches a host who has been configured to listen on that UDP port, the packet cargo will be transferred up to the 41530/UDP bound application. Otherwise, the UDP layer will instruct the ICMP layer to construct an ICMP error message of Type 3, Code 3 (Destination Unreachable, Port Unreachable) and direct it back to the source IP address. This inverse mapping

approach could be used as reconnaissance info.

The attacker may employ a denial of service scenario if she was to spoof the source address, such that as many ICMP Error Type 3/ Code 3 messages would be delivered to the spoofed victim as existing hosts who have no applications bound to the UDP port as specified in the destination port section of the UDP header responding to the broadcast IP address with ICMP Type3/Code3's. This creates an opportunity for the spoofer to create a large number of response packets for the cost of spoofing one broadcast packet. This technique is commonly referred to as 'smurfing,' or as the smurf attack.

Data supporting probability that source address was not spoofed:

- **Frequency of Stimuli**

  As can be seen in the **section 4, description of attack**, the frequency of all traffic specifying the xxx.yyy.121.255:1029 endpoint is almost exactly 3 hours and 4 seconds (10804 [*base 10]*, 2A34 [*base 16]*, 25064 [*base 8]* seconds,nope no correlation to 31337 here…) if small fluctuations in network latency are assumed. *As the stimuli are so infrequent, a DOS attack scenario does not look probable.*
- **Anticipated Response**

  A spoofed source address implies that the attack had motivation to spoof the source address. One possible motivation is that the attacker wishes to remain anonymous while effecting systems remotely. This is likely not the case for *this* traffic as all UDP cargo data is identical across packets from similair sources. If the attacker had wished to communicate some information to affect the remote systems listening to the 41530 UDP port remotely, it is a safe assumption that a variation in the cargo data would be necessary to communicate with the systems, barring a highly sophisticated covert channel communications mechanism.
- **Traceroute verification**

  A system from the monitored subnet was used to perform traceroutes back the source address on multiple occasions over multiple days, each rendered a consistent hop count (12) that while added to the TTL (116) tallies to 128, which is the default Microsoft TCP/IP stack TTL for UDP for NT4.0 and Win2k. While it is conceivable that an attacker could have selected the spoofed sources for this reason, it would generally require a more sophisticated attacker.

Conclusion

The only motivation for spoofing the traffic would be to employ a DOS attack, or to communicate a stream of info anonymously to the hosts listening to the destination broadcast address. Because the traffic is so infrequent, and because only one packet is sent every ~3 hours, it is improbable that the intent was to employ a DoS attack, nor is it probable that the information reaching the destination hosts would warrant anonymizing one's source address.

4. Description of attack: (note that all UDP cargo data colored in blue in Figure 3-3 is identical for each of 8 following packets)

[**] Inbound 41530 udp traffic [**]						
09/09-03:33:16.193328	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 33800	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-06:33:20.331654	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 46408	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-09:33:24.556451	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 4740	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-12:33:28.774108	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 53828	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-15:33:33.326148	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 25217	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-18:33:37.443275	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 14739	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/09-21:33:41.957779	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 49019	IpLen: 20	DgmLen: 127	Len: 107

[**] Inbound 41530 udp traffic [**]						
09/10-00:33:46.319598	xxx.yyy.129.84:1029		->	aaa.bbb.121.255:41530		
UDP	TTL: 116	TOS: 0x0	ID: 45325	IpLen: 20	DgmLen: 127	Len: 107

Over the course of the 24 hour (and ~32 second) period, only 8 packets from the source IP address were detected by the sensor. This was confirmed by increasing logging rates to watch for all traffic sourced from the xxx.yyy.129.84 source address. No additional data was detected. Unfortunately, the traffic existed (with same frequency) prior to installation of the snort sensor on the monitored subnet, and hence no insight can be made upon whether any previous activity existed that would deviate from the 3 hour and 4 second broadcast frequencies or packet content.

One possible, although remote possibility is that the attacker is communicating via a covert channel through the IP Fragmentation ID numbers, UDP Checksums, or packet generation frequency (the only variation other than checksums in the entire packet stream, which are generally not manipulate-able). The attacker has brought new meaning to the term 'low and slow' if she only need only communicates 16 Bytes(IP ID), or 8 bytes(UDP Checksum)/~3 hours. Possibly, the very frequency itself serves as a source of information for such a covert channel.

The following analysis will attempt at mapping out the structure of the identical packet cargo structures. Note that 2 other sources were also captured and will be brought into the analysis for use in comparing and contrasting similarities in the data organization of the 41530/UDP broadcast signature (full traces are deemed unnecessary other than information regarding UDP source ports, TTL's and frequencies included in table descriptions):

[**] Inbound 41530 udp traffic [**]																	
09/09-03:33:16.193328	xxx.yyy.129.84:1029						->		aaa.bbb.121.255:41530								
UDP	TTL:116			TOS: 0x0			ID:33800			IpLen:20			DgmLen: 127			Len: 107	
0x0000	FF	FF	FF	FF	FF	FF	00	03	31	B7	D0	80	08	00	45	00	........1.....E.
0x0010	00	7F	84	08	00	00	74	11	10	E5			81	54			......t......T..

0x0020	79	FF	04	05	A2	3A	00	6B	F1	4A	9C	4A	50	53	53	51	y....:.k.J.JPSSQ
0x0030	4C	30	30	34	00	00	00	00	00	00	00	25	00	00	00	AC	L004.......%....
0x0040	1D	81	54	4A	50	52	44	30	31	00	00	00	00	00	00	00	..TJPRD01.......
0x0050	00	00	00	01	00	68	02	A2	94	00	00	00	25	00	00	00	.....h......%...
0x0060	00	00	00	00	00	02	3C	00	00	00	00	00	05	00	00	00	......<.........
0x0070	32	32	32	32	32	32	32	32	00	B4	EA	0E	11	00	00	00	22222222........
0x0080	00	00	00	00	00	00	00	00	00	00	00	00	00				…………

Figure 3-4 Source A: Frequency 3 hrs, 4 seconds, traceroute hop count:12, predicted ttl:128 (see Figure 3-7 for proposed packet structure)

[**] Inbound 41530 udp traffic [**]																	
10/13-06:26:31.982215	ccc.ddd.32.224:1033							->		aaa.bbb.121.255:41530							
UDP	TTL:117				TOS:0x0			ID:54159			IpLen:20			DgmLen:127			Len: 107
0x0000	FF	FF	FF	FF	FF	FF	00	03	31	B7	D0	80	08	00	45	00	........1.....E.
0x0010	00	7F	D3	8F	00	00	75	11	C2	DC			81	54			......u..... ...
0x0020	79	FF	04	09	A2	3A	00	6B	1B	2F	9C	46	4D	53	43	50	y....:.k./.FMSCP
0x0030	45	44	45	56	30	31	00	00	00	00	00	25	00	00	00	0A	EDEV01.....%....
0x0040	13	20	E0	46	4D	52	44	30	31	00	00	00	00	00	00	00	. .FMRD01.......
0x0050	00	00	00	01	00	31	76	14	50	00	00	00	25	00	00	00	.....1v.P...%...
0x0060	00	00	00	00	00	02	3C	00	00	00	00	00	05	00	00	00	......<.........
0x0070	32	32	32	32	32	32	32	32	00	B4	EA	0E	11	00	00	00	22222222........
0x0080	00	00	00	00	00	00	00	00	00	00	00	00	00				....

Figure 3-5 Source B: Frequency 3 hrs, 9 seconds, traceroute hop count:11, predicted ttl:128 (see Figure 3-7 for proposed packet structure)

[**] Inbound 41530 udp traffic [**]																	
10/13-07:27:56.871178	rrr.sss.x.y1:**1027**							->		Aaa.bbb.121.255:41530							
UDP	TTL:116				TOS:0x0			ID:25349			IpLen:20			DgmLen:127			Len: 107
0x0000	FF	FF	FF	FF	FF	FF	00	02	7e	De	30	80	08	00	45	00	........~.0...E.
0x0010	00	7F	63	05	00	00	74	11	8E	05			2E	33			..c...t....!.3..
0x0020	79	FF	04	03	A2	3A	00	6B	7A	63	9C	49	52	53	45	43	y....:.kzc.IRSEC
0x0030	30	31	00	00	00	00	00	00	00	00	00	25	00	00	00	A3	01.........%....
0x0040	21	2E	33	49	52	52	44	30	33	00	00	00	00	00	00	00	!.3IRRD03.......

0x0050	00	00	00	01	00	5E	D2	6A	4C	00	00	00	25	00	00	00	.....^.jL...%...
0x0060	00	00	00	00	00	02	24	00	00	00	00	00	DA	4D	00	00	......$......M..
0x0070	32	31	31	30	36	39	21	36	00	9F	5B	B5	9D	00	00	00	21106916..[.....
0x0080	00	00	00	00	00	00	00	00	00	00	00	00	00				....

Figure 3-6 Souce C: Frequency 3 hrs, 2 seconds, traceroute hop count:12, predicted ttl:128 (see Figure 3-7 for proposed packet structure)

5. **Attack mechanism:**

If it is the case that this data is malicious, one might speculate upon

1. What the functionality might the packet payload contain as illustrated in **Figure 3-4**, **Figure 3-5**, and **Figure 3-6**?
2. What the significance are of the broadcast frequencies?
3. What other information within the packet could transport information between a source and listening system on 41530/UDP?
4. How the traffic would logically fit into a hostile and benign scenario?

The following analysis is intended to support the conclusion that the traffic should be presumed malicious in nature, with the assumption that the detects represent attempts from a trojan'ed machine to contact other compromised machines.

What the functionality might the packet payload contain as illustrated in Figure 3-4, Figure 3-5, and Figure 3-8?

The assumptions in Figure 3-5 can be validated by considering traffic sourced from 2 other sources with similar signatures. The following table listed in **Figure 3-7** will be used to quantify (where possible) and qualify the byte fields and propose explanations of each based upon a comparison and contrast of the 3 packet payloads listed in **Figure 3-4, Figure 3-5, Figure 3-6.**

Byte Range	**Width (Bytes)**	**Comments, suggested purpose**
0x0000-0x0029	84	Ethernet, IP, UDP data discussed above in **Figure 3-3**
0x002A	2	Possibly a placeholder? Consistent across all 3 packets
0x002B-0x003A	32	This field seems to be 32 bytes in length, containing what appears to be a NetBIOS name
0x003B	2	This appears to be a delimiter for the possible NetBIOS name
0x003C-0x0052	64 (including 0x0053-0x0058 and 0x0053-0x0058)	This 64 byte field seems to allow room for a Microsoft Domain in addition to some other ***mysterious*** data(see below)
0x0053–0x0058	12	**Mysterious** data, is this embedded in 0x003C-0x005b space?
0x0059-0x005b	3	See 0x003C-0x0052 comments above
0x005E	2	Another Delimiter?
0x005D-0x008C	16 (96, if counted to end of packet?)	0x005D to 0x008C marks 96 bytes, or 3 32 byte stanzas. It is likely that the remainder of the data inside these 96 bytes is subdivided in some way.

0x0065-0x0066	4	A delimiter?
0x0067-0x006B	10	All zeros in every packet, filler maybe?
0x006C-0x006E	4	Note how this data is consistent for **Figure 3-4** and **Figure 3-5, could this serve as a checksum** for a possible license key specified for locations 0x0070 – 0x0077 ?
0x006E-0x006F	4	More zero byte filler? Modify snort rules as seen in **9. Defensive Recommendations to verify that this isn't unused for covert channel information exchange**
0x0070-0x0077	16	Note how this data is **also** consistent for **Figure 3-4** and **Figure 3-5 , Is this a license key?** Could the source in 3-4 and 3-5 be using the same license key?
0x0078	2	More zero byte filler? Modify snort rules as seen in **9. Defensive Recommendations to verify that this isn't unused for covert channel information exchange**
0x0079-0x007C	8	*More consistency* between **Figure 3-4** and **Figure 3-5**
0x007D-0x008C	32	It is difficult to assume that this last 32 byte trailer is just filler, since it would seem to do nothing more than 'weigh-down' the packet. Is this a 'turned-off' field? Why would it have been included if just filler? Possibly bad programming of updating/kludging of older software? **This field arouses suspicion probably more than any other in packet payload. Is the 'real' data merely decoy in nature? Is it fields like this that we should focus upon? Address this with snort filters specified in 9. Defensive Recommendation**

Figure 3-7

What the significance are of the broadcast frequencies?

While the packet headers were not included for the packets outlined in **Figure 3-5** and **Figure 3-6** for the sake of brevity, the following table illustrates their dissimilarities:

Packet	**Frequency**	**Source Port**
Figure 3-4	3 hrs, 4 seconds	1029
Figure 3-5	3 hrs, 9 seconds	1033
Figure 3-6	3 hrs, 2 seconds	1027

Figure 3-8

What other information within the packet could transport information between a source and listening system on 41530/UDP?

Other than the dissimilarities listed in **Figure 3-8**, the IP ID's and UDP checksums could conceivably be arbitrarily processed by a compromised system in a covert channel scenario.

How the traffic would logically fit into a hostile and benign scenario?

Assuming the traffic is malicious, it might utilize some of the few variations highlighted in the packets, as well as the broadcast frequency itself, to transfer meaningful information to hosts listening to the aaa.bbb.121.255 broadcast address. While improbable, there is nothing other than error checking that would prevent a sufficiently sophisticated attacker from building a covert communications protocol upon the very concept of packet broadcast frequency itself. Even the reliability issues of such a nasty covert communications protocol could be mitigated at the expense of protocol efficiency, as is seen with the TCP/IP protocol discussion discussed in **section 3.**

Regarding the information which has been observed to vary across the monitored interval is IPID, UDP Checksum, IP Checksum (but this is considered to be unusable in covert channel communications since bad values would prevent delivery due to intermediate routers which would drop the presumed corrupt traffic). Another variation in the 41530 UDP traffic is source IP and source port. What if a node listening to aaa.bbb.121.255 is compromised by a Trojan that makes decisions based upon receiving broadcasts from different likewise infected systems? This seems far-fetched, especially since no such odd traffic has been seen to originate from the aaa.bbb.121.0/24 network with such strange signatures.

Assuming the traffic is not malicious, the most probable explanation regarding why the NIDS sensor sees it is due to mis-configured client software. Maybe the client was mis-configured to broadcast to the aaa.bbb.121.255 broadcast address instead of it's own? This seems improbable as 3 presumably independent sources (whom seem not to be spoofed) are transmitting the traffic to only one of 7 similar monitored stub networks. A weak but possible sign that the traffic is not hostile is that nowhere in the whole trace is the pattern '31337' used. Another possible argument that the traffic is merely a mis-configured client is that the structure of the packets as described in Figure 3-7 tend to suggest that the 'client' is trying to declare itself to someone or something, i.e., broadcasting it's NetBIOS name and domain affiliation. The 'license' number section suggests that maybe the 'client' is also trying to validate itself against some listening server, maybe in hopes of convincing that server to send it updates. This doesn't hold much water considering 3 seemingly independent sources have taken interest with our aaa.bbb.121.0/24 subnet, yet no others.

6. **Correlations:**

   A google search for '41530 broadcast UDP' yielded very little correlation with past detects. 1 detect cached from www.incidents.org on May 10, 2000 by Liudvikas Bukys, Liudvikas suggests that this traffic, amongst *other* traffic on 41508, 41524, 41620 is indicative of ARCserve autodiscovery and Inoculan virus software. (source: http://www.sans.org/y2k/051000.htm)

   Michael Roney, in his GCFW practical, mentioned detecting 41530 broadcast traffic at his firewall (http://www.giac.org/practical/Michael_Roney_gcfw.doc). Interestingly, he presents data with broadcasts not only to the .255 byte bounded broadcast address, but also to the .127 and .63 broadcast addresses, used for subnetting. The aaa.bbb.121.127 and aaa.bbb.121.63 addresses do not exist, so the default router may never have constructed a packet that the NIDS would have caught (ARP who-has .121.127,.121.63 broadcast requests would be useful here). Instead the router likely would have returned a ICMP Error Destination Unreachable, Host Unreachable to the source IP.

7. **Evidence of active targeting:**

   Of the 6 monitored networks, only one consistently receives this signal from the 3 independent sources. This network also contains the most critical system of all the monitored networks, and thus also sees the most routed traffic.

8. Severity:

Category	Explanation	Score (0 to 5)
Target Criticality	The aaa.bbb.121.0/24 network serves as home to a critical site name resolution/authentication server	5
Attack Lethality	The traffic seems to have a benign payload(at least within the time observed), but no legitimate scenario can be presented to vindicate it as innocent	3
System Countermeasures	Port 41530 was not open on any of the systems populating the target subnet, no HIDS systems exist on the targeted network	3
Network Countermeasures	No internal intranet firewalls protect the destination network, a Snort IDS is logging all external traffic	1
Severity (TC + AL – SC –NC)		4

9. Defensive recommendation:

As the possible hostile nature of the traffic cannot be ruled out, It is suggested that, if possible, all systems on the aaa.bbb.121.0/24 subnet be investigated for signs of compromise. Of the systems, Microsoft based systems should be investigated first, as the evidence of default TTL and NetBIOS name and Domain affiliation suggest that the traffic is Microsoft related.

Another effective way of monitoring the subnet would be to watch for any outgoing suspicious activity that would indicate one of the machines receiving the UDP broadcast is not taking some action on it.

Port scans of all systems should be done via a known good port scanning system to ensure that no systems have port 41530/UDP open.

If possible, the owner of the broadcasting systems should be contacted to investigate for signs of compromise, or inappropriately configured software.

For increased granularity, the 0x005D-0x0064, 0x0067-0x006B, and 0x007D-0x008C(especially this one!) zero-filled buffers should be watched to detect any possible deviations from established packet content expectations. This may highlight covert channel action:

```
alert udp $EXTERNAL_NET any -> any 41530 (msg:"Unusual 41530 udp traffic type 1\: 0x005D-0x0064
non zero"; content:!"|0000000000000000|"; offset:93; depth:16;)

alert udp $EXTERNAL_NET any -> any 41530 (msg:"Unusual 41530 udp traffic type 2\: 0x0067-0x006B
non zero"; content:!”|0000000000|"; offset:103; depth:10;)

alert udp $EXTERNAL_NET any -> any 41530 (msg:"Unusual 41530 udp traffic type 3\: 0x007D-0x008C
non zero"; content:!"|00000000000000000000000000000000|"; offset:126; depth:32;)
```

And naturally, incoming 41530 UDP traffic should be filtered at an incoming internal firewall, unless explicitly needed.

See Michael Roneys GCFW practical for further insight on dealing with 41530 incoming broadcast traffic. (http://www.giac.org/practical/Michael_Roney_gcfw.doc)

10. **Multiple choice test question:**
Which of the following sections of the UDP packet would be least helpful in developing a Trojan covert communication path?
    a. IP Fragment ID
    b. IP Checksum
    c. UDP source port
    d. UDP Checksum
    e. UDP Length

Answer: b. Manipulated IP checksums would cause the first intermediate router to drop the packet with a 'doctored' IP checksum under the assumption that the packet has been corrupted. All other answers could be utilized in communicating information between two hosts' IP stacks.

References

[1] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994
[2] Postel, Jon. RFC 793 TRANSMISSION CONTROL PROTOCOL. September 1981. URL: http://www.faqs.org/rfcs/rfc793.html (September 21, 2002)
[3] Hall, Eric. The Internet Core Protocols. Oreilly Publishing. Sebastopol, CA, 2000.
[4] Roney, Michael. "GIAC-GCFW Submission." (October 13, 2002) URL: http://www.giac.org/practical/Michael_Roney_gcfw.doc , 2001

<Back to TOC>

Assignment 3: Analyze This

Executive Summary and Overview

Network Security Audit Summary Overview

Network anomaly data from 3/21/02 through 3/27/02 was analyzed to assess the extent of network anomalies/abuse on the MY.NET Class B Network Block. The data collected represents network activity both to and from systems on the monitored MY.NET network segments detected by Snort NIDS sensors deployed on those respective network segments. Data collected from University NIDS repository is enumerated in table 3-1.

alert020321.gz	oos_Mar.21.2002.gz	scans.020321.gz
alert020322.gz	oos_Mar.22.2002.gz	scans.020322.gz
alert020323.gz	oos_Mar.23.2002.gz	scans.020323.gz
alert020324.gz	oos_Mar.24.2002.gz	scans.020324.gz
alert020325.gz	oos_Mar.25.2002.gz	scans.020325.gz
alert020326.gz	oos_Mar.26.2002.gz	scans.020326.gz
alert020327.gz	oos_Mar.27.2002.gz	scans.020327.gz

Table 3-1: Network Anomaly Data used to generate analysis

An initial review of the alert, out-of-spec, and scan logs was performed to assess the topology of the MY.NET monitored network segments, and to identify alerts that were likely to be non-malicious. Traffic dispositioned as non-malicious was analyzed and verified prior to analyzing the rest of the Alert data aforementioned data set.

To summarize, an analysis was done on internal hosts to determine which show probable signs of having fallen victim to compromise, the following nodes were seen to produce multiple and consistent signs of malicious traffic

1. MY.NET.6.52
2. MY.NET.6.48
3. MY.NET.253.10
4. MY.NET.6.49

All compromised hosts were seen to have participated in only a few variations of activity that would generally be considered malicious, these types include:

1. Microsoft IIS web server Unicode attacks
2. Scanning/Mapping activity to port 65535
3. RPC scanning packets originating from non-ephemeral ports (0, 57)
4. SubSeven scans/ possible keep-alive traffic
5. Back Orifice scans/ possible keep-alive traffic
6. Nmap Ping scans

This finite collection of attack types suggests that one attacker, or a small group of attackers with someone unrefined tools have compromised non-hardened servers in the environment. Most of the scans seem to have been targeted towards the MY.NET.150.0/24, MY.NET.151.0/24, MY.NET.152.0/24, and MY.NET.153.0/24 subnets. It is likely that one or two attackers are trying to get access to web servers on these subnets. Some credit must be given to the attackers for having acquired a diverse knowledge of attack scan tools that typically run on both UNIX based and Microsoft based systems. The attacker's strategy seems to be to scan the targeted subnets with Nmap pings, high UDP port pings, and attack UNIX systems with RPC attacks, and Windows machines with IIS based Unicode vulnerability attacks.

Defense Recommendation:

Install stateful packet filtering firewalls, as described in *(Ziegler, page 81)*, redundantly at the site boundary routers and on equipment routing packets to the MY.NET.150.0/24, MY.NET.151.0/24, MY.NET.152.0/24, and MY.NET.153.0/24 subnets.

Suspicious remote hosts were identified as follows:

1. 211.169.242.108
2. 200.53.87.9
3. 212.179.35.118
4. 61.132.208.63
5. 193.253.202.216

Alert traffic breakouts yielded 369,938 Alerts over the 1-week period, Figure 3-1.

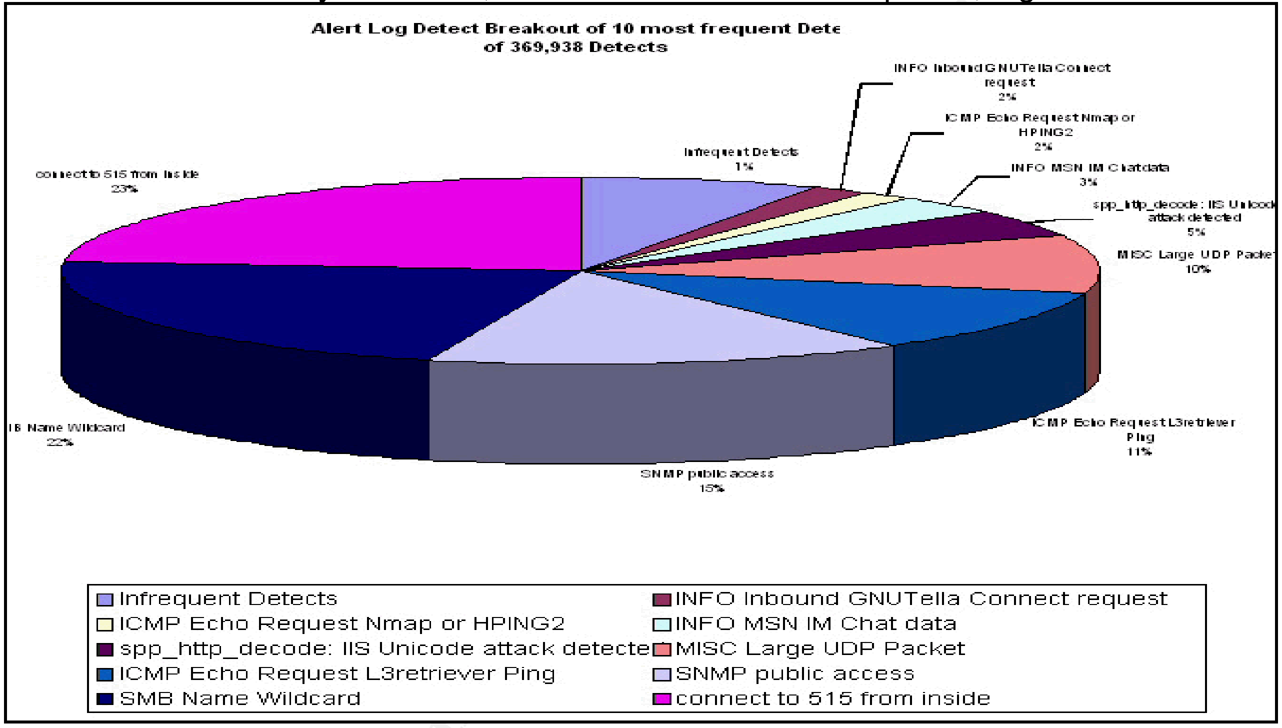


***Figure 3-1**. Alert Log File Breakout*

The remainder of the Audit will be based upon a summarization of the log files listed in table 3.1, followed by an analysis of the 10 most relevant detects. Following a general discussion on the nature and possible defensive tactics of these detects, an analysis will be presented with the goal of identifying the likely compromised local and remote hosts, with corrective and defensive suggestions for each, respectively.

SnortSnarf Summary Analysis

369938 alerts found using input module Snort File Input, with sources:
Earliest alert at **00:00:02**.632155 *on 03/21/2002*
Latest alert at **23:54:52**.240002 *on 03/27/2002*

Signature	# Alerts	# Sources	# Dests
EXPLOIT x86 stealth noop	2	2	2
INFO Inbound GNUTella Connect accept	4	3	3
MISC traceroute	4	1	1

BACKDOOR NetMetro Incoming Traffic	5	1	1
Back Orifice	6	5	6
Port 55850 UDP - Possible myserver activity - ref. 010313-1	6	4	6
EXPLOIT NTPDX buffer overflow	8	5	5
suspicious host traffic	8	6	2
WEB-MISC compaq nsight directory traversal	8	5	5
EXPLOIT x86 setgid 0	9	9	8
EXPLOIT x86 setuid 0	9	9	7
RFB - Possible WinVNC - 010708-1	11	5	5
FTP CWD / - possible warez site	11	1	11
ICMP Destination Unreachable (Protocol Unreachable)	12	1	1
Queso fingerprint	13	11	8
SUNRPC highport access!	13	1	1
Attempted Sun RPC high port access	14	5	10
TCP SRC and DST outside network	16	3	2
WEB-IIS Unauthorized IP Access Attempt	16	3	8
SCAN Synscan Portscan ID 19104	27	26	10
WEB-MISC http directory traversal	32	4	2
WEB-MISC 403 Forbidden	35	8	11
INFO Napster Client Data	36	5	24
Incomplete Packet Fragments Discarded	44	6	6
ICMP traceroute	51	24	6
WEB-CGI scriptalias access	51	5	1
EXPLOIT x86 NOOP	53	19	25
INFO Possible IRC Access	99	18	20
ICMP Destination Unreachable (Communication Administratively Prohibited)	103	1	1
NMAP TCP ping!	107	19	8
Watchlist 000222 NET-NCFC	147	3	3
Port 55850 TCP - Possible myserver activity - ref. 010313-1	173	10	10
INFO napster login	216	1	30
Null scan!	223	34	13
ICMP Echo Request Windows	325	30	13
INFO FTP anonymous FTP	328	9	24
WEB-FRONTPAGE _vti_rpc access	367	116	2
WEB-IIS _vti_inf access	389	127	2
spp_http_decode: CGI Null Byte attack detected	617	14	11
INFO - Possible Squid Scan	754	21	310
WEB-IIS view source via translate header	1008	49	1
ICMP Router Selection	1324	117	1
WEB-MISC Attempt to execute cmd	1621	29	32
SCAN Proxy attempt	1705	35	413
INFO Outbound GNUTella Connect request	1780	10	1196

Possible trojan server activity	1988	19	18
FTP DoS ftpd globbing	2626	29	6
ICMP Fragment Reassembly Time Exceeded	3269	43	97
Watchlist 000220 IL-ISDNNET-990517	3465	17	9
High port 65535 UDP - possible Red Worm - traffic	4562	136	145
INFO Inbound GNUTella Connect request	6307	5147	10
ICMP Echo Request Nmap or HPING2	6376	64	422
INFO MSN IM Chat data	11995	104	101
spp_http_decode: IIS Unicode attack detected	17188	139	678
MISC Large UDP Packet	37460	33	17
ICMP Echo Request L3retriever Ping	40274	117	15
SNMP public access	54756	24	148
SMB Name Wildcard	81135	189	182
connect to 515 from inside	86723	140	5

A Network Services Summary of the MY.NET.0.0/16 subnet

IP	Services		IP	Services
MY.NET.150.198	Network Printer (LPD)		MY.NET.153.174	FTP
MY.NET.11.6	Domain Controller		MY.NET.153.153	NTPD
MY.NET.11.7	Domain Controller		MY.NET.5.96	HTTP
MY.NET.11.5	Domain Controller		MY.NET.150.83	HTTP, FTP
MY.NET.153.204	FTP		MY.NET.88.233	FTP
MY.NET.153.194	FTP		MY.NET.153.191	FTP
MY.NET.150.46	FTP			

Detect 1: “connect to 515 from inside”

- **Top 5 source IP’s triggering this detect signature**

Source	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.153.125	8038	8253	1	21
MY.NET.153.123	5053	5322	1	26
MY.NET.153.171	4636	5248	1	46
MY.NET.153.118	3641	3696	1	14
MY.NET.153.120	3558	3574	1	2

- **All destination network segments receiving this detect signature**

Destination Networks	Detect Description	# Alerts (sig)
MY.NET.150	‘connect to 515 from inside’	86673

MY.NET.5	'connect to 515 from inside'	16
MY.NET.1	'connect to 515 from inside'	34

- **All destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.150.198	86668	86677	136	139
MY.NET.1.63	34	34	2	2
MY.NET.5.35	16	259	1	8
MY.NET.150.114	4	15	1	5
MY.NET.150.41	1	60	1	8

- **Detect Information**

Internal connections to port 515 UDP/TCP (the line printer port) typically would suggest a false positive describing normal printer activity traversing internal subnets. Upon finding that this signature constituted 21.8% of the detect traffic volume, it was determined that the detect traffic should be broken out by network segment. This effort was motivated by a desire to understand whether any particular network segments hosted a majority of the destination and/or source detects. It was found that the MY.NET.150.198 subnet received 99.94% of the traffic. This IP is likely used by a University network printer shared by the monitored subnets MY.NET.153, 256,256,152, MY.NET.149, 256,256,1, MY.NET,169, MY.NET,70.

  - **Application Protocol Specific Information:**
    - Line Printer Daemon Protocol http://www.faqs.org/rfcs/rfc1179.html
  - **CERT Advisories and General Security Information**
    - IDS456/LPR_LPRNG-REDHAT7-OVERFLOW-RDC vulnerability: http://www.whitehats.com/info/IDS456
    - IDS457/LPRNG-REDHAT7-OVERFLOW-SECURITY.IS vulnerability: http://www.whitehats.com/info/IDS457
    - Security problems in the lpd protocol http://www.insecure.org/sploits/lpd.protocol.problems.html
    - A format string vulnerability is documented for various versions of standard Linux builds including Redhat and Caldera here: http://www.cert.org/advisories/CA-2000-22.html.

- **Susceptibility to misdiagnosis / Probable False Positive**

The detects collected by the University NIDS sensors (notwithstanding lower layer internet protocol anomalies) are considered to be non-malicious for the traffic destined to the MY.NET.150.198 address, as the traffic patterns suggest that this traffic is normal printer activity between friendly local networks. None of the traffic sources are outside of the MY.NET network block, nor are any destined outside this block. Further verification could be attained by discussing nature of this destination with local Network and System Administrators, to ensure that such traffic is expected to this address.

Additionally, timestamps were investigated for the MY.NET.150.198 destined traffic, and traffic frequency reflected that which would be thought of as normal for printer activity.

- **Site Security Policy Modification Recommendations**

The following spectrum of policy modifications could be made depending on

University position (from more to less conservative)

1. Unapproved printer daemons could be explicitly prohibited in site security policy.
2. The site security policy could explicitly state that enabling network print services should be immediately followed by a notification to site administrators so that modifications to the 515 snort filter sets can be modified to accommodate the administrator approved list of printer daemons.
3. Site analysts could ignore 'regular' printer activity (that which is not matching predefined exploitation/network mapping signatures) by modifying the port 515 related IDS filter sets.

- **Defensive Recommendation**
  - **Suggested IDS Modifications**: Given the number of cert advisories enumerating vulnerabilities in various printer daemons it is suggested that site IDS filters be modified to watch for specifically known attacks, thereby reducing overall number of false positives for site analysts to follow up upon. The following SNORT rules are provided to monitor for UNIX related printer daemon vulnerabilities:
    - *alert TCP any any -> $HOME_NET 515 (msg:"EXPLOIT LPRng overflow"; flags:A+; content: "|43 07 89 5B 08 8D 4B 08 89 43 0C B0 0B CD 80 31 C0 FE C0 CD 80 E8 94 FF FF FF 2F 62 69 6E 2F 73 68 0A|"; reference:cve,CVE-2000-0917; reference:bugtraq,1712; classtype:attempted-admin; sid:301; rev:3;)*
    - *alert TCP any any -> $HOME_NET 515 (msg:"EXPLOIT redhat 7.0 lprd overflow"; flags:A+; content:"|58 58 58 58 25 2E 31 37 32 75 25 33 30 30 24 6E|"; classtype:attempted-admin; sid:302; rev:2;sid-msg.map:1515 || WEB-CGI input2.bat access || cve,CVE-1999-0947)*
  - **Suggested System Patch Updates**:
    - Linux-Based System Patch Updates:
      The above advisory suggests upgrading your LPD to versions exceeding: 3.6.25:
      ftp://ftp.astart.com/pub/LPRng/LPRng/LPRng-3.6.25.tgz , each of the destination machines should be audited and upgraded to this patch level or above if running a vulnerable implementation of UNIX.
    - Microsoft-Based System Patch Updates
      See www.microsoft.com for latest service pack updates.
  - **Suggested Firewall Modifications**
    Port 515 UDP/TCP traffic should be filtered and logged at the external firewall to ensure that no such traffic arrives at internal hosts on the MY.NET network from non-MY.NET sources.
  - Snort NIDS Tuning Suggestions
    Snort rule sets could be updated with the following rules to ensure that only 515 traffic with known exploit signatures causes a detect, instead of all traffic destined for the MY.NET network:

    alert TCP any any -> $HOME_NET 515 (msg:"EXPLOIT LPRng overflow"; flags:A+; content: "|43 07 89 5B 08 8D 4B 08 89 43 0C B0 0B CD 80 31 C0 FE C0 CD 80 E8 94 FF FF FF 2F 62 69 6E 2F 73 68 0A|"; reference:cve,CVE-2000-0917; reference:bugtraq,1712; classtype:attempted-admin; sid:301; rev:3;)

alert TCP any any -> $HOME_NET 515 (msg:"EXPLOIT redhat 7.0 lprd overflow"; flags:A+; content:"|58 58 58 58 25 2E 31 37 32 75 25 33 30 30 24 6E|"; classtype:attempted-admin; sid:302; rev:2;sid-msg.map:1515 || WEB-CGI input2.bat access || cve,CVE-1999-0947)

- **Correlation**
  No correlations can be made as all traffic is either destined for or originating from the MY.NET network.

Detect 2: "SMB Name Wildcard"

- **Top 5 source network segments triggering detect signature**

Source Networks	Detect Description	# Detects
MY.NET.11	SMB Name Wildcard	39460
MY.NET.152	SMB Name Wildcard	39399
MY.NET.5	SMB Name Wildcard	581
MY.NET.150	SMB Name Wildcard	488
MY.NET.97	SMB Name Wildcard	421

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.11.6	23379	23379	59	59
MY.NET.11.7	14265	14265	59	59
MY.NET.11.5	1816	1819	59	59
MY.NET.152.159	1478	3175	4	8
MY.NET.152.160	965	2167	3	5

- **Top 5 destination network segments receiving this detect signature**

Destination Networks	Detect Description	# Detects
MY.NET.152	SMB Name Wildcard	39480
MY.NET.11	SMB Name Wildcard	39390
MY.NET.5	SMB Name Wildcard	1254
MY.NET.97	SMB Name Wildcard	420
MY.NET.98	SMB Name Wildcard	189

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.11.6	23352	50055	59	59
MY.NET.11.7	14228	30742	59	59
MY.NET.11.5	1810	3636	59	59
MY.NET.152.159	1485	1602	4	10
MY.NET.152.160	976	1016	3	10

- **Detect Information**

The SMB Wildcard request is a standard transaction performed by the NetBIOS protocol implementation by Microsoft for Windows file sharing, name resolution, domain information, and workgroup information. The three most active destinations are very likely acting as domain controllers for University users.

- **Application Protocol Specific Information:**
  - For a good introductory lesson on the topic of SMB and NetBIOS, see the online open source SAMBA book: http://www.oreilly.com/catalog/samba/chapter/book/ch01_03.html .
  - Line Printer Daemon Protocol
  - PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT (CONCEPTS AND METHODS): http://www.faqs.org/rfcs/rfc1001.html
  - PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT (DETAILED SPECIFICATIONS) : http://www.faqs.org/rfcs/rfc1002.html
- **Cert Advisories *(All Descriptions from http://cve.mitre.org)***

Name	Description
CVE-1999-0153	Windows 95/NT out of band (OOB) data denial of service through NETBIOS port, aka WinNuke.
CVE-1999-0288	Denial of service in WINS with malformed data to port 137 (NETBIOS Name Service).
CVE-1999-0407	By default, IIS 4.0 has a virtual directory /IISADMPWD which contains files that can be used as proxies for brute force password attacks, or to identify valid users on the system.
CVE-1999-0810	Denial of service in Samba NETBIOS name service daemon (nmbd).
CVE-2000-0347	Windows 95 and Windows 98 allow a remote attacker to cause a denial of service via a NetBIOS session request packet with a NULL source name.
CVE-2000-0673	The NetBIOS Name Server (NBNS) protocol does not perform authentication, which allows remote attackers to cause a denial of service by sending a spoofed Name Conflict or Name Release datagram, aka the "NetBIOS Name Server Protocol Spoofing"vulnerability.
CVE-2000-0979	File and Print Sharing service in Windows 95, Windows 98, and Windows Me does not properly check the password for a file share, which allows remote attackers to bypass share access controls by sending a 1-byte password that matches the first character of the real password, aka the "Share Level Password" vulnerability.
CVE-2000-1003	NETBIOS client in Windows 95 and Windows 98 allows a remote attacker to cause a denial of service by changing a file sharing service to return an unknown driver type, which causes the client to crash.
CVE-2001-1162	Directory traversal vulnerability in the %m macro in the smb.conf configuration file in Samba before 2.2.0a allows remote attackers to overwrite certain files via a .. in a NETBIOS name, which is used as the name for a .log file.

- **Site Security Policy Modification Recommendations:**
  The following spectrum of policy modifications could be made depending on University position (from more to less conservative)
  1. Clarification to Security policy should prohibit enabling of Windows shares prior to site administrator approval.
  2. Site administrator maintained public file sharing services could be provided and monitored appropriately via internet and intranet firewalls and IDS configurations
  3. Windows file sharing hosts that result from one time requirements for user file transfer, and are not regularly needed by the user after the initial setup should be sought via network scanning scripts and turned off if no longer needed with system owner's permission.
- **Susceptibility to misdiagnosis / Probable False Positive**
  Being a typical occurrence on Microsoft populated networks, SMB wildcard requests need to be analyzed thoroughly to confidently infer malicious intent. As seen above, a majority of the SMB Wildcard detects are between subnets MY.NET.6, MY.NET.7, MY.NET.5, MY.NET.159, MY.NET.160. This traffic likely represents non-malicious NetBIOS traffic between local machines. Detects of greater interest are those originating from outside the MY.NET network block, these detects are listed below:

```
03/21-17:51:10.654524 [**] SMB Name Wildcard [**] 169.254.101.152:137 -> MY.NET.5.96:137
03/21-17:51:12.179402 [**] SMB Name Wildcard [**] 169.254.101.152:137 -> MY.NET.5.96:137
03/21-17:51:13.637037 [**] SMB Name Wildcard [**] 169.254.101.152:137 -> MY.NET.5.96:137
03/21-18:25:07.765735 [**] SMB Name Wildcard [**] 169.254.101.152:137 -> MY.NET.5.96:137
…
03/26-22:13:33.541604 [**] SMB Name Wildcard [**] 169.254.25.129:137 -> MY.NET.5.96:137
03/26-22:13:36.531676 [**] SMB Name Wildcard [**] 169.254.25.129:137 -> MY.NET.5.96:137
```

As both IP's originate from the same class B network block, and that a 5 day period exists between queries, and that the destinations are consistent, a probable explanation of this activity is that an employee or student is attempting to access University shares/resources from a DHCP enabled system at home.

- **Defensive Recommendation**
  - **Suggested IDS Modifications**: Considering that the SMB detects account for 22% of the alerts detected in the time of audit, it is suggested that the Snort NIDS rule set be updated to monitor for external SMB wildcard requests only:
  - **Suggested System Patch Updates**:
    - Linux-Based System Patch Updates:
      Samba UNIX sharing daemons should also be kept to the most recent version, which at the time of this writing is: 2.2.5, and can be found here: http://us1.samba.org/samba/download.html
    - Microsoft-Based System Patch Updates
      See www.microsoft.com for latest service pack updates.
  - **Suggested Firewall Modifications** It is recommended that external traffic originating from ports 137UDP/TCP be logged and possibly filtered at the site firewall.
- **Correlation**
  No correlations were made on traffic that is either destined for or originating from the MY.NET network. www.dshield.org was consulted for two external originations, and no results were found:
  - http://www.dshield.org/ipinfo.php?ip=169.254.25.129
  - http://www.dshield.org/ipinfo.php?ip=169.254.101.152

Detect 3: "MISC Large UDP Packets (for source/destination port 0"

As 10% of the detects are classified as "MISC Large UDP Packet," an attempt was made to identify suspicious patterns within this large data set, the following analysis will be done on suspicious destination ports '0'
This was produced by a procedure described in appendix

- **source IP's triggering detect signature**

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved
rank #1	108 alerts	**202.98.15.138**	1 signatures	MY.NET.153.152
rank #2	107 alerts	**210.94.0.146**	1 signatures	MY.NET.153.196
rank #3	64 alerts	**211.206.125.14**	1 signatures	MY.NET.153.159
rank #4	44 alerts	**63.240.15.204**	1 signatures	(3 destination IPs)
rank #5	30 alerts	**202.101.232.110**	1 signatures	MY.NET.153.159
rank #6	20 alerts	**63.240.15.199**	1 signatures	(3 destination IPs)
rank #7	6 alerts	**202.101.235.110**	1 signatures	MY.NET.153.159
rank #8	5 alerts	**63.250.205.3**	1 signatures	MY.NET.88.183
		140.142.8.72	1 signatures	MY.NET.153.157
		211.233.45.40	1 signatures	MY.NET.152.12
		211.233.45.41	1 signatures	MY.NET.153.196
		216.106.172.150	1 signatures	MY.NET.153.208
Rank #13	4 alerts	**63.211.65.81**	1 signatures	MY.NET.153.197
		63.240.15.205	1 signatures	MY.NET.153.197
		63.240.15.207	1 signatures	MY.NET.153.197
		207.189.78.231	1 signatures	MY.NET.152.13
		211.233.45.59	1 signatures	MY.NET.153.144
		216.106.173.146	1 signatures	MY.NET.153.208
Rank #19	3 alerts	**167.216.132.198**	1 signatures	MY.NET.152.13
Rank #20	2 alerts	**207.189.78.235**	1 signatures	MY.NET.150.46
		211.234.96.29	1 signatures	MY.NET.153.159

- **Destination network segments receiving this detect signature**

Destination Networks	Detect Description	# Detects
MY.NET.153	" MISC Large UDP Packet "	418
MY.NET.152	" MISC Large UDP Packet "	12
MY.NET.88	" MISC Large UDP Packet "	5
MY.NET.150	" MISC Large UDP Packet "	2

- **Destination IP's receiving this detect signature**

Rank	Total # Alerts	**Destination IP**	# Signatures triggered	Originating sources

rank #1	117 alerts	**MY.NET.153.196**	1 signatures	(4 source IPs)
rank #2	108 alerts	**MY.NET.153.152**	1 signatures	202.98.15.138
rank #3	102 alerts	**MY.NET.153.159**	1 signatures	(4 source IPs)
rank #4	48 alerts	**MY.NET.153.184**	1 signatures	(3 source IPs)
rank #5	24 alerts	**MY.NET.153.197**	1 signatures	(5 source IPs)
rank #6	9 alerts	**MY.NET.153.208**	1 signatures	216.106.172.150, 216.106.173.146
rank #7	7 alerts	**MY.NET.152.13**	1 signatures	167.216.132.198, 207.189.78.231
rank #8	5 alerts	**MY.NET.88.183**	1 signatures	63.250.205.3
		MY.NET.152.12	1 signatures	211.233.45.40
		MY.NET.153.157	1 signatures	140.142.8.72
rank #11	4 alerts	**MY.NET.153.144**	1 signatures	211.233.45.59
Rank #12	2 alerts	**MY.NET.150.46**	1 signatures	207.189.78.235
Rank #13	1 alerts	**MY.NET.153.153**	1 signatures	66.28.104.154

- **Detect Information**
  - **Application Protocol Specific Information:**
    - User Datagram Protocol: http://www.faqs.org/rfcs/rfc768.html
    - All data sent to TCP or UDP port 0 should be considered suspicious. At the least, such signatures could easily be used to assume active targeting of a host or network.
  - **Cert Advisories**
    - No UDP port 0 specific cert advisories are known to exist, as stated in TCP related RFC'es, no legitimate traffic should occur on port 0. This suggests that this port 0 traffic exists solely for network mapping by presumably hostile sources.
    - http://www.iss.net/security_center/static/3233.php, and http://online.securityfocus.com/archive/1/23615 state that UDP/port 0 traffic can be used to crash Checkpoint Firewall 1 3.0 and 4.0 through a VPN connection
- **Susceptibility to misdiagnosis / Probable False Positive**
  All traffic with source or destination port 0 should be considered anomalous as no legitimate port 0 traffic should occur, and since there are known attack signatures associated with Checkpoint Firewall vulnerabilities related to UDP/Port 0 traffic
- **Defensive Recommendation**
  - **Suggested IDS Modifications:** The following rules should be employed in the snort rule set to be alerted of future port 0 activity:

    - *alert TCP any any -> $HOME_NET 0 (msg:"Dest TCP Port 0 activity)*
    - *alert TCP any 0 -> $HOME_NET any (msg:"Src TCP Port 0 activity)*
    - *alert UDP any any -> $HOME_NET 0 (msg:"Dest UDP Port 0 activity)*
    - *alert UDP any 0 -> $HOME_NET any (msg:"Src UDP Port 0 activity)*
  - **Suggested System Patch Updates**:
    - Linux-Based System Patch Updates:
      See www.kernel.org for the latest Linux Kernel packages
    - Microsoft-Based System Patch Updates
      See www.microsoft.com for latest service pack updates.
  - **Suggested Firewall Modifications**
    - All packets arriving at the perimeter firewall with a destination port of 0 should be denied and logged. If Checkpoint Firewalls are used, A port 0 UDP ACL should be employed upstream from the firewall to ensure that such traffic is not seen by the Checkpoint firewall.
    - As seen from the '**Destination network segments receiving this detect signature'** table above, it is clear that MY.NET.153 receives a significant amount of incoming anomalous packets. It is recommended that a departmental firewall be placed upon the entry point to this network to specifically protect these machines with a reduced set of allowable packet rules, with increased logging.
    - All remote addresses responsible for port 0 source or destinations should be blocked at the exterior firewall, and their ISP should be contacted and provided with a copy of the logs.
- **Correlation**
  - Summary: Each of the machines that had received a significant amount of port 0 traffic should be audited to determine what listening UDP services are running on each machine. Due to the high fraction of port 0 traffic that was destined to MY.NET.153.196 and MY.NET.153.152, it is probably safe to assume that this traffic was not so much a network mapping motivated attack (by collecting ICMP service unreachable responses) as it is a possible denial of service attack, or possibly Trojan activity. Both the 196 and 152 hosts should be thoroughly audited for system compromise.
  - http://www.dshield.org/warning_explanation.php?fip=63.240.15.204 shows 17 correlations of other port0 activity associated with this host.
  - http://www.dshield.org/warning_explanation.php?fip=63.240.15.199 shows 11 correlations of other port 0 activity associated with this host.
  - http://www.dshield.org/warning_explanation.php?fip=63.250.205.3 shows 29 correlations of other port 0 activity associated with this host.
  - http://www.dshield.org/warning_explanation.php?fip=216.106.172.150 shows 1 correlations of other port 0 activity associated with this host.
  - http://www.dshield.org/warning_explanation.php?fip=63.240.15.205 shows 16 correlations of other port 0 activity associated with this host.

Detect 4: "High port 65535 UDP – possible Red Worm – traffic"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.6.52	1154	1157	83	83

MY.NET.6.48	965	971	70	70
MY.NET.6.49	852	859	83	83
MY.NET.6.50	698	706	79	79
64.124.157.32	175	177	1	1

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.153.153	177	271	14	32
MY.NET.153.46	175	197	1	7
MY.NET.153.209	152	160	8	11
MY.NET.152.171	137	155	6	12
MY.NET.153.150	129	139	7	11

- **Detect Information**
  - CERT Advisories
    No cert advisories are known to exist for the Adore/Red Worm Traffic
  - This traffic appears to be related to the Adore/Red worm known to have infected vulnerable Linux systems since 04/01 [1]. The original Adore worm is known to replace the 'ps' command with a Trojan version and saves the old ps binary as /usr/bin/adore. Further information can be found regarding this Linux worm at http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm , included at this location is a tool called Adorefind, which is capable of scanning network blocks for infected Linux hosts, and capable of fixing and removing associated files. The vulnerabilities that are exploited by this worm include: LPRng, rpc-statd, wu-ftpd, BIND. LPRng (which is open by default on RedHat 7.0 systems) (*Fearnow*). Upon successful inspection, the worm is known to email to the following addresses: adore9000@21cn.com, adore9000@sina.com, adore9001@21cn.com, adore9001@sina.com.
  - Sample traffic that is representative of this signature can be seen in the following SnortSnarf Log excerpt:

```
03/27-15:53:46.120288 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.158:65280
03/27-15:54:10.304664 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.158:65535
03/27-16:07:49.295683 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.158:65535
03/27-16:07:59.020873 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.158:65280
03/27-16:25:33.688903 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:28927 -> MY.NET.153.162:65535
03/27-16:46:50.232855 [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.45:65408
```

- **Susceptibility to misdiagnosis / Probable False Positive**
  The use of port 65535 as both source and destination is an anomalous occurrence in typical UDP traffic and should be considered hostile..
- **Defensive Recommendation**
  - **Patch Updates:** Considering the number of systems shown to have participated in UDP 65535 traffic over the time of analysis, it is strongly recommended that all Linux systems be scanned for evidence of this virus signature with the Adorefind utility and removed as soon as possible.
  - **Firewall Parameters** It is suggested that traffic destined for port 65535 be blocked both internally and externally, also, email to the aforementioned

accounts/systems should be blocked at the site boundary firewall.

  - **Snort NIDS Tuning Suggestions**: The current IDS configuration demonstrates necessary alert capability for port 65535 traffic and shouldn't need updates.

- **Correlation**
  The university systems that produced UDP traffic from and to port 65535 consisted of the following IP addresses:
  - MY.NET.6.52, MY.NET.6.48, MY.NET.6.49, MY.NET.6.50, MY.NET.6.60, MY.NET.6.53, MY.NET.6.45, MY.NET.60.43, MY.NET.88.148, MY.NET.152.171, MY.NET.153.164, MY.NET.152.158, MY.NET.152.174

Detect 5: "Possible Trojan server activity"

- **All source sockets triggering the source port 27374 detect signature**

Alert	Socket	# Detects
source: Possible trojan server activity	213.239.74.221:27374	680
source: Possible trojan server activity	MY.NET.5.44:27374	15
source: Possible trojan server activity	MY.NET.5.88:27374	15
source: Possible trojan server activity	MY.NET.5.77:27374	13
source: Possible trojan server activity	Y.NET.185.28:27374	5
source: Possible trojan server activity	MY.NET.5.19:27374	5
source: Possible trojan server activity	MY.NET.5.46:27374	5
source: Possible trojan server activity	MY.NET.5.78:27374	5
source: Possible trojan server activity	64.12.96.7:27374	5
source: Possible trojan server activity	MY.NET.70.177:27374	4
source: Possible trojan server activity	62.30.220.235:27374	2
source: Possible trojan server activity	24.244.128.66:27374	1
source: Possible trojan server activity	MY.NET.5.42:27374	1

- **Top 5 source IP's triggering detect signature**

Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #1	1163 alerts	**MY.NET.88.162**	1 signatures	213.239.74.221
rank #2	680 alerts	**213.239.74.221**	1 signatures	MY.NET.88.162
rank #3	58 alerts	**MY.NET.5.83**	1 signatures	(8 destination IPs)
rank #4	15 alerts	**MY.NET.5.44**	1 signatures	MY.NET.5.83
		MY.NET.5.88	1 signatures	MY.NET.5.83
rank #6	13 alerts	**MY.NET.5.77**	1 signatures	MY.NET.5.83

- **All destination sockets receiving the port 27374 detect signature**

Alert	Socket	# Detects
Possible trojan server activity	213.239.74.221:27374	1163
Possible trojan server activity	MY.NET.5.44:27374	13
Possible trojan server activity	MY.NET.5.88:27374	12
Possible trojan server activity	MY.NET.5.77:27374	11
Possible trojan server activity	MY.NET.70.177:27374	6

Possible trojan server activity	MY.NET.185.28:27374	4
Possible trojan server activity	MY.NET.5.19:27374	4
Possible trojan server activity	MY.NET.5.46:27374	4
Possible trojan server activity	MY.NET.5.78:27374	4
Possible trojan server activity	64.12.96.7:27374	3
Possible trojan server activity	212.166.188.33:27374	2
Possible trojan server activity	62.30.220.235:27374	2
destination: Possible trojan server activity	24.244.128.66:27374	1

- **Top 5 destination IP's receiving this detect signature**

rank #1	1163 alerts	**213.239.74.221**	1 signatures	MY.NET.88.162
rank #2	680 alerts	**MY.NET.88.162**	1 signatures	213.239.74.221
rank #3	68 alerts	**MY.NET.5.83**	1 signatures	(9 source IPs)
rank #4	13 alerts	**MY.NET.5.44**	1 signatures	MY.NET.5.83
rank #5	12 alerts	**MY.NET.5.88**	1 signatures	MY.NET.5.83

- **Detect Information**
  - **Cert Advisories:**
    - http://www.cert.org/incident_notes/IN-2001-07.html
    - http://www.whitehats.com/info/IDS279
  - **General Information:** A number of logged packets triggering the Trojan server activity logs have destination port 27374 as the source port. This port is typically associated with the SubSeven 2.1 server daemon, explicitly detailed in (*SANS Institute*). SubSeven 2.1 is a Microsoft Windows Trojan that typically arrives as an email attachment. SubSeven clients can be used to connect to servers allowing the attacker to perform a variety of automated tasks, in addition to allowing full administrator level access to the infected system. (*SANS Institute)*
- **Site Security Policy Modification Recommendations:**
  Users should be warned that University systems must not be used in any illegal activities.
- **Susceptibility to misdiagnosis / Probable False Positive**
  The SubSeven packages include an executable called editserver.exe that allows the attacker to modify server install parameters such as listening ports and passwords. Fortunately, attackers often forget or don't care enough to change the default port, leaving a clear trail of SubSeven activity for the ids systems.
- **Defensive Recommendation**
  - **Suggested IDS Modifications**: It is suggested that University NIDS sensors be updated with the most recent snort filter-sets regarding SubSeven activity:

```
alert TCP $HOME_NET 1243 -> !$HOME_NET any (msg:"TROJAN ACTIVITY-Possible SubSeven";
flags:SA;)
```

alert TCP any any -> any any (msg:"TROJAN ACTIVITY-Possible SubSeven access"; content:"connected. time/date"; flags:PA;)
alert TCP !$HOME_NET any -> $HOME_NET 6776 (msg:"TROJAN ATTEMPT- SubS even access"; flags:S;)b
alert TCP !$HOME_NET any -> $HOME_NET 6711 (msg:"TROJAN ATTEMPT- Deep Throat/SubSeven"; flags:S;)
alert TCP !$HOME_NET any -> $HOME_NET 1243 (msg:"TROJAN ATTEMPT- SubSeven"; flags:S;)

- **Patch Updates**: See (*SANS Institute*) for details on removal of Trojan files.
- **Firewall Parameters:** All externally destined port 27374 traffic should at least be logged, all internally destined port 27374 traffic should be blocked and logged..

- **Correlation**
  - MY.NET.5.83 appears to be either scanning for vulnerable SubSeven server daemons, or communicating with other infected daemons.
  - 213.239.74.221 appears to be infected with a SubSeven server and is being accessed frequently by the system at MY.NET.88.162.
  - MY.NET.5.44, MY.NET.5.88 both received 12 and 15 requests from MY.NET.5.83

Detect 6: "WEB-IIS _vti_inf access"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
151.196.241.39	27	49	1	1
198.200.181.209	18	48	1	1
131.118.250.197	16	30	1	1
68.50.252.86	15	30	1	1
68.50.36.142	13	24	1	1

- **Destination network segments receiving this detect signature**

Alert	Socket	# Detects
WEB-IIS _vti_inf access	MY.NET.5.96:80	385
WEB-IIS _vti_inf access	MY.NET.150.83:80	4

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.5.96	385	1956	124	192
MY.NET.150.83	4	121	3	22

- **Detect Information**
  - **General Information:** The *'_vti_in.f.html'* query suffix string (as in www.microsoft.com/_vti_inf.html) has been known to induce Microsoft FrontPage Servers running the appropriate FrontPage extensions to return both the version of those extensions and their location on the queried server's directory hierarchy [3](*pedward)* TCP streams on port 80 eliciting this signature indicate that the

source address is scanning for reconnaissance info. Upon finding a system returning information to such a request, the attacker can assume that the targeted server is running a Microsoft FrontPage server. A successful access to a *'_vti_inf.html'* query should return:

" FrontPage Configuration Information. In the HTML comments, this page contains configuration information that the FrontPage Explorer and FrontPage Editor need to communicate with the FrontPage server extensions installed on this web server. Do not delete this page."

- **CERT Advisories and Information:** http://www.insecure.org/sploits/Microsoft.frontpage.insecurities.html

- **Site Security Policy Modification Recommendations:**
  Site Security Policy should mandate that publicly accessible web servers be patched and configured to disable common 'out-of-the-box' vulnerabilities prior to serving any external web clients.
- **Susceptibility to misdiagnosis / Probable False Positive**
  It is unlikely that the '_vti_inf.html' string is commonly seen as a filename outside the realm of association with Microsoft FrontPage servers, Any TCP streams matching this pattern are likely trying to collect reconnaissance information.
- **Defensive Recommendation**
  - **Suggested IDS Modifications**: As the current IDS configuration is capable of identifying alerts of this nature already, no updates to site IDS filters are recommended.
  - **Patch Updates**: FrontPage servers should be disabled if possible, if this is not possible, care should be taken to disable default/out-of-the-box vulnerabilities where possible.
  - **Firewall Parameters :** It is suggested that sites responsible for any such traffic be blocked at the external site firewall
- **Correlation**
  The host 'MY.NET.5.96' received 385 requests for '_vti_inf.html' over the monitored time period, it is likely that this machine is returning info that has aroused the curiosity of would-be attackers.
  - No correlations were found at dshield.org for any of the external source addresses

Detect 7: "Queso fingerprint"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
212.83.73.254	2	2	2	2
217.235.144.33	2	4	1	1
80.144.189.160	1	1	1	1
217.1.76.143	1	1	1	1
212.76.43.171	1	1	1	1

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.153.178	5	1121	4	926
MY.NET.153.45	2	1643	2	1369
MY.NET.152.21	1	304	1	167
MY.NET.153.175	1	2035	1	1677
MY.NET.150.220	1	167	1	15

- **Detect Information**
  - **Cert Advisories:**
    - http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids29&view=event (*pedward)* [3]
    - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0454 (*Voemel)* [4]
  - **Protocol Information:**
    - RFC 793: http://www.faqs.org/rfcs/rfc793.html
  - **General Information:** The Queso program is one among the family of security tools including Nmap that can be used to perform remote operating system fingerprinting. Such reconnaissance is valuable to potential attackers as it provides valuable information about what default vulnerabilities may be available for exploitation purposes on the remote target. An example of Queso output is as follows:

    queso]# queso localhost
    127.0.0.1:80 * Standard: Solaris 2.x, Linux 2.1.???, MacOS

    The packets crafted by the queso app are TCP and have the CWR, ECN, and SYN bits set. By quantifying the reaction to such an anomalous packet based upon different implementations of the TCP/IP protocol stack, an attacker can predict the OS type based upon its reaction to the anomalous packets.
- **Site Security Policy Modification Recommendations:**
  Site policy should prohibit use of QOS services prior to notification of site security personnel so that details of usage can be communicated and compensated for in site security systems.
- **Susceptibility to misdiagnosis / Probable False Positive**
  The queso program is known to craft packets using the ECN and CWR bit in the TCP header flags. IDS filters can mistakenly pick up legitimate traffic making use of these bits for QOS (quality of service) benefits and attribute them to OS scanning attempts.
- **Defensive Recommendation**
  - **Suggested IDS Modifications**: It is suggested that all traffic not explicitly prearranged with site security personnel for the purpose of QOS routing be alerted upon and investigated by site security personnel.
  - **Patch Updates**: The attacker hopes to assess the target OS in an attempt to identify the best types of default operating system attacks for use against the victim by use of operating system fingerprinting methods. Such default vulnerabilities should be eliminated via internal vulnerability scanning by authorized personnel.
  - **Firewall Parameters:** It is suggested that traffic originating externally to the university network with the ECN and CWR TCP Flag bits set should be logged,

while filtering completely such data, legitimate traffic may be impacted, however such usage is unusual in nature and should be viewed with some suspicion. Additionally, the attackers: 212.83.73.254, 217.235.144.33, 80.144.189.160, 217.1.76.143, and 212.76.43.171 should be blacklisted (all incoming traffic from these IP addresses should be dropped).

- **Correlation**
  - No correlations were found at dshield.org for any of the external source addresses

Detect 8: "FTP DoS ftpd globbing"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
164.76.180.135	408	408	1	1
80.13.221.96	276	276	1	1
164.76.174.203	182	182	1	1
132.235.160.97	166	166	1	1
206.25.183.44	145	145	1	1

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.153.191	1560	1688	16	121
MY.NET.153.174	462	481	3	11
MY.NET.150.46	259	339	3	10
MY.NET.153.204	169	194	2	10
MY.NET.153.194	146	204	4	13

- **Detect Information**
  - **Related Cert Advisories:**
    - http://www.cert.org/advisories/CA-2000-13.html
    - ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02
    - http://www.w3.org/Protocols/rfc959/Overview.html
    - http://packetstormsecurity.nl/UNIX/scanners/indexdate.shtml
    - http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids487&view=event
  - **Protocol Information:**
    - http://www.w3.org/Protocols/rfc959/Overview.html
  - **General Information:** FTP wildcard globbing consists of sending excessively long wildcard strings, such as */../*/../*../*/.. …. to a vulnerable FTP client. Older implementations of various FTPD servers based off the original BSD ftp daemon are known to have difficulty dealing with long wildcard strings.
- **Site Security Policy Modification Recommendations:**
  Site policy should prohibit enabling FTP daemons prior to notification to site security personnel, so that it can be ensured that up-to-date ftp daemons can be deployed.
- **Defensive Recommendation**

- **Suggested IDS Modifications**: Site IDS filter should be verified up to date and consistent with the whitehats.com suggested filter:

  alert TCP $EXTERNAL any -> $INTERNAL 21 (msg: "IDS487/ftp_dos-ftpd-globbing"; flags: A+; content: "|2f2a|"; classtype: denialofservice; reference: arachnids,487;)

- **Patch Updates**: WU-FTPD daemons older than the 2.6.0 version are known to have issues with long wildcard strings, a recent copy of WU-FTPD can be downloaded from http://www.wu-ftpd.org .
- **Firewall Parameters:** The hosts 164.76.180.135, 80.13.221.96, 164.76.174.203, 132.235.160.97, 206.25.183.44 should be blocked at the external firewall such that traffic from them cannot enter the University network environment.

- **Correlation**
  - No correlations were found at dshield.org for any of the external source addresses

Detect 9: "spp_http_decode"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
MY.NET.153.197	259	3617	1	97
MY.NET.153.171	201	612	1	45
MY.NET.153.184	76	256	1	25
MY.NET.153.196	37	247	1	114
MY.NET.153.125	19	215	2	20

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
209.10.239.135	536	536	3	3
209.143.193.70	37	37	1	1
66.150.100.30	17	17	1	1
MY.NET.5.96	11	1956	1	192
205.188.180.25	7	18	3	7

- **Detect Information**
  - **Related Cert Advisories:**
    - http://xforce.iss.net/alerts/advise68.php
    - http://www.kb.cert.org/vuls/id/111677
  - **General Information:** spp_http_decode is among the family of attacks that involve sending Unicode to vulnerable Microsoft IIS 4 and 5 web servers. The vulnerability occurs due to the logic used in the IIS parser. IIS was programmed to delay Unicode decode until after path checking logic is completed. This allows an attacker to specify a path with directory traversal while avoiding the directory traversal detection mechanism [3](*Penward*). While Directory traversals are not

compromising in and of themselves, they represent an attempt to access unauthorized access and should raise the suspicion of site security personnel.

- **Site Security Policy Modification Recommendations:** No security policy modifications are suggested in light of this detect
- **Defensive Recommendation**
    - **Suggested IDS Modifications**: Use of the http_decode snort plugin is suggested if not currently used by University NIDS sensors.
    - **Patch Updates**: Microsoft has released a cumulative path for IIS here: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp
    - **Firewall Parameters:** No updates to site firewalls are suggested.
    - **HIDS:** An HTTP server host based intrusion detection system such as Tripwire would serve to augment the security of vulnerable IIS systems
- **Correlation**
    - No correlations were found at dshield.org for any of the external source addresses

Detect 10: "EXPLOIT NTPDX buffer overflow"

- **Top 5 source IP's triggering detect signature**

Source IP	# Alerts (sig)	# Alerts (total)	# Dsts (sig)	# Dsts (total)
64.232.138.141	3	14	1	1
63.250.205.44	2	14	1	3
63.250.205.9	1	10	1	1
66.28.225.156	1	8	1	2
66.38.171.141	1	19	1	1

- **Top 5 destination IP's receiving this detect signature**

Destinations IP's	# Alerts (sig)	# Alerts (total)	# Srcs (sig)	# Srcs (total)
MY.NET.151.125	3	17	1	3
MY.NET.153.153	2	271	1	32
MY.NET.150.120	1	21	1	7
MY.NET.150.215	1	23	1	3
MY.NET.153.185	1	100	1	27

- **Detect Information**
    - **Related Cert Advisories:**
        - http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids492&view=research
        - http://www.kb.cert.org/vuls/id/970472
    - **Protocol Information:**
        - http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1305.html
    - **General Information:** The ntpdx buffer overflow arrives in the form of UDP packet, this makes defense difficult to the extent that by the connectionless nature of UDP, malicious packets may be easily spoofed with the source address

of a legitimate ntpd source. Secondly, the buffer overflow may arrive in a single packet. The recipients of these packets during the monitored time should be audited for compromise.

- **Site Security Policy Modification Recommendations:** No security policy modifications are suggested in light of this detect
- **Defensive Recommendation**
  - **Suggested IDS Modifications**: Because of the spoof ability of the source address and the anonymity associated with the UDP protocol, merely monitoring for traffic originating from non-accepted ntpd servers is insufficient. Fortunately, only rare legitimate implementations of the protocol exceed a packet length of 120 bytes. As a result, the snort filter set should be verified to monitor for ntp traffic exclusive to the allowable set of ntp servers for the site, in addition to traffic exceeded 120 bytes in length that seems to originate from the allowed ntp servers, from www.whitehats.com, an example:

```
alert UDP $EXTERNAL any -> $INTERNAL 123 (msg: "IDS492/misc_ntpdx-buffer-
overflow"; dsize: >128; classtype: system-attempt; reference: arachnids,492;)
```

  - **Patch Updates**: Bugtraq has released a series of links to the most recent versions of the ntp implementations for most common platforms at : http://online.securityfocus.com/bid/2540/solution/
  - **Firewall Parameters:** It is suggested that intersite ntp traffic be limited to a small number of high stratum servers, where packet lengths greater than 120 bytes be filtered and logged. Additionally, all traffic from the sources: 64.232.138, 63.250.205, 66.38.171, 66.28.225, should be blocked
- **Correlation**
  - No correlations were found at dshield.org for any of the external source addresses

Top Ten Filtered Sources/Destinations

The top ten source/destinations were analyzed performed by utilized a filtered set of detects from the alert detect set as input into SnortSnarf. Of the excluded detects were:

- Connection to 515 from Inside (this was considered to be in large part regular site printing activity)
- SMB Name Wildcards (Considered to be related to legitimate Windows browsing traffic)
- SNMP Public Access (Considered to be network/system/platform analysis traffic)
- Misc UDP Large (Considered to be too broad of a justification of data to infer specific malicious intents)
- ICMP Echo Request L3retriever Ping(Considered to be network hardware analysis traffic)
- ICMP Router Selection traffic to multicast address 224.0.0.2 (Considered to be part of regular router interactions)
- MSN IM Data (assumed to be non-malicious chat traffic)

Rank	Total # Alerts	Dest IP	# Signatures triggered	Originating sources
rank #1	3232 alerts	MY.NET.11.6	1 signatures • 1 instances of ICMP Echo Request Nmap or HPING2	(54 source IPs)

rank #2	2209 alerts	MY.NET.11.7	1 signatures • 1 instances of ICMP Echo Request Nmap or HPING2	(50 source IPs)
rank #3	2022 alerts	MY.NET.153.175	10 signatures • 1 instances of *SCAN Synscan Portscan ID 19104* • 1 instances of Queso fingerprint • 1 instances of WEB-MISC compaq nsight directory traversal1 instances of EXPLOIT x86 setgid 0 • 1 instances of Attempted Sun RPC high port access • 2 instances of ICMP Echo Request Nmap or HPING2 • 3 instances of INFO - Possible Squid Scan4 instances of SCAN Proxy attempt • 20 instances of High port 65535 UDP - possible Red Worm traffic • 1988 instances of INFO Inbound GNUTella Connect request	(1669 source IPs)

rank #4	1943 alerts	MY.NET.5.96	20 signatures • 1 instances of SCAN Synscan Portscan ID 19104 • 1 instances of WEB-MISC webdav search access • 1 instances of NMAP TCP ping! • 1 instances of WEB-MISC whisker head • 1 instances of WEB-IIS encoding access • 1 instances of ICMP Echo Request Nmap or HPING2 • 1 instances of EXPLOIT x86 NOOP • 1 instances of WEB-MISC prefix-get // • 2 instances of INFO - Possible Squid Scan • 5 instances of Possible trojan server activity • 5 instances of SCAN Proxy attempt • 8 instances of WEB-MISC http directory traversal • 11 instances of spp_http_decode: CGI Null Byte attack detected • 19 instances of Port 55850 TCP - Possible myserver activity - ref. 010313-1 • 31 instances of spp_http_decode: IIS Unicode attack detected • 51 instances of WEB-CGI scriptalias access • 59 instances of WEB-MISC Attempt to execute cmd • 361 instances of WEB-FRONTPAGE _vti_rpc access • 384 instances of WEB-IIS _vti_inf access • 999 instances of WEB-IIS view source via translate header	(192 source IPs)

rank #5	1680 alerts	MY.NET.153.191	10 signatures • 1 instances of EXPLOIT x86 setgid 0 • 1 instances of Queso fingerprint • 1 instances of Null scan!1 instances of EXPLOIT x86 setuid 0 • 3 instances of INFO - Possible Squid Scan • 4 instances of Possible trojan server activity • 5 instances of SCAN Proxy attempt • 7 instances of SCAN Synscan Portscan ID 19104 • 104 instances of INFO Inbound GNUTella Connect request • 1553 instances of FTP DoS ftpd globbing	(120 source IPs)
rank #6	1630 alerts	MY.NET.153.45	7 signatures • 2 instances of Queso fingerprint • 2 instances of NMAP TCP ping! • 2 instances of ICMP Echo Request Nmap or HPING2 • 2 instances of INFO - Possible Squid Scan • 3 instances of INFO MSN IM Chat data • 4 instances of SCAN Proxy attempt • 1615 instances of INFO Inbound GNUTella Connect request	(1358 source IPs)
rank #7	1181 alerts	MY.NET.153.111	4 signatures • 2 instances of ICMP Echo Request Nmap or HPING2 • 2 instances of SCAN Proxy attempt • 3 instances of INFO - Possible Squid Scan • 1174 instances of INFO MSN IM Chat data	(7 source IPs)
rank #8	1159 alerts	213.239.74.221	1 signatures • Possible trojan server activity	MY.NET.88.162
rank #9	1115 alerts	MY.NET.153.178	5 signatures • 1 instances of INFO MSN IM Chat data • 2 instances of Null scan! • 3 instances of NMAP TCP ping! • 5 instances of Queso fingerprint • 1104 instances of INFO Inbound GNUTella Connect request	(922 source IPs)
rank #10	999 alerts	193.253.202.216	3 signatures • 11 instances of FTP CWD / - possible warez site • 28 instances of INFO FTP anonymous FTP • 960 instances of SCAN Proxy attempt	(383 destination IPs)

Rank	**Total # Alerts**	**Source IP**	**# Signatures triggered**	**Destinations involved**

rank #1	3603 alerts	MY.NET.153.197	6 signatures • 1 instances of High port 65535 UDP - possible Red Worm - traffic • 12 instances of INFO Possible IRC Access • 37 instances of INFO MSN IM Chat data • 258 instances of spp_http_decode: CGI Null Byte attack detected • 958 instances of ICMP Fragment Reassembly Time Exceeded • 2337 instances of spp_http_decode: IIS Unicode attack detected	(97 destination IPs)
rank #2	2401 alerts	212.179.35.118	1 signatures • 2401 instances of Watchlist 000220 IL-ISDNNET-990517	(4 destination IPs)
rank #3	1297 alerts	MY.NET.88.162	4 signatures • 8 instances of ICMP Fragment Reassembly Time Exceeded • 59 instances of Port 55850 TCP - Possible myserver activity - ref. 010313-1 • 71 instances of spp_http_decode: IIS Unicode attack detected • 1159 instances of Possible trojan server activity	(4 destination IPs)
rank #4	1243 alerts	61.132.208.63	2 signatures • 559 instances of SCAN Proxy attempt • 684 instances of INFO - Possible Squid Scan	(311 destination IPs)
rank #5	1152 alerts	MY.NET.6.52	3 signatures • 1 instances of ICMP Fragment Reassembly Time Exceeded • 2 instances of Back Orifice • 1149 instances of High port 65535 UDP - possible Red Worm - traffic	(83 destination IPs)
rank #6	1151 alerts	MY.NET.153.177	4 signatures • 2 instances of High port 65535 UDP - possible Red Worm - traffic • 16 instances of ICMP Fragment Reassembly Time Exceeded • 277 instances of INFO MSN IM Chat data • 856 instances of spp_http_decode: IIS Unicode attack detected	(79 destination IPs)
rank #7	1085 alerts	MY.NET.153.127	4 signatures • 5 instances of ICMP Router Selection • 96 instances of ICMP Fragment Reassembly Time Exceeded • 241 instances of INFO MSN IM Chat data • 743 instances of spp_http_decode: IIS Unicode attack detected	(66 destination IPs)
rank #8	1074 alerts	80.13.214.233	2 signatures • 431 instances of spp_http_decode: IIS Unicode attack detected • 643 instances of WEB-MISC Attempt to execute cmd	(28 destination IPs)

rank #9	1030 alerts	MY.NET.153.111	3 signatures • 3 instances of ICMP Router Selection • 362 instances of spp_http_decode: IIS Unicode attack detected • 665 instances of INFO MSN IM Chat data	(55 destination IPs)
rank #10	999 alerts	193.253.202.216	3 signatures • 11 instances of FTP CWD / - possible warez site • 28 instances of INFO FTP anonymous FTP • 960 instances of SCAN Proxy attempt	(383 destination IPs)

Possible Network Problem: ICMP Fragment Reassembly Time Exceeded

- **Description:**

  The following detect suggests that firewall issues may have existed during the duration of the incident. Typically ICMP Fragment Reassembly errors suggest a packet filter device exists between the source and destination, and is making decisions based upon TCP protocol flag specifications. For example, the following detect might have been caused by a University site firewall which was mis-configured in such a way that it allowed incoming IP fragments with no TCP header protocol bits set, because of the non-stateful nature of the firewall. It is likely that the logic followed by the firewall was to determine whether to drop based upon only the fragment containing the TCP header, while other fragments were passed along unaltered. This leaves an attacker an excellent way of network mapping for the existence of systems behind stateless firewalled sited.
- **Solution:**

  Site Security personnel should audit the firewall modification logs (They should start archiving them, if not already) to determine whether anyone was making modifications between 3/27 10:55 and 3/27: 11.32. Additionally, the University should look at migrating to more sophisticated firewall solutions that include stateful analysis.

```
03/27-10:55:54.993201 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:55.992730 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:55.992805 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
…
03/27-11:32:56.145280 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:57.163107 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:57.163306 [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
```

Summary of Suspicious Remote Hosts

Possibly Hostile Host: 212.179.35.118

- Summary:

  This node appears to be participating in some sort of port 80 related attack, it may be relying on poorly configured site firewalls to allow incoming port 80 traffic.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved

rank #2	2401 alerts	212.179.35.118	1 signature • 2401 instances of Watchlist 000220 IL-ISDNNET-990517	(4 destination IPs)

Alert Traffic suggesting Malicious intent from 212.179.35.118

```
03/21-16:06:44.684869  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.202:1420
03/21-16:06:44.692933  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.202:1420
03/21-16:06:44.694160  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.202:1420
….
03/21-16:09:51.816749  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:1214 -> MY.NET.153.202:1647
03/21-16:09:51.984698  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:1214 -> MY.NET.153.202:1647
03/21-16:09:51.985678  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:1214 -> MY.NET.153.202:1647
03/21-16:09:52.339991  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:1214 -> MY.NET.153.202:1647
03/23-13:59:42.204462  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.143:2561
03/23-13:59:42.205703  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.143:2561
03/23-13:59:42.207115  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.143:2561
…
03/27-12:09:04.449148  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.143:2750
03/27-12:09:04.450513  [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.35.118:80 -> MY.NET.153.143:2750
```

```
KRNIC is not ISP but National Internet Registry similar with APNIC.
Please see the following end-user contacts for IP address information.
IP Address          : 211.169.240.0-211.169.243.255
Network Name        : DACOM-KIDC
Connect ISP Name    : BORANET
Connect Date        : 20000601
Registration Date   : 20000703
[ Technical Contact Information ]
Name                : Taeung kim
Phone               : +82-2-6220-2920
Fax                 : +82-2-6220-2909
E-Mail              : support@kidc.net
```

Possibly Hostile Host: 61.132.208.63

- Summary:

  This host seems to hope that nodes on the MY.NET.0.0/16 network have been mis-configured to allow anonymous web proxy. While this inquiry isn't typically associated with directed attacks against the server, it should be taken seriously as it may be a covert attempt at network mapping. Even if the intent is only anonymous web proxying, such services are a legal liability and is an internet community responsibility to resolve. Site policy should prohibit non-local anonymous web proxy.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #4	1243 alerts	61.132.208.63	2 signatures • 559 instances of SCAN Proxy attempt • 684 instances of INFO - Possible Squid Scan	(311 destination IPs)

Alert Traffic suggesting Malicious intent from this host:

```
03/21-05:51:16.543605  [**] SCAN Proxy attempt [**] 61.132.208.63:1723 -> MY.NET.5.37:8080
03/21-05:51:16.543943  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1725 -> MY.NET.5.37:3128
03/21-05:51:16.544570  [**] SCAN Proxy attempt [**] 61.132.208.63:1726 -> MY.NET.5.38:8080
03/21-05:51:17.204013  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1725 -> MY.NET.5.37:3128
```

```
03/21-05:51:17.206441  [**] SCAN Proxy attempt [**] 61.132.208.63:1723 -> MY.NET.5.37:8080
03/21-05:51:19.073631  [**] SCAN Proxy attempt [**] 61.132.208.63:1863 -> MY.NET.5.83:8080
03/21-05:51:19.077184  [**] SCAN Proxy attempt [**] 61.132.208.63:1851 -> MY.NET.5.79:8080
03/21-05:51:19.077455  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1853 -> MY.NET.5.79:3128
03/21-05:51:19.519887  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1865 -> MY.NET.5.83:3128
03/21-05:51:19.521207  [**] SCAN Proxy attempt [**] 61.132.208.63:1873 -> MY.NET.5.85:8080
03/21-05:51:19.521545  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1875 -> MY.NET.5.85:3128
03/21-05:51:19.669290  [**] SCAN Proxy attempt [**] 61.132.208.63:1863 -> MY.NET.5.83:8080
03/21-05:51:19.687117  [**] SCAN Proxy attempt [**] 61.132.208.63:1851 -> MY.NET.5.79:8080
03/21-05:51:19.687990  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1853 -> MY.NET.5.79:3128
03/21-05:51:20.186421  [**] SCAN Proxy attempt [**] 61.132.208.63:1873 -> MY.NET.5.85:8080
03/21-05:51:20.186711  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1865 -> MY.NET.5.83:3128
03/21-05:51:20.186780  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1875 -> MY.NET.5.85:3128
03/21-05:51:20.299003  [**] SCAN Proxy attempt [**] 61.132.208.63:1863 -> MY.NET.5.83:8080
03/21-05:51:20.390180  [**] SCAN Proxy attempt [**] 61.132.208.63:1851 -> MY.NET.5.79:8080
03/21-05:51:20.390451  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1853 -> MY.NET.5.79:3128
03/21-05:51:20.483419  [**] SCAN Proxy attempt [**] 61.132.208.63:1896 -> MY.NET.5.92:8080
03/21-05:51:20.483624  [**] INFO - Possible Squid Scan [**] 61.132.208.63:1898 -> MY.NET.5.92:3128
```

```
inetnum:     61.132.128.0 - 61.132.255.255
netname:     CHINANET-AH
descr:       CHINANET Anhui province network
descr:       Data Communication Division
descr:       China Telecom
country:     CN
admin-c:     CH93-AP
tech-c:      JW89-AP
mnt-by:      MAINT-CHINANET
mnt-lower:   MAINT-CHINANET-AH
changed:     hostmaster@ns.chinanet.cn.net 20000701
source:      APNIC
person:      Chinanet Hostmaster
phone:       +86-10-66027112
fax-no:      +86-10-66027334
e-mail:      hostmaster@ns.chinanet.cn.net
nic-hdl:     CH93-AP
mnt-by:      MAINT-CHINANET
changed:     shenjun@cndata.com 20020627
source:      APNIC
```

Possibly Hostile Host: 80.13.214.233

- Summary:
  This node seems to be performing IIS Unicode attacks , and commands execute queries in random consecutive order to the university network.. Instead of taking the time to scan with ping probes, the attacker has assumed that no one is listening and is 'scanning' with attack signatures.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #8	1074 alerts	80.13.214.233	2 signatures • 431 instances of spp_http_decode: IIS Unicode attack detected • 643 instances of WEB-MISC Attempt to execute cmd	(28 destination IPs)

Alert Traffic suggesting Malicious intent from

```
…
03/21-14:35:05.329914  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3335 -> MY.NET.150.133:80
03/21-14:35:05.351234  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3338 -> MY.NET.150.147:80
03/21-14:35:05.351234  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3338 -> MY.NET.150.147:80
03/21-14:35:05.377645  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3337 -> MY.NET.150.143:80
03/21-14:35:05.377645  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3337 -> MY.NET.150.143:80
03/21-14:35:05.446469  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3340 -> MY.NET.150.197:80
03/21-14:35:05.446469  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3340 -> MY.NET.150.197:80
03/21-14:35:05.511123  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3342 -> MY.NET.151.114:80
03/21-14:35:05.511123  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3342 -> MY.NET.151.114:80
03/21-14:35:05.531306  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3343 -> MY.NET.150.243:80
03/21-14:35:05.531306  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3343 -> MY.NET.150.243:80
03/21-14:35:05.600803  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3345 -> MY.NET.153.219:80
03/21-14:35:05.600803  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3345 -> MY.NET.153.219:80
03/21-14:35:05.642138  [**] spp_http_decode: IIS Unicode attack detected [**] 80.13.214.233:3347 -> MY.NET.153.220:80
03/21-14:35:05.642138  [**] WEB-MISC Attempt to execute cmd [**] 80.13.214.233:3347 -> MY.NET.153.220:80
…
```

```
inetnum:     80.13.214.0 - 80.13.214.255
netname:      IP2000-ADSL-BAS
descr:       BSRBOR202 Bordeaux Bloc2
country:      FR
admin-c:      WITR1-RIPE
tech-c:      WITR1-RIPE
status:      ASSIGNED PA
remarks:      for hacking, spamming or security problems send mail to
remarks:      postmaster@wanadoo.fr AND abuse@wanadoo.fr
remarks:      for ANY problem send mail to gestionip.ft@francetelecom.com
mnt-by:      FT-BRX
changed:      gestionip.ft@francetelecom.com 20011218
changed:      gestionip.ft@francetelecom.com 20020415
source:      RIPE
route:      80.13.0.0/16
descr:       France Telecom
descr:       Wanadoo Interactive
remarks:      For Hacking, Spamming or Security problems
remarks:      SEND A EMAIL TO abuse@wanadoo.com
```

Possibly Hostile Host: 193.253.202.216

- Summary:
  This node seems to be interested in assessing whether the University provides any anonymous web proxying services, and is 'scanning' the MY.NET.150.0/24 network with ftp anonymous logins and ftp cwd queries.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #10	999 alerts	193.253.202.216	3 signatures • 11 instances of FTP CWD / - possible warez site • 28 instances of INFO FTP anonymous FTP • 960 instances of SCAN Proxy attempt	(383 destination IPs)

Alert Traffic suggesting Malicious intent from

```
03/21-13:41:13.546675  [**] SCAN Proxy attempt [**] 193.253.202.216:4460 -> MY.NET.153.154:1080
```

```
03/21-13:41:13.555960  [**] SCAN Proxy attempt [**] 193.253.202.216:4458 -> MY.NET.153.144:1080
03/21-13:41:13.797030  [**] SCAN Proxy attempt [**] 193.253.202.216:4703 -> MY.NET.153.207:1080
03/21-13:41:13.803241  [**] SCAN Proxy attempt [**] 193.253.202.216:4713 -> MY.NET.153.210:1080
03/21-13:41:14.099051  [**] SCAN Proxy attempt [**] 193.253.202.216:4691 -> MY.NET.153.220:1080
03/21-13:41:14.608775  [**] SCAN Proxy attempt [**] 193.253.202.216:4703 -> MY.NET.153.207:1080
03/21-13:41:14.616877  [**] SCAN Proxy attempt [**] 193.253.202.216:4707 -> MY.NET.153.209:1080
03/21-13:41:16.328910  [**] SCAN Proxy attempt [**] 193.253.202.216:4590 -> MY.NET.153.195:1080

03/21-17:13:11.917986  [**] INFO FTP anonymous FTP [**] 193.253.202.216:3508 -> MY.NET.150.231:21
03/21-17:13:12.855351  [**] FTP CWD / - possible warez site [**] 193.253.202.216:3508 -> MY.NET.150.231:21
03/21-17:13:13.824002  [**] FTP CWD / - possible warez site [**] 193.253.202.216:3511 -> MY.NET.151.114:21
03/21-17:13:21.850636  [**] INFO FTP anonymous FTP [**] 193.253.202.216:3588 -> MY.NET.5.95:21
03/21-17:13:50.839476  [**] INFO FTP anonymous FTP [**] 193.253.202.216:3782 -> MY.NET.150.220:21
03/21-17:13:53.161819  [**] FTP CWD / - possible warez site [**] 193.253.202.216:3796 -> MY.NET.150.243:21
03/21-17:14:25.717279  [**] FTP CWD / - possible warez site [**] 193.253.202.216:3966 -> MY.NET.88.187:21
03/21-17:14:29.175285  [**] FTP CWD / - possible warez site [**] 193.253.202.216:3991 -> MY.NET.150.197:21
03/21-17:14:30.405477  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4003 -> MY.NET.150.84:21
03/21-17:14:32.564991  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4009 -> MY.NET.150.195:21
03/21-17:14:34.475388  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4020 -> MY.NET.150.147:21
03/21-17:14:35.417609  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4020 -> MY.NET.150.147:21
03/21-17:14:37.673376  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4043 -> MY.NET.150.101:21
03/21-17:14:38.650462  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4043 -> MY.NET.150.101:21
03/21-17:14:38.866324  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4048 -> MY.NET.150.41:21
03/21-17:14:52.171903  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4120 -> MY.NET.150.83:21
03/21-17:14:53.023309  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4120 -> MY.NET.150.83:21
03/21-17:15:00.416841  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4136 -> MY.NET.150.16:21
03/21-17:15:03.964591  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4162 -> MY.NET.153.220:21
03/21-17:15:04.814705  [**] FTP CWD / - possible warez site [**] 193.253.202.216:4162 -> MY.NET.153.220:21
03/21-17:15:16.048652  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4229 -> MY.NET.153.219:21
03/21-17:15:21.565482  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4291 -> MY.NET.5.79:21
03/21-17:15:56.977077  [**] INFO FTP anonymous FTP [**] 193.253.202.216:4609 -> MY.NET.150.139:21
```

```
inetnum:     193.253.202.0 - 193.253.202.255
netname:     IP2000-ADSL-BAS
descr:       France Telecom IP2000 ADSL BAS
descr:       BSSGW103 Ste Genevieve Bloc1
country:     FR
changed:     gestionip.ft@francetelecom.fr 20001018
source:      RIPE
role:        Wanadoo Interactive Technical Role
phone:       +33 1 58 88 50 00
e-mail:      abuse@wanadoo.fr
e-mail:      postmaster@wanadoo.fr
admin-c:     FTI-RIPE
tech-c:      TEFS1-RIPE
nic-hdl:     WITR1-RIPE
notify:      gestionip.ft@francetelecom.com
```

Possibly Hostile Host: 211.169.242.108

- Summary:
  This host seems to have had conversations with locally compromised hosts, this suggests that this host may have tried to communicate, or is perhaps under control of the remote node. It is suggested that this IP and subnet block be filtered at the exterior, and interior router with a stateful firewall. Likely what has happened is that the remote machine sent TCP packets exceeding the path MTU, and thereby has suffered from the first packet being dropped from the university's non-stateful firewall.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved

Unranked	577 alerts	211.169.242.108	• ICMP Fragment Reassembly Time Exceeded	(1 destination IPs)

Alert Traffic suggesting Malicious intent from

```
03/27-10:55:54.993201  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:55.992730  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:55.992805  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:58.007929  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:55:59.978357  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:56:01.008833  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:56:01.010142  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-10:56:01.996499  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
…
03/27-11:32:51.153375  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:52.150643  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:52.150716  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:55.155621  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:56.145280  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:57.163107  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
03/27-11:32:57.163306  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.153.197 -> 211.169.242.108
```

```
KRNIC is not ISP but National Internet Registry similar with APNIC.
Please see the following end-user contacts for IP address information.
IP Address        : 211.169.240.0-211.169.243.255
Network Name       : DACOM-KIDC
Connect ISP Name   : BORANET
Connect Date       : 20000601
Registration Date  : 20000703
[ Technical Contact Information ]
Name               : Taeung kim
Phone              : +82-2-6220-2920
Fax              : +82-2-6220-2909
E-Mail            : support@kidc.net
Phone              : +82-2-6220-0101
Fax              : +82-6220-3489
E-Mail            : security@bora.net
```

Possibly Hostile Host: 200.53.87.9

- Summary::
  This host seems to have had conversations with locally compromised hosts, this suggests that this host may have tried to communicate, or is perhaps under control of the remote node. It is suggested that this IP and subnet block be filtered at the exterior, and interior router with a stateful firewall. Likely what has happened is that the remote machine sent TCP packets exceeding the path MTU, and thereby has suffered from the first packet being dropped from the university's non-stateful firewall.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
Unranked	8 alerts	200.53.87.9	• ICMP Fragment Reassembly Time Exceeded	(1 destination IPs)

Alert Traffic suggesting Malicious intent from

```
03/27-22:21:20.017484  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:21:20.017560  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:21:21.049982  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:21:21.050056  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:21:27.023935  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:21:37.035804  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
```

```
03/27-22:21:56.061703  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
03/27-22:22:34.110551  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.88.162 -> 200.53.87.9
```

```
inetnum:     200.53.64/19
status:      reallocated
owner:       TerraLycos Mexico
ownerid:     MX-TEME1-LACNIC
country:     MX
owner-c:     TMA1-ARIN
source:      ARIN-LACNIC-TRANSITION
inetnum-up:  200.52/15
nic-hdl:     TMA1-ARIN
person:      TerraLycos Mexico Administrator IP
e-mail:      ipmaster@CORP.TERRA.COM.MX
country:     MX
phone:       +52 81 8150-4000
source:      ARIN-LACNIC-TRANSITION
```

Summary of Suspicious Local Hosts

Apparently compromised host: MY.NET.6.52

- Summary: This host appears to be possibly externally controlled by local machine MY.NET.153.193, it has also partaken in Back Orifice discussion with 2 nodes on the MY.NET.152.0/24 subnet. Beyond this, the node appears to be utilized to scan the MY.NET.151.0/24, MY.NET.152.0/24, and MY.NET.153.0/24 subnets with high port UDP traffic.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #5	1152 alerts	MY.NET.6.52	3 signatures • 1 instances of ICMP Fragment Reassembly Time Exceeded to MY.NET.153.193 • 2 instances of Back Orifice to MY.NET.152.21:31337, MY.NET.152.164:31337 • 1149 instances of High port 65535 UDP - possible Red Worm – traffic to MY.NET.152.0/24, MY.NET.153.0/24, MY.NET.149.0/24	(83 destination IPs)

Alert Traffic suggesting compromise from MY.NET.6.52:

The below highport scan shows the attacker mapping multiple MY.NET.0.0/24 class subnets

```
03/21-07:32:10.685985  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.246:65535
03/21-07:56:08.079259  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.169:65280
03/21-08:40:08.042073  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.186:65280
03/21-11:32:17.295190  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.152.15:65535
03/21-13:45:51.537595  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.193:65535
```

Access to port 31337 is a dead giveaway to malicious activity:

```
03/22-08:58:25.996690  [**] Back Orifice [**] MY.NET.6.52:24946 -> MY.NET.152.21:31337
03/27-14:05:30.875254  [**] Back Orifice [**] MY.NET.6.52:29281 -> MY.NET.152.164:31337
```

The below scan suggests that the traffic to MY.NET.152.193 may be large enough to have induced fragmentation, this suggests that meaningful conversation may be taking place between these two systems.

```
…
03/22-14:06:57.038102  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.193:65532
03/22-14:07:18.112997  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.193:65280
03/22-14:07:23.463401  [**] ICMP Fragment Reassembly Time Exceeded [**] MY.NET.6.52 -> MY.NET.153.193
03/22-14:10:03.972427  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:59647 -> MY.NET.153.189:65535
03/22-14:19:19.544782  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.209:65535
03/22-14:19:19.546423  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.52:65535 -> MY.NET.153.209:65535
…
```

Apparently compromised host: MY.NET.6.48

- **Summary:**
  This node appears to have been the source of a 31337 Back Orifice destination port query to another internal node MY.NET.153.207, has participated in suspicious traffic to MY.NET.153.181, MY.NET.153.196 traffic, with source ports of non-ephemeral ports 0 and 57. Besides this, the node appears to have been used to scan internal subnets MY.NET.256.151.0/24, MY.NET.152.0/24, MY.NET.153.0/24

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #12	965 alerts	MY.NET.6.48	3 signatures • 1 instances of Back Orifice to MY.NET.153.207:31337 • 5 instances of Attempted Sun RPC high port access to MY.NET.153.181:32771, MY.NET.153.196:32771 • 959 instances of High port 65535 UDP - possible Red Worm – traffic to MY.NET.151.0/24, MY.NET.152.0/24, MY.NET.153.0/24	(70 destination IPs)

- **Alert Traffic suggesting compromise from MY.NET.6.48:**
  The interesting thing about this scan is that the attacker has gone to the trouble of parallel zing the mapping of different subnets, namely 256,256,151.0/24, MY.NET.152.0/24, MY.NET.153.0/24, also of interest is the slight random delay between scans.

```
…
03/21-00:16:05.670836  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:61695 -> MY.NET.151.191:65535
03/21-07:55:03.153241  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:61695 -> MY.NET.153.172:65535
03/21-09:06:53.543257  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.166:65535
03/21-09:06:54.241961  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:61695 -> MY.NET.153.166:65535
03/21-09:09:17.145878  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.150:65535
03/21-09:10:15.830198  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.150:20712
03/21-09:11:43.712380  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.172:65535
03/21-09:33:52.209263  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.160:65408
03/21-09:33:54.863775  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.160:65535
03/21-09:37:34.362118  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.21:65535
03/21-09:50:43.630246  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.21:65535
03/21-09:51:55.834812  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.21:65535
03/21-09:51:55.874046  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.21:65535
03/21-09:56:02.209870  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65535
```

```
03/21-09:56:03.234706  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65280
03/21-09:56:03.277070  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65535
03/21-09:56:08.119363  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:43263 -> MY.NET.153.171:65535
03/21-09:56:09.301963  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:55551 -> MY.NET.153.171:65535
03/21-09:56:15.538234  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:47359 -> MY.NET.153.171:65535
03/21-09:56:24.746480  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65535
03/21-09:56:24.748974  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65535
03/21-09:57:33.539432  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.171:65535
03/21-09:57:39.300987  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.153.162:65535
03/21-09:58:17.969619  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:41215 -> MY.NET.153.171:65535
03/21-09:59:32.751236  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:53503 -> MY.NET.153.209:65535
03/21-09:59:32.753033  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:53503 -> MY.NET.153.209:65535
03/21-10:05:19.795353  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:45116 -> MY.NET.152.160:65535
03/21-10:05:47.495449  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.176:65535
03/21-10:05:48.145879  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.176:65535
03/21-10:05:48.233410  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.176:65535
03/21-10:05:48.238431  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:65535 -> MY.NET.152.176:65535
03/21-10:10:29.851035  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:64511 -> MY.NET.152.179:65535
03/21-10:10:29.864736  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.48:64511 -> MY.NET.152.179:65535
…
```

Access to port 31337 is a dead giveaway to malicious activity:

```
03/22-14:04:19.441222  [**] Back Orifice [**] MY.NET.6.48:12554 -> MY.NET.153.207:31337
```

Here we have RPC port access originating from an obviously malicious source port 0, and 57 (non-ephemeral)

```
03/21-17:35:13.395725  [**] Attempted Sun RPC high port access [**] MY.NET.6.48:57 -> MY.NET.153.196:32771
03/21-17:35:13.396999  [**] Attempted Sun RPC high port access [**] MY.NET.6.48:57 -> MY.NET.153.196:32771
03/21-17:35:13.398256  [**] Attempted Sun RPC high port access [**] MY.NET.6.48:57 -> MY.NET.153.196:32771
03/21-16:46:04.826790  [**] Attempted Sun RPC high port access [**] MY.NET.6.48:0 -> MY.NET.153.181:32771
```

Apparently compromised host: MY.NET.253.10

- **Summary:**
  This node appears to have been used to perform IIS Unicode attacks on the MY.NET.150.0/24, MY.NET.5.0/24, and MY.NET.88.0/24 subnets, in addition to comprehensively scanning the aforementioned subnets with an NMAP ping probe. Strangely, the attacker did the ping scans as an afterthought to directing attacks at port 80 on apparently random hosts on the subnets.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #14	926 alerts	MY.NET.253.10	2 signatures • 31 instances of spp_http_decode: IIS Unicode attack detected to MY.NET.150/24:80, MY.NET.5.0/24:80, 256,256,88,217:80 • 895 instances of ICMP Echo Request Nmap or HPING2 to MY.NET.150.0/24, MY.NET.88.0/24 (obvious scans)	(420 destination IPs)

- **Alert Traffic suggesting compromise from MY.NET.253.10:**

An HTTP IIS Unicode scan, (note short duration between alerts):

…

```
03/21-09:40:53.490379  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33644 -> MY.NET.150.143:80
03/21-09:40:53.493100  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33645 -> MY.NET.150.220:80
03/21-09:40:53.495497  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33646 -> MY.NET.150.246:80
03/21-09:40:53.496738  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33647 -> MY.NET.150.41:80
03/21-09:40:53.497991  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33648 -> MY.NET.150.59:80
03/21-09:40:53.500157  [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.253.10:33649 -> MY.NET.5.92:80
…
```

An Nmap Ping scan mapping the MY.NET.88.0/24 subnet:

```
…
03/22-15:43:35.288624  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.135
03/22-15:43:35.288761  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.136
03/22-15:43:35.288891  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.137
03/22-15:43:35.289162  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.138
03/22-15:43:35.289362  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.140
03/22-15:43:35.289496  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.142
03/22-15:43:35.289700  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.145
03/22-15:43:35.289832  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.146
03/22-15:43:35.290031  [**] ICMP Echo Request Nmap or HPING2 [**] MY.NET.253.10 -> MY.NET.88.147
```

Apparently compromised host: MY.NET.6.49

- **Summary**

  This node also appears to have been used in mapping activity towards the to MY.NET.150/24:80, MY.NET.5.0/24:80, MY.NET.88.0/24 subnets. Additionally, the node was seen to produce traffic to port 31337 to local machine MY.NET.152.170. RPC packets were also seen to originate from a non-ephemeral port 29, destined to other local subnets.

Top Talker Rank	Total # Alerts	Source IP	# Signatures triggered	Destinations involved
rank #16	858 alerts	MY.NET.6.49	6 signatures • 1 instances of ICMP Fragment Reassembly Time Exceeded to MY.NET.152.163 • 1 instances of Port 55850 UDP - Possible myserver activity - ref. 010313-1 to MY.NET.153 • .164:55850 • 1 instances of Back Orifice to MY.NET.153.189:31337 • 4 instances of Attempted Sun RPC high port access to MY.NET.152.167:32771, MY.NET.152.170:32771, • 851 instances of High port 65535 UDP - possible Red Worm – traffic to MY.NET.150/24:80, MY.NET.5.0/24:80, MY.NET.88.0/24	(83 destination IPs)

- **Alert Traffic suggesting compromise from MY.NET.6.49:**

Note low source port on RPC access, this is very likely malicious:

```
03/21-12:48:46.996533  [**] Attempted Sun RPC high port access [**] MY.NET.6.49:29 -> MY.NET.152.167:32771
```

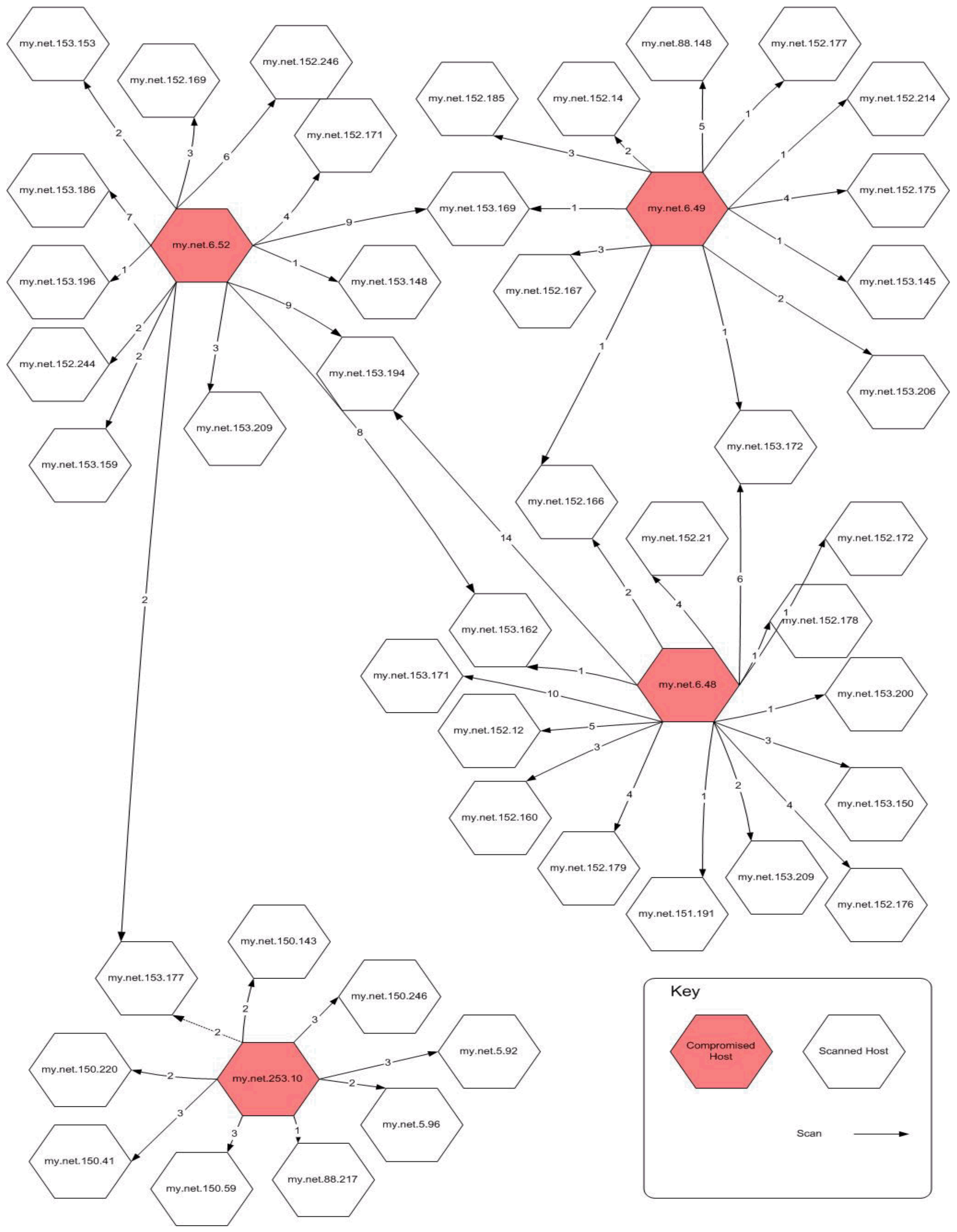
31337 is a dead giveaway:

```
03/21-18:41:03.609013  [**] Back Orifice [**] MY.NET.6.49:28015 -> MY.NET.153.189:31337
```

Red Worm High Port Traffic:

```
…
03/21-19:26:47.142681  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.49:65535 -> MY.NET.153.180:33732
03/21-19:26:52.201853  [**]High port 65535 UDP-possible Red Worm [**] MY.NET.6.49:57599 -> MY.NET.153.180:65535
…
```

Link Graph of Potentially Compromised Hosts



References:

[1] Fearnow, Matt; Stearns, William. "Adore Worm Detection and Removal." URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/tools/adorefind.htm (September 21st, 2002)

[2] SANS Institute. "Intrusion Detection FAQ [Version 1.52] SubSeven Trojan v 1.1."

URL: http://www.sans.org/newlook/resources/IDFAQ/subseven.htm (September 21st, 2002)
[3] pedward. "Many, many, many security holes in the Microsoft FrontPage extensions." 8 October, 2000. URL: http://www.insecure.org/sploits/Microsoft.frontpage.insecurities.html (September 21st, 2002)
[4] Voemel, Christof. "GCIA Practical." 7 September , 2001. URL: http://www.giac.org/practical/Christof_Voemel_GCIA.txt (September 21st, 2002).
[5] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. Reading: Addison Wesley Longman, Inc, 1994.
[6] Ziegler, Scott. Linux Firewalls, Second Edition, Indianapolis: New Riders Publishing, Inc., 2002.

Appendix A: Description of Analysis Processes

The alert log data analysis was supported by a Compaq Proliant DL380 Dual Processor 733 Mhz Pentium 3, with 1024 Mb of Ram and 512 Mb of swap space, running Redhat 7.2 with multiprocessor kernel support. The bash shell was used as the working shell. It should be noted that a filter of port 515 traffic from the dataset would have probably yielded a sufficiently small data set for use on the Celeron.

The first task was to concatenate the alert files together with the use of the UNIX 'cat' command:

```
$  cat alert.02032[1-7] > alert_all
```

In order to ensure that no redundancy occurred in the data set, all portscan logs were removed from the alert_all log file:

```
$ grep –v 'portscan' alert_all > alert_all.tmp
$ mv alert_all.tmp alert_all
```

Another known issue with SnortSnarf is that IP addresses are required for the source and destination fields. The 'MY.NET.x.y' representation of the University network broke this rule, and was fixed with perl:

```
$perl –p –i –e 's/MY\.NET/MY.NET/g' alert_all
```

The alert_all data file was then cleansed of non snort output by utilizing a key signature in all snort alerts, use of the '[**] alert name [**]' syntax. With the use of regular expressions, the following non-snort related data was parsed:

```
$ egrep  '\[\*\*\].*\[\*\*\]' alert_all > alert_all_cleansed
```

Next, an effort was made to ensure that no redundant data existed that would skew sensitive analysis efforts, this was performed by use of the UNIX 'uniq' command:

```
$ sort alert_all |uniq > alert_all_uniq
$ wc -l alert_all_uniq
   369946 alert_all_uniq
$ wc -l alert_all
   426555 alert_all_modified
$ expr 426555 \- 369946
56609
```

The expr command shows that elimination of redundant data has trimmed the log file by 56609 lines.

The log file was then used as an argument to SnortSnarf in the following manner:

```
$time ./snortsnarf.pl -d ../summary18  -homenet MY.NET.0.0/16
```

```
/usr/tmp2/.jba/alert_all_modified_uniq_cleansed
```

Use of the time command was used to appreciate the processing time necessary for dealing with this amount of data

```
egrep  '\[\*\*\].*\[\*\*\]' alert_all_modified_uniq > alert_all_modified_uniq_cleansed
```

Appendix B: Specific Data Analysis for SnortSnarf

Upon investigation of the MISC UDP Large packets, it was found that a sizeable amount of these detects were directed at source port 0, based upon this anomaly, it was decided that a specific analysis would be done to glean additional information on the sources and destinations of this traffic, this procedure was completed as follows:

```
$  grep 'MISC Large UDP Packet' data|grep  ':0$' |head
03/21-08:23:48.334363  [**] MISC Large UDP Packet [**] 207.189.78.231:0 -> MY.NET.152.13:0
03/21-08:23:55.646723  [**] MISC Large UDP Packet [**] 207.189.78.231:0 -> MY.NET.152.13:0
03/21-08:24:00.037044  [**] MISC Large UDP Packet [**] 207.189.78.231:0 -> MY.NET.152.13:0
03/21-08:24:00.849255  [**] MISC Large UDP Packet [**] 207.189.78.231:0 -> MY.NET.152.13:0
03/21-08:25:03.079058  [**] MISC Large UDP Packet [**] 167.216.132.198:0 -> MY.NET.152.13:0
03/21-08:25:03.889423  [**] MISC Large UDP Packet [**] 167.216.132.198:0 -> MY.NET.152.13:0
03/21-08:25:19.079023  [**] MISC Large UDP Packet [**] 167.216.132.198:0 -> MY.NET.152.13:0
03/21-10:13:11.691097  [**] MISC Large UDP Packet [**] 211.233.70.161:0 -> MY.NET.153.184:0
03/21-11:24:13.439329  [**] MISC Large UDP Packet [**] 202.98.15.138:0 -> MY.NET.153.152:0
03/21-11:24:19.769743  [**] MISC Large UDP Packet [**] 202.98.15.138:0 -> MY.NET.153.152:0
$ grep 'MISC Large UDP Packet' data|grep  ':0$' |wc -l
   437
$ grep 'MISC Large UDP Packet' data|grep  ':0$' > udp_large_port_0
$ ./snortsnarf.pl -d ../udp_large_port_0  -homenet MY.NET.0.0/16  udp_large_port_0
```