Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA) Practical Assignment, v3.4
SANS Madrid, September 2003.
Diego González Gómez, 29th February 2004.

Abstract

This document is divided in three main parts or sections, following the GIAC GCIA Practical Assignment version 3.4 specifications.

The first part consists of a white paper about the use of Anomaly Detection for identifying unknown attacks. Advantages and disadvantages are explained, SPADE Anomaly Detector is examined in detail and several implementation examples are suggested.

The second part consists of the exhaustive analysis of three network detects captured in the wild: "WEB-IIS view source via translate header", "NETBIOS DCERPC ISystemActivator bind attempt" and "WEB-IIS WEBDAV nessus safe scan attempt".

Finally, the third part is a security audit for a University. It consists of the comprehensive analysis of five consecutive days worth of data generated by a Snort sensor. The data analyzed corresponds to the period between the 27th to 31st January 2004.

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment, v3.4
(revised September 24, 2003)

SANS Madrid, September 2003

Diego González Gómez

Submitted: February 29, 2004

Table of Contents

Conventions Used in this Paper

Normal text: 12-point Arial.

```
$ Commands: Indented, 10-point Courier New, with a '$' or '#'
```

Log entries: Indented, 8-point Arial

```
Command output or file contents: Indented, 9-point Courier New
```

1. Assignment 1 - The state of Intrusion Detection

1.1. Detecting unknown attacks in the network traffic

1.1.1. Summary

These days the most commonly used network-based intrusion detectors utilize sets of attack patterns to identify attacks or signs of intrusion. Although this solution is safe, simple to implement and gives acceptable results, it is not always enough.

In this article the anomaly detection is emphasized as a possible solution to improve the intrusion detection capabilities. At the moment, anomaly detection is the only technology that can detect new attacks. Among the existing solutions based in anomaly detection, the features of SPADE detector [1] are discussed in detail.

1.1.2. Introduction

The Intrusion Detection Systems (IDS) have been integrated into an essential part of IT defense infrastructures, besides other more common elements such as firewalls or antivirus programs.

The detection methods of ID technologies consist of basically two types: Misuse Intrusion Detection (also known as pattern-based) and Anomaly Intrusion Detection (or behavior-based) [2]. In Misuse Intrusion Detection the hostile system activities are previously defined using signatures or patterns. The Anomaly Detection identifies each significant deviation from normal activity, as an intrusion, and it usually utilizes some type of statistical technique.

Most commercial Intrusion Detection Systems, particularly the network-based (NIDS), are pattern-based. This fact and the lack of good quality anomaly detection tools, makes the users associate the NIDS as 'some kind of antivirus program for network attacks'. However, this is a very limited vision of what is actually Intrusion Detection.

The NIDS that are based in patterns have demonstrated themselves to be very efficient at detecting signs of intrusion. However the nature of these systems makes them easy to avoid detection. The constant appearance of new attacks plus the experience gained from fast-spreading worm attacks such as SQL Slammer [3], makes the use of techniques that allows the ability to isolate and examine behaviors that are not identified by misuse detectors more necessary.

1.1.3. Anomaly Detection

Anomaly Detection is one of the most promising areas of Intrusion Detection. In this type of detection it is assumed that a significant deviation from a normal behavior profile can be caused by an intrusion. This behavior profile is a set of metrics and is created using mathematical algorithms from various data sources. [4]

It is possible to find numerous documents that develop a variety of methods applicable to Anomaly Detection [5]. For example Data Mining techniques, Genetic algorithms, Neural networks, Fuzzy Logic, Biological Immune System mechanisms, Protocol Anomaly Detections, etc.

Anomaly Detection can be used both in Host-based Intrusion Detection Systems (HIDS) as well as in Network-based ones. For example, ImSafe [6] is a HIDS that utilizes Anomaly Detection techniques at the process level to detect possible signs of intrusion.

There are several documents and experimental studies from academic organizations regarding Anomaly Detection in network traffic (mostly TCP/IP). All of them seem to obtain notable results. However, it is not easy to find tools that put them into practice.

1.1.4. Advantages and Disadvantages

Following is a summary of the most common features of Anomaly Detection techniques. Among the advantages of Anomaly Detection we can find:

- **Detection of unknown attacks (such as '0-day' attacks)**. The creation of a behavior profile provides the ability to identify certain activities as anomalies that could be attacks. This certainly reduces the possibility of false negatives[1].
- **Identification of extremely slow scan attempts**. An Anomaly Detector can tag as anomalous, the activity of an intruder that sends one packet per week, month, or even less frequently.
- **Anomaly Detection technologies do not use attack signatures (patterns)**. The Anomaly Detectors learn from their monitored system themselves and adjust their own behavior profiles for detection activities accordingly. This makes unnecessary to update the attack signatures or a knowledge base containing descriptions of hostile behavior based on knowledge of past attacks. Additionally, this allows to detect new attacks.
- **Many applicable theories and techniques**. As discussed before, almost any statistical method or learning technique can be exploited in Anomaly Detection. This feature makes it a very useful Intrusion Detection area.

Naturally, Anomaly Detection has many limitations. If these limitations are taken into account, they can be used to better understand Anomaly Detection and to gain maximum advantage of these technologies:

- **They do not identify what attack or intrusion is happening**. They trigger alarms that indicate anomalous behavior, but is the administrator who must complete a further analysis to determine the origin of the event.
- **They need a reasonable learning period**. Before identifying what is anomalous, an Anomaly Detector must first learn what is normal. That is why Anomaly Detectors usually have a learning period before they begin to work. The more learning time and the more data analyzed, the less probability of errors in the future.
- **Environment dependent**. The behavior profiles generated by statistical detectors are designed for the systems that they monitor. If the target system monitors changes, or if it presents significant changes, the data learned will not be useful and it will be necessary to repeat the learning phase.
- **They consume many resources**. The very first Intrusion Detectors were pattern-based. This was not only because they were easier to implement, but because the Anomaly Detectors needed more time and resources to corroborate their calculations and results. The appearance of faster processors and systems with more resources made possible the development of the first Anomaly Detectors that could be used in normal circumstances.
- **They can be deceived**. The Anomaly Detectors can consider hostile activities as normal. For example, if a SMTP server is constantly receiving scans, or buffer overflow attacks, the detector could eventually consider that these

[1] In Intrusion Detection, a false negative occurs when an alarm is not triggered when there are hostile activities.

activities are normal. Additionally, an Anomaly Detector can be gradually trained by malicious users to make it identify their activities as normal [7].

- **Generation of false positives**. In environments where the monitored activity changes frequently (like network traffic), sometimes it is impossible to create a profile that represents the 'normal' behavior.

1.1.5. SPADE/SPICE

SPADE is an Anomaly Detector for network traffic, licensed under GNU GPL and developed by James Hoagland and Stuard Staniford from Silicon defense. At present it can be installed as a Snort plugin. Nevertheless, its authors affirm that it can be used with almost any IDS.

SPADE is one of the components of SPICE (Stealthy Portscan and Intrusion Correlation Engine). The other component, still in development, is an event correlator. I will list the most important features of this Anomaly Detector. For additional information it is recommended to read the usage file [8].

SPADE utilizes statistical techniques to detect anomalies in the network traffic. It assigns a raw anomaly score A(X) to each packet X received by Snort using the formula: A(X) = -log2 (P(X)). This allows it to give high scores to the less frequent packets. To make reading the results easier, SPADE can generate relative anomaly scores, which are normalized values in the interval between 0 and 1.

In the first versions, SPADE only had one generic anomaly detector. The version v030125.1 has five detector types, with more advanced and specific features. The authors promise the addition of new types in future versions. Following is a list of the features of the available detectors:

- **closed-dport**: This is the traditional detector, used to look for packets destined for closed ports, or ports usually not used. This makes it especially useful for detection of port scans.
- **dead-dest**: This detector looks for unused or destination ip addresses that do not respond. It can be used to detect horizontal scans, where the intruder searches for a fixed service or services in a range or set of IP addresses.
- **odd-dport**: This detector looks for activity originated by machines that try to open connections in non-frequent destination ports. The success of this type of detector is directly proportional to the time that it previously dedicated to monitor its objective. This detector can help to identify compromised hosts.
- **odd-dport-dest**: This detector looks for sources which open connections to uncommon destinations according to the destination ports used. This detector is based on the fact that in some environments, most connections to a certain port or service are made against the same destination IP address. This is common in local networks that have POP, SMTP, DNS servers, etc. A connection attempt that doesn't show this normal type of behavior could be suspicious and could reveal signs of a compromise. This detector is one of the detectors that consumes more resources and that is a reason why it is used with caution.
- **odd-typecode**: This detector has been designed to look for ICMP packets with strange type and code values. This type of detector can always be useful. In addition, there is not a lot of ICMP network therefore this type of detector does not consume too many resources, and it can help to detect, for example, attempts of remote "fingerprinting" made by tools such as Xprobe [9].

Each type of detector has its own set of options, and there are several options common to all of them. These options are discussed in detail in the usage file. On

the other hand, it is possible to run several instances of the same type of detector with different configurations. Needless to say, the more detectors that are running, the more resources SPADE will need.

Each detector in SPADE has a predetermined threshold value that is adjusted regularly, according to the number of generated alarms. This value can change, but if it is very low, too many false positives are generated. On the contrary, a too high value can cause alarms to be missed (false negatives). The threshold value must be assigned considering the degree of activity (number of IP addresses and ports) of the monitored device. It is not the same as monitoring a server with P2P (Peer to Peer) traffic as monitoring an SMTP server. In any case, it is always recommended to leave SPADE working during a long period of time (several days) before allowing it to generate real alarms. Thus, the gathered data will be more accurate.

The messages that SPADE generates are of two types: messages indicating an anomalous activity, and messages indicating an adjustment of the threshold value used. These messages are sent to the Snort log file.

In addition, SPADE has its own log file (by default, "`spade.log`") that is regenerated with each SIGHUP, SIGQUIT, SIGINT and SIGUSR1 or with each Snort exit A set of statistics are appended to this file that can help to adjust its configuration.

On the other hand, SPADE also creates a file called by default "spade.rcv", in which it stores the generated probability tables from the monitored activities. When the detector starts, it looks for this file to recover the state of the network. If it is not available, it creates a new one. In order to avoid false positives or other errors it is necessary to eliminate this file if the network changes or in the case of running various tests.

1.1.6. SPADE Installation

At the time of installing SPADE it is necessary to consider some aspects that generally affect network-based Anomaly Detectors:

- The network traffic frequently has a dynamic nature. In some cases it is practically random. To find useful patterns in such source of data can sometimes be impossible. In order to minimize the effects produced by the appearance of non-relevant information, it is preferable to install the detector as close to the objective as possible. Thus, it will produce fewer errors.
- Most Anomaly Detectors need an initial learning period in which the normal traffic is observed. The data of the learning phase should be as free as possible from hostile activities, since the detector could consider them as normal later. SPADE begins to collect data and to construct the probability tables when it is executed for the first time.

In order to feed an Anomaly Detector with data free from attacks it usually uses specially prepared information, which is not always easy to obtain. In the case of SPADE, there is no need for special data and there is no need for a learning period. It learns at the same time as it works.

A possible solution is to place a Pattern-based Intrusion Detector (for example, Snort) in front of SPADE that discards or redirect dangerous packets. There are several ways to do this, but it requires firewall capabilities to be added to the misuse detector.

It is not recommended to allow the pattern-based detector to discard or to redirect all hostile activity, since some may be false alarms. It could block the most dangerous and well-known attacks. This is the administrator's choice and it depends on the systems they choose to defend. The figure below illustrates the proposed alternative. As you can see, the SPADE detector has been put close to the monitored device. Another way to reduce anomalies consist of applying some type of packet normalization technique in front of SPADE [10].

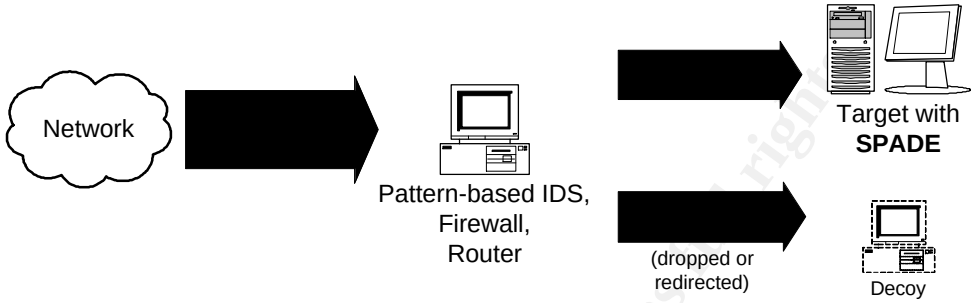

Figure 1 - SPADE instalation example

The more time the Anomaly Detector is in production, the more it will learn about its environment, and the less errors it will commit. However, this does not make Anomaly Detectors invulnerable to certain situations, eg. frequent attacks or frequent scans, that it may eventually consider to be normal. To avoid this, it is recommended to configure the installation as indicated above to reduce as far as possible the appearance of well-known hostile activities.

Once the most suitable location for the detector is known, the only thing required is to follow the instructions indicated in the installation file included in SPADE. In this case the version used was v030125.1. After compiling Snort with the SPADE sources, a SPADE configuration file can be found in the `Snort.conf`. The next step is to edit the configuration.

1.1.7. Configuration examples

Once installed, the SPADE version v030125.1 includes the following two configuration files. The comments have been eliminated for the sake of brevity. In order to make it easier to read, the differences in the second file are in bold.

#spade.conf

```
var SPADEDIR .

preprocessor spade:  dest=alert logfile=$SPADEDIR/spade.log
statefile=$SPADEDIR/spade.rcv

preprocessor spade-homenet: any

preprocessor spade-detect: type=closed-dport tcpflags=synonly wait=3
preprocessor spade-detect: type=closed-dport tcpflags=weird thresh=0.5
preprocessor spade-detect: type=dead-dest tcpflags=weird wait=2
preprocessor spade-detect: type=dead-dest tcpflags=synack wait=2
preprocessor spade-detect: type=dead-dest tcpflags=established wait=5
preprocessor spade-detect: type=dead-dest tcpflags=teardown wait=2
preprocessor spade-detect: type=dead-dest proto=udp wait=2
preprocessor spade-detect: type=dead-dest proto=icmp icmptype=noterr wait=2
preprocessor spade-detect: type=odd-typecode
preprocessor spade-detect: type=odd-typecode to=nothome
```

```
#spade.more.conf
var SPADEDIR .

preprocessor spade:  dest=alert logfile=$SPADEDIR/spade.log
statefile=$SPADEDIR/spade.rcv

preprocessor spade-homenet: any

preprocessor spade-detect: type=closed-dport tcpflags=synonly wait=3
preprocessor spade-detect:  type=closed-dport tcpflags=weird thresh=0.5
preprocessor spade-detect: type=closed-dport tcpflags=teardown
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=synonly
wait=5
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=syn ack
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=teardown
preprocessor spade-detect: type=dead-dest tcpflags=weird wait=2
preprocessor spade-detect: type=dead-dest tcpflags=synack wait=2
preprocessor spade-detect: type=dead-dest tcpflags=established wait=5
preprocessor spade-detect: type=dead-dest tcpflags=teardown wait=2
preprocessor spade-detect: type=dead-dest proto=udp wait=2
preprocessor spade-detect: type=dead-dest proto=icmp icmptype=noterr wait=2
preprocessor spade-detect: type=odd-dport proto=tcp wait=2
preprocessor spade-detect: type=odd-typecode
preprocessor spade-detect: type=odd-typecode to=nothome
```

Below is the contents of file `spade.more.conf`:

```
var SPADEDIR .

preprocessor spade:  dest=alert logfile=$SPADEDIR/spade.log
statefile=$SPADEDIR/spade.rcv

preprocessor spade-homenet: any
```

The variable `$SPADEDIR` indicates the base directory where SPADE will read and store it's logs and probability tables. Then, the option "`dest=alert`" indicates that the SPADE alarms will only be written in the Snort alert file (in case of the need to also send them to the Snort log file it is necessary to indicate "`dest=both`").

In addition the log file (`$SPADEDIR/spade.log`) and the SPADE state file (`$SPADEDIR/spade.rcv`) are specified. Lastly, the local network is specified. As SPADE is executed like a Snort plugin, it is recommended to change this value to the variable $HOME_NET.

The following block corresponds to the lines that specify the detectors' uses. Remember that the more that detectors are used, more resources will be consumed. Consider also that some types of detectors consume more resources than others.

```
preprocessor spade-detect: type=closed-dport tcpflags=synonly wait=3
preprocessor spade-detect: type=closed-dport tcpflags=weird thresh=0.5
preprocessor spade-detect: type=closed-dport tcpflags=teardown
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=synonly
wait=5
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=synack
preprocessor spade-detect: type=closed-dport to=nothome tcpflags=teardown
```

These lines reference the `closed-dport detector`, so they look for closed or non responding ports, and monitor the default protocol TCP. The three first lines

indicate an analysis of the traffic destined to the network indicated in the directive `spade-homenet` (by default) with several combinations of TCP values. In the first line a limit of time delay limit (3 seconds) has been specified, and in the second line a threshold value has been defined (0.5). The last three lines indicate an analysis of the traffic destined to any network different from `spade-homenet` with several combinations of TCP values. The line which matches only the TCP SYN packets also specifies a delay of 5 seconds.

```
preprocessor spade-detect: type=dead-dest tcpflags=weird wait=2
preprocessor spade-detect: type=dead-dest tcpflags=synack wait=2
preprocessor spade-detect: type=dead-dest tcpflags=established wait=5
preprocessor spade-detect: type=dead-dest tcpflags=teardown wait=2
preprocessor spade-detect: type=dead-dest proto=udp wait=2
preprocessor spade-detect: type=dead-dest proto=icmp icmptype=noterr wait=2
```

The above set of directives uses the `dead-dest detector`, so they analyze the network traffic looking for IP destinations that do not exist or do not respond. The first four lines specify several combinations of TCP values. The last two lines indicate protocols UDP and ICMP. The line corresponding to the ICMP specifies values of type ICMP that do not indicate errors.

```
preprocessor spade-detect: type=odd-dport proto=tcp wait=2
```

The above directive specifies the `odd-dport detector`. As explained, this detector looks for sources that initiate connections against unusual destination ports. In this case, a search for TCP ports that take approximately more than 2 seconds to respond.

```
preprocessor spade-detect: type=odd-typecode
preprocessor spade-detect: type=odd-typecode to=nothome
```

Finally, this set of directives references the `odd-typecode` detector. The lines indicate the look for uncommon ICMP values in the traffic destined to `spade-homenet` and to any other network. The configuration file `spade.more.conf` has not enabled by default any reference to the `odd-port-dest` detector. This detector correlates the opened destination ports to the destination addresses, and generates alarms when it considers that they are anomalous. Probably, this type of detector is not activated by default because it consumes more resources, since it has to maintain a 3-dimensional table of data that needs to be periodically maintained. However, the configuration file includes the following lines.

```
#preprocessor spade-detect: type=odd-port-dest proto=tcp Xdports=80
#preprocessor spade-detect: type=odd-port-dest proto=udp Xdports=80
#preprocessor spade-detect: type=odd-port-dest from=nothome proto=tcp
Xdports=80
#preprocessor spade-detect: type=odd-port-dest from=nothome proto=udp
Xdports=80
```

These directives indicate the analysis of TCP and UDP traffic destined to `spade-homenet` and to any other network, excluding the traffic destined for port 80 (usually Web traffic).

After the detectors configuration, SPADE includes the following additional options not enabled by default. They are related to the threshold values and the generation of anomaly scores and traffic statistics.

```
#preprocessor spade-adapt3: id=<label> target=0.01 obsper=60
```

The `spade-adapt` (threshold adapting) directive allows three ways of reporting the threshold value. Above, method number 3 has been used, and it is applied to the detector previously identified by means of the field 'id'.[1]

```
#preprocessor spade-threshadvise: id=<label> target=200 obsper=24
```

Another way to configure the reporting threshold is by using the `spade-threshadvise` (threshold advising) directive. It generates reports in the Snort log with the threshold value that will be required to generate "`target`" alarms every period of "`obsper`" hours. The detector is referenced through its "`id`" value.

```
#preprocessor spade-survey:  id=<label> surveyfile=$SPADEDIR/survey.txt
interval=60
```

The `spade-survey` directive periodically publishes information with the scores of anomalies obtained by each detector according to the indicated time interval. These reports are overwritten when the detector is initiated.

```
#preprocessor spade-stats: entropy uncondprob condpro b
```

The `spade-stats` directive generates statistics about the monitored traffic. This data is added to the SPADE log. These options must be used with caution. The entropy option usually consumes a lot of memory. However, writing the results of `uncondprob` and `condprob` can take a long time. The last options `spade-survey` and `spade-stats` are extremely useful to better understand the SPADE operations and can assist in adjusting their values.

As mentioned before, this version of SPADE supports four different methods to configure the threshold value, and none are activated by default. In previous versions, the value of threshold was more difficult to interpret since relative values were not used (between 0 and 1), but now the relative values are activated by default. However, the `spade-adapt` and `spade-threshadvise` options can always be used as an alternative solution.

1.1.8. Conclusions

The pattern-based Intrusion Detectors have experienced a rapid expansion and important improvements in the last few years. They have represented an important advance in IT security. Nevertheless, these powerful tools have serious limitations that can be partially addressed by other technologies such as Anomaly Detectors.

The security community and the development companies are conscious of the deficiencies of the misuse detectors. Gradually it is becoming more common to find security products that include some type of Anomaly Detection method among

[1] Note that in the example configuration fil es included in SPADE version v030125.1, the detector's lines have not 'id' fields by default.

their features. Note that one of the advantages of Anomaly Detection is the great number of techniques applicable to this technology, which allows the development of many different solutions.

In computer security it is crucial to take advantage of all resources that are available. If Anomaly Detection and Pattern-based Detection methods are combined, it will result in a more complete solution which could identify hostile activities that have previously gone unnoticed.

1.1.9. References

[1] Hoagland, Jim and Stuart Staniford. Silicon Defense. URL: http://www.silicondefense.com/software/spice/ (14 Nov. 2003).

[2] Sundaram, Aurobindo. "An Introduction to Intrusion Detection". 1996. Last updated: January, 2001.URL: http://www.acm.org/crossroads/xrds2-4/intrus.html (18 Nov. 2003).

[3] Moore, David et al. "Inside the Slammer Worm". URL: http://www.computer.org/security/v1n4/j4wea.htm?SMSESSION=NO (18 Nov. 2003).

[4] Debar, Herbe. "What is behavior-based intrusion detection?". Intrusion Detection FAQ. URL: http://www.sans.org/resources/idfaq/behavior_based.php (15 Nov. 2003).

[5] Bace, Rebecca. Intrusion Detection. Indianapolis: Macmillan Technical Publishing, 2000. chpt. 4, p. 100-117.

[6] ImSafe. URL: http://imsafe.sourceforge.net/ (20 Nov. 2003).

[7] Kumar, Sandeep and Spaord, Eugene H. "An Application of Pattern Matching in Intrusion Detection". 17 June 2000. URL: http://www.csee.umbc.edu/cadip/docs/NetworkIntrusion/pattern.pdf (15 Nov. 2003).

[8] Hoagland, Jim. "Usage file for Spade". URL: http://www.silicondefense.com/software/spice/spiceusage.shtml (15 Nov. 2003).

[9] Yarochkin, Fyodor and Ofir Arkin. "Xprobe". URL: http://www.sys-security.com/html/projects/X.html (20 Nov. 2003).

[10] Martin, Ian. GIAC practical. 17 July 2003. URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (24 Nov. 2003).

[11] Farshchi, Jamil. "Statistical-Based Intrusion Detection". Last updated 16 April 2003. URL: http://www.securityfocus.com/infocus/1686 (21 Nov. 2003).

[12] Farshchi, Jamil. "Statistical based approach to Intrusion Detection". Intrusion Detection FAQ. URL: http://www.sans.org/resources/idfaq/statistic_ids.php (21 Nov. 2003).

[13] Liston, Kevin. "Can you explain traffic analysis and anomaly detection?". Intrusion Detection FAQ. URL: http://www.sans.org/resources/idfaq/anomaly_detection.php (19 Nov. 2003).

2. Assignment 2 - Network Detects

The analysis of the three detects in this section were performed using a Pentium Celeron (Mendocino) 450 Mhz with 256 MB running Red Hat Linux 9.0 with kernel 2.6.0 and a Pentium IV 2.0 GHz with 512 MB running Windows XP Professional SP1a.

2.1. Detect #1 - WEB-IIS view source via translate header

This detect consist of the following Snort sensor alerts.

Snort alerts

```
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-11:40:03.736507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
68.36.170.9:33617 -> 32.245.166.119:80 TCP TTL:108 TOS:0x0 ID:28865 IpLen:20 DgmLen:222 DF
***AP*** Seq: 0xAB8BDA9A  Ack: 0xF9540ED4  Win: 0xF5F6  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-11:40:03.776507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x107
68.36.170.9:33617 -> 32.245.166.119:80 TCP TTL:108 TOS:0x0 ID:28870 IpLen:20 DgmLen:249 DF
***AP*** Seq: 0xAB8BDB50  Ack: 0xF9540FBA  Win: 0xFAF0  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-11:40:04.216507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
68.36.170.9:33639 -> 32.245.166.119:80 TCP TTL:108 TOS:0x0 ID:28914 IpLen:20 DgmLen:222 DF
***AP*** Seq: 0xAB95B678  Ack: 0xF9366D6F  Win: 0xF5F6  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-11:40:04.256507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0x107
68.36.170.9:33639 -> 32.245.166.119:80 TCP TTL:108 TOS:0x0 ID:28919 IpLen:20 DgmLen:249 DF
***AP*** Seq: 0xAB95B72E  Ack: 0xF9366E55  Win: 0xFAF0  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:11.196507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEB
213.58.17.245:1411 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11206 IpLen:20 DgmLen:221 DF
***AP*** Seq: 0xB690062A  Ack: 0x69CA4C17  Win: 0x4470  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:12.806507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
213.58.17.245:1411 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11208 IpLen:20 DgmLen:222 DF
***AP*** Seq: 0xB69006DF  Ack: 0x69CA4E7C  Win: 0x420B  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:14.066507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEB
213.58.17.245:1412 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11214 IpLen:20 DgmLen:221 DF
***AP*** Seq: 0xB69B3B2C  Ack: 0x69E06F9E  Win: 0x4470  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:14.936507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
213.58.17.245:1412 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11215 IpLen:20 DgmLen:222 DF
***AP*** Seq: 0xB69B3BE1  Ack: 0x69E07203  Win: 0x420B  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:16.236507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xCA
```

213.58.17.245:1413 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11221 IpLen:20 DgmLen:188 DF
AP Seq: 0xB6A4E831 Ack: 0x6AA9EB56 Win: 0x4470 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:17.326507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xD9
213.58.17.245:1413 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11222 IpLen:20 DgmLen:203 DF
AP Seq: 0xB6A4E8C5 Ack: 0x6AA9EC3D Win: 0x4389 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-16:41:17.856507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xDA
213.58.17.245:1413 -> 32.245.166.119:80 TCP TTL:108 TOS:0xA0 ID:11223 IpLen:20 DgmLen:204 DF
AP Seq: 0xB6A4E968 Ack: 0x6AA9EEA1 Win: 0x4125 TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]

2.1.1. Source of Trace

The packet trace used for this detect was found in file 2002.9.17, downloaded from http://www.incidents.org/logs/raw/ .

All the log files from that directory are provided to be able to complete the GIAC assignment. As stated in http://www.incidents.org/logs/raw/ README, these log files have the following characteristics:

- They have been recorded by an unknown version of Snort IDS with an unspecified set of pre-process filters running in binary logging mode. Therefore, only the packets that violate the unspecified rule set will appear in the log.
- They have been sanitized:
  - All of the local IP addresses have been "munged".
  - The checksums have been modified to prevent discovery of the original addresses.
  - It is possible to find certain keywords within packets replaced with "X"s.
  - There is no Web traffic.

To determine the network layout it is important to keep in mind that these network dump files only contain the packets triggered by Snort sensor. The following methods are similar to the used by Les Gordon [1] or Ian Martin [2].

```
# tcpdump -n -e -r 2002.9.17
```

A brief description of tcpdump parameters used:

-n	don't convert addresses to names.
-e	print the link-level header.
-r	read from file.

This is an example of the tcpdump output explained below:

00:08:00.546507 0:3:e3:d9:26:c0 0:0:c:4:b2:33 0800 74: IP 210.49.49.118.2555 > 32.245.217.136.1080: S 3972551175:3972551175(0) win 16384 <mss 1460,nop,wscale 0,nop,nop,timestamp 53992241 0> (DF)

00:08:00.546507	Time hh:mm:ss.
0:3:e3:d9:26:c0	Source hardware address.
0:0:c:4:b2:33	Destination hardware address.
0800	Protocol type (800 = IP).
74:	Link Layer frame size excluding CRC.
IP	Protocol type.
210.49.49.118.2555	Source network address. Source Port.
>	Direction.
32.245.217.136.1080:	Destination network address. Destination Port.
S	TCP flags (SYN flag).

3972551175:	Beginning TCP sequence number.
3972551175	Ending TCP seq. number (Beginning+Data bytes).
(0)	Data bytes.
win 16384	Window size.
<	TCP options:
mss 1460,	Maximum Segment Size: 0.
nop,	NOP.
wscale 0,	Windows Scale: 0.
nop,	NOP.
nop,	NOP.
timestamp 53992241 0	Timestamp: 53992241 0.
>	
(DF)	IP Flags (Don't fragment).

First, we examine which hardware addresses appear in the network trace.

```
# tcpdump -ner 2002.9.17 | cut -d " " -f 2,3 | sort | uniq -c

   6065 0:0:c:4:b2:33 0:3:e3:d9:26:c0
    508 0:3:e3:d9:26:c0 0:0:c:4:b2:33
```

There are only two different hardware addresses within the tcpdump binary log file. Snort sensor could have been installed in one of that devices or attached to a stealth monitoring device (for example, a network tap).

The first 24 bits of the hardware address correspond to the Organizationally Unique Identifier (OUI), assigned by the IEEE. A simple search at IEEE OUI assignments [3] reveals that the addresses correspond to Cisco devices:

```
00-00-0C   (hex)           CISCO SYSTEMS, INC.
00000C     (base 16)             CISCO SYSTEMS, INC.
                                 170 WEST TASMAN DRIVE
                                 SAN JOSE CA 95134-1706

00-03-E3   (hex)           Cisco Systems, Inc.
0003E3     (base 16)             Cisco Systems, Inc.
                                 170 West Tasman Dr.
                                 San Jose CA 95134
                                 UNITED STATES
```

As both devices are manufactured by Cisco (routers, switches, etc.), The Snort sensor is probably installed on a third stealth monitoring device.

The next step consists of analyzing the network layer and how it is related to both hardware addresses.

Source network addresses coming from 0:0:c:4:b2:33.

```
# tcpdump -ner 2002.9.17 ether src 0:0:c:4:b2:33  \
  | cut -d " " -f 6 | cut -d . -f 1-4 \
  | sort | uniq -c | sort /r

   6056 32.245.166.236
      9 32.245.166.119
```

These two IP addresses come from 0:0:c:4:b2:33 address, and they seem to belong to the same network.

Destination network addresses coming from 0:0:c:4:b2:33.

```
# tcpdump -ner 2002.9.17 ether src 0:0:c:4:b2:33  \
  | cut -d " " -f 8 | cut -d . -f 1-4 \
  | sort | uniq -c | sort /r

   1484     147.208.133.111
    719     64.12.51.118
    522     205.188.214.121
    513     64.12.42.117
[...]
      1     205.188.135.174
      1     199.108.253.20
      1     194.135.30.190
      1     151.164.144.126

# tcpdump -ner 2002.9.17 ether src 0:0:c:4:b2:33  \
  | grep "^32"
```

There are several IP addresses but none of them begins with 32. The next list specifies some source network addresses coming from 0:3:e3:d9:26:c0.

```
# tcpdump -ner 2002.9.17 ether src 0:3:e3:d9:26:c0  \
  | cut -d " " -f 6 | cut -d . -f 1-4 \
  | sort | uniq -c | sort /r

     73     64.125.138.190
     63     210.49.49.118
     57     63.111.48.133
     43     255.255.255.255
[...]
      1     131.107.3.86
      1     12.42.128.70
      1     12.36.134.2
      1     12.111.47.194

# tcpdump -ner 2002.9.17 ether src 0:3:e3:d9:26:c0  \
  | grep "^32"
```

Again, none of the above addresses begins with 32. Below are the destination addresses coming from 0:3:e3:d9:26:c0.

```
# tcpdump -ner 2002.9.17 ether src 0:3:e3:d9:26:c0  \
  | cut -d " " -f 8 | cut -d . -f 1-4 \
  | sort | uniq -c | sort /r

    276     32.245.166.236
     38     32.245.166.119
      6     32.245.98.171
      6     32.245.83.200
[...]
      1     32.245.104.48
      1     32.245.102.200
      1     32.245.102.195
      1     32.245.1.229
```

The network addresses of the last tcpdump command indicates that the network mask of the IP addresses is probably a class B. The next tcpdump filter confirms

that there is not any destination addresses from 0:3:e3:d9:26:c0 from a network different than 32.245.0.0/16.

```
# tcpdump -ner 2002.9.17 'ether src 0:3:e3:d9:26:c0 and \
  not dst net 32.245.0.0/16'
```

In summary:

- Hardware address 0:0:c:4:b2:33 has source traffic only from network 32.245.0.0/16 (just the two IP addresses) and destination traffic to IPs not belonging to network 32.245.0.0/16.
- Hardware address 0:3:e3:d9:26:c0 has source traffic from networks different than 32.245.0.0/16 and destination traffic only to network 32.245.0.0/16.

The information above suggests the network scenario shown below:

```
WAN ------ Cisco device 1 ---+--- Cisco device 2 ------ LAN
          0:3:e3:d9:26:c0    |    0:0:c:4:b2:33
                             |                  32.245.0.0/16
                        Snort sensor
           (TAP, receive-only cable, in-line mode,...)
```

Finally, for determining the quality or even the presence of a firewall we examine the destination ports from Cisco device number 1.

```
# tcpdump -nnr 2002.9.17 ether src 0:3:e3:d9:26:c0 \
| cut -d " " -f 5 | cut -d "." -f 5 | sort | uniq -c | cut -d ":" -f 1

     82 1080
      4 137
      4 139
     43 515
      2 53
      1 61000
      1 61053
      1 61079
[...]
      4 65039
      4 65044
      1 65045
      2 772
     95 80
```

The logs indicate that Cisco device number 1 allows traffic to network 32.245.0.0/16 to many different ports, most of them above of 61000. This is a poorly configured firewall, or more probably a border router. The next tcpdump command shows the source ports coming from Cisco device number 2.

```
# tcpdump -nnr 2002.9.17 ether src 0:0:c:4:b2:33 \
| cut -d " " -f 3 | cut -d "." -f 5 | sort | uniq -c | cut -d ":" -f 1

      1 61009
      1 61010
      2 61011
[...]
      6 65058
      2 65068
```

```
      9 80
```

Almost all source ports from IP address 32.245.166.236 were above 61000 and the only source port from IP address 32.245.166.119 was 80 (see analyzing network layer above). This is not enough to determine if Cisco device number 2 performs network traffic filtering.

2.1.2. Detect was generated by

For this detect I used Snort version 2.1.0 with default sets of rules. The configuration file had EXTERNAL_NET and HOME_NET variables set to 'any' and all rule files enabled. As the binary tcpdump file has only partial network traffic, `stream4` and `stream4_reassemble` pre-processors were disabled [4].

The command used to trigger the alerts was:

```
# Snort -c Snort.conf -e -k none -l log -N -r 2002.9.17-U -y
```

Command parameters in detail:

-c Snort.conf	Use Rules File Snort.conf
-e	Display the second layer header info
-k none	Checksum mode
-l log	Log to directory log
-N	Turn off logging (alerts still work)
-r 2002.9.17	Read and process tcpdump file 2002.9.17
-U	Use UTC for timestamps
-y	Include year in timestamp in the alert and log files

There were 6545 packets in the binary `tcpdump` file, and the Snort command generated 1649 alerts. They are summarized in the following list:

```
# grep "\[\*\*\]" alert | sed "s/\[\*\*\]//g" \
| cut -d "]" -f 2 | sort | uniq -c | sort /r

   1045  (http_inspect) BARE BYTE UNICODE ENCODING
    253  (http_inspect) APACHE WHITESPACE (TAB)
     66  SCAN SOCKS Proxy attempt
     41  SCAN nmap TCP
     38  BACKDOOR Q access
     34  (http_inspect) IIS UNICODE CODEPOINT ENCODING
     27  WEB-FRONTPAGE shtml.exe access
     25  (http_inspect) NON-RFC DEFINED CHAR
     24  SHELLCODE x86 NOOP
     21  (http_inspect) NON-RFC HTTP DELIMITER
     20  (http_inspect) DOUBLE DECODING ATTACK
     11  WEB-IIS view source via translate header
      9  ATTACK-RESPONSES 403 Forbidden
      6  WEB-IIS _vti_inf access
      6  WEB-CGI formmail access
      4  (http_inspect) OVERSIZE CHUNK ENCODING
      3  WEB-ATTACKS id command attempt
      3  SHELLCODE x86 inc ebx NOOP
      3  MISC Tiny Fragments
      2  WEB-IIS ISAPI .ida attempt
      2  SHELLCODE x86 setuid 0
      2  CHAT MSN message
      1  WEB-FRONTPAGE /_vti_bin/ access
      1  WEB-CGI search.cgi access
```

```
      1  WEB-CGI redirect access
      1  BAD-TRAFFIC ip reserved bit set
```

I focused my attention on the 11 "WEB-IIS view source via translate header" alerts. The logs at the beginning of this detect were extracted using the command:

```
grep -A5 "WEB-IIS view source via translate header"
```

Below is one of the Snort alerts.

```
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[Classification: access to a potentially vulnerable web application] [Priority: 2]
10/17/02-11:40:03.736507 0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33 type:0x800 len:0xEC
68.36.170.9:33617 -> 32.245.166.119:80 TCP TTL:108 TOS:0x0 ID:28865 IpLen:20 DgmLen:222 DF
***AP*** Seq: 0xAB8BDA9A  Ack: 0xF9540ED4  Win: 0xF5F6  TcpLen: 20
[Xref => http://www.securityfocus.com/bid/1578][Xref => http://www.whitehats.com/info/IDS305]
```

Alert fields explained.

[**] [1:1042:6] WEB-IIS view source via translate header [**]
: Rule title and brief explanation. Internet Inform ation Server bug.

[Classification: access to a potentially vulnerable web application]
[Priority: 2]
: Type and priority of alert.

10/17/02-11:40:03.736507
: Date and time in UTC (Coordinated Universal Time) format.

0:3:E3:D9:26:C0 -> 0:0:C:4:B2:33
: Source and destination hardware addresses.

type:0x800
: Encapsulated protocol (800 = IP).

len:0xEC
: Packet length without CRC (Cyclic Redundancy Check). Hex. format.

68.36.170.9:33617 -> 32.245.166.119:80
: Source and destination netw ork addresses and ports.

TCP
: TCP packet.

TTL:108
: IP Time to Live.

TOS:0x0
: IP Type of Service.

ID:28865
: IP Identification.

IpLen:20
: Length of IP header in decimal format.

DgmLen:222
: Length of Datagram, with headers and payload. Number of bytes.

DF
: Don't Fragment bit set.

AP
: TCP flags used: A = acknowledge, P = push.

Seq: 0xAB8BDA9A
: TCP sequence numb er.

Ack: 0xF9540ED4
: TCP acknowledgement number.

Win: 0xF5F6
: Window size, in hexadecimal format.

TcpLen: 20
: TCP header length.

[Xref => http://www.securityfocus.com/bid/1578]
[Xref => http://www.whitehats.com/info/IDS305]
: URL reference of the alert.

The alert was triggered by the next rule within web-iis.rules file:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
view source via translate header"; flow:to_server,established; content:
"Translate|3a| F"; nocase; reference:arachnids,305;
reference:bugtraq,1578; classtype:web-application-activity; sid:1042;
rev:6;)
```

Rule explanation:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
: Generate an alert for TCP packets from IP addresses defined by $EXTERNAL_NET variable from any source ports to $HTTP_SERVERS IP addresses to $HTTP_PORTS ports.

msg:"WEB-IIS view source via translate header"
: This is the message posted when the signature is fired.

Flow:to_server,established
: The packets triggered must come from an established session and from client to server direction.

content: "Translate|3a| F"; nocase
: The payload of the packet must have string “Translate: F” without being case-sensitive. Note that ‘:’ is a special character indicated in hexadecimal value between ‘|’.

reference:arachnids,305; reference:bugtraq,1578
: These are the references for additional information about the attack. The url addresses are within reference.config file.

classtype:web-application-activity
: The type of attack and priority can be found in classificacion.config file. In this case it corresponds to: Access to a potentially vulnerable web application,2.

sid:1042
: Snort Signature ID is 1042. Consult Snort Database Signature at http://www.Snort.org/snort-db/sid.html for additional information.

rev:6
: This rule have been revised 6 times.

The following remote IP addresses triggered the alerts.

```
# grep -A5 "WEB-IIS view source via translate header" alert \
| grep TCP | cut -d ":" -f 1 | sort | uniq -c

      7 213.58.17.245
      4 68.36.170.9
```

2.1.3. Probability the source address was spoofed

Probably spoofed?

The probability of a spoofed IP address is very low. The packets of this capture come from an established TCP communication. This attack consist of getting data from the victim, in this case the IIS Web server. Therefore, the attacker needs a response. On the other hand, as showed below, there is no anomalous value in the logged packet headers that could lead us to think that they are spoofed.[1]

```
# tcpdump -vv -nn -r 2002.9.17 'host 213.58.17.245 or host 68.36.170.9'
```

tcpdump parameters used:

```
-vv   be very verbose
```

[1] Note that there are more packets from IP address 68.36.170.9 than listed in the Snort alerts at the beginning of this detect analysis. This is because that particular IP address generated further different alerts. Additional details are available in the 'Evidence of Active Targeting' section.

```
-nn    don't convert addresses or ports to nam es.
-r     read from file.
```

```
12:40:01.916507 IP (tos 0x0, ttl 108, id 28828, len 305) 68.36.170.9.33636 > 32.245.166.119.8 0: P [bad tcp cksum ca0e (-
>267)!] 2877822063:2877822328(265) ack 4177956934 win 64240 (DF)bad cksum d278 (->e790)!
12:40:03.466507 IP (tos 0x0, ttl 108, id 28854, len 430) 68.36.170.9.33617 > 32.245.166.119.80:  P [bad tcp cksum 2601 (-
>7b3c)!] 2878069012:2878069402(390) ack 4183031410 win 64240 (DF)bad cksum  d1e1 (->e6f9)!
12:40:03.736507 IP (tos 0x0, ttl 108, id 28865, len 222) 68.36.170.9.33617 > 32.245.166.119.80: P [bad tcp cksum 9901 (-
>ee3c)!] 390:572(182) ack 4195 win 62966 (DF)bad cksum d2a6 (->e7be)!
12:40:03.776507 IP (tos 0x0, ttl 108, id 28870, len 249) 68.36.170.9.33617 > 32.245.166.119.8 0: P [bad tcp cksum 2b3d (-
>6395)!] 572:781(209) ack 4425 win 64240 (DF)bad cksum d286 (->e79e)!
12:40:03.826507 IP (tos 0x0, ttl 108, id 28875, len 305) 68.36.170.9.33617 > 32.245.166.119.8 0: P [bad tcp cksum 82a8 (-
>bb00)!] 781:1046(265) ack 4655 win 64010 (DF)bad cksum d249 (->e761)!
12:40:04.036507 IP (tos 0x0, ttl 108, id 28902, len 430) 68.36.170.9.33639 > 32.245.166.119.80:  P [bad tcp cksum e985 (-
>3ec1)!] 2878715122:2878715512(390) ack 4181089549 win 64240 (DF)bad cksum  d1b1 (->e6c9)!
12:40:04.216507 IP (tos 0x0, ttl 108, id 28914, len 222) 68.36.170.9.33639 > 32.245.166.119.80: P [bad tcp cksum 5e86 (-
>b3c1)!] 390:572(182) ack 4195 win 62966 (DF)bad cksum d275 (->e78d)!
12:40:04.256507 IP (tos 0x0, ttl 108, id 28919, len 249) 68.36.170.9.33639 > 32.245.166.119.80: P [bad tcp cksum f0c1 (-
>291a)!] 572:781(209) ack 4425 win 64240 (DF)bad cksum d255 (->e76d)!
17:41:11.196507 IP (tos 0xa0, ttl 108, id 11206, len 221) 213.58.17.245.1411 > 32.245.166.119.80: P [bad tcp cksum 7ff0 (-
>b848)!] 3062892074:3062892255(181) ack 1774865431 win 17520 (DF)bad cksum 1e01 (->3319)!
17:41:12.806507 IP (tos 0xa0, ttl 108, id 11208, len 222) 213.58.17.245.1411 > 32.245.166.119.80: P [bad tcp cksum 6f1b (-
>c456)!] 181:363(182) ack 614 win 16907 (DF)bad cksum 1dfe (->3316)!
17:41:14.066507 IP (tos 0xa0, ttl 108, id 11214, len 221) 213.58.17.245.1412 > 32.245.166.119.80: P [bad tcp cksum 2745 (-
>5f9d)!] 3063626540:3063626721(181) ack 1776316318 win 17520 (DF)bad cksum 1df9 (->3311)!
17:41:14.936507 IP (tos 0xa0, ttl 108, id 11215, len 222) 213.58.17.245.1412 > 32.245.166.119.80: P [bad tcp cksum 1670 (-
>6bab)!] 181:363(182) ack 614 win 16907 (DF)bad cksum 1df7 (->330f)!
17:41:16.236507 IP (tos 0xa0, ttl 108, id 11221, len 188) 213.58.17.245.1413 > 32.245.166.119.80: P [bad tcp cksum e92b (-
>3e67)!] 3064260657:3064260805(148) ack 1789520726 win 17520 (DF)bad cksum  1e13 (->332b)!
17:41:17.326507 IP (tos 0xa0, ttl 108, id 11222, len 203) 213.58.17.245.1413 > 32.245.166.119.80: P [bad tcp cksum fdfd (-
>3656)!] 148:311(163) ack 232 win 17289 (DF)bad cksum 1e03 (->331b)!
17:41:17.856507 IP (tos 0xa0, ttl 108, id 11223, len 204) 213.58.17.245.1413 > 32.245.166.119.80: P [bad tcp cksum b86f (-
>dab)!] 311:475(164) ack 844 win 16677 (DF)bad cksum 1e01 (->3319)!
```

Both remote IP addresses are legal and routable in the Internet. Spoofed addresses are typical in other attack scenarios such as Denial of Service or similar attacks, where the attackers want to flood their victims and do not need to complete the three-way handshake.

Probably not spoofed?

I believe that the remote IP addresses are not spoofed. As mentioned before, this kind of attack consists of some type of information gathering. And the packets logged are part of an established TCP communication. There are enough reasons to knowledge that the attacker wants to receive responses to their requests.

3rd party

This situation is not applicable here. The victim is not used to attack, neither gather information from a third party system. Alson the victim's IP was not used either to spoof an attack.

2.1.4. Description of attack

This attack exploits a Microsoft IIS 5.0 scripting engine vulnerability. One malicious user can obtain the source code of ASP, ASA, HTR files and other scripts by sending a special HTTP GET command to the Web Server.

The attacker sends an HTTP GET request with 'Translate: f' in the header, and a trailing backslash '\' at the end of the URL. This command makes the server directly send the source file to the attacker without processing it.

Unfortunately, 'Translate: f' is included in the header of several WebDAV[1] (Web-based Distributed Authoring and Versioning) [5] methods, causing false positives.

Additional information and references can be found at Correlations section.

2.1.5. Attack mechanism

In this attack the intruder sends special commands (a stimulus) to the vulnerable server to obtain information. The packets captured by the Snort sensor correspond to the commands sent by the remote user.

The rule that triggers the alert matches the predefined HTTP service ports. In this case was TCP number was 80.

To determine if the target had an HTTP server running I used tcpdump. The victim has IP address 32.245.166.119. If we are lucky perhaps a response from that server within Snort sensor logs will be found.

```
# tcpdump -s0 -X -nnr 2002.9.17 "src host 32.245.166.119 and src port 80"
```

tcpdump options:

-s0 set snaplen to any (dump complete packets).
-X dump in hexadecimal and ASCII format .
-nn don't convert IP addresses and ports to names.
-r read from file.

```
08:52:19.426507 IP 32.245.166.119.80 > 195. 29.131.59.1055: P
2636510832:2636511368(536) ack 632179 win 32696 (DF)
0x0000  4500 0240 2a29 4000 3f06 ecb1  20f5 a677       E..@*)@.?.....w
0x0010  c31d 833b 0050 041f 9d25 f670  0009 a573       ...;.P...%.p...s
0x0020  5018 7fb8 8337 0000 4854 5450  2f31 2e31       P....7..HTTP/1.1
0x0030  2034 3033 2046 6f72 6269 6464  656e 0d0a       .403.Forbidden..
0x0040  4461 7465 3a20 5468 752c 2031  3720 4f63       Date:.Thu,.17.Oc
0x0050  7420 3230 3032 2031 313a 3432  3a32 3220       t.2002.11:42:22.
0x0060  474d 540d 0a53 6572 7665 723a  2041 7061       GMT..Server:.Apa
0x0070  6368 652f 312e 332e 3132 2028  556e 6978       che/1.3.12.(Unix
0x0080  2920 2028 5265 6420 4861 742f  4c69 6e75       )..(Red.Hat/Linu
0x0090  7829 2046 726f 6e74 5061 6765  2f34 2e30       x).FrontPage/4.0
0x00a0  2e34 2e33 0d0a 4b65 6570 2d41  6c69 7665       .4.3..Keep-Alive
[...]
```

Above is an extract of one of the 9 packets from 32.245.166.119 port 80, showing that the server is Apache version 1.3.12 running under Red Hat Linux. This type of attack only affects Microsoft IIS 5.0, therefore the targeted server is not vulnerable.

Lets examine in detail the contents of the packets that fired the alerts. Following is one of the packets from remote IP number 68.36.170.9.

[1] WebDAV is a set of extensions to the HTTP protocol which allows user groups to edit and manage files on web servers. Some applications use WebDAV for publishing content on a Web server.

```
12:40:03.736507 IP 68.36.170.9.33617 > 32.245.166.119.80: P 390:572(182) ack 419
5 win 62966 (DF)
0x0000   4500 00de 70c1 4000 6c06 d2a6 4424 aa09        E...p.@.l...D$..
0x0010   20f5 a677 8351 0050 ab8b da9a f954 0ed4        ...w.Q.P.....T..
0x0020   5018 f5f6 9901 0000 4f50 5449 4f4e 5320        P.......  OPTIONS .
0x0030   2f20 4854 5450 2f31 2e31 0d0a  5472 616e        /.HTTP/1.1..  Tran
0x0040    736c 6174 653a 2066  0d0a 5573 6572 2d41       slate:.f ..User-A
0x0050   6765 6e74 3a20 4d69 6372 6f73 6f66 7420        gent:.Microsoft.
0x0060   4461 7461 2041 6363 6573 7320 496e 7465        Data.Access.  Inte
0x0070   726e 6574 2050 7562 6c69 7368 696e 6720        rnet.Publishing.
0x0080   5072 6f76 6964 6572 2050 726f 746f 636f        Provider.Protoco
0x0090   6c20 4469 7363 6f76 6572 790d 0a48 6f73        l.Discovery..Hos
0x00a0   743a 2077 7777 2e58 585 8 5858 5858 580d        t:.www.XXXXXXXX.
0x00b0   0a43 6f6e 7465 6e74 2d4c 656e 6774 683a        .Content -Length:
0x00c0   2030 0d0a 436f 6e6e 6563 7469 6f6e 3a20        .0..Connection:.
0x00d0   4b65 6570 2d41 6c69 7665 0d0a 0d0a             Keep -Alive....
```

We can see that the HTTP header concludes with the string 'Translate: f'. But it is an OPTIONS request, not a GET. Now, lets look at one of the packets from 213.58.17.245.

```
17:41:11.196507 IP 213.58.17.245.1411 > 32.245.166.119.80: P 3062892074:3062892  2
55(181) ack 1774865431 win 17520 (DF)
0x0000   45a0 00dd 2bc6 4000 6c06 1e01 d53a 11f5        E...+.@.l....:..
0x0010   20f5 a677 0583 0050 b690 062a 69ca 4c17        ...w...P...*i.L.
0x0020   5018 4470 7ff0 0000 5052 4f50 4649 4e44        P.Dp....  PROPFIND
0x0030   202f 6d61 696e 2048 5454 502f 312e 310d        ./main.HTTP/1.1.
0x0040   0a44 6570 7468 3a20 300d 0a 74 7261 6e73        .Depth:.0..  trans
0x0050    6c61 7465 3a20 66 0d 0a55 7365 722d 4167       late:.f ..User-Ag
0x0060   656e 743a 204d 6963 726f  736f 6674 2d57        ent:.Microsoft -W
0x0070   6562 4441 562d 4d69 6e69 5265 6469 722f        ebDAV-MiniRedir/
0x0080   352e 312e 3236 3030 0d0a 486f 7374 3a20        5.1.2600..Host:.
0x0090   7777 772e 5858 5858 5858 5858 0d0a 436f        www.XXXXXXXX..C o
0x00a0   6e74 656e 742d 4c65 6e67 7468 3a20 300d        ntent -Length:.0.
0x00b0   0a43 6f6e 6e65 6374 696f 6e3a 204b 6565        .Connection:.Kee
0x00c0   702d 416c 6976 650d 0a50 7261 676d 613a        p -Alive..Pragma:
0x00d0   206e 6f2d 6361 6368 650d 0 a0d 0a               .no -cache....
```

Again, the HTTP header terminates with 'translate: f' and it is not a GET request. It is a PROPFIND request, one of the methods included in WebDAV extensions.

There is a Snort signature with ID 1079[1] that matches specific PROPFIND requests that can be used by an attacker to get directory listings configured to support WebDAV, but this is not the case in this particular packet.

The packets above indicate that the alarms at the beginning of this detect were false positives. But lets going to get more information about these remote IPs. A simple search for IP number 68.36.170.9 in the Snort alert file reveals more alerts associated with it (10 alerts):

```
# grep -B3 68.36.170.9 alert | grep \[\*\*\]

[**] [1:990:5] WEB-IIS _vti_inf access [**]
[**] [1:962:6] WEB-FRONTPAGE shtml.exe access [**]
[**] [119:13:1] (http_inspect) NON-RFC HTTP DELIMITER [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]

[**] [1:990:5] WEB-IIS _vti_inf access [**]
[**] [1:962:6] WEB-FRONTPAGE shtml.exe access [**]
[**] [119:13:1] (http_inspect) NON-RFC HTTP DELIMITER [**]
```

[1] Snort SID 1079: http://www.Snort.org/snort-db/sid.html?sid=1079

```
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via tra nslate header [**]
```

Note that there is a pattern. The attacker generated two series of 5 alerts. These alerts are commented by Danny Boulineau [8] in his detect. The first two alerts appear to check the presence and exploit a vulnerability related with Frontpage Server Extensions.

Nevertheless, if we repeat the command with IP number 213.58.17.245 the results are different (7 alerts):

```
# grep -B3 213.58.17.245 alert | grep \[\*\*\]

[**] [1:1042:6] WEB-IIS view source via transla te header [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via translate heade r [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
[**] [1:1042:6] WEB-IIS view source via translate header [**]
```

In this case all the alerts are of the same type. And none of them are HTTP GET requests.

On the other hand, if we check the timestamps of the packets listed in the previous section 'Probability the source address was spoofed', the alerts from 68.36.170.9 were generated in a shorter interval of time (3 secs.) than from 213.58.17.245 (6 secs.).

The examined information shows that the user from 68.36.170.9 ran some kind of vulnerability scan or customized script designed to exploit Microsoft IIS servers with Frontpage Server Extensions. Moreover, it is very probable that the second attacker (from 213.58.17.245) was actually a legitimate user causing false alarms.

To reduce the number of false positives in this kind of attack, I have submitted the following rule to the Snort-sigs mailing list.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
view source via translate header"; flow:to_server,established; content:
"GET "; content: "|5c|"; content: "Translate|3a| F"; nocase;
reference:cve,CVE-2000-0778; reference:arachnids,305;
reference:bugtraq,1578; classtype:web-application-activity; sid:1042;
rev:7;)
```

The modifications from previous revision number 6 consist of a new vulnerability reference (CVE-2000-0778) and two content fields that match HTTP GET requests and a trailing backslash '\' (0x5c).

2.1.6. Correlations

The details of this attack are included in the following documents:

CVE-2000-0778
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0778

Bugtraq ID 1578: Input Validation Error
http://www.securityfocus.com/bid/1578

Whitehats IDS305 "HTTP-IIS_TRANSLATE_F"
http://www.whitehats.com/info/IDS305

Although this vulnerability is certainly old (it was made public on June 2000) it is still possible to find various posts concerning this topic today:

First, there is an comprehensive email from BugTraq mailing list explaining the details of the 'Translate: f' bug. It was published on 15th August 2000 by Daniel Docekal.
URL: http://www.securityfocus.com/archive/1/76387

I found one post from the Snort-users mailing list in November 2001 by Mark Rowlands providing several log alerts of this kind of attack.
URL: http://archives.neohapsis.com/archives/snort/2001 -11/0075.html

There are two more messages at Snort-users mailing list. The one below is from 5 Jan 2004 by Elena Escolano.
URL: http://sourcef orge.net/mailarchive/message.php?msg_ id=6892220

This one from 14 Jan 2004 was published by John Bradberry. URL:
http://sourceforge.net/mailarchive/forum.p hp?thread_id=3750513&foru m_id=3972

In addition, the GCIA practicals from Sanjay Menon [9], Murray Goldschmidt [10], David Barroso [11], and Marshall Heilman [12] analyze this type of attack.

It is worthwhile mentioning that a Computer Incident Advisory Capability (CIAC) security bulletin mentions the use of ‘Automated Web Interface Scans IIS for Multiple Vulnerabilities’ where the 'Translate: f' bug is included. The bulletin can be read at:
http://www.ciac.org/ciac/bulletins/k -068.shtml

Finally , the next security bulletins includes additional information of this type of attack.

CVE-2000-0778:
http://cve.mitre.org/cgi -bin/cvename.cgi?name=CVE -2000-0778

Security Focus BID 1578:
http://www.securityfocus.com/bid/1578

2.1.7. Evidence of active targeting

As analyzed in previous sections, the attacker from 68.36.170.9 generated false alarms regarding the 'Translate: f' vulnerability. Nevertheless, the presence of other alarms suggests the use of some type of vulnerability scan tool or script designed for Microsoft IIS against IP address 32.245.166.119. The attacker could have ran the scan against more IP addresses, but Snort log file 2002.9.17 does not include any packet from 68.36.170.9 to different target IPs, so they were probably targeting only that one server. On the other hand, the exploiting of a MS IIS bug against an Apache Server reveals the lack of interest by the attacker to gather information from their victims.

Log files 2002.9.15, 2002.9.21, 2002.9.22, 2002.9.23, 2002.9.25, 2002.9.26 and 2002.9.28 present the same kind of packets from 68.36.170.9, and all of them are destined for 32.245.166.119 port 80. This reinforces the view that it is an active targeting. On the other hand, the IP address 213.58.17.245 generated false alarms, therefore active targeting it is not applicable.

2.1.8. Severity

Severity formula[1] will be applied to each source IP address separately.

Attack from IP address 68.36.170.9

Criticality = 4
The targeted machine is an Apache Web server. Most companies include in their web sites not only public information but private zones where they manage sensitive information. It is considered a critical system.

Lethality = 1
This attack exploits a known vulnerability of MS IIS 5.0 that gives access to the source of server script files such as ASP, or HTR files. The targeted machine is an Apache Web Server, therefore it is unaffected by this particular form of attack.

System Countermeasures = 5
The provided information does not reveal any special system countermeasures. Regardless, this machine would not be affected by the attack because it is an Apache server, not an IIS one.

Network Countermeasures = 2
The source logs are taken from a Snort sensor, so there is an IDS installed. As described in section one, there is no evidence of the use of a firewall.

Severity = 4 + 1 - 5 - 2 = -2

Attack from IP address 213.58.17.245

Lethality = 1
This is a false alarm.

Criticality = 4, System Countermeasures = 3, Network Countermeasures = 3
Identical score and reasons as the previous IP address.

Severity = 4 + 1 - (5 + 3) = -3

2.1.9. Defensive recommendation

The attack specifically exploits a vulnerability of Microsoft IIS 5.0, therefore the recommended action is to patch the targeted machine.

The hotfix can be downloaded from:
http://www.microsoft.com/technet/security/bulletin/MS00 -058.asp

However, we recall that in this situation, the affected machine was Apache Web server, in which case a patch is not required.

2.1.10. Multiple choice test question

The following packet dump is enough to determine that the attacker is exploiting the 'Translate: f' vulnerability? Choose the most complete answer.

[1] Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures). Each value from 1 to 5.

```
17:41:11.196507 IP 213.58.17.245.1411 > 32.245.166.119.80: P
3062892074:30628922
55(181) ack 1774865431 win 17520 (DF)
0x0000   45a0 00dd 2bc6 4000 6c06 1e01 d53a 11f5        E...+.@.l....:..
0x0010   20f5 a677 0583 0050 b690 062a 69ca  4c17        ...w...P...*i.L.
0x0020   5018 4470 7ff0 0000 5052 4f50 4649 4e44        P.Dp....PROPFIND
0x0030   202f 6d61 696e 2048 5454 502f 312e 310d        ./main.HTTP/1.1.
0x0040   0a44 6570 7468 3a20 300d 0a74 7261 6e73        .Depth:.0..trans
0x0050   6c61 7465 3a20 660d 0a55 7365 722d 4167        late:.f..User -Ag
0x0060   656e 743a 204d 6963 726f 736f 6674 2d57        ent:.Microsoft -W
0x0070   6562 4441 562d 4d69 6e69 5265 6469 722f        ebDAV -MiniRedir/
0x0080   352e 312e 3236 3030 0d0a 486f 7374  3a20        5.1.2600..Host:.
0x0090   7777 772e 5858 5858 5858 5858 0d0a 436f        www.XXXXXXXX..Co
0x00a0   6e74 656e 742d 4c65 6e67 7468 3a20 300d        ntent -Length:.0.
0x00b0   0a43 6f6e 6e65 6374 696f 6e3a 204b 6565        .Connection:.Kee
0x00c0   702d 416c 6976 650d 0a50 7261 676d 613a        p -Alive..Pragma:
0x00d0   206e 6f2d 6361 6368 650d 0a0d 0a                .no -cache....
```

a) Yes. The packet header includes 'Translate: f'.
b) Yes. PROPFIND method with the 'Translate: f' header causes the server to send the source script page to the attacker.
c) No. This is a legitimate WebDAV HTTP request that does not exploit the 'Translate: f' vulnerability.
d) No. The request does not include the URL of a script file.

Correct answer: c
This packet does not exploit the 'Translate: f' bug. At the most it could be a reconnaissance action. This vulnerability is exploited using an HTTP GET, not a PROPFIND. In some circumstances WebDAV includes a special HTTP header with 'Translate: f' that triggers false alarms. It is always recommended to check as much information as possible before considering it as an attack.

Incorrect answers:
a) The mere appearance of 'Translate: f' in the header does not necessarily mean that an attack is taking place. Further information should be collaborated, beginning with the fact that the dump does not contain a HTTP GET command. Also, the packet is using WebDAV extensions that can fire false alarms.
b) The WebDAV PROPFIND does not retrieve the source code of a script page. It retrieves properties for a resource identified by the request Uniform Resource Identifier (URI). Additional information at:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wss/wss/_webdav_propfind.asp
d) This answer is not complete. Even if the packet includes a URL of the script file it has to be followed by a trailing backslash '\'. In addition, this vulnerability is not exploited using the PROPFIND method.

References

[1] Gordon, Les. SANS GIAC practical Assignment, version 3.3. URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (17 Dec. 2003).

[2] Martin, Ian. SANS GCIA Practical Assignment, version 3.3. URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (17 Dec. 2003).

[3] IEEE. OUI and Company_id Assignments. URL: http://standards.ieee.org/regauth/oui/index.shtml (17 Dec. 2003).

[4] Wesemann, Daniel. incidents.org mailing list. 5 Jan 2003. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/01/msg00018.html (17 Dec. 2003).

[5] RCF 2518. Goland, Y. et al. HTTP Extensions for Distributed Authoring -- WEBDAV. URL: http://www.ietf.org/rfc/rfc2518.txt (17 Dec. 2003).

[6] CVE-2000-0778. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0778. (17 Dec. 2003).

[7] SecurityFocus. BID 1578. URL: http://www.securityfocus.com/bid/1578. (17 Dec. 2003).

[8] Boulineau, Danny. GIAC GCIA Version 3.2 Practical Detect Analysis - Number 1. http://cert.uni-stuttgart.de/archive/intrusions/2002/09/msg00374.html . (17 Dec. 2003).

[9] Menon, Sanjay. GIAC GCIA version 3.2. URL: www.giac.org/practical/GCIA/Sanjay_Menon_GCIA.pdf. (17 Dec. 2003).

[10] Goldschmid, Murray. GIAC GCIA version 2.8. URL: www.giac.org/practical/Murray_Goldschmidt_GCIA.doc . (17 Dec. 2003).

[11] Barroso, David. GIAC GCIA version 3.3. September 2003. URL: www.giac.org/practical/GCIA/David_Barroso_GCIA.pdf. (17 Dec. 2003).

[12] Heilman, Marshall. GIAC GCIA version 3.3. June 2003. URL: www.giac.org/practical/GCIA/Marshall_Heilman_GCIA.pdf . (17 Dec. 2003).

2.2. Detect #2 - NETBIOS DCERPC ISystemActivator bind attempt

I am analyzing these events of interest this detect.

```
[**] [1:2192:1] NETBIOS DCERPC ISystemActivator bind attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-22:22:24.189180 213.250.229.6:1611 -> X.X.X.X:135
TCP TTL:114 TOS:0x0 ID:58153 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0xA9901D04  Ack: 0x8118A3BA  Win: 0x4410  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352]

[**] [1:2192:1] NETBIOS DCERPC ISystemActivator bind attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-22:24:13.796607 212.49.171.212:1749 -> X.X.X.X:135
TCP TTL:124 TOS:0x0 ID:61753 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x9714AED8  Ack: 0x82958AF5  Win: 0x4410  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352]

[**] [1:2192:1] NETBIOS DCERPC ISystemActivator bind attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-22:24:15.488317 212.49.171.212:1891 -> X.X.X.X:135
TCP TTL:124 TOS:0x0 ID:61852 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0x979C043F  Ack: 0x82B3250F  Win: 0x4410  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352]

[**] [1:2192:1] NETBIOS DCERPC ISystemActivator bind attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-22:36:48.503215 172.189.217.185:4806 -> X.X.X.X:135
TCP TTL:115 TOS:0x0 ID:32573 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0xEC2A4AA6  Ack: 0x8CE9373A  Win: 0xFAF0  TcpLen: 20
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352]
```

2.2.1. Source of Trace

Alerts were obtained from a home network between 21:00 and 23:00 (UTC/GMT +1) from the 29th of January 2004. They were generated by a `Snort` sensor version 2.1.0 with default sets of rules. The configuration file had all rule files enabled and used default pre-processor options.

The targeted machine is a honeypot running Windows XP Professional with SP1a and assorted patches. The network diagram is illustrated below.

```
Internet ------ Router ------ Switched LAN
                  |
        Snort sensor / Firewall
             (bridge mode)
                  |
               Honeypot
```

The sensor captures only honeypot network traffic and the firewall limits the outbound connections. They are both installed in a Red Hat Linux version 9.0 running in bridge mode.

The router is doing NAPT[1] (Network Address Port Translation [1]), and the honeypot is configured as default workstation, receiving by default every connection opened from outside (Internet) to a internal port not previously included by hand in NAPT settings.

2.2.2. Detect was generated by

The logs were generated by Snort IDS (http://www.Snort.org) version 2.1.0 with rules sets from the 26th of January 2004. All rule sets and default pre-processors were enabled. The rule that triggered the alert can be found in netbios.rules file version 1.32.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 135 (msg:"NETBIOS DCERPC
ISystemActivator bind attempt"; flow:to_server,established;
content:"|05|"; distance:0; within:1; content:"|0b|"; distance:1;
within:1; byte_test:1,&,1,0,relative; content:"|A0 01 00 00 00 00 00 00
C0 00 00 00 00 00 00 46|"; distance:29; within:16; reference:cve,CAN-
2003-0352; classtype:attempted-admin; sid:2192; rev:1;)
```

This rule matches very specific packets. It is worthwhile providing a detailed description.

alert tcp
: Generate an alert an log TCP packets.

$EXTERNAL_NET any
: Look for addresses defined by $EXTERNAL_NET (usually non local network addresses) and any source ports.

$HOME_NET 135
: Look for $HOME_NET (usually defined by local area network range addresses) and remote port number 135.

msg:"NETBIOS DCERPC ISystemActivator bind attempt"
: This is the message displayed once the rule is fired.

flow:to_server,established
: The packets triggered must belong to a established session (TCP three-way handshake completed) and the direction must be from client to server.

content:"|05|"; distance:0; within:1
: Look for value 0x05 at 0 bytes of distance from the previous content (beginning of the payload). In addition, 0x05 should be 1 byte deep.

content:"|0b|"; distance:1; within:1
: Look for 0x0b value. It should be after 1 byte from the previous pattern match (0x05), and it should be of 1 byte deep.

byte_test:1,&,1,0,relative
: At a relative offset of 0 bytes from the last pattern match (0x0b), take 1 byte of the payload and perform an AND binary operation (&) with value '1'.

content:"|A0 01 00 00 00 00 00 00 C0 00 00 00 00 00 00 46|"; distance:29; within:16;

[1] NAPT technique translates many network addresses and their TCP/UDP (Transmission Control Protocol/User Datagram Protocol) ports into a single network address and its TCP/UDP ports.

Look for hexadecimal value 'A0 01 00 … ' (the interface UUID [1] of the ISystemActivator Class) after 29 bytes from the last pattern match (0x0b). The value must be found within 16 bytes.

reference:cve,CAN-2003-0352
This rule includes a reference to CVE number CAN -2003-0352.

classtype:attempted-admin
Type of attack class. The reference can be found in classification.config file. In this case it corresponds to Attempted Administrator Privilege Gain,1.

sid:2192
Snort signature ID. It can be reviewed at:
http://www.Snort.org/snort-db/sid.html?sid=2192 .

rev:1
This is the first version of the rule.

The Snort sets of rules from the 8th of February 2004 includes the revision number 2 of the previous signature (SID 2192) and more rules associated with this attack with SIDs 2193, 2350 and 2352. The new version of the rule just adds two flowbits detection plugin options to help track the state of the application protocol.

On the other hand, there is a very complete set of alternative rules at Snort-users list and at Counterpane.
http://sourceforge.net/mailarchive/message.php?msg_id=5805765
http://www.counterpane.com/alert-v20030801-001.html

Now, lets see an example of the commented rule above. Following is one of the packets that fired the alerts.

```
22:22:24.189180 IP 213.250.229.6.1611 > X.X.X.X.135: P
2844794116:2844794188(72) ack 2165875642 win 17424 (DF)
0x0000   4500 0070 e329 4000 7206 a281 d5fa e506        E..p.)@.r.......
0x0010   XXXX XXXX 064b 0087 a990 1d04 8118 a3ba        xxxx.K..........
0x0020   5018 4410 ad74 0000 0500 0b03 1000 0000        P.D..t..........
0x0030   4800 0000 7f00 0000 d016 d016 0000 0000        H...............
0x0040   0100 0000 0100 0100 a001 0000 0000 0000        ................
0x0050   c000 0000 0000 0046 0000 0000 045d 888a        .......F.....]..
0x0060   eb1c c911 9fe8 0800 2b10 4860 0200 0000        ........+.H`....
```

As explained before, byte_test option performs operation AND between 0x03 value (0000011 in binary) and 1, giving a result of 0x01. Therefore, as the result is different from 0 it means that it is TRUE (in programming language terms).

2.2.3. Probability that the source address was spoofed

Probably spoofed?

I don't think the source was spoofed due to several reasons. This rule was triggered by a packet from a previously established TCP connection. The complete captured network trace demonstrates that the packet belongs to a known series of actions that exploits a Microsoft's DCOM RPC vulnerability (more details in Description of attack and Attack mechanism section).

Probably not spoofed?

As indicated above, each source address implicated in the attacks seems to be legitimate. The attacks could have been done by a person but the behavior, frequency and number of different sources points to the action of a worm.

[1] Universal Unique Identifier , an unique 128 bit number assigned to any object within a DCE cell .

3rd party

The victim is not used as a 3rd party in the attack scenario.

2.2.4. Description of attack

This attack exploits a vulnerability in DCOM for RPC in MS Windows NT 4.0, 2000, XP, and Server 2003.

This vulnerability has been very prominent in the area of security specialists due to the fast and effective spread of Blaster worm and later variations such as MSblast, LovSAN and Nachi/Welchia.

In the attack analyzed here, the worm accesses TCP port 135 of the victim and performs a buffer exploit to open a privilege shell on TCP port 4444. This behavior is typical of Blaster worm. More details in the next section.

There are other worm variants that use different shell ports. For instance, Welchia opens a random TCP port between 666 and 765[1]. Nevertheless, an attacker with the exploit code can modify it to use other arbitrary ports.

The compromised systems become unstable and may crash. The details of this attack are described in CERT references CAN-2003-0352, CA-2003-16, CA-2003-19, and CA-2003-20. See 'Correlations' section for additional information.

2.2.5. Attack mechanism

Following is an example of the network trace taken by attacker from 213.250.229.6. The honeypot was patched against this vulnerability, therefore the exploit did not have effect. The following information was captured by a sensor running `tcpdump` (http://www.tcpdump.org) recording every network packet.

The attack begins by connecting to the victim's TCP port 135. The next packets illustrate the three-way handshake.

```
22:22:03.013042 IP 213.250.229.6.1611 > X.X.X.X.135: S 2844794115:2844794115(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:22:03.013160 IP X.X.X.X.135 > 213.250.229.6.1611: S 2165875641:2165875641(0) ack 2844794116 win 65535 <mss
1452,nop,nop,sackOK> (DF)
22:22:03.175982 IP 213.250.229.6.1611 > X.X.X.X.135: . ack 1 win 17424 (DF)
```

Once the connection is established, the attacker checks if they can access the ISystem Activator COM object of the victim. To perform this, the attacker sends a RPC BIND request. That request triggered the Snort rule of this detect (in previous sections was provided the hexadecimal dump as an example).

```
22:22:24.189186 IP 213.250.229.6.1611 > X.X.X.X.135: P 1:73(72) ack 1 win 17424 (DF)
22:22:24.189772 IP X.X.X.X.135 > 213.250.229.6.1611: P 1:61(60) ack 73 win 65463 (DF)
```

When the attacker is allowed to access the ISystemActivator, they send the buffer exploit and shellcode, included in the first of the following packets. Note that it has a data length of 1452 bytes.

```
22:22:24.260759 IP 213.250.229.6.1611 > X.X.X.X.135: . 73:1525(1452) ack 1 win 17424 (DF)
22:22:24.275860 IP 213.250.229.6.1611 > X.X.X.X.135: P 1525:1777(252) ack 1 win 17424 (DF)
22:22:24.275930 IP X.X.X.X.135 > 213.250.229.6.1611: . ack 1777 win 65535 (DF)
22:22:24.276825 IP X.X.X.X.135 > 213.250.229.6.1611: P 61:101(40) ack 1777 win 65535 (DF)
22:22:24.278881 IP 213.250.229.6.1611 > X.X.X.X.135: F 1777:1777(0) ack 1 win 17424 (DF)
22:22:24.278965 IP X.X.X.X.135 > 213.250.229.6.1611: . ack 1778 win 65535 (DF)
22:22:24.279161 IP X.X.X.X.135 > 213.250.229.6.1611: F 101:101(0) ack 1778 win 65535 (DF)
22:22:24.350719 IP 213.250.229.6.1611 > X.X.X.X.135: R 2844795893:2844795893(0) win 0 (DF)
```

[1] In the most cases the port is 707, because of the way the worm-threading model interacts with the implementation of the Windows C runtime .dll . [3]

After sending the exploit code the worm tries to open TCP port 4444 (three times). As the attack was not successful, the shell was not available on that port and the target refuses to connect.

```
22:22:24.587375 IP 213.250.229.6.2111 > X.X.X.X.4444: S 2872969082:2872969082(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:22:24.587447 IP X.X.X.X.4444 > 213.250.229.6.2111: R 0:0(0) ack 2872969083 win 0
22:22:25.213771 IP 213.250.229.6.2111 > X.X.X.X.4444: S 2872969082:2872969082(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:22:25.213851 IP X.X.X.X.4444 > 213.250.229.6.2111: R 0:0(0) ack 1 win 0
22:22:25.882685 IP 213.250.229.6.2111 > X.X.X.X.4444: S 2872969082:2872969082(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:22:25.882765 IP X.X.X.X.4444 > 213.250.229.6.2111: R 0:0(0) ack 1 win 0
```

This is a stimulus action because the attackers are actively performing an exploit on the victim. Targeted service is on TCP port 135 (DCE endpoint resolution [2]), which is present in many Microsoft Operating Systems. As mentioned in the previous section, the service running on that port presents a well known vulnerability that can allow the obtaining of administrator privileges.

Curiously, in some circumstances I found that the DCOM RPC alerts occur at the same time of several ICMP Snort alerts. I think that these ICMP alerts, repeated three times by each IP, take place as a collateral effect of this attack. To clarify this, lets have a look at the following alarm:

```
[**] [1:402:4] ICMP Destination Unreachable (Port Unreachable) [**]
[Classification: Misc activity] [Priority: 3]
01/29/04-22:23:54.524036 212.49.171.212 -> X.X.X.X
ICMP TTL:124 TOS:0x0 ID:60590 IpLen:20 DgmLen:56
Type:3  Code:3  DESTINATION UNREACHABLE: PORT UNREACHABLE
** ORIGINAL DATAGRAM DUMP:
X.X.X.X:137 -> 212.49.171.212:137
UDP TTL:92 TOS:0x0 ID:55770 IpLen:20 DgmLen:78
Len: 50
** END OF DUMP
```

ICMP Destination Unreachable (Port Unreachable) packets are sent from destination when a remote port is unreachable. In this case, the original datagram was sent from the honeypot to IP address 212.49.171.212 and remote UDP port 137. This is the dump of the original UDP packet.

```
22:23:54.283083 IP 192.168.7.51.137 > 212.49.171.212.137: udp 50
0x0000   4500 004e d9da 0000 6011 38e3 c0a8 0733        E..N....`.8....3
0x0010   d431 abd4 0089 0089 003a f802 804a 0000        .1.......:...J..
0x0020   0001 0000 0000 0000 2043 4b41 4141 4141        .........CKAAAAA
0x0030   4141 4141 4141 4141 4141 4141 4141 4141        AAAAAAAAAAAAAAAA
0x0040   4141 4141 4141 4141 4100 0021 0001             AAAAAAAAA..!..
```

UDP port 137 is associated with the NetBIOS Name Service (NBNS). NBNS, also known as Windows Internet Name Service (WINS) matches IP addresses with NetBIOS names. The packet above represents a NetBIOS name table retrieval query, also known as NetBIOS wildcard query.

I realized that only IP addresses without inverse domain name appeared in the Snort ICMP alerts. Actually, the odd ICMP packets were sent as result of NetBIOS queries sent from the honeypot to get the names of the attackers' IPs.

The following tcpdump log is an extract of the attack performed from a remote IP address without inverse domain name. Note the three UDP packets sent from the honeypot from port 137 to port 137 (NetBIOS queries) and the later ICMP

packets (Snort alerts). NetBIOS wildcard queries were sent just after completing TCP connection over TCP port 135.

```
22:23:51.884657 IP 212.49.171.212.1749 > X.X.X.X.135: S 2534715095:2534715095(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:23:51.884768 IP X.X.X.X.135 > 212.49.171.212.1749: S 2190838516:2190838516(0) ack 2534715096 win 65535 <mss
1452,nop,nop,sackOK> (DF)
22:23:52.042719 IP 212.49.171.212.1749 > X.X.X.X.135: . ack 1 win 17424 (DF)
22:23:54.283083 IP X.X.X.X.137 > 212.49.171.212.137: udp 50
22:23:54.524041 IP 212.49.171.212 > X.X.X.X: icmp 36: 212.49.171.212 udp port 137 unreachable
22:23:55.777349 IP X.X.X.X.137 > 212.49.171.212.137: udp 50
22:23:55.935143 IP 212.49.171.212 > X.X.X.X: icmp 36: 212.49.171.212 udp port 137 unreachable
22:23:57.277353 IP X.X.X.X.137 > 212.49.171.212.137: udp 50
22:23:57.434261 IP 212.49.171.212 > X.X.X.X: icmp 36: 212.49.171.212 udp port 137 unreachable
22:24:00.126105 IP 212.49.171.212.1891 > X.X.X.X.135: S 2543584318:2543584318(0) win 16384 <mss
1460,nop,nop,sackOK> (DF)
22:24:00.126220 IP X.X.X.X.135 > 212.49.171.212.1891: S 2192778510:2192778510(0) ack 2543584319 win 65535 <mss
1452,nop,nop,sackOK> (DF)
22:24:00.287535 IP 212.49.171.212.1891 > X.X.X.X.135: . ack 1 win 17424 (DF)
22:24:13.796612 IP 212.49.171.212.1749 > X.X.X.X.135: P 1:73(72) ack 1 win 17424 (DF)
22:24:13.796996 IP X.X.X.X.135 > 212.49.171.212.1749: P 1:61(60) ack 73 win 65463 (DF)
22:24:13.954600 IP 212.49.171.212.1749 > X.X.X.X.135: . 73:1525(1452) ack 1 win 17424 (DF)
[...]
```

2.2.6. Correlations

The details of this vulnerability are described in detail in the following CERT advisories:

CAN-2003-0352: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352
CA-2003-16: http://www.cert.org/advisories/CA-2003-16.html
CA-2003-19: http://www.cert.org/advisories/CA-2003-19.html
CA-2003-20: http://www.cert.org/advisories/CA-2003-20.html

Information about of the action of Blaster an Welchia worms can be read at following Symantec web pages:
http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html
http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html

The announcement and patch of the first vulnerability of DCOM RPC was published by Microsoft on July 16th 2003.
URL:
http://www.microsoft.com/technet/security/bulletin/MS03-026.asp.

There was a period of calm until August 11th, the date when the worm named 'Blaster' or 'LovSan' propagated over Internet at an incredibly fast rate affecting a great number of unpatched MS Windows systems and causing general panic in the Internet community. In addition, the worm was programmed to launch a Denial of Service against windowsupdate.com on August 16th.

Many documents at Internet analyze this vulnerability and the exploit code. One example of the first official news about this attack was titled 'RPC DCOM Worm Hits the Net', published by Kevin Poulsen at SecurityFocus on 11th August 2003 URL: http://www.securityfocus.com/news/6689 .

The exploit code was quickly published, and accelerated the appearance of several worms variations. Frederic Perriot, from Symantec Security, provided network traces and instructions to distinguish between several worms that exploit DCOM vulnerability. The document is 'Detecting network traffic that may be due to RPC worms'. URL:

http://securityresponse.symantec.com/avcenter/venc/data/detecting.traffic.due.to.rpc.worms.html

Many security specialists have been made numerous and excellent analysis of this attack and the exploit code. Some examples are from GCIHs Aaron Hackworth [4] and Brian Porter [5]. Moreover, Enric S. Hines [6] and Shannon Atkinson [7] have analyzed this attack in their practical detects.

2.2.7. Evidence of active targeting

The attacked system was targeted from more than 20 different IP addresses in less than an hour. Furthermore, although the system had a dynamic IP it received attacks immediately after it opened TCP port 135.

As discussed before, the victim was not actively targeted by the attackers. This was worm activity (Blaster or very similar) trying to exploit RPC DCOM vulnerability.

2.2.8. Severity

Seriousness = 1
The victim was a honeypot. Each service was designed to be attacked.

Lethality = 1
The exploit can give administrator privileges to the attacker. Nevertheless, the honeypot was patched, therefore it was immune to this kind of attack.

System Countermeasures = 4
The system was patched.

Network Countermeasures = 4
The inbound firewall rules permitted network traffic to every port on the honeypot, but the output traffic was highly controlled by severe output rules. In addition, a network IDS was installed to detect known attacks.

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermea sures)
Severity = 1 + 1 - (4 + 4) = -6

2.2.9. Defensive recommendation

Several tasks can be adopted for protecting against this kind of attack. All of them are provided in CERT Advisory CA-2003-16, CA-2003-19 or CA-2003-20.
URL: http://www.cert.org/advisories/CA -2003-16.html
URL: http://www.cert.org/advisories/CA -2003-19.html
URL: http://www.cert.org/advisories/CA -2003-20.html

If the system has not yet been compromised:

Filter network traffic on the following ports:

69/UDP	Trivial File Transfer (TFTP)
135/TCP	DCE endpoint resolution
135/UDP	DCE endpoint resolution
139/TCP	NETBIOS Session Service
139/UDP	NETBIOS Session Service
445/TCP	Microsoft-DS
445/UDP	Microsoft-DS
4444/TCP	Default shellcode port opened by the exploit

Apply the patch provided at Microsoft Security Bulletin MS03-026.
URL: http://microsoft.com/technet/security/bulletin/MS03 -026.asp

Disable DCOM
Generally this recommendation is not needed. If the DCOM service is disabled it can cause undesirable side effects, but the system is protected against this vulnerability. The instructions are provided at Microsoft Knowledge Base Article 825750.
URL: http://support.microsoft.com/default.aspx?scid=kb;en -us;825750

Install an updated antivirus software.

If the system has been compromised:

To determine if the system has been successfully attacked by W32/Blaster worm, we can check the following the registry key: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\windows auto update" with a value of msblast.exe. If this key is present, perform the following instructions:

- Remove it using a registry editor.
- Terminate the running copy of msblast.exe using the Task Manager.
- Take one of the previous commented steps to protect against the compromise before applying the Microsoft Security Bulletin MS03-26 patch.
- Read the document 'Recovering Windows XP systems from the W32/Blaster worm' for additional information at URL: http://www.cert.org/tech_tips/w32_blaster.html

If the system has been compromised by a modification of Blaster worm, the following document includes additional information for recovering a system:

Steps for Recovering from a UNIX or NT System Compromise
http://www.cert.org/tech_tips/win -UNIX-system_compromise.html

2.2.10. Multiple choice test question

What indicates the following Snort alert:

```
[**] [1:2192:1] NETBIOS DCERPC ISystemActivator bind attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-22:36:48.503215 172.189.217.185:4806 -> X.X.X.X:135
TCP TTL:115 TOS:0x0 ID:32573 IpLen:20 DgmLen:112 DF
***AP*** Seq: 0xEC2A4AA6 Ack: 0x8CE9373A Win: 0xFAF0 TcpLen: 20
```

a) A RCP BIND request has been sent from IP 172.189.217.185. The alert announces an imminent buffer overflow attack using vulnerability of DCOM interface for RPC in Microsoft Windows.
b) A successful overflow attack has been performed from IP 172.189.217.185, exploiting a known vulnerability of DCOM interface for RPC in Microsoft Windows.
c) One crafted packet was sent to check if the victim presents a vulnerability of DCOM interface for RPC in Microsoft Windows that can be exploited by a buffer overflow.

Correct answer: a
The rule that fired the alert was designed to trigger the BIND request made by the intruder to check if they can access the ISystem Activator COM object of the victim.

Incorrect answers:
b) The alerts do not indicate the buffer overflow itself but a RCP BIND request that precedes the ulterior buffer overflow attempt.
c) A crafted packet does not trigger this alert because the rule only fires packets from a previously established TCP connection.

References

[1] Traditional IP Network Address Translator (Traditional NAT). January 2001. URL: http://www.ietf.org/ rfc/rfc3022.txt. (1 Feb. 2004).

[2] IANA Port Numbers. URL: http://www.iana.org/assignments/port -numbers (1 Feb. 2004).

[3] Symantec. W32.Welchia.Worm. August 18, 2003. URL: http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html (1 Feb. 2004).

[4] Hackworth , Aaron. GCIH. DComExpl_UnixWin32 Windows RPC DCOM Buffer Overflow Exploit. URL: http://www.giac.org/practical/GCIH/Aaron_Hackworth_GCIH.pdf (1 Feb. 2004).

[5] Porter, Brian. GCIH. RPC-DCOM Vulnerability & Exploit. URL: http://www.giac.org/practic al/GCIH/Brian_Porter_GCIH.pdf. (1 Feb. 2004).

[6] Hines, Enric S. Part II: Network Detects Analyzing 3 Network Detects. URL: http://www.appliedwatch.com/ehines_gcia_detect1.pdf (1 Feb. 2004).

[7] Shannon Atkinson. GIAC GCIA Versi on 3.3 Practical Detect. URL: http://cert.uni-stuttgart.de/archive/intrusions/2003/08/msg00321.html (1 Feb. 2004).

[8] CAN-2003-0352. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0352 (1 Feb. 2004).

[9] Advisory CA-2003-16 Buffer Overflow in Microsoft RPC. URL: http://www.cert. org/advisories/CA-2003-16.html (1 Feb. 2004).

[10] Advisory CA-2003-19 Exploitati on of Vulnerabilities in Microsoft RPC Interface. URL: http://www.cert.org/advisories/CA -2003-19.html (1 Feb. 2004).

[11] Advisory CA-2003-20 W32/Blaster worm. URL: http://www.cert.org/advisories/CA -2003-20.html (1 Feb. 2004).

Top three question and response

The first version of this practical detect was posted on the intrusions@incidents.org mailing list the 2nd and 4th February 2004 at:
http://cert.uni-stuttgart.de/archive/intrusions/2004/02/msg00010.html

I did not receive any public response. However, I received several private emails with a lot of observations from Don Murdoch. I would like to thank him for replying to my email. I would also like to thank him for helping me to improve my practical detect with his suggestions and comments.

2.3. Detect #3 - WEB-IIS WEBDAV nessus safe scan attempt

In this detect I used a Snort log and the corresponding packet dump.

```
[**] [1:2091:2] WEB-IIS WEBDAV nessus safe scan attempt [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
01/29/04-21:28:52.377586 148.223.83.130:2707 -> X.X.X.X:80
TCP TTL:108 TOS:0x0 ID:14564 IpLen:20 DgmLen:71 DF
***AP**F Seq: 0xA04EB5B6  Ack: 0x5CD514E  Win: 0xFAF0  TcpLen: 20
[Xref => http://cgi.nessus.org/plugins/dump.php3?id=11412][Xref => http://www.securityfocus.com/bid/7116][Xref =>
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109]

---------

21:28:52.377586 IP (tos 0x0, ttl 108, id 14564, len 71) 148.223.83.130.2707 >
X.X.X.X.80: FP [tcp sum ok] 2689512886:2689512917(3 1) ack 97341774 win 64240 (DF)
0x0000  4500 0047 38e4 4000 6c06 2590  94df 5382       E..G8.@.l.%...S.
0x0010  xxxx xxxx 0a93 0050 a04e b5b6 05cd 514e       xxxx...P.N....QN
0x0020  5019 faf0 982f 0000 5345 4152  4348 202f       P..../..SEARCH./
0x0030  2048 5454 502f 312e 31 0d 0a48 6f73 743a       .HTTP/1.1..Host:
0x0040  2025 730d 0a0d 0a                              .%s....
```

The victim's IP address was hidden for privacy.

2.3.1. Source of Trace

The logs were extracted from a Linux box machine at my home network running `netcat` (http://www.atstake.com/research/tools/network_utilities/) listening on the most common service ports (such as TCP port 80) form the Internet to `dev/null`.

All the network traffic of this machine is recorded and analyzed by a Snort sensor installed in the same machine as `netcat`.

2.3.2. Detect was generated by

The alert was generated by Snort version 2.1.0 with sets of rules from the 26th of January 2004. All rules and default pre-processors were enabled. Additionally, every network packet was recorded by tcpdump.

Following is the rule that fired the alert, from web-iis.rules version 1.65 (20th November 2003).

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS
WEBDAV nessus safe scan attempt"; flow:to_server,established;
content:"SEARCH / HTTP/1.1|0d0a|Host|3a|"; content:"|0d0a0d0a|";
within:255; reference:cve,CAN-2003-0109; reference:bugtraq,7116;
reference:nessus,11412; classtype:attempted-admin; sid:2091; rev:2;)
```

Detailed rule description:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
: Alert TCP packets from $EXTERNAL_NET defined IPs and any source ports to $HTTP_SERVERS defined Ips and $HTTP_PORTS destination ports.

msg:"WEB-IIS WEBDAV nessus safe scan attempt"
: Message sent when the rule is fired.

flow:to_server,established
: To trigger the alert, TCP communication must be previously established and must come from the client to the server.

content:"SEARCH / HTTP/1.1|0d0a|Host|3a|"
: The payload of the packet must contain the string above and hexadecimal characters indicated between '|'.

content:"|0d0a0d0a|"
: The payload of the packet must have hexadecimal string 0x0d0a0d0a.

within:255
The search should not go 255 bytes past the last two contents.
reference:cve,CAN-2003-0109; reference:bugtraq,7116; reference:nessus,11412
Set of references with detailed information about the alarm. The 'Reference:' field can be found in the reference.config file.
classtype:attempted-admin
Type of attack class. In the classification.config file it c orresponds to Attempted Administrator Privilege Gain,1.
sid:2091
Snort signature ID. Additional information at: http://www.Snort.org/snort-db/sid.html?sid=2091.
rev:2
The rule has been revised twice.

2.3.3. Probability the source address was spoofed

Probably spoofed?

I don't believe that the source address was spoofed. The object of this action is to determine if the victim supports a specific service. It is a reconnaissance action. If the attacker spoofs their identity, they will never receive a response, unless they can sniff the victim's traffic (not in this case). Furthermore, packets triggered by the Snort rule come from an established TCP session.

The use of spoofed addresses is more common in Denial of Service attacks or attacks on a similar line, where the attackers only want to flood or crash their victims' equipment.

Probably not spoofed?

The information above reveals that this probability is very high. It is highly probable that the attacker did not spoof their identity.

3rd party?

This option is not applicable in this scenario. The alert was not originated by a crafted packet with victim's IP address. Moreover, the victim's IP was not used either to spoof an attack.

2.3.4. Description of attack

The attacker sent a special 'HTTP / SEARCH/1.1' command to determine if the victim supports WebDAV (http://www.ietf.org/rfc/rfc2518.txt). This converts the malicious action into a stimulus.

The port targeted was 80/tcp. As commented before there was not any HTTP server running on that port, only `netcat` to `dev/null`.

The object of this reconnaissance is to exploit a buffer overflow vulnerability in the Win32 API libraries shipped with all versions of Microsoft Windows 2000 and Microsoft Windows NT 4.0. This bug allows the remote execution of arbitrary code.

This vulnerability, described in CERT CAN-2003-0109, exists in the ntdll.dll library, a core operating system component used to interact with the Windows kernel. This dynamic link library (DLL) is used by different Windows components, such as WebDAV to process incoming requests.

A special WebDAV request sent to a Microsoft Internet Information Services (IIS) 5.0 (http://www.microsoft.com/windows2000/technologies/web/default.asp) server could permit an attacker to execute arbitrary code in the Local System, giving the attacker complete control of the system.

Note that because the vulnerable Win32 API component is utilized by other applications, it is possible to exploit the vulnerability using other vectors.

Further information and references can be found in the 'Correlations' section.

2.3.5. Attack mechanism

The Snort alert triggered by SID 2091, suggests the use of Nessus vulnerability scanner (http://www.nessus.org).

We can find details of the included Nessus plugin ID 11412 at http://cgi.nessus.org/plugins/dump.php3?id=11412 .

Following is an extract of Nessus plugin source code showing that it sends a extremely long WebDAV request to determine if it presents the vulnerability.

```
req = string("SEARCH /", crap(65535), " HTTP/1.1 \r\n",
    "Host: ", get_host_name(), " \r\n",
    "Content-Type: text/xml \r\n",
    "Content-Length: ", strlen(body), " \r\n\r\n",
    body);
```

But as we see, the captured packet from the attacker at the beginning of this detect, is only 71 bytes long including headers and payload. In addition, it did not include ‘Content-Type’ and ‘Content-Length’ fields.

Therefore, the attacker was not exploiting the buffer overflow but searching for a vulnerable victim. Not in vain, the Snort alert tells that this is a ‘safe’ scan.

2.3.6. Correlations

The details of this attack are described at:

CERT® Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL
http://www.cert.org/advisories/CA-2003-09.html

CAN-2003-0109
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109

Microsoft Windows ntdll.dll Buffer Overflow Vulnerability
http://www.securityfocus.com/bid/7116

Additionally, there are several informative documents at giac.org written by GCIHs Brandon Young [6], David Smithers [7], Bill LaRiviere [8], Trent Healy [9] and Lasse Overlier [10]. All of them study in some way the ntdll.dll buffer overflow and its relationship with WebDAV.

2.3.7. Evidence of active targeting

There was not any evidence of the attacker's IP address in the Snort logs and network traffic. It is possible that they used other machines to make earlier or subsequent actions.

The targeted machine did not have any HTTP server running at targeted TCP port 80. A clever attacker would check it first before sending more specific (and noisier) probes. This certainly discards the possibility of previous reconnaissance actions using different IP addresses.

On the other hand, as explained in the ‘Attach mechanism’ section, the attacker was not actually sending the buffer overflow itself but looking for vulnerable targets.

Therefore, I do not believe that the attacker was actively targeting the victim.

2.3.8. Severity

The following offers a grading mechanism to determine the severity of this particular attack, 5 being the most dangerous, and 1 being the most innocuous.

Criticality = 2
The targeted machine is used for testing purposes only. It provides fake services and captures network traffic.

Lethality = 1
The attack is a reconnaissance to a inexistent service.

System Countermeasures = 3
The system is up to date and file integrity tools have been installed.

Network Countermeasures = 3
The system has an NIDS installed and it is isolated from the rest of the network infrastructure.

Severity = (Criticality + Lethality) - (System Countermeasures + Network Countermeasures)
Severity = 2 + 1 - (3 + 3) = -3

2.3.9. Defensive recommendation

The following instructions can be used to protect a system against this attack.

Apply the patch included in Microsoft Security Bulletin MS03-007
http://www.microsoft.com/technet/security/bulletin/MS03 -007.asp

Disable IIS or WebDAV
Microsoft instructions to disable IIS are available at:
http://support.microsoft.com/ default.aspx?scid=kb;en-us;321141

If disabling IIS is not possible, you can disable WebDAV using the IIS lockdown tool. Information about this tool is available at:
www.microsoft.com/technet/security/tools/locktool.asp

Alternatively, you can disable WebDAV by following the instructions located in Microsoft's Knowledgebase Article 241520, "How to Disable WebDAV for IIS 5.0":
http://support.microsoft.com/ default.aspx?scid=kb;en-us;241520

Restrict buffer size
If you cannot use IIS lockdown tool, it is recommended to limit the size of the buffer that IIS utilizes to process requests. This can be achieved by using Microsoft's URL Buffer Size Registry Tool. This tool can be run against a local or remote Windows 2000 system running Windows 2000 Service Pack 2 or Service Pack 3. The tool and additional information are available at:

URL Buffer Size Registry Tool:
http://go.microsoft. com/fwlink/?LinkId=14875

Microsoft Knowledge Base Article 816930:
http://support.microsoft.com/default.aspx?scid=kb;en -us;816930

Microsoft Knowledge Base Article 260694:
http://support.microsoft.com/ default.aspx?scid=kb;en-us;260694

Use URLScan
As final recommendation, you can use URLScan to block the following WebDAV HTTP requests that attempt to exploit this vulnerability: OPTIONS, PROPFIND, PROPPATCH, MKCOL, DELETE, PUT, COPY, MOVE, LOCK, UNLOCK, OPTIONS, and SEARCH. URLScan is available at:
http://support.microsoft.com/default.aspx?scid=kb;[LN];326444

Additional recommendations and information about this attack can be read at:
CERT® Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL
http://www.cert.org/advisories/CA-2003-09.html

SecurityFocus.com. Microsoft Windows ntdll.dll Buffer Overflow Vulnerability:
http://www.securityfocus.com/bid/7116/info/

2.3.10. Multiple choice test question

Existing buffer overflow vulnerability in Windows NT/2000 ntdll.dll library (a component used to interact with the kernel) can be exploited through: (chose only one answer)

a) WebDAV.
b) WebDAV and other vectors.
c) IIS 5.0.

Correct answer: b
As noted in CERT CA-2003-09 (http://www.cert.org/advisories/CA-2003-09.html), there is buffer overflow vulnerability in Win32 API libraries of Windows NT/2000. And as the vulnerable Win32 API component is utilized by many other applications, it is possible that other exploit vectors exist. Not only WebDAV.

Incorrect answers:
a) Although this answer is correct, it is also incomplete.
c) The vulnerability cannot be exploited just through IIS 5.0. It must support an application that utilizes the affected library (ntdll.dll), such as WevDAV.

References

[1] IIS : WebDAV Overflow (MS03-007). URL: http://cgi.nessus.org/plugins/dump.php3?id=11412 (1 Feb. 2004).

[2] Snort database ID 2091. URL: http://www.Snort.org/snort-db/sid.html?sid=2091 (1 Feb. 2004).

[3] CERT. CAN-2003-0109. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109 (1 Feb. 2004).

[4] CERT Advisory CA-2003-09 Buffer Overflow in Core Microsoft Windows DLL. URL: http://www.cert.org/advisories/CA-2003-09.html (1 Feb. 2004).

[5] SecurityFocus. Microsoft Windows ntdll.dll Buffer Overflow Vulnerability. URL: http://www.securityfocus.com/bid/7116 (1 Feb. 2004).

[6] Brandon_Young "WebDAV: The new nemesis of IIS Administrators". URL: http://www.giac.org/practical/GCIH/Brandon_Young_GCIH.pdf (1 Feb. 2004).

[7] David_Smithers. “Deconstructing the NTDLL.DLL Vulnerability”. URL: http://www.giac.org/practical/GCIH/David_Smithers_GCIH.pd f (1 Feb. 2004).

[8] Bill_LaRiviere. “A KaHT in the Wild; Exploiting a Buffer O verflow in NTDLL.dll Thru WebDAV”. URL: http://www.giac.org/practical/GCIH/Bill_LaRi viere_GCIH.pdf (1 Feb. 2004).

[9] Trent_Healy. “Responding to th e WebDAV exploit”. URL: http://www.giac.org/practic al/GCIH/Trent_Healy_GCIH.pdf (1 Feb. 2004).

[10] Lasse_Overli er. “Compromising Windows 2000 core: IIS WebDAV ex ploit “. URL: http://www.giac.org/practical/GCIH/Lasse_Overlier_GCIH.pdf (1 Feb. 2004).

3. Assignment 3 - Analyze This

3.1. Executive summary

Due to its nature, a University has special security policies that make it different from other organizations. Normally it is intended to provide a certain grade of freedom. But this can be a difficult task and the source of serious security problems. The goal is to reach an ideal position between freedom of action and a restricted and secured environment. The enormous amount of traffic received from outside added to the febrile and occasionally dangerous activities carried out by the students does not facilitate the work of the administrators.

This analysis and the security recommendations were made keeping in mind the principles mentioned above, proposing further analysis or examination of suspect hosts when possible, always trying to maintain the University objectives and values. At the end of the analysis there are some several general recommendations based on the activities examined and my own experiences.

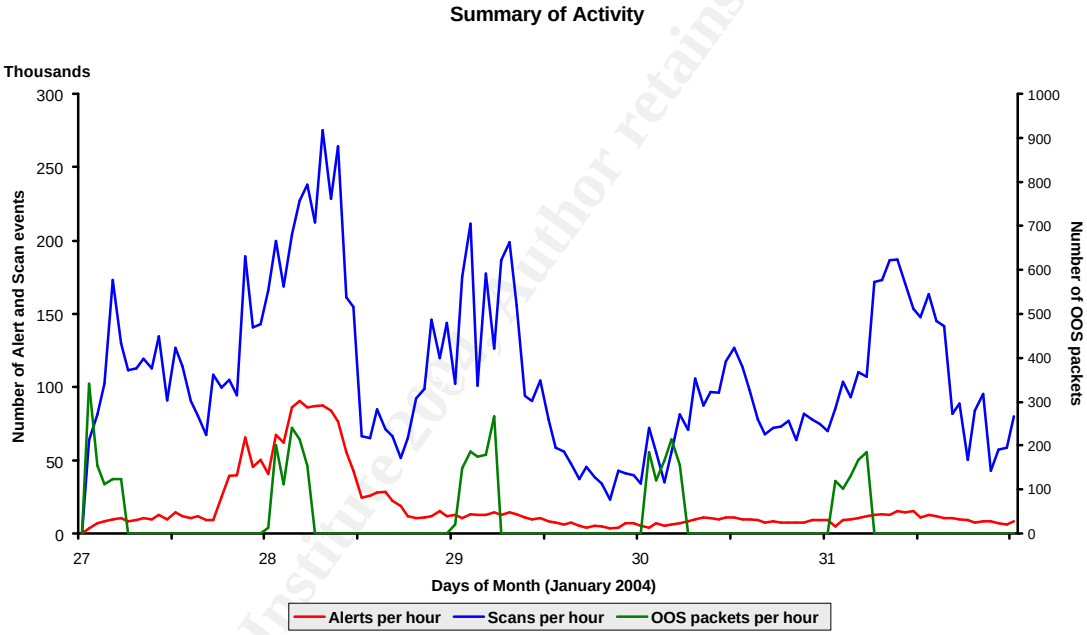

Figure 1 - Summary of Activity between 27th and 31st January 2004

The graphic above represents the overall activity recorded at the UMBC University during the period from the 27th to the 31st of January 2004. The campus hosts addresses were hidden in Alert and OOS files, but I noted that for some reason the Scans files had the real IP addresses of the University. Therefore, to avoid confusion I decided to adapt them in the same manner as in the other two types of log files. The strange OOS graphic is due to a lot of blank lines in the OOS files, probably due to data corruption. The OOS files from the analysis period include information only from about 0:00 to 4:00 hours.

3.2. Alert summary

The following table is courtesy of Les Gordon [1] and it provides an overall insight of the activity registered in alert files.

Alert name	# alerts	Ext Src	Int Dst	Int Src	Ext Dst	In	Out	I->I	E->E
High port 65535 tcp - possible Red Worm - traffic	986960	80	39	37	122	495256	491703	1	
MY.NET.30.4 activity	38035	348	1			38035			
MY.NET.30.3 activity	11841	133	1			11841			
Incomplete Packet Fragments Discarded	5692	64	278	3	14	397	5295		
High port 65535 udp - possible Red Worm - traffic	3379	81	22	16	83	1652	1727		
EXPLOIT x86 NOOP	3256	425	195			3256			
SMB Name Wildcard	2813			69	660		2813		
Null scan!	1929	137	128			1929			
Possible trojan server activity	960	33	263	15	30	836	124		
NMAP TCP ping!	955	215	58			955			
[UMBC NIDS IRC Alert] IRC user /kill detected- possible trojan.	820	44	33			820			
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	460			1	1		460		
TCP SRC and DST outside network	306	47			76				306
SUNRPC highport access!	286	21	27			286			
External RPC call	154	2	141			154			
FTP passwd attempt	139	70	5			139			
[UMBC NIDS] External MiMail alert	138	31	1			138			
Tiny Fragments - Possible Hostile Activity	116	6	7			116			
SMB C access	106	41	3			106			
Traffic from port 53 to port 123	94	1	1			94			
ICMP SRC and DST outside network	56	32			48				56
EXPLOIT x86 setgid 0	41	31	31			41			
EXPLOIT x86 setuid 0	40	32	27			40			
TFTP - Internal UDP connection to external tftp server	34	5	4	1	1	33	1		
RFB - Possible WinVNC - 010708-1	14	5	2	3	6	6	8		
EXPLOIT x86 stealth noop	13	8	8			13			
FTP DoS ftpd globbing	11	3	1			11			
Probable NMAP fingerprint attempt	10	9	5			10			
EXPLOIT NTPDX buffer overflow	7	5	4			7			
SYN-FIN scan!	7	4	4			7			
TCP SMTP Source Port traffic	6	1	1			6			
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	6	1	2			6			
NETBIOS NT NULL session	5	2	3			5			
DDOS shaft client to handler	4	4	4			4			
IRC evil - running XDCC	4			1	2		4		
Attempted Sun RPC high port access	3	3	3			3			
Fragmentation Overflow Attack	3	2	2			3			
TFTP - Internal TCP connection to external tftp server	3	1	2	1	1	2	1		
External FTP to HelpDesk MY.NET.53.29	2	2	1			2			
External FTP to HelpDesk MY.NET.70.49	2	2	1			2			
NIMDA - Attempt to execute cmd from campus host	2			2	2		2		
TFTP - External UDP connection to internal tftp server	2	1	1			2			

Alert name	# alerts	Ext Src	Int Dst	Int Src	Ext Dst	In	Out	I->I	E->E
[UMBC NIDS IRC Alert] K\:line'd user detected-possible trojan.	2	2	2			2			
[UMBC NIDS IRC Alert] Possible drone command detected.	2	1	1			2			
EXPLOIT identd overflow	1	1	1			1			
External FTP to HelpDesk MY.NET.70.50	1	1	1			1			
FTP .forward	1	1	1			1			
Happy 99 Virus	1	1	1			1			
TFTP - External TCP connection to internal tftp server	1	1	1			1			
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	1	1	1			1			
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	1			1	1		1		
Totals:	1058725	1941	1318	150	1047	556223	502139	1	362

 # alerts: Number of alerts
Ext Src: Number of unique external sources.
Int Dst: Number of unique internal destinations.
Int Src: Number of unique internal sources.
Ext Dst: Number of unique external destinations.
In: Number of inbound alerts.
Out: Number of outbound alerts.
I->I: Number of alerts triggered by both internal sources and destinations.
E->E: Number of alerts triggered by both external sources and destinations.

Note that the alert "High port 65535 tcp - possible Red Worm - traffic" has more than 93% occurrences, and the rest of the alerts only 7%. This is unusual and is analyzed in detail below.

3.3. Analysis process

Among the top ten alerts by frequency we find very similar alerts. These can be grouped by type and commented on in order of frequency. I also wanted to comment on all the alerts, so in this case I decided to adopt a similar solution to Ian Martin. The first set of alerts, analyzed in detail, includes the top ten talkers by volume. The second set, contains the alerts triggered more than 500 times. The last set contains the rest of alerts.

3.4. Alerts triggered more than 1000 times (and related)

3.4.1. High port 65535 traffic

Alert name	Severity	# alerts	In	Out	I->I	E->E
High port 65535 tcp - possible Red Worm - traffic	Medium	986960	495256	491703	1	
High port 65535 udp - possible Red Worm - traffic	Medium	3379	1652	1727		

01/27-15:15:53.047408 [**] High port 65535 tcp - possible Red Worm - traffic [**] 172.147.190.112:65535 -> MY.NET.97.189:4976
01/27-17:30:43.608988 [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.163.76:6257 -> 24.45.132.55:65535

Snort rule

There are no standard Snort rules for these alerts. The following are examples:

```
alert tcp any any -> any 65535 (msg:"High port 65535 tcp - possible Red
Worm - traffic"; flow:established; classtype:trojan-activity; rev:1;)
```

```
alert udp any any -> any 65535 (msg:"High port 65535 udp - possible Red
Worm - traffic"; classtype:trojan-activity; rev:1;)
```

Summary

The amount of this type of alerts is unusually high compared with the rest. The presence of port 65535 in network traffic suggests the activity of Red Worm. Red Worm, also known as Linux.Red.Worm, Linux/Adore, Linux/Red, is a Linux worm that opens a shell on port 65535. However, as pointed out by Doug Kite [2] this port can be used for other tools such as UNIX traceroute tool, or the WinMX file-sharing program. The following table contains the top ten TCP IP addresses and port pairs.

Top ten TCP IP and port pairs

	SRC IP	SRC Port	DST IP	DST Port	Alerts
1	24.45.132.55	65535	MY.NET.163.76	3267	**493349**
2	MY.NET.163.76	3267	24.45.132.55	65535	**489720**
3	MY.NET.84.164	1304	203.198.250.203	65535	844
4	203.198.250.203	65535	MY.NET.84.164	1304	799
5	172.147.190.112	65535	MY.NET.97.189	4976	694
6	MY.NET.97.189	4976	172.147.190.112	65535	615
7	211.23.199.82	65535	MY.NET.153.153	4662	108
8	MY.NET.153.153	4662	211.23.199.82	65535	58
9	MY.NET.34.5	65535	128.164.127.227	25	34
10	MY.NET.25.66	65535	66.93.100.200	25	28

We have found the IP addresses from which almost all the alerts originated. They were 24.45.132.55 (ool-182d8437.dyn.optonline.net) and MY.NET.163.76. The University host used TCP port 3267, associated to IBM dial-out [3]. This amount of alerts is extremely unusual and the affected host can be analyzed in detail.

The next figure illustrates the activity of the long conversation maintained by both IPs 27th Jan at 17:30 a.m., 28th Jan at 17:30 a.m. The numbers of incoming and outgoing alerts are almost identical, so they were produced by a conversation.

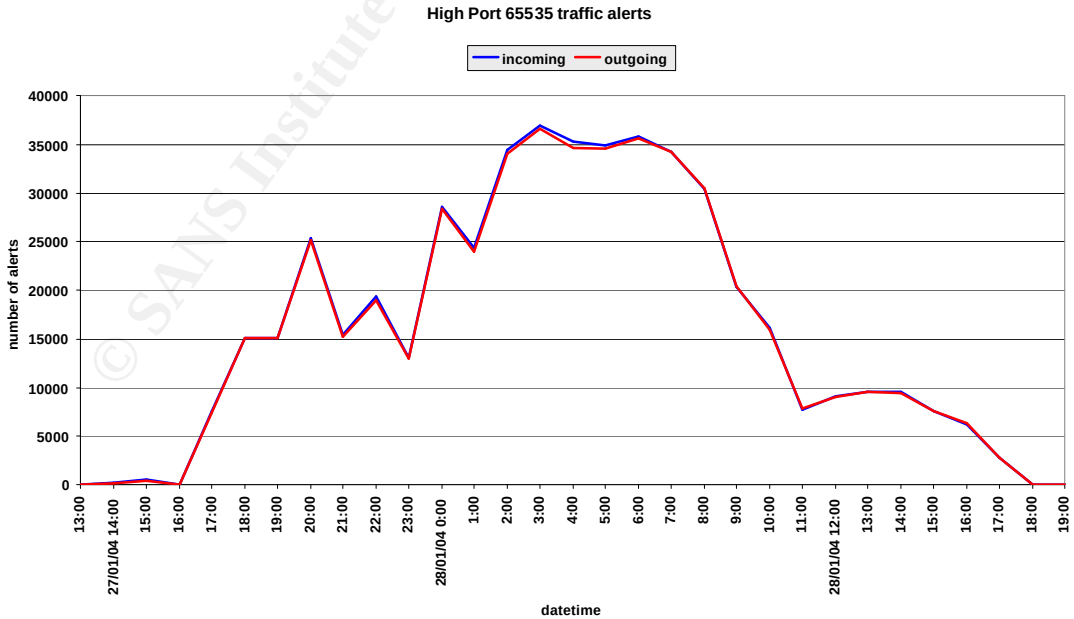

Figure 2 - High Port 65535 traffic

The next TCP port numbers in the table are 1304 and 4976. These are not used by known services or tools, but the number of alerts is high. TCP port 4662 is the default client port for eDonkey and eMule P2P file-sharing applications, and it is used by a local host. TCP port 25 is mainly used for SMTP and although it is not common, it is possible for the client to use TCP port 65535.

The table below represents the top ten UDP IP and port pairs.

Top ten UDP IP and port pairs

	SRC IP	SRC Port	DST IP	DST Port	Alerts
1	24.45.132.55	65535	MY.NET.163.76	6257	1097
2	MY.NET.163.76	6257	24.45.132.55	65535	1089
3	61.203.171.230	65535	MY.NET.163.76	6257	160
4	MY.NET.163.76	6257	61.203.171.230	65535	147
5	MY.NET.163.76	6257	218.121.232.71	65535	95
6	218.121.232.71	65535	MY.NET.163.76	6257	91
7	MY.NET.163.76	6257	61.25.24.181	65535	54
8	61.202.86.86	65535	MY.NET.152.184	6257	28
9	MY.NET.163.76	6257	81.77.18.252	65535	27
10	MY.NET.163.76	6257	218.123.68.34	65535	26

The only interesting UDP port here is number 6257 used by WinMX application [4]. Doug Kite analyzed this attack in his practical and concluded they are probably false positives. See correlations. The presence again of IP 24.45.132.55 is very suspicious and emphasizes my recommendation to investigate host MY.NET.163.76.

Except for the alerts to remote port number 25, all of the University hosts listed in the top ten TCP and UDP tables use local ports different from 65535. This suggests that they are not infected with Red Worm, but had generated false alarms.

MY.NET.25.66 also appears in a 'NMAP TCP ping!' alert from 194.206.100.2 and four 'SUNRPC highport access!' alerts from 144.126.75.19. MY.NET.34.5 is not included in any different alert.

Correlations

As mentioned above, Doug Kite [2] comments on this attack in his practical GCIA attempt, and provides logs to explain the activity of traceroute and WinMX.

Log files

The host MY.NET.84.164 presented more alerts from port 1304 different to address port 65535. This campus host was also the destination in 2 'NMAP TCP ping!' (to ports 80 and 1304) alerts, 2 'Incomplete Packet Fragments Discarded' alerts, 1 'EXPLOIT NTPDX buffer overflow' (to port 123) and 1 'EXPLOIT x86 setuid 0' (to port 1304) alert. These events seem to be isolated from the 'High port 655235' alerts.

The host MY.NET.24.74 was the objective in 14 'High port 65535' alerts against port 443 and also the target of the following alerts:

Alerts to MY.NET.24.74 destination	Alerts
High port 65535 tcp - possible Red Worm - traffic	14
NMAP TCP ping!	12

Alerts to MY.NET.24.74 dest ination	Alerts
Possible trojan server activity	9
EXPLOIT x86 setuid 0	1
Tiny Fragments - Possible Hostile Activity	1
Null scan!	1
Incomplete Packet Fragments Discarded	1

Although the response alerts from host MY.NET.24.74 does not denote signs of compromise, it should be checked as a safety measure.

The address 63.199.242.82 (adsl-63-199-242-82.dsl.sndg02.pacbell.net) that generated 75 'Incomplete Packet Fragments' to MY.NET.97.215, also triggered 6 'Null scan!' and 2 'Fragmentation Overflow Attack' against the same internal host on the 29th between 01:06:27 and 02:54:55. The address MY.NET.97.215 does not seem compromised, but it would be advisable to make contact with the ISP. The whois information is provided below:

IP address	WHOIS information	Abuse or Coordinator
63.199.242.82	SNDG02 Rback4 PPPoX Pool SBCIS - 000202-1405 (NET-63-199-240-0-1) 63.199.240.0 - 63.199.247.255 CustName: SNDG02 Rback4 PPPoX Pool Address: 303 2nd St. Address: San Francisco, CA City: StateProv: PostalCode: Country: US RegDate: 2000-02-03 Updated: 2000-02-03	OrgAbuseHandle: APB2-ARIN OrgAbuseName: Abuse - Pacific Bell OrgAbusePhone: +1-888-212-5411 OrgAbuseEmail: abuse@pacbell.net

Recommendations

If the MY.NET.163.76 host does not use IBM dial out services, it should be immediately revised to identify the reason for the 24h duration communication maintained with remote IP 24.45.132.55 from the 27th to the 28th of January.

MY.NET.84.164 and MY.NET.97.189 should be examined to determine the motives for their unusual number of alerts.

MY.NET.34.5 and MY.NET.25.66 hosts maintained communications from port 65535 to remote port 25. They should be checked as a safety measure to verify signs of compromise.

I propose to modify the customized Snort rules used to discern whether port 65535 is used by a local or a remote machine. It could help to quickly determine if there are possible infected local machines, and to reduce false alarms. Below is an example.

```
alert tcp $HOME_NET 65535 -> any any (msg:"High port 65535 tcp - possible
Red Worm internal infected machine"; flow:from_server,established;
classtype:trojan-activity; rev:1;)
```

Finally, a general recommendation in these situations is to install some type of antivirus on the hosts. This method not only protects against these kinds of threats but it can also prevent future attacks.

3.4.2. Host activity

Alert name	Severity	# alerts	In	Out	I->I	E->E
MY.NET.30.4 activity	Unknown	38035	38035			
MY.NET.30.3 activity	Unknown	11841	11841			

```
01/27-02:31:00.131442  [**] MY.NET.30.4 activity [**] 68.55.116.84:41413 -> MY.NET.30.4:524
01/27-06:29:52.533814  [**] MY.NET.30.3 activity [**] 165.247.98.160:1029 -> MY.NET.30.3:524
```

Snort rule

Once again this alert has no equivalent in the Snort rule database. However the signatures used can be similar to the following.

```
alert tcp any any -> $MY.NET.30.4 any (msg:"MY.NET.30.4 activity";
classtype:misc-activity; rev:1;)

alert udp any any -> $MY.NET.30.4 any (msg:"MY.NET.30.4 activity";
classtype:misc-activity; rev:1;)
```

Summary

These rules match the traffic where the above host addresses are involved. For any reason these hosts are interesting enough to design specific rules for them. A possible explanation for this is that they are critical servers or even honeypots. Following are the lists with the top IP addresses by volume.

Source IPs to host 30.3

	SRC IP	Alerts
1	68.50.114.89	3192
2	151.196.21.153	2246
3	131.92.177.18	2199
4	68.57.90.146	1286
5	68.55.27.157	524
6	68.55.178.168	515
7	68.55.243.80	299
8	151.196.245.167	199
9	68.81.0.87	197
10	12.65.48.159	163

Source IPs to host 30.4

	SRC IP	Alerts
1	68.54.168.204	7700
2	64.242.195.86	3274
3	68.55.241.46	2652
4	68.55.241.230	2644
5	68.55.194.168	2013
6	68.55.250.229	1862
7	68.48.213.168	1854
8	24.35.58.199	1787
9	67.20.160.15	1661
10	66.68.62.250	985

Note that both lists contain an uncommon number of IP addresses from 68.55.x.x. If these IP addresses belong to the same subnet, they could be used by an attacker with dynamic IP, or it could be some type of co-ordinated action from the same subnet. The next tables include the top five targeted (TCP or UDP) ports on both machines.

Top destination ports to host 30.3

	DST Port	Alerts
1	524	11601
2	80	85
3	6129	69
4	3019	37
5	4899	14

Top destination ports to host 30.4

	DST Port	Alerts
1	51443	32518
2	524	3047
3	80	2364
4	6129	62
5	4899	11

The searches at IANA and DShield showed the following results:

524 (tcp/udp) NCP "Netware Core Protocol" (Novell).
3019 (tcp/udp) Resource Manager (Novell).
4899 (tcp/udp) RAdmin Port [5]
6129 (tcp/udp) Dameware Remote Admin [6]

Port number 80 is mainly for HTTP but can be used by several Trojans. However the number of alerts and duration of connections doesn't point to the presence of a Trojan.
Port number 51443 (tcp/udp) is used by iFolder on a NetWare 6 server (Novell) where other NetWare Web applications are previously installed. In that situation, iFolder uses ports 51080/52080 and 51433/52433 instead of ports 80 and 443 (SSL). [7]

The exposed information denotes that these hosts are servers running Novell services. There is activity of remote administration tools like RAdmin from Famatech. This tool is well-known among attackers due to its small size (1,31 MB).

There are alerts to ports 4899 and 6129 during all the analysis period. Unless the use of these tools is known and accepted, this suggests that the hosts have been successfully compromised at some point before the 27th of January and then remotely controlled by the attackers. However the amount of these alerts is too small to assure that.

The dedicated rules that triggered these alerts reveal the importance of the targeted hosts and make it appropriate to investigate the remote IPs. The following table contains the details of the top five IP addresses by number of appearances.

IP address	WHOIS information	Abuse or Coordinator
68.48.213.168 68.50.114.89 68.54.168.204 68.55.27.157 68.55.194.168 68.55.178.168 68.55.241.46 68.55.241.230 68.55.243.80 68.55.250.229 68.57.90.146	CustName: Comcast Cable Communications, Inc Address: 3 Executive Campus Address: 5th Floor City: Cherry Hill StateProv: NJ PostalCode: 08002 Country: US RegDate: 2004-02-10 Updated: 2004-02-10	OrgAbuseHandle: NAPO-ARIN OrgAbuseName: Network Abuse and Policy Observance OrgAbusePhone: +1-856-317-7272 OrgAbuseEmail: abuse@comcast.net
64.242.195.86	OrgName: PM Hospitality Strategies, Inc. - Enginuiti, Linth OrgID: PHSIEL-1 Address: Spring Hill Suites 899 Elkridge Landing Road City: Linhicum StateProv: MD PostalCode: 21090 Country: US	OrgTechHandle: JB3051-ARIN OrgTechName: Brodt, Joe OrgTechPhone: +1-410-694-0555 OrgTechEmail: jbrodt@pmhs.com
151.196.21.153	CustName: Verizon Internet Services Address: 1880 Campus Commons Drive City: Reston StateProv: VA PostalCode: 20191 Country: US RegDate: 2002-03-21 Updated: 2002-03-21	OrgAbuseHandle: VISAB-ARIN OrgAbuseName: VIS Abuse OrgAbusePhone: +1-703-295-4583 OrgAbuseEmail: abuse@verizon.net
131.92.177.18	OrgName: Army Information Systems Command - Aberdeen (EA) OrgID: AISCAE Address: AMSSB-SCI-N/BLDG E5234 City: ABERDEEN PROVING GROUND StateProv: MD PostalCode: Country: US	TechHandle: RW943-ARIN TechName: Ward, Ronnie TechPhone: +1-410-436-4755 TechEmail: RONNIE.WARD@sbccom.apgea.army.mil

IP address	WHOIS information	Abuse or Coordinator
24.35.58.199	OrgName: Cablespeed - Maryland OrgID: CSPE Address: 406 Headquarters Dr. City: Millersville StateProv: MD PostalCode: 21108 Country: US	OrgAbuseHandle: CMAA -ARIN OrgAbuseName: Cablespeed MD Abuse Account OrgAbusePhone: +1 -410-987-9300 OrgAbuseEmail: abuse@cablespeed.com

Correlations
Ian Martin analyzed in his GCIA practical these kinds of alerts. [8]

Recommendations
The alerts indicate that both hosts have been compromised (MY.NET.30.3 and MY.NET.30.4). I suggest they be isolated from the rest of the network and a detailed forensics analysis be performed to determine the origin of the attack and data recovery. In addition, it would be advisable to examine the activity related to the affected hosts before 27th of January.

Installing a sniffer to capture all the network traffic from these hosts (at least on ports 524, 6129 and 4899) would in the future make it possible to perform a more detailed analysis of their activity.

3.4.3. Fragmentation

Alert name	Severity	# alerts	In	Out	I->I	E->E
Incomplete Packet Fragments Discarded	Low	5692	397	5295		
Tiny Fragments - Possible Hostile Activity	Medium	116	116			
Fragmentation Overflow Attack	High	3	3			

01/27-13:00:46.590567 [**] Incomplete Packet Fragments Discarded [**] 172.184.249.48:0 -> MY.NET.69.238:0
01/27-13:43:59.686482 [**] Tiny Fragments - Possible Hostile Activity [**] 141.156.55.191 -> MY.NET.24.74
01/28-21:35:57.251069 [**] Fragmentation Overflow Attack [**] 141.157.19.136:0-> MY.NET.29.3:0

Snort rule
There are not standard Snort rules for these alerts. The closest match, for tiny fragments, is the next one.

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC Tiny Fragments";
fragbits:M; dsize: < 25; classtype:bad-unknown; sid:522; rev:1;)
```

Summary
Packet fragmentation is a very common technique used to avoid NIDS and other network security devices. It is mandatory to refer to the related work by Thomas H. Ptacek in "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" of indispensable lecture." [9]

Fragmented packets can be used to scan networks using inverse mapping techniques. The attacker who sends fragmented packets to a network receives ICMP unreachable packets from the router of non existent hosts.

The 'Incomplete Packet Fragments Discarded' are produced by defrag preprocessor, superseded by newer frag2 preprocessor [10]. A significant number of these alerts can be false positives produced by transmission errors and broken TCP/IP stacks. The next table includes the top five source IP addresses by volume of this alert.

	SRC IP	Alerts
1	MY.NET.21.67	2040
2	MY.NET.21.68	1667
3	MY.NET.21.69	1588
4	193.77.45.105	222
5	63.199.242.82	57

There is an unusual amount of alerts from the three hosts MY.NET.21.67, MY.NET.21.68, and MY.NET.21.69 to the top following addresses:

	DST IP	Reverse DNS name	Alerts
1	202.129.15.241	not resolvable, from Milton, Australia (APNIC)	1291
2	213.189.88.208	dana-208.dananet.net (Amsterdam) (RIPE)	841
3	216.176.65.165	client-216-176-65-165.consolidated.net (IL, US)	686
4	83.108.190.56	ti300720a080-7736.bb.online.no	413
5	68.92.157.49	adsl-68-92-157-49.dsl.snantx.swbell.n et (TX, US)	399
6	81.76.206.46	modem-3630.fruitbat.dialup.pol.co.uk	393

The next address by frequency, IP 193.77.45.105, performed an exhaustive scan against 222 internal hosts.

The 'Tiny Fragments' packets could be used as a covert channel, sending commands within their small payload. But in this case, the number of conversations in the next table and the correlations below do not confirm this theory. They are more probably some sort of reconnaissance action.

	SRC IP	DST IP	Alerts
1	141.156.55.191	MY.NET.12.6	99
2	203.125.5.116	MY.NET.69.226	7
3	80.222.25.50	MY.NET.163.76	3
4	218.61.25.251	MY.NET.100.132	3
5	68.33.95.20	MY.NET.12.4	2

The 'Fragmentation Overflow Attack' is an alert from discontinued spp_defrag Snort preprocessor and it can be found in version 1.5.1 from July 2001 [11]. This spp_defrag was used with Snort 1.8. As commented above it is strongly recommended to update Snort to the most recent version.

Correlations

Tiny fragments, Doug kite [2], Mark Embrich [12].
Thomas H. Ptacek in "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" of indispensable lecture [9].

Log files

The IP 193.77.45.105 is also found as source address in 24 'Null scan!' alerts against miscellaneous internal hosts on the 27th beginning at 7:35:50.

The IP 141.156.55.191 also triggered 23 'Null scan!' alerts against host MY.NET.12.6.

The scan logs include activity from IP 141.156.55.191 to MY.NET.12.7 (port 38702), MY.NET.24.74 (ports 0 and 443), and MY.NET.12.6 (ports 0, 25, and several ports above 3912).

Recommendations

It is recommended to use the frag2 preprocessor to take advantage of the latest features included and reduce false positives. Moreover, as preventive measure, hosts MY.NET.21.67, MY.NET.21.68, and MY.NET.21.69 should receive a close examination to fix a possible configuration error or to find signs of compromise.

3.4.4. Exploit x86

Alert name	Severity	# alerts	In	Out	I->I	E->E
EXPLOIT x86 NOOP	Low	3256	3256			
EXPLOIT x86 setgid 0	High	41	41			
EXPLOIT x86 setuid 0	High	40	40			
EXPLOIT x86 stealth noop	High	13	13			

01/27-19:05:29.006838 [**] EXPLOIT x86 NOOP [**] 64.240.29.227:80 -> MY.NET.98.21:1178
01/27-22:22:38.333770 [**] EXPLOIT x86 setuid 0 [**] 218.5.74.158:80 -> MY.NET.98.44:2045
01/27-22:41:45.518177 [**] EXPLOIT x86 setgid 0 [**] 66.218.95.196:80 -> MY.NET.97.21:1155
01/28-15:22:42.823758 [**] EXPLOIT x86 stealth noop [**] 207.46.249.126:80 -> MY.NET.82.124:3072

Snort rule

Again, there are not any standard Snort signatures with these alert messages. I assume they were obtained from the Snort signatures SHELLCODE x86 NOOP (648 and 1394), SHELLCODE x86 setgid 0 (649), SHELLCODE x86 setuid 0 (650), SHELLCODE x86 stealth noop (651). Following are SID 648 and SID 651.

```
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE
x86 NOOP"; content: "|90 90 90 90 90 90 90 90 90 90 90 90 90 90|"; d epth:
128; reference:arachnids,181; classtype: shellcode-detect; sid:648;
rev:6;)

alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE
x86 stealth NOOP"; content: "|eb 02 eb 02 eb 02|";
reference:arachnids,291; classtype:shellcode -detect; sid:651; rev:6;)
```

Summary

These rules are designed to detect patterns in network traffic that can identify the existence of a shell code. A shell code is a piece of code used by attackers to open a shell from buffer overflow vulnerability in the compromised system.

It is important to remember that extensive use of these types of rules can degrade the sensor performance. Additionally, they are not infallible. It is common to see false positives caused for example by the transmission of binary files (see the practical detect by Terry MacDonald [13], or the post by Dragos Ruiu at [14]) or encrypted traffic [15].

'EXPLOIT x86 NOOP' is the most frequent 'EXPLOIT x86' type of alert found. Below are two tables with the top ten source and destination ports by volume.

SRC Port	Alerts	%
80	65	2,00%
4135	40	1,23%
1316	40	1,23%
4137	36	1,11%
3647	33	1,01%

DST Port	Packets	%
80	2641	81,11%
135	272	8,35%
445	89	2,73%
119	76	2,33%
6881	39	1,20%

Note that the source ports are more diversified than the destination ports. Among the destination ports we can observe that there is an unusual amount of alerts to port 80. If the destination hosts are web servers it is my belief that those alerts are

false alarms fired by the transfer of binary files in web traffic. The most active source addresses were:
65.93.189.44 (Sherbrooke-HSE-ppp3611661.sympatico.ca) with 509 alerts on the 29th
65.93.186.178 (Sherbrooke-HSE-ppp3611033.sympatico.ca) with 418 alerts on the 28th
81.166.219.254 (dyn-81-166-219-254.ppp.tiscali.fr) with 147 alerts on the 30th
The first two source addresses targeted many different internal hosts in a short interval of time, suggesting the possibility of an attack. On the other hand, the most active destination hosts were:
MY.NET.5.44, MY.NET.5.67, MY.NET.111.72, MY.NET.29.8, MY.NET.189.62, MY.NET.190.95, MY.NET.27.186 and MY.NET.95.102.

The next destination ports in the table are 135, 445 commonly used by Windows DCOM (Distributed Component Object Model). They do not represent an important number of alerts, but the University hosts could be exposed to some kind of exploit on those services. See recommendations below.

The activity corresponding to the other destination ports are summarized in the following table. It should be possible to verify the targeted hosts to discern the origin of these alerts.

	DST IP	DST Port	Alerts	%
1	MY.NET.24.8	119	76	29,92%
2	MY.NET.84.230	6881	39	15,35%
3	MY.NET.84.230	3348	8	3,15%
4	MY.NET.75.6	6129	8	3,15%
5	MY.NET.15.219	1601	7	2,76%

The rest of 'EXPLOIT x86' alerts are very distributed. The most important communications are figured below. Except for host MY.NET.24.8 (that appeared before too), they do not represent a significant threat:

	Alert message	DST IP	DST IP	DST Port	Alerts
1	EXPLOIT x86 setgid 0	131.118.254.130	MY.NET.24.8	119	8
2	EXPLOIT x86 setgid 0	216.168.224.69	MY.NET.53.45	3848	4
3	EXPLOIT x86 setuid 0	208.17.100.9	MY.NET.190.102	5049	4
4	EXPLOIT x86 stealth noop	64.152.2.62	MY.NET.97.23	1048	3
5	EXPLOIT x86 setgid 0	216.27.93.20	MY.NET.98.35	2262	2

Correlations

In addition to the references offered above, there is a must read document about buffer overflows and shell codes titled 'Smashing the Stack for Fun and Profit' by Aleph One [16].

Log files

IP 65.93.186.178 (Sherbrooke-HSE-ppp3611033.sympatico.ca) caused 5 'MY.NET.30.x activity' types of alerts to port 80 on the 28th. The other 418 alerts were destined to an important number of different internal hosts during the same day, beginning at 21:18:14. The behavior of this attacker suggests that those 5 alerts were actually 'EXPLOIT x86 NOOP' alerts against MY.NET.30.x hosts.

The internal hosts MY.NET.190.95, MY.NET.190.97 and MY.NET.190.102 presented frequent SMB type alerts and 'Possible Trojan server activity' alerts during the five days of analysis.

The IP address 65.93.186.178 (Sherbrooke-HSE-ppp3611033.sympatico.ca) that fired more than 400 'EXPLOIT x86 NOOP' alerts to internal hosts at port 80,

was also the source of 5 'MY.NET.30.3 activity' and 'MY.NET.30.4 activity' alerts on the 28th at 19:37:32. If those MY.NET.30.x hosts are honeypots, the ISP should be contacted.

IP address	WHOIS information	Abuse or Coordinator
65.93.186.178	CustName: Bell Nexxia (High Speed) Address: 400 King Street West City: Sherbrooke StateProv: Quebec PostalCode: J1H 1R4 Country: CA RegDate: 2002-01-08 Updated: 2002-01-08	OrgTechHandle: SYSAD1-ARIN OrgTechName: Sys Admin OrgTechPhone: +1-613-785-0886 OrgTechEmail: ip_prov@bellglobal.com

Recommendations

The hosts with destination port number 80 should be revised to be sure that they generated false alarms. If so, the port number 80 could be removed from the Snort signatures to prevent further noisy false alarms.

The internal hosts with destination ports 135 and 445 in the alerts should be checked and updated with the latest hotfixes. Furthermore, these ports should be blocked from outside if sharing of files is not needed.

Hosts MY.NET.24.8 and MY.NET.84.230 should be analyzed to check if they were compromised.

Finally, I personally recommend the use of spp_fnord pre-processor [17] as this reduces false positives and improves performance.

3.4.5. Server Message Block (SMB)

Alert name	Severity	# alerts	In	Out	I->I	E->E
SMB Name Wildcard	Low	2813		2813		
SMB C access	High	106	106			
NETBIOS NT NULL session	Medium	5	5			

01/27-08:39:13.048326 [**] NETBIOS NT NULL session [**] 61.197.253.6:2482 -> MY.NET.190.102:139
01/27-14:11:01.849459 [**] SMB Name Wildcard [**] MY.NET.80.197:1024 -> 192.168.1.200:137
01/27-22:22:30.387954 [**] SMB C access [**] 218.20.212.3:2202 -> MY.NET.190.95:139

Snort rule

Below are examples of Snort rules for these alerts:

```
alert udp $HOME_NET any -> $EXTERNAL_NET 137 (msg:"SMB Name Wildcard";
content:"CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA|0000|"; classtype:attempted -
recon;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS NT NULL
session"; flow:to_server,established; con tent: "|00 00 00 00 57 00 69 00
6E 00 64 00 6F 00 77 00 73 00 20 00 4E 00 54 00 20 00 31 00 33 00 38 00
31|"; reference:bugtraq,1163; reference:cve,CVE -2000-0347;
reference:arachnids,204; classtype:attempted -recon; sid:530; rev:7;)
```

The following rule is a modified version from SID 533 to match not only C$ accesses:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB C access";
flow:to_server,established; content: "|5c|C|00 41 3a
00|";reference:arachnids,339; classtype:attempted -recon;)
```

Summary

These alerts are associated to NetBIOS services running over TCP/IP (NBT). NetBIOS is commonly present in Windows Systems and it use ports 137/udp (NetBIOS Name Service, or WINS), 138/udp (NetBIOS Datagram Service) and 139/tcp (NetBIOS Session Service).

SMB (Server Message Block) is the protocol utilized by Windows over NBT.

'SMB Name Wildcard' alerts are used to obtain the name of the machine, its domain name, the network it shares, or the users online. It is common to see false positives in these alerts (see correlations). This case does not seem to be any different. They are usually fired by 'nbtstat-a' [18] or even windows explorer [19]. These alerts have source port 137 (normally Windows boxes) or a high number source port (usually UNIX machines).

The following tables contains the top five source and destination addresses by volume of 'SMB Name Wildcard' alerts:

	SRC IP	Alerts
1	MY.NET.80.197	796
2	MY.NET.150.44	396
3	MY.NET.75.13	386
4	MY.NET.150.198	330
5	MY.NET.11.4	141

	DST IP	Alerts
1	169.254.47.44	137
2	63.163.24.78	38
3	12.161.223.46	37
4	209.202.128.240	26
5	61.177.215.228	25

MY.NET.80.197 made a comprehensive UDP port 137 scan against every host in network 192.168.1.0/24. This network is included in the RFC 1918 [20] as a Private Address Space. These alerts seem to be false positives caused by windows explorer. The next hosts, MY.NET.150.44, MY.NET.75.13 and MY.NET.150.198 triggered repeated alerts to various remote IP addresses during all of the analysis period. Some of these alerts seem to be legitimate requests to determine the host name, such as the alerts to the not resolvable IPs 210.22.122.202, 65.119.229.51, 219.95.187.65. However, the numbers of alerts are high enough to make detailed investigations necessary. MY.NET.11.4 triggered 141 alerts to IP 169.254.47.44 constantly from the 27th to the 31st.

The first destination IP number is a curious one since 169.254.0.0/16 network is assigned as "link local block" by IANA in RFC 3330 [21]. In normal circumstances this IP should not be seen on the Internet. These alerts seem to be response action caused by crafted packets from that network. See recommendations below. The remaining addresses do not represent an important amount of alerts and, as mentioned below, do not have correlations in OOS or Scans logs files

The 'SMB C access' alert denotes an attempt to access the administrative share C$, enabled by default. This alert is more serious than the previous one. If the attack is successful, the intruder would have access to the C: filesystem.

There is no significant number of alerts from a unique source IP, and the top remote addresses by volume did not have any other associated alerts. All the alerts were generated to MY.NET.190.95, MY.NET.190.97 and MY.NET.190.102.

The 'NETBIOS NT NULL session' alert represents a login in Windows NT as Nobody. NULL sessions permit access to list shares and users on a Windows NT server/client.

There are only five 'NULL session' alerts with only two different source addresses: 61.197.253.6 and 203.1.68.237 and they did not cause any other alerts. Both addresses tried to open NULL sessions against MY.NET.190.95, MY.NET.190.97 and MY.NET.190.102.

Correlations

Max Vision provided recommends to trigger only incoming SMB alerts for reduce false positives [22].

There is an interesting explanation by Bryce Alexander of the alerts on UDP port 137 [18].

Daniel Martin affirms that in some circumstances the 'SMB Name Wildcard' alert is caused by windows explorer [19].

Daniel Wesemann comments the "SMB C access" attack in the second practical detect of his GCIA practical [23].

Log files

MY.NET.150.44 was the target in several 'EXPLOIT x86 NOOP' alerts: 14 from IP 65.93.186.178 on the 28th, and 6 from 217.229.150.35 on the 31st. On the other hand, MY.NET.75.13 appeared as the target in several 'EXPLOIT x86 NOOP' alerts from IP 65.93.186.178 on the 28th, and from 217.229.150.35 on the 31st, and 'Possible Trojan server activity' from 216.74.144.14 on the 30th.

Internal hosts MY.NET.190.95, MY.NET.190.97 and MY.NET.190 presented various 'EXPLOIT X86 NOOP' and 'Possible Trojan server activity' alerts in addition to the SMB type alerts during all the analysis period.

There are interesting correlations in alert files about hosts MY.NET.190.95, MY.NET.190.97 and MY.NET.190. See 'EXPLOIT x86' alerts correlations for more information.

I found that IP 202.76.92.160 (not resolvable, from Hong Kong), with 22 'SMB Name Wildcard' alerts on the 29th beginning at 00:39:50, was the source address in several 'External FTP to HelpDesk', 'MY.NET.30.3 activity' and 'MY.NET.30.4 activity' alerts too.

The OOS and Scans files do not contain information about the addresses included in the top active connections by volume showed before.

Recommendations

The top internal source addresses listed in 'SMB Name Wildcard' alerts table should be investigated to determine the origin of such an odd number of alerts. Host MY.NET.11.4 configuration should be fixed to stop the alerts to IP 169.254.47.44. The hosts from subnet MY.NET.190.0/24 should be investigated for signs of compromise.

I suggest some egress filtering to prevent spoofed attacks such as DoS (Denial of Service). The source addresses from network 169.254.0.0/16 should be blocked. For a complete list of recommended network addresses, follow the instructions provided by SANS in their document "Help Defeat Denial of Service Attacks: Step-by-Step" [24].

Additionally, it is recommended that similar ingress filtering be applied to prevent DoS attacks from illegal remote addresses.

As referred above, to reduce false positives in 'SMB Name Wildcard' alerts it is recommended to trigger incoming alerts. To avoid loosing the external alerts, I suggest to use an additional rule for them with a specific message.

Lastly, if there it is not necessary to provide NetBIOS services to the internet; it is highly recommended to immediately block incoming traffic to NetBIOS ports. Also, block outgoing NetBIOS traffic if it is not necessary, to reduce false alarms and traffic overload.

3.4.6. Scans and Fingerprinting

Alert name	Severity	# alerts	In	Out	I->I	E->E
Null scan!	Low	1929	1929			
NMAP TCP ping!	Low	955	955			
Probable NMAP fingerprint attempt	Low	10	10			
SYN-FIN scan!	Low	7	7			

01/27-23:23:36.703947 [**] Null scan! [**] 211.217.235.155:0 -> MY.NET.12.6:0
01/27-09:14:44.496887 [**] SYN-FIN scan! [**] 63.251.52.75:14297 -> MY.NET.81.125:44998
01/29-09:30:07.175341 [**] NMAP TCP ping! [**] 216.5.176.162:80 -> MY.NET.1.5:53
01/29-16:43:46.993170 [**] Probable NMAP fingerprint attempt [**] 66.135.213.40:443 -> MY.NET.97.62:3291

Snort rule
The rules below are some standard Snort rules equivalent to these alerts.

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN NULL"; stateless;
flags:0; seq:0; ack:0; reference:arachnids,4; classtype:attempted -recon;
sid:623; rev:2;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN SYN FIN";
stateless; flags:SF,12; reference:arachnids,198; classtype:attempted -
recon; sid:624; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap TCP";
stateless; flags:A,12; ack:0; reference:arachnids,28;
classtype:attempted-recon; sid:628; rev:3;)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN nmap fingerprint
attempt"; stateless; flags:SFPU; reference:arachnids,05;
classtype:attempted-recon; sid:629; rev:2;)
```

Summary
These reconnaissance actions use special TCP techniques and values as a stimulus to force responses from the targets, making possible the existence of a remote service or for fingerprinting.

Among the five most active remote addresses in 'Null scan!' alerts are the IP 211.217.235.155 (not resolvable, from Korea) and three addresses belonging to 203.210.128.0/18 network (strangely all of them have 'localhost' as inverse name; the network is associated to 'Vietnam Posts and Telecommunications (VNPT)'). These remote IP addresses generated more than 650 alerts and they targeted only the host MY.NET.12.6 to a variety of ports including port 0. In addition, address 211.217.235.155 and 203.210.158.251 generated a 'Probable NMAP fingerprint attempt' alert against MY.NET.12.6. The other remote address, IP 63.251.52.75 (www.shockwave.com) generated 202 'Null scan!' alerts to different hosts and ports during all the analysis period. Additionally it is involved in a 'SYN-FIN scan!' and a 'Probable NMAP fingerprint attempt' to host MY.NET.81.125. This amount of alerts is unusual, and it is possible that these alerts were triggered by a spoofed source address. Further investigation is recommended.

The most targeted hosts in 'Null scan!' alerts were:
MY.NET.12.6 (1084). Mainly to port 0 from top talkers 211.217.235.155 (from Korea) and three hosts from subnet 203.210.158.0/24 (from Vietnam).
MY.NET.81.125 (202) mostly to port 110 from 68.122.128.111 (adsl-68-122-128-111.dsl.sndg02.pacbell.net)
And MY.NET.12.4 (155), MY.NET.152.173 (93) and MY.NET.152.177 (47).

The top source addresses by volume that fired 'Nmap TCP Ping!' alerts, with about 408 alerts (42%) were false positives caused by load-balancing systems. The alerts were fired by 63.211.17.228 (proximitycheck1.allmusic.com) and 64.152.70.68 (proximitycheck2.allmusic.com). The remainder of source IP addresses did not generate an interesting number of alerts such as the IP 200.199.143.244 with 13 packets to MY.NET.70.164 and port 4662 (eDonkey). The most targeted hosts were MY.NET.1.3 (502), MY.NET.12.6 (96) and MY.NET.24.44 (65).

The significant information from 'Probable NMAP fingerprint attempt' alerts was commented above.

Correlations

Ian Martin examined this kind of alerts in his GCIA practical [8].

Log files

The host MY.NET.12.6 is the target of 9 different types of alerts that occurred during all the analysis period. Below is a summary. See recommendations.

Alerts to MY.NET.12.6 destination	Alerts
Null scan!	1084
[UMBC NIDS] External MiMail alert	138
Tiny Fragments - Possible Hostile Activity	99
NMAP TCP ping!	91
High port 65535 tcp - possible Red Worm - traffic	33
Possible trojan server activity	9
TCP SMTP Source Port traffic	6
Probable NMAP fingerprint attempt	5
Happy 99 Virus	1

The campus host MY.NET.12.4 was also the target of the following alerts.

Alerts to MY.NET.12.4 destination	Alerts
Null scan!	155
Incomplete Packet Fragments Discarded	13
NMAP TCP ping!	8
High port 65535 tcp - possible Red Worm - traffic	7
Tiny Fragments - Possible Hostile Activity	2

The host MY.NET.1.3 was the target of the following alerts too:

Alerts to MY.NET.1.3 destination	Alerts
NMAP TCP ping!	502
Traffic from port 53 to port 123	94
TFTP - Internal UDP connection to external tftp server	14
High port 65535 udp - possible Red Worm - traffic	3
Incomplete Packet Fragments Discarded	1

Additionally, the host MY.NET.24.44 was also actively targeted:

Alerts to MY.NET.24.44 destination	Alerts
NMAP TCP ping!	65
Possible trojan server activity	13
Null scan!	5
Incomplete Packet Fragments Discarded	5
High port 65535 tcp - possible Red Worm - traffic	4

The internal host MY.NET.70.164 received several types of scan alerts to port 4662, and their alert responses seem to indicate that it is open.

The scans files contain 8 entries from IP 211.217.235.155 to port 113 of IPs MY.NET.25.67, MY.NET.25.68 and MY.NET.25.73 on the 28th beginning at 02:49.10. Also, there are 10 entries from hosts MY.NET.25.66, MY.NET.25.67, MY.NET.25.71 to IPs 203.210.158.147 and 203.210.158.251 to port 113 on the 28th beginning at 05:12:35.

The host MY.NET.12.6 has an important presence in the OOS files. This address appeared as destination in 1173 OOS ECN SYN packets (27.17%) only to port 25 and as source in 4 ECN RST packets from port 25. Additionally, the host MY.NET.12.4 was the destinationfor 104 packets and sourcefor 1 packet, mainly to ports 25 and 110 with ECN SYN packets and TCP packets with no flags set, probably as scanning or fingerprinting attempts.

Recommendations

Due to the amount and variety of alerts the hosts mentioned in correlation section should be verified for signs of compromise. However, the alert responses from these hosts do not confirm signs of compromise or being used as 3rd party.

The host MY.NET.70.164 should be investigated to verify it accomplished the policies about P2P sharing programs.

The scan practices are widely extended in the Internet. They generally announce imminent intrusive actions. The best practice is to use them to determine attacking trends. They not present themselves as a menace.

3.5. Alerts triggered more than 500 times (and related)

3.5.1. Trojan server activity

Alert name	Severity	# alerts	In	Out	I->I	E->E
Possible trojan server activity		960	836	124		

01/30-09:10:57.663044 [**] Possible trojan server activity [**] MY.NET.75.13:25 -> 216.74.144.14:27374

Snort rule

There is no standard Snort rule for this alert and the identity of the Trojan is not supplied, so it is impossible to offer the signature used. Nevertheless all the alerts of this type that present the port number 27374, are related to Trojans and worms such as SubSeven, BadBlood, EGO, FakeSubseven, Lion, Rame, Seeker, TheSaint, Tftloader ad Webhead. The signature below could be a valid example. I assume TCP was used. However, I recommend the use of more specific alert messages to avoid confusion:

```
alert tcp any any <> any 27374 (msg:"Possible trojan server activity";
flow:established; reference:arachnids,485; classtype:trojan -activity;)
```

Summary

The most frequent alerts were caused by legitimate communications to services provided by University hosts at ports 25 (SMTP) 80 (HTTP), 443 (SSL), and 8765 (Ultraseek HTTP). The remote addresses used the port 27374 that triggered the alerts and they have not triggered different alerts.

On the other hand, there are a lot of alerts caused by incoming scanning activities against almost every host of network MY.NET.190.0/24 to TCP port 27374. The remote addresses were:

Timestamp (beginning)	SRC IP	Reverse DNS name	Alerts
2004/01/29-04:43:47	217.122.72.254	cp306825-a.gelen1.lb.home.nl	185
2004/01/31-12:57:53	68.112.209.79	cable-68-112-209-79.sli.la.charter.com	148
2004/01/31-09:21:44	24.128.135.233	h0000e88e831e.ne.client2.attbi.com	124
2004/01/31-12:37:41	67.37.224.199	adsl-67-37-224-199.dsl.chcgil.ameritech.net	117
2004/01/31-17:13:12	68.85.119.8	pcp03087321pcs.selrsv01.pa.comcast.net	75
2004/01/31-12:13:31	24.88.14.203	cae88-14-203.sc.rr.com	56

Correlations

Log files

The scan log files include the scanning activities against TCP port 27374 commented above.

Recommendations

The following internal hosts responded to TCP port 27374 scans indicating possible infection. They should be investigated for signs of compromise: MY.NET.190.1, MY.NET.190.102, MY.NET.190.202, MY.NET.190.203, MY.NET.190.95, MY.NET.190.97, and MY.NET.6.15.

In this analysis, the rule used generated a significant number of false alarms. In addition, the rule contributed to the generation of needless duplicate information about the scan activities against port 27374. To fix these problems and to be able to distinguish between simple scan activities and real worm infections, I recommend updating the signature. For instance, Snort rules with SID 103 and 107 identify subseven trojan activity, and SID 506 and 514 matches ramen worm traffic.

3.5.2. IRC

Alert name	Severity	# alerts	In	Out	I->I	E->E
[UMBC NIDS IRC Alert] IRC user /kill detected- possible trojan.	Low	820	820			
[UMBC NIDS IRC Alert] XDCC client detected attempting to IRC	High	460		460		
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	High	6	6			
[UMBC NIDS IRC Alert] K\:line'd user detected- possible trojan.	High	2	2			
[UMBC NIDS IRC Alert] Possible drone command detected.	High	2	2			
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	High	1	1			
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	High	1		1		
IRC evil - running XDCC	High	4		4		

01/29-10:23:57.176146 [**] [UMBC NIDS IRC Alert] K\:line'd user detected, possible trojan. [**] 165.123.140.251:6885 -> MY.NET.150.133:2876
01/29-10:39:44.399870 [**] [UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan. [**] 216.248.61.76:6667 -> MY.NET.42.2:2235
01/31-21:53:39.393252 [**] [UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected. [**] 216.194.70.10:7000 -> MY.NET.82.79:1111
01/31-21:53:40.792169 [**] IRC evil - running XDCC [**] MY.NET.82.79:1111 -> 216.194.70.10:7000

Snort rule

These alerts are customised and there are not any standard Snort rules similar to them. Fortunately, I found a set of IRC rules that seems to be the type used in this situation.

The following signature examples were obtained at http://arpa.com/~nick/snort, but now the URL seems to be down. The alternative address http://coders.meta.net.nz/~perry/irc.rules works fine, but has fewer rules:

```
alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content:
"ERROR:Closing Link: "; nocase; msg: "IRC user /kill detected, possible
trojan."; classtype:misc-activity;)

alert tcp $HOME_NET any -> $EXTERNAL_NET 6660:7000 (content: "USER ";
content: "dcc"; nocase; flow: established; msg: "XDCC client detected
attempting to IRC"; classtype :misc-activity;)

alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: " 324 ";
offset:5; content: "xdcc"; flow: established; msg: "User joining XDCC
channel detected. Possible XDCC bot"; classtype:misc -activity;)
```

Summary

These alerts are related to IRC (Internet Relay Chat) traffic. IRC is actively used by the hacker community to exchange information and programs and it should be closely watched. XDCC is used to share files using IRC. XDCC is like an automated file server. The document "XDCC – An .EDU Admin's Nightmare" by TonikGin [25] explains XDCC in detail.

The 'IRC user /kill detected' reveals the presence of active IRC users at hosts MY.NET.15.198, MY.NET.42.1, MY.NET.42.2, MY.NET.42.3 and MY.NET.151.72. The most visited IRC servers are:

SRC IP	Reverse DNS name	SRC Port	Alerts
64.157.246.22	not resolvable, from CO, US	6667	456
216.194.70.9	report.abuse.to.abuse.at.cjb.net, from Canada	7000	69
216.194.70.10	report.abuse.to.abuse.at.cjb.net, from Canada	7000	60
216.194.70.11	report.abuse.to.abuse.at.cjb.net, from Canada	6667	57
216.194.70.8	report.abuse.to.abuse.at.cjb.net, from Canada	7000	56

The IRC server at 64.157.246.22 address seemed to be the origin of almost all the 'XDCC clients detected attempting to IRC' alerts from host MY.NET.15.198. They were generated during all the analysis period.

IP address	WHOIS information	Abuse or Coordinator
64.157.246.22	OrgName: Tera-byte Dot Com Inc. OrgID: TRBY Address: Suite 900, CN Tower, 10004-104 Ave City: Edmonton StateProv: AB PostalCode: T5J0K1 Country: CA	TechHandle: NO58-ORG-ARIN TechName: Network Operations Centre TechPhone: +1-780-413-1868 TechEmail: noc@tera-byte.com

The IRC server at 216.194.70.8:6667 (report.abuse.to.abuse.at.cjb.net) was seen in all the 'User joining XDCC channel detected' alerts to host MY.NET.42.10 on the 27th at 22:46:55 and to host MY.NET.53.219 on the 31st after 16:45. Additionally the source host MY.NET.82.79 generated 'IRC evil - running XDCC' alerts to IRC server 216.194.70.10 on the 27th and to 216.194.70.11 on the 31st.

The internal hosts MY.NET.97.184, MY.NET.21.89, MY.NET.42.10, MY.NET.150.133, MY.NET.42.13, and MY.NET.53.219 appeared in the rest of IRC alerts.

Correlations

MY.NET.42.3 generated 13 'SMB Name Wildcard' alerts to several destination addresses on the 28th and 29th, but they do not show any relationship with these alerts.

As mentioned above, there is an extensive document about XDCC written by TonikGin titled "XDCC – An .EDU Admin's Nightmare" [25].

Logs

In addition to the IRC alerts mentioned, the source addresses 216.194.70.8, 216.194.70.9 and 216.194.70.11 using IRC source ports, triggered some 'EXPLOIT x86 NOOP' to a number of hosts at MY.NET.42.0/24 on the 29th and the 31st. See recommendations.

Furthermore, the host MY.NET.42.1, with 241 'IRC /kill' alerts, is the destination for more different alerts beginning on the 28th. The following table summarizes the alerts of this host.

Alerts to MY.NET.42.1 destination	Alerts
[UMBC NIDS IRC Alert] IRC user /kill detected - possible trojan.	247
Null scan!	21
EXPLOIT x86 NOOP	6
Incomplete Packet Fragments Discarded	3
High port 65535 tcp - possible Red Worm - traffic	1
SUNRPC highport access!	1
Probable NMAP fingerprint attempt	1
EXPLOIT x86 setuid 0	1

MT.NET.42.2, MT.NET.42.3, MT.NET.42.4 and MT.NET.42.5 were also seen as targets of similar alerts to MY.NET.42.1 during the five days of analysis.

Several campus hosts from subnet MY.NET.42.0/24 (specially the first 4 hosts) presented an important number of events in Scan files to remote ports such as: 6346/tcp, 6347/tcp, 6348/tcp, 6349/tcp (Gnutella [26] / BearShare [27])
6881/tcp to 6889/tcp (Bit Torrent [28])
4662/tcp, 4672/udp, 4665/udp (eDonkey [29] / eMule [30])
14567/udp, 14690/udp, 23000/udp (Battlefield 1942 PC game [31])

The amount of IRC alerts and the presence of P2P traffic makes interesting to represent the activity of the most active hosts. In this case, the following link diagram illustrates the behavior of campus hosts at subnet MY.NET.24.0/24.

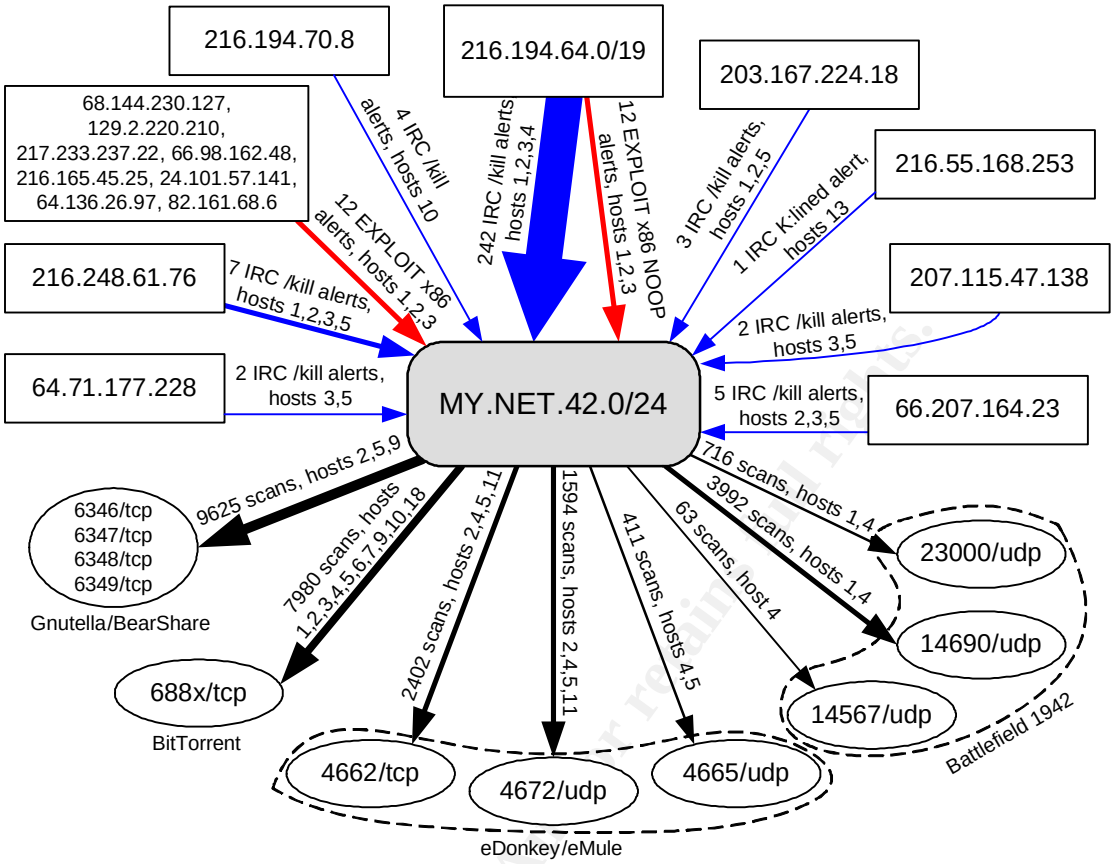

Figure 3 - Link Diagram: MY.NET.42.0/24

Recommendations

All the mentioned University hosts in this set of alerts should be revised to ensure they achieve the University policies regarding IRC programs and games. Specially the hosts at subnet MY.NET.42.0/24. IRC is a known source of security threats and although in this case it was not harmful, their activity should be always closely examined. If required there are instructions on cleaning XDCC, offered Duke University [32].

3.6. Alerts with a small number of occurrences (related)

Due to the small amount of alerts below and the maximum length restrictions, I will provide brief descriptions and recommendations for them.

3.6.1. SRC and DST outside home network and DDoS

Alert name	Severity	# alerts	In	Out	I->I	E->E
TCP SRC and DST outside network	High	306				306
ICMP SRC and DST outside network	High	56				56
DDOS shaft client to handler	Medium	4	4			

01/29-23:45:32.459412 [**] TCP SRC and DST outside network [**] 172.146.32.216:2060 -> 152.163.9.18:13784
01/29-23:01:19.818871 [**] ICMP SRC and DST outside network [**] 172.133.17.23 -> 172.136.109.238

Summary and Recommendations

These alerts appear when both source and destination addresses do not belong to the home network. In TCP alerts, the top source addresses were 192.168.1.100, 192.168.1.100, 192.168.1.103 and 127.0.0.1 (localhost) being almost 52% of the

total. The top destination IP was 206.112.85.71 (wbal.com) with 65 alerts. The ICMP alerts did not reveal remarkable information. This kind of traffic should not be seen in normal circumstances and should be properly investigated to determine their origin. Taking a look at the hardware addresses is a good beginning. Some reasons for these types of packets are configuration errors and spoofed actions (such as Denial of Service attacks, or reconnaissance attempts from an internal listening intruder). Finally, the 'DDOS shaft' was caused by 4 different sources to 4 different internal destinations, and the logs did not show interesting correlations or any responses to these stimuli from the targeted addresses.

3.6.2. RPC

Alert name	Severity	# alerts	In	Out	I->I	E->E
SUNRPC highport access!	Low	286	286			
External RPC call	Low	154	154			
Attempted Sun RPC high port access	Low	3	3			

01/30-15:06:22.492139 [**] Attempted Sun RPC high port access [**] 63.250.207.110:57258 -> MY.NET.80.44:32771
01/30-16:34:34.376947 [**] SUNRPC highport access! [**] 128.174.80.128:443 -> MY.NET.163.142:32771

Summary and Recommendations

Sun RPC (Remote Procedure Call), described in RFC 1831 [33], is basically a protocol to allow clients to execute programs on a server. These alerts are related to UNIX systems. Most of 'SUNRPC highport access!' alerts seem to be false positives caused by web and SMTP traffic. For example, the connections to web sites on ports 80 and 443 such as 207.242.93.22 (wwwa.accuweather.com), 206.98.174.20 (raba-020.raba.com) or 66.187.232.101 (xmlrpc.rhn.redhat.com). Modifying 'SUNRPC highport access!' Snort rule for trigger incoming connections (instead of any packet) to RPC port 32771 in the home network should help to reduce the high number of false positives shown. Use "flow:to_server,established" option.

The results of 'External RPC call' alerts are more interesting, because they were generated from only two source addresses: Address 61.222.174.36 (61-222-174-36.HINET-IP.hinet.net) (9 alerts) made a scan against port 111 of hosts in subnet MY.NET.190.0/24 on the 29th beginning at 18:01:58. Address 129.93.1.102 (nospam.unl.edu) (145 alerts) made a larger scan against port 111 to subnet MY.NET.190.0/24, and hosts MY.NET.6.15, MY.NET.5.5 and MY.NET.16.106 on the 29th beginning at 03:19:02. None of the targeted addresses seems to have been compromised.

3.6.3. FTP and TFTP

Alert name	Severity	# alerts	In	Out	I->I	E->E
FTP passwd attempt	Medium	139	139			
TFTP - Internal UDP connection to external tftp server	High	34	33	1		
TFTP - Internal TCP connection to external tftp server	High	3	2	1		
TFTP - External UDP connection to internal tftp server	High	2	2			
TFTP - External TCP connection to internal tftp server	High	1	1			
FTP DoS ftpd globbing	High	11	11			
External FTP to HelpDesk MY.NET.53.29	Low	2	2			
External FTP to HelpDesk MY.NET.70.49	Low	2	2			
External FTP to HelpDesk MY.NET.70.50	Low	1	1			
FTP .forward	High	1	1			

01/27-14:54:35.444098 [**] External FTP to HelpDesk MY.NET.70.49 [**] 80.13.14.66:3709 -> MY.NET.70.49:21
01/27-06:45:03.930002 [**] FTP DoS ftpd globbing [**] 213.133.108.156:44625 ->MY.NET.24.27:21
01/28-03:14:10.157656 [**] FTP passwd attempt [**] 66.149.10.46:4659 -> MY.NET.24.47:21
01/28-03:36:49.729401 [**] TFTP - Internal UDP connection to external tftp server [**] 63.71.84.104:69 -> MY.NET.1.5:123

01/31-15:56:46.004960 [**] TFTP - External TCP connection to internal tftp server [**] 81.17.55.2:60467 -> MY.NET.98.75:69

Summary and Recommendations

These customized alerts are related to FTP (File Transfer Protocol) and TFTP (Trivial FTP) and most of them were designed to trigger on the presence of certain sources or targets instead of particular exploits.

The 'FTP password attempt' alerts were performed against five internal host addresses from many different sources. The top source addresses were not found in scan logs, therefore they can be legitimate users or users that scanned the network before the analysis period. The most active source was 200.56.149.252 (customer-VER-149-252.megared.net.mx) with 48 alerts against 3 hosts in subnet MY.NET.9.0/24 and MY.NET.1.4 on the 30th. Curiously the remainder of source addresses performed FTP password attempts exclusively against MY.NET.24.97 (91 hits).

The 'FTP to HelpDesk' alerts revealed interesting correlations. The source IP 80.13.14.66 (ALyon-205-1-1-66.w80-13.abo.wanadoo.fr) triggered 1 'External FTP to HelpDesk MY.NET.70.49' and 1 'MY.NET.30.3 activity'. This was as result of a huge and noisy TCP SYN scan against TCP port 21 to the internal network beginning on the 27th at 14:47:01. Similar behavior was noted from address 202.76.92.160 (not resolvable, from Prime Spot Co Ltd, Hong Kong), firing both 'MY.NET.30.x activity' alerts, and 'External FTP to HelpDesk' alerts as a result of a large FTP scan beginning on the 29th at 00:38:13. This address also triggered several 'SMB Name Wildcard' alerts during the same period, presumably because of its lack of reverse DNS name.

The 'FTP DoS ftpd globbing' alerts seem to be related to a wu-ftpd vulnerability described in CERT CA-2001-33 [34]. The only targeted address was MY.NET.24.27 and it does not seem to have been affected by this exploit.

Due to the nature of TFPT, the hosts included in these types of alerts should be closely examined. TFTP are more insecure since this protocol does not require the use of passwords. Additionally as noted by Brian Cahoon and Ian Martin, it could be used by Nimda worm. More details in CERT CA-2001-26 [35]. The targeted campus hosts at port 69 udp/tcp did not show signs of compromise.

3.6.4. Miscellaneous

Alert name	Severity	# alerts	In	Out	I->I	E->E
[UMBC NIDS] External MiMail alert	Medium	138	138			

01/28-23:23:38.797882 [**] [UMBC NIDS] External MiMail alert [**] 131.172.138.188:4495 -> MY.NET.12.6:25

Summary and Recommendations

This customized alert seems to be related to W32/Mimail virus, described in CERT Incident Note IN-2003-02 [36]. This virus exploits vulnerability in Microsoft Outlook Express 5.5 and 6.0 and it is delivered by email within a file attachment.

The only targeted campus machine was MY.NET.12.6, during the five days of analysis. This machine was also the target of more alerts and, as commented before, it should be investigated as a safety measure. See Scans and Fingerprinting for a complete list. The most active source addresses were: 68.50.193.149 (pcp690027pcs.rtchrd01.md.comcast.net) (87) on the 31st beginning at 15:32:59, and 68.55.129.228 (pcp295208pcs.owngsm01.md.comcast.net) (12) on the 29th beginning at 13:46:16.

Alert name	Severity	# alerts	In	Out	I->I	E->E
Traffic from port 53 to port 123	High	94	94			

01/28-14:52:34.928921 [**] Traffic from port 53 to port 123 [**] 65.107.99.68:53 -> MY.NET.1.3:123

Summary and Recommendations

Both ports are well-known: Port 123 tcp/udp is assigned to NTP (Network Time Protocol) defined in RFC 958 [37] and port 53 udp/tcp to DNS (Domain Name Server). This kind of traffic should not be seen in normal circumstances. Usually for NTP communications, symmetric mode is used. NTP requests (to UDP port 123) are sent from UDP port 123. The 94 alerts were generated from the address 65.107.99.68 to MY.NET.1.3 mostly on the 29th beginning at 10:47.53.

Alert name	Severity	# alerts	In	Out	I->I	E->E
RFB - Possible WinVNC - 010708-1	High	14	6	8		

01/27-19:51:46.206462 [**] RFB - Possible WinVNC - 010708-1 [**] MY.NET.111.34:5900 -> 151.196.113.239:4481

Summary and Recommendations

This is an information alert that denotes WinVNC activity. VNC stands for Virtual Network Computing, and WinVNC [38] is a free VNC server that allows remote desktop administration on Windows systems. VNC server runs at TCP port 5900 by default. The alerts did not reveal any interesting VNC conversations. The most frequent internal host was MY.NET.111.34, scanned from different sources.

Alert name	Severity	# alerts	In	Out	I->I	E->E
NIMDA - Attempt to execute cmd from campus host	High	2		2		
Happy 99 Virus	High	1	1			

01/30-14:42:40.106021 [**] Happy 99 Virus [**] 67.163.149.58:3286 -> MY.NET.12.6:25
01/31-15:27:34.823920 [**] NIMDA - Attempt to execute cmd from campus host [**] MY.NET.84.190:1058 -> 64.70.33.122:80

Summary and Recommendations

The scanning activities performed by NIMDA Worm [35] can be identified and included in a Snort rule. These worm alerts were fired by source addresses MY.NET.84.190 and MY.NET.92.12. The source and destination hosts did not have any other correlations in the log files. As these alerts occurred on the 31st at 15:27 and at 23:28 it should be possible to verify their activity in the subsequent days.

Happy 99 virus is included in CERT Incident Note IN-99-02, and it is also known as SKA, WSOCK32.SKA, SKA.EXE, I-Worm.Happy, PE_SKA, Trojan.Happy99, Win32/SKA, and Happy99.Worm. This virus is delivered by email. This is one of a long list of alerts triggered to host MY.NET.12.6. The infected machine was 67.163.149.58 (c-67-163-149-58.client.comcast.net).

Alert name	Severity	# alerts	In	Out	I->I	E->E
EXPLOIT NTPDX buffer overflow	High	7	7			
EXPLOIT identd overflow	High	1	1			

01/27-07:19:19.181549 [**] EXPLOIT NTPDX buffer overflow [**] 65.19.157.242:28-> MY.NET.97.32:123
01/31-18:45:19.398748 [**] EXPLOIT identd overflow [**] 67.124.40.20:51326 -> MY.NET.162.164:113

Summary and Recommendations

These alerts trigger on two buffer overflow vulnerabilities. One in ntpd, Probably as described in CVE-2001-0414 [40], and another in identd.

The NTP attacks were received from 5 different sources to 4 different internal hosts. Although the attacked hosts do not show signs of compromise, they should be examined to verify they are not vulnerable to this attack. The most interesting attacks came from the address 220.124.143.13 (not resolvable, from Korea Telecom) against MY.NET.142.18. This source also triggered RPC and a 'High port 65535 udp' alerts against the same IP on the 27th beginning at 02:31:01. Two days later, the source IP 199.106.211.172 (nycny112ins-e0a.equip.icdsatt.net, from CERF, San Diego, CA) fired a NTP and 'High port 65535 udp' alerts against the same campus host beginning at 06:18:59.

The identd buffer overflow attack was sent to MY.NET.162.164 from 67.124.40.20 (adsl-67-124-40-20.dsl.pltn13.pacbell.net, from Pac Bell Internet Services, San Ramon, CA) on the 31st at 18:45:19, but the target did not show signs of compromise. It would be interesting to study further activities related to these hosts.

Alert name	Severity	# alerts	In	Out	I->I	E->E
TCP SMTP Source Port traffic	Medium	6	6			

01/28-09:18:44.343378 [**] TCP SMTP Source Port traffic [**] 65.36.154.22:25 -> MY.NET.12.6:25

Summary and Recommendations

These alerts seem to be designed to inform about outgoing traffic from TCP port 25 (SMTP, Simple Mail Transfer Protocol). The logs contain anomalous traffic since both source and destination ports are the same. Normally, the SMTP traffic (to port 25) is sent from a client port, above 1024. Additionally the 6 alerts were generated by the same source 65.36.154.22 (billingemails.hostmysite.com, from LNH Inc., Newark, DE) and destination MY.NET.12.6 addresses, on the 28th beginning at 09:18:37. The address hostmysite.com is a real web site that offers hosting solutions. The source IP has no other correlations in the log files. These alerts may have originated as a spoofed scan attempt (that could explain the odd source port) from an internal attacker listening for responses, or as a configuration error. It must be remembered that the host MY.NET.12.6 should be analyzed to determine the reasons for their unusual number of alerts.

3.7. Top 10 talkers

Top 10 Alert sources

	Internal SRCs	Alerts
1	MY.NET.163.76	491360
2	MY.NET.21.67	2040
3	MY.NET.21.68	1667
4	MY.NET.21.69	1588
5	MY.NET.84.164	865
6	MY.NET.80.197	796
7	MY.NET.97.189	615
8	MY.NET.15.198	460
9	MY.NET.150.44	396
10	MY.NET.75.13	390

External SRCs	Reverse DNS name	Alerts
24.45.132.55	ool-182d8437.dyn.opto nline.net	494449
68.54.168.204	pcp02772508pcs.howard01.md.comcast.net	7700
68.50.114.89	pcp04615078pcs.gambrl01.md.comcast.net	3519
64.242.195.86	PMHospitalityStrategies	3323
151.196.21.153	pool-151-196-21-153.balt.east.verizon.net	2932
68.55.241.46	pcp313440pcs.woodln01.md.comcast.net	2652
68.55.241.230	pcp313624pcs.woodln01.md.comcast.net	2644
131.92.177.18	aeclt-cfdoa4.apgea.army.mil	2199
68.55.194.168	pcp229869pcs.catonv01.md.comcast.net	2013
68.55.250.229	pcp261188pcs.howard01.md.comcast.net	1862

Top 10 Alert destinations

	Internal DSTs	Alerts
1	MY.NET.163.76	494855
2	MY.NET.30.4	38035
3	MY.NET.30.3	11841
4	MY.NET.12.6	1466
5	MY.NET.84.164	816
6	MY.NET.97.189	694
7	MY.NET.1.3	614
8	MY.NET.15.198	456
9	MY.NET.42.1	281
10	MY.NET.190.95	211

External DSTs	reverse DNS name	Alerts
24.45.132.55	ool-182d8437. dyn.optonline.net	490811
202.129.15.241	Not resolvable	1291
203.198.250.203	awork078203.netvigator.com	844
213.189.88.208	dana-208.dananet.net	841
216.176.65.165	client-216-176-65-165.consolidated.net	686
172.147.190.112	AC93BE70.ipt.aol.com	615
64.157.246.22	Not resolvable	460
83.108.190.56	ti300720a080-7736.bb.online.no	413
68.92.157.49	adsl-68-92-157-49.dsl.snantx.swbell.net	399
81.76.206.46	modem-3630.fruitbat.dialup.pol.co.uk	393

Top 10 alert source ports

	Internal SRCs	Alerts
1	3267	489722
2	0	5295
3	6257	1679
4	137	1362
5	1304	865
6	4976	615
7	1026	344
8	65535	332
9	1024	321
10	1081	138

	External SRCs	Alerts
1	65535	496750
2	1033	2423
3	0	2040
4	4789	1650
5	1158	1200
6	80	981
7	1047	812
8	3082	652
9	1050	641
10	6667	579

Top 10 alert destination ports

	Ports from internal SRCs	Alerts
1	65535	493097
2	0	5295
3	137	2813
4	6667	460
5	25	294
6	27374	75
7	113	38
8	427	4
9	7000	4
10	2758	3

	Ports from external SRCs	Alerts
1	3267	493349
2	51443	32518
3	524	14648
4	80	5479
5	0	2036
6	6257	1589
7	1304	812
8	27374	756
9	4976	695
10	53	561

Top source hosts with different alerts

DST IPs	Different Alerts	Alerts
202.76.92.160	5	External FTP to HelpDesk MY.NET.53.29 , External FTP to HelpDesk MY.NET.70.49 , External FTP to HelpDesk MY.NET.70.50 , MY.NET.30.3 activity , MY.NET.30.4 activity
MY.NET.153.159	3	High port 65535 udp - possible Red Worm - traffic , SMB Name Wildcard , TFTP - Internal UDP connection to external tftp server
65.93.186.178	3	EXPLOIT x86 NOOP , MY.NET.30.3 activity , MY.NET.30.4 activity
63.251.52.75	3	Null scan! , Probable NMAP fingerprint attempt , SYN-FIN scan!
63.199.242.82	3	Fragmentation Overflow Attack , Incomplete Packet Fragments Discarded , Null scan!

Top source hosts with different alerts

SRC IPs	Different Alerts	Alerts
MY.NET.12.6	9	Happy 99 Virus , High port 65535 tcp - possible Red Worm - traffic , NMAP TCP ping! , Null scan! , Possible trojan server activity ,

SRC IPs	Different Alerts	Alerts
		Probable NMAP fingerprint attempt , TCP SMTP Source Port traffic , Tiny Fragments - Possible Hostile Activity , [UMBC NIDS] E xternal MiMail alert
MY.NET.42.1	8	EXPLOIT x86 NOOP , EXPLOIT x86 setuid 0 , High port 65535 tcp - possible Red Worm - traffic, Incomplete Packet Fragments Discarded, Null scan! , Probable NMAP fingerprint attempt SUNRPC highport access! , [UMBC NIDS IRC Alert] IRC user /kill detected- possible trojan.
MY.NET.24.74	7	EXPLOIT x86 setuid 0 , High port 65535 tcp - possible Red Worm - traffic, Incomplete Packet Fragments Discarded , NMAP TCP ping! Null scan! , Possible troja n server activity, Tiny Fragments - Possible Hostile Activity
MY.NET.84.164	6	EXPLOIT NTPDX buffer overflow , EXPLOIT x86 set uid 0, High port 65535 tcp - possible Red Worm - traffic, High port 65535 udp - possible Red Worm - traffic, Incomplete Packet Fragments Discarded, NMAP TCP ping!
MY.NET.42.2	6	EXPLOIT x86 NOO P, High port 65535 tcp - possible Red Worm - traffic, Incomplete Packet Fragments Discarded , Null scan! , Probable NMAP fingerprint attempt , [UMBC NIDS IRC Alert] IRC user /kill detected - possible trojan.

Top Scan Types

	Type	Flags	Scans
1	UDP		7439853
2	SYN	******S*	5742541
3	SYN	12****S*	9034
4	FIN	*******F	2219
5	INVALIDACK	***A*R*F	936
6	NULL	********	826
7	NOACK	**U**RS*	108
8	NOACK	**U**RSF	103
9	UNKNOWN	*2***R**	76
10	UNKNOWN	*2*A**S*	67

	Type	Flags	Scans
11	UNKNOWN	*2*A****	57
12	UNKNOWN	1****R**	53
13	NOACK	**U*P*S*	49
14	UNKNOWN	12***R**	27
15	FIN	*2*****F	27
16	VECNA	****P***	26
17	NOACK	**U*PRSF	16
18	INVALIDACK	***A*RSF	15
19	VECNA	**U*****	15
20	FULLXMAS	12UAPRSF	14

Top 10 Scan sources

	Internal SRCs	Scans
1	MY.NET.1.3	3647953
2	MY.NET.162.92	1761767
3	MY.NET.81.39	1332464
4	MY.NET.80.243	878754
5	MY.NET.111.34	676795
6	MY.NET.1.4	662930
7	MY.NET.84.164	616856
8	MY.NET.153.37	447503
9	MY.NET.72.155	202545
10	MY.NET.163.76	195932

	External SRCs	Scans
1	24.224.248.157	30033
2	61.177.215.228	27481
3	217.215.115.22	26789
4	216.15.9.86	24877
5	62.103.164.195	20713
6	207.219.125.129	19829
7	62.69.96.242	19640
8	211.217.193.170	19628
9	62.101.37.108	19065
10	194.36.1.119	18790

Top 10 Scan destinations

	Internal DSTs	Scans
1	MY.NET.42.1	14971
2	MY.NET.153.18	8761
3	MY.NET.12.6	4754
4	MY.NET.6.7	2988
5	MY.NET.153.149	1271
6	MY.NET.153.173	834

	External DSTs	Scans
1	192.26.92.30	84057
2	192.5.6.30	52097
3	192.55.83.30	48630
4	203.20.52.5	46815
5	64.136.109.242	44095
6	131.118.254.34	38131

7	MY.NET.69.253	812
8	MY.NET.42.4	729
9	MY.NET.97.215	686
10	MY.NET.190.95	654

7	192.48.79.30	37527
8	216.109.116.17	37229
9	131.118.254.33	33983
10	165.230.209.227	33117

Top 10 Scan destination ports

Port	Protocol, comments	Scans
53/udp	DNS	4292083
135/tcp	DCE endpoint resolution	4119200
6129/tcp	Dameware Remote Admin - http://www.dameware.co.uk	677580
41170/udp	Piolet (P2P music sharing program) - http://www.piolet.com	467887
25/tcp	SMTP	243369
6257/udp	WinMX - http://www.winmx.com	194160
80/tcp	HTTP	143231
4899/tcp	RAdmin - http://www.famatech.com	93607
4000/tcp	[trojan] Skydance, [trojan] Connect -Back Backdoor, Terabase	65774
20168/tcp	[trojan] Lovgate	61544

Top 10 OOS talkers

	External IPs	OOS
1	68.54.84.49	1250
2	80.184.128.207	187
3	203.199.140.162	182
4	66.90.86.10	170
5	80.185.11.3	155
6	66.225.198.20	105
7	67.114.19.186	90
8	35.8.2.252	75
9	216.95.201.26	71
10	216.95.201.11	61

	Internal IPs	OOS
1	MY.NET.6.7	1281
2	MY.NET.12.6	1173
3	MY.NET.24.44	361
4	MY.NET.42.4	299
5	MY.NET.42.1	186
6	MY.NET.24.34	179
7	MY.NET.34.11	167
8	MY.NET.12.4	107
9	MY.NET.42.10	95
10	MY.NET.42.2	84

3.8. OOS files

The OOS (Out of Specification) log files contains packets that presents strange combinations of TCP flags or options or packets that do not meet the standards for valid packet structures.

The most common OOS entries (4055 alerts, 93.9%) were ECN SYN packets as follows. ECN (Explicit Congestion Notification) is defined in RFC 3168 [41].

```
01/27-00:14:00.391602 68.54.84.49:53119 -> MY.NET.6.7:110
TCP TTL:51 TOS:0x0 ID:5031 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xB32E3011  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 258548662 0 NOP WS: 0
```

ECN was designed to manage congestion in routers using the 8 and 9 bits in the TCP header. Nevertheless, this kind of packets is commonly used too by scanner and fingerprinting tools such as Nmap [42], Queso [43] or p0f [44]. Toby Miller [45] wrote an interesting article about this subject at sans.org.

The next types of packets by volume were TCP packets with no flags set, with only 128 entries (2.96%). They were triggered during the analysis period. Taking a closer look at their field values, I believe that most of them were crafted to scan and make OS fingerprinting against the campus hosts.

```
01/27-01:34:41.875156 68.122.128.111:42766 -> MY.NET.12.4:110
TCP TTL:80 TOS:0x0 ID:4660 IpLen:20 DgmLen:40
******** Seq: 0x1E0C001  Ack: 0xE37A6CAA  Win: 0x800  TcpLen: 20
```

The rest of packets (133, 3.08%) consist of OS fingerprints and scans attempts mostly to campus hosts MY.NET.42.2, MY.NET.42.1 and MY.NET.42.3, and corrupted TCP/IP stacks.

3.9. SPP_Preprocessor alerts

The activity of spp_preprocessor was separated from the rest of alerts. Among the top 20 talkers by source address, 19 are campus hosts. Number 18 is address 68.54.84.49 and it has no correlations in alert logs. I believe that the trigger limit, at 12 connections, should be raised to reduce the noise and false alarms.

After selecting STEALH alerts, the top talker source host was 68.54.84.49 (pcp01741335pcs.howard01.md.comcast.net) with 2777 alerts, and I found that in 10th position was MY.NET.42.2. It should be closely examined to determine the origin of those alerts.

The top external and internal talkers in stealth scans are listed below:

External SRCs	Type	Scans
68.54.84.49	STEALTH	2777
80.184.128.207	STEALTH	372
66.225.198.20	STEALTH	296
217.233.123.35	STEALTH	253
35.8.2.252	STEALTH	229
80.185.38.230	STEALTH	224
66.90.86.10	STEALTH	182
216.95.201.13	STEALTH	177
219.137.87.124	STEALTH	156
216.95.201.24	STEALTH	154

Internal SRCs	Type	Scans
MY.NET.42.2	STEALTH	155
MY.NET.97.16	STEALTH	70
MY.NET.97.215	STEALTH	57
MY.NET.12.6	STEALTH	48
MY.NET.12.4	STEALTH	27
MY.NET.12.7	STEALTH	10
MY.NET.97.166	STEALTH	8
MY.NET.12.2	STEALTH	3
MY.NET.70.218	STEALTH	2
MY.NET.82.79	STEALTH	1

The following addresses are some examples of important scanning activities that should be further examined.

```
2004/01/28-06:35:28.271793,End of portscan,MY.NET.42.5,TOTAL time(296s) hosts(16346) TCP(1343) UDP(15162)
2004/01/28-05:56:24.825485,End of portscan,MY.NET.1.3,TOTAL time(897s) hosts(10153) TCP(0) UDP(10600)
2004/01/28-07:07:29.481000,End of portscan,24.224.248.157,TOTAL time(96s) hosts(9434) TCP(9482) UDP(0)
2004/01/28-09:09:39.891929,End of portscan,216.15.9.86,TOTAL time(94s) hosts(8212) TCP(8259) UDP(0)
2004/01/27-05:57:00.232782,End of portscan,218.148.170.93,TOTAL time(859s) hosts(7261) TCP(7679) UDP(0)
2004/01/29-05:54:18.068398,End of portscan,MY.NET.1.3,TOTAL time(740s) hosts(7036) TCP(0) UDP(7405)
2004/01/29-06:39:43.065789,End of portscan,62.103.164.195,TOTAL time(94s) hosts(6839) TCP(6885) UDP(0)
2004/01/29-01:23:49.279885,End of portscan,207.219.125.129,TOTAL time(95s) hosts(6633) TCP(6680) UDP(0)
2004/01/31-14:58:42.828687,End of portscan,MY.NET.53.225,TOTAL time(589s) hosts(6300) TCP(287) UDP(6313)
2004/01/31-12:47:29.507253,End of portscan,MY.NET.111.34,TOTAL time(809s) hosts(6267) TCP(58) UDP(6609)
2004/01/30-12:50:22.274270,End of portscan,MY.NET.1.3,TOTAL time(738s) hosts(5931) TCP(0) UDP(6299)
```

3.10. Final conclusions and general recommendations

The logs analyzed did not represent especially dangerous security problems, taking into account that they come from a very active University with a lot of network traffic. But there are a couple of things that could be done to increase the security.

One of the most important security problems is usually associated to IRC traffic. IRC is commonly used by hackers, and their presence usually indicates the presence of a compromised host. It should definitely be blocked or at least strictly controlled. On the other hand the use of P2P (Peer-to-Peer) programs is constantly

growing and they are difficult to stop. I think that the best solution is to adopt one solution for all and deny the rest. This should make administration tasks Easier.

The installed Snort sensor does seem to be an old version such as 1.8. I strongly recommend an update of it and the set of rules, to the latest version as soon as possible. Additionally the rules used to trigger worms and special Trojan traffic has been shown to be very noisy and generated numerous false positives. I suggest they be modified to reduce such noises, as this complicates the finding of real attacks.

Finally, if not done yet, I recommend isolating the campus machines that provide external services, from the rest. This could prevent against attacks in case they are compromised.

3.11. Methodology

At the beginning of the analysis I decided to manage the log files with some common UNIX commands like grep, cut, sort or uniq. After a while I discovered that I spent more time waiting for the command responses, watching the screen like a zombie, than analyzing the results. Then I noted that lots of students had excellent results using script languages such as Perl and some type of database. From that moment I thought the best solution was to use Perl a MySQL. I designed a customized Perl script, based on the excellent work by Tod Beardsley [46] and Andre Cormier [47] and I specially improved the techniques to discern between correct and corrupted logs. For the database, I combined the tables used by Andre Cormier and Les Gordon [1].

3.11.1. Files analyzed

The following files were used for the analysis. They were decompressed and processed by a customized Perl script based on csv.pl by Tod Beardsley and db_loader.pl by Andre Cormier. The script processed the files separately and merged them into four groups: alerts, alert_spp_scans, oos, and scans, saving space and avoiding losses derived from concatenation. The total size after decompressing and processing the logs was about 1.04 GB.

	size (bytes)		size (bytes)		size (bytes)
alert.040127.gz	4.498.344	scans.040127.gz	20.018.546	oos_report_040123	270.336
alert.040128.gz	9.035.779	scans.040128.gz	26.275.556	oos_report_040124	1.310.720
alert.040129.gz	2.301.305	scans.040129.gz	16.434.604	oos_report_040125	1.735.680
alert.040130.gz	2.233.954	scans.040130.gz	17.140.053	oos_report_040126	270.336
alert.040131.gz	2.688.552	scans.040131.gz	25.556.327	oos_report_040127	237.568
	20.757.934		105.425.086		3.824.640

The names of the OOS log files did not match with the timestamps of the packets. The OOS packets were captured from the 27th to the 31st of January. By some reason, these files are processed with four days of delay.

3.11.2. Examples of log entries

My Perl script put the wrong log entries into separate files. These were the results:

```
ALERT:
Total:   2208262  Accepted:   2205230 (99.863%)  Rejected:   3032 (0.137%)
SCAN:
Total:  13196996  Accepted:  13196962 (100.00%)  Rejected:     34 ( 0.00%)
OOS:
Total:      4321  Accepted:      4316 (99.88 4%)  Rejected:      5 (0.116%)
```

Below are several examples of the logs after being processed with the script:

Accepted logs

Alerts

```
2004/01/27-02:30:34.901167,MY.NET.30.4 activity,68.55.116.84,41413,MY.NET.30.4,524
2004/01/27-07:10:28.353575,Null scan!,193. 77.45.105,0,MY.NET.15.30,0
2004/01/27-07:10:32.653639,Incomplete Packet Fragments Discarded,193.77.45.105,0,MY.NET.13.2,0
```

Scans

```
2004/01/27-03:37:51,211.217.235.107,0, MY.NET.12.6,0,VECNA,12U****F,RESERVEDBITS
2004/01/27-04:01:30,211.217.235.107,38, MY.NET.12.6,4544,NULL,********,
2004/01/27-05:12:55,206.41.205.250,59310, MY.NET.97.18,6346,FIN,*******F,
```

OOS

```
2004/01/27-00:34:01.407572,203.199.140.162,3175,MY.NET.24.34,80,TCP,46,0,17980,20,60,X,,  \
12****S*,30 01170644,0,5840,40,5,MSS: 1460 SackOK TS: 368540904 0 N  OP WS: 0
2004/01/31-04:35:49.843951,68.111.35.228,52962,MY.NET.69.226,6883,TCP,108,0,59545,20,40,X,,  \
********,704905609,234193192,0,0,None,None
```

Rejected logs

Alerts

```
:65535 -> MY.NET.163.76:3267
01/27-18:19:40.595113  [**] High port 65535 tcp  - possible Red Worm - traffic [**] \
24.45.132.5501/27-18:43:15.147845  [**] spp_portscan: portscan status from MY.NET.1.4:  \
8 connections across 8 hosts: TCP(0), UDP(8) [**]
:65535 -> MY.NET.163.76:3267
:65535
:1304 -> 203.198.250.203:65535
```

Scans

```
JaJan 27 03:48:38  MY.NET.162.92:3015 -> 177.136.177.177:135 SYN ******S*
n 27 03:39:41 195.136.73.130:50216  -> MY.NET.100.133:1524 SYN ******S*
Jan 27 03:39:42 195.1Jan 27 03:48:38  MY.NET.81.39:1755 -> 84.72.183.225:135 SYN ******S*
36.73.130:50400  -> MY.NET.101.59:1524  SYN ******S*
198.247.172.10:25 SYN ******S*
S*
```

OOS

```
01/31-04:52:13.511518
TCP TTL:51 TOS:0x0 ID:
```

3.11.3. Examples of SQL queries used

Top 10 internal source addresses in alert files
SELECT src_ip, count(*) as num FROM alerts where src_ip like 'MY.NET%' group by src_ip order by num desc limit 10

Top 10 source ports from external addresses in alert files
SELECT src_port, count(*) as num FROM alerts where src_ip not like 'MY.NET%' group by src_port order by num desc limit 10

TOP 10 external source addresses in scan files
SELECT src_ip, count(*) as num from scans where src_ip not like ' MY.NET.%' group by src_ip order by num desc limit 10

Top 10 source addresses in OOS files
SELECT src_ip, count(*) num FROM oos group by src_ip order by num desc limit 10

3.12. References

[1] Gordon, Les. GCIA practical. 22 November 2002 . URL: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc (10 Feb. 2004).

[2] Kite, Doug. GCIA practical. URL: http://www.giac.org/practic al/GCIA/Doug_Kite_GCIA.pdf (12 Feb. 2003).

[3] IANA Port Numbers. Last updated 26 February 2004. URL: http://www.iana.org/assignm ents/port-numbers (29 Feb. 2003).

[4] Freezmizer. Post on WinMXCommunity.com Forum: "Changing TCP and UDP port settings". 12 April 2003. http://forums.winmxcommunity.com/viewtopic.php?t=198 7 (10 Feb. 2004).

[5] Famatech. Radmin. URL: http://www.famatech.com/ (10 Feb. 2004).

[6] Dameware. Remote Admin. URL: http://www.dameware.co.uk/thankyouremote. asp (10 Feb. 2004).

[7] Novell. BorderManager Port Numbers (iFolder). URL: http://developer.novell.co m/research/sections/netsupport/abend/2002/august/x02080 1.doc (10 Feb. 2004).

[8] Martin, Ian. GIAC practical. 17 July 2003. URL: http://www.giac.org/practical/GCIA/Ian_Martin_GCIA.pdf (12 Feb. 2003).

[9] Ptacek, Thomas H. Insertion, "Evasion, and Denial of Service: Eluding Network Intrusion Detection". 16 Oct. 2002.URL: http://secinf.net/info/ids/idspaper/idspaper.html (10 Feb. 2004).

[10] Roesch, Martin. "Re: [Snort -users] Incomplete P acket Fragments Discarded". Email on snortusers@lists.sourceforge.net mailing list. 26 Nov 2001. URL: http://www.mc abee.org/lists/snort-users/Nov-01/msg00820.html (10 Feb. 2004).

[11] Ruiu, Dragos. "[Snort -users] spp_defrag.c v1.5.1". Email on snortusers@lists.sourceforge.net mailing list. 10 July 2001. URL: http://archives.neohapsis.com/archives/snort/2001 -07/0202.html (11 Feb. 2004).

[12] Embrich, Mark. GCIA Practical. 12 February 2002. URL: http://www.giac.org/practical/Mark_Embrich_GCIA.htm (11 Feb. 2004).

[13] MacDonald, Terry. GCIA Practical. Practical 3. 21 June 2003. URL: www.giac.org/ practical/GCIA/Terry_MacDon ald_GCIA.pdf (13 Feb. 2003).

[14] Ruiu, Dragos. "Re: [Snort -users] SHELLCODE x86 unicode NOOP". Email on snortusers@lists.sourceforge.net mailing list. 22 April 2002. URL: http://www.mc abee.org/lists/snort-users/Apr-02/msg00763.html (22 Feb. 2004).

[15] Thiele, Fred GCIA Practical. 2003. URL: http://www.giac.org/practic al/GCIA/Fred_Thiele_GCIA.pdf (21 Feb. 2003).

[16] One, Aleph. "Smashing The Stack For Fun And Profit". 8 Nov. 1996. URL: http://www.phrack.org/phrack/49/P49 -14 (14 Feb. 2004).

[17] Ruiu, Dragos. "mutants! - spp_fnord. c (It can see the FNORDs! : -)". Email on bugtraq@securityfo cus.com mailing list. 1 March 2002. URL: http://cert.uni -stuttgart.de/archive/bugtraq/2002/03/msg00088.html (20 Feb. 2004).

[18] Alexander, Bryce. "Port 137 Sca n". 10 May 2000. URL: http://www.sans.org/re sources/idfaq/port_137.php (12 Feb. 2004).

[19] Martin, Daniel. "Re: Spoofed SMB name wildcard probes". Email on incidents@lists.securityfocus.com mailing list. 4 May 2001 . URL: http://cert.uni - stuttgart.de/archive/incidents/2001/05/msg00041.html (12 Feb. 2004).

[20] Rekhter, Y. et al. RCF 1918: "Address Allocation for Private Internets". February 1996. URL: http://www.ietf.org/rf c/rfc1918.txt (17 Feb. 2004).

[21] IANA. RCF 3330: "Special -Use IPv4 Addresses". September 2002. URL: http://www.ietf.org/ rfc/rfc3330.txt (17 Feb. 2004).

[22] Vision, Max. "Re: [snort] 'SMB Name Wildcard'". Email on Snort IDS mailing list by Snort.org. 17 Jan. 2000. URL: http://archi ves.neohapsis.com/archives/snort/2000 -01/0220.html (14 Feb. 2004).

[23] Wesemann, Daniel. GCIA Practical. Pra ctical Detect 2. 24 March 2003. URL: http://www.giac.org/practic al/GCIA/Daniel_Wesemann_GCIA.pdf (12 Feb. 2004).

[24] Misc. authors. SANS Institute: "Help Defeat Denial of Service Atta cks: Step-by-Step" version 1.4. 23 March 2003. URL: http://www.sans.org/dosstep/index.php (14 Feb. 2004).

[25] TonikGin. "XDCC – An .EDU Admin's Nightmare". 11 Sep. 2002. URL: http://www.cs.rochester.edu/~bukys/host/tonikgin/EduHacking.html (14 Feb. 2004).

[26] Gnutella. URL: http://www.gnutella.com/ (17 Feb. 2004).

[27] Bearshare. URL: http://www.bearshare.com/ (17 Feb. 2004).

[28] Bit Torrent. URL: http://bitconjurer.org/BitTorrent/ (17 Feb. 2004).

[29] eDonkey. URL: http://www.edonkey2000.com/ (17 Feb. 2004).

[30] eMule. URL: http://www.emule-project.net/ (17 Feb. 2004).

[31] Battlefield 1942. URL: http://www.battlefield1942.com/ (17 Feb. 2004).

[32] OIT Security. Duke University. "Instructions on Cleaning IRC bot & backdoor: XDCC". Last updated 25 April 2002. URL: http://security.duke.edu/cleaning/xdcc.html (17 Feb. 2004).

[33] Srinivasan, R. et al. RCF 1831: "RPC: Remote Procedure Call Protocol Specification Version 2". August 1995. URL: http://www.ietf.org/rfc/rfc1831.txt (17 Feb. 2004).

[34] CERT® Advisory. CA-2001-33 "Multiple Vulnerabilities in WU-FTPD". Last updated 15 Feb. 2002. URL: http://www.cert.org/advisories/CA-2001-33.html (14 Feb. 2004).

[35] CERT® Advisory. CA-2001-26 "Nimda Worm". Last updated 25 Sep. 2001. URL: http://www.cert.org/advisories/CA-2001-26.html (15 Feb. 2004).

[36] CERT® Incident Note IN-2003-02 "W32/Mimail Virus". 2 August 2003. URL: http://www.cert.org/incident_notes/IN-2003-02.html (14 Feb. 2004).

[37] Mills, D.L. RCF 958: "Network Time Protocol (NTP)". September 1985. URL: http://www.ietf.org/rfc/rfc958.txt (17 Feb. 2004).

[38] WinVNC. URL: http://www.realvnc.com/winvnc.html (18 Feb. 2004).

[39] CERT® Incident Note IN-99-02. "Happy99.exe Trojan Horse". 29 March 1999. URL: http://www.cert.org/incident_notes/IN-99-02.html (21 Feb. 2004).

[40] CVE-2001-0414. "Buffer overflow in ntpd ntp daemon". 2 April 2003. URL: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0414 (20 Feb. 2004).

[41] Ramakrishnan, K. et al. RFC 3168: "The Addition of Explicit Congestion Notification (ECN) to IP". Sep. 2001. URL: http://www.ietf.org/rfc/rfc3168.txt (22 Feb. 2004).

[42] Fyodor. Nmap. URL: http://www.insecure.org/nmap (22 Feb. 2004).

[43] Queso. URL: http://ftp.cerias.purdue.edu/pub/tools/unix/scanners/queso/ (24 Feb. 2004).

[44] Zalewski, Michal. p0f v2.0.3. URL: http://lcamtuf.coredump.cx/p0f.shtml (24 Feb. 2004).

[45] Miller, Toby. "ECN and it's impact on Intrusion Detection". 1999 -2000. URL: http://www.sans.org/y2k/ecn.htm (26 Feb. 2004).

[46] Beardsley, Tod. GCIA Practical. 8 May 2002. URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc (26 Feb. 2004).

[47] Cormier, Andre. GCAI Practical. 2 May 2003. URL: http://www.giac.org/practical/GCIA/Andre_Cormier_GCIA.pdf (26 Feb. 2004).

Last Updated: May 29th, 2017

Upcoming Training

Security Operations Center Summit & Training	**Washington, DC**	**Jun 05, 2017 - Jun 12, 2017**	**Live Event**
SANS Houston 2017	**Houston, TX**	**Jun 05, 2017 - Jun 10, 2017**	**Live Event**
SANS Columbia, MD 2017	**Columbia, MD**	**Jun 26, 2017 - Jul 01, 2017**	**Live Event**
SANS London July 2017	**London, United Kingdom**	**Jul 03, 2017 - Jul 08, 2017**	**Live Event**
SANSFIRE 2017	**Washington, DC**	**Jul 22, 2017 - Jul 29, 2017**	**Live Event**
SANSFIRE 2017 - SEC503: Intrusion Detection In-Depth	**Washington, DC**	**Jul 24, 2017 - Jul 29, 2017**	**vLive**
SANS San Antonio 2017	**San Antonio, TX**	**Aug 06, 2017 - Aug 11, 2017**	**Live Event**
SANS Boston 2017	**Boston, MA**	**Aug 07, 2017 - Aug 12, 2017**	**Live Event**
SANS Virginia Beach 2017	**Virginia Beach, VA**	**Aug 21, 2017 - Sep 01, 2017**	**Live Event**
SANS Adelaide 2017	**Adelaide, Australia**	**Aug 21, 2017 - Aug 26, 2017**	**Live Event**
SANS Network Security 2017	**Las Vegas, NV**	**Sep 10, 2017 - Sep 17, 2017**	**Live Event**
SANS vLive - SEC503: Intrusion Detection In-Depth	**SEC503 - 201709,**	**Sep 11, 2017 - Oct 18, 2017**	**vLive**
SANS Baltimore Fall 2017	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
SANS London September 2017	**London, United Kingdom**	**Sep 25, 2017 - Sep 30, 2017**	**Live Event**
Baltimore September 2017 - SEC503: Intrusion Detection In-Depth	**Baltimore, MD**	**Sep 25, 2017 - Sep 30, 2017**	**vLive**
Community SANS Scottsdale SEC503	**Scottsdale, AZ**	**Oct 02, 2017 - Oct 07, 2017**	**Community SANS**
SANS October Singapore 2017	**Singapore, Singapore**	**Oct 09, 2017 - Oct 28, 2017**	**Live Event**
Community SANS Boston SEC503	**Boston, MA**	**Oct 09, 2017 - Oct 14, 2017**	**Community SANS**
Community SANS Ottawa SEC503	**Ottawa, ON**	**Oct 16, 2017 - Oct 21, 2017**	**Community SANS**
SANS Seattle 2017	**Seattle, WA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
SANS San Diego 2017	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**Live Event**
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	**San Diego, CA**	**Oct 30, 2017 - Nov 04, 2017**	**vLive**
SIEM & Tactical Analytics Summit & Training	**Scottsdale, AZ**	**Nov 28, 2017 - Dec 05, 2017**	**Live Event**
SANS Cyber Defense Initiative 2017	**Washington, DC**	**Dec 12, 2017 - Dec 19, 2017**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**