Global Information Assurance Certification Paper

GIAC Certified Intrusion Analyst (GCIA)
Practical Assignment
Version 4.0

Craig Bartels

Submitted: April 17, 2004

Executive Summary

Below are the findings of a detailed analysis of capture files taken from the University's network. This is a shorted discussion of the findings discussed in more detail below. The files used for these findings are 2002.10.10, 2002.10.11, and 2002.10.12. They show network traffic from October 10th to October 12th.

After careful analysis of the capture files provided there seems to be very little evidence to support the network having been compromised during this period. There does not appear to be any active scanning of the network during this timeframe as well. However, there is no way to determine if there was a successful attack prior to October 10th.

From the files provided a total of 1322 alerts were generated. The alerts were from a limited number of sources, only 49. Many of the alerts generated are from the same internal based host. In most cases these alerts are caused by the alerting system looking too closely at each packet and causing what is called a false positive alert. By removing these from the total alerts, only 315 alerts were generated. It is not unusual to see such a high number of alerts for just a few days. The internet is filled with packets caused by both virus and malicious users that generate similar packets. However both are hoping to stumble upon networks that have not taken the proper precautions to protect their systems.

It appears this local network is missing a few of these precautions. Based on the traffic seen, it appears there is no firewall in between the local network and the external network. This is a concern, as a firewall is the first line of a defense against the same attacks seen in the packet captures. A properly configured firewall would keep many of these attacks from ever reaching their intended target, and make this network less of a target for potential attackers. The routers in use on the network also look to have almost no restrictions in place.

My first recommendation would be implement a more restrictive policy on the edge firewall if one is in place, if not place a firewall at the edge of the network. This firewall should allow inbound only the service ports needed. If inbound services are needed, then the servers these services reside on should be placed into a DMZ. A DMZ is a network that is segmented off from the internal network. By segmenting the internal network off from all inbound traffic attackers have are less likely to gain access into the internal network. I would also recommend limiting all outbound traffic to only the services / ports needed by the university. As for internal routers, additional steps should be taken to limit RFC1918 space that is not being used. This type of traffic is not routed via the internet and does not look to be used on the local network. There looks to be no reason to stop this traffic from being routed on the local network. Without further information, I cannot make a recommendation on the status of the host based security. None of the hosts targeted looked to send an answer to the

attacking machine. However, best practice for host based systems would be to close any unnecessary ports on each host, especially those hosts that are running a service used by hosts on the internet. My last recommendation would be to expand the segments watched by the IDS / packet sniffer. And IDS can only perform as good as the information it gathers, in this case there is too little information to determine the level of security for the network. Additional IDS machines can only improve how well the IDS performs.

Detailed Analysis

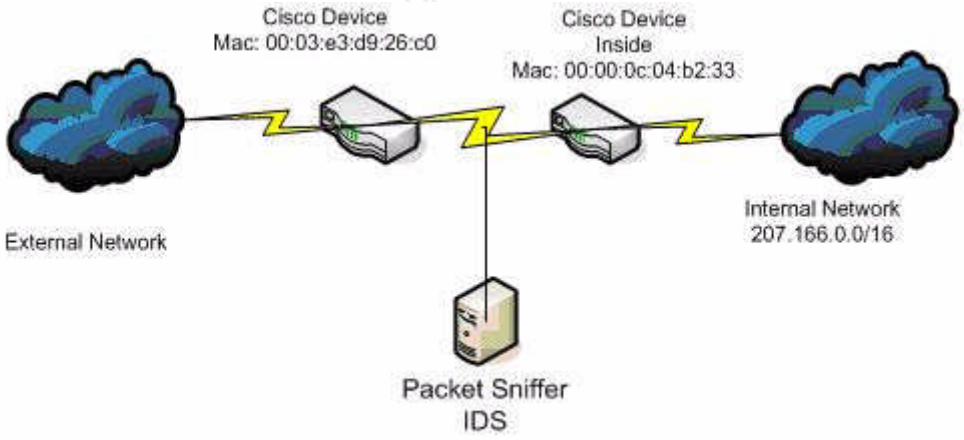
Detect #1 BackDoor Q access (CAN-1999-0660)

Source of detect:

The alert was generated from the raw pcap file 2002.10.12. The file was obtained from http://isc.sans.org/logs/raw as outlined in practical guidelines.

Network topology:

The network topology was not provided, however some information can be gathered from the packet captures. To get a better overall understanding of the local network a capture file was created by merging all the pcap files for the month of October (2002.10.1-2002.10.13) into a file named 2002.10.all. Analyzing this file showed the sensor used to create the capture file is sniffing traffic between two Cisco devices.



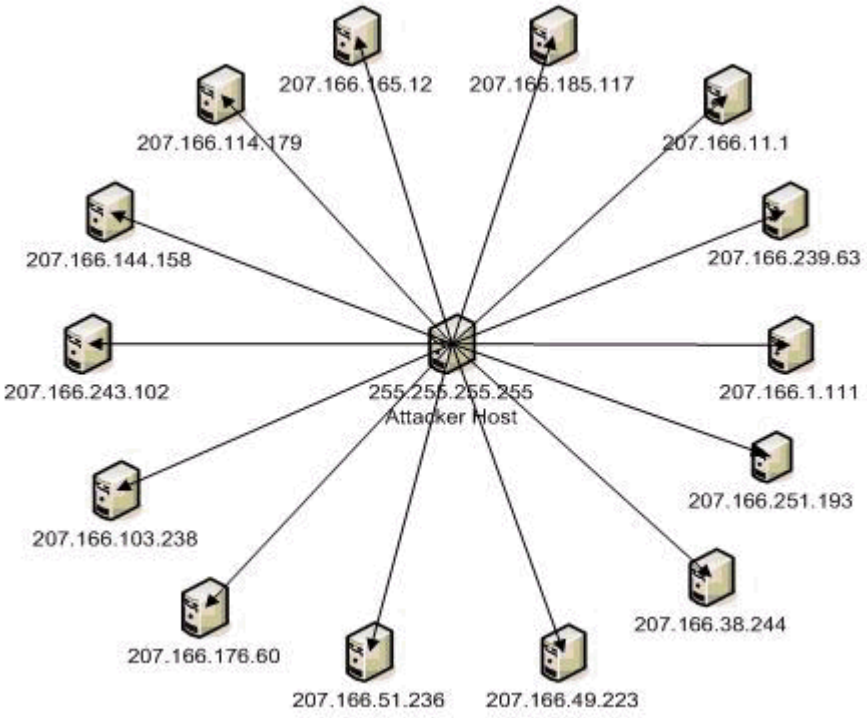
The source and destination mac addresses can be found using both ethereal and tcpdump. In ethereal (Version 0.10.8) choosing Statistics -> Endpoint List -> Ethernet will create a table showing all mac address found in the capture. The same information can be found with tcpdump using the –e flag as demonstrated below:

Tcpdump –e –nr 2002.10.all
Output:
01:53:45.836507 ***00:00:0c:04:b2:33*** *>* ***00:03:e3:d9:26:c0****, ethertype IPv4*

(0x0800), length 2974: IP 207.166.87.157.62559 > 64.12.137.56.80: P 0:2920(2920) ack 3346149 win 24820

Using the mac address header (the first 6 characters found in address) it is possible to determine the manufacturer of the device. Each manufacturer is assigned a specific range they use for the mac addresses header of all the devices they build. By looking up these ranges on websites such as http://www.coffer.com/mac_find, it can be determined that both of these mac address are associated with devices manufactured by Cisco Systems. Based on this information, the packet captures provided were taken from between two Cisco devices. Unfortunately, there is no way to determine if this is default path for all traffic on the inside network, or what the purpose of this connection might be. Based on the network traffic analyzed for this paper an assumption has been made this is indeed the default route to the internet for all traffic on the inside network.

Link graph:



Above is the link graph for the detect detailed below. Only a sample of hosts where chosen based on the size this graph would require to include all hosts receiving the suspicious packet.

Description of detect:

” Q is a primarily Unix-based remote-access tool that provides stealth capabilities to make it's presence less obvious both on the host, and in network

traffic, "as described by Les Gordon (http://www.sans.org/resources/idfaq/qtrojan.php). Many of the descriptions for the Q Trojan are learned from Mr. Gordon's paper. The purpose of the Q Trojan is to allow a remote user to execute commands on remote hosts as root. The program acts much like netcat in that it does not require a typical tcp session to be established. "Newer versions of the Q Trojan allows the attacker to assign the source IP or to choose the source IP randomly, also randomly choose the source port number and initial TTL (>=200). This Trojan also chooses what protocol to use randomly between TCP, ICMP, and UDP" (Gordon) The corresponding CVE is CAN-1999-0660 found at http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660 and is currently listed as "Under Review". Other descriptions of the Q Trojan can be found at http://www.whitehats.com/info/IDS203.

Reason detect was selected:

The evidence of Q traffic suggests local machines may have been compromised. The Q Trojan allows an attacker to remotely control UNIX based machines to further their infiltration into the local network, garner information off the local machine and use the host as a tool to attack other networks. As this tool is not used as for reconnaissance but as a control tool for a compromised machine, it was chosen for further investigation.

Detect was generated by:

This detect was generated by Snort Version 2.2.0 (Build 30), all rules have been enabled.
The command run was:

snort -k none -r 2002.10.12 -c /etc/snort/snort.conf -l /home/giac1/logs/

Flags used:

"-k none" flag was used to ignore checksum verification in the pcap file as the files have been changed
"-r $path" tells snort which pcap file to use as input.
"-c $path" gives the path to the snort configuration file
"-l $path" gives snort the location to place the log files.

The alert:

[**] [1:184:6] BACKDOOR Q access [**]
[Classification: Misc activity] [Priority: 3]
11/11-17:26:59.016507 255.255.255.255:31337 -> 207.166.51.236:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
****A*R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20*
[Xref => http://www.whitehats.com/info/IDS203]

The signature that triggered the attack is:

alert tcp 255.255.255.0/24 any -> $HOME_NET any (msg:"BACKDOOR

Q access"; flow:stateless; dsize:>1; flags:A+; reference:arachnids,203; classtype:misc-activity; sid:184; rev:7;)

The alert triggers when the following events occur in a TCP packet:

Any source IP within the ranges of 255.255.255.0-255 (255.255.255.0/24).
The packet is considered valid regardless of stream state (flow:stateless).
The payload must be greater than 1 byte (dsize:>1).
Lastly the ACK flag must be set plus any others.

For further reading and additional definitions for reading snort rules can be found in the snort documentation under writing rules.

The alert also gives the following information.

*[**] [**1:184:6**] BACKDOOR Q access [**]*
[Classification: Misc activity] [Priority: 3]
11/11-17:26:59.016507 255.255.255.255:31337 -> 207.166.51.236:515
TCP TTL:15 TOS:0x0 ID:0 IpLen:20 DgmLen:43
****A*R** Seq: 0x0 Ack: 0x0 Win: 0x0 TcpLen: 20*
[Xref => http://www.whitehats.com/info/IDS203]

The first number 1 (highlighted above) is actually the generatorid. This value is found in the gen-msg.map in the ./snort/ directory. This value tells the users what part of snort generated the alert, be it a general alert, tagged packet, or an alert from the preprocessor. SID 184 is the alert triggered (found in both backdoor.rules and sid-msg.map. The SID is the Snort Signature ID and each alert is assigned and ID. This is also the 6th revision of this particular rule as denoted by the 6 in 1:184:6. This usually implies that this rule has been improved over previous rules. This alert has been given a Priority of 3, this field can be edited at the engineer's discretion. The purpose in changing the priority would be to rank more import alerts higher based on the local architecture. Lastly we are given the arachNIDS number as a reference, IDS203.

One of the packets to generate the alert is shown below:

15:32:19.606507 IP (tos 0x0, ttl 15, id 0, offset 0, flags [none], length: 43, bad cksum 4718 (->fccd)!) 255.255.255.255.31337 > 207.166.223.89.515: R [bad tcp cksum fc3f (->b1f5)!] 0:3(3) ack 0 win 0 [RST cko]
(note: the bad cksum errors are caused by the original packet having been altered)

Snort found 44 alerts for 2002.10.12 and for each alert a new the only thing to change for each packet was the destination IP. 44 different IP were targeted with this packet. Each packet has a source of 255.255.255.255 with a source port of 31337. The destination address changes, however the destination port 515 (typically used for UNIX printing) does not change. Each packet has the

ACK and RST flags set. There does not appear to be a trend based on the time each scan was started.

Probability the source was spoofed:

The probability the source was spoofed is high. 255.255.255.255 is used for network broadcast traffic. As stated in RFC 919 "The address 255.255.255.255 denotes a broadcast on a local hardware network, which must not be forwarded. This address may be used, for example, by hosts that do not know their network number and are asking some server for it." (http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc0919.html) There is some question if the Q Trojan would use the broadcast address to respond the packets received via broadcast. To validate exactly how the Q Trojan would react to a source address of 255.255.255.255, a test environment was created. Q was set up to listen on port 515 on the victim machine. A Q packet was sent to the victim machine with a source address of 255.255.255.255 to port 515, the victim machine did not respond to this packet.

Attack mechanism

The packets snort found to generate this alert have many things in common. All packets have a source IP of 255.255.255.255 with a source port of 31337. They also contain the same destination port of 515 typically used for UNIX printing, targeting various local networks all within the 207.166.0.0/16 network. As described above this alone does not trigger the alert the packets also have the ACK flag set. It is interesting to note that each packet has the RST flag set and all have the same payload of “CKO”

As the source IP of all the packets is 255.255.255.255 we can assume this packet is might be used as the control packet. Meaning the attacker may be using this as a means to find hosts that have been previously compromised with the Q Trojan. If CKO is indeed a control packet the victim host may respond to a predetermined IP. An inference can also be made that the same machine is being used to create the packets as each packet is identical to each other. This includes a TTL of 15 and a starting sequence number of 0. The packets also share the same payload of CKO. It is important to note that “client/server versions of the Q Trojan are not cross-compatible by default” (Meyer, Amanda p.23). This would significantly reduce the possibility of an attacker trying to probe the network for victim hosts the attacker did not install.

The payload of CKO by itself does not give us much information, this could be encrypted traffic or a specific command that the compromised host may understand or the user is just learning the tool and has miss configured the packet being sent. However as the RST flag is also set made these packets more interesting. Further searching on the web provided more information; Dustin Decker found that a SonicWall firewall may have generated this traffic. The SonicOS will send various types of resets; one of the reset codes is “CKO” which matches the payload found in our capture (Decker,

http://www.ethereal.com/lists/ethereal-users/200409/msg00057.html)

To determine if these packets are Q control packets, or reset packets from a firewall or something else altogether it is necessary to gather more data. For the packets to be a reset from a firewall an established session should be seen in previous captures. Looking at the merged capture file 2002.10.all sorting on just the IP targeted in the alerts there are no signs that any host had any active connections that traversed the sensor. There is also no evidence that any of the hosts targeted in the alerts have responded to any possible Q Trojan control packets. Once again we are limited to packets generated at the sensor and do not know if there are other network routes to the internet machines on the internal network can utilize. A better understanding of the architecture of the network is needed. There is also insufficient traffic to garner the type the OS of the target machines, as only UNIX systems are susceptible to the Q Trojan.

Correlations

Les Gordon has written an excellent paper on the Q Trojan (http://www.sans.org/resources/idfaq/qtrojan.php). Detailed in this paper are many packet captures from various versions of the Q Trojan. Fortunately there are no response packets within our capture to correlate to packets contained within the paper. This paper was used to further my knowledge of the Q Trojan as source for many of the conclusions for this detect.

There is a CVE for the Q Trojan CAN-1999-0660 (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660). There was very little information included in the CVE, however the CVE was useful in further web searches.

Amanda Meyer's paper was used to verify the findings for this detect (http://www.giac.org/practical/GCIA/Amanda_Meyer_GCIA.pdf). Within her paper she mentions that different client/server versions of the Q Trojan will not function with each other. I had originally overlooked this in my original evaluation of this detect.

Dustin Decker's online post was highly informational as he determined that the payload of CKO (found in our packets) may actually be a reset from a SonicWall firewall (http://www.ethereal.com/lists/ethereal-users/200409/msg00057.html). Without further information about the local network there is no way to validate this finding.

Evidence of active targeting:

Looking that the merged pcap file 2002.10.all it is evident that this is not active targeting but rather a random scan of the network. No host is targeted more than once, and hosts seem to be chosen at random.

Severity

Severity = (Criticality + Lethality)(System Countermeasures + Network Countermeasures) (*note: in V4.0 it does not appear that the two are subtracted from each other but rather multiplied, this is different than other practical submissions*)
64 = (3+5)(3+5)

Criticality --3
As there is not much known about the type of systems being targeted a rank of 3 would seem appropriate. If the systems being targeted are solely windows boxes or student machines sitting in a DMZ at the university this number would be lower. If the systems are UNIX based systems or critical servers this number would increase.
Lethality --5
If the boxes have indeed been compromised the attacker has full control of the compromised host(s). The attacker would have unrestricted access on both the system and network a rank of 5 is required.
System Countermeasures --3
There is no evidence that any host has been compromised however there is no information if there has been any attempt to lock down the systems in question or any system within the network. The victim hosts did not respond to this attack based on our capture; however that does not mean they did not have a predetermined response to a host on the local network based on the CKO payload. A rank of 3 is practical till more information can be gathered.
Network Countermeasures --5
Again there is no information has been given in regards to the network architecture. There is no evidence of network acls being placed on the network devices or any evidence of a network firewall as both are allowing the broadcast address to be used. Based solely on this capture there does not seem to be much filtering of IP's or ports on the Cisco devices in this environment, there also does not appear to be an active firewall. Based on this observation a value of 5 is given.

Defensive Recommendation

All though we are uncertain of the network topology, some assumptions can be made. Both of the Cisco devices noted in the network diagram should be reconfigured to be more restrictive. Acl's should be places on the Cisco devices to restrict RFC 1918 and RFC 3171 reserved IP space unless needed for network connectivity. Edge routers and firewalls should also be configured not to route this same network space based on both source and destination. In this case port 515 is being targeted; this port is associated with Unix Printing. Based on the location of the IDS there does not appear to be a need to allow printing across this network. It would also seem prudent to limit all inbound traffic to only the ports needed for any services being offered on the internal segment.

Detect #2 BAD-TRAFFIC tcp port 0 traffic

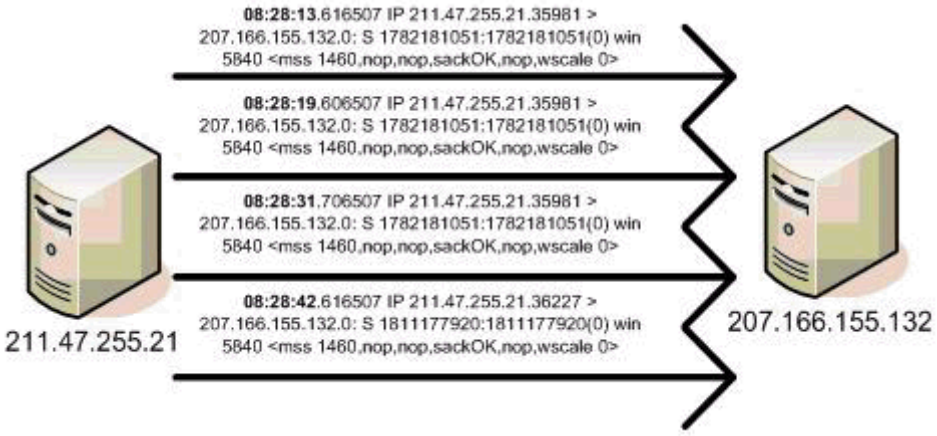
Source of trace:

The alert was generated from the raw pcap file 2002.10.11. The file was obtained from http://isc.sans.org/logs/raw as outlined in practical guidelines.

Network topology:

The network topology is the same as the topology discussed for Detect #1 BackDoor Q access (CAN-1999-0660)

Link graph:



The detect above shows one of the eight connection attempts made for this detect. This represents the time interval between each attempt outlined below.

Description of detect:

This detect is a series (29) of attempts to connect to port 0 on two internal hosts. Each packet is attempting to establish a tcp connection with a SYN. This is unusual traffic in that the destination port is port 0, a port not used in normal tcp traffic. What makes this detect even more unusual is that 29 alerts were generated for only 2 sources IP's trying to connect to only 2 destination IP's. Using tcpdump to single out the attacker IP's 211.47.255.22 and 211.47.255.21 showed an interesting increment for the scans.

```
~/working/4 $ tcpdump -r 2002.10.11 host 211.47.255.22
18:00:18.616507 IP 211.47.255.22.60086 > 207.166.237.132.0: S
2614277515:2614277515(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0>
18:00:21.616507 IP 211.47.255.22.60086 > 207.166.237.132.0: S
2614277515:2614277515(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0>
18:00:27.616507 IP 211.47.255.22.60086 > 207.166.237.132.0: S
2614277515:2614277515(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0>
18:00:39.616507 IP 211.47.255.22.60086 > 207.166.237.132.0: S
2614277515:2614277515(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0>
```

Looking at the numbers in bold we notice typical tcp behavior for a session that is timing out. At 18:00:18 we see the first packet, at 18:00:21, 3 seconds later we see the same packet, again at 18:00:27, 6 seconds later we see the same packet, and finally at 18:00:39, 12 seconds later we see the final attempt. This is similar behavior for normal tcp traffic that is unable to make a connection to the port specified, it will try 4 times at an interval of 0,3,6.and 12 seconds apart. Yet another unusual part of this detect is that in the same capture as above the IP ID does not change but rather stays the same with a value in this example of 616507. This is not normal behavior. Below is an example of proper behavior, the IP add (in bold) in increased with each attempt.

*20:01:32.**318706** IP 192.168.2.33.44582 > 10.2.22.22.23: S 4126933886:4126933886(0)*
*20:01:35.**318209** IP 192.168.2.33.44582 > 10.2.22.22.23: S 4126933886:4126933886(0)*
*20:01:41.**317297** IP 192.168.2.33.44582 > 10.2.22.22.23: S 4126933886:4126933886(0) 20:01:53.**315474** IP 192.168.2.33.44582 > 10.2.22.22.23: S 4126933886:4126933886(0)*

The last piece of this detect that should be examined would be the source port number of each packet The source port only changes after the previous packet looks to have timed out. By looking at the source port number we can quickly gather that the attacker attempted to connect to each target host 207.166.155.132 and 207.166.237.132 for a total of 4 times each. So the detect of 29 alerts is actually 8 separate tcp SYN connects. The only reasoning for the IP ID not to change would be that the attacker crafted each originating packet by hand. It can also be inferred that these packets have been crafted for a purpose. The snort alert references snort SID 1:524 which describes this alert as “possible reconnaissance activity” (Snort, http://www.snort.org/snort-db/sid.html?sid=524). In a successful attempt the attacker would expect to see an ACK or in other words a response to the SYN as per a typical 3 way tcpip handshake. This would give the attacker knowledge that there is a machine listening at this IP, the attacker can also garner the type of OS based on its response. Sorting the merged capture file 2002.10.for all traffic from 211.47.255.21 and 211.47.255.22, no internal host responded.

Reason detect was selected:

This detect was chosen due in part to the curious use of port 0. Under normal circumstances this port is not used, and should be investigated.

Detect was generated by:

This detect was generated by Snort Version 2.2.0 (Build 30), all rules have been enabled.
The command run was:

snort -k none -r 2002.10.11 -c /etc/snort/snort.conf -l /home/giac/logs/

Explanations of the various switches used are explained above.

The alert:

[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
11/10-18:00:18.616507 211.47.255.22:60086 -> 207.166.237.132:0
TCP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
*******S* Seq: 0x9BD2B58B Ack: 0x0 Win: 0x16D0 TcpLen: 32*
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 0

The signature that triggered the attack is:

alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD-TRAFFIC tcp port 0 traffic"; flow:stateless; classtype:misc-activity; sid:524; rev:8;)

The alert triggers when the following events occur in a TCP packet:

Packet must be a TCP packet.
Destination port must be 0.
The packet is considered valid regardless of stream state (flow:stateless).

The SID (Snort Signature ID) for this alert is 524, and has been revised 8 times. This alert has a default priority of 3. The SID from snort can be found at: http://www.snort.org/snort-db/sid.html?id=524. There are no arachNIDS or BUGTRAQ references for this alert.

Probability the source was spoofed:

The probability that the source IP's have been spoofed is very low. For the attacker to gain the information they are seeking, the attack requires the victim host to respond to the SYN sent by the attacker's machine with a SYN, ACK. If the address in spoofed this information would not reach the attacker but rather the spoofed address. For the attacker to see the victim hosts SYN, ACK sent back to the spoofed host, the attacker would need to be able to sniff packets on a host on the return path of the packet.

Attack mechanism:

The attacker is sending specially crafted packets into the local network in hopes of receiving a response. With this response the attacker can identify that the host in live and attempt to determine the OS of the victim machine. This would allow our attacker to learn which hosts to target for further probing and filter down the vulnerabilities to use for each host.

Correlations

Saro Hayan used a similar detect in his GIAC practical (http://www.giac.org/practical/GCIA/Saro_Hayan_GCIA.pdf). Mr. Hayan's findings were very similar to my own.

Evidence of active targeting

Using the merged pcap file 2002.10.all, it is difficult to determine a pattern in the attackers failed network probes. The attacker has yet to find an active host within the network as well. At this time the scans look to be random, which would mean the attacker is not currently targeting this network. Once the attacker stumbles upon an active host to their liking this may change and should be monitored.

Severity

Severity = (Criticality + Lethality)(System Countermeasures + Network Countermeasures)
30 = (3+2)(1+5)

Criticality --3
As there is not much known about the type of systems being targeted a rank of 3 would seem appropriate. The attacker's choice of IP's looks to be random and is not actively targeting any one type of machine.

Lethality --2
At this time the attacker is simply probing each IP at random. Depending on the intention of the attacker this value could change however probing does not cause damage to the target machine. At this time assigning a value of 2 seems adequate.

System Countermeasures --1
It is very difficult to asses the system countermeasures without knowing if the hosts being probed were on the network. As the systems did not respond to the probes it appears they may have some sort of host based firewall. It is typical for a machine to respond to this type of probing under normal circumstances. However, it may be that there was no machine with the host IP's assigned. In either event, the host existent or not has some countermeasures in place.

Network Countermeasures --5
Again there is no information has been given in regards to the network architecture. However if a firewall is in place, it seems very unusual that port 0 would be allowed past. A rank of 5 is given.

Defensive Recommendation

A few simple countermeasures can be taken to aid in keeping erroneous traffic as this from getting into the network. The firewalls should be configured to allow only ports required by the hosts with the service. ICMP traffic (ping) should also be configured to be blocked for all hosts where it is not needed. Servers offering up services to the world may need this allowed however any host without a service running on it

Detect #3 WEBROOT DIRECTORY TRAVERSAL

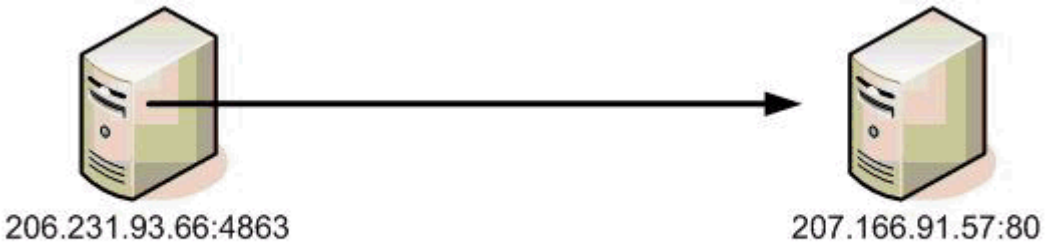
Source of trace:

The alert was generated from the raw pcap file 2002.10.11. The file was obtained from http://isc.sans.org/logs/raw as outlined in practical guidelines.

Network topology:

The network topology is the same as the topology discussed for Detect #1 BackDoor Q access (CAN-1999-0660)

Link graph:



The graph above shows the single host 206.231.93.66 trying to connect with a source port of 4863 to victim host 207.166.91.57 on port 80.

Description of detect:

This detect looks to be an attempt by a tool or script to gain access to an IIS server using a well known exploit to unpatched systems. In this capture there are only two alerts for the day. Both have a source IP of 206.231.93.66 with a destination address of 207.166.91.57. It should be noted that the attacks are 3 seconds apart which could mean it was one attack but simply a typical retry as the attacking host did not get a response to the initial request. The victim host in this case did not look to be active as it did not respond to the attack. There is also no evidence found of this host being active on the network for the month of October as well. Interestingly, the attacking IP is only seen in the merged pcap file 2002.10.all twice, the packets generating the alert. A whois was performed at http://samespade.org and resulted in the following results:

OrgName: JAMES SEXTON
OrgID: JAMESS-5
Address: 5140 SOUTH 450 EAST
City: PERU
StateProv: IN
PostalCode: 46970
Country: US
NetRange: 206.231.93.0 - 206.231.93.127
CIDR: 206.231.93.0/25
NetName: FON-347126912094242
NetHandle: NET-206-231-93-0-1

Parent: NET-206-228-0-0-1
NetType: Reassigned
Comment:
RegDate: 2002-02-14
Updated: 2002-02-14
TechHandle: JS3208-ARIN
TechName: SEXTON JAMES
TechPhone: 1-765-473-9695
TechEmail: CNK30@hotmail.com

Oddly this address range is assigned to a single person James Sexton, however it appears he did own this IP range during the same timeframe of the alert.

Reason detect was selected:

The reason this detect was chosen was based on the implications of a successful attack. As I will explain below, if this attack is successful the attacker could assume control of the victim machine. Without knowing the local network environment the assumption must be made that there are vulnerable machines within the local network and should be investigated. The second reason for choosing this attack was based on the type of alert; this alert was generated with Snort's preprocessor. My understanding of the Snort's preprocessors was limited and this alert gave me the opportunity to dig deeper into how these types of alerts are generated.

Detect was generated by:

This detect was generated by Snort Version 2.2.0 (Build 30), all rules have been enabled.
The command run was:

snort -k none -r 2002.10.11 -c /etc/snort/snort.conf -l /home/giac/logs/

Explanations of the various switches used are explained above.

The alert:

[**] [119:18:1] (http_inspect) WEBROOT DIRECTORY TRAVERSAL [**]
11/10-01:28:28.196507 206.231.93.66:4863 -> 207.166.91.57:80
TCP TTL:106 TOS:0x0 ID:26945 IpLen:20 DgmLen:185 DF
****AP*** Seq: 0x8BCFF5D6 Ack: 0xFEEFAE66 Win: 0x4470 TcpLen: 20*

There are two parts to this signature the first is found below:

*preprocessor http_inspect: global *
iis_unicode_map unicode.map 1252

The preprocessors allow snort to manipulate incoming traffic prior to the packets being handed off to the detection engine. In this case the preprocessor will try to normalize all http packets. By normalizing the packet the preprocessor

attempts to reformat each http packet to a standard format. This means the preprocessor will attempt to decode an http packet based on the unicode.map file. In the case of this detect, the unicode.map will see the %5c and decode this to be \. By normalizing the packet to a standard format fewer rules need to be created and snort is able to analyze packets quicker.

The alert is generated by a second preprocessor:

*preprocessor http_inspect_server: server default *
profile all ports { 80 8080 8180 } oversize_dir_length 500

The configuration string is enabling the preprocessor to look at http packets, with a default server type (types include, IIS and Apache). The default profile has all profiles enabled which includes a setting for webroot traversals. All packets are examined for ports 80, 8080, and 8180. It is important to note the preprocessor is unable to monitor port 433 or https traffic as the traffic is encrypted.

The alert also gives us some additional information about how the preprocessors generate the alert.

*[**] [**119:18:1**] (http_inspect) WEBROOT DIRECTORY TRAVERSAL [**]*
11/10-01:28:28.196507 206.231.93.66:4863 -> 207.166.91.57:80
TCP TTL:106 TOS:0x0 ID:26945 IpLen:20 DgmLen:185 DF
****AP*** Seq: 0x8BCFF5D6 Ack: 0xFEEFAE66 Win: 0x4470 TcpLen: 20*

A typical alert is generated based on the rules found in the ./snort/*.rules file. This file tells snort the SID, Rev, Priority, Name, and some additional information covered above. For alerts generated by the preprocessors they do not use these same files, but instead use a file called gen-msg.map. The gen-msg.map contains the basic alert information similar to normal snort rule files. To use our alert as an example, the 119 correlates to the generator that generated the alert, in this case http_inspect. All rules being triggered by an http_inspect rule will have this same number. The next number is 18, this is the alertid for this alert. Looking at a portion of the gen-msg.map file this becomes clearer.

 # Format: generatorid || alertid || MSG
1 || 1 || snort general alert
2 || 1 || tag: Tagged Packet
(snip)
119 || 16 || http_inspect: OVERSIZE CHUNK ENCODING
119 || 17 || http_inspect: UNAUTHORIZED PROXY USE DETECTED
119 || 18 || http_inspect: WEBROOT DIRECTORY TRAVERSAL
120 || 1 || http_inspect: ANOMALOUS HTTP SERVER ON UNDEFINED HTTP PORT
121 || 1 || flow-portscan: Fixed Scale Scanner Limit Exceeded
121 || 2 || flow-portscan: Sliding Scale Scanner Limit Exceeded

121 || 3 || flow-portscan: Fixed Scale Talker Limit Exceeded
(snip)

Probability the source was spoofed:

It is highly unlikely that the source IP is spoofed. For this attack to be successful the attacking hosts requires a fully established tcp handshake to occur. The attacking machine sends the exploit to the victim machine, if successful the victim machine will send the results of the exploit back to the attacking machine. If this address is spoofed the attacker will not receive the results of the exploit, the packets will instead be sent to the real owner of the spoofed IP. There is always the possibility that the attacker has control of a device capable of sniffing the traffic between the spoofed address and the victim host. This would allow the attacker to receive the results of the exploit by simply sniffing the traffic as it passes on the network.

Attack mechanism

The attack attempts to gain access by tricking the web server into executing a program outside of the web directory structure. In this case the attacker is trying to run: /winnt/system32/cmd.exe?/c+dir which would give the attacker a listing of all the files in the directory. The attacker tricks the web server by using a well known exploit of IIS (Internet Information Services), Microsoft's web server. The attacker crafts a custom GET request; within this request are specially encoded commands. The commands are encoded with standard UTF-8 Unicode that the web server will decode automatically. In the case of this packet the attacker is sending the following sequence:

..%5c../..%5c../..%5c/..55../..c1..

This decodes as:

..\../..\../..\/..55../..c1..

The web server will try to follow this link which on most vulnerable servers will place the attacker at the root of the system C:\. From this location the attacker attempts to run:

/winnt/system32/cmd.exe?/c+dir

By running cmd.exe the user will get a file listing of the C:\ directory. This output gives the attacker the knowledge the server he has targeted is susceptible to further exploits, though by itself is not harmful. The ..55.. and ..c1.. look be errors as neither would accomplish any added value as they are. The ..55.. is probably a mistype for ..%5... And the ..c1.. is also a mistype and should be ..%c1.. which decoded with the Chinese u Unicode as found by Ernest Eustace in his GIAC practical (Eustace P.30). Further investigation of the merged pcap file 2002.10.all there seems to be no evidence that this attack was successful. Beyond the 2 packets sent from 206.231.93.66 there is no further communication from either the source or destination IP. This attack could be an early variant of the rather widespread virus Nimda. Nimda utilizes the same exploit to gain access to vulnerable IIS. Nimda typically encodes its

payload twice. In our packet the \ is encoded as %5c, Nimda typically encodes this one more time to %255. Nimda typically tries to connect too many hosts at a time. In this case we see no other attempts from this host for the month of October. For these reasons ruling out these packets as Nimda seems logical. The packets could be web vulnerability tools such as nickto or Nessus. In looking at captures for the month of October we do see this same attack executed from other hosts, albeit unsuccessfully. This leads me to believe that this attack is part of a probing/ vulnerability tool.

Correlations:

As cited above, Ernest Eustace's paper was used as a reference (http://www.giac.org/practical/GCIA/Ernest_Eustace_GCIA.doc). Mr. Eustace detailed the correlations between the packet he analyzed and a typical Nimda packet more so than done in this paper. Mr. Eustace noted the use of "Connnection: close" in the packet. From his research he determined that the misspelling of the word "Connnection" was a known signature of Nimda. Mr. Eustace also noted the use of the "Host: www" which also a known signature of Nimda. However, he also came to the conclusion that this capture is not Nimda.

Jeremy Junginger posted a question on Insecure.org requesting more information on a web traversal attack against his web servers. The attack payload was near identical to the one found in our alert. Nick FitzGerald responded to Mr. Junginger's question, validating much of the information provided above (http://seclists.org/lists/incidents/2002/Nov/0076.html). Mr. Fitzgerald also believes this scan to be that of a "Unicode vulnerability scanner". However, his response was used to help validate the conclusions found in this paper.

Tom Rodriguez has written a rather extensive paper on "What are Unicode vulnerabilities on Internet Information Server (IIS)?". This paper can be found on the Sans.org website at: http://www.sans.org/resources/idfaq/iis_unicode.php. Mr. Rodriguez goes into far more detail on how the directory traversal works and additional ways this vulnerability can be utilized. This paper was also used to validate the conclusions found above.

Evidence of active targeting

Based on this alert alone, there is no evidence of active targeting. There were only two packets for the month of October from this source IP, and they were directed at only one host. The victim host does not look to be an active server as we have not seen packets from this host during the month of October. The 2nd packet also seemed to be a retransmitted of the first packet as it occurred 3 seconds after the original packet.

Severity

Severity = (Criticality + Lethality)(System Countermeasures + Network Countermeasures)
48 = (5+1)(3+5)

Criticality --3
As there is not much known about the type of systems being targeted a rank of 3 would seem appropriate. Depending on the type of environment the ids is watching this value would increase. It is known the captures are from a university, if the captures are from a subnet only accessed by students, computer lab, dorm room network access, this value would decrease as these systems would not be critical to the university. However if the university web servers or servers with student records are within this local network this number would increase. More information about the local network is needed to judge an acceptable value.

Lethality --5
Had this attack succeed the attacker may have been able to gain root level access to this machine. The value for Lethality if the attack were indeed successful would need to placed at 5.

System Countermeasures --1
From the logs given, it looks as though this victim is not susceptible to this type of attack or does not have a web server installed. The attack will only work against unpatched IIS servers and as the attack was unsuccessful it would seem appropriate to give this a value of 1.

Network Countermeasures --5
Assuming the packets captured are not from in front of the universities firewall, it appears as though there is a very liberal rulebase on the Cisco devices as well as any firewall if any. Without more information, it would seem prudent to rate the network countermeasures with a high score of 5.

Defensive Recommendation:

It would be good practice to audit all IIS servers within the environment to validate they have the proper patches applied. An added step would be to change the default path web files are stored on all machines. Many of the web traversal exploits take advantage of default installs where the starting directories are known. By renaming these directories the attacker can no longer rely on basic scripts to gain access to the host. Additional steps should be taken to limit both inbound and outbound traffic. An edge firewall should be put in place, and all unused inbound ports should be blocked at the edge to limit traffic such as this. All outbound traffic should be limited to only what is needed as well. By limiting outbound traffic with proxy servers or a firewall with only a few select ports open, this limits the avenues attackers have exploit vulnerable machines.

Network Statistics

Top Talkers.

Rank	*Total # Alerts*	*Source IP*	*# Signatures triggered*	*Destinations involved*
rank #1	1007 alerts	207.166.87.157	6 signatures	(48 destination IPs)
rank #2	132 alerts	255.255.255.255	1 signatures	(132 destination IPs)
rank #3	36 alerts	192.77.15.39	1 signatures	207.166.87.40
rank #4	32 alerts	211.47.255.20	1 signatures	207.166.184.92, 207.166.93.224
	32 alerts	211.47.255.22	1 signatures	207.166.237.132, 207.166.12.203

Table 1a.

Table 1a above is a list as compiled by snortsnarf of the top 5 Source IP's generating Alerts for the dates 10.10.2002 through 10.12.2002. To generate the table above, the pcap files 2002.10.10 through 2002.10.12 were merged together using ethereal to create one file. This file was then analyzed using snort to generate a single alert file. Snortnarf is a freely available perl script that will analyze a snort alert file and create a set of HTML pages that aid in sorting through a large number of alerts quickly. The program also creates three summary pages: list of signatures generated, top 20 source IP's and lastly the top 20 destination IP's.

The top 10 sources generating alerts was chosen as the criteria for "Top Talkers" because these IP's are causing the most load on the IDS sensors. In most cases the hosts generating the most alerts is not an attacker but rather is an opportunity to tune the IDS better as these alerts are typically false positives. In this case we see that an internal host is generating far more events than all the other IP's combined. Taking a closer look at each of the alerts, each of the 1007 alerts were http_inspect alerts generated by the preprocessor. In large environments the top offending source IP's generally are alerting on poorly constructed rules or IDS that is not configured properly for the environment it is watching. A brief scan of the traffic in this capture shows the end user surfing to a few sites that use Unicode within the URL. Correlating that back to the alerts being generated we see many of the alerts are trigged due to the use of Unicode in the URL. The snort.conf file used to generate this alert file was not tuned, rather had every rule enabled, and the default setting for preprocessor settings.

Top Five Targeted Ports

Signature	# Alerts	# Sources	# Dests
(http_inspect) WEBROOT DIRECTORY TRAVERSAL	10	2	10
BACKDOOR Q access [sid] [arachNIDS]	132	1	132
BAD-TRAFFIC tcp port 0 traffic [sid]	93	4	6
(snort_decoder) WARNING: TCP Data Offset is less than 5!	8	4	8
(snort_decoder): Short UDP packet, length field > payload length	1	1	1

Table 2a

Table 2a above shows what I believe to be the top five targeted services or ports. The top three are the three detects I choose to examine as they are what I believe to be the more important alerts. The top two alerts were chosen because; if successful it is possible for the attacker to assume control of the server. The third alert is important as this it is not common to see port 0 being used in normal traffic; this could be an attempt to probe for active hosts. The second to last attack made the list because a large number of sources generated the same alert. The last alert is interesting as there is very little UDP traffic in the capture files and I found it interesting that an alert was generated. Based only on the three days of captures, it is difficult to determine what service is truly being targeted. However based on the table above, IIS web servers are being probed for more than the other services.

Top 3 external source IP's

The first IP I choose to look closer into was 192.77.15.39. I choose this IP as it was the 3rd highest IP on the top source IP's with alerts found in table 1a above. The alert generated by the IP 192.77.15.39 is a preprocessor http_inspect alert:

> *[**] [119:13:1] (http_inspect) NON-RFC HTTP DELIMITER [**]*
> *11/10-16:15:01.266507 192.77.15.39:50656 -> 207.166.87.40:80*
> *TCP TTL:239 TOS:0x0 ID:51883 IpLen:20 DgmLen:115 DF*
> ****AP*** Seq: 0x72F035C0 Ack: 0xE8B02201 Win: 0x2238 TcpLen: 20.*

The source IP generated 132 alerts all with the same destination IP of 207.166.87.40 to port 80 (http). A quick scan of the traffic generating the logs shows what looks to be normal web traffic. The source hosts appears to be attempting to access web pages on the destination host, however there does not appear to be an active web server on the destination host. The IDS could be tuned to ignore this type of traffic by changing the default setting of the http_inspect preprocessor not to include the non_rfc_char setting.
To determine more about the host itself I conducted a whois search from http://www.samspade.org/. The output of this query is below:

```
OrgName:    Information Handling Services
OrgID:      IHS-7
Address:    15 Inverness Way East
City:       Englewood
StateProv:  CO
PostalCode: 80112
Country:    US
NetRange:   192.77.15.0 - 192.77.15.255
CIDR:       192.77.15.0/24
NetName:    IHSNET
NetHandle:  NET-192-77-15-0-1
Parent:     NET-192-0-0-0-0
NetType:    Direct Assignment
NameServer: NS1.IHS.COM
NameServer: NS2.IHS.COM
Comment:
RegDate:    1990-10-17
Updated:    2002-06-12
TechHandle: DA35-ARIN
TechName:   Anderson  David
TechPhone:  1-303-397-2835
TechEmail:  dave.anderson@ihs.com
```

By using the whois database we can learn many facts. In this case the IP is registered to Information Handling Services (IHS) in Englewood CO. The company owns the entire class C, 192.77.15.0-199.77.15.255. We also know that this company has owned this IP range since 1990-10-17. If necessary we can contact the technical contact provided. However, we should first determine what type of service Information Handling Services provides to their customers. Quoting from the "About IHS" webpage "Information Handling Services (IHS) is the leading worldwide provider of technical content and information solutions for standards, regulations, parts data, design guides, and other technical information.' (http://ihs.com/engineering/index.html) Taking this information and examining the original packet captures we see the pages the source host was trying to access might be technical documents. While there is no way to validate the intent of the user it looks as though this may have been a mistyped URL or bad DNS entry. We can garner one more piece of information however; we can try to learn the OS using a program called P0F. P0F is a passive network analyzer that will analyze a packet capture to try to determine OS of all IP's in the capture. The program uses the following values to attempt to determine the type of OS:

```
# wwww     - window size (can be * or %nnn or Sxx or Txx)
#          "Snn" (multiple of MSS) and "Tnn" (multiple of MTU) are allowed.
# ttt      - initial TTL
# D        - don't fragment bit (0 - not set, 1 - set)
# ss       - overall SYN packet size (* has a special meaning)
# OOO      - option value and order specification
# QQ       - quirks list
```

The program will then take these values and try to match them up to a database file included with the program. However the program was unable to determine the type of OS.

The second IP chosen was 211.47.255.22. The IP was chosen for further review based on the probing activity seen above for the detect BAD-TRAFFIC tcp port 0 traffic. This source IP was also 2nd highest on the list of top talkers for external IP address. Based on the analysis already done we know that most of the alerts generated by this host are actually the same packet being resent. Using the same techniques as used above we learn that KRNIC has current ownership of this IP range:

```
211.47.255.22 = [  ]
 (www.nic.or.kr)  Whois
  query: 211.47.255.22
   ENGLISH
  KRNIC is not a ISP but a National Internet Registry similar to APNIC.
  The IPv4 address is allocated from APNIC to KRNIC.
  KRNIC is holding the IPv4 address for further allocation to its member ISPs
  in the future. If you have any question with the IPv4 address
  Please contact at hostmaster@nic.or.kr

   KOREAN
  KRNIC       IPv4 APNIC
   KRNIC    KRNIC ISP   KRNIC
   IPv4   IPv4   ISP
   IPv4    hostmaster@nic.or.kr

  - KRNIC Whois Service -
```

A query of the KRNIC whois website (http://whois.nic.or.kr/english/) gave us what looks to be the same output:

```
?????????(www.nic.or.kr)?? ???? Whois ??? ???.
query: 211.47.255.0
# ENGLISH
KRNIC is not a ISP but a National Internet Registry similar to APNIC.
The IPv4 address is allocated from APNIC to KRNIC.
KRNIC is holding the IPv4 address for further allocation to its member ISPs
in the furture. If you have any question with the IPv4 address,
Please contact at hostmaster@nic.or.kr
# KOREAN
KRNIC? ?? ??? ????? ???? ?????. ???? IPv4??? APNIC
???? KRNIC? ??? ????, KRNIC? ??ISP?? ???? ?? KRNIC
?? IPv4?? ???? ?? IPv4??? ??? ?? ??ISP?? ??? ????. ????
???? IPv4??? ?? ????? ?? hostmaster@nic.or.kr? ????? ????.
- KRNIC Whois Service -
```

Under normal circumstances it would be prudent to try the same lookup under the Korean site and not the English version to see if there is any difference.

However my understanding of Korean is non-existent and it would be necessary to utilize additional resources to decipher the site. Using p0f to determine the OS of this IP did not return a valid answer. I did however Saro Hayan's GIAC practical paper does an excellent job of describing how to determine the OS type by hand. His results listed this OS as a "Linux 2.4.1-14 (1) kernel OS" (Hayan p.11)

The last IP chosen is 208.45.79.122. The IP was chosen based on the type of attack conducted, a WEBROOT DIRECTORY TRAVERSAL. This alert is similar to the detect analyzed above however the exploit used is not the same. The payload, the attacker is trying to execute /scripts/..%5c%5c../winnt/system32/cmd.exe?/c+dir. As we learned above this once decoded translates to: /scripts/..\\../winnt/system32/cmd.exe?/c+dir. This attack was chosen for further review based on the type of alert that was generated. This attacker is actively trying to execute an exploit against the local network. If successful this attack may gain the attacker the ability to take ownership of the victim host. A whois lookup of the address garnered the following results:

OrgName: Qwest Communications
OrgID: QWST
Address: 950 17th Street
Address: Suite 1900
City: Denver
StateProv: CO
PostalCode: 80202
Country: US
NetRange: 208.44.0.0 - 208.47.255.255
CIDR: 208.44.0.0/14
NetName: NET-QWEST-BLK
NetHandle: NET-208-44-0-0-1
Parent: NET-208-0-0-0-0
NetType: Direct Allocation
NameServer: DCA-ANS-01.INET.QWEST.NET
NameServer: SVL-ANS-01.INET.QWEST.NET
Comment: ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
Comment: NOTE: For abuse issues please email abuse@qwest.net.
(snip)

The address is assigned to Qwest Communications in Denver Colorado. Qwest is an internet provider for the Denver/Metro area. As the alert was generated in 2002 there is little chance of contacting Qwest to determine who held the address at the time of the alert. We can see that Qwest held this range of IP's during the time of the alert, the IP range was registered in 1999-06-24. If the

alert had occurred more recently, checking Dshield for previous reports from this IP should be conducted. A request could be made to Qwest to try and learn who had the address assigned to them during the attack, and if additional attacks were logged, a request to have the IP investigated by Qwest. P0f was unable to determine the type of machine generating the logs.

Analysis Process

The platform used for all the analysis was a Dell GX270 i686 Intel(R) Pentium(R) 4 CPU 2.80GHz running Linux 2.6.8-gentoo-r3.

The first step I took was to merge all the relevant pcap files into one file. This helped me see the large picture, and allowed me to gain an idea of what IP ranges would be needed for the local network. These files were merged into a file named 2002.10.all. A combination of ethereal and tcpdump were used to merge the files, and later to analyze the basic network structure.

The next step taken was to configure snort. Settings such as the $HOME_NET needed to be configured. The only other configuration change made to the snort.conf file was to enable all the rules available. In a production IDS enabling everything would have been a bad choice, causing many false positives. However, for the purpose of this practical the more information the better.

With snort configured I choose three concurrent files at random and created separate directories for each. After that was completed I ran snort on each file, I also included the 2002.10.all file as a reference.

Reading text based alert logs is not always fun so I ran snortsnarf on each of the newly generated alert files. Snortsnarf is a perl program that takes snort alert files and creates easily read html files. Snortsnarf also creates a nice summary page giving the user a quick rundown of the events logged.

The last tool I used prior to evaluating the alerts was p0f. Typically this tool generates a great deal of information with very little however it came up with very little usable information from these protocols.

Having some familiarity with some snort alert files, deciding on which alerts to choose from was not difficult. I choose the two I felt were the biggest risk and one that was out of place. Snortsnarf was a valuable tool to help quickly sort the many alerts that were generated.

Ethereal and tcpdump were both used to analyze individual log files. I found ethereal very useful to conduct quick sorting of the files.

References:

Decker, Dustin. "Reset Cause and further info (SonicOS Specific)." Online Posting. ethereal.com. 6 Sep 2004 URL: http://www.ethereal.com/lists/ethereal-users/200409/msg00057.html.

Eustace, Ernest. ""SANS Intrusion Detection & Analysis Certification." GIAC Certified Intrusion Analysts (GCIA). URL: http://www.giac.org/practical/GCIA/Ernest_Eustace_GCIA.doc.

FitzGerald, Nick. "Security Incidents: Re: Unicode Attack." Online Posting. Seclists.org. 13 Nov 2002 URL: http://seclists.org/lists/incidents/2002/Nov/0076.html.

Gordon, Les. "What is the Q Trojan?." GIAC Website. URL: http://www.sans.org/resources/idfaq/qtrojan.php.

Hayan, Saro. "SANS Intrusion Detection & Analysis Certification." GIAC Certified Intrusion Analysts (GCIA). URL: http://www.giac.org/practical/GCIA/Saro_Hayan_GCIA.pdf.

Meyer, Amanda. "SANS Intrusion Detection & Analysis Certification." GIAC Certified Intrusion Analysts (GCIA). URL: http://www.giac.org/practical/GCIA/Amanda_Meyer_GCIA.pdf.

Mogul, Jeffery. "Broadcasting Internet Datagrams". October 1984. URL:

http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc0919.html

SID 524. “BAD-TRAFFIC tcp port 0 traffic.” Snort Signature Database. URL: http://www.snort.org/snort-db/sid.html?sid=524.

Rodriguez, Tom. “What are Unicode vulnerabilities on Internet Information Server (IIS)?.” URL: http://www.sans.org/resources/idfaq/iis_unicode.php.

vision@whitehats.com. "IDS203 "TROJAN-ACTIVE-Q-TCP"." URL: http://www.whitehats.com/cgi/arachNIDS/Show?_id=ids203&view=research.

Wagoner, Andrew. "SANS Intrusion Detection & Analysis Certification." GIAC Certified Intrusion Analysts (GCIA). URL: http://www.giac.org/practical/GCIA/Andrew_J_Wagoner_GCIA.pdf.

Whitehats Network Security Resource. “IDS203 "TROJAN-ACTIVE-Q-TCP"”. arachNIDS Intrusion Detection Database. URL: http://www.whitehats.com/info/IDS203.