Global Information Assurance Certification Paper

SANS GIAC Level Two – Intrusion Detection in Depth

GCIA Practical Assignment – SANS 2002 Great Lakes
Practical Assignment Version 3.2 – Revised May 20th 2002

By

Aaron T. Hackworth

TABLE OF CONTENTS

Assignment 1: Describe the State of intrusion detection

Advantages of Distributed Agents for Network Intrusion Detection and Analysis

We're Surrounded

With the number of security threats on the rise, the struggle to maintain a safe network environment for your organization's business is a battle that is heating up. Whether you have a small single segment network or a global network composed of hundreds of sites, the number attacks on your IT assets continues to grow at an alarming rate, nearly doubling each year [1]. To control the risk these attacks pose, you harden your perimeter, secure your hosts using best practices and deploy Intrusion Detection Systems to watch the entry points into our networks. The issue is that today's attacks don't always use the front door. They often times invite themselves in through unsecured lab connections, links to external business partners, email, or remote access VPN connections, completely bypassing your firewall and network IDS sensors. There is also the increasing problem associated with mobile workers who take their laptops home or on the road and connect them to the Internet, only to come back to work the next day and hook into your "secured" network. All of this adds up to attacks coming at you from all directions and an increase in the difficulty of blocking and monitoring intrusions.

Traditional Technologies

Traditional network based IDS systems can be a powerful weapon in detecting misuse of our network resources but they are only effective if the packets we are interested in cross the path of their sensors. This limited field of vision can result in many attacks not being seen, especially those that are sourced from inside your network.

One strategy to address the "field of vision" issue is to turn to Host Intrusion Detection based systems. HID systems install on the individual hosts you want to monitor, so they are always in the right place, but they also have shortfalls in design and deployment that leave us not seeing the whole picture. First, HID agents are generally deployed only to the centralized servers in an organization and not to user workstations. The idea that all of your valuable data resides on the central servers is increasingly flawed thinking. Who among us doesn't have some critical document or other data on their laptop? An attack on these decentralized resources can be a serious blow to an organization's operations if data is lost or worse, if control of the workstation is lost so that it can be used as an internal launching board for other attacks. A second issue with most HID systems is that they focus almost entirely on detecting events related to the individual host's applications and operating system, using log analysis as their primary means of detection. So while the HID systems are in the right place to see the attacks, they often are not focusing on the right data to make the detect.

To see the whole picture, we need a host based IDS deployment model with agents that focus on collecting and analyzing network based data. Using distributed agents for network IDS can help up cover our blind spots.

An Eye Witness at Every Event

Placing the network IDS module on the host means you always see what the target sees. There is no "back door" path that can be used to evade the IDS if it is in-line with the network stack. It also means that as your assets move, your IDS sensors automatically move with them. A model like this auto adjusts as people work from home, on the road, or in different office locations in your company.

Another strong advantage of this proximity to the host is that it neutralizes the effectiveness of some of today's more popular firewall and IDS evasion techniques like fragmentation and encryption by allowing us to asses packets after fragment re-assembly and possibly even after decryption. There will always be ways to evade detection, but taking these two powerful techniques away from the attackers represents a step forward.

Being at the target host also means you can see how the host responds to a stimulus. Did it send any information back to the attacker or just silently drop the packet? Knowing what the response was can help you determine the intent of the packet and the success of the attack. Seeing what went back to the source of the stimulus also provides you with insight into what the attacker may now know about your host. This can be valuable in determining your exposure factor or deciding what to look for in the future.

Synergies from the Collective Viewpoints

Using a distributed agent model can allow you to see through the eyes of many different hosts. This high density distribution allows you to correlate scans and attacks that you may not have seen before. An example would be a SYN scan from an internal host against an internal network segment. A traditional network IDS deployment probably would not see this activity at all, due to the internal nature of the traffic, and your HID systems may not be looking for these kinds of network level events. Monitoring for network events at the host, we will see a single SYN packet come to a port. If the host is listening on the port, it will respond with a SYN ACK, if not, a RST should be sent back. Either way, we have a stimulus and a response. In the case of a SYN ACK being sent, we expect to receive an ACK back, but what if this never comes? That constitutes suspicious behavior and could be flagged. A SYN to a closed port in and of itself is strange and could also raise a flag. Obviously, you would not want to alert on each individual event like this, but if you could see that 100 hosts across your enterprise reported a similar event in a relatively brief period of time the scanning attempt would now be in clear focus and you could begin to make informed decisions about how to react.

Having hundreds or even thousands of these "electronic eyes" on the network can also help us by revealing the data from different vantage points. Such a diverse perspective can give new meaning to network data in the hunt for the source of a stealthy attacker.

Tracking the Attacker

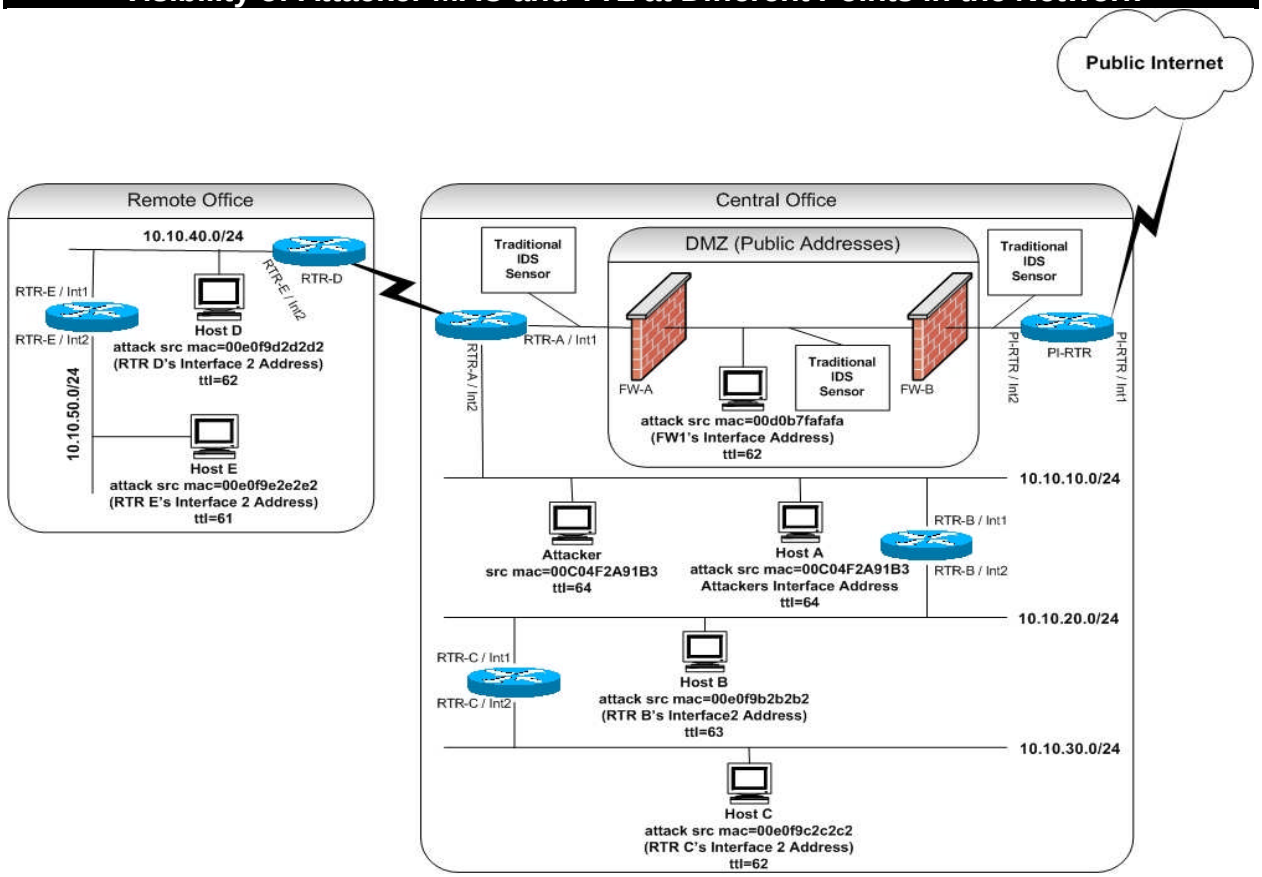
A common technique employed by attackers to avoid being tracked is to send their malicious packets out with an incorrect source IP address, a technique known as IP spoofing [2]. Obviously this won't work for attacks where the attacker needs a reply to the

initial sender, but not all attacks require a reply. There are Denial of Service attacks and backdoors like Mixter's Q [3] that don't need any response to be effective. The receipt of the traffic at the destination host can be all it takes to complete the attacker's goal. Even network scanning can be accomplished using a spoofed source IP address through techniques like the "idle host scan" [4]. In situations like these, we still would like a way to determine where on our network the attack is originating from. In these cases, having many vantage points from which to collect data can give new value to data like the source MAC address and IP TTL in hunting the source and stopping attackers.

Track the MAC

On Ethernet networks, 6 Byte source and destination MAC addresses are included in every frame that is transmitted on the network. The source MAC uniquely identifies the network card that transmitted the frame and the destination MAC identifies the next hop network card that should receive the frame. The issue with using this data to track attackers is that MAC addresses are only relevant for local segment transmissions so once a packet passes through a router, the source MAC changes to the MAC address of the routers egress interface and the destination MAC changes to the MAC of the next hop network interface. The result is that after one hop through a router, the source MAC no longer identifies the specific source computer that sent the packet. This is information we can use to our advantage if we can make our detects at various points in the network.

Visibility of Attacker MAC and TTL at Different Points in the Network



Consider an attacker on the 10.10.10.0/24 segment in the diagram above. He would like to perform a network scan of our internal systems from inside our network. The IDS sensor monitoring our Internet and DMZ ingress connections would not be able to detect this activity because routing would not allow any of the scans packets to cross a link that the sensors are monitoring. Not knowing where our sensors are located, the attacker also decides he will spoof his source IP to avoid detection. This presents him with the problem of monitoring for replies to his scanning queries but as mentioned above, this issue can be overcome in a number of ways including being located on a non-switched segment, tapping the lines somewhere, ARP poisoning or flooding attacks on the switch, … Catching this attacker will require clues to be collected and an alarm to go off. This is where the distributed IDS agent running on Host A comes into the picture.

As the attacker scans, he will begin raising flags on our sensor agents located around the various network segments. Eventually, enough of these flags will signal an alert at the central correlation engine and the packet headers and other information that triggered the flag at the hosts could be requested for central analysis. Agents that are separated from the attacker's network segment by routers would see the spoofed IP address and the MAC address of the egress interface of their segment router. However, a sensor agent running on the attackers network segment (Host A above or any other active agent on the same segment as the attacker), would see the attacker's real MAC value when his scanning or attack packets arrived. If we are clever in our agent or analysis engine design, we could also monitor for and detect spoofed MAC addresses from the local segment.

If we determine that the source IP of a received packet is local to our subnet then as our agent examines the reply to that packet it can also examine our ARP cache (or send an ARP request) for the MAC address that matches the source IP in the attacker's packet. If the arriving MAC didn't match the MAC we got on reply to this query or if we got no ARP reply for the original source IP, we could draw some conclusions about validity of the source address on the packets we received. There are occasions where this might lead to false alerts, but a single sensor's data combined with data from other systems that raised similar flags can help us filter out the false positive noise and see the attacks in clear focus. If a machine receives a packet from a remote subnet source IP and it reports a source MAC that does not match a router interface for your LAN segment, you can again tell that the attacker is spoofing a remote IP address and is actually local to the subnet the detect happened on. Advanced analysis of this kind would require the host's IDS agent to maintain state, but it would not cause the same load issues as maintaining state at centralized network IDS sensors because you only need to maintain the state for a short period of time and only for the individual PC where the agent is running. The value of this kind of data is that we can determine if an attack is coming from the local subnet. Having enough agents around the network will allow us to see the subnet that the attack came from and track the attacking host back to a particular switch port or desk and shut them down.

There is one additional clue to be had from the MAC address. If the attacker's real MAC address was sent out with the attacking/scanning packets, we may be able to determine the attacker's hardware type. The first 3 bytes of the MAC address represent the vendor

that manufactured the NIC card. Information on vendor MAC codes can be found at http://standards.ieee.org/regauth/oui/oui.txt. Seeing a source MAC registered to a particular vendor can help you focus on the types of devices that could be the source. For example, you would not look for a Dell laptop if the MAC was registered to Hewlett Packard. This is of course of less useful to us if the MAC is registered to a vendor that is a mass producer like 3Com or Intel, but it still has value as an additional piece of intelligence data we can use to fight the battle.

Attack Source Isolation using TTL values

A second piece of IP packet data that has increased value under this model is the IP TTL. This value is set in the IP when a packet is created at the sending host. At each hop in the network, the TTL value is decreased by one. When the value finally reaches 0, the packet is discarded by the device and an ICMP error message is sent back to the source IP address that is set in the IP header field [5]. This TTL mechanism is intended to prevent packets from getting stuck in routing loops and just bouncing around your network forever, but if we can collect enough packets at different points in the network, we can use this value to close in on the proximity of an attacker on our network.

Different hosts set the TTL value to different initial values when they construct a packet to be sent out. Even in the event of a crafted packet, not all malicious code randomizes this initial TTL so if you can determine the original value of the TTL then you may also be able to determine the number of hops (i.e. routers) that the packet traveled through to get to your sensor agent. Other tracking techniques using ICMP TIMEX packets' TTL values have been used in the past to track attackers to sections of the Internet [6], but in the case of a private networks with agents on all or nearly all subnets, you can use the TTL in the original packets to identify the proximity or even the exact segment of the attacking host. If the attacking computer uses a non-random TTL then agents reporting in the highest TTL values would be the ones closest to the source. Having at least one agent on each segment could positively identify the segment that the attack was originated from because these would report the highest TTL value of all sensors (the initial TTL). Having two or more agents on a segment could also be used to detect randomized TTL values. If packets were received from the same IP but with significantly different TTL values, you can infer that there is either a serious routing issue that needs to be dealt with or someone is trying to manipulate this value. This kind of analysis presents a more computationally expensive task for a central analysis engine, but it is within the realm of possibility.

Even if you didn't happen to have a PC running your IDS agent on the same segment, you should still be able to close in on the attacker by getting samples from segments further away from the attacker. An agent seeing ttl = x on one side of a router and an agent seeing ttl = x-1 on the opposite side of a router can eliminate all networks on the side of the router where ttl = x-1 as the source. This can help you close in on the attacker and work towards isolating them to a single subnet.

Once you have the exact segment or some directional vectors to work with, you can use packet sniffing devices or other switch statistics to track and isolate the attacker's location.

A final benefit of capturing TTL data is its benefit towards detecting the attackers OS type. OS fingerprinting is the use of network stack characteristics to determine the host OS type that generated a packet. One characteristic that varies between IP stack implementations is the initial TTL value [7]. While this statistic alone is far from conclusive, it can help you narrow the field of possibilities and that adds to the clues that ultimately help you in hunting the hacker.

Current State of Distributed IDS

Currently available products that have made the most progress in the area of distributed agent IDS are from the personal firewall software vendors. Vendors like Symantec, with their Client Security solution, and ISS with their acquisition of BlackICE, have begun to integrate intrusion detection capabilities with personal firewall products. These products can make detects, stop the attack and report information about the events to a centralized data store for further correlation or analysis. These products and others like them are powerful computer security tools and represent a strong start at what I believe will be a key element to the future of IDS systems.

The primary issue with the current product releases is that they are still applying traditional IDS thinking and not fully leveraging the power of their unique distribution architecture and proximity to the client. The detect engines are primarily based on signature matches at the local PC and don't tend to report enough detailed data or use that data for central correlation of network events from multiple PCs. In the example of a single SYN packet to port 80 across each individual host, these systems would not detect that a mapping effort was taking place. We need to see a shift to focusing on network events and "big picture" thinking if the true power of the distributed agent design is to be realized.

Future Possibilities

Do you speak my language?

Attackers share code, tips, and tricks. They also mentor each other and work together to exploit your vulnerabilities. To effectively defend against this, we need our security devices and our community to work together. This involves sharing information and tools in a timely manner. A good start down this path would be the development and adoption of a standards based reporting language for IDS systems. This is needed for complete internal coverage as well as for sharing of data across organizational boundaries.

Internally, implementation of a standard reporting language would enable your firewall, routers, switches, server, PCs, printers, etc. to all report in to a single IDS data store for event correlation. On the people side, it would ease the sharing of information and tools with peers and government agencies to help track and stop attacks in cyberspace. Other areas of security are adapting standards based languages for their reporting and/or

message passing (ex. SAML and the dozens of other XML based tagging languages) and the IDS community should make this move as well.

Monitoring the Internet

An area where distributed network IDS agents could have a potentially huge impact is in monitoring of the Internet. The success of distributed computing projects like Distributed.NET [8] and SETI@Home [9], demonstrate that thousands of people are willing to donate their space CPU cycles to a cause. Why shouldn't this "cause" be monitoring and reporting anomalous activity on the Internet? An agent model like the one deployed by SETI@Home, where the agents would turn on as a screen saver, could be applied as an IDS sensor. When these agents activated, they could check in, get their signatures and instructions from a central console and go to work.

On the Internet, having agents around the globe that could report centrally for analysis and correlation would help us detect attacks, profile them and isolate their sources more quickly. This kind of early warning system might allow other participants enough time to prepare for attacks before they reach their network and they are exploited. The end result would be better security and cost savings for everyone in cyberspace.

Collection of attacker data from thousands of Internet connections could also help our profiling efforts. TTL values from these locations could possibly help narrow the search for an attacker to a particular ISP, Country, City, or even street address. This combined with existing techniques and proximity of the first alerts, could be valuable in our attempts to find the IP spoofers.

With the increase in the number of Cable and xDSL connections, there are millions of hosts in every address space imaginable connected to the Internet 24x7. These computers are not always in use and their spare cycles and vantage points could be used to help secure the Internet. It may even be possible to embed IDS agent code in the firmware of cable/DSL modems, home routers or other access devices. This way, even if the computers are off, the IDS sensors and the hunt for the attackers would still be on.

Conclusion

Further development of distributed IDS agents for network intrusion detection will expand the ability to monitor for attacks, especially from internal threats. Their position at the targeted hosts allows them to see more and reduce the effectiveness of many IDS evasion techniques. As the numbers and locations of deployed sensor agents grows, the collective view from many network locations can allow the Security Analyst to see the attacks more clearly and also to track the attackers more accurately.

All IDS technologies have strengths and weaknesses. The strengths of each model should be capitalized on, but the weaknesses can be compensated for by using a blended IDS approach. The ability to implement this blended approach will become more viable as a common event reporting standard is developed and implemented by industry vendors. This will be the key to centralized collection from all IDS and other network sources and will help to enable powerful centralized event correlation and analysis.

Finally, the development of lightweight non-intrusive agents that exist in communications hardware such as home routers, modems, NIC or as software agents on host systems will continue to increase our electronic vision across our organization or possibly even the entire Internet. This expanded coverage would enable more detects, faster detects and improve on techniques for tracking attackers like TTL analysis and/or MAC address examination. While 100% coverage and "real time" reaction may be impossible to attain, these kinds of tools and the additional information collection by them help us move much closer to this goal than we have ever been before.

References

[1] CERT/CC Statistics 1988-2002 < http://www.cert.org/stats/cert_stats.html>

[2] Zwicky, Elizabeth D, et all - Building Internet Firewalls 2nd Edition. June 2000 p98-100

[3] Mixter http://mixter.warrior2k.com

[4] Erik J. Kamerling - The Hping2 Idle Host Scan. February 26, 2001 <http://rr.sans.org/audit/hping2.php>

[5] Stevens, Richard W. TCP/IP Illustrated, Volume 1 The Protocols, 1994 Addison Wesley p.36.

[6] Northcutt, Stephen and Judy Novak. Network Intrusion Detection, An Analysts Handbook 2nd Edition. Indianapolis, IN: New Riders Publishing, 2000. pp 360-361.

[7] Max Vision http://www.whitehats.com/library/passive/

[8] http://www.distributed.net/

[9] http://setiathome.ssl.berkeley.edu/

Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, Diego Zamboin. Architecture for Intrusion Detection Using Autonomous Agents. Coast Technical Report 98/05. June 11, 1998. <http://www.cerias.purdue.edu/about/projects/aafid/>

Assignment 2: Network Detects

Detect #1 – Crafted Packets from limited broadcast address to LPD port

Relevant packets from tcpdump capture

```
21:22:52.144488 255.255.255.255.31337 > 226.185.171.177.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 8ed3!)
22:25:01.194488 255.255.255.255.31337 > 226.185.58.58.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 4b!)
23:24:04.174488 255.255.255.255.31337 > 226.185.27.146.515: R [bad tcp cksum
908d!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 20f2!)
00:03:13.214488 255.255.255.255.31337 > 226.185.159.127.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 9b05!)
00:22:40.204488 255.255.255.255.31337 > 226.185.236.129.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 4e03!)
00:37:04.194488 255.255.255.255.31337 > 226.185.156.239.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 9d95!)
00:58:01.254488 255.255.255.255.31337 > 226.185.231.7.515: R [bad tcp cksum
8d90!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 527f!)
01:03:37.194488 255.255.255.255.31337 > 226.185.92.82.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum de32!)
01:27:46.254488 255.255.255.255.31337 > 226.185.7.58.515: R [bad tcp cksum
908d!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 354a!)
01:48:28.254488 255.255.255.255.31337 > 226.185.142.214.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum abae!)
02:16:28.234488 255.255.255.255.31337 > 226.185.212.79.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 6635!)
03:35:07.234488 255.255.255.255.31337 > 226.185.9.30.515: R [bad tcp cksum
8e8e!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 3268!)
04:02:31.264488 255.255.255.255.31337 > 226.185.85.228.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum e4a0!)
04:38:22.284488 255.255.255.255.31337 > 226.185.137.228.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum b0a0!)
04:41:31.284488 255.255.255.255.31337 > 226.185.95.129.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum db03!)
06:00:52.284488 255.255.255.255.31337 > 226.185.66.66.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum f842!)
06:05:04.334488 255.255.255.255.31337 > 226.185.129.157.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum b8e7!)
06:29:52.314488 255.255.255.255.31337 > 226.185.114.198.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum c7be!)
06:56:22.294488 255.255.255.255.31337 > 226.185.226.106.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 581a!)
07:16:16.364488 255.255.255.255.31337 > 226.185.25.1.515: R [bad tcp cksum
8e8e!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14,id 0, len 43, bad cksum 2285!)
07:20:37.324488 255.255.255.255.31337 > 226.185.63.24.515: R [bad tcp cksum
8d90!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum fa6e!)
08:01:43.344488 255.255.255.255.31337 > 226.185.153.179.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum a0d1!)
08:09:13.354488 255.255.255.255.31337 > 226.185.125.167.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum bcdd!)
08:24:10.374488 255.255.255.255.31337 > 226.185.79.169.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum eadb!)
```

```
09:33:10.374488 255.255.255.255.31337 > 226.185.243.222.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 46a6!)
```

1. Source of Trace: http://www.incidents.org/logs/Raw/2002.5.3

2. Detect was generated by: *tcpdump* capture and manual analysis. According to the information at intrusion.org, these captures were triggered by a Snort rule. If this is true, the rule was most likely the rule listed below or a derivative of it. This rule was taken from the backdoor.rules file which is included in the snortrules-current.tar available for download at http://www.snort.org

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;)

3. Probability the source address was spoofed: Nearly 100% - Reasons for this conclusion include:

- The source address is the limited broadcast address. This is commonly used as a destination address for DHCP and other local network services that need to send a broadcast to their local subnet before they have their IP configuration, but it should never appear as a source address.
- For a TCP connection to form, a 3-way handshake between the source and destination IP would be required. This is not possible with this source address because:
    - TCP is point to point and 255.255.255.255 is not an individual host address.
    - The source address is not routable and the ttl = 14. Unless the initial ttl = 14 (i.e. the packet was sourced from the same segment), there is no chance that a connection could ever complete.

There is a very low probability that this is a third-party effect because of the following:

- The original packets would have needed to be spoofed from the 226.185/16 multicast address space to 255.255.255.255. While that may be possible, the hosts that received the packets to 255.255.255.255 would respond from their own individual IP if at all.

There is further evidence detailed in the next section (Description of Attack) to support the theory that these are crafted packets, giving further weight to the argument that the source IP is not true.

4. Description of Attack: This attack generates packets with the following characteristics:

- Source IP Address = 255.255.255.255 (Limited Broadcast Address)
- Source TCP Port = 31337 (spells ELEET in “Hacker” speak)
- Destination Networks are from the IANA reserved Multicast space (This may be the result of address obfuscation in the dump file, if not; using multicast for

distribution is a brilliant attack distribution scheme.). The fact that there are various multicast addresses as the destination address for these packets may indicate that the attacker is looking for addresses that are currently being forwarded through the target network(s) multicast backbone.

- Destination TCP Port = 515 (usually associated with LPD / printing services)
- IP ID is always set to 0
- RST / ACK flags are set (this may be an attempt to pass packet filters or firewalls)
- Acknowledgement Number = 0 (For a normal RST packet, this should equal the Sequence number of the packet that caused the RST + 1)
- Packet has 3Bytes of data in it. The data is = "cko"
- The attack is averaging a packet about every 29 minutes but not at specific intervals. The longest delay between these packets was around 1 hour 19 minutes. The shortest time was around 3 minutes 9 seconds. There were never more than 4 in a single hour of this log. This timing could be the result of:
    - Segment where the traffic was captured could be the path to end networks that are joining and leaving multicast groups that match the packets.
    - Could be an attempt to avoid IDS detection (although doubtful because of the other overt qualities of the packets)
    - A manually executed attack
    - We may not be seeing all of the attack packets. Given an expanded time window (past and future packets), we may be able to form a better conclusion on why the timing is the way it is or begin to see full coverage of the address space.

These packets were most likely detected on an Ethernet segment because the Frames are padded to 46Bytes as dictated in RFC 894 (A Standard for the Transmission of IP Datagrams over Ethernet Networks)

External References to an older version of this attack include:

- Whitehats - arachNIDS Reference: IDS203/TROJAN_TROJAN-ACTIVE-Q-TCP - < http://www.whitehats.com/info/IDS203>
- CVE Reference Number: CAN-1999-0660 (NOTE: No CVE Exists for this attack but there was a CAN (Candidate) entry proposed on August 4th, 1999)

Source code the latest version of Q can be found at the author's web site (http://mixter.warrior2k.com/). The site also contains an excellent paper on backdoor theory based on the Q concept.

5. Attack Mechanism: (Stimulus / Communications to Backdoor)

This attack is sending crafted packets sourced from the limited broadcast address to IP addresses in the multicast ranges. The packet headers indicate that they are bound for port 515/tcp which is normally used for LDP printing services.

A definite purpose for this attack is hard to determine. The most likely possibility is that the attacker is looking for or sending instructions to Backdoor Q or a similar backdoor.

Q consists of a client and a server. The server lives on the compromised host. It is often set up to hide itself completely or to disguise itself as another process so that it is hard to detect with commands like 'ps'. The Q daemon can be set up to ignore OS signals making it hard to kill as well. It has the capability to listen on a RAW socket which allows it to receive commands from the attacker and possibly do something without needing to complete the TCP 3-way handshake, without having a valid/reachable source address to respond to, and without binding to a specific socket. Some implementations for this backdoor include the ability to encrypt the instructions sent to it. This can make the intent of the packet payloads very hard to determine. The "cko" we are seeing in the trace above could be a wake-up command or almost anything else.

6. Correlations:
Sage, John - May 13, 2002 - http://www.incidents.org/archives/intrusions/msg11889.html

These packets all seems to be directed at port 515/tcp but other similar scans have seen this directed at port 80 as shown in the report documented by John Sage at: http://www.incidents.org/archives/intrusions/msg11889.html

Date: Mon, 13 May 2002 13:52:08 -0700
From: John Sage <jsage@xxxxxxxxxxxxxx>
Subject: LOGS: ACID Incident Report 05/10/02

```
/* Let's see if I've got it together enough
 to submit these on a regular basis, once again */

Subject: ACID Incident Report 05/10/02
From: ACID Alert <acid@xxxxxxxxxxxxxx>

Generated by ACID v0.9.6b21 on Mon May 13, 2002 13:12:04

#(114 - 3) [2002-05-10 08:18:37]  TCP to 80 http
IPv4: 216.17.51.149 -> 12.82.141.69
     hlen=5 TOS=0 dlen=43 ID=0 flags=0 offset=0 TTL=18 chksum=912
TCP:  port=9203 -> dport: 80  flags=***A*R** seq=0
     ack=0 off=5 res=0 win=0 urp=0 chksum=5089
Payload:  length = 3

000 : 63 6B 6F                                    cko

p0f:

[Fri May 10 08:10:50 2002] 216.17.51.149 [15 hops]: Windows 2000 (9)
 + 216.17.51.149:9203 -> 12.82.141.69:80 (timestamp: 2817765 @1021043450)
[Fri May 10 08:11:00 2002] 216.17.51.149 [15 hops]: Windows 2000 (9)
 + 216.17.51.149:9203 -> 12.82.141.69:80 (timestamp: 2817765 @1021043460)
```

The fact that a backdoor like Q can listen on a RAW socket makes the destination port arbitrary. As long as Q is listening for packets that match a specific signature, it can act

off a packet bound for any destination port. Good ports to pick for the client commands would be ports commonly allowed through firewalls or routers (like 80/tcp for web traffic).

7. Evidence of Active Targeting: The packets all appear to be directed at a specific type of backdoor. The packets are all directed to port 515, but this may be arbitrary.

8. Severity Ranking: Because these are directed to multicast addresses, It is unknown what types of systems will actually receive the packets. Without more information we have to assume the worst. Assigning a value from 0 to 5 for each of the following elements, this attack is ranked as follows:

- **Criticality** = 5 (System importance is unknown so we must assume the worst)
- **Lethality** = 5 (If this is Backdoor.Q or a variant, it could be a full system compromise including installation of a root kit making it hard to detect and cleanup)
- **System Countermeasures** = 0 (We don't know the system's response)
- **Network Countermeasures** = 0 (Unless this detect was captured outside the firewall, then the packets made it into the network. Since network architecture and defenses where this was captured are unknown, we must assume the worst.)
- **Overall Severity Score = (5 + 5) – (0 + 0) = 10**

9. Defensive Recommendations: Block all packets with source IP addresses of 255.255.255.255 at your firewall. As a general rule, you should block all packets entering your network that are sourced from broadcast addresses of any kind including limited broadcast, net directed broadcast (both all 1's and all 0's in the host field), any local addresses and all other reserved address spaces as defined in the IETF draft located at http://www.ietf.org/internet-drafts/draft-manning-dsua-08.txt. This should help to limit the amount of spoofed packets that make it into your network. In most cases, it is also wise to block packets that have a broadcast destination address.

Since these packets are directed to multicast addresses, we don't know which host will see them. Routers usually control the distribution of multicast traffic and only deliver it to segment that have host officially participating in the multicast broadcast. This means that if there is a multicast enabled hosts listening on a particular segment, the packet should be forwarded to that segment. This means different segments may see this traffic as legitimate multicast hosts come online and drop off. Since we can't tell who will see them, we need to check all hosts for backdoor applications. Remember that these backdoors often hide or disguise themselves as other applications so a thorough verification of the running services will be necessary to make sure they are what they appear to be and nothing more. Be careful on this step. If Q is running, the attacker will have had root level access to the system and a root kit may be installed. You might not be able to trust what you see from ifconfig, ps, lsof, …

Check the multicast routing configuration in your network. Improper multicast configuration could result in routers forwarding these packets out interfaces even if there are no hosts participating in a multicast group for the address. Even if the routing is properly configured, this could have been a carefully thought out plan to distribute the

packets across multiple segments at once, especially if the timing of these packets matched a large multicast event on the network like a web or audio cast to all employees.

If these packets came to use via the public MBone, then consider if you really want to allow this kind of traffic into your organization. This is a policy issue that needs to be reviewed.

10. Multiple Choice Question:
In the following tcpdump output, the length field on the following packet is 43Bytes but the packet has 46Bytes of data in it. Why?

```
09:33:10.374488 255.255.255.255.31337 > 226.185.243.222.515: R [bad tcp cksum
8f8f!] 0:3(3) ack 0 win 0 [RST cko] (ttl 14, id 0, len 43, bad cksum 46a6!)
0x0000   4500 002b 0000 0000 0e06 46a6 ffff ffff        E..+......F.....
0x0010   e2b9 f3de 7a69 0203 0000 0000 0000 0000        ....zi..........
0x0020   5014 0000 facd 0000 636b 6f00 0000             P.......cko...
```

a) The length field is crafted and is causing tcpdump to show incorrect information
b) The physical media's minimum frame length required 3 Bytes of padding
c) The actual number of Bytes in an IP packet must be padded to a multiple of 8
d) None of the above

Answer: b (Ethernet requires a minimum frame size of 46Bytes. If this is not met by the packet, additional 0's are padded to the end to meet the requirement.)

Questions and Answers from posting to Intrusion.org:

Q: What are some of the backdoors that listen on port 31337?

A: The most famous backdoor that listens on this port is Back Orifice, a popular backdoor for the Windows NT operating system. Some others are listed in the Intrusion Detection FAQ located at
http://www.sans.org/newlook/resources/IDFAQ/oddports.htm

Q: What is running on port 515/tcp and what is LPD?

A: Port 515/tcp is a port that is commonly used for the Line Printer Daemon (LPD) service. The purpose of this service is to accept print jobs from client hosts running and Line Printer Requestor (LPR) client and then control an assigned printer to print the documents. If a system is a print server then this would be a normal port to be listening on a host. If it were not a print server, this still might be listening as a lot of poorly configured systems load the LPD daemon by default.

Q: What kind of host usually gets infected with backdoor Q?

A: UNIX and other UNIX like OS(s).

Q: How can I tell if a backdoor is running on my system?

A: This is a tough question that is probably best left to a professional trained in incident response and system forensics. My best advice would be to turn this over to that kind of specialist. Short of that, some things that a system admin could check would include:

Check if there is anything listening on port 515/tcp on the target system. On most systems, this can be accomplished using the "netstat -pan | grep :515" command. If it is, you will see something like:

tcp 0 0.0.0.0:515 0.0.0.0:* LISTEN 9871/lpd Waiting

The 9871 in front of the slash is the process ID. You can now do a ps -ae | grep lpd and look for any other process IDs. If they are there, you might want to unload all instances of lpd and see if it still shows up in your ps -ae list. A source code analysis of the latest version of Q reveals that I can start Q using another name. This lets it hide under an alias and the output of the ps command will show the alias. If you stop all legit versions of lpd and one is still showing up, this could indicate that something else is masquerading.

Since Q may not listen on 515, but rather on a RAW socket, you can try to use the lsof (list open files) command to check for a RAW socket. Use the command: lsof | grep ' sock ' to see if there are any that have unidentified protocol listed. This can indicate an open RAW socket. NOTE: List open files can detect open network sockets because *NIX based systems basically treat every device (including the network card) as a file for I/O purposes.

Finally, a test that might help you detect Q on the system would be to check if the network card is in promiscuous mode. Some versions of ifconfig can tell you this in the output. To do this, simply type “ifconfig” at the command line and look for PROMISC or something else in the output to indicate that the card is in promiscuous mode. Some versions of ifconfig do not report this so you can not rely on this test 100%.

The last thing to remember with these tests is that they might not turn anything up even if you are compromised. If Q is really present, the attacker that put it there might have also installed a root kit and replaced the standard versions of lsof, ifconfig, and other tools that could have been used to detect its presence. Only a trained computer forensics specialist could determine this for you. A properly configured tool like tripwire may also help you detect the presence of a root kit but at this point it would be too late to try to install it. These tools are best installed at system build time because

once a system has been running “in the wild”, we are unable to be 100% sure about the integrity of its files.

Q: Should the machine be isolated, disconnected, reformatted? Should the user be questioned? What should the typical Incident Response be for the highest level?

A: Reaction to this kind of “unknown” is a decision that each organization would have to make for themselves. Some would justifiably shut off and disconnect of targeted hosts until a full forensics investigation could be completed. This could be a high cost to the business, but if the (probability of the target actually being infected) x (potential financial loss if a successful attack occurs) >= a dollar figure higher than the business impact of the shutdown, then by all means they should hedge their risk and shut it down. In the end, the right decision is all about risk management principals and the dollars and sense of the individual situation.

Detect #2 – Spida Worm

This detect was pulled from my firewall logs just a few days after the Spida Worm was reported by CERT. These logs show an active attempt by compromised systems to further spread the infection.

Log Files: These logs were generated by a Netscreen-5XP hardware firewall. Log file format is:

Date Time Action Source IP:Source Port->Destination IP:Destination Port Duration of event Application

```
==========================================================================
   Self Log, (Current system time: Fri, 31 May 2002 10:48:56)
==========================================================================

 Date      Time     Action  Source->Destination              Duration Application
2002-05-31 07:27:18 Deny  66.253.1.12:1528->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-31 07:27:12 Deny  66.253.1.12:1528->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-31 07:27:09 Deny  66.253.1.12:1528->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-31 03:56:17 Deny  202.97.172.28:4372->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 03:56:11 Deny  202.97.172.28:4372->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 03:56:08 Deny  202.97.172.28:4372->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 01:44:25 Deny  64.240.169.45:3265->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 01:44:19 Deny  64.240.169.45:3265->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 01:44:16 Deny  64.240.169.45:3265->a.b.c.d:1433     0 sec          TCP PORT 1433
2002-05-31 00:04:45 Deny  12.250.75.23:1667->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-31 00:04:38 Deny  12.250.75.23:1667->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-31 00:04:35 Deny  12.250.75.23:1667->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-30 23:14:33 Deny  209.139.209.132:4132->a.b.c.d:1433    0 sec          TCP PORT 1433
2002-05-30 23:14:27 Deny  209.139.209.132:4132->a.b.c.d:1433    0 sec          TCP PORT 1433
2002-05-30 23:14:24 Deny  209.139.209.132:4132->a.b.c.d:1433    0 sec          TCP PORT 1433
2002-05-30 19:24:24 Deny  211.219.21.34:4497->a.b.c.d:1433      0 sec          TCP PORT 1433
2002-05-30 19:24:18 Deny  211.219.21.34:4497->a.b.c.d:1433      0 sec          TCP PORT 1433
```

2002-05-30 19:24:15	Deny	211.219.21.34:4497->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 15:19:32	Deny	64.221.130.82:2012->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 15:19:26	Deny	64.221.130.82:2012->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 15:19:24	Deny	64.221.130.82:2012->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 14:22:08	Deny	213.21.158.19:2039->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 14:22:01	Deny	213.21.158.19:2039->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 14:21:58	Deny	213.21.158.19:2039->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 13:53:30	Deny	64.76.134.162:1512->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 13:53:24	Deny	64.76.134.162:1512->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 13:53:21	Deny	64.76.134.162:1512->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 10:47:58	Deny	210.21.226.49:1383->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 10:47:52	Deny	210.21.226.49:1383->a.b.c.d:1433	0 sec	TCP PORT 1433
2002-05-30 10:47:49	Deny	210.21.226.49:1383->a.b.c.d:1433	0 sec	TCP PORT 1433

1. Source of Trace: Cable modem connected to public Internet at my home.

2. Detect was generated by: NetScreen 5-XP Firewall (The firewalls external interface was replaced by the IP address a.b.c.d in the logs entries above).

3. Probability the source address was spoofed: There is almost no possibility that the source addresses were spoofed. The purpose of the packets is a SYN scan for systems running MS SQL on port 1433/tcp in hopes of generating a SYN ACK response. The next step is to complete the 3-way handshake and exploit a common weakness on the remote systems. To receive this SYN ACK response and continue with the attack, the source IP must be alive and reachable. There are a few possible ways to circumvent this requirement but they require a great deal of expertise and are highly unlikely in this case.

As further proof that the source addresses were not spoofed, I was able to contact individuals responsible for some of the addresses in my logs. After they checked their systems they confirmed that they had found the infection and executed cleanup.

4. Description of Attack: This attack generates packets with the following characteristics:

- Source IP Address = Real IP of an infected system
- Source TCP Port = My logs all show the use of various an ephemeral ports
- Destination IP Address = Semi-Sequential through classful network ranges
- Destination TCP Port = 1433 (Commonly used for MS-SQL Server)
- Initial SYN Packets hoping to generate a SYN ACK response
- (3) attempts from each source address using standard TCP timeout retry values (initial packet then 3 seconds later then 6 seconds later). There is no final attempt 12 seconds after the (3) tries.

External references:

- CERT® Incident Note IN-2002-04
  - <http://www.cert.org/incident_notes/IN-2002-04.html>
- CVE reference number: CAN-2002-0154 (Candidate for CVE list)

5. Attack Mechanism (Stimulus / Trying to Connect to SQL Server to spread worm):

The packets we can see here are a SYN Scan for MS SQL server running on 1433/tcp (the default port). If the packets reach a host with this server running, it will generate a SYN ACK and then the attacker will send an ACK to complete the 3-way handshake. Once the connection is established, the attacker will attempt to access the SQL Server instance using the 'sa' account (SQL's administrative account) and a blank password (a common installation error with MS SQL Server). If successful, the Spida Worm will spread to the new victim and the compromise is complete. The newly infected system also becomes an attacker looking for other systems to infect.

6. Correlations: CyberArmor personal firewall logs on my laptop (which is connected to the Internet via a separate direct ISP connection) also detected these signatures.

```
CyberArmor Log format
=====================
EventID Date Time        Action Protocol Direction  SrcIP/SrcPort DstPort

0009705d 2002/05/29 07:36:50   Deny Alarm  TCP Inbound    139.130.220.94/1760    1433
0009705e 2002/05/29 07:36:50   Deny Alarm  TCP Inbound    139.130.220.94/1760    1433
0009705f 2002/05/29 07:36:53   Deny Alarm  TCP Inbound   139.130.220.94/1760    1433
00097060 2002/05/29 07:36:53   Deny Alarm  TCP Inbound    139.130.220.94/1760    1433
00097061 2002/05/29 07:37:00   Deny Alarm  TCP Inbound    139.130.220.94/1760    1433
00097062 2002/05/29 07:37:00   Deny Alarm  TCP Inbound    139.130.220.94/1760    1433
00097071 2002/05/29 10:24:35   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
00097072 2002/05/29 10:24:35   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
00097073 2002/05/29 10:24:38   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
00097074 2002/05/29 10:24:38   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
00097075 2002/05/29 10:24:44   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
00097076 2002/05/29 10:24:44   Deny Alarm  TCP Inbound    61.136.9.50/3026    1433
```

The fact that this attack was seen directed at my laptop brings up an important point about the use of a perimeter firewall as your only network defense.

If a remote laptop is infected and then connects to your internal network, the attack is now coming from the inside and your firewall has no effectiveness against it.

7. Evidence of Active Targeting: This is not active targeting against a specific machine, but it is clearly a search for a server running on port 1433/tcp. (Most likely MS-SQL)

8. Severity Ranking: Assigning a value from 0 to 5 for each of the following elements, this attack is ranked as follows:

- **Criticality** = 1 (No systems on this network that are running MS SQL Server. The general purpose of the systems at this location is for personal research and home use.)
- **Lethality** = 5 (If the attack were to be successful, the system would be compromised. Accounts would be made available for re-entry to the system and valuable data about the database structure on the host could be obtained.)

- **System Countermeasures** = 5 (There are no database servers installed on the network and personal firewall software is blocking these packets)
- **Network Countermeasures** = 4 (Perimeter firewall is blocking this traffic)
- **Overall Severity Score = (2 + 5) – (5 + 4) = -2**

9. Defensive Recommendations:

Firewalls should block inbound traffic to 1433/tcp unless absolutely necessary for operations. Unless your operations require connections to this port on remote hosts, the firewalls should also block outbound traffic to 1433/tcp so that if your site does get infected, you can not spread it to other systems. If traffic to 1433/tcp is required then it should be allowed on a host specific basis where possible.

MS SQL Server and/or related software should not be installed without a specific operational need. If this software must be run, the "sa" account should be securely configured and regular security audits should be done to check the current security state. When installing the application, all recommended security guidelines should be followed. These guidelines can be located at Microsoft's website at the following URL(s):

http://www.microsoft.com/sql/techinfo/administration/2000/security.asp
http://www.microsoft.com/sql/evaluation/features/security.asp

All anti-Virus signatures should be kept up to date.

Additional information on assessment of the issue, patches and cleanup is available from CERT at http://www.cert.org/incident_notes/IN-2002-04.html.

10. Multiple Choice Question:

In the following log file output, what is the attacker most likely looking for?

```
=================================================================
   Self Log, (Current system time: Fri, 31 May 2002 10:48:56)
=================================================================

 Date      Time     Action  Source->Destination              Duration Application
2002-05-31 07:27:18 Deny  e.v.i.l:1528->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 07:27:12 Deny  e.v.i.l:1528->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 07:27:09 Deny  e.v.i.l:1528->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 03:56:17 Deny  h.a.c.k:4372->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 03:56:11 Deny  h.a.c.k:4372->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 03:56:08 Deny  h.a.c.k:4372->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 01:44:25 Deny  w.o.r.m:3265->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 01:44:19 Deny  w.o.r.m:3265->a.b.c.d:1433        0 sec                TCP PORT 1433
2002-05-31 01:44:16 Deny  w.o.r.m:3265->a.b.c.d:1433        0 sec                TCP PORT 1433
```

a) Workstations running Windows XP
b) Gopher Servers
c) MS SQL Servers

d) Systems infected with Backdoor Q

Answer: c – MS SQL Server

Detect #3 – Connection Attempts to TCP Port 0

Relevant packets from tcpdump capture

17:36:44.664488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.45955 > 46.5.76.25.0: S [bad tcp cksum b4fa (->adf3)!] 1771847888:1771847888(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:36:47.654488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.45955 > 46.5.76.25.0: S [bad tcp cksum b4fa (->adf3)!] 1771847888:1771847888(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:36:53.654488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.45955 > 46.5.76.25.0: S [bad tcp cksum b4fa (->adf3)!] 1771847888:1771847888(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:05.654488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.45955 > 46.5.76.25.0: S [bad tcp cksum b4fa (->adf3)!] 1771847888:1771847888(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:21.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46104 > 46.5.76.25.0: S [bad tcp cksum 3877 (->3170)!] 1806416559:1806416559(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:24.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46104 > 46.5.76.25.0: S [bad tcp cksum 3877 (->3170)!] 1806416559:1806416559(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:30.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46104 > 46.5.76.25.0: S [bad tcp cksum 3877 (->3170)!] 1806416559:1806416559(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:42.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46104 > 46.5.76.25.0: S [bad tcp cksum 3877 (->3170)!] 1806416559:1806416559(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:53.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46246 > 46.5.76.25.0: S [bad tcp cksum 6735 (->602e)!] 1842645306:1842645306(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:37:56.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46246 > 46.5.76.25.0: S [bad tcp cksum 6735 (->602e)!] 1842645306:1842645306(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:02.914488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46246 > 46.5.76.25.0: S [bad tcp cksum 6735 (->602e)!] 1842645306:1842645306(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:14.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46246 > 46.5.76.25.0: S [bad tcp cksum 6735 (->602e)!] 1842645306:1842645306(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:25.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46380 > 46.5.76.25.0: S [bad tcp cksum f317 (->ec10)!] 1883896412:1883896412(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:28.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46380 > 46.5.76.25.0: S [bad tcp cksum f317 (->ec10)!] 1883896412:1883896412(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:34.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46380 > 46.5.76.25.0: S [bad tcp cksum f317 (->ec10)!] 1883896412:1883896412(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

17:38:46.894488 IP (tos 0x0, ttl 47, id 0, len 52) 211.47.255.22.46380 > 46.5.76.25.0: S [bad tcp cksum f317 (->ec10)!] 1883896412:1883896412(0) win 5840 <mss 1460,nop,nop,sackOK,nop,wscale 0> (DF)bad cksum 667 (->ff5f)!

1. Source of Trace: http://www.incidents.org/logs/Raw/2002.6.7

2. Detect was generated by: tcpdump capture and manual analysis. According to the information at intrusion.org, these captures were triggered by a Snort rule. If that is true, the rule was most likely the rule listed below or a derivative of it. This rule was taken from the bad-traffic.rules file that is included in the snortrules-current.tar that is available for download at http://www.snort.org

```
alert tcp $EXTERNAL_NET any <> $HOME_NET 0 (msg:"BAD TRAFFIC tcp port 0 traffic"; sid:524; classtype:misc-activity; rev:3;)
```

Relevant packets were pulled from the file using tcpdump. Output above was generated with the following command:

```
tcpdump -nvr 2002.6.7 'tcp[2:2]=0'
```

3. Probability the source address was spoofed: Low - These packets were probably the result of user error or a programming mistake, but exploring the possibility that they had a malicious purpose; the possible uses would be an attempt to connect to a service running on port 0 or a host scan. In either case, the packet's sender needs a response to accomplish their goal so the source is not likely spoofed.

The fact that each group of packets sends (4) packets and they are spaced at the 3, 6, 12 TCP back-off intervals further supports the case that the sender was looking for a response but not getting it.

4. Description of Attack: Received 4 groups of packets with the following characteristics:

- IP Version = 4

- IP Header Len = 20 Bytes
- DiffServ Bits = 0
- Total Length = 52 Bytes (20 for IP Header, 32 for TCP Header, 0 Bytes of Data)
- Source IP Address = 211.47.255.22
- IP Identification = 0
- IP Flags = DF (Don't Fragment)
- Fragment Offset = 0
- TTL = 47
- Header checksum does not compute properly (This could indicate crafting or it could be a result of address obfuscation in the tcpdump file)
- Source TCP Port = Ephemeral Ports (Increasing source port numbers for each attempt looks normal)
    - 45955 for first group of packets
    - 46104 for second group of packets
    - 46246 for third group of packets
    - 46380 for fourth and final group of packets
- Destination Address = 46.5.76.25
- Destination TCP Port = 0 (Very unusual. This port is reserved according to TCP/IP standards and should not be used by any service – This is what triggered the Snort Rule)
- SYN flag is set for each packet. This probably indicates one of two things:
    - A connection attempt is being made or
    - The attacker wants to illicit a RST response
- Packets ISN (values look normal – No fixed increase between attempts):
    - 1771847888 for first group of packets
    - 1806416559 for second group of packets
    - 1842645306 for third group of packets
    - 1883896412 for fourth and final group of packets
- Packet has 0 Bytes of data. (This is normal for an initial SYN packet)

At first sight of the destination port 0/tcp, this looks like it might be the work of hping2. A closer look at the timing of the packets, source ports, initial sequence numbers and TCP options show that this is probably not the work of hping2.

- The detect show (4) groups of (4) packets each. Each group comes at different intervals.
    - First packet of first group arrives at 17:36:44 then we see 3 more packets in the group at 3, 6 and 12 second intervals. This is not the default behavior of hping2. Its default is to send packets at 1 second intervals or at some other specific (but fixed) interval. This timing looks like an actual connection attempt from a client. I was able to generate a similar capture by using the command "telnet x.x.x.x 0" command. It sent the first packet and then 3 more on the standard TCP back-off intervals or 3,6,12 seconds.
    - Second Group of packets arrive at 17:37:21 – about 21 seconds after the last packet of group 1

    - Third group arrives at 17:37:53 – about 11 seconds after the last packet of group 2
    - Fourth group arrives at 17:38:25 – about 11 seconds after the last packet of group 3
    - These intervals are closer together but not steady. The 11 second intervals are not enough time to squeeze another 3,6,12 attempt to another host in there so I don't think the delay was due to some automated scanner trying another host before coming back to our host. To me, the timing indicates that these packets were manually initiated and not part of an automated scan or attack attempt.
- The SYN packets have TCP options set. Examination of the source code for hping2 (available at http://www.hping.org/) shows that only the TCP timestamp option is supported in the packets it generates. You may be able to use the RAWIP mode and inject the options manually, but this is probably not what we are looking at. The options being present add further weight to the argument that this is some sort of client attempting a connection to port 0.
- The source ports of these packets (45955, 46104, 46246 and 46380 as shown above in Section 4 – Description of Attack) are normal source ports for a client connecting to a remote host. They are increasing on each attempt and at what looks like a normal rate. Again, an examination of the source code for hping2 shows us that it does not behave in this way. If you do not manually specify the source port, it will be chosen randomly according to the following formula (1024 + rand() % 2000). This formula will always generate an initial source port between 1024 and 3023. Also, the source port stays constant for each of the 4 packets in each grouping. With hping2, unless a fixed port is specified, the source port will increment by one on each successive packet.
- Finally, the initial sequence numbers sent in the packets are not random. They are the same ISN for each packet in a cluster and they look like normally increasing ISN values. Unless a specific TCP sequence number is specified, hping2 generates a random sequence number for each packet it sends.

5. Attack Mechanism (Stimulus / Attempting TCP Connect() to 0/tcp):

Multiple connection attempts to port 0/tcp at regular TCP (3/6/12) retry intervals.

So if it is not hping2, what is sending connection requests to TCP port 0? I believe this traffic is the result of user error or a programming mistake.

- Client configuration error – An improperly configured client application (examples: user typed "telnet 46.5.76.25 0" either on purpose or accidentally, user accidentally set ftp client to connect to port 0, etc …). Some operating systems will not allow this, but others, like Linux, will.
- A poorly written proxy or NAT device that the user is traversing is causing the destination port to go to 0.

If this command were not a mistake on the part of the user, then it was someone either goofing around, trying to force an IDS alert or trying to illicit a RST packet from the host as a very overt technique for checking if the host is alive or not.

Some more remote possibilities for what could be happening here include:

- User has a broken IP stack that is sending traffic to port 0 through programming error. This is unlikely with most of today's operating systems, but it is a possibility.
- User has a custom client that tries to connect to a backdoor listening on tcp port 0 or even a backdoor listening in promiscuous mode (see analysis of Mixter's backdoor Q for more information). I rank this as a very low possibility based on the fact that most IP stacks should not allow a service to bind to this port and modern firewalls would block this traffic. Odds are also low that this is an attempt to communicate with an agent listening in promiscuous mode because there would be no need to send 4 packets on TCP back-off intervals for this kind of message passing. Also, there is no data in the packet so unless it was a trigger based on the packet header signature, there is no message to listen for.
- An unknown or customized scanning tool generated these packets.
- I found reference on the Internet that some IRIX systems can listen on tcp port 0. Although I doubt this is the case, I have not been able to confirm or refute the statement. If the destination host is an IRIX system we should consider this.

Bottom line is this is believed to be fairly benign traffic generated out of user or programming error.

6. Correlations:
A search of June 2002 log files located at http://www.incidents.org/logs/Raw/ turned up this pattern in the following logs.

- 2002.6.3
- 2002.6.5 – 2002.6.7
- 2002.6.9 – 2002.6.17

Outside of these log files and other students' practical posts (which came from the same pool of source files), I have found no other posted detects of this pattern. There are other attack patterns that go to TCP port 0, but I turned up no other posts that matched the signature in the other areas like tcp options, etc.

7. Evidence of Active Targeting: This is clearly active targeting. All packets are all destined for the same host and port.

8. Severity Ranking: It is unknown what types of systems these targets are and without more information or access to the targeted hosts, we have to assume the worst.

- **Criticality = 5** (System importance is unknown so assume the worst)
- **Lethality = 1** (Odds are extremely low that there is a service listening on port 0)
- **System Countermeasures = 3** (The system doesn't appear to be responding to the connect request. This is shown by the 4 packets on 3, 6, 12 second TCP

back-off times. This is still a 3 because we have some element of the unknown to contend with.)

- **Network Countermeasures** = 0 (Unless this detect was captured outside the firewall, then the packets made it into the network. Since network architecture and defenses where this was captured are unknown, we must assume the worst.)
- **Overall Severity Score = (5 + 1) – (3 + 0) = 3**

9. Defensive Recommendations:

- Firewall should block packets with source or destination tcp port equal to 0.
- Inform the target host's system administrator and ask them to check the targeted host just to be sure there really is nothing running on that port or any other anomalous behavior on the system. While we believe this is highly unlikely, due diligence dictates that we should at least have a look.
- Notify the source address owner. They may be able to locate the problem and correct it or, if this really is an attack, shut down the connection.

10. Multiple Choice Question:

In the following tcpdump output, what is special about the timing of the packet arrivals?

16:36:44.664488 IP 211.47.255.22.45955 > 46.5.76.25.0: S 1771847888:1771847888(0) win 5840 <mss1460,nop,nop,sackOK,nop,wscale 0> (DF)
16:36:47.654488 IP 211.47.255.22.45955 > 46.5.76.25.0: S 1771847888:1771847888(0) win 5840 <mss1460,nop,nop,sackOK,nop,wscale 0> (DF)
16:36:53.654488 IP 211.47.255.22.45955 > 46.5.76.25.0: S 1771847888:1771847888(0) win 5840 <mss1460,nop,nop,sackOK,nop,wscale 0> (DF)
16:37:05.654488 IP 211.47.255.22.45955 > 46.5.76.25.0: S 1771847888:1771847888(0) win 5840 <mss1460,nop,nop,sackOK,nop,wscale 0> (DF)

a) Timing indicates a SYN Flood attack
b) Timing indicates a "low and slow" IDS evasion technique
c) Timing indicates normal TCP back-off
d) Timing indicates each packet was individually generated using manual a technique

Answer: c (Timing indicates normal TCP back-off)

Assignment 3: Analyze This – "Peer to Peer Pressure"

Executive Summary

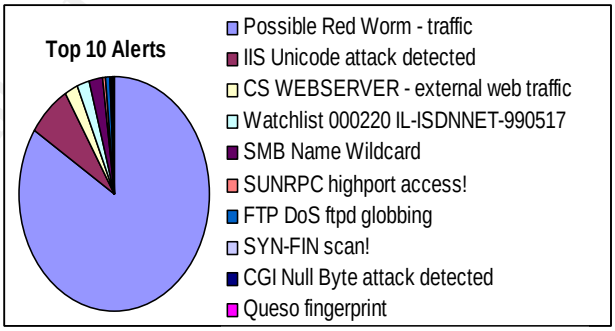
This report represents an analysis of the GIAC University site security based on network Intrusion Detection System logs from the five day period Oct. 10, 2002 – Oct 15, 2002.

Once the details of the log entries were investigated, the majority of the activity appears to be "noise" from Peer to Peer applications, online gaming and a poorly configured IDS rulebase. While these don't represent malicious attacks against University resources, they do raise questions around appropriate use of resources and effectiveness of the IDS logs as a security tool.

In addition to the "noise", there were several log entries that paint a picture of attempted system compromise, DoS attacks, scanning and virus activity. While these can not be confirmed without additional information, there is enough evidence to raise suspicions to a level that warrants further investigation. Where alert entries are suspicious on their own, the analysis and recommendations are provided with the details of the alert. Correlations of events and other general security concerns are detailed in the "High Level Security Issues" section of this report. A high level summary of the alert and scanning activity is also included here for your reference.

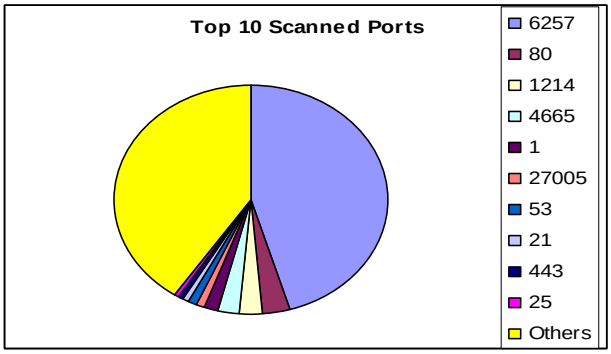
Alerts

There were ***902,201*** alerts generated by the IDS system. The top 10 alerts generated are displayed here. See the "Alert *Ranking by Count and Direction" section for a complete list of alerts.*



Scanning

There were ***2,669,298*** packets that caused scanning alarms to be generated. Most of scanning alerts were false positives generated by Peer to Peer file sharing. Other commonly "scanned" ports were web services (80), tcpmux (1), Counterstrike (27005), DNS (53), FTP (21), HTTPS (443) and TELNET (25). Additional analysis of the scanning data can be found in the ***Scanner Activity*** section of this report.



Recommendations

Where appropriate, specific security suggestions are listed with the detailed analysis of individual alerts or scanning events. Some of these include IDS tuning suggestions to cut down on the number of "noise" related alerts and allow the security staff to better focus on real issues. In addition, high level suggestions for improving overall site security are provided in the "Defensive Recommendations" section of this report.

Data Files Used for This Analysis

Alert Files	Scans	Out Of Spec Captures
Alert.021011.gz	scans.021011.gz	OOS_Report_2002_10_11_21861.txt
Alert.021012.gz	scans.021012.gz	OOS_Report_2002_10_12_29999.txt
Alert.021013.gz	scans.021013.gz	OOS_Report_2002_10_13_9575.txt
Alert.021014.gz	scans.021014.gz	OOS_Report_2002_10_14_21815.txt
Alert.021015.gz	scans.021015.gz	OOS_Report_2002_10_15_13854.txt

Description of Files

The files used for this analysis were obtained from http://www.incidents.org/logs and represent data collected between the dates of October 11th, 2002 and October 15th, 2002.

For each day, there is one file from each of the following three categories:

- Alerts – These represent alerts generated by matching the SNORT signatures.
- Scans – These represent output from Snort's scan pre-processor and are meant to show host and port scanning activity.
- Out Of Spec – These files contain data on captured packets that have strange or invalid characteristics. An example would be a packet with an illegal TCP flag combination such as SYN and FIN set in the same packet.

Some of the entries in the provided files were incomplete or incorrectly formatted. To normalize the data, the files were filtered through regular expressions using PERL script. As the scripts located entries that could not be recovered, the data was separated out of the main data file and placed in a separate log for manual analysis. These unrecoverable entries amounted to less than 1% of the entire sample set and upon manual examination, proved to be immaterial to the analysis process because the information was either redundant, as in the case of a port scan entry that showed up in both Alerts and Scans, or not related to an actual attack.

Data Analysis

Alert Activity

In general, the alerts fell into one of 4 categories. Noise from peer to peer traffic, noise from online gaming, noise from poorly configured or out of data IDS rules and finally, alerts that legitimately signaled possible security issues.

The following sections provide detailed analysis of each alert as well as some ranking based on number of times an alert was seen and the most frequent source and destination IP addresses for the alerts.

Even though several of the top alerts, talkers and destinations are the result of false positives, listing them by number of detects was selected to illustrate the magnitude of the false alerts at the site. Details of all alerts can be found in the "Alert Ranking …" and the "Alert Description and Analysis" sections of this report.

Top 10 Alert Sources

Source IP	Primary Alert Type(s) for Source	Total Count
24.59.33.240	High port 65535 tcp - possible Red Worm - traffic	379887
MY.NET.83.146	High port 65535 tcp - possible Red Worm – traffic High port 65535 udp - possible Red Worm – traffic	371819
MY.NET.85.74	IIS Unicode attack detected	24614
212.179.83.64	Watchlist 000220 IL-ISDNNET-990517	12388
MY.NET.84.133	IIS Unicode attack detected	4571
128.8.120.85	SUNRPC highport access!	3166
152.101.81.195	SYN-FIN scan!	3063
66.77.73.144	CS WEBSERVER - external web traffic	2808
MY.NET.152.22	IIS Unicode attack detected	2583
212.179.97.145	Watchlist 000220 IL-ISDNNET-990517	2029

Top 10 Alert Destinations

Destination IP	Primary Alert Type(s) for Destination	Total Count
MY.NET.83.146	RedWorm x86 setgid x86 setuid	380141
24.59.33.240	RedWorm CS Webserver External Web Traffic	371362
MY.NET.100.165	CS Webserver External Web Traffic CS Webserver External FTP Traffic	21078
207.200.86.66	IIS Unicode Attack	12400
207.200.86.97	IIS Unicode Attack Possible RedWorm	12396
MY.NET.114.88	Watchlist 000220 IL-ISDNNET-990517 NMAP Ping WinVNC	12377
218.55.184.152	IIS Unicode Attack	6893
MY.NET.100.158	FTP DoS GLOB SYN-FIN Watchlist 000220 IL-ISDNNET-990517 x86 setgid(0) IDS552/web-iis_IIS ISAPI Overflow ida nosize IIS Unicode Attack	3744
MY.NET.99.205	SUN RPC Highport Access from 128.8.120.85:22 (normal SSH) (1) SYN-FIN Scan alert	3167
211.115.212.150	IIS Unicode Attack	2247

Alert Ranking by Count and Direction

Attack	Direction of Attack Out->In	In->In	In->Out	Out->Out	Total
High port 65535 tcp - possible Red Worm - traffic	380225	10	371764	5	752004
spp_http_decode: IIS Unicode attack detected	3707		60467		64174
CS WEBSERVER - external web traffic	20923			1	20924
Watchlist 000220 IL-ISDNNET-990517	20005				20005
SMB Name Wildcard	19253				19253
SUNRPC highport access!	5488				5488
FTP DoS ftpd globbing	3735				3735
SYN-FIN scan!	3063				3063
spp_http_decode: CGI Null Byte attack detected	1		2539		2540
Queso fingerprint	2355				2355
IDS552/web-iis_IIS ISAPI Overflow ida nosize	2188				2188
High port 65535 udp - possible Red Worm - traffic	764		1170		1934
Incomplete Packet Fragments Discarded	1015		17		1032
External RPC call	964				964
Watchlist 000222 NET-NCFC	963				963
EXPLOIT x86 NOOP	618				618
Port 55850 tcp - Possible myserver activity - ref. 010313-1	139		163		302
MYPARTY - Possible My Party infection			190		190
connect to 515 from outside	184				184
Null scan!	182				182
Tiny Fragments - Possible Hostile Activity	160				160
CS WEBSERVER - external ftp traffic	149				149
EXPLOIT x86 setuid 0	136				136
IRC evil - running XDCC			123		123
NMAP TCP ping!	104				104
SMB C access	94				94
TCP SRC and DST outside network				68	68
Port 55850 udp – Possible myserver activity - ref. 010313-1	7		41		48
EXPLOIT x86 setgid 0	46				46
Possible trojan server activity	22		21		43
TFTP - Internal UDP connection to external tftp server	26		6		32
External FTP to HelpDesk MY.NET.70.49	17				17
External FTP to HelpDesk MY.NET.70.50	13				13
Bugbear@MM virus in SMTP	8		5		13
TFTP - External TCP connection to internal tftp server	8		5		13
RFB - Possible WinVNC - 010708-1	6		4		10
Attempted Sun RPC high port access	7				7
TFTP - External UDP connection to internal tftp server	6				6
HelpDesk MY.NET.70.49 to External FTP			4		4
HelpDesk MY.NET.70.50 to External FTP			4		4
EXPLOIT NTPDX buffer overflow	3				3
HelpDesk MY.NET.83.197 to External FTP			2		2
ICMP SRC and DST outside network				2	2
Back Orifice	1				1
DDOS TFN client command BE	1				1
DDOS shaft client to handler	1				1
External FTP to HelpDesk MY.NET.83.197	1				1
Fragmentation Overflow Attack	1				1
Probable NMAP fingerprint attempt	1				1
TOTAL ALERT DETECT COUNT					**903201**

Alert Descriptions and Analysis

This section describes each alert type seen in the five day period. The descriptions cover alert classification, log samples of the alert type, a summary of the alert in general and an assessment of the threat with recommendations for the current case. References are also provided where appropriate to help clarify the information covered in each alert.

Alert: Attempted Sun RPC high port access

Log Sample(s):

10/11-19:05:19.036764 [**] Attempted Sun RPC high port access [**] 211.233.25.61:55588 -> MY.NET.84.133:32771
10/12-06:45:07.577690 [**] Attempted Sun RPC high port access [**] 129.6.15.29:37 -> MY.NET.162.67:32771
10/14-07:52:17.726156 [**] Attempted Sun RPC high port access [**] 65.59.116.64:32095 -> MY.NET.151.115:32771
10/15-15:23:22.830211 [**] Attempted Sun RPC high port access [**] 66.28.10.84:0 -> MY.NET.84.198:32771

Summary: Indicates an attempt to connect to Sun RPC portmapper running on port 32771. If a successful connection to an RPC tool can be made, vulnerabilities in these tools may make it possible to execute commands as root via buffer overflow.

Analysis and Recommendations:

The host MY.NET.157.115 may be compromised. There is a series of alerts surrounding this host that could indicate the traffic from 65.59.116.64 was part of an attack. A similar pattern is also seen for host MY.NET.84.198. See the "High Level Security Issues" section of this report for more information and a diagram of the attack.

The last log entry shown above is also curious. The source port is 0, this not normal behavior. Our logs had no other scanning attempts from this host, indicating that we are not seeing noise from a massive scan. The hostname returned from a DNS reverse lookup indicates that the target machine is probably located in an engineering building. That means it may very well be a Sun workstation. This combined with the individual targeting means the alert should be taken seriously and the machine should be checked for any signs of exploit.

The alert source address (66.28.10.84) did not resolve with NSLOOKUP, but using visualroute.com turned up the name ntbs4.jumptv.com. A search for this name on Google revealed the url: http://ntbs4.jumptv.com/thai-hi-safety in another site's posted proxy logs. Connecting to this turned up a Thai video cast. This looks like some kind of TV media server.

The alerts from 129.6.15.29 are harmless replies to MY.NET.162.67 from a NIST time.

In general, externally initiated traffic to 32771 should be blocked at the firewall unless there is a very strong business case for allowing it. RPC services should also be disabled where they are not needed.

IDS Suggestion: These alerts appear to be generated based on port number alone. The IDS rules should be upgraded to look for RPC content in addition to the port number.

Reference(s):

- http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0189

Alert: Back Orifice

Log Sample(s):

10/14-23:32:48.168415 [**] Back Orifice [**] 63.250.205.9:5669 -> MY.NET.152.17:31337

Summary: Back Orifice is a Backdoor that allows remote control of Windows Machines.

Analysis and Recommendations:

This alert doesn't guarantee a compromised system, but since there was no other detect of this type in the logs, it does not appear to be peer to peer traffic or someone trolling around for instances of BO. One packet to one host on that particular port looks targeted and should raise suspicions. This could be a wake up call for a previously planted backdoor. Without additional information about the hosts, MY.NET.152.17 should be checked.

IDS Suggestion: This rule appears to be based on port number only. Replace it with the newer Back Orifice preprocessor.

Reference:

- ISS Alert – http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise5
- CAN-1999-0660

Alert: Bugbear@MM virus in SMTP

Log Sample(s):

10/11-00:51:59.344562 [**] Bugbear@MM virus in SMTP [**] 195.92.193.19:3736 -> MY.NET.6.40:25
10/11-17:16:14.341249 [**] Bugbear@MM virus in SMTP [**] MY.NET.6.40:42295 -> 65.212.73.209:25

Summary: Worm with keystroke logging and backdoor capabilities. It can disable some Anti-Virus and Personal Firewall Software. It spreads via email and unprotected file shares. Signs of infection include connection attempts to other systems file shares and erratic behavior from LAN printers as it tries to connect to them as file shares.

Analysis and Recommendations:

There were five instances of this alert showing mail leaving the University. This indicates that a host on the internal network has been exposed. If possible mail headers in the mail server logs should be reviewed around the time of the alerts. This may give some clues useful to track down the source of the infected mail messages. This could be useful towards finding any infected machines.

There were other signs of bugbear attack and possible compromise related to "SMB Name Wildcard" and "SMB C access" alerts. All Anti-Virus software should be updated.

IDS Suggestion: The best rule I could find for this comes from Shane Williams. If we are using a less capable rule in our configuration, we should consider upgrading to the following rule:

```
alert tcp any any -> any 25 (msg:"Bugbear@MM virus in SMTP";
content:"uv+LRCQID7dIDFEECggDSLm9df8C/zSNKDBBAAoGA0AEUQ+FEN23f7doqAT/dCQk/xWc
EQmDxCTD"; sid:900001; classtype:misc-activity; rev:1;)
```

Reference:

- http://securityresponse.symantec.com/avcenter/venc/data/w32.bugbear@mm.html
- http://vil.nai.com/vil/content/v_99728.htm
- http://www.sophos.com/virusinfo/analyses/w32bugbeara.html
- Williams, Shane - http://www.mcabee.org/lists/snort-users/Oct-02/msg00067.html

---

Alert: CS WEBSERVER – external ftp traffic

Log Sample(s):

10/13-10:29:40.669698 [**] CS WEBSERVER - external ftp traffic [**] 213.156.56.135:3828 -> MY.NET.100.165:21

Summary: Records ftp connection events to a particular internal FTP server.

Alert: CS WEBSERVER - external web traffic

Log Sample(s):

10/13-12:08:15.497058 [**] CS WEBSERVER - external web traffic [**] 66.196.73.14:35505 -> MY.NET.100.165:80
10/13-12:08:15.957945 [**] CS WEBSERVER - external web traffic [**] 66.77.73.236:2686 -> MY.NET.100.165:80

Summary: Records http connections to a particular internal web server.

Alert: DDOS TFN client command BE

Log Sample(s):

10/13-13:07:10.944921 [**] DDOS TFN client command BE [**] 130.132.252.244 -> MY.NET.140.9

Summary: DDoS Tool capable of ICMP/SYN/UDP flood and Smurf attacks. TFN can also have backdoor capabilities. It uses ICMP echo reply messages for communications.

Analysis and Recommendations:

This is a false positive. Both the source and destination hosts for this alert entry are part of the Active Measurement Project (AMP) project run by the National Laboratory for Applied Network Research (NLANR). This application generates ping (i.e. icmp) and traceroute traffic to measure network statistics.

The alert source (130.132.252.244) is an AMP server (amp.its.yale.edu) at Yale U.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 228. If desired, pass rules can be configured for the AMP servers.

Reference(s):

- Dittrich, David. The "Tribe Flood Network" distributed denial of service attack tool. Oct. 21st, 1999 <http://staff.washington.edu/dittrich/misc/tfn.analysis>
- Treurniet, Joanne, September 22, 2000 SANS GIAC Practical Assignment for GCIA, <http://www.giac.org/practical/JoanneTreurniet.html>
- AMP Information Website - http://sd.wareonearth.com/woe/amp.htm
- NLANR Website - http://moat.nlanr.net/
- arachNIDS IDS184 - http://www.whitehats.com/IDS/184

Alert: DDOS shaft client to handler

Log Sample(s):

10/11-15:08:49.655655 [**] DDOS shaft client to handler [**] 205.188.165.121:80 -> MY.NET.60.88:20432

Summary: Shaft is a DDoS Tool similar to TFN (*see "DDOS TFN client command BE"*)

Analysis and Recommendations:

Examination of the source port for the sending host shows that this is return traffic from a web server. There was a single entry in the alert files for this detect.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 230.

Reference(s):

- http://security.royans.net/info/posts/bugtraq_ddos3.shtml
- arachNIDS IDS254 - http://www.whitehats.com/IDS/254

Alert: EXPLOIT NTPDX buffer overflow

Log Sample(s):

10/14-09:32:18.166442 [**] EXPLOIT NTPDX buffer overflow [**] 195.92.252.254:123 -> MY.NET.111.11:123
10/14-11:48:27.329843 [**] EXPLOIT NTPDX buffer overflow [**] 64.7.192.181:52779 -> MY.NET.88.164:123
10/15-11:28:47.733978 [**] EXPLOIT NTPDX buffer overflow [**] 216.148.215.98:123 -> MY.NET.117.25:123

Summary: Buffer Overflow attack against Network Time Protocol daemon that could yield root access. This attack can use a spoofed source address.

Analysis and Recommendations:

This is probably streaming media traffic. The alert sources are all Media Servers and the destinations are all systems that have hostnames registered in DNS that look like they are in shared areas of the University campus. This probably indicates that they are running Windows OS and are not vulnerable to this NETDX buffer overflow. If these hosts are running NTP services then check the version for vulnerabilities. If they exist, fix them and check for previous exploits. In general, NTP from the outside should be controlled

through a hierarchy architecture where only one or two devices get NTP data from external sources. Internal hosts should use authenticated NTP services.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 312.

Reference(s):

- arachNIDS IDS492 - http://www.whitehats.com/IDS/492
- BugTraq 2540 http://www.securityfocus.com/bid/2540
- Media Server example from log entry above: mms://**195.92.252.254**/jazzfmstation

---

Alert: EXPLOIT x86 NOOP

Log Sample(s):

```
10/11-16:45:34.912390  [**] EXPLOIT x86 NOOP [**] 131.118.254.38:80 -> MY.NET.116.80:4533
10/11-18:34:23.602766  [**] EXPLOIT x86 NOOP [**] 207.172.2.141:46010 -> MY.NET.162.101:16099
10/12-20:27:06.383816  [**] EXPLOIT x86 NOOP [**] 80.15.150.192:3562 -> MY.NET.116.47:35999
10/15-13:47:07.008272  [**] EXPLOIT x86 NOOP [**] 211.157.248.47:3952 -> MY.NET.162.91:1251
10/15-22:01:54.813420  [**] EXPLOIT x86 NOOP [**] 66.156.43.11:1771 -> MY.NET.139.10:1906
```

Summary: Indicates a pattern matching the Intel x86 NoOP instruction was detected in the data portion of the packet. This can signal an attempted buffer overflow attack or it can occur naturally in the transfer of graphics and other binary files. If the traffic is being returned from a web server or ftp server then the traffic may not be malicious as in the case of the first log sample listed above.

Analysis and Recommendations:

Examining the port numbers, frequency and timing of the alerts, it is believed that these are noise from binary transfers involving web sites, ftp sites, or peer to peer file sharing.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 684.

Reference(s):

Oborn, David - SANS GCIA Practical Assignment
<http://www.giac.org/practical/David_Oborn_GCIA.html#detect4>

Alert: EXPLOIT x86 setuid 0

Log Sample(s):

```
10/12-13:49:39.479454  [**] EXPLOIT x86 setuid 0 [**] 134.30.102.18:4774 -> MY.NET.185.48:6346
```

Summary: Attempt to change effective user id on system running on x86 architecture in an effort to gain higher access privileges.

Analysis and Recommendations:

The Snort rule that detects this exploit looks for the setuid(0) call ("|b017 cd80|"). This can appear in normal binary transfer and often generate false positives. All of these alerts appear to involve traffic where this kind of normal binary traffic could occur.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 650. Carefully define the $SHELLCODE_PORTS variable in the SNORT configuration.

Reference(s):

- Fiddler, Matthew – May 10th, 2002 - Intrusion Detection In Depth - GCIA Practical Assignment - Version 3.0 - <http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc>
- Whitehats - 2001 - "IDS436 "SHELLCODE-X86-SETUID0-UDP" - http://www.whitehats.com/IDS/436

Alert: EXPLOIT x86 setgid 0

Log Sample(s):

10/12-11:16:48.326690 [**] EXPLOIT x86 setgid 0 [**] 24.52.56.240:2789 -> MY.NET.70.176:6699
10/15-15:21:07.933370 [**] EXPLOIT x86 setgid 0 [**] 216.135.160.48:52274 -> MY.NET.137.66:9000

Summary: Attempt to change effective group id on system running on x86 architecture in an effort to gain higher access privileges.

Analysis and Recommendations:

Like the "Explout x86 setuid 0" alert above, this alert is often the result of false positives from binary data transfers. The majority of this traffic is to ports associated with file sharing services. The last entry above is to port 9000/tcp, a default port for IBM's WebSphere server. Web servers are a common source of false positives for this rule.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 649. Carefully define the $SHELLCODE_PORTS variable in the SNORT configuration.

Reference(s):

Whitehats, 2001 - IDS284 "SHELLCODE-X86-SETGID0" http://www.whitehats.com/IDS/284

Alert(s): External FTP to HelpDesk MY.NET.70.49
External FTP to HelpDesk MY.NET.70.50
External FTP to HelpDesk MY.NET.83.197

Log Sample(s):

10/12-09:48:43.695693 [**] External FTP to HelpDesk MY.NET.70.49 [**] 62.123.114.218:3791 -> MY.NET.70.49:21
10/11-19:39:15.317895 [**] External FTP to HelpDesk MY.NET.70.50 [**] 66.200.47.182:2855 -> MY.NET.70.50:21
10/15-19:19:36.443218 [**] External FTP to HelpDesk MY.NET.83.197 [**] 209.240.169.251:1601->MY.NET.83.197:21

Summary: This rule is specifically to log FTP access to the helpdesk servers.

Alert: External RPC call

Log Sample(s):

10/12-14:08:28.310028 [**] External RPC call [**] 210.46.90.254:57400 -> MY.NET.135.9:111
10/12-17:38:07.606700 [**] External RPC call [**] 24.123.46.10:2328 -> MY.NET.190.92:111

Summary: This is an attempt to locate RPC services running on a host. If a response is sent from the host then the attacker can attempt to exploit a number of well documented vulnerabilities in the RCP services.

Analysis and Recommendations:

The source address 24.123.46.10 in the alerts above appears on DeScan.com as a known scanner (see reference below). This IP scanned a large set of the internal address space for port 111 and should be watched carefully.

210.46.90.254 also appeared as a source for this alert. This address is coming from Harbin Medical College in mainland China. The log activity clearly shows that this IP is also the source of a scan for this service. It and other addresses from that University should be watch for further portmapper or other suspicious traffic.

Unless there is a strong organizational reason to allow access to port 111 from external sources, it should be blocked. RPC services should be disabled where possible.

IDS Suggestion: The rule that generated these alerts appears to be based on port alone. It should be replaced by more current Snort rules that also examine content. In this case, we detected the scan, but these kinds of "port only" rules can lead to false positives.

Reference(s):

- http://www.wjsolutions.com/scanner/?curpage=SummaryScan
- http://www.descan.net/searchresults.html?command=specific&source=24.123.46.10

Alert: FTP DoS ftpd globbing

Log Sample(s):

10/15-05:16:06.636217 [**] FTP DoS ftpd globbing [**] 80.11.105.159:2461 -> MY.NET.100.158:21

Summary: This attack works by sending commands with wildcards in it in hopes that when the expression is expanded, it overflow the heap on the target and cause a denial or service or allow arbitrary code execution.

Analysis and Recommendations:

The log entries show 3736 entries for this alert from various Internet connections in Europe. These do appear to be DoS or heap overflow attempts because of the timing and frequency of the alerts. When the alerts are happening, they are fairly continuous occurring every 3-15 seconds.

The source IP would not be spoofed because the attack requires a TCP session to be established and hijacking is not really an option for the duration and number of packets sent. The source addresses in the alerts are all from ISPs in Brussels, France and Germany. These addresses should be watched for further activity. The ftp server software should be checked for vulnerability to these kinds of globbing attacks. The

target ftp host is also a DNS server. There could be a connection between this and the reason it was selected as a DoS target.

This attack also requires a login to the ftp server to work. The logs should be examined to find what account was used for the authentication. This ID, if not anonymous, should have a mandatory password change. If anonymous, consider disabling this feature.

Reference(s):

- Allen, Jennifer - WU-FTPD Heap Corruption Vulnerability - GCIH Practical Assignment v2.0. Dec. 2001. < www.giac.org/practical/Jenn_Allen_GCIH.doc>
- CERT - CA-2001-07 - <http://www.cert.org/advisories/CA-2001-07.html>
- Whitehats - IDS487 "*DOS-FTPD-GLOBBING*" – http://www.whitehats.com/info/IDS487

Alert: Fragmentation Overflow Attack

Log Sample(s):

10/14-05:33:15.354479 [**] Fragmentation Overflow Attack [**] 219.165.170.64:0 -> MY.NET.24.44:0

Summary: I could find no specific references to this particular alert other than the source code for spp_degrag.c. From a look at the pre-processor source code, I believe this alert signals when fragments are received beyond the final fragment's offset + size.

Analysis and Recommendations:

There were other fragment related interactions between these two hosts. Based on the DNS name returned from a reverse lookup, the target appears to be a web server that holds student web pages. This kind of high profile host is often the target of scans and attacks. Watch the source IP and further investigate our internal target for signs of possible compromise.

Reference(s): Ruiu, Dragos – spp_defrag.c – Snort defrag pre-processor source code

---

Alert(s):

HelpDesk MY.NET.70.49 to External FTP
HelpDesk MY.NET.70.50 to External FTP
HelpDesk MY.NET.83.197 to External FTP

Log Sample(s):

10/12-09:50:08.471092 [**] HelpDesk MY.NET.70.50 to External FTP [**] MY.NET.70.50:4123 -> 161.69.201.237:21
10/12-12:35:13.033398 [**] HelpDesk MY.NET.70.49 to External FTP [**] MY.NET.70.49:2170 -> 161.69.201.238:21
10/15-08:47:19.523967 [**] HelpDesk MY.NET.83.197 to External FTP [**] MY.NET.83.197:1041 -> 161.69.201.238:21

Summary: Logs FTP access from the helpdesk servers to external FTP servers.

Analysis and Recommendations:

There are 10 entries of this type in the alert logs. 9 of them are to support ftp servers at Network Associates so this looks like legitimate attempts to get support files from NAI. 1

entry was to a Comcast address. The reason for this entry is unknown but does not appear hostile.

Alert: High port 65535 tcp - possible Red Worm - traffic

Log Sample(s):

10/12-16:15:22.218202 [**] High port 65535 tcp - possible Red Worm - traffic [**] 24.59.33.240:65535 -> MY.NET.83.146:1379
10/12-16:15:22.218501 [**] High port 65535 tcp - possible Red Worm - traffic [**] MY.NET.83.146:1379 -> 24.59.33.240:65535

Summary: Code Red (a.k.a. Adore) is a Worm that attacks vulnerable services and then installs a backdoor listener as trojaned klogd on 65535/tcp for later use. This can indicate that a system has been compromised.

Analysis and Recommendations:

MY.NET.83.146 should be investigated for signs of Adore because it had interactions with 214.66.75.113, a host on the telia.com network. There are reports on the Internet at http://www.rud.dk of hosts from the telia.com domain contacting their web server with an Adore and Adorev2 signature.

Based on traffic pattern analysis and the host list involved in the traffic, the remainder of these alerts appears to be file transfer, authentication and peer to peer traffic.

Reference(s):

- SANS - Adore Worm - Version 0.8 - April 12, 2001 <http://www.sans.org/y2k/adore.htm>
- Fiddler, Matthew – May 10th, 2002 - Intrusion Detection In Depth - GCIA Practical Assignment - Version 3.0 - <http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc>

Alert: High port 65535 udp - possible Red Worm - traffic

Log Sample(s):

10/12-08:38:34.943203 [**] High port 65535 udp - possible Red Worm - traffic [**] 61.26.132.120:65535 -> MY.NET165.24:6257
10/12-08:39:34.014950 [**] High port 65535 udp - possible Red Worm - traffic [**] 68.97.54.51:65535 -> MY.NET.15.24:6257
10/12-08:47:29.496730 [**] High port 65535 udp - possible Red Worm - traffic [**] MY.NET.70.176:6257 -> 68.97.5.51:65535

Summary: Worm that attacks vulnerable services and then installs a backdoor listener as trojaned klogd on 65535/udp for later use. This can indicate that a system has been compromised. This worm is also known as Adore. It is not “Code Red”.

Analysis and Recommendations:

Possible Adore issue related to MY.NET.83.146. See Analysis of “High port 65535 tcp - possible Red Worm – traffic” for more information.

MY.NET.104.178 was looking for SLP services by sending multicast packets to 239.255.255.253 port 427/udp. This kind of false positive is common for rules like this. The remainder of the traffic is believed to be primarily the work of WinMX and other Peer to Peer file sharing applications.

Reference(s):

- SANS - Adore Worm - Version 0.8 - April 12, 2001 <http://www.sans.org/y2k/adore.htm>
- Fiddler, Matthew – May 10th, 2002 - Intrusion Detection In Depth - GCIA Practical Assignment - Version 3.0 - <http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc>

Alert: ICMP SRC and DST outside network

Log Sample:

10/13-17:56:55.171009 [**] ICMP SRC and DST outside network [**] 10.52.188.55 -> 169.254.10.22

Summary: ICMP packets from external source to external destination were detected.

Analysis and Recommendations:

Most likely the result of an improperly configured host or an IP spoof from inside the network. Properly configured routers will keep this on the local segment.

The 169.254.10.22 destination address is from "Link Local" range defined in RFC 3330. This range is commonly used operating systems when they do not have a static IP and can not get a DHCP response. This could indicate an improperly configured host, a switch port being in the wrong VLAN and not getting DHCP or improperly configured route that is sending packets to this address range to the IDS sensor. Knowledge of the IDS sensor may help us better understand these alerts.

IDS Suggestion: Consider the value of this rule. It would be better to eliminate the rule and perform ingress / egress filtering for these kinds of packets.

Alert: IDS552/web-iis_IIS ISAPI Overflow ida nosize

Log Sample(s):

10/15-23:18:26.047102 [**] IDS552/web-iis_IIS ISAPI Overflow ida nosize [**] 61.179.120.45:4839 -> MY.NET.21.114:80

Summary: Signals attempted exploit of Microsoft IIS Index Server with data payloads larger than the acceptable length (> 239 Bytes). The goal is to exploit an unchecked buffer and execute arbitrary code. This is a sign of worms like Code Red or NIMDA.

Analysis and Recommendations:

Alert the destination system owners and ensure that any running web servers are patched or that the Indexing Services are disabled.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 1243

Reference(s):

- http://www.incidents.org/react/code_redII.php
- http://www.incidents.org/react/nimda.pdf

- Whitehats - IDS552 "IIS ISAPI OVERFLOW IDA" - http://www.whitehats.com/info/IDS552
- Ellis, Joe – GCIA Practical Assignment v3.0, Intrusion Detection In Depth – May 14th, 2002 - <http://www.giac.org/practical/Joe_Ellis_GCIA.doc>
- Poor, Mike - Intrusion Detection in Depth. GCIA Practical Assignment, v3.0 - <http://www.giac.org/practical/Mike_Poor_GCIA.doc>

Alert: IRC evil - running XDCC

Log Samples(s):

10/11-04:00:18.915628 [**] IRC evil - running XDCC [**] MY.NET.100.220:1232 -> 66.250.105.173:6667
10/15-00:59:57.756002 [**] IRC evil - running XDCC [**] MY.NET.100.220:4584 -> 64.45.60.200:6667

Summary: Detection of IRC file distribution agent activity.

Analysis and Recommendations:

All entries in logs were from a single internal source (MY.NET.100.220). While destination port 6667 is commonly used by IRC, it has also been associated with backdoors including WinSatan, ScheduleAgent, SubSeven, and a host of others. This could be the result of IRC use, a XDCC bot on the host uploading files to the IRC channel or a backdoor application. Host should be checked for signs of exploit. If you can connect to any IRC channels at the destination sites, you may be able to see if your internal IP shows up on the site as a file share.

If this is an official host, one way to prevent this is to specify what services the host can connect to from your network. An example of this is not allowing your DMZ web servers to initiate any outbound request through your firewall. This prevents some attacks from completing and limits the usefulness of the boxes in case they do get “0wn3d”.

IDS Suggestion: The best rule pair I could find for this comes from Christopher Cramer and is listed below.

```
alert tcp any any -> any 6667 (msg:"IRC evil - running XDCC"; content:"To
request a file type"; nocase;)

alert udp any any -> any 6667 (msg:"IRC evil - running XDCC"; content:"To
request a file type"; nocase;)
```

Reference(s):

- TonikGin – XDCC – An .EDU Admin’s Nightmare” – September 11th, 2002 - <http://www.russonline.net/tonikgin/EduHacking.html>
- Pest Patrol – About Ports and Trojans – 2002 – http://www.pestpatrol.com/Support/About/About_Ports_And_Trojans.asp#portlist
- Cramer, Christopher - http://www.theorygroup.com/Archive/Unisog/2002/msg00677.html

---

Alert: Incomplete Packet Fragments Discarded

Log Sample(s):

```
10/11-00:24:26.929947  [**] Incomplete Packet Fragments Discarded [**] 216.54.222.175:0 -> MY.NET.53.36:0
10/14-16:01:01.567200  [**] Incomplete Packet Fragments Discarded [**] 202.102.233.93:0 -> MY.NET.163.235:0
10/14-15:52:07.874446  [**] Incomplete Packet Fragments Discarded [**] MY.NET.83.189:0 -> 68.48.137.49:0
```

Summary: This alert indicates that Snort's defrag pre-processor has given up on rebuilding a packet because the final fragment was received but it had not yet received enough of the original datagram. By current default, it needs to have 50% of a packet larger than 8K when the final fragment is received or this message will be generated. This does not mean that the packets did not get to the destination address, just that the Snort pre-processor gave up on them.

Analysis and Recommendations:

In the complete logs, there are a large number of these alerts coming from external source addresses directed to internal network hosts. This may be an indication of a DoS attack against the internal hosts using fragments that never complete. Examination of the logs shows several packets per minute which would be indicative of a buffer starvation attempt on the part of the attacker.

Of particular interest are the packets from 219.165.170.64 to MY.NET.24.44. There were also several alerts of Fragment Overflow attacks related to these two addresses. See "Fragmentation Overflow Attack" above.

Reference(s):

- Peck, Edward - GCIA Practical Version 3.0 – 2001 - <http://www.giac.org/practical/Edward_Peck_GCIA.doc>
- Ruiu, Dragos – Feb. 12th, 2002 - http://archives.neohapsis.com/archives/snort/2001-02/0320.html

---

Alert: MYPARTY - Possible My Party infection

Log Sample(s):

```
10/13-22:43:53.114667  [**] MYPARTY - Possible My Party infection [**] MY.NET.87.185:2869 -> 209.151.250.170:80
10/13-22:43:53.114989  [**] MYPARTY - Possible My Party infection [**] MY.NET.87.185:2869 -> 209.151.250.170:80
```

Summary: Backdoor that spreads via email worm.

Analysis and Recommendations:

It appears that an old copy of the MYPARTY virus has infected MY.NET.87.185 because it tried several times to make a connection to the web site 209.151.250.170. This site was where the original instructions for the virus were held (this site has since been shutdown). The reason this is believed to be an old copy and not a new infection is that MYPARTY was only programmed to spread from January 25th to January 29th, 2002.

Check MY.NET.87.185 for signs of infection and ensure all anti-Virus signatures are up to date on this host.

Reference(s):

- http://its.marquette.edu/virus/alert.html
- Symantec – http://securityresponse.symantec.com/avcenter/venc/data/w32.myparty@mm.html

---

Alert: NMAP TCP ping!

Log Sample(s):

10/11-10:16:37.939451 [**] NMAP TCP ping! [**] 12.39.160.31:80 -> MY.NET.83.201:6346
10/11-11:46:27.290685 [**] NMAP TCP ping! [**] 67.36.84.5:80 -> MY.NET.185.48:6346
10/11-12:10:37.496055 [**] NMAP TCP ping! [**] 62.0.34.130:80 -> MY.NET.83.201:6346

Summary: Detection of NMAP Tool Being Used to Scan Hosts. This alert happens when an ICMP echo request arrives with 0 bytes of data in its payload, a signature of NMAP.

Analysis and Recommendations:

Block inbound ICMP echo request if possible. If possible, you should also configure your firewall to simulate and maintain state for outbound ICMP echo requests so that only solicited replies can enter your network.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 469

Reference(s):

Stearns, William - http://www.sans.org/y2k/practical/william_stearns_gcia.html

---

Alert: Null scan!

Log Sample(s):

10/11-06:30:15.215561 [**] Null scan! [**] 24.112.24.50:1 -> MY.NET.179.180:1524
10/15-06:59:12.151802 [**] Null scan! [**] 195.46.65.63:0 -> MY.NET.185.48:0

Summary: TCP packets with no tcp flags set were detected. This is a signature of a scanning attempt. This can also be used in OS fingerprinting attempts.

Analysis and Recommendations:

Because of their destination ports, there were 2 log entries of interests. These were headed to port 1524, a known port for the backdoor ingresslock. This could represent someone looking for a previously planted backdoor. MY.NET.179.180 should be checked for any signs of compromise.

There were also several entries where the source and destination port were 0, this could indicate the use of hping2.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID 623

Reference(s):
Chappell, Laura – "Your Being Watched" – March 2001 –
<http://www.nwconnection.com/2001_03/pdf31/cybercrm31.pdf>

Alert: Port 55850 tcp - Possible myserver activity - ref. 010313-1

Log Sample(s):

```
10/11-19:23:03.338610  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] MY.NET.6.40:55850 ->  209.236.57.18:25
10/12-00:11:21.807189  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] 216.137.26.131:55850 -> MY.NET.185.48:6346
10/12-00:11:21.807539  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] MY.NET.185.48:6346 ->216.137.26.131:55850
10/12-23:27:16.268836  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] 134.59.10.172:55850->MY.NET.111.214:4662
10/12-23:27:16.376502  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] MY.NET.111.214:4662->134.59.10.172:55850
10/14-23:45:09.824842  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] MY.NET.6.40:55850-> 63.251.244.194:113
10/15-08:21:39.459997  [**] Port 55850 tcp-Possible myserver activity-ref. 010313-1 [**] 66.180.244.27:55850-> MY.NET.179.78:25
```

Summary: Myserver is a DDOS tool that uses port 55850. These alerts are an indication that traffic using this port was detected.

Analysis and Recommendations:

Examination of the port used for the other side of these conversations shows that all alerts of this type were the result of peer to peer traffic or other normal traffic using 55850 as the ephemeral port.

NOTE: This is TCP traffic and all reference information found on the Internet indicates that MyServer uses UDP only. I am not sure if this is an oversight on my part or a rule error in the IDS.

IDS Suggestions: Consider the validity of this rule. If this rule is not needed or can not be improved, perhaps it should be dropped for clarity in the alert logs.

Reference(s): Lam, Jason - Intrusion Detection in Depth – GCIA Practical Assignment Version 2.9. <www.giac.org/practical/Jason_Lam_GCIA.doc>

Alert: Port 55850 udp - Possible myserver activity - ref. 010313-1

Log Sample(s):

```
10/14-02:25:57.857978  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] MY.NET.104.66:55850 -> 239.255.255.253:427
10/12-16:50:51.445109  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] MY.NET.188.24:55850-> 10.0.1.1:192
10/12-18:23:48.383012  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 137.78.21.22:55850 -> MY.NET.140.9:33485
10/12-20:07:28.208257  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 128.182.61.50:55850-> MY.NET.140.9:33474
10/12-20:07:28.231847  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 128.182.61.50:55850-> MY.NET.140.9:33475
10/12-20:07:28.255228  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 128.182.61.50:55850-> MY.NET.140.9:33476
10/13-06:29:50.915672  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] MY.NET.188.24:55850-> 10.0.1.1:192
10/15-01:06:30.005936  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] MY.NET.140.9:55850 -> 134.75.30.5:33466
10/15-05:31:22.395664  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] MY.NET.90.118:55850-> 10.0.1.1:192
10/15-10:07:36.946014  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 63.250.205.22:46173-> MY.NET.163.125:55850
10/15-10:10:28.009618  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 63.250.205.22:51412-> MY.NET.177.61:55850
10/15-20:13:28.234253  [**]Port 55850 udp-Possible myserver activity-ref.010313-1[**] 211.233.58.229:59717>MY.NET.153.159:55850
```

Summary: Myserver is a DDOS tool that uses port 55850. These alerts are an indication that traffic using this port was detected.

Analysis and Recommendations:

Further investigation of hosts MY.NET.163.125, MY.NET.177.61 and MY.NET.153.159 using scanning tools is recommended to determine if there is an active listening agent on port 55850/udp of these hosts.

Other traffic appears non-malicious - The traffic to 239.255.255.253 port 427 is related to the Service Locator Protocol and is not malicious. Likewise, the traffic to and from MY.NET.140.9 is not malicious. This is traceroute traffic form the AMP application running there (for information on AMP, see information on "DDOS TFN client command BE" alert above). The traffic to 10.0.1.1:192 is probably an attempt to locate a wireless access point (see info at http://archive.dbforums.com/181/2002/9/497520 for reference)

IDS Suggestions: Consider the validity of this rule. There were several alerts and we don't know if any of them are legitimate. If this rule can not be improved, perhaps it should be dropped. Pass rules should be installed for the AMP servers to reduce alerts.

Reference(s): Lam, Jason - Intrusion Detection in Depth – GCIA Practical Assignment Version 2.9. <www.giac.org/practical/Jason_Lam_GCIA.doc>

Alert: Possible trojan server activity

Log Sample(s):

```
10/13-04:05:08.249704  [**] Possible trojan server activity [**] MY.NET.179.77:80 -> 211.28.96.9:27374
10/13-04:05:08.480481  [**] Possible trojan server activity [**] 211.28.96.9:27374 -> MY.NET.179.77:80
10/13-05:22:13.255792  [**] Possible trojan server activity [**] 213.84.22.18:27374 -> MY.NET.84.147:1214
10/13-05:22:13.256202  [**] Possible trojan server activity [**] MY.NET.84.147:1214 -> 213.84.22.18:27374
10/14-05:30:33.272875  [**] Possible trojan server activity [**] MY.NET.25.21:143 -> 4.35.54.173:27374
10/14-05:30:33.361151  [**] Possible trojan server activity [**] 4.35.54.173:27374 -> MY.NET.25.21:143
10/14-09:21:35.815883  [**] Possible trojan server activity [**] MY.NET.83.201:6346 -> 213.177.137.33:27374
10/14-09:21:35.815963  [**] Possible trojan server activity [**] MY.NET.83.201:6346 -> 213.177.137.33:27374

10/15-14:07:51.188995  [**] Possible trojan server activity [**] 138.16.135.1:27374 -> MY.NET.116.68:7625
10/15-14:07:51.189114  [**] Possible trojan server activity [**] MY.NET.116.68:7625 -> 138.16.135.1:27374
10/15-14:07:54.656084  [**] Possible trojan server activity [**] 138.16.135.1:27374 -> MY.NET.116.68:7625
```

Summary: This alerts represent traffic to or from port 27374, a common port for backdoor applications like SubSeven or Ramen worm.

Analysis and Recommendations:

In the alert files, there were two kinds of entries for this alert. The first set of log entries above represent harmless traffic that just happened to use 27374 as the ephemeral port when connecting to a web server, KaZaa, an IMAP email server and Gnutella.

The last three entries are more suspicious because I can't associate the port 7625 with anything. This appears to be a custom port for an application (possibly file sharing) on the host, but without more data, we should watch the host's activities to see if they are controlling a backdoor on the external host. Update the Snort rules to include content.

IDS Suggestion: Rule that generated the alert above seems to fire on traffic to/from port 27374 alone. Upgrade to rules Snort ID 103 and 514, they are more specific and include content checking.

Reference(s): Graham, Robert http://www.robertgraham.com/pubs/firewall-seen.html

---

Alert: Probable NMAP fingerprint attempt

Log Sample(s):

10/13-23:04:48.979396 [**] Probable NMAP fingerprint attempt [**] 203.46.103.130:60448 -> MY.NET.53.8:23

Summary: This detects a packet with an invalid TCF flag combination of URG, PSH, SYN and FIN.

Analysis and Recommendations:

This single entry was directed at port 23 which is used for telnet services. The logs also show other scan packets from the same source to the same destination. These included an NMAP TCP ping, a NULL Scan and two SYN packets. This is not legitimate traffic and represents a recon attempt against the host. It is not harmful in and of itself, but it could be the start of something more dangerous. Consider adding the source to a watch list.

You should use a stateful firewall or filtering device that stops these kinds of OOS packets from entering your network.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID *629*

Reference(s): http://www.insecure.org/nmap

Alert: Queso fingerprint

Log Sample(s):

10/12-13:25:57.682235 [**] Queso fingerprint [**] 209.116.70.75:58859 -> MY.NET.100.217:25
10/12-13:42:42.317586 [**] Queso fingerprint [**] 66.216.110.139:36534 -> MY.NET.6.40:25
10/12-13:58:32.493569 [**] Queso fingerprint [**] 209.116.70.75:51312 -> MY.NET.100.217:25

Summary: Queso sends 7 specifically crafted packets and then evaluates the responses to attempt to determine the OS running on the target host. One of the packets sets the 2 high order bits in the TCP flags. This signature helps IDS systems pick up Queso scans.

Analysis and Recommendations:

More and more systems are implementing ECN for congestion notification. As this becomes more common, this rule with generate an increasing number of false positives.

The traffic shown in the logs represents normal traffic from hosts using the ECN bits. An example of this is 209.116.70.75. Reverse DNS on this host resolves to vger.kernel.org.

Pointing your web browser to that page brings up in big red letters the message "TCP/ECN IS ON!"

IDS Suggestion: Consider eliminating this rule. As ECN becomes common, it will produce more and more false positives.

Reference(s):

- CAN-1999-0454
- http://vger.kernel.org
- Ramakrishnan, K. – RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP – September 2001 - <ftp://ftp.isi.edu/in-notes/rfc3168.txt>

Alert: RFB - Possible WinVNC - 010708-1

Log Sample(s):

10/11-20:32:22.314119 [**] RFB - Possible WinVNC - 010708-1 [**] MY.NET.70.225:5900 -> 68.55.61.117:2659
10/13-15:47:16.170965 [**] RFB - Possible WinVNC - 010708-1 [**] 68.48.21.151:1607 -> MY.NET.83.12:5900

Summary: VNC is a remote control application.

Analysis and Recommendations:

Some classify VNC as an administration tool and others consider it a backdoor. The difference is whether the host's user knows what it is being used for. This could represent a compromised system or something as harmless as help from a helpdesk.

If the use of this tool is not allowed for help desk or other remote work, the internal hosts should be examined for signs of WinVNC running.

Organization should consider blocking these protocols from outside the network.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID *560*

Reference(s):

AT&T Laboratories Cambridge, 1999 - <http://www.uk.research.att.com/vnc/winvnc.html>

Alert: SMB C access

Log Sample(s):

10/15-09:10:16.495464 [**] SMB C access [**] 200.193.200.156:1260 -> MY.NET.190.34:139
10/15-21:42:33.970078 [**] SMB C access [**] 211.0.254.196:2077 -> MY.NET.132.20:139

Summary: These represent attempts to connect to the C$ share on a Windows server.

Analysis and Recommendations:

This share is enabled by default to allow administrative access to the C:\ drive. There are several worms that look for unprotected C$ shares to spread themselves. Connection to

network shares should not be allowed from outside the firewall. From internal hosts, this should only be allowed if necessary and the file system on the servers should be configured to use NTFS with proper permissions set on the files and shares.

When we see this alert immediately after a "SMB Name Wildcard" alert that is part of a scan, we have a good indication of an attack attempting to exploit unprotected file shares. An example of this would be the BugBear worm. There were instances of this pattern seen in the logs. The issue is detailed in the "High Level Security Issues" section.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID *533*

References:

- Whitehats - IDS339 "NETBIOS-SMB-C$ACCESS" – http://www.whitehats.com/info/IDS339

Alert: SMB Name Wildcard

Log Sample(s):

```
10/11-00:07:11.260858  [**] SMB Name Wildcard [**] 200.53.80.43:1032 -> MY.NET.134.95:137
10/11-00:07:11.704718  [**] SMB Name Wildcard [**] 200.53.80.43:1032 -> MY.NET.134.98:137
10/11-00:07:12.142271  [**] SMB Name Wildcard [**] 200.53.80.43:1032 -> MY.NET.134.101:137
```

Summary: This represents a NetBIOS scan looking for a status response from NetBIOS and/or SAMBA hosts.

Analysis and Recommendations:

This alert is common "noise" on many networks, but in the case where the alert sources are scrolling through our address ranges, the scanning intent is clear. Many of the alerts in these logs are probably the result of Bugbear or a similar worm. There was an outbreak of this worm around early October that spread VERY quickly through the Internet and many organizations internal networks. It is scanning for victims with unprotected network shares to spread itself.

Don't leave unprotected shares on your host. Consider blocking these ports at the firewall or host's personal firewall if there is no justification for leaving them open.

Examine the notes under "SMB C Access" for additional information.

Reference(s):

- Alexander, Bryce – Port 137 Scan – May 10th, 2000 - <http://www.sans.org/newlook/resources/IDFAQ/port_137.htm>
- Alexander, Bryce, SANS **–** Followup on a Honeypot Catch **-** 2000 - <http://www.sans.org/y2k/honeypot_catch.htm>
- CERT - IN-2000-02, CVE-1999-0225, CAN-1999-0495, CAN-2000-0544

Alert: SUNRPC highport access!

Log Sample(s):

```
10/11-09:00:53.947283  [**] SUNRPC highport access! [**] 64.12.163.214:9898 -> MY.NET.55.70:32771
10/12-07:37:41.454824  [**] SUNRPC highport access! [**] 128.183.252.83:40406 -> MY.NET.163.132:32771
10/13-13:05:50.322968  [**] SUNRPC highport access! [**] 64.71.163.204:888 -> MY.NET.149.49:32771
10/14-02:10:05.535810  [**] SUNRPC highport access! [**] 12.233.125.20:2471 -> MY.NET.21.24:32771
10/14-07:04:24.862470  [**] SUNRPC highport access! [**] 169.229.70.201:39490 -> MY.NET.70.207:32771
10/14-10:51:35.646849  [**] SUNRPC highport access! [**] 64.12.161.185:5190 -> MY.NET.55.59:32771
10/14-13:50:11.165881  [**] SUNRPC highport access! [**] 64.12.25.95:5190 -> MY.NET.168.218:32771
10/14-22:02:20.303909  [**] SUNRPC highport access! [**] 64.12.161.153:5190 -> MY.NET.168.65:32771
```

Summary: Indicates a connection to port 32771. This is most often used for the SUN RPC Service and can be utilized to exploit RPC services on a SUN box.

Analysis and Recommendations:

With the exception of the entries above, all of these alerts from the (5) days of logs were from source ports like 22 (SSH), 80 (HTTP) and 21 (FTP), … For these alerts, the overwhelming likelihood is that they are return traffic from a legitimate session setting off the alarm. The alerts shown above could represent successful connections and should be investigated.

RPC services should be disabled if not needed and the associated ports should be blocked at the firewall. See “High Level Security Issues” section for information on how this relates to an attack.

IDS Suggestion: This alert appears to be generated based on port number alone. Replace this rule with more specific Snort rules that also check for content.

Reference(s): CVE-1999-0189

Alert: SYN-FIN scan!

Log Sample(s):

```
10/11-21:12:42.987754  [**] SYN-FIN scan! [**] 152.101.81.195:21 -> MY.NET.199.250:21
10/11-21:12:43.007479  [**] SYN-FIN scan! [**] 152.101.81.195:21 -> MY.NET.199.251:21
10/11-21:12:43.028977  [**] SYN-FIN scan! [**] 152.101.81.195:21 -> MY.NET.199.252:21
```

Summary: Detection of TCP packets with both the SYN and FIN TCP Flags set.

Analysis and Recommendations:

This is a scan looking for FTP services running on TCP port 21 (usually FTP). Since this is an invalid flag combination, we know that these packets are crafted. The logs show that 152.101.81.195 made 3063 attempts at various hosts on the MY.NET network. No other source addresses were recorded for this SCAN type.

You should use a stateful firewall or filtering device that stops these kinds of packets from entering your network.

IDS Suggestion: Ensure that the Snort rule is up to date - Snort ID *624*

Reference(s):
Chappell, Laura – "Your Being Watched" – March 2001 – <http://www.nwconnection.com/2001_03/pdf31/cybercrm31.pdf>

Alert: TCP SRC and DST outside network

Log Sample(s):
10/12-11:33:35.827091 [**] TCP SRC and DST outside network [**] 192.168.1.101:2268 -> 62.195.94.163:1214
10/14-12:00:24.323264 [**] TCP SRC and DST outside network [**] 10.0.1.2:59443 -> 63.94.193.67:443
10/14-12:03:13.826074 [**] TCP SRC and DST outside network [**] 10.0.1.2:59444 -> 206.64.194.8:80
10/14-12:07:44.325904 [**] TCP SRC and DST outside network [**] 10.0.1.2:59443 -> 63.94.193.67:443

Summary: Indicates that packets were detected with source and destination addresses that are not part of your defined network.

Analysis and Recommendations:
This could be the results of 3rd party noise from a spoofed scanning attempt, a routing error or an application configuration error in the case of the top three log entries above. It could also represent a spoofing attempt from inside your network.

Filter traffic at your border that is not bound for your networks. You should also filter traffic that is not sourced from an internal network address to prevent your users from spoofing attacks on other Internet hosts. Check positioning of network sensors and your home network definitions in your SNORT config.

IDS Suggestion: This rule appears to primarily detect improperly configured hosts or applications. If you setup ingress/egress filtering, you can consider eliminating this rule.

Alert: TFTP - External TCP connection to internal tftp server

Log Sample(s):
10/13-10:22:19.163790 [**] TFTP - External TCP connection to internal tftp server [**] 80.13.42.234:1838 -> MY.NET.111.194:69
10/13-10:22:19.164057 [**] TFTP - External TCP connection to internal tftp server [**] MY.NET.111.194:69 -> 80.13.42.234:1838
10/14-00:50:32.298070 [**] TFTP - External TCP connection to internal tftp server [**] 12.233.125.20:4766 -> MY.NET.83.150:69
10/14-00:50:32.298111 [**] TFTP - External TCP connection to internal tftp server [**] MY.NET.83.150:69 -> 12.233.125.20:4766
10/14-02:10:06.027625 [**] TFTP - External TCP connection to internal tftp server [**] 12.233.125.20:2531 -> MY.NET.21.24:69
10/14-02:10:06.029202 [**] TFTP - External TCP connection to internal tftp server [**] MY.NET.21.24:69 -> 12.233.125.20:2531
10/14-07:03:55.429941 [**] TFTP - External TCP connection to internal tftp server [**] 169.229.70.201:38498 -> MY.NET.70.207:69
10/15-19:28:25.261128 [**] TFTP - External TCP connection to internal tftp server [**] MY.NET.190.100:69 -> 12.213.46.19:2260
10/15-19:28:27.513531 [**] TFTP - External TCP connection to internal tftp server [**] 12.213.46.19:2260 -> MY.NET.190.100:69

Summary: These alerts record possible connections to the internal TFTP servers from external systems.

Alert: TFTP - External UDP connection to internal tftp server

Log Sample(s):
10/14-11:39:52.937591 [**] TFTP - External UDP connection to internal tftp server [**] 66.70.17.91:48769 -> MY.NET.111.11:69

10/14-18:36:48.976645 [**] TFTP - External UDP connection to internal tftp server [**] 64.200.105.51:5005 -> MY.NET.152.163:69
10/15-08:36:28.379563 [**] TFTP - External UDP connection to internal tftp server [**] 65.59.116.64:45555 -> MY.NET.151.115:69
10/15-16:00:24.435966 [**] TFTP - External UDP connection to internal tftp server [**] 66.28.10.84:5510 -> MY.NET.84.198:69
10/15-17:50:57.987857 [**] TFTP - External UDP connection to internal tftp server [**] 63.250.219.188:20802 -> MY.NET.168.253:69

Summary: Records possible connection to the internal TFTP servers from external host.

Analysis and Recommendations:

Buffer overflow attacks often plant only a small program, referred to as a "grappling hook", that is responsible to retrieve a larger exploit and install it. This retrieval can be accomplished with many protocols including TFTP.

This should be investigated. If these are known systems and this is considered valid activity then updating of the Snort configuration should be considered to avoid these alert entries. Normally, un-secure services like TFTP should be avoided, especially from external sources.

These entries combined with other alert log entries for MY.NET.151.115 and MY.NET.84.198 represent possibly compromised systems. These hosts should be checked immediately. See section on "High Level Security Issues" for additional details.

Alert: TFTP - Internal UDP connection to external tftp server

Log Sample(s):

10/11-22:53:54.382910 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.82.2:1075 -> 130.130.21.29:69
10/12-07:54:18.068261 [**] TFTP - Internal UDP connection to external tftp server [**] MY.NET.83.146:6257 -> 219.161.161.86:69
10/12-18:48:50.402100 [**] TFTP - Internal UDP connection to external tftp server [**] 68.14.128.176:69 -> MY.NET.83.146:6257
10/15-08:01:37.824103 [**] TFTP - Internal UDP connection to external tftp server [**] 65.59.116.64:69 -> MY.NET.151.115:8128

Summary: This indicated that an internal host has accessed a TFTP server that is outside of your network space.

Analysis and Recommendations:

TFTP is not in and of itself harmful, but it is very insecure in that it uses clear text and no authentication. TFTP is also a common protocol in attacks that use "Grappling Hook" programs. These are small pieces of code that go out to network resources and retrieve the real code that the attacker wants to load on the compromised host. These are popular because of the small amount of code necessary. It makes them easy to deliver via buffer overflows, worms, etc.

Unless there is a strong organizational case for allowing this kind of traffic, it should be blocked at the network edge via a stateful firewall. In addition, the internal hosts listed should be checked for compromise.

These entries combined with other alert log entries shown in the "TFTP - External UDP connection to internal tftp server" section above, represent possibly compromised systems. See section on "High Level Security Issues" for additional details.

Alert: Tiny Fragments - Possible Hostile Activity

Log Sample(s):

10/15-12:50:11.222243 [**] Tiny Fragments - Possible Hostile Activity [**] 68.55.87.49 -> MY.NET.168.80
10/15-13:32:00.222801 [**] Tiny Fragments - Possible Hostile Activity [**] 68.83.182.149 -> MY.NET.91.240

Summary: This alarm is triggered by Snort's minifrag preprocessor when fragments smaller than the minimum configured threshold are received.

Analysis and Recommendations:

This can represent the use of NMAP, fragrouter or the use of other fragment generating tools. The purpose is probably to evade IDS detection by breaking up packets that would match an IDS signature into smaller chunks that would not match.

Check the SNORT minifrag preprocessor config to make sure it is set correctly. If this is an attack, you can block the source IP or configure your routers / firewall not to pass fragments. Both of these have risk and would need to be weighed against the value to the organization. You should also contact the owner of the source IP and inform them of what you are seeing.

Reference(s):

- http://archives.neohapsis.com/archives/snort/2000-05/0103.html
- CVE-1999-0683, CVE-1999-0804

Alert: Watchlist 000220 IL-ISDNNET-990517

Log Sample(s):

10/11-03:39:25.871028 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.104.119:4438 -> MY.NET.150.113:26963
10/11-04:08:51.229450 [**] Watchlist 000220 IL-ISDNNET-990517 [**] 212.179.19.161:15203 -> MY.NET.198.204:1214

Summary: This is a rule setup to monitor traffic coming from a particular network range. These rules are generally created based on past detects from the network range. This particular rule looks like it was created on May 17th 1999.

Analysis and Recommendations:

A lot of these detects are to port 1214. This is the default port for KaZaa, a popular peer to peer file sharing system. These kinds of file sharing systems are notorious for spreading infected applications so while this is probably normal activity, it can still represent an indirect security risk.

There was a disproportionate amount of packets to port 2939 from 212.179.83.64 (member of the watch list) port 3871. If resources are available, investigate this communications and others conversations like it. They may represent the use of non-standard ports for peer to peer applications, but without additional information, this is hard to determine.

IDS Suggestion: If this rule was created in 1999, it may no longer be needed. If we still suspect hosts from this watchlist then blocking these addresses at the firewall may be a better solution than monitoring their traffic.

Reference(s): Neel, Robert - http://www.giac.org/practical/Robert_Neel.doc

---

Alert: Watchlist 000222 NET-NCFC

Log Sample(s):
10/15-21:20:22.732984 [**] Watchlist 000222 NET-NCFC [**] 159.226.124.70:1638 -> MY.NET.154.30:80

Summary: This is a rule setup to monitor traffic coming from a particular network range.

Analysis and Recommendations:
Traffic listed in these log entries looks fairly innocuous, but because this identifies traffic from sources that have been sources of issues in the past, these should be investigated as resources are available.

IDS Suggestion: If there is no organizational reason to allow communications with these subnets, a more effective strategy may be to block these ranges at the network perimeter.

Alert: connect to 515 from outside

Log Sample(s):
10/11-03:14:20.480967 [**] connect to 515 from outside [**] 217.83.3.90:1413 -> MY.NET.135.121:515
10/11-03:14:20.489060 [**] connect to 515 from outside [**] 217.83.3.90:1414 -> MY.NET.135.122:515
10/11-03:14:20.496975 [**] connect to 515 from outside [**] 217.83.3.90:1415 -> MY.NET.135.123:515

Summary: Connection attempts to port 515 from hosts outside our network range.

Analysis and Recommendations:
This traffic represents a scan for port 515 which is generally used for printing services (LPD). These entries represent a scan that is likely trying to locate a host that is susceptible to a buffer overflow or other exploit on its printing service.

Printing service should not be allowed from external hosts unless there is a strong organization reason to do so. Block these at the firewall and you will stop the scan as well. Also be sure to keep all print daemons up to date. Snort rules should be updated to look for specific exploit content.

Reference(s): CVE-2000-0917 / Bugtraq 1712

Alert: spp_http_decode: CGI Null Byte attack detected

Log Sample(s):
10/11-08:34:19.807413 [**] spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.146:1915 -> 209.10.239.135:80
10/11-08:34:19.807413 [**] spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.146:1915 -> 209.10.239.135:80

10/11-08:34:19.807413 [**] spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.146:1915 -> 209.10.239.135:80
10/11-08:34:19.807413 [**] spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.146:1915 -> 209.10.239.135:80
10/11-08:34:19.807413 [**] spp_http_decode: CGI Null Byte attack detected [**] MY.NET.153.146:1915 -> 209.10.239.135:80

Summary: This alert is triggers when "%00" is detected in a http request. This can be valid content in many cases so this rule is highly prone to false positives.

Analysis and Recommendations:

All entries for this alert except one were from internal hosts going to external sites. The internal hosts that may need further investigation are listed below. These should be examined because they also triggered IIS Unicode attacks. While this does not guarantee a true attack is taking place, it does raise a flag that we should investigate.

Suspect Internal Servers (Triggered IIS Unicode and CGI Null Byte Alerts)			
MY.NET.80.134	MY.NET.168.62	MY.NET.53.40	MY.NET.153.174
MY.NET.183.23	MY.NET.80.134	MY.NET.168.62	MY.NET.178.78
MY.NET.53.40	MY.NET.183.23	MY.NET.146.17	MY.NET.29.3

IDS Suggestion: Since this is such a common False Positive, you may want to reconfigure this SNORT preprocessor using the –cginull switch.

Reference(s): Ellis, Joe – GCIA Practical Assignment v3.0, Intrusion Detection In Depth – May 14th, 2002 - <http://www.giac.org/practical/Joe_Ellis_GCIA.doc>

Alert: spp_http_decode: IIS Unicode attack detected

Log Sample(s):

10/11-01:00:57.366351 [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.84.133:1551 -> 211.115.212.50:80
10/11-01:00:57.366351 [**] spp_http_decode: IIS Unicode attack detected [**] MY.NET.84.133:1551 -> 211.115.212.50:80

Summary: This indicates that Unicode was detected by Snort's HTTP pre-processor. This could represent an attempt to use the UNICODE representation of characters to hide commands or intentions from log files and IDS systems. It has also been a technique used to include characters that would not normally be processed by the URL decoding logic at the server, allowing the attacker to traverse directories and execute code. This technique has been used by various Internet worms.

This can be a common false positive as more and more web sites begin to use Unicode. This is especially true of web sites that include content in Asian and other languages that utilize Unicode for their representation.

Analysis and Recommendations:

For alerts where the web server was external to the University, many of the IP addresses of the servers were registered to sites in Korean. This indicates that any web pages there are probably in Unicode to support the Korean character sets and would cause this alert to go off.

For the Inbound attacks, the targets should be examined for signs of exploit. This is an old vulnerability and all web servers should be patched for this vulnerability.

Some of the internal systems that were targeted in these alerts were HP Printers with JetDirect cards installed, Cisco wireless access points and other appliance type devices. It is important to remember that many of these appliance devices run web servers based on IIS or other popular web server code and can be just as vulnerable to attack as web servers running on more traditional platforms.

IDS Suggestion: Since this is such a common False Positive, you may want to reconfigure this SNORT preprocessor using the –unicode switch.

Reference(s):

- Mohan, Potheri - <http://www.giac.org/practical/Potheri_Mohan_GCIH.doc>
- http://archives.neohapsis.com/archives/snort/2001-08/0528.html
- MS Bulletin (MS00-057)
- IIS4.0 Patch:http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/
- IIS 5.0 Patch:http://www.microsoft.com/windows2000/downloads/critical/q269862

Scanner Activity

Top 10 Scanner Source Addresses

Top 10 Scanner Source IPs			
IP	Count	Reverse DNS Lookup on IP	Scan Type
MY.NET.70.176	610490	phaser.ucs.giac.edu	UDP and SYN
MY.NET.84.147	566210	engr-84-147.pooled.giac.edu	UDP and SYN
MY.NET.165.24	254247	slate.giac.edu	UDP and SYN
MY.NET.91.240	152509	Fh-91-240.pooled.giac.edu	UDP and SYN
MY.NET.83.146	135804	aciv-83-146.pooled.giac.edu	UDP and SYN
MY.NET.198.204	125375	rwd-204.giac.edu	UDP and SYN
MY.NET.150.113	98767	ill2.lib.giac.edu	UDP and SYN
MY.NET.88.165	92032	lib-88-165.pooled.giac.edu	UDP and SYN
MY.NET.111.214	76805	raychen-16.engr.giac.edu	UDP and SYN
MY.NET.70.207	68729	Ecs020pc09.giac.edu	UDP

Top 10 Scanned Destination Addresses

Top 10 Scanned Destination IPs			
IP	Count	Reverse DNS Lookup on IP	Scan Type
204.183.84.240	10979	N/A	UDP and SYN
24.120.194.178	7617	cm178.194.120.24.lvcm.com	UDP and SYN
12.220.145.126	5620	12-220-145-126.client.insightBB.com	UDP and SYN
12.245.31.155	3936	12-245-31-155.client.attbi.com	UDP and SYN
68.39.48.75	3694	Bgp599746bgs.midltn01.nj.comcast.net	UDP and SYN
MY.NET.70.207	2863	Ecs020pc09.giac.edu	UDP and SYN
151.204.131.129	2636	pool-151-204-131-129.ny325.east.verizon.net	UDP and SYN
146.115.121.119	2486	146-115-121-119.c3-0.smr-ubr2.sbo-smr.ma.cable.rcn.com	UDP and SYN
141.149.54.140	2238	pool-141-149-54-140.ny325.east.verizon.net	UDP and SYN
200.52.195.1	2225	customer-GDL-195-1.megared.net.mx	UDP

Top Scanned Ports

Top 10 Scanned Ports			
Port	Count	Applications That Commonly Found on These Ports	Primary Scan
6257	1211443	WinMX Peer to Peer Application	UDP
80	85438	HTTP - Web Services	SYN
1214	74779	Kazaa Peer to Peer Application	UDP/SYN/VECNA
4665	63180	eDonkey or other Peer to Peer Application	UDP
1	49244	TCPMUX - Used for SGI IRIX Systems	SYN and UDP
27005	31247	Half-Life Interactive Gaming	UDP
53	21704	DNS Domain Name Service	UDP and SYN
21	21017	FTP File Transfer Protocol	SYN and UDP
443	13941	SSL Web Services	SYN
25	11750	SMTP eMail Services	SYN and UDP

Scanner Summary

There were a total of 2,669,298 scan events recorded over these 5 days. The majority of this traffic looks like it is not truly scanning activity but rather alerts generated by Peer to Peer networking applications such as WinMX, KaZaa, eDonkey and others. This peer to peer chatter accounts for over 76% of the scan events.

Of the remaining events, there were a few that were of particular interest.

Scan to Port 111 – RPC Portmapper

24.123.46.10 and 210.46.90.254 did an extensive scan of hosts looking for port 111/tcp (portmapper for RPC services). This is a particularly dangerous scan because it is a targeted search for services with known exploits.

Scan for port 1 (possible attempt to locate IRIX systems)

IRIX systems have several default accounts with no password assigned. There have also been a number of vulnerabilities identified with services running on these systems. Around the time of these file captures, there were current advisories for SGI IRIX including a tooltalk vulnerability that could allow remote command execution or Denial of Service against vulnerable boxes. Reference information for known SGI vulnerabilities is listed at SGI's website. Information on this specific tooltalk vulnerability is listed below:

SGI Security Advisory

Title: IRIX ToolTalk rpc.ttdbserverd vulnerabilities
Number: 20021102-01-P
Date: November 6, 2002
Reference: CAN-2002-0677
Reference: CAN-2002-0678
Reference: CERT CA-2002-20
Reference: SGI Security Advisory 19981101-01-A
Reference: SGI BUGS 833473 860031 861480 866829
Fixed in : IRIX 6.5.18 or patch 4669

---

http://www.cert.org/advisories/CA-2002-20.html
http://www.entercept.com/news/uspr/08-12-02.asp

This vulnerability has been assigned the following CVEs:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0677
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0678

Additional CVE entry related to IRIX can be found at
http://www.cert.org/advisories/CA-1995-15.html

Scan to Port 0

Oct 15 00:31:53 65.83.150.166:0 -> MY.NET.185.48:0 NULL ********

This is most likely the work of hping2. As a general rule, packets with source port 0 should be blocked at the border routers or firewall.

Incoming UDP 137 Scans

There were numerous scan entries that were bound to UDP port 137 on the destination machine. This is most likely the work of BugBear or some similar Worm that searches for unprotected windows file shares. This was a very widespread worm and through the five day period, this site was scanned every day by multiple addresses from the table below. UDP 137 and the other NetBIOS ports should not be allowed to pass in through the border firewall unless absolutely required.

Probable BugBear or Similar Worm Attackers			
205.125.41.108	61.63.193.73	202.102.142.54	194.153.206.33
203.166.102.129	61.219.65.78	62.82.17.134	80.239.170.161
203.73.193.142	61.164.165.209	62.83.5.49	24.197.148.247
217.225.13.153	62.71.200.194	210.58.252.159	193.248.47.204
165.228.237.15	66.100.114.81	211.222.125.214	210.183.221.23
211.48.217.235	80.62.155.24	61.9.40.80	64.89.86.123
213.97.33.189	211.92.141.222	211.45.26.243	61.202.34.28
210.119.192.108	61.36.119.19	61.124.40.82	211.18.53.139
140.116.60.199	211.195.198.156	143.248.227.69	202.63.168.36
211.205.36.78	62.46.138.216	64.144.9.1	211.221.142.60
202.233.7.197	66.169.253.123	218.18.49.20	61.118.210.172
195.142.62.149	207.164.21.21	212.95.99.142	211.38.60.177
24.83.97.26	200.84.71.52	211.227.157.205	200.206.236.80
61.251.225.13	151.203.164.131	195.175.204.6	80.32.215.206
148.240.99.175	61.197.120.227	61.165.8.75	202.180.97.10
65.67.157.75	148.246.66.20	210.203.98.16	218.226.40.109
211.221.213.169	218.32.92.136	65.93.141.6	200.161.197.9
80.133.129.170	63.207.61.110	194.153.206.33	216.231.61.149
212.64.81.223	211.212.108.172	211.55.145.20	210.123.155.3

FTP Scanner

64.52.4.180 was involved in a mass scan for port 21/tcp (ftp servers). This showed up in the OOS packet logs.

Miscellaneous Scanning

There were several miscellaneous scans that did not appear to be related to an attack (ex. MY.NET.165.20 Scanned MY.NET.90.114 looking for open ports, 169.229.70.201 scanned MY.NET.70.201 …). Some of this could have been fire drawn from connections to IRC channels or P2P networks. These often draw the wrong kind of attention.

Scan alerts from MY.NET.87.50 were the result of an extensive Counterstrike game. Other than wasting bandwidth, this is harmless traffic from an online gaming event. This game did however draw some other Out of Spec scanning packets on the host of the game.

Scanning from host MY.NET.140.179, MY.NET.122.123, MY.NET.149.63, MY.NET.55.87 and MY.NET.55.88 were really the result of AFS file system running on those hosts. All destination addresses were universities, research laboratories or membrain.com (where they posted on their website that they were running AFS on the hosts seen in the communications.).

Out of Spec Activity

Top 10 OOS By Source Address

Source IP	OOS Description	Count
152.101.81.195	SYN-FIN SCAN	7186
MY.NET.28.2	TCP NULL SCAN / NMAP SCAN	3638
64.52.4.180	SYS SCAN WITH RESERVED BITS SET	3558
209.116.70.75	Telnet connections from Linux Box (ECN bits set)	814
MY.NET.70.183	TCP NULL SCAN	542
200.221.192.194	KaZaa Traffic	485
MY.NET.165.20	NULL SCAN of MY.NET.90.114 (NMAP) – 2 Second Scan	207
200.221.192.245	KaZaa Traffic	69
148.65.203.115	Kazaa Traffic	62
148.63.246.3	Kazaa Traffic	55

Top 10 OOS By Destination

Destination IP	OOS Description	Count
MY.NET.100.217	TELNET TRAFFIC FROM 209.116.70.75 (795 Detects) SYN-FIN SCAN FROM 152.101.81.195 (1 Detect)	796
MY.NET.91.81	KaZaa TRAFFIC FROM 200.221.192.194 (485 Detects) KaZaa TRAFFIC FROM 200.221.192.245 (69 Detects) SYN-FIN SCAN FROM 152.101.81.195 (1 Detect)	555
MY.NET.1.4	NULL TCP FLAGS SET FROM MY.NET.70.183	542
MY.NET.6.40	RESERVED BITS SET IN TCP FLAGS (ECN enabled) Connections to port 25 (SMTP) and RST packets from attempts to connect from this host to MY.NET.12.3 (possible attempt to connect to FreeStyle Chat server or some other service that was not listening)	384
MY.NET.90.114	NULL TCP FLAGS FROM MY.NET.28.2 (1 Detect) NULL SCAN FROM MY.NET.165.20 (207 Detects) 1 then wait 2 minutes then 206 in 2 seconds.	208
MY.NET.185.48	(1) SYN-FIN SCAN OF PORT 21 FROM 152.101.81.195 (3) SCANS FROM MY.NET.28.2 TO PORT 22 (161) Gnutella packets FROM HOST WITH ECN BITS SET	165
MY.NET.150.133	(62) KaZaa FROM 148.65.203.115 (1) KaZaa FROM 200.221.193.116	63

MY.NET.84.245	(2) SNAN PACKETS FROM MY.NET.28.2 (1) SYN-FIN SCAN FROM 152.101.81.195 to PORT 21 (55) KaZaa FROM 148.63.246.3	58
MY.NET.168.105	Traffic to port 4662 with ECN bits set. This IP is registered in DNS as a wireless device. It is most likely a wireless access point and 4662 is either a mgt or negotiation port for it. The ECN Bits are set because the boxes are probably running a LINUX kernel with this feature enabled.	52
MY.NET.91.240	OOS Packets from IP registered to a Brazilian ISP OOS Packets from IP registered to Spacenet NULL SCAN PACKET FROM 202.52.202.248	27

OOS Summary

Around 5200 of the OOS packets were generated due to the 2 high order bits of the TCP Flags being set. These two bit were historically reserved and were always set to 0. In modern IP Stacks however, there is a feature called Explicit Congestion Notification. This is defined in RFC3168 (ftp://ftp.isi.edu/in-notes/rfc3168.txt). Using this standard, the high order bits of the TCP Flags can be used to carry information about link congestion. This information can then be used by the sending and / or receiving host to make adjustments to the packet transfer parameters as it sees fit. This allows IP to compensate for issues instead of just dropping packets when link congestion occurs.

As these become more common in IP stacks, this test will need to be removed or it will generate too many False Positives, alerting you of scans and OS fingerprinting attacks that are not really happening. In our alert files, this also generated several Queso Scan alerts that were actually legitimate traffic.

Sample(s):

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

10/10-00:35:51.188697 216.143.112.105:51002 -> MY.NET.140.2:80
TCP TTL:48 TOS:0x0 ID:38766 IpLen:20 DgmLen:60 DF
12****S* Seq: 0xDC3CB602  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (7) => MSS: 1380 SackOK TS: 80584891 0 NOP NOP NOP NOP

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

10/10-00:49:22.941544 209.47.251.18:60778 -> MY.NET.6.40:25
TCP TTL:49 TOS:0x0 ID:34423 IpLen:20 DgmLen:60 DF
12****S* Seq: 0x5E07FF5B  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1380 SackOK TS: 382999658 0 NOP WS: 0

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

In these two packets, you can see a connection to a web server and a connection to an SMTP mail gateway. Both of these alerted as OOS but are in fact normal traffic. The source IP address on the mail server is registered to Red Hat, Inc. Linux is one of the Operating Systems where ECN implementation is becoming popular.

NMAP Scan

On 10/10, it looks like the University did its own NMAP scan from MY.NET.28.2. A check of the mailing list archives at the University indicates that they do this regularly and from hosts in this range. You can see NMAP checking for a telnet port in the entries below. It uses NULL Flags and other strange TCP flag combinations to test for open ports and for OS fingerprinting.

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

10/10-14:00:47.685370 MY.NET.28.2:39634 -> MY.NET.10.121:23
TCP TTL:46 TOS:0x0 ID:34569 IpLen:20 DgmLen:60
******** Seq: 0x4215A114  Ack: 0x0  Win: 0x1000  TcpLen: 40
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

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

10/10-14:00:47.685420 MY.NET.28.2:39635 -> MY.NET.10.121:23
TCP TTL:46 TOS:0x0 ID:65419 IpLen:20 DgmLen:60
**U*P*SF Seq: 0x4215A114  Ack: 0x0  Win: 0x1000  TcpLen: 40  UrgPtr: 0x0
TCP Options (5) => WS: 10 NOP MSS: 265 TS: 1061109567 0 EOL

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

SYN FIN Scan for FTP Servers

There were 7186 SYN/FIN Packets sent from 155.101.81.195 to various subnets on the Universities internal network. Because the packets were all directed at port 21 on the remote host, this appears to be a scan looking for FTP servers.

SYN/FIN combination is used for various reasons. One is to bypass certain packet filters that might allow all FIN packets through. Another reason to use SYN/FIN is simply to get a specific reaction from the remote host without actually opening a connection. This combination of packets produces different results on different IP stacks, but it is always an illegal TCP flag combination and is easy to spot.

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

10/11-20:50:07.763499 152.101.81.195:21 -> MY.NET.7.3:21
TCP TTL:23 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x460B42FE  Ack: 0x37DDBC9A  Win: 0x404  TcpLen: 20

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

10/11-20:50:07.784518 152.101.81.195:21 -> MY.NET.7.4:21
TCP TTL:23 TOS:0x0 ID:39426 IpLen:20 DgmLen:40
******SF Seq: 0x460B42FE  Ack: 0x37DDBC9A  Win: 0x404  TcpLen: 20

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

Don’t Fragment…no wait, More Fragments

This OOS packet showed up from 217.227.143.85 (pD9E38F55.dip.t-dialin.net). The packets are strange because they have both the Don’t Fragment and the More Fragments bits set in the IP Header. This is not valid because If the don’t fragment bit were set on the original packet then there could be no fragment to start with, let alone MORE of them to come. The purpose of this packet is still unknown. There were not

enough to cause Denial of service. One possible theory is that this is a mapping technique where the sender hopes that a fragment will make it through firewalls or packet filters because it is a fragment and not a complete packet. If it were to get to the destination host and then no more fragments came, the destination host might send a ICMP Fragment Reassembly Time Exceeded message and the remote host would know that machine was alive.

=+

10/12-18:00:01.637274 217.227.143.85 -> MY.NET.150.209
TCP TTL:116 TOS:0x0 ID:53701 IpLen:20 DgmLen:752 DF MF
Frag Offset: 0x0   Frag Size: 0x2DC
2D 88 20 9F 1B 8D CA AA 04 A2 50 38 D4 07 06 11 -. ....P8....
(data payload of packet omitted for space considerations)

=+

External Source Address Information

The following section includes information on external IP addresses that were involved in events of interest. For each address the reason it was chosen is listed.

External IP address 209.116.70.75

This address was selected because it was one of the OOS Top Talkers and it showed up in the alert files as Queso fingerprint alerts. The reason this was flagged as OOS was the reserved bits were set in the TCP flags. This is common for newer Linux Kernels and seeing that the address is registered to Red Hat reinforces the fact that this is actually “in-spec” traffic.

Search results for: ! NET-209-116-70-64-1

```
CustName:   Red Hat, Inc.
Address:    4518 South Miami Blvd. Suite #100 Durham NC 27703
Country:    US
RegDate:    2002-09-23
Updated:    2002-09-23

NetRange:   209.116.70.64 - 209.116.70.95
CIDR:       209.116.70.64/27
NetName:    INFLOW-18773-5591
NetHandle:  NET-209-116-70-64-1
Parent:     NET-209-116-68-0-1
NetType:    Reassigned
Comment:
RegDate:    2002-09-23
Updated:    2002-09-23
```

NSLOOKUP RESULTS:
Name: vger.kernel.org
Address: 209.116.70.75

External IP address 64.52.4.180

This address was selected because was the source for 1546 alerts that indicate a scan for ftp servers. These appeared in the alert files as Queso fingerprint alerts because the two high order bits were set in the TCP Flags along with the SYN flag. This represents a clear reconnaissance attempt against a specific service on our systems. The owner is Eureka Broadband. Scans from broadband ISPs are fairly common, but we may want to add the source to a watch list IDS rule since they may have an exploit planned for when they find a target running ftp.

Search results for: 64.52.4.180

```
OrgName:    Eureka Broadband
OrgID:      EBRB

NetRange:   64.52.0.0 - 64.52.255.255
CIDR:       64.52.0.0/16
NetName:    EUREKA-BLK1
NetHandle:  NET-64-52-0-0-1
Parent:     NET-64-0-0-0-0
NetType:    Direct Allocation
NameServer: AUTH1.EUREKADNS.NET
NameServer: AUTH2.EUREKADNS.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2000-03-06
Updated:    2001-10-12
```

NSLOOKUP RESULTS:

Name: nat.64-52-4.180.ip.ebrb.net
Address: 64.52.4.180

External IP address 210.46.90.254

This address was selected because it was scanning all of our hosts for open port 111. This kind of focused scanning indicates that they probably have an exploit for one of the RPC services and are looking for a system to use it on. This address is registered to a University in China so it may be the source of the attack or it may be a system that was taken over by someone else at another location. Either way, its behavior is clearly hostile and it should be watched or blocked.

Search results for: 210.46.90.254

```
inetnum:      210.46.88.0 - 210.46.95.255
netname:      HRBMU1-CN
descr:        ~{9~6{1uR=?F4sQ'~}
descr:        Harbin Medical University
descr:        Harbin, Heilongjiang 150086, China
country:      CN
admin-c:      YH39-AP
tech-c:       YH39-AP
notify:       address-allocation-staff@net.edu.cn
mnt-by:       MAINT-NULL
changed:      hostmaster@net.edu.cn 19990903
status:       ALLOCATED PORTABLE
source:       APNIC
```

```
person:       Ying He
address:      Department Of Network Center
address:      Harbin Medical University
address:      Harbin, Heilongjiang 150086, China
phone:        +86-451-6680174
e-mail:       heying@ems.hrbmu.edu.cn
nic-hdl:      YH39-AP
notify:       address-allocation-staff@net.edu.cn
mnt-by:       MAINT-NULL
changed:      hostmaster@net.edu.cn 19990903
source:       APNIC
```

NSLOOKUP RESULTS:
NO INFO

External IP address 217.227.143.85
This address was selected because they were sending us OOS packets. These packets had both the Don't Fragment and the More Fragments bits set in the IP header. In addition, they were initial fragments but there was no final fragment. This is possibly a mapping technique and more activity could follow. This IP range should be watched.

```
inetnum:      217.224.0.0 - 217.237.161.47
netname:      DTAG-DIAL15
descr:        Deutsche Telekom AG
country:      DE
admin-c:      DTIP-RIPE
tech-c:       ST5359-RIPE
status:       ASSIGNED PA
remarks:      ************************************************************
remarks:      * ABUSE CONTACT: abuse@t-ipnet.de IN CASE OF HACK ATTACKS, *
remarks:      * ILLEGAL ACTIVITY, VIOLATION, SCANS, PROBES, SPAM, ETC.   *
remarks:      ************************************************************
notify:       auftrag@nic.telekom.de
notify:       dbd@nic.dtag.de
mnt-by:       DTAG-NIC
changed:      auftrag@nic.telekom.de 20020108
source:       RIPE
route:        217.224.0.0/11
descr:        Deutsche Telekom AG, Internet service provider
origin:       AS3320
mnt-by:       DTAG-RR
changed:      bp@nic.dtag.de 20010405
source:       RIPE
person:       DTAG Global IP-Adressing
address:      Deutsche Telekom AG
address:      Bayreuther Strasse 1
address:      D-90409 Nuernberg
address:      Germany
phone:        +49 911 68909856
e-mail:       ripe.dtip@telekom.de
nic-hdl:      DTIP-RIPE
mnt-by:       DTAG-NIC
changed:      ripe.dtip@telekom.de 20020717
source:       RIPE
```

```
person:       Security Team
address:      Deutsche Telekom AG
address:      Technikniederlassung Schwaebisch Hall
address:      D-89070 Ulm
address:      Germany
phone:        +49 731 100 84055
fax-no:       +49 731 100 84150
e-mail:       abuse@t-ipnet.de
nic-hdl:      ST5359-RIPE
notify:       auftrag@nic.telekom.de
notify:       dbd@nic.dtag.de
mnt-by:       DTAG-NIC
changed:      auftrag@nic.telekom.de 20010321
source:       RIPE
```

NSLOOKUP RESULTS:
Name: pD9E38F55.dip.t-dialin.net
Address: 217.227.143.85

External IP address 24.123.46.10
This address was selected because they were scanning our network on port 111. They were also listed by *DeScan.com* as a known scanner. This makes them suspect and the address should be watched.

Search results for: 24.123.46.10

```
OrgName:    ROADRUNNER-COMMERCIAL-CENTRAL
OrgID:      RCCT

NetRange:   24.123.0.0 - 24.123.255.255
CIDR:       24.123.0.0/16
NetName:    RR-COMMERCIAL-CENTRAL
NetHandle:  NET-24-123-0-0-1
Parent:     NET-24-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.BIZ.RR.CO
NameServer: NS2.BIZ.RR.COM
NameServer: DNS4.RR.COM
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2001-09-26
Updated:    2002-04-09
```

NSLOOKUP RESULTS:
Name: rrcs-central-24-123-46-10.biz.rr.com
Address: 24.123.46.10

DeScan.Com INFORMATION:
http://www.descan.net/searchresults.html?command=specific&source=24.123.46.10

Links off of Road Runners NorthEast Ohio network based on traceroute data returned from visualroute.com.

Summary of Internal Hosts and Possible Services

This following table lists probable web servers at the site. These are believed to be running a service on port 80/tcp because we can observe IIS Unicode attacks detected against the, which require completion of the 3-way handshake to port 80 and also because of the Queso scan alerts to some of these IPs on port 80. In the case of the Queso scans, the pattern of the alerts were across many packets to a single host, a check of the logs show that the alert was due to the ECN bits being set. This is not abnormal for some newer kernels so the alerts most likely represent normal web traffic.

Probable Web Server on MY.NET			
MY.NET.1.201	MY.NET.21.59	MY.NET.27.3	MY.NET.100.187
MY.NET.1.205	MY.NET.21.60	MY.NET.29.11	MY.NET.100.217
MY.NET.5.14	MY.NET.21.61	MY.NET.24.44	MY.NET.100.221
MY.NET.5.92	MY.NET.21.62	MY.NET.179.77	MY.NET.100.251
MY.NET.5.99	MY.NET.21.64	MY.NET.137.66	MY.NET.104.104
MY.NET.10.176	MY.NET.21.69	MY.NET.150.83	MY.NET.104.113
MY.NET.15.227	MY.NET.21.94	MY.NET.145.18	MY.NET.104.114
MY.NET.21.12	MY.NET.22.8	MY.NET.100.15	MY.NET.104.128
MY.NET.21.14	MY.NET.22.36	MY.NET.100.27	MY.NET.104.139
MY.NET.21.17	MY.NET.22.52	MY.NET.100.69	MY.NET.104.145
MY.NET.21.20	MY.NET.22.54	MY.NET.100.71	MY.NET.104.177
MY.NET.21.48	MY.NET.22.67	MY.NET.100.133	MY.NET.104.180
MY.NET.21.47	MY.NET.22.70	MY.NET.100.143	MY.NET.104.213
MY.NET.21.56	MY.NET.22.102	MY.NET.100.145	
MY.NET.21.57	MY.NET.22.103	MY.NET.100.158	
MY.NET.21.58	MY.NET.22.111	MY.NET.100.165	

Additionally, the following table lists other services and the hosts we suspect are running them. The data that led us to these conclusions is listed in each case. Please note that this list is only a sample, there were several other hosts and services that could be mapped based on the data in the logs. Especially in the area of Peer to Peer services. If additional granularity is needed, the same principals and process used to determine these hosts and services can be reapplied.

Host	Service	Why We Think This
MY.NET.137.7 MY.NET.1.2 MY.NET.100.158 MY.NET.113.208 131.118.254.1(Ext) 66.28.32.88(Ext)	DNS	Saw reply traffic from PORT 53 in the Scan logs. Used NSLOOKUP and then set my default server to this host. IPs responded and answered DNS queries are listed here.
MY.NET.21.24 MY.NET.83.150 MY.NET.111.194	TFTP	Alerts of TCP TFTP Connections in both directions
MY.NET.100.158 MY.NET.70.49 MY.NET.70.50 MY.NET.83.197	FTP	FTP GLOB Alert or Custom Rules to monitor ftp traffic.
MY.NET.6.40 MY.NET.145.9	SMTP	Queso Fingerprint alerts that were actually mail traffic with ECN bits set and “Bugbear@MM virus in SMTP”

MY.NET.139.230 MY.NET.144.59 MY.NET.179.78		alerts
MY.NET.25.21	IMAP	This host is a mail router and we saw IMAP port traffic
MY.NET.185.48 MY.NET.182.94 MY.NET.86.102 MY.NET.83.201 MY.NET.70.27	Gnutella	Connections to it with ECN Bits showed up as Queso
MY.NET.111.214	P2P (eDonkey, etc…)	Queso detect to 4662/tcp
MY.NET.84.147 MY.NET.91.81	KaZaa WinMX	MY.NET.91.81 is running KaZaa. Contents of the OOS packets show it is sharing copyrighted movies.
MY.NET.70.198 MY.NET.70.176 MY.NET.83.146	IRC	Traffic to port this IP on port 6667 and 6669 shows up in the OOS and Alert logs
MY.NET.55.87 MY.NET.55.88 MY.NET.122.123 MY.NET.140.179 MY.NET.149.63	AFS	Scanning logs show alerts from these systems answering client connection requests

High Level Security Issues

The Alert Descriptions section covers an assessment of possible security issues related to each alert type. This section covers other high level security concerns that he site should consider and provides further details for attacks that need correlation of multiple alerts.

IDS Needs Tuning to reduce False Positives

The majority of the IDS logs represent False Positives or other noise. This indicates a system that is not properly tuned for its environment. Having this much clutter in the logs drastically reduces the IDS's effectiveness as a security tool and could lead to missing legitimate alerts.

Distribution of Copyrighted Material via Peer to Peer Applications

Several hosts on the network are participating in the sharing of copyrighted materials. In one example, we could see that the movie "The Count of Monte Cristo" was being advertised for download. While this kind of activity may not be a security risk per say, it does present a legal exposure and appropriate use of resources question.

Peer to Peer applications are also notorious for spreading hacked and virus infected code. This represents another point of entry into the network that completely bypasses the firewall, email scanners and all other perimeter defenses.

BugBear Activity

Looking at the Alert files shows a disproportionate number of connection attempts to UDP port 137 with an occasional follow up to port 139. The pattern of these connection attempts goes through the IP addresses on a given subnet in what is near sequential order. This is a strong indication that the scanning machines are infected with the BugBear virus/worm and are looking for other victims. The internal systems that may be compromised are ones that showed both a SMB Wildcard alert and a SMB C$ Access alert. This indicates that the internal host responded to the SMB Wildcard and then the external host attempted to connect to its C$ Share. If successful, the worm may have spread itself to the target host.

Possible Infected Internal Hosts (Hosts Showing Both SMB Alerts)		
MY.NET.132.20	MY.NET.137.34	MY.NET.190.17
MY.NET.132.22	MY.NET.137.35	MY.NET.190.19
MY.NET.132.24	MY.NET.137.36	MY.NET.190.41
MY.NET.132.26	MY.NET.137.46	MY.NET.190.100
		MY.NET.190.102

Each of these systems should be checked for infection. Reverse DNS lookup using NSLOOKUP on these machines either turned up no hostname or a name like "pooledxxx-xx.giac.edu". This could indicate that they are shared machines in areas such as library spaces or users personal workstation that are getting DHCP address assignments and therefore have no name in DNS. These kinds of systems would probably run Windows and are prime areas where mail might be read and attachments executed. These factors also make them likely candidates for this infection.

In the alert files, we can see (8) alerts caused by email messages coming into our mail router at MY.NET.6.40 but more importantly, there are (5) alerts of emails containing the Bugbear signature *exiting* our campus mail system. If we trust our IDS rule then these are clear indications of exposure and infection from BugBear.

10/11-17:16:14.341249 [**] Bugbear@MM virus in SMTP [**] **MY.NET.6.40**:42295 -> 65.212.73.209:25
10/12-14:53:13.578638 [**] Bugbear@MM virus in SMTP [**] **MY.NET.139.230**:2822 -> 64.156.215.5:25
10/12-14:53:25.357489 [**] Bugbear@MM virus in SMTP [**] **MY.NET.139.230**:2824 -> 64.12.137.89:25
10/12-15:24:18.750691 [**] Bugbear@MM virus in SMTP [**] **MY.NET.139.230**:2842 -> 128.231.4.227:25
10/15-23:25:23.626264 [**] Bugbear@MM virus in SMTP [**] **MY.NET.144.59**:55482 -> 128.183.107.56:25

If we could retrieve the message headers from the logs of the mail servers that sent and received the bugbear infected messages, we might be able to determine what users and computer sent or received the messages. This could help with tracking infected systems.

To clean up and prevent BugBear and other similar infections, all Anti-Virus software should be updated. Network shares should be properly protected with both share and file system permissions. Personal firewall software can also be installed to protect network shares.

Possible DoS Attempts

There were two types of possible DoS attempts against internal systems. The first is related to an ftp glob attack. These alerts were detected directed at MY.NET.100.158 from various hosts in Europe. The second type of DoS attempts were related to IP fragments. There were various types of alerts including Tiny Fragments sent to MY.NET.168.80 and MY.NET.91.240, fragments that never completed to several host and one fragment overflow attempt. These may represent an attempt to shutdown or hinder the performance of a particular host. These can be stopped by not allowing fragmentation at the border or by filtering malicious fragments with border devices.

Sun RPC Portmapper Exploit

Link Graph

The following Link Graph illustrates a possible RPC attack against MY.NET.151.115 that may have resulted in a successful compromise. For comparison, there is a graph of a similar attack against MY.NET.84.198 that may not have been as successful. Unless there is a clear explanation for the traffic pattern, a thorough exam of both hosts should be performed.

In the exchange between 65.59.116.64 (Attacker) and MY.NET.151.115 (Target), we can see that the Attacker first connects to port 32771 on the Target host. This is a common port that is used as a portmapper for RPC tools. These are known to have many exploits including buffer overflow issues that could be used to load grappling hook code that would then go to the Internet and download more robust exploit code for the attacker's later use.

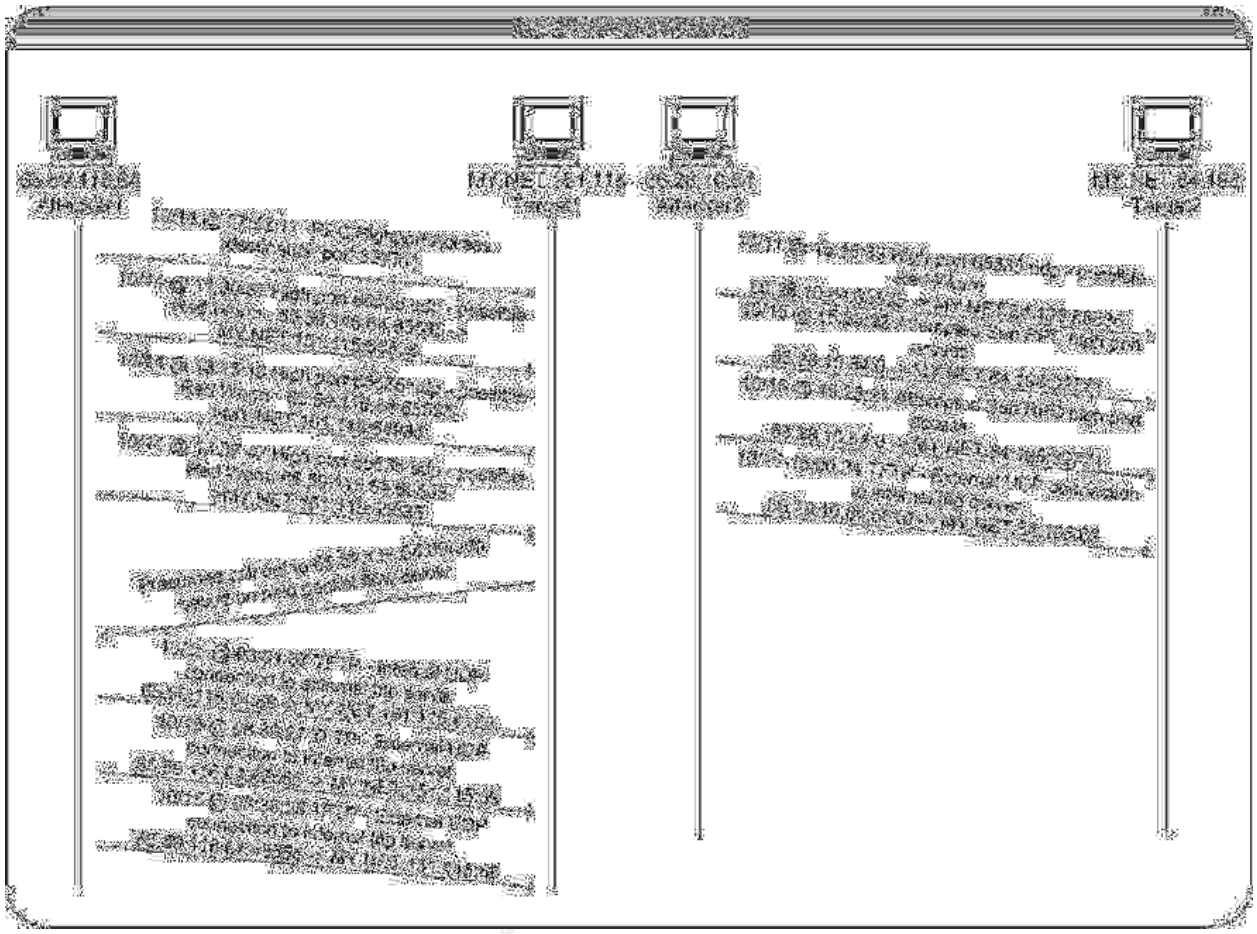
After this connection, we can see traffic from Attacker port 65535 to the Target on the same port. This port combination looks strange as most services should not be listening on this port. It also raises attention because it is a known port for worms including Adore (Red Worm). We see two additional packets from the Attacker's port 65535 to the Target (one to port 64935 and the second to port 65535). These interactions could have been commands to the code planted during the original RPC call above.

We assume there was a call out from the Target to port 69/udp on the Attacker because we see an alert that there is response tftp traffic from the Attacker's host to the target. In this attack scenario, this could be for the purposes of code retrieval based on the interaction we saw between the two hosts earlier.

Finally, we see the attacker making tftp connections to the target's port 69 (presumably tftp). This is possibly for retrieval of some collected data or other purposes.

On the right side of the graph, we see a similar observed between 66.28.10.84 (Attacker2) and MY.NET.84.198 (Target2). In this case, the initial RPC call was not seen until after an initial attempt from Attacker2 on port 65535 to Target2 port 65535. Later in the week, we see attempted RPC access to Target2 port 32771. This time the packets are from source port 0, this in an indication that the packets are forged and should not be trusted. Finally, we see an attempt from Attacker2 to connect to Target2 on port 69/udp.

We assume that the attacker may not know if the exploit to the RPC service worked or not, and by attempting connection to tftp port, they can check the success or failure.



Further inspection of these hosts is advised to see if there are any inappropriate listening ports or running services. A file integrity check my also be in order. If these systems were compromised at this level, a root kit may have been installed.

Host MY.NET.70.207 should be examined

This host had what appeared to be several IIS, RPC and other privilege escalation attacks launched against it. It also showed up as a heavy port scanner due to a large number of connections to it on 12203/udp and 12300/udp. It is believed that this system was hosting an online game (like Medal of Honor) off an on during the week. This may have drawn hostile attention to it. External host 169.229.70.201 did a fairly comprehensive port scan of this host. There is no one event that makes me suspicious of this host, but there area enough individual suspicious events that indicate this host should be checked for compromise or other dangerous activity.

Defensive Recommendations

Just based on the log entries, it is difficult to judge how secure the perimeter and internal network segments are. There are however a few recommendations that should be considered to improve the current security and reduce the probability of future events.

Above all else, all systems on the campus (especially the shared systems) should have Anti-Virus and Personal Firewall software installed. There are several products that can be centrally managed and updated so the administrative load is low compared to the benefit. Just by looking at the logs, we know that there are internal users scanning, using peer to peer services, hosting online games and possibly launching attacks on internal hosts. Perimeter firewalls have little effect on these kinds of activities so software that can protect at the host level would be a powerful addition to the sites security plan.

The second most important security step for this environment is to review of the SNORT configuration and rule base. There were a lot of what appear to be false positives generated. The trouble with false positives is that as long as they exist, you don't know which ones are real and which ones are false. This means you have to examine every alert in detail to see if it is legitimate or not. If the number of alerts in an environment is high, as in this case, you have a serious security issue for the site because it takes analysis time that could be spent working on real security issues and improvements. The following actions should be considered at a minimum:

- The IDS rules should be updated to use rules that examine content.
- Pass rules should also be installed for services and systems that we know will set off rules incorrectly.
- Rules that look for reserved bit set in the TCP flags need to be updated or removed since this is becoming common practice.
- The Snort http pre-processor may need some adjustments for this environment to eliminate the Unicode and Null Byte false positives.
- Snort Scan preprocessor should have its thresholds adjusted until the number of false positives from peer to peer and other "normal" traffic are reduced to a manageable level.
- IDS config and rules should be reviewed during each site security Audit.

Separate DNS systems should be used for internal and external name resolution. The DNS information on internal hosts should not be freely available on the public Internet. Using reverse lookup I was able to collect information that indicated to me where the physical assets were located and in some instances, what kinds of hardware or services were associated with those machines. This kind of information is extremely valuable when planning a socially engineered attack (ex. Sounding convincing with calling the University helpdesk for information or just walking onto the campus and gaining access to the physical assets).

Consider installation of Virus scanning software for the email gateways. We saw alerts that indicated BugBear was entering and leaving the campus mail system. These kinds

of threats that spread via multiple methods (shares, buffer overflows, email …) can only be stopped with a multi layer defense strategy that includes virus control, perimeter defense and hardening of the hosts. Since these blended threats are on the rise, this is an ounce of prevention that could save several pounds of cleanup down the road.

There were several scans and connection attempts for services with known vulnerabilities. If there are no legal restrictions prohibiting GIAC University from closing these ports from the outside, they should consider doing so. Allowing access to what are often unsecured services from external hosts is a dangerous practice.

As a general rule, all traffic in or out of the network should be blocked at the firewall unless there is a specific organizational reason to allow it. This will prevent systems from becoming compromised, limit the usefulness of systems that do become compromised and help prevent your site from becoming an unknowing launching pad for other attacks. This network environment is overrun with Peer to Peer applications like KaZaa, Gnutella, WinMX … This traffic, while wasteful of the University's resources, in and of itself is not necessarily dangerous. However, these software swapping grounds are notorious for having tainted software on them and could become a pathway for virus and other attacks to enter your network. Another issue with these services is that when you connect to the other systems your IP is shared. This often times draws the wrong kind of attention from attackers. Finally, there are legal issues related to the distribution of Copyrighted material like music, movies and software. Allowing this on your network may create legal exposure for GIAC University at some point in the future.

The University should carefully review its policy on this kind of traffic. These discussions should involve legal counsel and the security team. If there are no legal or organizational barriers, blocking this sort of traffic at the firewall and even on the internal LAN segments would improve the security of the site and reduce bandwidth consumption. If this is not an option, an information campaign to share the dangers and methods to defend against them should be used to help mitigate the risk.

GIAC University should schedule regular audits and vulnerability tests of the firewall, internal systems and infrastructure. Work spent detecting and correcting vulnerabilities before they are exploited reduces overall security costs versus the cost to recover should an exposure get exploited. These exercises should be done at least quarterly, but an automated system that continuously monitors for vulnerabilities would be preferred.

Finally, cleanup the SCANS files to obfuscate the IP addresses of the University. These are freely downloadable from incidents.org and that means that the attackers can get them as well. Using DNS reverse lookup and other tools; I was able to get a lot of information about the internal network once I knew the real network addresses. This was helpful for my work, but it would also be helpful for those with malicious intent.

Overall Analysis Process

To perform my analysis of this data, I first had to clean the log files of incomplete or incorrectly formatted entries. I used regular expressions and PERL scripts to look for all

entries that did not match the correct format and then corrected them or discarded them into a separate file. I then reviewed the file with the unrecoverable entries in it by hand.

To process the bulk of the entries, I again turned to PERL's hash features to total up the data in different ways. I used the same basic script format and just modified the key structures depending on what data I needed for each run (key on source IP, target IP, by conversation – which was a combination of source IP and destination IP, … In each case, I used the scripts to total the number of attacks in each category or scans by type and output the data to CSV format. I also wrote a small script to format the OOS records in CSV format.

Once the data was in CSV format, I loaded it into an Excel spreadsheet and used the sorting, filtering and other data analysis tools to view the data in different ways.

When I needed to see more specific details for an IP address or port, I turned to GREP, FIND and MORE and applied them against the log files.

My basic concept was to get the attacks totaled up and then to look at the individual attacks one by one to understand what each one was trying to do. Looking at the sources and destinations ports for each alert and then using DNS to find the nature of the source and destination host gave insight into whether it was a real attack or a false positive. Given the volume of data, I had to pick a focus for the analysis. I chose to start my focus on the scanners. I made this choice based on the general flow that a hacker goes through to execute a successful attack. They need to do their recon work first so I figured that if we saw them scanning, then we might see them doing more targeted work later.

Looking at the top scanners and top sources for alerts helped me see the large amount of peer to peer traffic that this University has to deal with. It also shed light on a potential BugBear issue and some online gaming activity at the site. As I discovered the nature of a certain traffic profile, I used "grep –v " commands to filter those entries out of the logs. This allowed me to see the remaining entries with less "noise" as I looked for other signs of trouble.

As I found possibly compromised systems or other issues in this section of the analysis, I looked at other alerts and scans related to the hosts involved.

Example of PERL Script Used for Analysis

I wrote this script and then hacked it up as needed to get different data from the files. Each script was based in one way or another on the basic form below. I am sharing it for informational purposes only. Hopefully, it will help get some folks started or at least get some good laughs from real programmers 8-)

countalerts.pl

To use this script, call it and give it the names of your alert files as arguments. The output will be a list of Source IP Addresses and the number of times they were the source for a

particular alert event. The use of hash of hashes was particularly useful to me to identify unique elements and to keep the code size down.

```
# Read file and count alerts by name and direction and total
# Usage: perl countalerts.pl <file1> <file2> ... <filen> > <outputfile.csv>
# Then load into your favorite spreadsheet program for sorting, filtering, ...
while (<>) {
# Ignore spp_portscan entries - get from scan files w/ different script!
    if (!/spp_portscan/) {
        m/(\S+).*] (.*)\[.*] (.*) -> (.*)/;
        (my $EventDate, my $EventTime) = split(/-/,$1);
         my $EventName = my $Event = $2;
        (my $SrcIP, my $SrcPort) = split(/:/,$3);
        (my $DstIP, my $DstPort) = split(/:/,$4);

#Set Direction of Attack for recording count
        ($SrcIP =~ /MY.NET./) ? $Dir="I" : $Dir="O";
        ($DstIP =~ /MY.NET./) ? $Dir .="I" : $Dir .="O";

        $AlertCount{$Event}++;
        $DirCount{$Event}{$Dir}++;
        $TotalEvents++;                                                    }
}

# Print Header Row
print"Alert,OI,II,IO,OO,Total\n";

#Output Each Alert Name and the number of times seen
foreach $Eventtype (sort keys %AlertCount) {
        print "$Eventtype,";
        print "$DirCount{$Eventtype}->{OI},";
        print "$DirCount{$Eventtype}->{II},";
        print "$DirCount{$Eventtype}->{IO},";
        print "$DirCount{$Eventtype}->{OO},";
        print "$AlertCount{$Eventtype}\n";
}

#Output Total Alert Count
print"\nTotal Events = $TotalEvents";
```

List of References

Whitehats - arachNIDS Reference (http://www.whitehats.com)

- TROJAN_TROJAN-ACTIVE-Q-TCP - http://www.whitehats.com/info/IDS203
- DDOS-TFN-CLIENT-COMMAND-BE - http://www.whitehats.com/IDS/184
- DDOS-SHAFT-CLIENT-TO-HANDLER - http://www.whitehats.com/IDS/254
- SHELLCODE-X86-SETGID0 - http://www.whitehats.com/IDS/284
- SHELLCODE-X86-SETUID0-UDP - http://www.whitehats.com/IDS/436
- DOS-FTPD-GLOBBING - http://www.whitehats.com/IDS/487
- NTPDX-BUFFER-OVERFLOW - http://www.whitehats.com/IDS/492
- IIS ISAPI OVERFLOW IDA - http://www.whitehats.com/info/IDS552
- NETBIOS-SMB-C$ACCESS - http://www.whitehats.com/info/IDS339

GIAC Student Practicals

- Treurniet, Joanne, September 22, 2000 SANS GIAC Practical Assignment for GCIA, <http://www.giac.org/practical/JoanneTreurniet.html>
- Peck, Edward - GCIA Practical Version 3.0 – 2001 - <http://www.giac.org/practical/Edward_Peck_GCIA.doc>
- Lam, Jason - Intrusion Detection in Depth – GCIA Practical Assignment Version 2.9. <www.giac.org/practical/Jason_Lam_GCIA.doc>
- Oborn, David - SANS GCIA Practical Assignment <http://www.giac.org/practical/David_Oborn_GCIA.html#detect4>
- Fiddler, Matthew – May 10th, 2002 - GCIA Practical Assignment - Version 3.0 - <http://www.giac.org/practical/Matthew_Fiddler_GCIA.doc>
- Allen, Jennifer - WU-FTPD Heap Corruption Vulnerability - GCIH Practical Assignment v2.0. Dec. 2001. < www.giac.org/practical/Jenn_Allen_GCIH.doc>
- Ellis, Joe – GCIA Practical Assignment v3.0, Intrusion Detection In Depth – May 14th, 2002 - <http://www.giac.org/practical/Joe_Ellis_GCIA.doc>
- Poor, Mike - Intrusion Detection in Depth. GCIA Practical Assignment v3.0 - <http://www.giac.org/practical/Mike_Poor_GCIA.doc>
- Neel, Robert - http://www.giac.org/practical/Robert_Neel.doc
- Mohan, Potheri - <http://www.giac.org/practical/Potheri_Mohan_GCIH.doc>
- Stearns, William – 2000 - http://www.sans.org/y2k/practical/william_stearns_gcia.html

CERT

- IN-2002-04 - http://www.cert.org/incident_notes/IN-2002-04.html
- CA-1995-15 - http://www.cert.org/advisories/CA-1995-15.html
- IN-2000-02 - http://www.cert.org/incident_notes/IN-2000-02.html
- CA-2001-07 - http://www.cert.org/advisories/CA-2001-07.html

Common Vulnerabilities and Exposures (http://cve.mitre.com)

- CVE-1999-0189 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0189
- CVE-2000-0225 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0225
- CAN-1999-0454 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0454
- CAN-1999-0495 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0495
- CAN-1999-0660 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-1999-0660
- CVE-1999-0683 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0683
- CVE-1999-0804 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0804
- CAN-2000-0544 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0544

- CVE-2000-0917 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0917
- CAN-2002-0154 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0154
- CAN-2002-0677 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0677
- CAN-2002-0678 - http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0678

Microsoft

- MS-SQL Server Security Information Web Page - http://www.microsoft.com/sql/techinfo/administration/2000/security.asp
- MS-SQL Server Security Information Web Page - http://www.microsoft.com/sql/evaluation/features/security.asp
- MS Bulletin MS00-057 - Patch Available for 'File Permission Canonicalization' Vulnerability - August 10, 2000 - http://www.microsoft.com/technet/security/bulletin/ms00-057.asp
- IIS4.0 Security Update, August 15, 2000 - http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/
- IIS 5.0 Security Update, August 15, 2000 - http://www.microsoft.com/windows2000/downloads/critical/q269862

Bugtraq

- BugTraq ID 1712 - http://www.securityfocus.com/bid/1712
- BugTraq ID 576 - http://www.securityfocus.com/bid/576
- BugTraq ID 2540 http://www.securityfocus.com/bid/2540

Additional References

- Entercept - Multi-Vendor Remote Buffer Overflow Vulnerability in CDE ToolTalk Database Server - http://www.entercept.com/news/uspr/08-12-02.asp
- Mixter - .mixter security web site - http://mixter.warrior2k.com
- Sage, John - May 13, 2002 - http://www.incidents.org/archives/intrusions/msg11889.html
- HPING Web Site - http://www.hping.org/
- ISS Alert – http://bvlive01.iss.net/issEn/delivery/xforce/alertdetail.jsp?id=advise5
- Symantec.com - W32.Bugbear@mm Security Response - http://securityresponse.symantec.com/avcenter/venc/data/w32.bugbear@mm.html
- McAfee Security - http://vil.nai.com/vil/content/v_99728.htm
- Sophos.com - http://www.sophos.com/virusinfo/analyses/w32bugbeara.html
- Williams, Shane - http://www.mcabee.org/lists/snort-users/Oct-02/msg00067.html
- Dittrich, David. The "Tribe Flood Network" distributed denial of service attack tool. Oct. 21st, 1999 <http://staff.washington.edu/dittrich/misc/tfn.analysis>
- AMP Information Website - http://sd.wareonearth.com/woe/amp.htm
- NLANR Website - http://moat.nlanr.net/
- Sven Dietrich - Analysis of the Shaft distributed denial of service tool - 16 Mar 2000 - http://security.royans.net/info/posts/bugtraq_ddos3.shtml
- mms://195.92.252.254/jazzfmstation
- Incidents.org - Code Red II Worm Analysis Update – August 7th, 2002 - http://www.incidents.org/react/code_redII.html
- Incidents.org – Nimda Worm/Virus Report – October 3rd, 2001 - http://www.incidents.org/react/nimda.pdf
- WJSolutions.com - http://www.wjsolutions.com/scanner/?curpage=SummaryScan
- descan.net - http://www.descan.net/searchresults.html?command=specific&source=24.123.46.10
- Ruiu, Dragos – spp_defrag.c – Snort defrag pre-processor source code

- Web site reporting Adore contact from telia.com hosts - http://www.rud.dk
- SANS - Adore Worm - Version 0.8 - April 12, 2001 <http://www.sans.org/y2k/adore.htm>
- TonikGin – XDCC – An .EDU Admin's Nightmare" – September 11th, 2002 - <http://www.russonline.net/tonikgin/EduHacking.html>
- Pest Patrol – About Ports and Trojans – 2002 – http://www.pestpatrol.com/Support/About/About_Ports_And_Trojans.asp#portlist
- Cramer, Christopher - http://www.theorygroup.com/Archive/Unisog/2002/msg00677.html
- Ruiu, Dragos – Feb. 12th, 2002 - http://archives.neohapsis.com/archives/snort/2001-02/0320.html
- Marquette University – Virus Alert – October 2002 - http://its.marquette.edu/virus/alert.html
- Symantec.com – W32.Myparty@mm Security Response - http://securityresponse.symantec.com/avcenter/venc/data/w32.myparty@mm.html
- Chappell, Laura – "Your Being Watched" – March 2001 – <http://www.nwconnection.com/2001_03/pdf31/cybercrm31.pdf>
- Graham, Robert http://www.robertgraham.com/pubs/firewall-seen.html
- Insecure.org - http://www.insecure.org/nmap
- http://vger.kernel.org – reference to site using TCP / ECN
- Bill Manning – May 26th, 2002 - Documenting Special Use IPv4 Address Blocks that have been registered with IANA/RIR - http://www.ietf.org/internet-drafts/draft-manning-dsua-08.txt
- Ramakrishnan, K. – RFC 3168 – The Addition of Explicit Congestion Notification (ECN) to IP – September 2001 - <ftp://ftp.isi.edu/in-notes/rfc3168.txt>
- AT&T Laboratories Cambridge, 1999 - <http://www.uk.research.att.com/vnc/winvnc.html>
- Alexander, Bryce – Port 137 Scan – May 10th, 2000 - <http://www.sans.org/newlook/resources/IDFAQ/port_137.htm>
- Alexander, Bryce, SANS **–** Followup on a Honeypot Catch **-** 2000 - <http://www.sans.org/y2k/honeypot_catch.htm>
- Martin Roesch – Re: [snort] Tiny Fragments - May 14th, 2000 - http://archives.neohapsis.com/archives/snort/2000-05/0103.html
- John Berkers and Andrew Daviel – [snort-users] group posting – 2001 - http://archives.neohapsis.com/archives/snort/2001-08/0528.html
- Google - www.google.com
- VisualWare Web Site – http://www.visualroute.com
- Snort Web Site – http://www.snort.org
- Martin Roesch, Chris Green - Snort Users Manual v1.9.1 - http://www.snort.org/docs/writing_rules
- Dragos Ruiu - Snort FAQ v1.8 - http://www.snort.org/docs/faq.html
- American Registry for Internet Numbers – http://www.arin.net
- Réseaux IP Européens – http://www.ripe.net
- Asia Pacific Network Information Center – http://www.apnic.net/
- Stevens, Richard W. TCP/IP Illustrated, Volume 1 The Protocols, 1994 Addison Wesley
- Northcutt, Stephen and Judy Novak. Network Intrusion Detection, An Analysts Handbook 2nd Edition. Indianapolis, IN: New Riders Publishing, 2000
- Northcutt, Stephen et. all. – IDS Signatures and Analysis, Parts 1 and 2 – SANS courseware – 2002
- Joakim von Braun – Intrusion Detection FAQ - What port numbers do well-known trojan horses use? - 2000 - http://www.sans.org/newlook/resources/IDFAQ/oddports.htm