GIAC Certified Intrusion Analyst (GCIA)

Practical Assignment Version 3.5

Benjamin Fabian

August 16, 2004

Abstract

This work is a practical to fulfill the requirements of certification as a SANS Institute's GIAC Certified Intrusion Analyst (GCIA), Assignment version 3.5 (see www.sans.org and www.giac.orc). It consists of three main parts: Design an Enterprise Intrusion Detection Architecture, Analysis of three network detects, Analysis of a week of network traffic.

Practical Assignment Part One - Option One:

I. Design an Enterprise Intrusion Detection Architecture

It is nice to offer a honey pot to a stranger who appeared on your doorstep in the middle of the night ... [1]

Executive Summary

The company GIAC Fortune Cookies (GIAC FC) was inspired by the GCFW paper of Alfredo Lopez. [2,3]

The principal Internet business model of GIAC Fortune Cookies (GIAC FC) has stayed the same as described by security consultant Alfredo Lopez. GIAC FC has specialized in online-selling of Fortune Cookie Sayings. In the last year GIAC FC saw a tremendous increase in sales, which let to worldwide dominance in this very lucrative but sensitive market.

The downside of market success was an increase of security incidents by sophisticated attackers, which two times succeeded in compromising the main cookie server and placed some counterfeit fortune sayings, that were not as accurate as the original texts and included elusive misinformation. There had been legal actions by disappointed customers like international governments and intelligence agencies, who had used to trust the GIAC FC fortune sayings more than their own informational networks and analysis teams. This resulted in serious financial losses and diminished credibility. As a reaction GIAC FC decided to deploy intrusion detection systems (IDS) to monitor hosts and networks, which should also give a detailed insight into the way expected future attackers proceed. The existing infrastructure should also be made more resilient to network outages.

GIAC CF still has only one central location that needs protection; it allows inbound VPN connections by Cookie text providers and Sales people. The most important points were securing the Service Net, where customers buy the

[1] W. Pooh: "Pooh's 101 Uses for a Honey Pot", Entry No. 1, New York 1997.
[2] Please note that all links given throughout this document have been valid in August 2004.
[3] Alfredo Lopez, GCFW Practical describing the company: http://www.giac.org/practical/GCFW/Alfredo_Lopez_GCFW.pdf

cookies, and securing the log files and device management to collect judicially sound evidence on break-in attempts.

The approach was to implement a "Defense-in-Depth" strategy for IDS, too, which was already in place for firewall architecture. We chose a layered, multi-vendor approach, which included different kind of IDS: Network, Host, and Deception based IDS. For example: Even if a server in the Service Net gets compromised, the real time delivery of logs concerning the first break-in phase to additionally protected log servers could not be made undone by the attacker. Network IDS observing traffic in the Service Net would still be out of reach for the attacker by physical separation.

Paranoia was aimed for - a fair amount of paranoia we delivered, still manageable and affordable.

1. Intrusion Detection Systems used

Symantec SESA and Incident Manager

The main goal is on concentrating log file and alarm collection throughout all or at least most of the deployed security systems into a single architecture, and to monitor security events from a single console. As a central collecting instrument we chose Symantec's SESA (Enterprise Security Architecture) in its version 2, combined with Symantec Incident Manager.[4]

SESA consists of the SESA manager, the data store and the directory, which we will deploy in a distributed fashion on dedicated servers each. Since version 2.0 SESA supports Oracle version 9i on Solaris 8 Sparc as a SESA DataStore; we will use a Sun Fire V440 Server[5] for it (4*1.28 GHz, 16 GB RAM, Solaris 8). The SESA Manager and SESA Directory are now also supported on Solaris 8. We choose a Sun Fire V250 Server[6] (2*1.28 GHz, 8 GB RAM, Solaris 8) for each.

Though at the moment not all used security systems – not even every used Symantec product - can be integrated directly with SESA, this integration seems to be a strong focus as newer versions of established products are being released.

In the meantime there are workarounds like using special event collectors (in the case of Symantec Antivirus and Cisco PIX) or in the case of Snort using integration with ManHunt first which itself uses a "bridge" to be integrated with SESA.

[4] Symantec Incident Manager: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=166&EID=0
[5] Sun Fire V440 Server: http://www.sun.com/servers/entry/v440/index.xml
[6] Sun Fire V250 Server: http://www.sun.com/servers/entry/v250/index.xml

The products without SESA support are the Cisco routers and the open source Honeyd, for which we deploy a master Syslog server in the SESA Net, having the option of correlating these logs someday by using some open source solution which might give us a kind of fall back to the SESA approach, which would fit into the dual nature of all deployed security solutions. At the moment no such product will be recommended, however.

On the SESA Manager we deploy Incident Manager, a powerful log aggregation and correlation engine, which refines the large amount of log data from SESA-enabled devices into manageable incidents that can be displayed in a single console.

However, as always in general intrusion detection, the real work will be done over months after the deployment when signatures are being tuned, normal traffic identified and possible network misconfiguration solved, so that a real white listing of traffic can be achieved. At that state IDS and firewall rules can be tightened and unusual and suspicious traffic more easily identified.

Screening Routers

The original single screening router Cisco 7206 VXR[7] will be joined by a second router of its kind that connects to a different ISP; to handle routing in this scenario HSRP and BGP are being used.[8] This setup gives better connectivity to the rest of the Internet and can provide failover in case the one of the ISP-connections drops.

We refrain from using the Cisco IOS IDS-features on these routers mainly due to possible performance issues and the fact that attacks already blocked at these routers have no high importance for us. The following PIX firewalls will have roughly equivalent signatures and alarming capabilities. But there is room for experimenting with this option, if a complete picture is needed and if observed performance allows it.

Logging and managing will be done by use of a dedicated interface connected to Management & Log Net II, where a Syslog server resides, which doubles all of the messages it receives to a Master Syslog in the SESA Net, to give redundancy.

Cisco PIX Firewall with basic IDS features

The first real firewall in the original design is a PIX 535-UR firewall[9,10], which will be kept in the new concept, however doubled it in a clustered configuration for stateful failover.

---

[7] Cisco 7200 Routers: http://www.cisco.com/univercd/cc/td/doc/pcat/7200.htm
[8] How to Use HSRP to Provide Redundancy in a Multihomed BGP Network: http://www.cisco.com/warp/public/459/hsrp_bgp.pdf
[9] Cisco PIX Firewall: http://www.cisco.com/go/pix
[10] Cisco PIX 535: http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/535_ds.pdf

Besides its fast stateful inspection we have a special interest in the basic IDS capabilities of the PIX Finesse operating system, which will be upgraded to version 6.3.

The following steps activate these IDS features:

First so-called policies are created for signatures deemed informational, then for those considered attacks; we can define different policies for inside, outside and DMZ, which might be useful in giving a bit more granularity in attack responses. Here is an example for an attack policy:

```
pixie01(config)# ip audit name ATTACKPOL attack action alarm drop
pixie01(config)# ip audit interface outside ATTACKPOL
```

If an signature classified as an attack triggers, the PIX sends and alarm to the Syslog server and drops the packet – this behavior is defined by the options following the "action" keyword; we could also have reset the connection instead of dropping it silently, or just alarm on it. These commands refer to whole classes of signatures. Single signatures can be disabled globally by using their reference number:

```
pixie01(config)# ip audit signature 6100 disable
```

All in all this system is not as flexible as a full-fledged IDS but sufficient and customizable enough to give some supporting information to the network analysts.

The PIX 535 firewalls have an 1 GHz processor, upgraded SDRAM of 1 GB and will be equipped with 4 PIX-1GE-66 Gigabit-NICs in slots 0 to 3 for outside, inside, DMZ and stateful failover (shown as a small line in the diagram).

A fifth Fast Ethernet NIC will be used for logging and remote management.

The Pixies will log through these dedicated interfaces to the Syslog server in the Management & Log Net II, where a SESA event collector will report the PIX messages to the SESA manager for central reporting and monitoring.

Symantec Gateway Security: Application Layer Firewall with IDS

The former pair of Raptor firewalls will be upgraded by a cluster of two Symantec Gateway Security 5400 firewall appliances (SGS)[11], which are mostly pure application layer firewalls equipped with improved clustering and intrusion detection capabilities. It features Protocol Anomaly Detection (PAD), IDS signatures and traffic analysis.

[11] Symantec SGS 5400: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=133&EID=0

The SGS will be managed through a dedicated interface, which connects to the SESA manager via Management & Log Net II.

A third SGS will be deployed internally with very strict rules settings to give additional protection to the SESA Net, which includes the SESA manager, directory and database as well as the Master Syslog server.

To see the ease in configuring signatures on a SGS you can navigate to an example signature. After having logged in via HTTPS to default port 2456 first verify that your license includes the IDS features: System -> Features -> License Summary. Then check that "Intrusion detection and prevention" is activated globally under System -> Features -> System Features. Now you can navigate for example to:

Policy -> IDS/IPS -> Base Event Types -> HTTP_ETC_PASSWD_ACCESS -> Properties

Besides some information about the signature and references you will find the following buttons very useful for tuning the IDS component:

- Enable / disable the signature
- Gated (drop traffic) / not gated (permit traffic)

As stated before the SGS also supports PAD and traffic anomaly detection besides signatures.

Symantec Manhunt

The bulk of intrusion detection will be carried by two Symantec ManHunt 3.02[12] machines (MH) which each will use several monitoring interfaces connected to SPAN switch ports for stealthy network sniffing at crucial points. Each MH will monitor different subnets with dedicated interfaces. MH is capable of sniffing multiple Gigabit links with success[13], so it is very well suited for monitoring even several of the high traffic lines.

Administration and logging will take place by using a dedicated NIC to the Management & Log Net II, which connects to the central SESA manager. At the current time MH communicates with SESA by the use of a Manhunt-SESA bridge.[14] It is expected that future releases of MH will interact directly with SESA.

All MHs will be deployed in the form of iForce appliances based on hardened Solaris x86.[15,16] Ease of installation is combined with the possibility of using the

[12] Symantec ManHunt: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=156&EID=0
[13] NSS Gigabit IDS test: http://www.nss.co.uk/gigabitids/edition2/index.htm
[14] Cf. "Symantec ManHunt SESA Bridge Installation Guide", delivered with this product add-on.
[15] Symantec iForce page: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=284&EID=0
[16] Sun iForce page: http://www.sun.com/aboutsun/media/presskits/symantec_iforce_ids/

iButton technology[17] to guarantee the authenticity of evidence collected by these machines. We choose two V60X-2000C models[18] (4 Gigabit copper interfaces, vendor given maximum of analyzing 2 GBit/s from all interfaces, 4 GB RAM, 36 GB hard disk).

All IDS systems will have to undergo a tuning process to reduce false positives. For signature tuning from the MH console choose Setup -> Policy -> Filter Rules or press (SHIFT- F). Choose the node on which you want to filter signatures on. The "Filter Rule" window opens; pressing "Add" lets you specify which combination of event (signature in larger sense, includes anomalies), source and destination IP (ranges) MH should ignore. The JAVA-GUI generates a filter text file under `/usr/manhunt/etc/espfilters.txt`, which you also could modify manually. No restart of any processes should be necessary. An example for an `espfilters.txt` filter file:

```
# ManHunt event droplist file
# Edit with care!
#
# rule format:
#
# !begin_rule
# !drop
# !matchtype
# DNS_DATA_AFTER_END
# HTTP_UNKNOWN_STATUS
# !matchiplist
# 172.16.0.0/16:*, *:*
# !end_rule
#
# in IP lists * is a wildcard for address and port.
# example:
#
# src/mask:port        ,   dst/mask:port
#
# 145.1.45.67/32:6780  ,   123.56.78.9/32:37456
# 134.34.56.7/32:*     ,   *:80  # all ports from 134.34.56.7; any IP to port 80
!begin_rule
!drop
!matchtype
COUNTER_ICMP_HIGH
!matchiplist
10.0.0.0/16:*,10.0.5.1:*
!end_rule

!begin_rule
!drop
!matchtype
HTTP_BAD_REQUEST
!matchiplist
10.0.0.0/16:* , 10.0.5.5:80
!end_rule
```

The first rule removes false alarms due to network management and uptime checks, the second due to internal access to a non RFC-conform web server. Note that for a single host the addition of "/32" might cause the signature not to be recognized, so just use the IP like MH itself does when you are modifying the filter list from the GUI.

[17] iButton technology: http://www.ibutton.com/ibuttons/index.html
[18] iForce FAQ, models: http://enterprisesecurity.symantec.com/content/displaypdf.cfm?pdfid=625

An interesting detail on MH filtering is that filtered events still appear in the logs after some aggregated event count limit is reached. As is stated in the MH User Guide, p. 84:

"The first event that matches a filter list entry is sent to the Analysis Framework. After that, it will ignore a certain number of these types of events. The number to ignore is specified as a value for the Drop Filter Threshold configuration parameter. If it is not defined, the default is 500. When the drop threshold has been reached, one matching event will again be sent to the Analysis Framework and displayed in the administration interface. The event will indicate how many dropped events occurred in the aggregation count for the event."

If there are still events generated after filtering, you probably see an aggregated event count of 500 - the default value. You can change this to a much higher value (for all filters on a node) under: Configuration -> Man Hunt Node -> Local Node Parameters -> (node name) -> Incidents/Event Parameters -> Drop Filter Threshold.

Snort

The Snort IDS in the service network is kept and upgraded to version 2 - this is to have a multi-"vendor" approach in place, in case one IDS misses something or is compromised by an attack. This mirrors the "Defense-in-Depth" approach used in the firewall deployment.

The "ManHunt Smart Agent for Snort 2.0"[19] enables MH to collect events from the snort alert file, convert these to MH format and correlate them with its own events. As MH itself reports to SESA this enables the integration of Snort into the proposed monitoring and alerting architecture.

The manual for this "MSA for Snort" states as system requirements Solaris 8 Sparc or Intel, Red Hat 8 or Mac OSX, if you install it on the same system as Snort, which is what we do; if you configure Snort for sending alerts to another machine, you could install it there. As most of the servers run Solaris 8, we will use a Sun Fire V 120 (650 MHz UltraSparc III, 4 GB, 2* 73 GB hard disks, Solaris 8 with all patches).[20]

Tuning Snort signatures can be done multiple ways. First you can choose to not include whole signature classes from the `snort.conf` by commenting them out:

```
# include $RULE_PATH/abc.rules
```

Or you comment single signatures out in `abc.rules` itself. A third possibility is to modify rules themselves or write them new. For this approach confer to "How to

[19] Cf. "MSA for Snort 2.0 Installation Guide", delivered with this product add-on.
[20] Sun Fire V120 Server: http://www.sun.com/servers/entry/v120/

Write Snort Rules and Keep Your Sanity"[21] or the Snort 2.1 book by Baker, Caswell, Poor et alii[22].

If you like to disable a signature just for a particular host or subnet, but keep it globally switched on an unmodified, you can use the inclusion of a `threshold.conf` into your `snort.conf`:

```
include threshold.conf
```

You certainly could enter the content of this included file directly into snort.conf, but this approach seems to better manageable.

Example: Disable the following Snort signatures just for destination IP 10.10.10.50 (false positives by some kind of Windows logon traffic):

```
10015   DCE RPC Interface Buffer Overflow Exploit
2175    NETBIOS SMB winreg access (unicode)
538     NETBIOS SMB IPC$access
```

Enter the following into your `threshold.conf`:

```
suppress gen_id 1, sig_id 538, track by_dst, ip 10.10.10.50/32
suppress gen_id 1, sig_id 2175, track by_dst, ip 10.10.10.50/32
suppress gen_id 1, sig_id 10015, track by_dst, ip 10.10.10.50/32
```

This suppresses the signature locally.

Symantec Host IDS

It is most advisable to protect every mission-critical server by using host-based intrusion detection systems. In the context of the presented security solution we will deploy Symantec Host IDS 4.1[23] (SHIDS), which can be seamlessly integrated into the SESA architecture. SHIDS is available for Windows and Solaris - the latter fact is meeting our needs with the dominance of Solaris 8 servers in the company.

SHIDS will be watching the main and internal (production) Fortune Cookie servers, the web, DNS and mail servers as well as the Syslog servers. Each host has a second network card that is reserved for IDS management and log traffic.

At the heart of SHIDS lie the policies:

"A policy is a group of rules that are designed to detect a particular type of suspicious activity and take appropriate action. For example, a policy might be designed to watch for more than three failed logon attempts to a single account within a short period of time and disable the account if those conditions are met."[24]

---

[21] Snort Documentation: http://www.snort.org/docs/snort_manual/node9.html
[22] A. Baker, B. Caswell, M. Poor: Snort 2.1 - Intrusion Detection. Syngress, Rockland, 2004.
[23] Symantec Host IDS: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=48&EID=0
[24] Symantec Host IDS 4.1 Courseware, p. 232.

Information about default policies can be gathered from the SESA console after installation and setup, which includes installing the "SESA Console Extensions from the Symantec Host IDS". Navigate to Symantec Host IDS -> Policy Library -> Policy Information. Preconfigured policies can be used as templates for new, custom policies that can be created and edited by the Policy Editor. A vast realm of customization and tuning is available that way.

Honeyd

To add a deception environment we deploy Honeyd[25] in the Service Net. This low interaction honey pot will be configured to emulate listening host at until now unused IP addresses. Contact attempts to these fake hosts should never occur in regular traffic and should raise an alarm.

Configuration example[26], `honeyd.conf`:

```
### Solaris ###
create solaris
set solaris personality "Solaris 2.6"
set solaris default tcp action reset
set solaris default udp action reset
add solaris tcp port 21 "sh /home/honeyd/honeydscripts/ftp.sh"
add solaris tcp port 110 "sh /home/honeyd/honeydscripts/pop3.sh"
add solaris tcp port 25 "sh /home/honeyd/honeydscripts/smtp.sh"
bind 192.168.11.5 solaris
bind 192.168.11.6 solaris
bind 192.168.11.7 solaris
...
```

This tells Honeyd to listen at IP 192.168.11.5 (.6, ...) and respond to Nmap scans on this IP in a way that Nmap things it sees a Solaris 2.6 system; Honeyd is using the same fingerprint file to emulate that Nmap uses to analyze, so this should work quite well. Upon connections to ports 21, 25, 110 the given scripts will be started, which could emulate server banners and capture exploits. The same profile "solaris" will be applied to every free IP in the subnet by adding a line "`bind 192.168.11.X solaris`" for each IP address. Management and Syslog-style logging will take place by using a second NIC and the dedicated sub net. We use a Sun Fire V 120 (650 MHz UltraSparc III, 4 GB RAM, 2* 73 GB hard disks, Solaris 8 with all patches).

Symantec Decoy Server

For high interactive deception a Decoy Server[27] with four cages is mirroring the four main servers in the service network. This is called a "Minefield Deployment", as the honey pots or cages are situated in the same subnet as the production servers. The cages mirror the main servers and might be able to distract attackers from the real servers long enough for countermeasures to be taken.

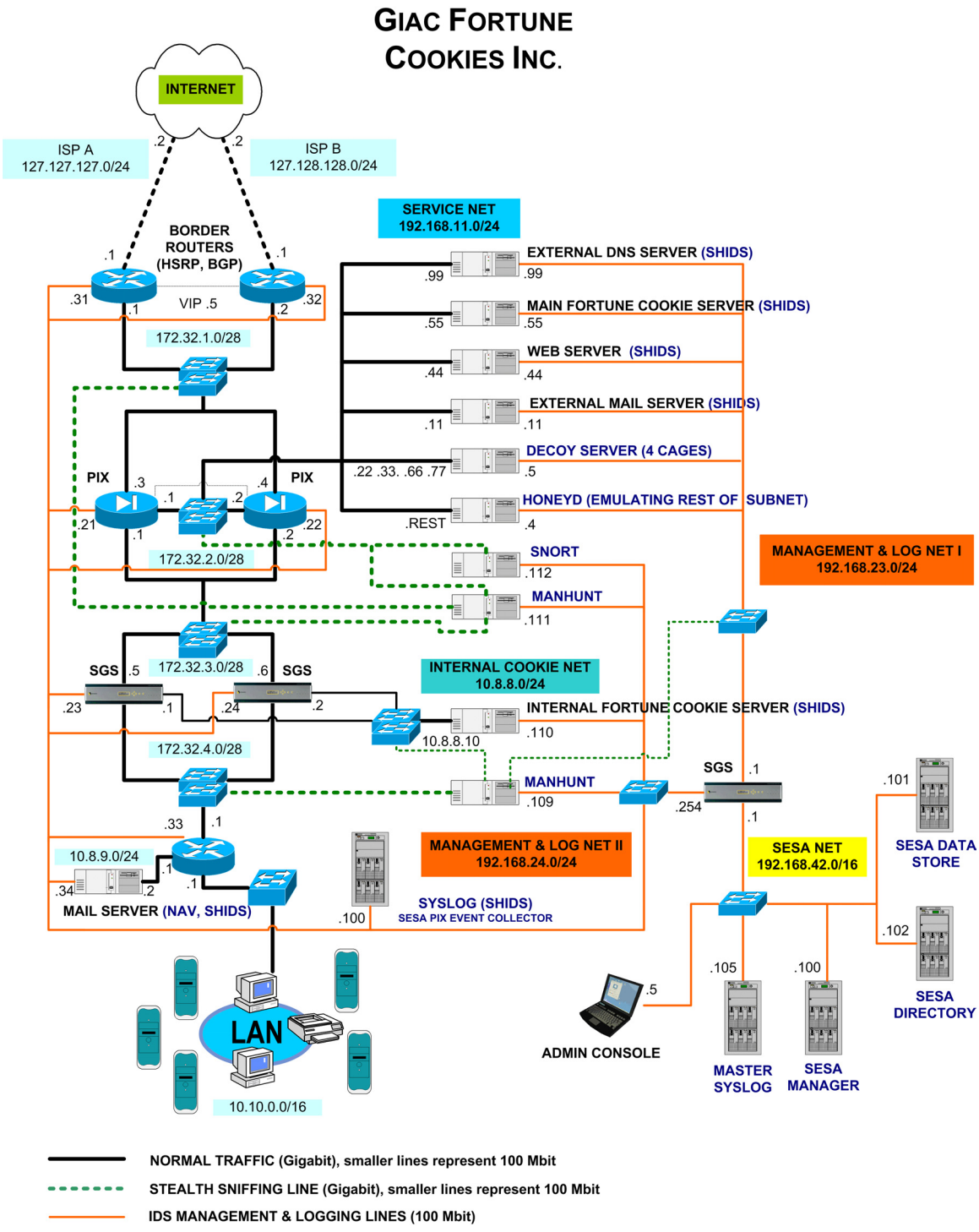
[25] Honeyd virtual Honey pot: http://www.honeyd.org/
[26] Inspired by John Lyons, "Honeypots & deploying Honeyd": http://www.sage-ie.org/slides/honeypots.pdf
[27] Symantec Decoy Server: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=157&EID=0

Decoy server is at this point of time only capable of presenting Solaris environments - but this neatly fits in place with the fact that all servers are running Solaris 8. Each cage includes a full Solaris installation, which mirrors one of the main servers in the service net and requires a dedicated NIC. A fifth NIC is used for to access the hosting environment itself from Management & Log Net II and for logging.

For the web cage the vendor recommends mirroring most of the public files of the web site and keeping it up-to-date. For the mail server cage we use the Content Generation Module (CGM) of Decoy Server to generate user accounts and mail addresses. The CGM can be fed by the company name of GIAC FC to create a bit more realistic content that might be enough for deception at first glance. As we need much horse power to run the host system plus four fully installed cages, we will use a Sun Fire V 250[28] (2*1.28 GHz UltraSparc III processors, 8 GB RAM, 5*73 GB hard disks - for each cage one, Solaris 8 all patches).

[28] Sun Fire V250 Server: http://www.sun.com/servers/entry/v250/index.xml

2. Network Diagram



3. Sensor Management

All sensors, IDS and deception systems are managed out-of-band by the use of dedicated NICs plugged into two pure management and log traffic networks. These networks, a more "external" Net I, combining devices protected by the PIX, and a more "internal" Net II for devices behind the SGS cluster, are connected to different interfaces on the third, the SESA-Net protecting SGS. This is to prevent someone having compromised a machine in the external service net from gaining further access to the more internally situated IDS systems.

4. Traffic flow, Alert and Log flow, Management

User Traffic

a) DNS

We use a split DNS approach here, strictly separating the DNS information available internally from externally accessible name-space.

Customers from the Internet access the external DNS in the Service Net – via NAT by the PIX cluster - to receive name to official IP resolution of just the externally available services DNS, Main Fortune Cookies, Web and Mail.

External Cookie Saying authors coming in via the VPN, which terminate at the SGS cluster, may only access the Internal Fortune Cookie Server by its IP address, so do not need any internal name resolution.

Internal users use the DNS proxies of the SGS firewalls to access external DNS servers on the Internet and receive – by the use of special zone files for which the SGS feels authoritative – the internal RFC 1918 IP addresses for name resolution of the hosts in the Service Net.

b) NTP

The clustered SGS firewalls use official timeservers on the Internet.[29]

All networks use the NTP proxies of the SGS Cluster as timeservers, even the hosts on the Management & Log Net II by using the appropriate NICs of the firewalls.

[29] NTP resource page: http://www.ntp.org; List of public NTP servers: http://www.eecis.udel.edu/~mills/ntp/servers.html

The hosts in the Service Net use their dedicated NIC to Management & Log Net I to access the NTP service on the third, internal, SGS, which itself is a client of to the SGS Cluster.

Hosts in the SESA Net use the internal interface of the third SGS as their NTP server.

c) Inbound HTTP

Users from the Internet access the company main Web site by accessing the official IP address, which is statically NAT-ed by the PIX firewalls onto the Web server in the Service network. Parts of the Web site are leading to the Cookie shop.

d) Customer Cookie Access

The Cookie server is a second web server, which accepts SSL connections for online Cookie purchases by customers from anywhere in the world. The PIX does the static NAT like in the case of the main Web site.

e) Cookie Text Delivery

Freelancing Cookie authors deliver and store their proto-Cookies on the Internal Cookie Server. This happens via dedicated VPN connections, which terminate on the SGS firewalls.

Recently the access was changed from pure SSH to `scponly`[30], which prevents users from gaining a real shell on the server they access the files on via SCP.

As this inbound traffic is encapsulated into ESP the external PIX firewalls are configured to let this VPN traffic including IKE pass to the SGS cluster.

f) Outbound HTTP & FTP

The protocol-specific proxies of the SGS firewalls handle outbound HTTP and FTP traffic.

g) Mail Traffic

Inbound mail arrives at the Mail server in the Service Net and is virus-scanned and spam-analyzed. Then it is forwarded through the SGS cluster to the internal Mail server. Outbound mail takes the same way in the reverses direction.

[30] SSH tool "scponly": http://www.sublimation.org/scponly/

Management & Log Traffic

The main focus of the network design was that security sensible traffic like device management and logging should be physically separated from user traffic. Even if a server is compromised and switches are ARP-flooded only parts of the management network should be accessible.

Therefore we use dedicated network interfaces for logging and management on every device. We separate the log and management networks, so that by compromising a server in the Service Net you cannot get access to or sniff the alarm traffic of the Networks IDS, which observe this Service network.

As a final line of defense we deploy a very strictly configured third SGS with additional interface filters in front of its proxies as a guardian of the internal "Sanctum", the SESA Net, where all logs and alarms are delivered for correlation and incident documentation.

5. Traffic Capturing

Capturing traffic for examination in crucial networks by is realized by configuring Switched Port Analyzer (SPAN) ports on the main Cisco switches in the network; only those important switches are depicted in the network diagram. The SPAN ports are also known as mirroring or monitoring ports, or in older MH terminology “copy ports”; they cause the switch to pass all or selected traffic to one switch port for traffic analysis and statistics or in our case to an IDS.

Normally these SPAN ports only send the captured traffic and do not receive anything, which in our case is just the right thing. Even if - by crafting special packets designed to exploit flaws in a special IDS – an IDS gets compromised, there would be no direct way back for return traffic through the SPAN port to let the attacker take remote control over the machine, e.g. using a backdoor shell or to spread an IDS-compromising worm like W32.Witty.Worm[31]. Care must be taken, though, that no routing back to the Internet is possible by using the IDS managing NIC. As in our network the managing subnets are strictly separated from the networks having routes to the Internet, we are safe here as long as no accidental network misconfiguration occurs.

We do not implement any kind of response tactics like crafting spoofed RST packets to try to break down malicious connections between an attacker and a victim, as we deem its effectiveness too insecure – and if you use this feature, which both Snort and MH would be capable of, in response to false positives you would shut down legitimate traffic on your network. Trouble-shooting connectivity problems and customer complains could be very challenging. If at some point of time such response tactics are to be implemented care must be taken of that the

[31] Symantec Security Response, W32.Witty.Worm: http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html

switches are capable of letting the spoofed traffic back into the SPAN ports, which also might cause problems with a switch's port security features. Configuration of a SPAN port for Cisco routers is given in Section 10 below.

6. Alert Collection, Analysis, Storage

All Symantec security devices and programs use the dedicated Management & Log Nets to forward their alerts and normal logs via the third SGS firewall into the SESA net; this happens either by a push or a pull from the SESA manager or the SESA console, respectively. These connections are encrypted by SSL.

The log traffic from the PIX firewalls, the routers, the Honeyd and all applications (Web, Mail, DNS, host logs) will be collected on the Syslog servers. On the Syslog in the Management & Log Net II a SESA event collector for Cisco PIX will transform the PIX messages in a SESA compatible format and send them to the Manager. SESA and Incident Manager will correlate these messages with all other SESA-enabled devices' log files.

The SESA device messages will be stored in the SESA Data Store, a database that has a massive RAID array attached.

The Master Syslog, which we will base on Apple G5 Xserve[32] (Dual 2 GHz G5on Mac OS X) as no special platform requirement is given here - the PIX event collector will run on the Linux Syslog server in Management and Log Net II -, combined with a big RAID array[33] (3.5 TB) to store the raw logs from all devices. The G5 Xserve is used as a pilot install, because it has a good power for cash ratio and excellent administrative tools. In future times adding some diversity from Linux/Windows on Intel and Solaris on Sparc might lessen the possible risks of monocultures with ready-made exploits available for everyone.

Depending on network traffic all log files should be kept for at least 30 days, though the actual length of storage time should be adjusted after some months of actions - the longer, the better for future forensic analysis of late-discovered break-ins. Additional daily backups and weekly rotation can be realized by eight mobile and affordable Kano XSPAND Firewire systems of type XPD-2X250 (500 GB).[34]

The firsts steps of event analysis will be automated after some initial preparations and ongoing fine-tuning. Incident Manager will reduce the massive amount on log messages and alerts into a lower amount of "incidents" which can be opened and evaluated by a human analyst.

[32] Apple G5 Xserve: http://www.apple.com/xserve/
[33] Apple Xserve RAID: http://www.apple.com/xserve/raid/
[34] Kano XSPAND: http://www.kanotechnologies.com/prod/x_spand.cfm

7. Incident Handling

The term "incident" will have two meanings at GIAC FC. First, the broad meaning of "An act of violating an explicit or implied security policy."[35] Second, the pure technical term for the abstract entity that SESA and Symantec "Incident" Manager generate by log collection and correlation. The latter is the main tool at GIAC FC tool to discover the former, but may result in false positives and negatives.

Handling of incidents (in general sense) will take place according to six phases:[36]

1) Preparation
2) Identification
3) Containment
4) Eradication
5) Recovery
6) Follow up

The main goal of deploying and constantly tuning the proposed IDS structure (part of Phase 1) is to be able to cope with Phase 2, the accurate identification of real incidents.

After having reduced false positives there is the problem of prioritizing messages from the systems. As a guideline and example, attacks from the outside that do not pass the PIX firewalls would be considered low priority. Alarm messages from Honeyd in the Service Net have very high priority, and even higher are those from interaction with the Decoy Cages, as normal traffic should not generate any activity on the Honey pots. The company's most valuable assets are the Fortune Cookie Sayings, therefore the host intrusion detection systems on the Cookie servers will reserve highest attention.

Thus defining a framework of priority-reaction pairs will result in an incident handling policy that can also be reflected back into the more technical "Incident" building process of Symantec Incident Handler. One result, for example, will be an evolution of finer granularity in alarming and countermeasure procedures. As usual, this will be an ongoing process.

8. Implementation of 24/7 monitoring

During the first months at least monitoring will be part of administrative work, which also includes fine-tuning the signatures and correlation procedures by Symantec Incident Manager. Because an experimental Snort device and a MH

[35] US Department of Homeland Security, Incident Definition: http://www.fedcirc.gov/incidentReporting/incidentDefinition.html
[36] Stephen Northcutt: Computer Security Incident Handling, SANS Press, 2003.

have been deployed before, some basic knowledge is expected to exist with the network administrators. Employing at least one very highly trained IDS-specialist and analyst who has experience in interpreting log files, signature tuning and incident response should be the next step to enhance this.

As the administrators at GIAC FC already work in two daytime shifts from 6 am - 4 pm and 3 pm - 23 pm, and there is an administrator on call for the night time, this might be sufficient for some time if signature are tuned well and nightly alerts via email are just issued for really high level incidents, which rarely show false positives, like access to the Decoy servers or scans versus Honeyd-simulated networks.

After the new infrastructure got going and depending on experience with daily incident analysis, a regular company-internal night-shift might be introduced, or the security-monitoring as a whole could be outsourced to an Monitored (and perhaps Managed) Security Service provider who offers 24/7 service.

Outsourcing should be planned carefully, taking into consideration which of the deployed devices the service provider is able to support, which set of signatures they are able to analyze and correlate, what kind of event classification system they have and what their alarming policies are, for example, on what do they alarm by phone and on what by mail, what is the maximum timeframe between event and alert? Especially this part of the contract should be studied intensively.

And not least: are they able to support custom signatures, which might be interesting or necessary for a complete overview of the network security landscape?

It might also be wise to outsource the monitoring of one central device first, like one of the MHs, to give the service a try. This might perhaps be enough for covering monitoring at night, the in-house administrators with the help of the deployed Symantec Incident Manager could handle the rest of the devices.

9. Encryption for securing Events

The SESA log and alert traffic is encrypted by SSL by default.

The Syslog traffic is clear text, so only protected by the physical separation of the log networks from the production networks, and to some extent, by a switched environment. To encrypt the Syslog messages from the routers and the PIX firewalls will turn out difficult; all Linux/Unix based devices could use SSL tunneling or real VPNs, though the amount of management necessary has prevented us from implementing such measure in this early phase of deployment. A solution might be envisaged by using firewall/VPN-equipped intelligent NICs.

After the logs and alerts have reached the SESA Net they are held relatively secure by another strict separation from all other networks. Only the administrators have access to this network.

The log files from the iForce MH appliances will have a cryptographic fingerprint to prove their authenticity. In the future a solution covering more logging devices might be engineered.

10. Configuring a Stealth Interface (Cisco Switch, Sun Solaris)

a) Configuration of a SPAN port on Cisco switches[37]

On Catalyst 3500XL switches like the 3508G XL[38] we have two ways of configuring SPAN ports. The first uses the **`port monitor`** command. Say we want to monitor switch port 0/1 (which belongs to VLAN 10) with SPAN port 0/3; we have to enter interface mode for 0/3, assign the interface to the same VLAN and state the “source port” (in switch terminology) to use:

```
cata01# configure terminal
cata01(config)# interface gigabitethernet 0/3
cata01(config-if)# switchport access vlan 10
cata01(config-if)# port monitor gigabitethernet 0/1
```

The second method uses the monitor session command issued in configuration mode:

```
cata01# configure terminal
cata01(config)# monitor session 1 source interface gigabitethernet 0/1 both
cata01(config)# monitor session 1 destination interface gigabitethernet 0/3
```

We create a monitoring session that captures all traffic on 0/1 ("**both**", not just received "**rx**" or transmitted "**tx**") and mirrors it to 0/3.

Please note that for Catalyst 4000 and 6500 running CatOS we have even more possibilities by using the **`set span`** and **`set rspan`** commands. The latter can create monitoring RSPAN VLANS so we can monitor the traffic on a different switch from the one on which the original traffic occurred. If we decide to extend the redundancy of routers and firewalls to the more important switches as indicated in the network diagram, this may come in handy, if there is no budget for doubling the NIDS structure also.

b) Configuration of a stealth interface on Solaris[39]

[37] Cf. Earl Carter: Cisco Secure Intrusion Detection System (CSIDS), Cisco Press, Indianapolis, 2004; pages 135-157.
[38] Cisco Catalyst 3508G XL Switch: http://www.cisco.com/en/US/products/hw/switches/ps637/ps639/index.html
[39] Sun Bigadmin - Implementing a stealth Ethernet interface: http://www.sun.com/bigadmin/content/submitted/stealth_ethernet.html

Basically this just requires a start-up script – to keep the configuration after a reboot - using `ifconfig` without an IP-address. This is a script for a SPARC machine using a gigabit interface. Save it for example under `/etc/rc3.d/S40stealth-int`:

```
#!/bin/sh
/usr/sbin/ifconfig ge0 up
```

Then adjust the file permissions by using:

```
chmod 744 /etc/rc3.d/S40stealth-int
```

Either execute the script by hand or reboot the machine. An interesting resource for troubleshooting gigabit interfaces under Solaris can be found at the Sun web site.[40]

11. Analysis of encrypted traffic (SSL, VPN)

Under normal circumstances encrypted traffic cannot be analyzed as a network IDS is normally no endpoint of the connection.

But we place SHIDS at every endpoint of encrypted traffic in our LAN: at the main Fortune Cookie Server which receives SSL connections from customers world wide, as does our company web server, and our Internal Fortune Cookie Server that is accessed by freelancers via VPN tunnels which terminate at the SGS cluster; as the traffic between SGS and Internal Cookie Server, i.e. in the Internal Cookie Net, is not encrypted, the responsible MH will have no difficulties identifying attacks conducted through this channel.

12. Additional Logs from Routers and Applications

In addition to the main SESA alarming, logging and correlation framework we deploy a second Syslog structure, where all device alerts are doubled in Syslog format and collected at the Main Syslog in the SESA net. These raw logs are not kept as long as the refined SESA events, but at least 14 days in compressed form before being log rotated. To achieve this the Main Syslog is connected to a hard disk storage array.

This second log structure gives us a backup in case of failures with the main SESA facilities, and a rich and deep log reservoir for human forensic analysis in the case of a security incident.

[40] Sun Product Documentation - "ge" driver: http://docs.sun.com/db/doc/806-7745-10/6jgjlqs7a?q=GigabitEthernet&a=view

13. Integration into an Enterprise Security Management

The Symantec SESA framework is aiming to provide a centralized logging and management solution for many different Symantec and non-Symantec products. It is relatively young, but seems to be heavily developed and extended. All future Symantec products will be able to talk to SESA directly without "bridges" or the like. Support for other vendors is also on the way, there is for example also an event collector for Checkpoint firewalls.

It is possible to integrate the whole corporate Antivirus-alerting structure into this framework, if we have already Symantec/Norton AV deployed, which is the case with GIAC FC.

An IDS is also a great feature to have for network troubleshooting, Snort and MH have many signatures and/or protocol models for detecting abnormal or malformed traffic caused by network misconfiguration. This is an extra value, which should not be underestimated.

The deployed security management structure can also be extended to remote offices or merged companies; special management and logging VPNs can extend the local dedicated management networks. MHs can be deployed in different clusters that correlate local events before sending them to the SESA manager.

The SESA and Incident Manager are capable of handling distributed locations and devices. If the number of SESA-enabled devices gets larger than a few hundred (maximum number given by vendor is 1000, cum grano salis), further local SESA managers can be installed in the remote locations, which can be accessed from the headquarters via the SESA console.

Practical Assignment Part Two:

II. Network Detects

Detect 1: "Get the Balance Right"

1. Source of the Trace

a) The trace:

```
[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/10-08:51:10.544488 193.144.127.9:80 -> 46.5.227.141:137
TCP TTL:46 TOS:0x0 ID:998 IpLen:20 DgmLen:40
***A**** Seq: 0x37  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/10-08:51:15.514488 193.144.127.9:80 -> 46.5.227.141:137
TCP TTL:46 TOS:0x0 ID:1329 IpLen:20 DgmLen:40
***A**** Seq: 0xA5  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/10-08:51:20.504488 195.77.24.2:80 -> 46.5.227.141:137
TCP TTL:46 TOS:0x0 ID:1621 IpLen:20 DgmLen:40
***A**** Seq: 0xFE  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]

[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/10-08:51:25.504488 195.77.24.2:80 -> 46.5.227.141:137
TCP TTL:46 TOS:0x0 ID:1931 IpLen:20 DgmLen:40
***A**** Seq: 0x161  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
```

Source IPs 193.144.127.9 and 195.77.24.2 are sending TCP-packets, which just have the ACK-bit set, from source port 80 to port 137. Here is the corresponding trace captured by tcpdump:

```
08:51:10.544488 193.144.127.9.80 > 46.5.227.141.137: . [bad tcp cksum f9f9!] 55:55(0) ack
0 win 1400 (ttl 46, id 998, len 40, bad cksum 3cc4!)
0x0000   4500 0028 03e6 0000 2e06 3cc4 c190 7f09        E..(......<.....
0x0010   2e05 e38d 0050 0089 0000 0037 0000 0000        .....P.....7....
0x0020   5010 0578 5d26 0000 0000 0000 0000             P..x]&........

08:51:15.514488 193.144.127.9.80 > 46.5.227.141.137: . [bad tcp cksum f9f9!] 110:110(0)
ack 1 win 1400 (ttl 46, id 1329, len 40, bad cksum 3b79!)
0x0000   4500 0028 0531 0000 2e06 3b79 c190 7f09        E..(.1....;y....
0x0010   2e05 e38d 0050 0089 0000 00a5 0000 0000        .....P..........
0x0020   5010 0578 5cb8 0000 0000 0000 0000             P..x\.........

08:51:20.504488 195.77.24.2.80 > 46.5.227.141.137: . [bad tcp cksum f9f9!] 254:254(0) ack
0 win 1400 (ttl 46, id 1621, len 40, bad cksum 9f9f!)
0x0000   4500 0028 0655 0000 2e06 9f9f c34d 1802        E..(.U.......M..
0x0010   2e05 e38d 0050 0089 0000 00fe 0000 0000        .....P..........
0x0020   5010 0578 c1a9 0000 0000 0000 0000             P..x..........

08:51:25.504488 195.77.24.2.80 > 46.5.227.141.137: . [bad tcp cksum f9f9!] 99:99(0) ack 1
win 1400 (ttl 46, id 1931, len 40, bad cksum 9e69!)
0x0000   4500 0028 078b 0000 2e06 9e69 c34d 1802        E..(.......i.M..
0x0010   2e05 e38d 0050 0089 0000 0161 0000 0000        .....P.....a....
0x0020   5010 0578 c146 0000 0000 0000 0000             P..x.F........
```

b) The source of the trace:

The logs of this detect have been taken from the "raw" logs section of www.incidents.org.[41] A first look with tcpdump gives us information about the network the logs have been taken from:

```
tcpdump -vvenr 2002.5.10 | more
```

-vv : use just a medium level of verbosity; feel free to experiment for deeper inspection.
-e : show the ethernet (MAC) addresses.
-n : do not resolve IP addresses to DNS names.
-r : traffic not taken from a network interface card (NIC), but from a given file.

The traffic thus inspected hints at a "home net" being part of 46.5.0.0/16 (IANA reserved IP addresses, therefore probably obfuscated as stated in the README to the log files), as this network is source or destination of every packet we see. Given this, we can look at the Ethernet layer:

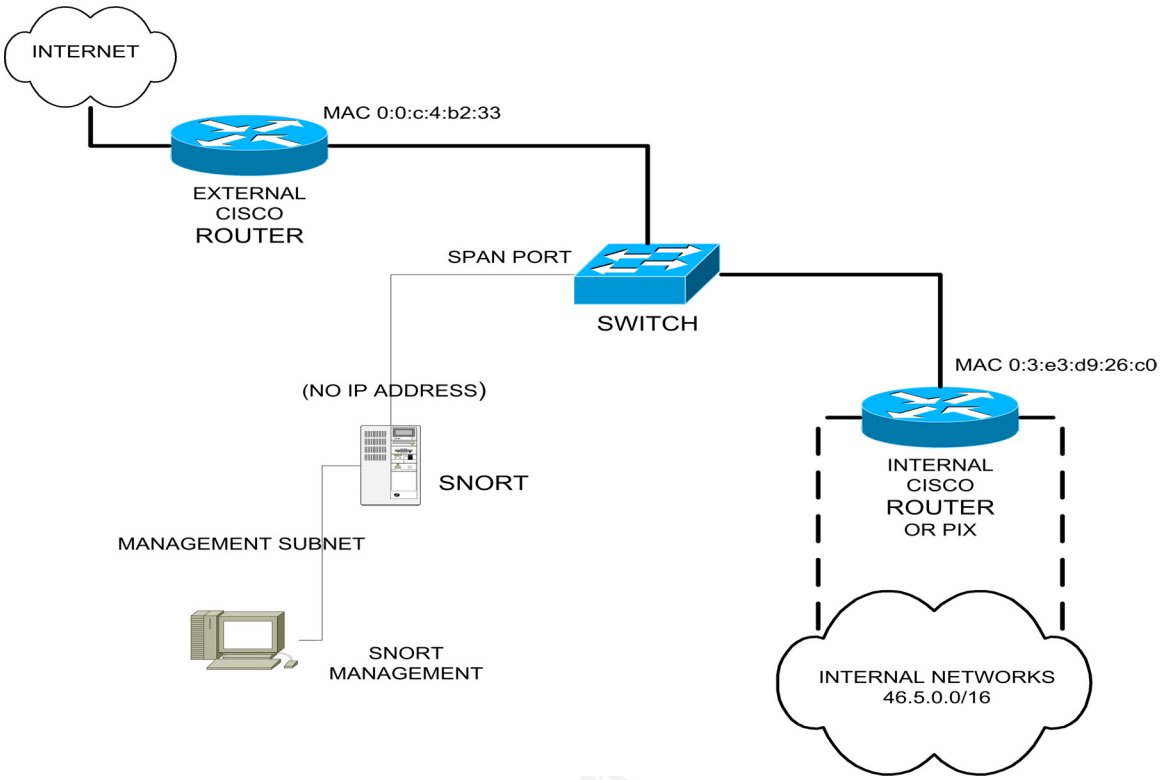
MAC external device, NIC looking inside: `0:0:c:4:b2:33`
MAC internal device, NIC looking outside: `0:3:e3:d9:26:c0`

The first three bytes identify the vendor of the NIC (according to RFC 1700, obsolete after RFC 3232, which just states that RFC 1700 has been replaced by an online database[42]). Especially for MAC addresses we refer to the IANA homepage.[43] There is a nice searchable version that is useful for reference.[44] We use it to look up the vendors by the MAC-address prefixes:

```
00000C          Cisco Systems, Inc
0003E3          Cisco Systems, Inc
```

We can conclude that the IDS sensor is somehow placed between two Cisco devices, probably routers or a PIX firewall. The network might look somehow like the following pictorial attempt:

[41] Raw Logs on Incidents.org: http://www.incidents.org/logs/raw/2002.5.10
[42] IANA Protocol Numbers and Assignment Services: http://www.iana.org/numbers.html
[43] IANA Ethernet Numbers: http://www.iana.org/assignments/ethernet-numbers
[44] Vendor/Ethernet MAC Address Lookup and Search: http://www.coffer.com/mac_find/



We look deeper for interesting traffic by using Snort as a display tool:

```
snort -c /etc/snort/giac.conf -k none -h 46.5.0.0/16 -l T2002.5.10 -r 2002.5.10
```

-c : use the given configuration file; note that my `giac.conf` uses all available standard rules as included with version 2.1.1 (Build 24).
-k none : ignore checksum errors (which resulted from obfuscating the IP addresses); as the TCP/UDP checksums use a pseudo-header including the IP addresses, just ignoring the IP checksum is not enough to prevent Snort from ignoring these modified packets).
-h : state the point-of-view for snort, this is our guessed home net, deducted from first look above.
-l : states the directory where Snort saves the output.
-r : as in tcpdump, states Snort should read his input in binary format from a file.

With the help of Snortalog[45] we try to identify the top alarms (option `-attack`) from the alert file generated by snort during the previous step - this is located in the directory T2002.5.10: `snortalog.pl -attack -file alert` . Skimming through the messages we have the unusual freedom of choice where to look closer. As traffic concerning port 137/tcp (!) looks very interesting, this is what we will focus on.

[45] SnortAlog Homepage: http://jeremy.chartier.free.fr/snortalog/

2. Detect was generated by …

The detect above was generated by Snort 2.1.2 with a full rule set applied to the binary capture file - which was itself created by an older Snort of unknown exact version and rule set. It is probable that the original alert was generated by the same rule as mine, perhaps having an older version as 3:

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

This rule looks for inbound TCP-packets (even packets not part of regular and already initiated sessions, as "stateless" implies) with just the ACK-bit (possibly together with the reserved or ECN bits) set and an acknowledgement number of 0, which would only be possible after cycling the whole range and even then highly improbable. So an ACK-scan like with `nmap -sA` would be a likely diagnosis, and this is what the alert messages state. Note, however, that the Nmap 3.5 man page[46] states that the acknowledgement numbers used in this kind of scan would be random.

Later I found a reference and lab test regarding this issue in Antony Gummery's GIAC practical posting[47], where he gave a link to proof that earlier Nmap versions indeed used 0 as acknowledgement number[48] - a fact which seems to be the motivation to write this Snort rule.

The second set of traces was generated by applying tcpdump to the downloaded file. By variations of filters (`tcpdump -nvvXr 2002.5.10 host 46.5.227.141 / host 195.77.24.2`, respectively: `host 193.144.127.9`) we made sure that the only captured packets from these two sources were directed at the destination IP 46.5.227.141.

3. Probability the source address was spoofed

We see no signs of an attempted Denial-of -Service here, because we have only a few packets from two source IP addresses, nor does the detect look like a one-packet exploit, as this would most likely require a completed three-way-handshake which most probably did not occur.

It would be reasonable that the source wants something in return – either a RESET or an ICMP unreachable message – or by a lack thereof be able to draw conclusions about filtering devices in the path. So spoofing seems not probable here.

[46] Nmap network security scanner man page: http://www.insecure.org/nmap/data/nmap_manpage.html
[47] Firewall incidents at the FinchHaven datacenter: http://www.finchhaven.com/pages/incidents/031202_tcp_123.html
[48] Fyodor on Snort-users mailing list, Aug 11, 2000: http://archives.neohapsis.com/archives/snort/2000-08/0152.html

4. Description of the attack

The source IPs above are sending TCP-packets with only the ACK-bit set from source port 80 to port 137 on IP 46.5.227.141 which resides in our "home" net. The acknowledgement number is 0, which is highly improbable (if not impossible) in normal traffic. This could only occur if during long and intensive data transfers the sequence numbers had to be recycled, and even then the last packet before this would have to have included as last byte that with the maximum number of 2^{32}.

To describe what is really going on we have to look at the larger picture of observed traffic from, see Section 9, Correlations below. The whole picture presents itself as follows. Both IP addresses seem to belong to traffic-shaping or load-balancing devices, which try to determine an optimal route to our "home" net. This is with all probability a response to some client on our side connecting to a service in the load balancers' realm, not a stimulus. All the correlation detects below indicate that this is a normal and mostly harmless procedure, which generates an awful lot of strange packets and general network noise.

5. Attack mechanism

What would be the reason to send such packets as seen in the traces above? The `nmap` man page[49] states:

"ACK scan: This advanced method is usually used to map out firewall rulesets. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets.
This scan type sends an ACK packet (with random looking acknowledgment/sequence numbers) to the ports specified. If a RST comes back, the ports is classified as "unfiltered". If nothing comes back (or if an ICMP unreachable is returned), the port is classified as "filtered". Note that nmap usually doesn't print "unfiltered" ports, so getting no ports shown in the output is usually a sign that all the probes got through (and returned RSTs). This scan will obviously never show ports in the "open" state."

So the normal purpose for an ACK-scan would be to probe firewalls - reconnaissance versus the network defenses. Or it might try to pass through to find a host alive - first reconnaissance versus a single host.

But if a reset returns, it cannot be determined if the targeted port was open or closed, in both cases an unsolicited ACK should return a RESET (RFC 793, headline "Reset Generation", page 36):[50]

"1. If the connection does not exist (CLOSED) then a reset is sent in response to any incoming segment except another reset.
(...)

[49] Nmap network security scanner man page: http://www.insecure.org/nmap/data/nmap_manpage.html
[50] RFC 793, Transmission Control Protocol, Sep 1981: ftp://ftp.rfc-editor.org/in-notes/rfc793.txt

2. If the connection is in any non-synchronized state (LISTEN, SYN-SENT, SYN-RECEIVED), and the incoming segment acknowledges something not yet sent (the segment carries an unacceptable ACK), or if an incoming segment has a security level or compartment which does not exactly match the level and compartment requested for the connection, a reset is sent."

Normally, some more information about the target host may be gathered by passive fingerprinting the returning packet, e.g. by the use of `p0f`.[51] But in this case `p0f` was not able to come to a conclusion due to the contrary information in the packet.

If we have a look at the huge history of strange traffic originating from these IPs in: 6. Correlations a), b), c), this is still ongoing activity even after two years. Especially a) is ringing a bell: Load balancer traffic.

In our case the source seems to be more interested in the returning packet itself than in reconnaissance of target networks or hosts. It carries a TTL that helps to calculate the distance in hop counts between each load balancer and the client network. The roundtrip time between sending the ACK and receiving the RESET also can be used to optimize the choice which server should respond.

Interesting is the destination port of 137/tcp, which we do not see in younger examples of traffic from these sources - but this could be due to limited visibility and not enough samples. Is this just a choice of a port assumed closed on almost every existing machine (as Windows Netbios name service uses 137/udp)?

Or was this port chosen by design - perhaps assuming that port 137/(any protocol) at that time was open in some very simple packet filtering devices instead of narrowing to 137/udp? It is hard to say - but with all caveats this port seems to be replaced by 123/tcp nowadays, perhaps in the same hope of slipping through loose filters that were build to allow NTP (123/udp) in.

Just two examples of load balancing products are:

i) 3-DNS[52]
ii) LinkProof[53] - there is also white paper giving an overview of its operation.[54]

An example of just one kind of strange traffic from this kind of devices is given in Section 6, Correlations, d).

We use `whois` to examine the source IPs:

```
whois -h whois.ripe.net 195.77.24.2 ...

inetnum:      195.77.24.0 - 195.77.24.255
netname:      GVANET
```

[51] p0f Homepage: http://lcamtuf.coredump.cx/p0f.shtml
[52] F5 3-DNS Controller: http://www.f5.com/f5products/3dns/
[53] Radware LinkProof: http://www.radware.com/content/products/lp/default.asp
[54] LinkProof - White Paper: http://www.radware.com/content/products/lp/whtpaper/default.asp?_v=about&document=1316

```
descr:          Generalitat Valenciana
descr:          Internet access for Valencia State (NCC#1998103531)
country:        ES
…

whois -h whois.ripe.net 193.144.127.9 ...

inetnum:        193.144.104.0 - 193.144.127.255
netname:        GVA
descr:          Red GVA De La Generalitat Valenciana
descr:          Valencia
country:        ES
…
```

Both addresses are registered with an official sounding “Generalitat Valenciana” in Spain. This Generalitat has a very flashy and fancy web portal that I found by using Google: www.gva.es

And what do we find if we do a reverse lookup on this hostname?

```
www.gva.es      canonical name = aesgard.gva.es.
Name:   aesgard.gva.es
Address: 193.144.127.85
Name:   aesgard.gva.es
Address: 195.77.24.70
```

Strike. This name has two possible IP addresses - one in each of the same subnets that are seen as source in our traces: 193.144.127.0/24 and 195.77.24.0/24. And now we know how much noise they generate just to spare us some milliseconds waiting time, see 6. Correlations e).

6. Correlations

a) First we look at the DShield database.[55] At the time of writing IP 195.77.24.2 appears 1,512 (!) times in the database, whereas IP 193.144.127.9 does not appear a single time. Some of the TCP-ports IP 195.77.24.2 seems to have “scanned” for are: 53, 80, 123, 1915, 4671, 4672, 36296, 54367. Many of these destination ports have also been used as source ports; in addition we see for example 20 and 3128. Concerning the flags there are plain SYN, lonely ACK and sometimes no flags. Some records might indicate ICMP echo requests (source “port” 8) that seem not have been normalized correctly.

A direct correlation for destination port 137/tcp is not listed, but lots of packets have been sent with source port 80. One similarity occurs: destination 123/tcp; normal usage of this port only occurs with NTP (123/udp), though some databases list Trojans using this TCP port.[56] As most of the other scans are not Trojan-related, I would dismiss this possibility in the whole context. All these patterns indicate probing for normally open and normally closed ports with source ports and flags used to bypass simple packet filtering devices.

[55] DShield Homepage: http://www.dshield.org/
[56] Firewall incidents at the FinchHaven datacenter: http://www.finchhaven.com/pages/incidents/031202_tcp_123.html

b) A correlation for IP 193.144.127.9 is the following, which shows more common signs of load-balancer traffic, note that the IP is still active in 2004.[57] No explanation is given there.

```
[**] SCAN nmap TCP [**]
03/12-04:09:22.980000 0:6:53:3:7E:20 -> 0:10:DB:8:9C:C1 type:0x800
len:0x3C
193.144.127.9:80 -> xxx.xxx.xxx.33:53 TCP TTL:40 TOS:0x0 ID:23818
IpLen:20 DgmLen:40
***A**** Seq: 0x2A4  Ack: 0x0  Win: 0x578  TcpLen: 20
```

c) Antony Gummery investigated parallel traffic while preparing his GIAC practical. His detects come from raw logs 4 months later.[58] They have the same source port 80/tcp and destination port 137/tcp.

He is doing extensive and very well arguing analysis on these traces with similar results but a little different conclusion about the severity.[59]

```
[**] SCAN nmap TCP [**]
10/17-17:25:45.266507 193.144.127.9:80 -> 32.245.136.215:137 TCP TTL:44
TOS:0x0 ID:54253 IpLen:20 DgmLen:40
***A**** Seq: 0x282  Ack: 0x0  Win: 0x578  TcpLen: 20

17:25:45.266507 193.144.127.9.http > 32.245.136.215.netbios-ns: .
[bad tcp cksum 1815!] 642:642(0) ack 0 win 1400
(ttl 44, id 54253, len 40, bad cksum bb64!)
0x0000   4500 0028 d3ed 0000 2c06 bb64 c190 7f09        E..(....,..d....
0x0010   20f5 88d7 0050 0089 0000 0282 0000 0000        .....P..........
0x0020   5010 0578 a783 0000 0000 0000 0000              P..x.........
```

d) An example of LinkProof traffic by Chris Brenton.[60]

e) The final test. I just visited www.gva.es[61] from IP MY.COMPANY.NET.99 - and what do our Snort and Raptor firewall report (in an abstract, normalized log-format)?

```
195.77.24.2     80      MY.COMPANY.NET.51      53      TCP
193.144.127.9   80      MY.COMPANY.NET.51      53      TCP
193.144.127.9   53      MY.COMPANY.NET.51      53      TCP
193.144.127.9   -       MY.COMPANY.NET.51      -       (pingd)
```

For the record, MY.COMPANY.NET.51 is not a DNS server.

7. Evidence of active targeting

Yes, these packets seem to be actively targeted, there are no horizontal scan patterns. The destination server might be a DNS server. But what we see is no stimulus – it is most probably a reply to not recorded outgoing connections

[57] Pedro Bueno on Intrusions list, Apr 8, 2002: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00110.html
[58] Raw Logs on Incidents.org: http://www.incidents.org/logs/RAW/2002.9.17
[59] Antony Gummery on Intrusions list, Mar 22, 2003: http://www.dshield.org/pipermail/intrusions/2003-March/007239.php
[60] Chris Brenton on Intrusions list, Apr 26, 2002: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00318.html
[61] Generalitat Valenciana portal site: http://www.gva.es

attempts to web servers. A group of load-balancing devices tries to deliver an optimized path for the request.

8. Severity

We estimate the severity of the attack according to the formula recommended by SANS:

Severity = *(Criticality + Lethality) - (System Countermeasures + Network Countermeasures)*

Each value is ranked on a scale from 1 (lowest) to 5 (highest).

a) **Criticality**: What is the function of 46.5.227.141? The above trace is the only occurrence of that IP. Most load balancer traffic seems to target DNS servers, but that would be arguing a posteriori. Thus spoke sagacious Salomon: **3**.

b) **Lethality**: The probable function of the observed traffic is route optimization, so no much danger here. But the methods are indeed scans that gather information about the home network. Who knows what happens with that information? So I would give it Lethality: **2**.

c) **System Countermeasures**: Frankly speaking – we do not know. We do not know if the fact that we do not see return packets is due to the fact that there were none, even if a possible positive response from 137/tcp would be somehow remarkable. So we have to be neutral here: **3**.

d) **Network Countermeasures**: We do not know much. But they have a network IDS in place which indicates a sharpened sense for security. Let's rate it with **4**.

Severity = (3+2) - (3+4) = **-2**

9. Defense recommendation

The port 137 (UDP and TCP) should definitely be blocked from entering any network, perhaps already at the border router if performance allows.

Stateful inspection should be installed which keeps track of inbound and outbound sessions, so a lonely unsolicited ACK-scan would be blocked. The same applies to all not internally presented services.

If possible, a general "Deny all" strategy should be implemented, with just the necessary openings. Inbound ICMP messages should be handled restrictive but with care, as some error messages (e.g. fragmentation related) are necessary for normal traffic.

The deployment of an IDS is a very good move, though we recommend in general fine-tuning of signatures to reduce the number of false positives. In our case the signature was triggering correctly, just the motivation for the observed scan is non-aggressive - but how should Snort know? We cannot exclude all possible load-balancers from investigation.

10. Multiple Choice Question

Given the trace below, which answer is true?

```
[**] [1:628:3] SCAN nmap TCP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/10-08:51:10.544488 193.144.127.9:80 -> 46.5.227.141:137
TCP TTL:46 TOS:0x0 ID:998 IpLen:20 DgmLen:40
***A**** Seq: 0x37  Ack: 0x0  Win: 0x578  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS28]
```

(a) This is a false positive; we see return traffic from a web server.

(b) This is part of a normal Netbios name resolution process between a Windows XP client and a WINS server.

(c) An acknowledge-number field of zero after Three-way-handshake is a security feature of Open BSD 3.4 or later.

(d) Such and similar packets are indicating the presence of load-balancers.

Correct answer: (d)

Public Posting Note:

A draft of this detect was posted as required to the Intrusions mailing List (Sat Jul 17 21:38:59 UTC 2004):

http://lists.sans.org/pipermail/intrusions/2004-July/008177.html

No follow-ups or questions did arise.

Detect 2: "Crafter's Call"

1. Source of the Trace

a) The trace:

i) Snort alert messages:

```
[**] [1:504:4] MISC source port 53 to <1024 [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
04/24-01:30:21.352051 219.238.233.157:53 -> MY.SE.RV.ER:139
TCP TTL:40 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A  Ack: 0x0  Win: 0x80  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS07]

[**] [1:504:4] MISC source port 53 to <1024 [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
04/24-02:07:12.764419 219.238.233.157:53 -> MY.SE.RV.ER:139
TCP TTL:40 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A  Ack: 0x0  Win: 0x80  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS07]

[**] [1:504:4] MISC source port 53 to <1024 [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
04/24-02:55:25.823312 219.238.233.158:53 -> MY.SE.RV.ER:139
TCP TTL:38 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A  Ack: 0x0  Win: 0x80  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS07]
```

ii) Details given by applying tcpdump to the capture file (tcpdump -nvvXr snort.log):

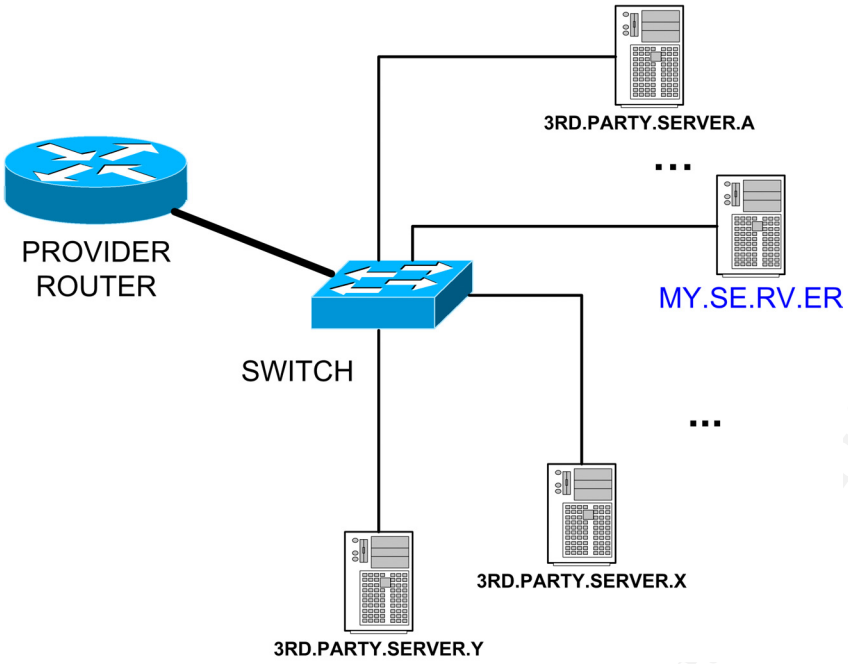
```
01:30:21.352051 219.238.233.157.53 > MY.SE.RV.ER.139: S [tcp sum ok] 666:666(0) win 128
(ttl 40, id 666, len 40)
0x0000   4500 0028 029a 0000 2806 15fb dbee e99d        E..(....(.......
0x0010   XXXX XXXX 0035 008b 0000 029a 0000 0000        XXXX.5..........
0x0020   5002 0080 31cd 0000 3100 0000 0000             P...1...1.....
```

02:07:12.764419 219.238.233.157.53 > MY.SE.RV.ER.139: S [tcp sum ok] 666:666(0) win 128 (ttl 40, id 666, len 40)

```
0x0000   4500 0028 029a 0000 2806 15fb dbee e99d        E..(....(.......
0x0010   XXXX XXXX 0035 008b 0000 029a 0000 0000        XXXX.5..........
0x0020   5002 0080 31cd 0000 3100 0000 0000             P...1...1.....

02:55:25.823312 219.238.233.158.53 > MY.SE.RV.ER.139: S [tcp sum ok] 666:666(0) win 128
(ttl 38, id 666, len 40)
0x0000   4500 0028 029a 0000 2606 17fa dbee e99e        E..(....&.......
0x0010   XXXX XXXX 0035 008b 0000 029a 0000 0000        XXXX.5..........
0x0020   5002 0080 31cc 0000 3100 0000 0000             P...1...1.....
```

b) The source of the trace:

This trace was captured at a lone-standing Debian server hosted at a service provider. This machine runs an only temporarily available Apache web server - and serves as an experimental outpost for the rest of the time. The targeted port 139/tcp on my Linux machine was not in use by any process at the time of the capture, in addition there was an iptables rule in place to silently drop these packets. The topology looks like this:



2. Detect was generated by …

The local guardian on my server who captured the original packets is a Snort 2.1 (Version 2.1.2) with the standard rule set and set up for logging into an ACID database. The Snort rule that triggers on these packets is:

```
alert tcp $EXTERNAL_NET 53 -> $HOME_NET :1023 (msg:"MISC source port 53 to <1024";
flags:S,12; stateless; reference:arachnids,07; classtype:bad-unknown; sid:504; rev:4;)
```

This rule looks for inbound TCP-packets with a source port of 53 and a destination port in the non-ephemeral range below 1024, which have the SYN-bit set with the possible addition of the reserved or ECN-bits.

The traces above 1)a)ii) have been generated by `tcpdump` reading the binary trace file which accompanies Snort's alert file.

3. Probability the source address was spoofed

Given the non-ephemeral source port of 53 and the highly suspicious IP identification number of 666, which is - for an extra twist - the same as the TCP sequence number, and given the fact that all three of these dimensions seem to be constant in time and in "source space" you can almost hear the message: "I am crafted!", therefore the question if the source IP is spoofed becomes absolutely valid.

The probability of spoofing is directly related to the assumed function of the captured packets. As the protocol is TCP, which demands for its Three-way-

handshake the successful arrival of a return packet (SYN-ACK) to the sender, spoofing is less likely than with UDP. These few packets do have no context in my log files which could indicate any kind of Denial-of-service attempt, be it distributed (from multiple sources) or not. Therefore it is improbable that both of the source IP addresses are spoofed. It is, however, possible that only one of these is truly genuine and the other is spoofed - to add a little decoy (like using `nmap -D` for port scanning). If the function of these packets is reconnaissance or the preparation for an exploit then only one true source would be sufficient.

4. Description of attack

The first and main indicator of the function of these packets is scanning the destination port of 139/tcp. This is the Netbios session port used for Windows file and printer sharing in non-pure Windows 2000 (or later) networks. The vulnerabilities concerning this port are numerous, for an overview search for that port at CERT.[62]

No service was listening at that port on our server, so it is not possible to determine if the attacker would have tried an exploit once an open port had been discovered. We see only SYN packets and no session data; a later deployed `netcat` did not capture anything similar, alas.

Interesting to note is that the repetition of packets is not due to the usual TCP retransmission as the time interval between the packets from the same source is around 37 minutes, followed by a packet from a neighboring IP address 48 minutes later.

5. Attack mechanism

First let us focus our attention at the source hosts that sent the packets. A `nslookup` did not resolve the IPs successfully to a DNS name. Next we try a `whois`:

```
whois -h whois.apnic.net 219.238.233.157 ...
% [whois.apnic.net node-1]
% Whois data copyright terms    http://www.apnic.net/db/dbcopyright.html

inetnum:        219.238.0.0 - 219.239.255.255
netname:        DXTNET
country:        CN
descr:          Beijing Dian-Xin-Tong Networking Technologies Co., Ltd.
admin-c:        PP40-AP
tech-c:         PP40-AP
status:         ALLOCATED PORTABLE
mnt-by:         MAINT-CNNIC-AP
mnt-lower:      MAINT-CNNIC-AP
changed:        hm-changed@apnic.net 20030305
source:         APNIC
```

[62] CERT search page: http://search.cert.org/

We apply `p0f`[63] to the dump file (`p0f -s snort.log`) to passively fingerprint the source, but we get no exact operating system match:

```
219.238.233.157:53 - UNKNOWN [128:40:0:40:.:.:?:?]
  -> MY.SE.RV.ER:139 (link: unspecified)
219.238.233.158:53 - UNKNOWN [128:38:0:40:.:.:?:?]
  -> MY.SE.RV.ER:139 (link: unspecified)
```

The syntax of the result gives in condensed form what we already see from the traces above. The values in brackets are described within the signature file:

```
# wwww:ttt:D:ss:OOO...:QQ:OS:Details
#
# wwww     - window size (can be * or %nnn or Sxx or Txx)
...
# ttt      - initial TTL
# D        - don't fragment bit (0 - not set, 1 - set)
# ss       - overall SYN packet size (* has a special meaning)
# OOO      - option value and order specification (see below)
# QQ       - quirks list (see below)
# OS       - OS genre (Linux, Solaris, Windows)
# details  - OS description (2.0.27 on x86, etc)
```

While having a look at the "classical" source of passive fingerprints[64] one problem for p0f seems to be the very low Window Size of 128 bytes. This is not to be found within the p0f fingerprinting entries either. Given the high amount of packet crafting already observed this fits into place.

A `traceroute` some days after the packet's arrival gave the same result for both sources, timing out after 21 hops:

```
21  210.82.191.118 (210.82.191.118)  462.066 ms  456.808 ms  451.109 ms
22  * * *
```

In correspondence with the observed arriving TTL of 40 or 38 respectively it is probable that the initial TTL was 64. This is supported by some of the correlations below, which indicate that the TTL seems not to have been crafted.

So the source host could run a BSD or Linux - the latter is observed in correlation 6) f). This observation and the occurrence of another destination port as 139/tcp makes it highly improbable that these scan were generated by a propagation attempt of a Windows-based worm.

The mechanism seems to be scanning a target by crafting a packet that might pass through simple packet filters - like the not so sophisticated stateless access-list filters on Cisco routers, which have not been replaced by their (more or less) stateful counterparts. In stateless filtering packets with source port 53/tcp must be allowed if TCP responses after truncated UDP using DNS answers are to be let through.

A really strange effect of crafting or an artifact of the network driver seems to be the non-zero byte after the announced packet length.

[63] p0f homepage: http://lcamtuf.coredump.cx/p0f.shtml
[64] Project Honeynet Fingerprints: http://project.honeynet.org/papers/finger/traces.txt

6. Correlations

a) A search at Dshield[65] gave no result for 219.238.233.157 or 219.238.233.158. But searching at Mynetwatchman[66] had some success:

For IP 219.238.233.157:[67]

Date/Time (UTC)	Agent Alias	Agent Type	Log Type	Target IP	# of IPs Targeted	Protocol/ Port	Port/ Issue Description	Source Port	Event Count
25 Apr 2004 20:15:10	jaack	win32	Zone Alarm	202.160.x.x	1	6/6112	dtspcd CDE Buffer Overflow	40483	1

For IP 219.238.233.158:[68]

Date/Time (UTC)	Agent Alias	Agent Type	Log Type	Target IP	# of IPs Targeted	Protocol/ Port	Port/ Issue Description	Source Port	Event Count
27 Apr 2004 17:15:44	GPW	Perl	Cisco Rtr	216.128.x.x	44	6/139	NETBIOS Session Service NETBIOS Session Service	1215	44
20 Apr 2004 09:52:58	kacos	win32	Zone Alarm	212.251.x.x	1	6/1524	ingreslock Common Solaris Backdoor	26785	1
17 Apr 2004 11:34:16	sburina	Perl	iptables	213.240.x.x	22	6/443	HTTPS - HTTP over TLS/SSL HTTPS - HTTP over TLS/SSL	32295	33
16 Apr 2004 10:42:10	Mander	win32	Zone Alarm	213.203.x.x	1	6/443	HTTPS - HTTP over TLS/SSL HTTPS - HTTP over TLS/SSL	11270	1

We can conclude that both IPs have been used for scanning activity before, at least one of them (219.238.233.158) in scans for 139/tcp. The reported time frame matches. But, alas, the recorded source ports are ordinary high ports, not 53/tcp.

b) The database of offending IP-addresses at my company gives the following:

```
219.238.233.157
Horizontal Scan for Ingreslock (1524/tcp)
Horizontal Scan for CDE Subprocess Control (6112/tcp)
```

[65] DShield homepage: http://www.dshield.org
[66] myNetWatchman homepage: http://www.mynetwatchman.com
[67] myNetWatchman incident - archived now (was: http://www.mynetwatchman.com/LID.asp?IID=89601318)
[68] myNetWatchman incident - archived now (was: http://www.mynetwatchman.com/LID.asp?IID=87377146)

```
219.238.233.158
Horizontal scan for SSL (443/tcp)
Horizontal Scan for Netbios (139/tcp)
```

These scans appeared at some of our customers' defenses in roughly the same week as the two observed at my server and at Mynetwatchman.

c) Querying Google for some packet details did result in some interesting parallels, but no exact match. The number **666** is characteristic for some tools that are mentioned here for reference, surely not a complete list. None of these seem to apply in our case, but the findings indicate this number is a sure sign of crafting.

i) The infamous Stacheldraht DDOS tool uses ICMP packets with id 666.[69]

ii) There is a tool for sending ICMP messages called icmpnum[70] that sends packets with a characteristic IP id of 666, but a TTL of 255.

iii) Another tool using default IP id 666 is S.I.N.N. ("Sinn Is Not Naphta")[71] which was written to test the kind of DOS by TCP-stack resource starvation as presented by Naphta.[72]

iv) Twoface is a DDOS tool that uses TCP sequence and acknowledge number of 666.[73]

v) SucKIT[74] uses a fixed TCP sequence number of 666 in its source code.

vi) But sometimes there are issues with network byte order - see synscan[75] that has a fixed IP ID of 39426[76] (hton(666) -> 39426) - as Donald Smith explains:[77]

">hton(666) -> 39426. NOW explain what that means:-)
This is a very important networking issue that you should understand.
This is what you get when you compile it in a little endian machine. The ID
39426 should have been 666 but the authour of the source code probably
forgot to change the IP ID variable from host to network byte order.
The decimal value 666 is equal to 029A in hex but when you read 029A in a
little endian machine, it would be read as 9A02 which gives us the value of 39426."

d) Similar source port, destination port and IP identification and TCP sequence numbers both 666 (666_d = 0x29A) which looks like a direct correlation:[78]

---

[69] David Dittrich, Stacheldraht Analysis: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis
[70] Sys-Security Group, Identifying ICMP Hackery Tools: http://www.sys-security.com/archive/securityfocus/icmptools.html
[71] S.I.N.N. - Sinn Is Not Naphta: http://archives.neohapsis.com/archives/vuln-dev/2000-q4/0663.html
[72] Naphta Dos Vulnerability: http://www.bindview.com/Support/RAZOR/Advisories/2000/adv_NAPTHA.cfm
[73] Twoface DDOS: http://www.darkaxis.com/devel/c/security/exploit/ddos/twoface.c
[74] Phrack, Linux on-the-fly kernel patching without LKM: http://www.phrack.org/show.php?p=58&a=7
[75] Synscan homepage: http://synscan.sourceforge.net/
[76] Arachnids on Synscan: http://www.digitaltrust.it/arachnids/IDS441/event.html
[77] Donald Smith on Intrusions, Nov 4 2002: http://www.dshield.org/pipermail/intrusions/2002-November/005813.php
[78] "<krist>" posting on Whitehats forum, Sep 21 2003: http://whitehats.com/cgi/forum/messages.cgi?bbs=get_topic&f=9&t=000075

```
[**] [1:504:2] MISC source port 53 to <1024 [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
09/17-19:00:10.289008 157.22.219.3:53 -> xx.xxx.xx.xx:139
TCP TTL:52 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A Ack: 0x0 Win: 0x80 TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS07]

19:00:10.289008 157.22.219.3.53 > xx.xxx.xx.xx.139: S [tcp sum ok]
666:666(0) win 128 (ttl 52, id 666, len 40)
0x0000 4500 0028 029a 0000 3406 ac3b 9d16 db03 E..(....4..;....
0x0010 5087 0f5a 0035 008b 0000 029a 0000 0000 P..Z.5..........
0x0020 5002 0080 d40d 0000 P.......
```

e) Another similar detect has been described in the GIAC practical of Kurt Anderson:[79]

```
[**] MISC source port 53 to <1024 [**]
12/10-00:31:13.297175 0:E0:D0:13:4D:16 -> 0:C0:DF:E0:33:1A type:0x800 len:0x3C
193.135.0.83:53 -> 10.0.0.2:139 TCP TTL:48 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A  Ack: 0x0  Win: 0x80  TcpLen: 20
```

f) A very similar packet from another source and destination port 111/tcp detected weeks later at my server:

```
[**] [1:504:4] MISC source port 53 to <1024 [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
05/17-01:29:57.652264 200.63.196.130:53 -> MY.SE.RV.ER:111
TCP TTL:49 TOS:0x0 ID:666 IpLen:20 DgmLen:40
******S* Seq: 0x29A  Ack: 0x0  Win: 0x80  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS07]
```

This seems to be the same tool in use! By tracerouting the source I could verify that the TTL value is correct. I would conclude that most probably the TTL in my original traces above is correct, too. I connected to port 80 to pull the web banner while capturing the traffic with tcpdump:

```
This site is powered by: Apache-AdvancedExtranetServer/2.0.44 (Mandrake Linux/11mdk)
mod_perl/1.99_08 Perl/v5.8.0 mod_ssl/2.0.44 OpenSSL/0.9.7a PHP/4.3.1
```

Applying `p0f` to the trace seems also to hint at Linux, so we could infer that in case of the first traces a unixoid operating system has been used, too.

g) A little work in the lab:

As a proof of concept I constructed similar packets with `hping2 (-N IP-id, -M TCP-seqnum, -w TCP window size)`:

```
hping2 -c 1 -N 666 -s 53 -p 139  -M 666 -VD -S -w 128 localhost

15:04:04.477451 127.0.0.1.53 > 127.0.0.1.139: S [tcp sum ok] 666:666(0) win 128 (ttl 64,
id 666, len 40)
0x0000   4500 0028 029a 0000 4006 7a34 7f00 0001        E..(....@.z4....
0x0010   7f00 0001 0035 008b 0000 029a 76bf 7997        .....5......v.y.
0x0020   5002 0080 bdaf 0000
```

As a part of trivia, I first did this from another host through a Pix firewall - but the Pix scrambles TCP-sequence numbers by default - so most of the effect was mangled.

[79] Kurt Anderson on Intrusions list, Dec 22, 2003: http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00132.html

7. Evidence of active targeting

Well, we have no way of proving that this was *not* directly targeted at our server because no other systems hosted at my provider belong to me, therefore no other log files are available. But had the attacker done reconnaissance beforehand he would probably not have bothered with scanning a Linux host for a port typically in use by Windows, and no Samba service was running here, neither.

The fact that our host was scanned a few times from neighbored IP addresses could indicate repeated coordinated scans of a whole net range. Given fact e) from 6. Correlations above, a horizontal sweep over larger net blocks seems even more probable.

8. Severity

We estimate the severity of the attack according to the SANS formula:

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

Each value is ranked on a scale from 1 (lowest) to 5 (highest).

a) **Criticality**: The server targeted has no mission critical function, it is a playground; even the hosted web server is more of a testing facility. But on the other hand, once compromised all attacks from here would be conducted with me as the suspected originator. So I would rate Criticality as: **2**.

b) **Lethality**: It is not certain that this attack is just a scan or if an exploit would have followed immediately; there are many possible ways to exploit vulnerabilities in un-patched Windows machines over this port. But just looking at the packet we conclude: Lethality: **2**

c) **System Countermeasures**: As the port targeted only affects Windows machines or Samba servers my machine would not be vulnerable to any follow-up. The machine itself is well patched and secured and running an iptables firewall and two intrusion detection systems. System Countermeasures: **5**

d) **Network Countermeasures**: As far as I know there is no network defense in place by my provider. This server is located directly within a dark evil world. Network Countermeasures: **1**.

Severity = 2+3-(5+1)= **-1**. Well, we are somewhat safe here...

9. Defense recommendation

One thing is clear: network countermeasures beyond the confines of my machine are non-existent. So if this server should become mission-critical it is highly advisable to put it behind a screening router and deploy an additional stateful or proxy firewall under our control with a "deny all" default policy and openings just for necessary Internet services. A second network IDS could be set up in front of the firewall to get a clear picture of allowed traffic by using a differential traffic analysis. The host defenses should be updated often and pen-tested regularly. The choice of a unixoid operating system for internet-reachable services should be upheld given the security landscape of today (which may be reevaluated at some later date).

10. Multiple Choice Question

Given the trace below, which answer is most accurate:

```
01:30:21.352051 219.238.233.157.53 > MY.SE.RV.ER.139: S [tcp sum ok] 666:666(0) win 128
(ttl 40, id 666, len 40)
0x0000   4500 0028 029a 0000 2806 15fb dbee e99d        E..(....(.......
0x0010   XXXX XXXX 0035 008b 0000 029a 0000 0000        XXXX.5..........
0x0020   5002 0080 31cd 0000 3100 0000 0000            P...1...1.....
```

(a) This packet is the second in a series of communication attempts by the Stacheldraht DDOS agent after the zombie sends a packet with IP-id 667.

(b) This packet is a DNS response issued by Bind 9.2 or later. This response is issued over a new TCP connection if a former UDP response was truncated.

(c) This packet is Netbios traffic with more than three signs of probable crafting.

(d) This kind of packet is issued by DNS anti-balancers asking for NOTB records.

Correct answer: (c) The signs of crafting are the low source port, IP identification is equal to the TCP sequence number, both numbers equal 666, and a very unusual initial window size of 128. All other answers include false elements.

Detect 3: "A Blast from the Past"

1. Source of the Trace

a) The traces:

i) An example trace found in the wild:

1) Symantec SGS II Firewall

(Normalized) Trace Format:

```
Date - Time - Source IP - Source Port - Destination IP - Destination port - IP protocol -
SGS message (including protocol, flags, interface)


5/24/2004 7:41:04 PM   127.0.0.1      80      MY.COMPANY.160.52      1484    TCP     Deny
       NOTICE: IP packet dropped due to bad source address, IP Code=TCP,   Flag=ACK,
       Adapter=eth1
5/24/2004 7:31:05 PM   127.0.0.1      80      MY.COMPANY.160.52      1639    TCP     Deny
       NOTICE: IP packet dropped due to bad source address, IP Code=TCP,   Flag=ACK,
       Adapter=eth1
5/24/2004 7:30:15 PM   127.0.0.1      80      MY.COMPANY.160.52      1383    TCP     Deny
       NOTICE: IP packet dropped due to bad source address, IP Code=TCP,   Flag=ACK,
       Adapter=eth1
5/24/2004 7:20:14 PM   127.0.0.1      80      MY.COMPANY.160.33      1138    TCP     Deny
       NOTICE: IP packet dropped due to bad source address, IP Code=TCP,   Flag=ACK,
       Adapter=eth1
```

2) Snort IDS (excerpt of more than 1000 similar logs):

(Normalized) Trace Format:

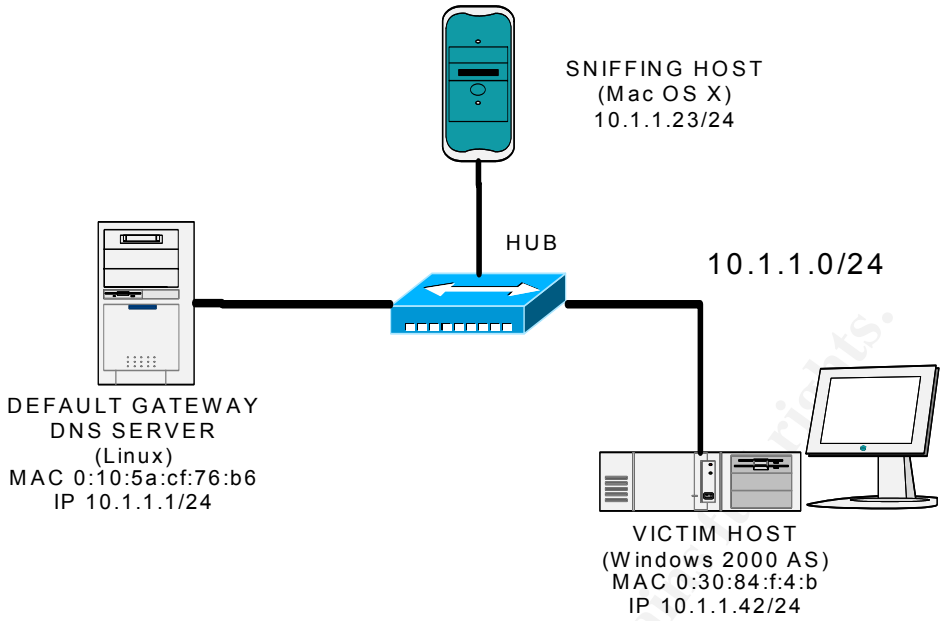
```
Date - Time - Source IP - Source port - Destination IP - Destination port - IP protocol
Snort message & classification - TTL - IP ID - TCP flags


5/24/2004 6:28:19 PM   127.0.0.1      80      MY.COMPANY.164.160     1304    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     23199   ***A*R**

5/24/2004 6:28:20 PM   127.0.0.1      80      MY.COMPANY.164.133     1835    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     24479   ***A*R**

5/24/2004 6:28:20 PM   127.0.0.1      80      MY.COMPANY.163.106     1365    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     26015   ***A*R**

5/24/2004 6:28:20 PM   127.0.0.1      80      MY.COMPANY.163.80      1896    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     27295   ***A*R**

5/24/2004 6:28:20 PM   127.0.0.1      80      MY.COMPANY.165.192     1054    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    118     28575   ***A*R**

5/24/2004 6:28:20 PM   127.0.0.1      80      MY.COMPANY.164.165     1585    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     30623   ***A*R**

...

5/24/2004 7:49:14 PM   127.0.0.1      80      MY.COMPANY.170.54      1706    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     52713   ***A*R**

5/24/2004 7:49:15 PM   127.0.0.1      80      MY.COMPANY.170.238     1136    TCP
BAD-TRAFFIC loopback traffic  Attempted Information Leak    119     50666   ***A*R**
```

ii) A companion trace from my lab:

```
10:13:46.837279 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 66: 10.1.1.42.1160 > 146.45.91.2.135:
S [tcp sum ok] 3780867190:3780867190(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128,
id 42795, len 48)
0x0000   4500 0030 a72b 4000 8006 5b42 0a01 012a        E..0.+@...[B...*
0x0010   922d 5b02 0488 0087 e15b 7476 0000 0000        .-[.....[tv....
0x0020   7002 4000 efe3 0000 0204 05b4 0101 0402        p.@.............
0x0030   73d2 217b                                      s.!{

10:13:46.837760 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 66: 10.1.1.42.1161 > 146.45.91.3.135:
S [tcp sum ok] 3780910362:3780910362(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128,
id 42796, len 48)
0x0000   4500 0030 a72c 4000 8006 5b40 0a01 012a        E..0.,@...[@...*
0x0010   922d 5b03 0489 0087 e15c 1d1a 0000 0000        .-[......\......
0x0020   7002 4000 473d 0000 0204 05b4 0101 0402        p.@.G=..........
0x0030   1cf1 c34f                                      ...O

10:13:46.838258 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 66: 10.1.1.42.1162 > 146.45.91.4.135:
S [tcp sum ok] 3780949200:3780949200(0) win 16384 <mss 1460,nop,nop,sackOK> (DF) (ttl 128,
id 42797, len 48)
0x0000   4500 0030 a72d 4000 8006 5b3e 0a01 012a        E..0.-@...[>...*
0x0010   922d 5b04 048a 0087 e15c b4d0 0000 0000        .-[......\......
0x0020   7002 4000 af84 0000 0204 05b4 0101 0402        p.@.............
0x0030   d01d a0d0                                      ....

10:13:46.838448 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.63.201.1691: R
[tcp sum ok] 0:0(0) ack 1445789697 win 0 (ttl 128, id 42799, len 40)
0x0000   4500 0028 a72f 0000 8006 cad5 7f00 0001        E..(./..........
0x0010   0a01 3fc9 0050 069b 0000 0000 562d 0001        ..?..P......V-..
0x0020   5014 0000 89ec 0000 2020 2020 2020 4e93        P.............N.
0x0030   cc0e                                           ..

10:13:46.857930 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.128.201.1526:
R [tcp sum ok] 0:0(0) ack 2080440321 win 0 (ttl 128, id 42801, len 40)
0x0000   4500 0028 a731 0000 8006 89d3 7f00 0001        E..(.1..........
0x0010   0a01 80c9 0050 05f6 0000 0000 7c01 0001        .....P......|...
0x0020   5014 0000 23bd 0000 2020 2020 2020 b361        P...#..........a
0x0030   43ea                                           C.

10:13:46.878050 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.194.73.1361: R
[tcp sum ok] 0:0(0) ack 567541761 win 0 (ttl 128, id 42803, len 40)
0x0000   4500 0028 a733 0000 8006 4851 7f00 0001        E..(.3....HQ....
0x0010   0a01 c249 0050 0551 0000 0000 21d4 0001        ...I.P.Q....!...
0x0020   5014 0000 3d0f 0000 2020 2020 2020 0037        P...=..........7
0x0030   e6c5                                           ..

10:13:46.898014 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.4.200.1197: R
[tcp sum ok] 0:0(0) ack 1202192385 win 0 (ttl 128, id 42805, len 40)
0x0000   4500 0028 a735 0000 8006 05d1 7f00 0001        E..(.5..........
0x0010   0a01 04c8 0050 04ad 0000 0000 47a8 0001        .....P......G...
0x0020   5014 0000 d560 0000 2020 2020 2020 f5b5        P....`..........
0x0030   5855                                           XU

10:13:46.918025 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.69.200.1032: R
[tcp sum ok] 0:0(0) ack 1836777473 win 0 (ttl 128, id 42807, len 40)
0x0000   4500 0028 a737 0000 8006 c4ce 7f00 0001        E..(.7..........
0x0010   0a01 45c8 0050 0408 0000 0000 6d7b 0001        ..E..P......m{..
0x0020   5014 0000 6f32 0000 2020 2020 2020 38ec        P...o2........8.
0x0030   8ece                                           ..

10:13:46.938155 0:30:84:f:4:b 0:10:5a:cf:76:b6 0800 64: 127.0.0.1.80 > 10.1.134.72.1868: R
[tcp sum ok] 0:0(0) ack 323944449 win 0 (ttl 128, id 42809, len 40)
0x0000   4500 0028 a739 0000 8006 844c 7f00 0001        E..(.9.....L....
0x0010   0a01 8648 0050 074c 0000 0000 134f 0001        ...H.P.L.....O..
0x0020   5014 0000 859a 0000 2020 2020 2020 4ba5        P.............K.
0x0030   1d08                                           ..
```

b) The source of the traces:

The corresponding trace ii) was generated in my home lab, which was build like this:



2. Detect was generated by …

Trace number i) is an example trace from the wild to give a real example of the phenomenon described here. It was detected at my companies defenses, where a high level automatic correlation of log files from Snort and a Symantec Gateway Security II (SGS II) firewall is taking place, hence the given normalized log format. The format of the traces is given above.

Trace number ii) was generated by a specially prepared setup from my home lab. VICTIM HOST, a Windows 2000 Server machine, was artificially infected by the Windows 2000 Version of the Blaster.A Worm, which had been captured from a compromised Honey pot in late August 2003 and been archived until now. This host is running the default install of IIS on port 80. The time on the machine was set to 17th of June 2004, as the testing day was not later than the 16th of the month, which is important for the described events to take place.

3. Probability the source address was spoofed

The whole lab setup was done to solve the riddle of similar detects to i). If you see a sender address of 127.0.0.1 (localhost), spoofing comes immediately to mind as in normal traffic there would be no way that such a packet would leave a host - instead of using a real IP address. The next problem is, as the transport layer protocol is TCP that would require return packets for completing the three-way handshake, there is no way that this could be accomplished, as no return traffic would even leave a possible responding host.

With UDP you could expect for example that a single packet exploit might work. But what could be gained by sending this kind of traffic with any kind of intent?

As further investigation shows, the source IP is not really spoofed in the classical sense, instead we see a valid return packet to a source spoofing attack - in some sense a specialized form of "Third-party-effect" in the sense of Richard Bejtlich[80]. In our case some special DNS configuration adds an additional twist.

4. Description of attack

This trace i) and similar detects are most probably related to the W32.Blaster Worm[81], which tries to exploit the Microsoft DCOM vulnerability (MS03-26).[82,83,84,85,86] The propagation vector of this worm uses port 135/tcp.

But as was noted early by code analysis, infected machines would start a Denial-of-service attack versus "windowsupdate.com", one of the aliases of Microsoft's Windows update service. That attack would only take place at certain times of the year and be conducted by sending floods of SYN packets to port 80/tcp (HTTP).

In the days before Microsoft removed the corresponding A-record for this address to escape this DOS-attack there were some advisories on the web who proposed the following mitigating strategy for companies with infected hosts: reconfigure the DNS-servers under your control with a fake entry for windowsupdate.com - as the worm would try to resolve this name and attack the IP address that was delivered to it.

One natural candidate seemed to be localhost (127.0.0.1) in hope that then the DOS traffic would not even leave the infected host. Obviously, at least some people followed this strategy - which had one serious flaw that was not seen at that time.

As the worm spoofed the source IP for this DOS attack with random legal addresses, the TCP/IP stack on the machine reacts to these countless "connection attempts" to localhost by sending a plain RESET "back" to the spoofed IP addresses - if no HTTP-server was listening; but even if there is such a server running, it gets overwhelmed within seconds and refuses further attempts by RESET. This behavior was confirmed in the lab that we set up, an excerpt of captured traffic is seen above as trace ii).

[80] Richard Bejtlich, Network Intrusion Detection of Third Party Effects: http://home.satx.rr.com/bejtlich/nid_3pe_v101.pdf
[81] Symantec Security Response, Blaster Worm: http://securityresponse.symantec.com/avcenter/venc/data/pf/w32.blaster.worm.html
[82] Microsoft Security Bulletin MS03-026: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
[83] Microsoft "What You Should Know About the Blaster Worm": http://www.microsoft.com/security/incident/blast.asp
[84] Microsoft PSS Security Response Team Alert: http://www.microsoft.com/technet/security/alerts/msblaster.mspx
[85] eEye Blaster Worm Advisory: http://www.eeye.com/html/Research/Advisories/AL20030811.html
[86] eEye Blaster Worm Analysis: http://www.eeye.com/html/Research/Advisories/Blaster_Analysis.txt

5. Attack mechanism

To test our thesis that trace i) could be very well generated by Blaster backlash we set up a Bind 9 name-server on the Linux machine that additionally functions as default gateway and as the only DNS server for the Windows machine "VICTIM". I configured Bind to be authoritative for "windowsupdate.com" and constructed a corresponding zone file that included a mapping of windowsupdate.com to 127.0.0.1.

Excerpt of `/etc/named.conf`:

```
zone "windowsupdate.com" in {
       type master;
       file "windowsupdate.com.zone";
};
```

Content of `/var/lib/name/windowsupdate.com.zone`:

```
$TTL 1W
@               IN SOA  @   root (
                                42              ; serial (d. adams)
                                2D              ; refresh
                                4H              ; retry
                                6W              ; expiry
                                1W )            ; minimum

                IN NS           @
                IN A            127.0.0.1
```

After resolving "windowsupdate.com" to 127.0.0.1 the worm started the attack because the date was set to the right time. A quote from the above referenced Symantec Security Response article:

"If the current date is the 16th through the end of the month for the months of January to August, or if the current month is September through December, the worm will attempt to perform a DoS on Windows Update. (...)

The DoS traffic has the following characteristics:
Is a SYN flood on port 80 of windowsupdate.com.
Tries to send 50 HTTP packets every second.
Each packet is 40 bytes in length.
If the worm cannot find a DNS entry for windowsupdate.com, it uses a destination address of 255.255.255.255.

Some fixed characteristics of the TCP and IP headers are:
IP identification = 256
Time to Live = 128
Source IP address = a.b.x.y, where a.b are from the host ip and x.y are random. In some cases, a.b are random.
Destination IP address = dns resolution of "windowsupdate.com"
TCP Source port is between 1000 and 1999
TCP Destination port = 80
TCP Sequence number always has the two low bytes set to 0; the 2 high bytes are random.
TCP Window size = 16384"

It did really send - invisible to us as this traffic did not leave the machine, but a corresponding `netstat` confirmed this - SYN-packets to port 80 on 127.0.0.1 from randomly spoofed IP addresses and random (mostly ephemeral) high ports between 1000 and 2000.

The TCP/IP stack tried to deliver RESET as return packet - from source port 80 on 127.0.0.1 to the high port. As the targets for these packets are valid, they left the host and got transported on the net. If no egress filtering or anti-spoofing mechanism is in place that prevents packets from localhost leaving a network, this traffic will reach the target addresses, because these are valid routing targets.

6. Correlations

a) The original solution to this kind of detects was given by Dan Hanson on the Incidents mailing list.[87] My lab setup gives some good arguments in support for that theory.

b) Similar sighting on a MS ISA-server newsgroup:[88]

```
   Date Time Source IP Target IP Protocol Source Port/ICMP Type Target
Port/ICMP Code Flag(s) Rule Header Payload

      01/09/2003 00:17:12 127.0.0.1 213.48.225.206 Tcp 80 1519 RST ACK
BLOCKED 45 00 00 28 06 37 00 00 78 06 06 99 7f 00 00 01 d5 30 e1 ce 00 50 05
ef 00 00 00 00 6f 93 00 01 50 14 00 00 03 fd 00 00
      01/09/2003 00:26:41 127.0.0.1 213.48.225.206 Tcp 80 1360 RST ACK
BLOCKED 45 00 00 28 32 64 00 00 78 06 da 6b 7f 00 00 01 d5 30 e1 ce 00 50 05
50 00 00 00 00 3b a6 00 01 50 14 00 00 38 89 00 00
      01/09/2003 00:34:49 127.0.0.1 213.48.225.206 Tcp 80 1519 RST ACK
BLOCKED 45 00 00 28 05 f4 00 00 78 06 06 dc 7f 00 00 01 d5 30 e1 ce 00 50 05
ef 00 00 00 00 6f 93 00 01 50 14 00 00 03 fd 00 00
      01/09/2003 11:05:23 127.0.0.1 213.48.225.206 Tcp 80 1279 RST ACK
BLOCKED 45 00 00 28 02 6e 00 00 7a 06 08 62 7f 00 00 01 d5 30 e1 ce 00 50 04
ff 00 00 00 00 6e 56 00 01 50 14 00 00 06 2a 00 00
      01/09/2003 12:21:17 127.0.0.1 213.48.225.206 Tcp 80 1750 RST ACK
BLOCKED 45 00 00 28 2f e4 00 00 78 06 dc eb 7f 00 00 01 d5 30 e1 ce 00 50 06
d6 00 00 00 00 12 fa 00 01 50 14 00 00 5f af 00 00
      01/09/2003 12:21:31 127.0.0.1 213.48.225.206 Tcp 80 1279 RST ACK
BLOCKED 45 00 00 28 05 db 00 00 78 06 06 f5 7f 00 00 01 d5 30 e1 ce 00 50 04
ff 00 00 00 00 6e 56 00 01 50 14 00 00 06 2a 00 00
      01/09/2003 16:25:51 127.0.0.1 213.48.225.206 Tcp 80 1750 RST ACK
BLOCKED 45 00 00 28 07 de 00 00 78 06 04 f2 7f 00 00 01 d5 30 e1 ce 00 50 06
d6 00 00 00 00 12 fa 00 01 50 14 00 00 5f af 00 00
      01/09/2003 16:42:38 127.0.0.1 213.48.225.206 Tcp 80 1152 RST ACK
BLOCKED 45 00 00 28 d9 c2 00 00 78 06 33 0d 7f 00 00 01 d5 30 e1 ce 00 50 04
80 00 00 00 00 1e e6 00 01 50 14 00 00 56 19 00 00
      01/09/2003 16:43:29 127.0.0.1 213.48.225.206 Tcp 80 1408 RST ACK
BLOCKED 45 00 00 28 ee cd 00 00 78 06 1e 02 7f 00 00 01 d5 30 e1 ce 00 50 05
80 00 00 00 00 5d 0a 00 01 50 14 00 00 16 f5 00 00
      01/09/2003 17:09:58 127.0.0.1 213.48.225.206 Tcp 80 1152 RST ACK
BLOCKED 45 00 00 28 8f 96 00 00 7a 06 7b 39 7f 00 00 01 d5 30 e1 ce 00 50 04
80 00 00 00 00 1e e6 00 01 50 14 00 00 55 19 01 00
```

Quote from the same source:

[87] Dan Hanson on Incidents list, Oct 28, 2003: http://www.securityfocus.com/archive/75/342726/2003-10-26/2003-11-01/0
[88] Jim Harrison on microsoft.public.isaserver, Sep 10, 2003:
http://groups.google.com/groups?selm=2qn7b.5299%24F_4.49860809%40news-text.cableinet.net

"I ran Network monitor for a brief period last night and I noticed the Source IP (127.0.0.1) always has the same MAC address as my external (dialup) adapter. I don't know, but I assumed 127.0.0.1 represented my LAN adapter"

7. Evidence of active targeting

We see responses to packets from spoofed source IP addresses, which have been randomly generated. So this is not a case of active targeting, just a kind of backscatter effect.

8. Severity

We estimate the severity of the trace i) according to the formula recommended by SANS:

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

Each value is ranked on a scale from 1 (lowest) to 5 (highest).

a) **Criticality**: Trace i) resemble a horizontal scan of many hosts with different function on our network, so we apply a summarizing value here: **3**.

b) **Lethality**: No response could be expected from sending RESETs. No reconnaissance value would return due to the source of localhost. No single packet exploit has been traced - and this is TCP. So I would rate it as Lethality: **1**.

c) **System Countermeasures**: As many hosts are concerned here we assign a summary value here, too: **3**.

d) **Network Countermeasures**: Firewall, IDS system, 24/7 monitoring - **5**.

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

9. Defense recommendation

The targeted network is well protected. But on the sender's side there should be removal of the 127.0.0.1 entry for windowsupdate.com - if no entry is to be found, the attack does not take place, and no return packets leave the network.

This brings up the second major point: apply network ingress[89,90] and egress filtering and prevent packets with non-valid sources or destination (private addresses/RFC 1918, multicast, localhost)[91].

[89] RFC 2827, Ingress Filtering for Multihomed Networks: ftp://ftp.rfc-editor.org/in-notes/rfc2827.txt
[90] Updated by RFC 3704: ftp://ftp.rfc-editor.org/in-notes/rfc3704.txt

Better still, this should be done on every router in-between and also on the border router of the target network - so this would be a recommendation for improving any network defense.

10. Multiple Choice Question

You are sending a TCP SYN-packet from a Windows 2000 machine with IP 10.1.1.42 (Host A) to the closed port 80 on the same machine (still Host A) by using a spoofed IP address of 10.1.1.23 (Host B) and specifying a destination IP address of 127.0.0.1 (localhost) in the packet. What would happen?

(a) Nothing, as this kind of traffic is dropped by default.

(b) If the Host A is using Windows 2000 SP1 it will be caught in an infinite loop.

(c) Host A will send a RESET from port 80 and IP 127.0.0.1 to Host B.

(d) Host A will send a RESET from port 80 and source IP 10.1.1.42 to Host B.

Correct answer: (c)

[91] SANS, "Help Defeat Denial of Service Attacks": http://www.sans.org/dosstep/

Practical Assignment Part Three:

III. Analyze This

Executive Summary

We analyzed traffic on your GCIA University (UGCIA) network from 7th until 12th April 2004.

The analysis was initially hampered by the fact that we had not enough data from consecutive days due to the corruption of some of the provided log files. This would be our first recommendation: ensure that log files are collected constantly and verified for file or compression corruptions. This data is at the core for any further analysis.

We discovered some oddities in the way your Snort IDS is reacting on horizontal inbound scans; it is possible that it is missing part of the traffic. High priority for future defense is to verify that Snort sees what it needs.

After identifying your most valuable network assets by the traffic that reaches and emanates from them, our main focus lies on possible infected or compromised hosts on your network. This means, that our main interest lies in outbound traffic, but surely inbound traffic is correlated, too.

We have found some hosts on your network that have shown signs of compromise or infection. The possible worms are Blaster Worm, Gaobot, Adware/Spyware and a possible HTTP worm.

A large amount of suspicious IRC traffic is entering and leaving your network. We supply a list of your hosts that need investigation due to the high possibility of compromise and use as file-sharing bots.

In the course of investigations we discovered evidence of peer-to-peer activity on your network. While not per se malicious, chances are that users might violate copyrights (depending on the nature of files shared) and circumvent possible gateway virus-protection.

Inbound scans to your network are quite usual activity on the Internet today, we have investigated some cases in more detail. Be as restrictive in your access-policy as possible and block unnecessary ports, especially high ports, eitherbound.

We have identified false positives where necessary. Fine-tuning your signatures now will help you detect the truly malicious events in the future.

Log files

The source of the log files is a Snort located in a University network. Nothing is known about the exact version and rule set, though it is clear from the logs that many custom rules do exist.

The logs have been downloaded from the GIAC section of the SANS homepage.[92] It was difficult to find five consecutive days not to long in the past that had a complete set from Alerts, Scans and OOS ("out-of-spec") files, even if you could manage to extract data from damaged archives.

After consulting SANS I was authorized to use the following set, where "i" denotes incomplete records due to archive damage, which was compensated by using an extra day:

```
alert.040407.gz
alert.040408.gz
alert.040409.gz
alert.040410.gz
alert.040411.gz

oos_report_040403
oos_report_040404
oos_report_040405
oos_report_040406
oos_report_040407
oos_report_040408

scans.040407.gz
scans.040408.gz (i)
scans.040409.gz (i)
scans.040410.gz
scans.040411.gz (i)
scans.040412.gz
```

The naming convention of the "OOS" log files is somewhat misleading. If you look at the dates of the actual logs they are four days later than the filename seems to imply. I chose the set with correct log dates.

The timeframe for the logs is therefore from 7th until 12th April 2004.

The university network (a class B) was obfuscated throughout the logs as the network "MY.NET.0.0/16". I did this manually where the original IP was to be seen; additionally I masked the identity of the university as "UGCIA" when it was necessary in the logs from custom signatures. All this was done according to the GCIA guidelines and best practices among other analysts.

Important Hosts and Relations

[92] SANS GIAC logs: http://isc.sans.org/logs/

The following list is an attempt to identify some major landmarks in the UGCIA network. As this is based on information from the given log files combined with some cautious inverse DNS resolutions, this cannot be an exhaustive list.

DNS

MY.NET.1.3
MY.NET.1.4
MY.NET.1.5

NTP

MY.NET.1.3

HTTP

MY.NET.6.7
MY.NET.12.11: www.ugcia.edu (Main site)
MY.NET.12.12
MY.NET.24.33: my.ugcia.edu
MY.NET.24.34
MY.NET.24.44
MY.NET.34.11
MY.NET.60.14
MY.NET.75.13
MY.NET.109.53
MY.NET.110.82

HTTPS

MY.NET.12.7: webauth.ugcia.edu
MY.NET.24.33: my.ugcia.edu
MY.NET.24.34
MY.NET.24.74: webmail.ugcia.edu
MY.NET.30.3: lan1.ugcia.edu (Netware?)
MY.NET.30.4: lan2.ugcia.edu (Netware?)

FTP

MY.NET.24.27
MY.NET.24.47: ftp1.ugcia.edu
MY.NET.30.3
MY.NET.53.29 (Helpdesk)
MY.NET.70.49 (Helpdesk)
MY.NET.70.50 (Helpdesk)

MAIL (SMTP, IMAP, IDENT)

MY.NET.12.2: smtp.ugcia.edu
MY.NET.12.4
MY.NET.12.6
MY.NET.25.10
MY.NET.25.12
MY.NET.25.66
MY.NET.25.66 - 75
MY.NET.34.5
MY.NET.34.14
MY.NET.60.38
MY.NET.111.34

SSH

MY.NET.28.22
MY.NET.34.3
MY.NET.34.4
MY.NET.60.16
MY.NET.60.38
MY.NET.60.39

DIAL-UP

MY.NET.97.0/24

Detects

This is the list of alerts generated between 7th and 11th April 2004:
('COMMA' is representing ',' in my database)

count	alert
28826	EXPLOIT x86 NOOP
12996	MY.NET.30.3 activity
12170	SMB Name Wildcard
10664	High port 65535 tcp - possible Red Worm - traffic
10207	MY.NET.30.4 activity
8010	Tiny Fragments - Possible Hostile Activity
3258	DDOS mstream handler to client
1127	Null scan!
1098	NMAP TCP ping!
1081	Possible trojan server activity
930	External RPC call
637	SUNRPC highport access!
511	Incomplete Packet Fragments Discarded
309	TCP SRC and DST outside network
244	High port 65535 udp - possible Red Worm - traffic
210	ICMP SRC and DST outside network
158	[UGCIA NIDS] Internal MiMail alert
147	[UGCIA NIDS IRC Alert] IRC user /kill detectedCOMMA possible trojan.
142	DDOS shaft client to handler
108	[UGCIA NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC
100	FTP passwd attempt
83	TCP SMTP Source Port traffic
72	IRC evil - running XDCC
66	EXPLOIT x86 setuid 0
55	SMB C access
47	[UGCIA NIDS] External MiMail alert
46	connect to 515 from outside
33	EXPLOIT x86 setgid 0
28	EXPLOIT x86 stealth noop
25	[UGCIA NIDS IRC Alert] Possible drone command detected.
24	RFB - Possible WinVNC - 010708-1
22	FTP DoS ftpd globbing
17	[UGCIA NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.
15	NIMDA - Attempt to execute cmd from campus host
14	Attempted Sun RPC high port access
14	TFTP - Internal UDP connection to external tftp server
13	SYN-FIN scan!
10	EXPLOIT NTPDX buffer overflow
8	EXPLOIT x86 NOPS
6	Probable NMAP fingerprint attempt
6	DDOS mstream client to handler
4	TFTP - External TCP connection to internal tftp server
3	NETBIOS NT NULL session
2	[UGCIA NIDS IRC Alert] K:line'd user detectedCOMMA possible trojan.
2	[UGCIA NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot
2	PHF attempt
1	[UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC
1	External FTP to HelpDesk MY.NET.70.50
1	External FTP to HelpDesk MY.NET.53.29
1	External FTP to HelpDesk MY.NET.70.49
1	Fragmentation Overflow Attack

MY.NET.30.3 activity & MY.NET.30.4 activity

Obviously, the hosts MY.NET.30.3 and MY.NET.30.4 do host important applications if there are custom Snort rules to watch any activity targeting them - including probably fair use of the services, if the signature name is to be trusted. This signature has not captured any outbound activity originating from these hosts.

MY.NET.30.3 activity:

destip	destport	count
MY.NET.30.3	21	3
MY.NET.30.3	80	455
MY.NET.30.3	389	5
MY.NET.30.3	427	3
MY.NET.30.3	443	2
MY.NET.30.3	446	1
MY.NET.30.3	524	12298
MY.NET.30.3	554	3
MY.NET.30.3	715	3
MY.NET.30.3	1025	8
MY.NET.30.3	1080	9
MY.NET.30.3	1433	2
MY.NET.30.3	2745	75
MY.NET.30.3	2812	3
MY.NET.30.3	3019	13
MY.NET.30.3	3128	8
MY.NET.30.3	3389	3
MY.NET.30.3	3410	7
MY.NET.30.3	4000	4
MY.NET.30.3	4899	20
MY.NET.30.3	5000	8
MY.NET.30.3	6129	43
MY.NET.30.3	8000	3
MY.NET.30.3	9898	1
MY.NET.30.3	12849	2
MY.NET.30.3	20168	11
MY.NET.30.3	55838	1

MY.NET.30.4 activity:

destip	destport	count
MY.NET.30.4	21	2
MY.NET.30.4	80	2253
MY.NET.30.4	389	5
MY.NET.30.4	446	1
MY.NET.30.4	524	447
MY.NET.30.4	554	2
MY.NET.30.4	715	3
MY.NET.30.4	1025	9
MY.NET.30.4	1080	7
MY.NET.30.4	1433	2
MY.NET.30.4	2745	85
MY.NET.30.4	2812	2
MY.NET.30.4	3128	9
MY.NET.30.4	3389	3
MY.NET.30.4	3410	7
MY.NET.30.4	3862	1
MY.NET.30.4	4000	4
MY.NET.30.4	4899	20
MY.NET.30.4	5000	9
MY.NET.30.4	6129	59
MY.NET.30.4	8000	3
MY.NET.30.4	8888	1
MY.NET.30.4	9898	1
MY.NET.30.4	10080	1
MY.NET.30.4	12849	1
MY.NET.30.4	20168	11
MY.NET.30.4	20480	1
MY.NET.30.4	26112	1
MY.NET.30.4	51443	7255
MY.NET.30.4	55838	1
MY.NET.30.4	57778	1

Dominating in frequency is port 524/tcp, that is usually associated with NCP, Netware Core Protocol.[93,94] This protocol is the main "interface" protocol used at Novell's Netware servers to manage access to its resources. So most probably these hosts are really Netware servers.

The other common port sighted is 80/tcp; as an Apache/Tomcat is a part of Netware 6 installation[95], this is no big surprise. In addition we see some HTTPS (443/tcp) access (attempts?) at MY.NET.30.3.

Most of the other ports (1025, 1080, 1433, 2745, 3128, 5000, 6129) are typical for non-specifically targeted proxy scans, bot net (e.g. Gaobot variants[96]) and worm activity versus Internet-exposed hosts. The non-targeted character is confirmed by query results from the "Scan" logs.

The IP 212.45.4.202 is scanning horizontally most part of MY.NET for the unusual port 446/tcp, that is officially associated with "ddm-rdb", or DDM-Remote

[93] Entry on NCP from protocols.com: http://www.protocols.com/pbook/novel.htm#NCP
[94] Entry on NCP at Networksorcery: http://www.networksorcery.com/enp/protocol/ncp.htm
[95] Novell: "Apache/Tomcat Environment on Novell as a Java development environment on Novell Netware 6": http://www.novell.com/info/collateral/docs/4621204.01/4621204.pdf
[96] Symantec Security Response on Gaobot.AFW (example): http://www.symantec.com/avcenter/venc/data/w32.gaobot.afw.html

Relational Database Access, which seems to be mainly used by DB2 distributed databases.[97] Some problems and possible vulnerabilities of DDM have been outlined by Michael Walsh.[98] However, this is no active targeting of our host.

Most interesting is the really impressive peak in activity for port 51443/tcp on MY.NET.30.4. At first glance this might indicate a back door on a random high port. But is it probable that a backdoor is more frequently accessed than the main server port? In addition, this port is used as an alternate HTTPS port in a Netware Installation, if Netstorage and Novell Enterprise Server are installed on the same host:[99]

"During the NetWare 6 Support Pack installation and configuration process, an administrator might inadvertently set the port number used by NetStorage to the same port number used by NetWare Enterprise Server. Manually changing the NetStorage port number after the installation also might cause a port conflict or an invalid port setting. NetStorage requires the Apache Web Server and must use the same port number as the Apache Web Server. The default port number for NetWare Enterprise Server is 80 for HTTP and 443 for HTTPS. If you have NetWare Enterprise Server installed, by default the Apache Web Server will get port 51080 for HTTP and 51443 for HTTPS."

So there is no hint at a possible backdoor on these servers, nor at other high amount of unusual traffic. The caveat however remains, that we probably only see inbound connection attempts.

High port 65535 tcp - possible Red Worm - traffic

As the name of implies, this seems to be a purely port based signature, that is custom made. The Red Worm, later called Adore Worm[100], is a Linux based worm exploiting weaknesses in LPRng, rpc-statd, wu-ftpd and BIND.[101] Under special circumstances it opens a backdoor on port 65535/tcp.[102]

Exactly 4861 alerts are caused by a SSH connection from IP 141.157.102.155 (see Lookups) to server MY.NET.60.16 from 23:45:06.61 on April 8th until 00:59:40.58 on April 9th, duration around 75 minutes. This connection has source port 65535. We see other packets from source port 65535 to well-known destination ports (22, 25, 80, 110, 113, and Gnutella on 6346). These alerts are probably false positives.

Between the 8th and 10th April we see activity between IP 24.5.46.4 (Port 65535) and several IPs from MY.NET (2348 logs). "OOS" isn't helpful here, and "Scans" just gives us two MY.NET IPs being seen scanning IP 24.5.46.4:

---

[97] IBM iSeries Information Center, TCP/IP communication support concepts for DDM : http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm?info/ddp/rbal1sockcon.htm
[98] Michael Walsh: "Some of the Dangers of Connecting your AS/400 to a Network2; SANS Reading Room: http://www.sans.org/rr/papers/56/308.pdf
[99] NetWare 6.0 Support Pack 3 - TID2965459: http://support.novell.com/cgi-bin/search/searchtid.cgi?/2965459.htm
[100] Anthony Dell, GSEC practical: http://www.giac.org/practical/gsec/Anthony_Dell_GSEC.pdf
[101] SANS on Adore Worm: http://www.sans.org/y2k/adore.htm
[102] Sophos on Adore Worms: http://www.sophos.com/virusinfo/analyses/linuxadore.html

sourceip	sourceport	destip	destport
MY.NET.97.51	3322	24.5.46.4	65535
MY.NET.97.51	3322	24.5.46.4	65535
MY.NET.97.51	3482	24.5.46.4	65535
MY.NET.97.51	3482	24.5.46.4	65535
MY.NET.97.51	4698	24.5.46.4	65535
MY.NET.97.51	4294	24.5.46.4	65535
MY.NET.97.51	4294	24.5.46.4	65535
MY.NET.97.196	4835	24.5.46.4	65535
MY.NET.97.196	4835	24.5.46.4	65535

Obviously, the "Scans" files just give us a small piece of the whole picture. This might be caused by the longevity of the phenomenon, which might partially elude the scanning engine by staying well below detection thresholds.

Without flags we cannot be sure of the scanning direction; the scan logs could be caused by return traffic. I find it more probable that an external IP is slowly scanning multiple high ports on multiple internal machines, each time using source port 65535. Either way, there is no indication of an internal Red Worm infection.

From the remaining alerts, only the following have destination port 65535 and destination IP from MY.NET, and insofar might indicate a backdoor left by the worm on our net:

day	timestamp	sourceip	sourceport	destip	destport
08	11:02:29.297724	64.12.200.89	5190	MY.NET.71.248	65535
08	11:02:29.342317	64.12.200.89	5190	MY.NET.71.248	65535
08	11:02:29.360719	64.12.200.89	5190	MY.NET.71.248	65535
08	11:02:29.774756	64.12.200.89	5190	MY.NET.71.248	65535

But this is return traffic from AOL Instant messenger, so a false positive again.

In general it would be wise to tune the snort signature to only alert on initial connection attempts (SYN) to internal hosts on port 65535. An example might be (using a recommended local SID):

```
alert tcp any any -> $HOME_NET 65535 (flags:S,12; msg: "Possible Red Worm backdoor access"; sid:1000002; rev: 2;)
```

Possible trojan server activity

This custom signature just seems to look for any activity involving port 27374/tcp, that is associated with SubSeven and other "Remote access trojans"[103], thereby causing many false positive from return traffic to this source port.

[103] LinkLogger on SubSeven : http://www.linklogger.com/TCP27374.htm

By filtering out this return traffic to common ports we discover returning Edonkey and Gnutella traffic, which will not be focused on here.

Visible now are horizontal scans from two apparently neighbored addresses, IP 213.189.89.54 and IP 213.189.89.109 (see Lookups section), that scan internal hosts mainly from MY.NET.190.0/24 for port 27374/tcp. "Scans" sees the activity vs. MY.NET.190.0/24, "Alerts" has some more logs:

day	timestamp	sourceip	sourceport	destip	destport
10	14:14:52.280387	213.189.89.54	2899	MY.NET.5.5	27374
10	14:15:29.197499	213.189.89.54	1439	MY.NET.6.15	27374
10	14:22:08.647800	213.189.89.54	1865	MY.NET.16.90	27374
10	14:22:08.653556	213.189.89.54	1881	MY.NET.16.106	27374
10	14:22:14.599045	213.189.89.54	2029	MY.NET.16.114	27374
10	16:14:09.858628	213.189.89.54	2944	MY.NET.190.1	27374
10	16:14:10.560797	213.189.89.54	2944	MY.NET.190.1	27374
10	16:14:12.066310	213.189.89.54	2945	MY.NET.190.2	27374
10	16:14:12.072928	213.189.89.54	2943	MY.NET.190.0	27374
...	...	...	...	...	...
10	14:16:42.857265	213.189.89.109	1862	MY.NET.16.90	27374
10	14:16:45.845623	213.189.89.109	1862	MY.NET.16.90	27374
10	14:16:45.876434	213.189.89.109	2038	MY.NET.16.106	27374
10	14:16:45.878369	213.189.89.109	2046	MY.NET.16.114	27374
10	14:16:48.856613	213.189.89.109	2038	MY.NET.16.106	27374
10	14:16:48.857133	213.189.89.109	2046	MY.NET.16.114	27374
10	16:08:20.407134	213.189.89.109	2239	MY.NET.190.0	27374
10	16:08:20.407193	213.189.89.109	2240	MY.NET.190.1	27374
...	...	...	...	...	...

Considering the timestamps it is possible that Snort is missing some parts of the Scan; this should definitely be looked after, perhaps by performing some testing scans.

Some hosts do reply to the scan, though it is unclear if by sending an RESET or SYN-ACK, as this is not included in the logs. The set of hosts that are observed to react is the same for both scanning IPs:

```
MY.NET.6.15
MY.NET.190.1
MY.NET.190.93
MY.NET.190.95
MY.NET.190.97
MY.NET.190.102
MY.NET.190.202
MY.NET.190.203
```

Perhaps an administrator should conduct an authorized, focused to see if these ports are really open.

IRC Activity: [UGCIA NIDS IRC Alert]

This is an overview of IRC-related activity detected on the network:

alert
IRC evil - running XDCC [UGCIA NIDS IRC Alert] IRC user /kill detectedCOMMA possible trojan.

```
| [UGCIA NIDS IRC Alert] K:line'd user detectedCOMMA possible trojan.          |
| [UGCIA NIDS IRC Alert] Possible drone command detected.                     |
| [UGCIA NIDS IRC Alert] Possible Incoming XDCC Send Request Detected.        |
| [UGCIA NIDS IRC Alert] Possible sdbot floodnet detected attempting to IRC   |
| [UGCIA NIDS IRC Alert] User joining XDCC channel detected. Possible XDCC bot|
| [UGCIA NIDS IRC Alert] XDCC client detected attempting to IRC               |
+-----------------------------------------------------------------------------+
```

We look at some alerts with a detailed view:

User joining XDCC channel detected. Possible XDCC bot

A Snort signature with that message can be found on a rich IRC-rules web resource by Perry Lorier[104], that seems to offer many signatures in use at UGCIA:

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

XDCC client detected attempting to IRC

Signature from the same source:

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

Possible Incoming XDCC Send Request Detected

Signature from the same source:

```
alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any (content: " |3a 01|XDCC "; msg:
"Possible Incoming XDCC Send Request Detected."; classtype: misc-activity; )
```

Possible drone command detected

Signature from the same source:

```
alert tcp $EXTERNAL_NET 6660:7000 -> $HOME_NET any ( content: " PRIVMSG "; content:
"\:.login"; nocase; msg: "Possible drone command detected."; classtype:misc-activity;)
```

IRC evil - running XDCC

Possibly the signature resembles one I found on another location (C. Cramer):[105]

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

The author states:

"For the XDCC bots we've been using the following snort (v 1.7) rules on the outbound connections. The content is a string that the XDCC bots periodically pop up."

[104] Perry Lorier's IRC rules: http://coders.meta.net.nz/~perry/irc.rules
[105] Christopher Cramer on unisog list: http://lists.sans.org/pipermail/unisog/2002-May/001485.php

Discussion:

Broadbandreports' FAQ has the following definition of XDCC bots:[106]

> ***Q: What's an XDCC?*** *(#4493)*
>
> ***A:*** *With IRC in full swing, XDCC bots are common sights in channels these days. An XDCC is a bot that has certain packets uploaded to it. These packets may be anything from the recent game to a good movie. XDCCs are usually r00ted (hacked), and transfer at very high speeds because they are on fast lines.*

This looks like serious activity, though we cannot state anything with absolute certainty without actual session data. It is possible that there are compromised internal hosts functioning as file sharing and Warez bots.[107] Interestingly, the log files show different hosts than those observed by Erik Montcalm, who looked at logs from end of October 2003.[108] Some cleaning might have taken place. The list of new hosts that should be investigated is not small:

MY.NET.5.44	MY.NET.42.2	MY.NET.43.2
MY.NET.43.5	MY.NET.43.7	MY.NET.43.10
MY.NET.53.51	MY.NET.53.58	MY.NET.53.113
MY.NET.53.161	MY.NET.55.32	MY.NET.60.11
MY.NET.60.40	MY.NET.66.56	MY.NET.69.208
MY.NET.70.96	MY.NET.70.101	MY.NET.70.175
MY.NET.70.203	MY.NET.71.243	MY.NET.80.5
MY.NET.80.28	MY.NET.80.224	MY.NET.82.79
MY.NET.82.101	MY.NET.84.203	MY.NET.84.224
MY.NET.84.235	MY.NET.97.10	MY.NET.97.11
MY.NET.97.24	MY.NET.97.30	MY.NET.97.44
MY.NET.97.45	MY.NET.97.56	MY.NET.97.58
MY.NET.97.66	MY.NET.97.95	MY.NET.97.119
MY.NET.97.138	MY.NET.97.145	MY.NET.97.156
MY.NET.97.158	MY.NET.97.184	MY.NET.97.211
MY.NET.97.232	MY.NET.97.243	MY.NET.98.47
MY.NET.98.72	MY.NET.112.152	MY.NET.112.163
MY.NET.150.199	MY.NET.151.75	MY.NET.152.215
MY.NET.153.14	MY.NET.153.174	MY.NET.153.195

External RPC call

Two IPs are scanning parts of MY.NET for 111/tcp (portmap), causing all of the generated logs, some hosts are scanned multiple times (viewable from "Alerts" and "Scans"):

[106] Broadbandrports.com FAQ: http://www.dslreports.com/faq/4493
[107] TonikGin, XDCC – An .EDU Admin's Nightmare: http://www.ncsu.edu/it/security/papers/EduHacking.html
[108] Erik Montcalm GCIA Practical: http://www.giac.org/practical/GCIA/Erik_Montcalm_GCIA.pdf

```
+----------------+------------+----------+
| sourceip       | destip     | destport |
+----------------+------------+----------+
| 213.46.246.46  | MY.NET.5.5 |      111 |
| 217.160.94.163 | MY.NET.5.5 |      111 |
+----------------+------------+----------+
```

Again we see the whole MY.NET.190.0/24 being scanned, and a handful other addresses with a time gap in-between, but the gap is shorter than in the previous SubSeven scanning case. The portmap scan is performed much faster:

```
+------+-----------------+----------------+----------------+----------+
| day  | timestamp       | sourceip       | destip         | destport |
+------+-----------------+----------------+----------------+----------+
| 10   | 03:34:29.124548 | 217.160.94.163 | MY.NET.5.5     |      111 |
| 10   | 03:34:29.163663 | 217.160.94.163 | MY.NET.6.15    |      111 |
| 10   | 03:34:29.265375 | 217.160.94.163 | MY.NET.6.15    |      111 |
| 10   | 03:34:29.597424 | 217.160.94.163 | MY.NET.16.90   |      111 |
| 10   | 03:34:29.599332 | 217.160.94.163 | MY.NET.16.106  |      111 |
| 10   | 03:34:29.600360 | 217.160.94.163 | MY.NET.16.114  |      111 |
| 10   | 03:34:36.820267 | 217.160.94.163 | MY.NET.190.1   |      111 |
 ...          ...              ...              ...            ...
```

It should really be verified that Snort is seeing all the traffic it should!

As the name of the signature implies it seems just to looks for external sources, no internal source has been observed; in addition it is not clear if any hosts did respond, even by looking at "Scans" and "OOS".

SUNRPC highport access!

This alert again seems to be purely port-base, as in the example from the web[109]:

```
alert tcp any any -> $HOME_NET 32771 (msg: "SUNRPC highport access!";)
```

Most of the logs are clearly due to return traffic from well-known ports (Mail, Web, Instant Messenger, AOL). The only interesting IP is 68.55.193.50, which uses (source?) port 3671 (probably TCP) to connect to IP MY.NET.60.11. As the `whois` queries show (see Lookup section below), it belongs to Comcast Cable Communications, which might be a provider of UGCIA. In addition, the network name of "JUMPSTART-1" hints at a Unix network. No "Scan" or "OOS" or other "Alerts" is originating from this IP. So this RPC high port activity might be authorized and normal activity. This should be verified. In addition the port-based Snort signature should be tuned to look for SYN to this port, like:

```
alert tcp any any -> $HOME_NET 32771 (flags:S,12; msg: "SUNRPC highport access!"; sid:
1000100; rev: 2;)
```

TCP SRC and DST outside network

It is to assume that this signature looks for packets that have both source and destination not defined in $HOME_NET. In a perfect world Snort should not see

[109] Sample Snort rules document: http://cvs.sourceforge.net/viewcvs.py/snort/snort/Attic/RULES.SAMPLE?rev=1.7

any packets like these. We see some RFC 1918 addresses as sources like 192.168.1.22 connecting to Hotmail, 192.168.1.41 visiting lists.gnu.org and 192.168.123.195 browsing MSN. This seems to be pre-source-NAT traffic, ok. On the other hand we see returning traffic from IP 207.46.134.24, port 80 (a server from windowsupdate.microsoft.com) to IP 192.168.0.236. Here Snort sees post-destination-NAT return traffic, ok. But why do we not see both directions for each connection with this alert?

IP 192.168.0.52 is infected by Gaobot, which is indicated by its scanning for TCP ports 2745, 3127, 3410, 5000, 6129.[110] This host should be located and disinfected.

Should Snort see non-translated addresses? We do not know for sure, this depends on the exact location of the network tap. Looking at the "Scans" and "OOS" for other signs of 192.168.0.0/16 addresses we see nothing. So the answer might be "No", and there could be a NAT or tap location problem. This should be verified.

FTP passwd attempt

All alerts trigger for destination IP MY.NET.24.47, a host we have identified as a main FTP server before. For these 100 alerts there are 91 distinct sources. The signature name "FTP passwd attempt" is not used in newer Snort rule-sets, an equivalent may be SID 356:[111]

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP passwd retrieval attempt";
flow:to_server,established; content:"RETR"; nocase; content:"passwd";
reference:arachnids,213; classtype:suspicious-filename-detect; sid:356; rev:5;)
```

As it would be somehow unusual that in such a short time frame nearly one hundred different attackers try to download the /etc/passwd from the server, or access a possibly "chrooted" version of the file, I consider it very probable that there exist popular files on the server that contain the string "passwd" in the name. It is highly recommendable to verify this assumption and to adjust the signature or filename accordingly, as this would reduce the rate of false positives.

NIMDA - Attempt to execute cmd from campus host

This looks like a custom signature looking for outbound Nimda, i.e. infected hosts on MY.NET. We have just 15 of these "Nimda" alerts during 5 days - this is surely not enough for an infected host. What causes these false positives? Let's have a look at the entries:

day	timestamp	sourceip	destip	destport
08	05:29:42.925507	MY.NET.97.36	69.90.32.141	80
08	07:20:44.479101	MY.NET.10.79	64.70.33.115	80

[110] See for example: http://lists.sans.org/pipermail/unisog/2004-April/007158.php
[111] Snort signature 356: http://www.snort.org/snort-db/sid.html?sid=356

08	17:30:26.611199	MY.NET.97.228	69.90.32.141	80
08	17:30:56.244930	MY.NET.97.228	69.90.32.141	80
08	17:30:56.613572	MY.NET.97.228	69.90.32.141	80
08	23:16:33.750945	MY.NET.97.166	69.90.32.141	80
08	23:16:34.534091	MY.NET.97.166	69.90.32.141	80
09	16:55:44.014937	MY.NET.97.69	69.90.32.141	80
10	11:09:31.015313	MY.NET.97.180	216.64.193.20	80
10	19:53:18.715368	MY.NET.97.74	69.90.32.141	80
10	19:53:20.493171	MY.NET.97.74	69.90.32.141	80
10	21:05:02.383880	MY.NET.97.25	69.90.32.141	80
10	21:05:04.321864	MY.NET.97.25	69.90.32.141	80
11	12:45:53.297255	MY.NET.17.45	69.90.32.141	80
11	12:46:12.323507	MY.NET.17.45	69.90.32.141	80

The IP 69.90.32.141 does not resolve to a name by inverse queries; but after some research I found that "thinstall.abetterinternet.com" resolves to this IP 69.90.32.141. The hosts connecting to that IP should be investigated for possible infections by an adware or spyware called "Adware/Twaintech"[112] that is dropped by banner adds and is connecting "home" to deliver information about the infected host via XML and HTTP. It is probable that part of the information triggered the "Nimda" signature. From the Scans log file we find another affected host:

day	timestamp	sourceip	destip	destport
12	18:50:28	MY.NET.80.119	69.90.32.141	80
12	18:55:01	MY.NET.80.119	69.90.32.141	80

These hosts have also been connecting to the sister IP 69.90.32.140. According to the cited article from Fortinet you can disinfect a host by executing: `regsvr32 /u twaintec.dll`. Then you can manually delete the "twaintec.dll" from the Windows folder.

The IP 216.64.193.20 belongs to Cable & Wireless and is possibly related to Windows updates.[113] This seems to be a variant of likewise "calling home".[114] IP 64.70.33.115 belongs to Cable & Wireless, too - it might have a similar function.[115]

TFTP - Internal UDP connection to external tftp server

A closer look at the log files:

day	timestamp	sourceip	sourceport	destip	destport
08	05:33:11.116066	80.53.47.2	69	MY.NET.111.34	4672
08	05:33:11.116380	MY.NET.111.34	4672	80.53.47.2	69
09	15:44:58.240589	MY.NET.60.16	52786	128.183.103.201	69
09	20:59:26.747419	213.146.117.89	69	MY.NET.111.34	45559
10	01:37:35.397406	213.146.117.89	69	MY.NET.111.34	54645
10	03:31:43.692491	217.255.161.196	69	MY.NET.70.225	4672
10	03:31:43.693073	MY.NET.70.225	4672	217.255.161.196	69

[112] Fortinet Virus Encyclopedia: http://www.fortinet.com/VirusEncyclopedia/search/encyclopediaSearch.do?method=viewVirusDetailsInfoDirectly&fid=1089
[113] Posting from Broadband Forums: http://www.dslreports.com/forum/remark,10024358~mode=flat
[114] Posting on Plug list: http://mail.plug.linux.org.au/pipermail/plug/2004-May/052668.html
[115] Posting from Broadband Forums: http://www.dslreports.com/forum/remark,8434598~mode=flat

10	08:11:25.509626	80.138.24.157	69	MY.NET.70.225	4672
10	08:11:25.510239	MY.NET.70.225	4672	80.138.24.157	69
11	03:32:58.001721	MY.NET.70.225	4672	217.224.153.172	69
11	03:57:06.491882	217.80.240.43	69	MY.NET.70.225	4672
11	03:57:06.492637	MY.NET.70.225	4672	217.80.240.43	69
11	08:42:44.514239	217.224.153.172	69	MY.NET.70.225	4672
11	08:42:44.516064	MY.NET.70.225	4672	217.224.153.172	69

Port 4672/udp is used by Emule[116], this would coincide with detected Peer-to-peer file sharing activity on the network. Especially host MY.NET.70.225 seems to be heavily involved, which can be seen from the "Scans" file. The same holds true for host MY.NET.111.34. These hosts must be investigated and possibly cleaned from malware.

MY.NET.60.16, previously identified as possible Unix host accessible by SSH, is not part of P2P activity by judging from the logs. We see some `traceroutes` from there. It could be an admin machine. The machine connects to a NASA server (IP 128.183.103.201), see the Lookups section. This is probably no malicious activity.

TFTP - External TCP connection to internal tftp server

day	timestamp	sourceip	sourceport	destip	destport
10	13:14:17.698739	193.78.88.147	2312	MY.NET.97.10	69
10	13:14:41.689124	193.78.88.147	2312	MY.NET.97.10	69
11	05:39:49.593545	213.180.193.68	45101	MY.NET.60.38	69
11	05:39:49.593633	MY.NET.60.38	69	213.180.193.68	45101

This alert is somewhat strange. I did not find a reference on TFTP using TCP instead of the common UDP, RFC 1350 just talks about UDP[117]. I can only assume that this is a port-based signature looking for connections using port 69/tcp. It is highly probable that IP 213.180.193.68 scanned IP MY.NET.60.38 for port 69/tcp; this IP resolves to proxychecker.yandex.net (see Lookups section). DShield has at the time of this writing over 97.000 attack entries[118] from this IP; it is scanning - as its name implies - for a plethora of usual and unusual proxy ports.

Port 69/tcp is stated at Treachery Unlimited[119] as a possible back door port: BackGate trojan, which is explained by another source (that does not list 69/tcp) as a "A trojanized version of Wingate proxy server"[120], which might explain the inclusion of that port into a proxy scan. This fits into place with our previous identification of 213.180.193.68 as a possible mail server; the destination of a mail is scanning (or delegates the scan) for open proxies at the source.

[116] Emule project on ports: http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=122&rm=show_topic
[117] RFC 1350: http://www.networksorcery.com/enp/rfc/rfc1350.txt
[118] DShield entries: http://www.dshield.org/warning_explanation.php?source=213.180.193.068&d=d&start=0
[119] Treachery Unlimited: http://www.treachery.net/tools/ports/lookup.cgi
[120] Blackcode: http://www.blackcode.com/trojans/details.php?id=1469

Top Ten Talkers

Alerts

Top 10: Source IPs - Alerts

sourceip (from alerts)	count
212.76.225.24	7561
MY.NET.11.7	7016
MY.NET.84.235	3954
199.131.21.34	3480
68.81.0.87	2993
141.157.102.155	2693
MY.NET.60.16	2169
131.92.177.18	2166
68.57.90.146	1660
69.138.77.62	1628

Top 10: Destination IPs - Alerts

destip (from alerts)	count
MY.NET.30.3	12996
MY.NET.30.4	10207
MY.NET.43.3	7558
169.254.0.0	5722
82.48.242.184	3240
MY.NET.60.16	2693
141.157.102.155	2168
169.254.25.129	1848
MY.NET.84.235	1598
24.5.46.4	1249

Scans

Top 10: Source IPs - Scans

sourceip (from scans)	count
MY.NET.1.3	3978521
MY.NET.111.51	2147612
MY.NET.81.39	1715408
MY.NET.153.35	1522557
MY.NET.70.96	1188407
MY.NET.1.4	1102069
MY.NET.112.152	1082054
MY.NET.151.75	759754
MY.NET.84.235	468739
MY.NET.97.28	385518

Top 10: Destination IPs - Scans

destip (from scans)	count
69.6.57.4	146352
69.6.57.7	124796
69.6.57.9	124230
192.26.92.30	92988
192.48.79.30	74903
69.6.57.8	67228
69.6.57.10	66769
192.5.6.30	63439
128.194.254.5	56389
195.228.156.17	56140

OOS

Top 10: Source IPs - OOS

sourceip (from oos)	count
68.54.84.49	2044
202.144.28.167	700
141.224.64.4	265
193.170.194.27	216
66.225.198.20	200
62.174.236.17	171
80.54.249.136	164
80.38.206.68	116
MY.NET.199.202	114
68.121.194.43	101

Top 10: Destination IPs - OOS

destip (from oos)	count
MY.NET.6.7	2072
MY.NET.12.6	1406
MY.NET.70.225	917
MY.NET.12.4	376
MY.NET.24.44	240
MY.NET.110.82	215
MY.NET.43.3	152
MY.NET.42.10	116
MY.NET.5.67	109
MY.NET.111.34	73

Short Remarks: Outbound Scans

- MY.NET.1.3, the top talker, is a DNS and NTP server, therefore generating lots of false positives. Likewise DNS traffic with MY.NET.1.4.
- MY.NET.111.51 is most probably infected by a MS RPC Worm, scanning outbound for port 135/tcp; it might be a Blaster variant.
- MY.NET.81.39 seems likewise infected by an MS RPC worm.
- MY.NET.153.35 has a lot of high port to high port traffic, perhaps multimedia or IP telephony (port 3247/udp for example used in multimedia).
- MY.NET.70.96 seems to be infected by a Gaobot variant, it is scanning outbound for 135, 445, 1025, 2745, 3127, 5000, 6129.
- MY.NET.112.152 seems likewise infected by Gaobot.
- MY.NET.151.75, again Gaobot.
- MY.NET.84.235 seems to be involved in heavy P2P activity; there are many connections using port 4662/tcp (Edonkey).
- MY.NET.97.28 is scanning massively for HTTP, suspiciously increasing the first octet of the destination IP. This host should be checked for infection by an HTTP using worm.

Short Remarks: "Outbound" OOS

At the TOP 10 OOS we only see MY.NET.199.202 as a source. All 114 tracked connections are internally, to the HTTPS (webauth) server MY.NET.12.7, HTTP to MY.NET.24.34 and a handful other non-malicious looking connections. I suppose the OOS characterization comes from the SYN together with the two set reserved or ECN bits, which can be found at every logged packet here and in many other OOS logs. Though the timestamps look a bit unusual for retransmission, the whole picture does not look like scanning activity.

Lookups

For the queries I used `nslookup`, whois, Sam Spade, Geektools[121] and the DShield web site[122].

[121] Geektools: http://www.geektools.com/whois.php
[122] DShield: http://www.dshield.org/

IP 141.157.102.155 ("High port 65535 tcp - possible Red Worm - traffic")

```
nslookup 141.157.102.155
Canonical name: pool-141-157-102-155.balt.east.verizon.net
Addresses:141.157.102.155

whois -h whois.arin.net 141.157.102.155  ...
Verizon Internet Services VIS-141-149 (NET-141-149-0-0-1) 141.149.0.0 - 141.158.255.255
Verizon   Internet   Services   VZ-DSLDIAL-CYVLMD-9   (NET-141-157-57-0-1)141.157.57.0   -
141.157.126.255
CustName:   Verizon Internet Services
Address:    1880 Campus Commons Drive
City:       Reston
StateProv:  VA
PostalCode: 20191
Country:    US
RegDate:    2002-03-21
Updated:    2002-03-21

whois -h whois.arin.net !net-141-157-57-0-1 ...
NetRange:   141.157.57.0 - 141.157.126.255
CIDR:              141.157.57.0/24,  141.157.58.0/23,  141.157.60.0/22,  141.157.64.0/19,
141.157.96.0/20, 141.157.112.0/21, 141.157.120.0/22, 141.157.124.0/23, 141.157.126.0/24
NetName:    VZ-DSLDIAL-CYVLMD-9
NetHandle:  NET-141-157-57-0-1
Parent:     NET-141-149-0-0-1
NetType:    Reassigned
Comment:
RegDate:    2002-03-21
Updated:    2002-03-21
TechHandle: ZV20-ARIN
TechName:   Verizon Internet Services
TechPhone:  +1-703-295-4583
TechEmail:  noc@gnilink.net
OrgAbuseHandle: VISAB-ARIN
OrgAbuseName:   VIS Abuse
OrgAbusePhone:  +1-703-295-4583
OrgAbuseEmail:  abuse@verizon.net
OrgTechHandle: ZV20-ARIN
OrgTechName:   Verizon Internet Services
OrgTechPhone:  +1-703-295-4583
OrgTechEmail:  noc@gnilink.net
```

IP 213.189.89.54 / .109 ("Possible trojan server activity ")

```
nslookup 213.189.89.54
Canonical name: tahani.qualitynet.net
Addresses: 213.189.89.54

nslookup 213.189.89.109
Canonical name: nmc1.qualitynet.net
Addresses: 213.189.89.109

inetnum:      213.189.89.0 - 213.189.89.255
netname:      STAFF-NET
descr:        STAFF SEGMENT
country:      KW
admin-c:      QNET1-RIPE
tech-c:       AA581-RIPE
status:       ASSIGNED PA
notify:       admin-c@qualitynet.net
mnt-by:       QNET-NOC
changed:      admin-c@qualitynet.net 20030611
source:       RIPE

route:        213.189.64.0/19
descr:        QualityNet Kwait
origin:       AS9155
member-of:    RS-QNET
mnt-by:       QNET-NOC
changed:      hia@qualitynet.net 20000401
source:       RIPE
person:       Qnet Admin Contact
address:      Kuwait
phone:        +965 80 8888
```

```
e-mail:       admin-c@qualitynet.net
nic-hdl:      QNET1-RIPE
notify:       stinger@qualitynet.net
mnt-by:       MOC-MNT
changed:      stinger@qualitynet.net 20030611
source:       RIPE
person:       Abdulaziz Al-osaimi
address:      Ministry of Communications
address:      Po box 318 Safat, 1111 Kuwait
phone:        +965 481 1036
```

IP 68.55.193.50 ("SUNRPC highport access!")

```
IP Address: 68.55.193.50
HostName: pcp229495pcs.catonv01.md.comcast.net
DShield Profile: Country:   US
Contact E-mail: abuse@comcastpc.com
AS Number: 22909
Whois:
CustName:   Comcast Cable Communications, Inc.
Address:    3 Executive Campus
Address:    5th Floor
City:       Cherry Hill
StateProv:  NJ
PostalCode: 08002
Country:    US
RegDate:    2003-03-19
Updated:    2004-07-02
NetRange:   68.55.0.0 - 68.55.255.255
CIDR:       68.55.0.0/16
NetName:    BALTIMORE-A-6
NetHandle:  NET-68-55-0-0-1
Parent:     NET-68-32-0-0-1
OrgName:    Comcast Cable Communications, Inc.
OrgID:      CMCS
Address:    1800 Bishops Gate Blvd
City:       Mt Laurel
StateProv:  NJ
PostalCode: 08054
Country:    US
NetRange:   68.32.0.0 - 68.63.255.255
CIDR:       68.32.0.0/11
NetName:    JUMPSTART-1
NetHandle:  NET-68-32-0-0-1
Parent:     NET-68-0-0-0-0
NetType:    Direct Allocation
NameServer: DNS01.JDC01.PA.COMCAST.NET
NameServer: DNS02.JDC01.PA.COMCAST.NET
Comment:    ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
RegDate:    2001-11-29
Updated:    2003-11-05
TechHandle: IC161-ARIN
TechName:   Comcast Cable Communications Inc
TechPhone:  +1-856-317-7200
TechEmail:  cips_ip-registration@cable.comcast.com
OrgAbuseHandle: NAPO-ARIN
OrgAbuseName:   Network Abuse and Policy Observance
OrgAbusePhone:  +1-856-317-7272
OrgAbuseEmail:  abuse@comcast.net
OrgTechHandle: IC161-ARIN
OrgTechName:  Comcast Cable Communications Inc
OrgTechPhone:  +1-856-317-7200
OrgTechEmail: cips_ip-registration@cable.comcast.com
```

IP 128.183.103.201 ("TFTP - Internal UDP connection to external tftp server")

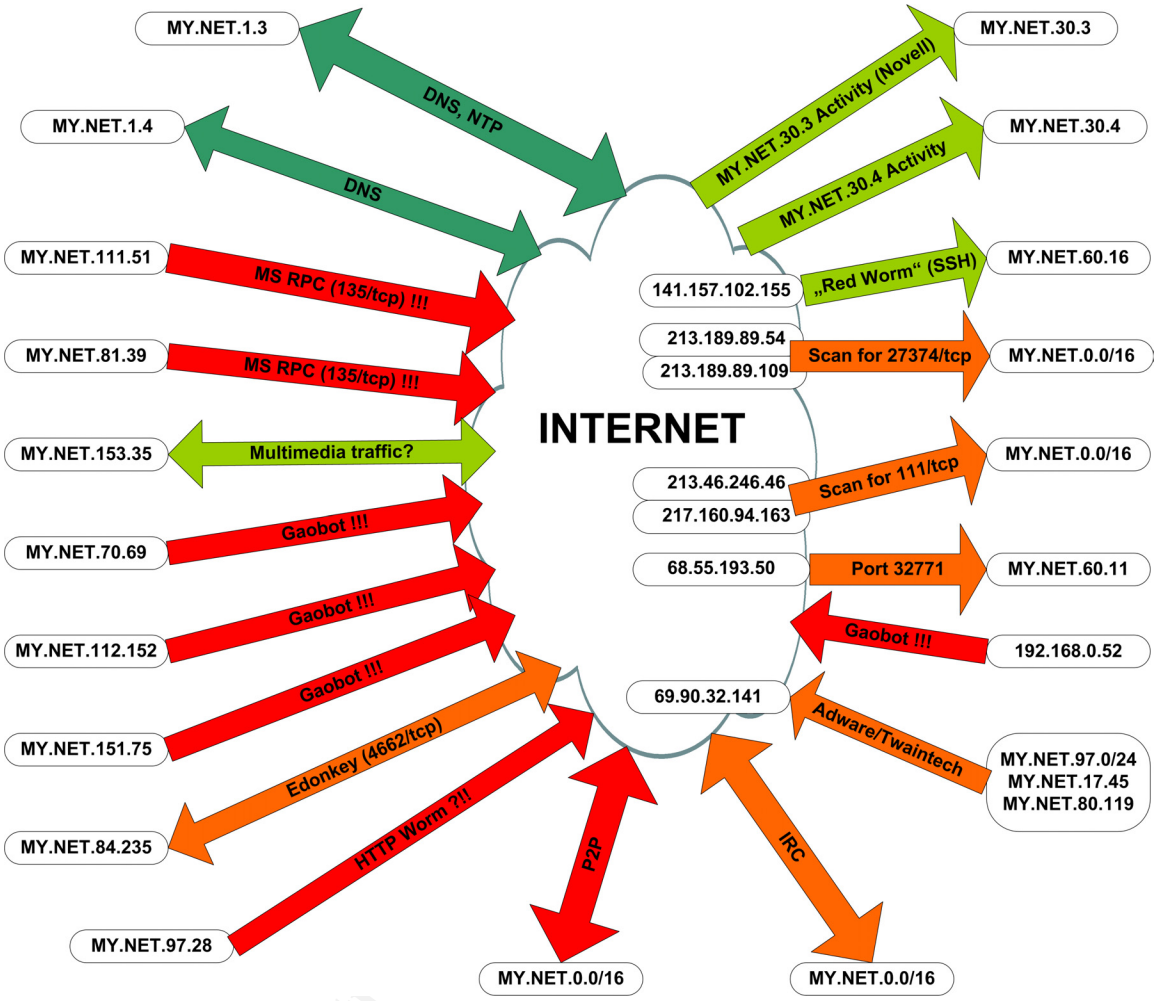
```
nslookup 128.183.103.201
Canonical name: neptune.gsfc.nasa.gov
Addresses: 128.183.103.201

OrgName:    National Aeronautics and Space Administration
OrgID:      NASA
Address:    AD33/Office of the Chief Information Officer
City:       MSFC
StateProv:  AL
PostalCode: 35812
Country:    US
NetRange:   128.183.0.0 - 128.183.255.255
```

```
CIDR:       128.183.0.0/16
NetName:    GSFC
NetHandle:  NET-128-183-0-0-1
Parent:     NET-128-0-0-0-0
NetType:    Direct Allocation
NameServer: NS.GSFC.NASA.GOV
NameServer: NS2.GSFC.NASA.GOV
RegDate:    1993-04-01
Updated:    2003-02-05
TechHandle: ZN7-ARIN
TechName:   National Aeronautics and Space Administration
TechPhone:  +1-256-544-5623
TechEmail:  dns.support@nasa.gov
OrgAbuseHandle: NASAA-ARIN
OrgAbuseName:   NASA Abuse
OrgAbusePhone:  +1-800-762-7472
OrgAbuseEmail:  abuse@nasa.gov
OrgNOCHandle: NISN-ARIN
OrgNOCName:   NASA Information Services Network
OrgNOCPhone:  +1-256-961-4000
OrgNOCEmail:  noc@nisn.nasa.gov
OrgTechHandle: WEBBN-ARIN
OrgTechName:   Webb, Nancy
OrgTechPhone:  +1-256-544-3245
OrgTechEmail:  dns.support@nasa.gov
```

IP 213.180.193.68 ("TFTP - External TCP connection to internal tftp server")

```
nslookup 213.180.193.68
Canonical name: proxychecker.yandex.net
Addresses: 213.180.193.68

inetnum:      213.180.192.0 - 213.180.193.255
netname:      COMPTEK-NET1
descr:        CompTek International/Yandex LLC
descr:        3, Gubkina str., Moscow, 117809
country:      RU
admin-c:      YNDX1-RIPE
tech-c:       YNDX1-RIPE
status:       ASSIGNED PA
notify:       noc@yandex.net
mnt-by:       YANDEX-MNT
changed:      wawa@comptek.ru 20020607
changed:      gvs@yandex-team.ru 20040625
source:       RIPE
route:        213.180.192.0/20
descr:        Yandex enterprise network
origin:       AS13238
notify:       noc@yandex.net
mnt-by:       YANDEX-MNT
changed:      wawa@comptek.ru 20010123
changed:      gvs@yandex-team.ru 20040625
source:       RIPE
role:         Yandex LLC Network Operations
address:      Yandex LLC
address:      40A Vavilova st.
address:      117333, Moscow, Russia
phone:        +7 095 9743555
fax-no:       +7 095 9743565
e-mail:       noc@yandex.net
trouble:      ------------------------------------------------------
trouble:      Points of contact for Yandex LLC Network Operations
trouble:      ------------------------------------------------------
trouble:      Routing and peering issues:  noc@yandex.net
trouble:      SPAM issues:                 abuse@yandex.ru
trouble:      Network security issues:     abuse@yandex.ru
trouble:      Mail issues:                 postmaster@yandex.ru
trouble:      General information:         info@yandex.ru
trouble:      ------------------------------------------------------
admin-c:      VLI1-RIPE
admin-c:      GVS-RIPE
tech-c:       KBG2-RIPE
notify:       noc@yandex.net
nic-hdl:      YNDX1-RIPE
mnt-by:       YANDEX-MNT
changed:      gvs@yandex-team.ru 20040625
source:       RIPE
```

Link Graph



Short Remarks on the Link Graph

The center of the graph is the Internet, with some of the IP addresses identified in the sections above. Around the center you have IP addresses from MY.NET (or presumed to be from there, as in the case of RFC 1918 IP 192.160.0.52, which we saw as Gaobot infected). The arrow pointers indicate the main direction of detected traffic (Stimulus), nothing is stated here about possible return traffic (Response). Severity of alerts is coded by the color of the traffic arrows:

Red: high severity - investigate immediately!

Orange: misuse or potential malicious activity - might need quick investigation and defense optimization.

Bright Green: probably non-malicious activity. If time allows, this should be controlled.

Dark Green: highly probable that this is normal, non-malicious traffic.

Analysis Methodology

The main goal of preparing the log files was getting a flexible method to look for almost every entry and detail. The solution was to set up a database and find ways to transfer the downloaded log files into it. In this case my choice was MySQL, which I already had installed on my hosted server (Intel CPU 2.00 GHz) that is running Debian. I created a new database called "giac" with tables for each log type:

```
+----------------+
| Tables_in_giac |
+----------------+
| alerts         |
| oos            |
| scans          |
+----------------+
```

The single log files got first concatenated into a main file for each type. Then long cleaning procedures using `sed` started to get rid of corrupt entries (mainly from the corrupt files, but not only), fix missing line breaks and remove other abundant breaks. In some places the name of the University needed to be changed to UGCIA (watch the signature names), and some IP addresses had to be obfuscated by replacing the first two octets to "MY.NET".

A very useful script for "Alerts" and "Scans" is `csv.pl` from Tod Beardsley[123], which I used with a small modification, that just doesn't write the output type ("alert") to the CSV file. This Perl script was also a nice base for my own experiments, which resulted in `oos2csv.pl` to parse the OOS files into CSV format (see Appendix). A good inspiration for this was Ricky Smith's `parse-oos.pl`[124], also.

The database needs tables, and these need a column structure corresponding to the sequence of fields in the CSV file. Very useful was the SQL script from Les Gordon[125], which I adapted to my OOS table. Via the "`load data local infile`" directive MySQL reads the CSV files and transforms it into rows. Now the whole power and flexibility of the SQL command line lie at your fingertips.

[123] Tod Beardsley, GCIA Practical, p. 61: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
[124] Ricky Smith, GCIA Pratical, p. 64: http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf
[125] Les Gordon, GCIA Practical, p. 73: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc

Appendices

oos2csv.pl

```
#!/usr/bin/perl

# oos2csv.pl

# Function: Parse GIAC GCIA OOS logs into CSV format for further processing.
# Syntax ./oos2csv.pl ooslogfile [outputfile (default: `ooslogfile`.csv]
# Version: 0.5
# Date: 07-30-2004
# Author: Ben Fabian
# Inspired by scripts from: Todd Beardsly, Rick Smith

# Output format: line of comma separated values.
# $month, $day, $timestamp, $sourceip, $sourceport, $destip, $destport,
#  $protocol, $ttl, $tos, $id, $iplen, $dgmlen, $fragbit, $tcpflags, $seq, $ack, $win,
$tcplen, $tcpoptions, data

$NULL = '\N'; # NULL value for import to mysql

unless ($ARGV[0]) {die "Please supply an input file!\n";}
unless ($ARGV[1]) {$outfile = "$ARGV[0].csv";}
        else {$outfile = "$ARGV[1]";}

open(INFILE,"$ARGV[0]") || die "I am unable to open $ARGV[0] for reading!\n";
open(OUTFILE,">$outfile") || die "I am unable to open $ARGV[1] for writing!\n";
$count = 1;

while (<INFILE>) {

        # Ignore Whitespace:
        next if (/^(\s)+$/);
# Ignore lines consisting of "=+=+":
        next if ((/^(\=\+)+$/) and ($count == 1));
        # If we have reached the end of one log entry, output action:
        if (/^(\=\+)+$/) {
                for ($i = 1; $i < $count; $i++) {
                        $result = join ("," , @step[$i]);

                        print OUTFILE $result;
                        @step[i] = "";
                        $result = "";
                }
                print OUTFILE $NULL.",".$NULL."\n";
                $count = 1;
                next;
        }
        # Remove the EOL character from input line:
        chomp($_);
        # A fresh record:
        if ($count == 1) {
                $month = $day = $sourceip = $sourceport = $destip = $destport = $NULL;
                @line = split (/ /);
                $time = @line[0];
                ($month, $rest1) = split (/\//, $time);
                ($day, $timestamp) = split (/-/, $rest1);
                ($sourceip, $sourceport) = split(/:/, @line[1]);
                ($destip, $destport) = split(/:/, @line[3]);
                @prestep      =      ($month,      $day,      $timestamp,      $sourceip,
$sourceport,$destip,$destport);
                @step[$count] = join ("," , @prestep , "");
                @prestep = "";
                $count++;
                next;
        }

        # A second line from a given record. The structure of entries has changed, watch
additional spaces!
        if ($count == 2) {
                $protocol = $ttl = $tos = $id = $iplen = $dgmlen = $fragbit = $NULL;
                @line = split (/ /);
                $protocol = @line[0];
                ($foo, $ttl) = split(/:/, @line[1]);
                ($foo, $tos) = split(/:/, @line[2]);
```

```
                ($foo, $id) = split(/:/, @line[3]);
                ($foo, $iplen) = split(/:/, @line[4]);
                ($foo, $dgmlen) = split(/:/, @line[5]);
                $fragbit = @line[6];
                @prestep = ($protocol, $ttl, $tos, $id, $iplen, $dgmlen, $fragbit);
                @step[$count] = join ("," , @prestep , "");
                @prestep = "";
                $count++;
                next;
        }

        # A third line from a given record. Watch the spaces!
        if ($count == 3) {
                $tcpflags = $seq = $ack = $win = $tcplen = $NULL;
                @line = split (/ /);
                $tcpflags = @line[0];
                $seq = @line[2];
                $ack = @line[5];
                $win = @line[8];
                $tcplen = @line[11];
                @prestep = ($tcpflags, $seq, $ack, $win, $tcplen);
                @step[$count] = join ("," , @prestep , "");
                @prestep = "";
                $count++;
                next;
        }

        if ($count == 4) {
                # Do we have TCP options?
                $tcpoptions = $NULL;
                if (/^TCP Options/) {
                        ($foo, $tcpoptions) = split (/=> /);
                        @prestep = ($tcpoptions);
                        @step[$count] = join ("," , @prestep , "");
                        @prestep = "";
                }
                # Or already data, replace CSV delimiter to avoid possible conflicts:
                else {
                        s/,/COMMA/g;
                        @step[$count] = $NULL.",".$_;
                        $count++;
                }
                $count++;
                next;
        }

        else {
                # Data, replace CSV delimiter to avoid possible conflicts:
                s/,/COMMA/g;
                @step[$count] = $_;
                $count++;
                next;
                }

}
```

References

Please note that all links given throughout this document have been valid in August 2004.

1. W. Pooh: "Pooh's 101 Uses for a Honey Pot", Entry No. 1, New York 1997.
2. (-)
3. Alfredo Lopez, GCFW Practical describing the company: http://www.giac.org/practical/GCFW/Alfredo_Lopez_GCFW.pdf
4. Symantec Incident Manager: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=166&EID=0
5. Sun Fire V440 Server: http://www.sun.com/servers/entry/v440/index.xml
6. Sun Fire V250 Server: http://www.sun.com/servers/entry/v250/index.xml
7. Cisco 7200 Routers: http://www.cisco.com/univercd/cc/td/doc/pcat/7200.htm
8. How to Use HSRP to Provide Redundancy in a Multihomed BGP Network: http://www.cisco.com/warp/public/459/hsrp_bgp.pdf
9. Cisco PIX Firewall: http://www.cisco.com/go/pix
10. Cisco PIX 535: http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/prodlit/535_ds.pdf
11. Symantec SGS 5400: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=133&EID=0
12. Symantec ManHunt: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=156&EID=0
13. NSS Gigabit IDS test: http://www.nss.co.uk/gigabitids/edition2/index.htm
14. "Symantec ManHunt SESA Bridge Installation Guide"
15. Symantec iForce page: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=284&EID=0
16. Sun iForce page: http://www.sun.com/aboutsun/media/presskits/symantec_iforce_ids/
17. iButton technology: http://www.ibutton.com/ibuttons/index.html
18. iForce FAQ, models: http://enterprisesecurity.symantec.com/content/displaypdf.cfm?pdfid=625
19. "MSA for Snort 2.0 Installation Guide"
20. Sun Fire V120 Server: http://www.sun.com/servers/entry/v120/
21. Snort Documentation: http://www.snort.org/docs/snort_manual/node9.html
22. A. Baker, B. Caswell, M. Poor: Snort 2.1 - Intrusion Detection. Syngress, Rockland, 2004.
23. Symantec Host IDS: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=48&EID=0
24. Symantec Host IDS 4.1 Courseware, p. 232.
25. Honeyd virtual Honey pot: http://www.honeyd.org/
26. John Lyons, "Honeypots & deploying Honeyd": http://www.sage-ie.org/slides/honeypots.pdf
27. Symantec Decoy Server: http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=157&EID=0
28. Sun Fire V250 Server: http://www.sun.com/servers/entry/v250/index.xml
29. NTP resource page: http://www.ntp.org; List of public NTP servers: http://www.eecis.udel.edu/~mills/ntp/servers.html
30. SSH tool "scponly": http://www.sublimation.org/scponly/
31. Symantec Security Response, W32.Witty.Worm: http://securityresponse.symantec.com/avcenter/venc/data/w32.witty.worm.html
32. Apple G5 Xserve: http://www.apple.com/xserve/
33. Apple Xserve RAID: http://www.apple.com/xserve/raid/
34. Kano XSPAND: http://www.kanotechnologies.com/prod/x_spand.cfm
35. US Department of Homeland Security, Incident Definition: http://www.fedcirc.gov/incidentReporting/incidentDefinition.html
36. Stephen Northcutt: Computer Security Incident Handling, SANS Press, 2003.
37. Cf. Earl Carter: Cisco Secure Intrusion Detection System (CSIDS), Cisco Press, Indianapolis, 2004; pages 135-157.
38. Cisco Catalyst 3508G XL Switch: http://www.cisco.com/en/US/products/hw/switches/ps637/ps639/index.html
39. Sun Bigadmin - Implementing a stealth Ethernet interface: http://www.sun.com/bigadmin/content/submitted/stealth_ethernet.html
40. Sun Product Documentation - "ge" driver: http://docs.sun.com/db/doc/806-7745-10/6jgjlqs7a?q=GigabitEthernet&a=view
41. Raw Logs on Incidents.org: http://www.incidents.org/logs/raw/2002.5.10
42. IANA Protocol Numbers and Assignment Services: http://www.iana.org/numbers.html
43. IANA Ethernet Numbers: http://www.iana.org/assignments/ethernet-numbers
44. Vendor/Ethernet MAC Address Lookup and Search: http://www.coffer.com/mac_find/
45. SnortAlog Homepage: http://jeremy.chartier.free.fr/snortalog/
46. Nmap network security scanner man page: http://www.insecure.org/nmap/data/nmap_manpage.html
47. Firewall incidents at the FinchHaven datacenter: http://www.finchhaven.com/pages/incidents/031202_tcp_123.html
48. Fyodor on Snort-users mailing list, Aug 11, 2000: http://archives.neohapsis.com/archives/snort/2000-08/0152.html
49. Nmap network security scanner man page: http://www.insecure.org/nmap/data/nmap_manpage.html
50. RFC 793, Transmission Control Protocol, Sep 1981: ftp://ftp.rfc-editor.org/in-notes/rfc793.txt
51. p0f Homepage: http://lcamtuf.coredump.cx/p0f.shtml
52. F5 3-DNS Controller: http://www.f5.com/f5products/3dns/
53. Radware LinkProof: http://www.radware.com/content/products/lp/default.asp
54. LinkProof - White Paper: http://www.radware.com/content/products/lp/whtpaper/default.asp?_v=about&document=1316
55. DShield Homepage: http://www.dshield.org/
56. Firewall incidents at the FinchHaven datacenter: http://www.finchhaven.com/pages/incidents/031202_tcp_123.html
57. Pedro Bueno on Intrusions list, Apr 8, 2002: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00110.html
58. Raw Logs on Incidents.org: http://www.incidents.org/logs/RAW/2002.9.17
59. Antony Gummery on Intrusions list, Mar 22, 2003: http://www.dshield.org/pipermail/intrusions/2003-March/007239.php
60. Chris Brenton on Intrusions list, Apr 26, 2002: http://cert.uni-stuttgart.de/archive/intrusions/2002/04/msg00318.html
61. Generalitat Valenciana portal site: http://www.gva.es
62. CERT search page: http://search.cert.org/
63. p0f homepage: http://lcamtuf.coredump.cx/p0f.shtml

64. Project Honeynet Fingerprints: http://project.honeynet.org/papers/finger/traces.txt
65. DShield homepage: http://www.dshield.org
66. myNetWatchman homepage: http://www.mynetwatchman.com
67. myNetWatchman incident - archived now (was: http://www.mynetwatchman.com/LID.asp?IID=89601318)
68. myNetWatchman incident - archived now (was: http://www.mynetwatchman.com/LID.asp?IID=87377146)
69. David Dittrich, Stacheldraht Analysis: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis
70. Sys-Security Group, Identifying ICMP Hackery Tools: http://www.sys-security.com/archive/securityfocus/icmptools.html
71. S.I.N.N. - Sinn Is Not Naphta: http://archives.neohapsis.com/archives/vuln-dev/2000-q4/0663.html
72. Naphta Dos Vulnerability: http://www.bindview.com/Support/RAZOR/Advisories/2000/adv_NAPTHA.cfm
73. Twoface DDOS: http://www.darkaxis.com/devel/c/security/exploit/ddos/twoface.c
74. Phrack, Linux on-the-fly kernel patching without LKM: http://www.phrack.org/show.php?p=58&a=7
75. Synscan homepage: http://synscan.sourceforge.net/
76. Arachnids on Synscan: http://www.digitaltrust.it/arachnids/IDS441/event.html
77. Donald Smith on Intrusions, Nov 4 2002: http://www.dshield.org/pipermail/intrusions/2002-November/005813.php
78. "<krist>" posting on Whitehats forum, Sep 21 2003: http://whitehats.com/cgi/forum/messages.cgi?bbs=get_topic&f=9&t=000075
79. Kurt Anderson on Intrusions list, Dec 22, 2003: http://cert.uni-stuttgart.de/archive/intrusions/2003/12/msg00132.html
80. Richard Bejtlich, Network Intrusion Detection of Third Party Effects: http://home.satx.rr.com/bejtlich/nid_3pe_v101.pdf
81. Symantec Security Response, Blaster Worm: http://securityresponse.symantec.com/avcenter/venc/data/pf/w32.blaster.worm.html
82. Microsoft Security Bulletin MS03-026: http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx
83. Microsoft "What You Should Know About the Blaster Worm": http://www.microsoft.com/security/incident/blast.asp
84. Microsoft PSS Security Response Team Alert: http://www.microsoft.com/technet/security/alerts/msblaster.mspx
85. eEye Blaster Worm Advisory: http://www.eeye.com/html/Research/Advisories/AL20030811.html
86. eEye Blaster Worm Analysis: http://www.eeye.com/html/Research/Advisories/Blaster_Analysis.txt
87. Dan Hanson on Incidents list, Oct 28, 2003: http://www.securityfocus.com/archive/75/342726/2003-10-26/2003-11-01/0
88. Jim Harrison on microsoft.public.isaserver, Sep 10, 2003: http://groups.google.com/groups?selm=2qn7b.5299%24F_4.49860809%40news-text.cableinet.net
89. RFC 2827, Ingress Filtering for Multihomed Networks: ftp://ftp.rfc-editor.org/in-notes/rfc2827.txt
90. RFC 3704: ftp://ftp.rfc-editor.org/in-notes/rfc3704.txt
91. SANS, "Help Defeat Denial of Service Attacks": http://www.sans.org/dosstep/
92. SANS GIAC logs: http://isc.sans.org/logs/
93. Entry on NCP from protocols.com: http://www.protocols.com/pbook/novel.htm#NCP
94. Entry on NCP at Networksorcery: http://www.networksorcery.com/enp/protocol/ncp.htm
95. Novell: "Apache/Tomcat Environment on Novell as a Java development environment on Novell Netware 6": http://www.novell.com/info/collateral/docs/4621204.01/4621204.pdf
96. Symantec Security Response on Gaobot.AFW (example): http://www.symantec.com/avcenter/venc/data/w32.gaobot.afw.html
97. IBM iSeries Information Center, TCP/IP communication support concepts for DDM : http://publib.boulder.ibm.com/infocenter/iseries/v5r3/ic2924/index.htm?info/ddp/rbal1sockcon.htm
98. Michael Walsh: "Some of the Dangers of Connecting your AS/400 to a Network2; SANS Reading Room: http://www.sans.org/rr/papers/56/308.pdf
99. NetWare 6.0 Support Pack 3 - TID2965459: http://support.novell.com/cgi-bin/search/searchtid.cgi?/2965459.htm
100. Anthony Dell, GSEC practical: http://www.giac.org/practical/gsec/Anthony_Dell_GSEC.pdf
101. SANS on Adore Worm: http://www.sans.org/y2k/adore.htm
102. Sophos on Adore Worms: http://www.sophos.com/virusinfo/analyses/linuxadore.html
103. LinkLogger on SubSeven : http://www.linklogger.com/TCP27374.htm
104. Perry Lorier's IRC rules: http://coders.meta.net.nz/~perry/irc.rules
105. Christopher Cramer on unisog list: http://lists.sans.org/pipermail/unisog/2002-May/001485.php
106. Broadbandrports.com FAQ: http://www.dslreports.com/faq/4493
107. TonikGin, XDCC – An .EDU Admin's Nightmare: http://www.ncsu.edu/it/security/papers/EduHacking.html
108. Erik Montcalm GCIA Practical: http://www.giac.org/practical/GCIA/Erik_Montcalm_GCIA.pdf
109. Sample Snort rules document: http://cvs.sourceforge.net/viewcvs.py/snort/snort/Attic/RULES.SAMPLE?rev=1.7
110. Brian Eckman on unisog: http://lists.sans.org/pipermail/unisog/2004-April/007158.php
111. Snort signature 356: http://www.snort.org/snort-db/sid.html?sid=356
112. Fortinet Virus Encyclopedia: http://www.fortinet.com/VirusEncyclopedia/search/encyclopediaSearch.do?method=viewVirusDetailsInfoDirectly&fid=1089
113. Posting from Broadband Forums: http://www.dslreports.com/forum/remark,10024358~mode=flat
114. Posting on Plug list: http://mail.plug.linux.org.au/pipermail/plug/2004-May/052668.html
115. Posting from Broadband Forums: http://www.dslreports.com/forum/remark,8434598~mode=flat
116. Emule project on ports: http://www.emule-project.net/home/perl/help.cgi?l=1&topic_id=122&rm=show_topic
117. RFC 1350: http://www.networksorcery.com/enp/rfc/rfc1350.txt
118. DShield entries: http://www.dshield.org/warning_explanation.php?source=213.180.193.068&d=d&start=0
119. Treachery Unlimited: http://www.treachery.net/tools/ports/lookup.cgi
120. Blackcode: http://www.blackcode.com/trojans/details.php?id=1469
121. Geektools: http://www.geektools.com/whois.php
122. DShield: http://www.dshield.org/
123. Tod Beardsley, GCIA Practical, p. 61: http://www.giac.org/practical/Tod_Beardsley_GCIA.doc
124. Ricky Smith, GCIA Pratical, p. 64: http://www.giac.org/practical/GCIA/Ricky_Smith_GCIA.pdf
125. Les Gordon, GCIA Practical, p. 73: http://www.giac.org/practical/GCIA/Les_Gordon_GCIA.doc