CISCO IOS/IOS XE Risk Mitigation

Version 1.5 – October 2014

1 Introduction

Following a risk assessment with respect to possible compromises in the network infrastructure of its constituents, CERT-EU has documented best practices to mitigate risks against the CISCO IOS and IOS XE operating systems. Among others, this involves a centralized logging facility aiming to monitor specific execution and configuration commands on a CISCO device. Also, alerts to administrators can be raised not only from the SIEM, but from the device itself.

2 Cisco IOS/IOS XE Risks

There are two major risks against the Cisco IOS devices. The first involves an IOS image that could potentially be modified offline by an attacker in order to operate in a malicious manner. The second involves executing arbitrary code during runtime. In addition the IOS XE inherits security threats that are derived from the underlying linux based operating system.

2.1 Low level rootkit

In [5] authors describe a procedure that will produce a compromised IOS image. This procedure involves an IOS image unpack process, the malware injection process into the unpacked image, the process of repacking and final the delivery of the compromised image to the target device. The latter requires privileged access to target devices as well as rebooting the device.

Of course this kind of procedure may take place much easier by intervening in the supply-chain of the manufacturer. Because of the large diversity of IOS images that are developed only for a specific hardware platform, it is not expected that such an approach will lead in a massive threat against a CISCO network infrastructure. However, it is possible for a malicious user to design an attack against a specific organization.

2.2 High level rootkit

2.2.1 Gnu Debugger (GDB) [6]

GDB is an embedded GNU debugger that is present inside every Cisco networking device (switch, router). GDB is used by Cisco developers for online debugging of the operation of the device.
There are three modes of GDB operation that are activated only by **privileged** users from the command line interface (CLI):

- *gdb examine pid*: which gives the ability to inspect memory and CPU registers (read only)
- *gdb debug pid* : which gives the ability to remotely (via telnet!!!) modify memory and CPU registers (read/write), while the system still runs. The latter may potentially lead in infection during runtime.
- *gdb kernel* : which is used by the developers when serial access is available. This mode freezes the system.

Although GDB is not useful during normal device operation and it appears to be a serious security risk, it cannot be disabled.

2.2.2 Tool Command Language (Tcl)

The tcl (tcl shell) support provides scripting functionality for IOS devices. Tclsh is enabled to accounts with privilege level 15. However, backdoors have been developed with tclsh [7].

2.2.3 IOS XE additional risk

IOS XE runs as a daemon, named *iosd*, on a linux based operating system. The adversary can potentially gain privileged access to the system and install a unix based rootkit.

3 Detection of Compromises

Detection method relies on the fact that specific region of the memory of a Cisco network device should be marked as RO (read-only) as it contains the instructions to be executed. These instructions should not be overwritten. This specific memory area is called as *text memory* area.

There are two main methods to check the integrity of code running an IOS/IOS XE image, and both require a memory dump of the device. Memory dump is produced with a built-in command of the IOS, which implies trust in the memory-dumping process, which may itself be compromised. Described procedures do not apply to Cisco's XR, NX-OS and PIX-OS operating systems.

3.1 Method with two memory dumps [9, 10]

First, a memory dump file of a possible infected router needs to be obtained. Then, the hash of the *text memory area* region of the memory dump file needs to be computed.

Then, it is necessary to load the **same** IOS / IOS XE version from a known-good image to the **same** router platform, and repeat again the process.

Integrity of code executed is verified by comparing the hashes of the two text memory area extracts.

However, it appears that this procedure is applicable only to some versions of the IOS (12.x and 15.x family) that have not implemented the ASLR technique. It is also a procedure that only detects the code integrity being executed in memory. No other information is revealed.

Finally, it is worth mentioning that this method is independent of the start-up configuration or running configuration of the CISCO device.

3.1.1 Case Study of method with two memory dumps

This method is followed for a CISCO WS-C3750G-24TS layer 3 switch, with IOS software version 12.2(55)SE6 but it is also similar for ASR1K series routers that run IOS XE.

3.1.1.1 Device preparation

Device configuration must take place in order to be able to store the memory dump core file on a remote server. Although this can be implemented with a protocol like *tftp*, the ***ftp*** is preferred because of no limitations on memory core file size.

```
router#conf  t
```

```
ip ftp username Cisco
ip ftp password 7 0321xxxxxxxxx710A1016141D
exception core-file r-router compress timestamp
exception protocol ftp
exception region-size 65536
exception dump ip_address
end
```

Listing 1: Configuration for the memory dump

The memory dump is actually produced by the execution command ***write core***.

```
router# write core
```

Listing 2: Memory dump command

3.1.1.2 Extracting text memory area from memory dump

Now that the memory dump file is produced, an uncompressing is needed with a common uncompressing tool and extraction of the text memory area. In order to do that one has to find the **starting address** as well the **size** of the text memory area. This information is provided by the ***show region*** command with the description as "coredump:text".

```
router#sh region
Region Manager:

      Start         End      Size(b)     Class  Media          Name
 0x00000000  0x07FFFFFF   134217728     Local  R/W            main
 0x00000020  0x07FFFFFF   134217696     Local  R/W            main:coredump
 0x00003054  0x00403053     4194304     Local  R/W            coredump:heap
 0x004030A8  0x008030A7     4194304     Local  R/W            coredump:heap
 0x008030FC  0x00FFFFFF     8376068     Local  R/W            coredump:heap
 0x01000000  0x02DD9C07    31300616      Text  R/W            coredump:text
 0x02E00000  0x02EFFFFF     1048576      Text  R/W            coredump:dltext
 0x02F00000  0x038D2DDB    10300892      Data  R/W            coredump:data
 0x035A179C  0x035E179B      262144     Local  R/W            data:reclaimed_heap
 0x038D2DDC  0x0433592B    10890064       Bss  R/W            coredump:bss
 0x04335930  0x063FFFFF    34383568     Local  R/W            coredump:heap
 0x06400000  0x06FFFFFF    12582912     Iomem  R/W            coredump:iomem


Free Region Manager:

      Start         End      Size(b)  Class  Media  Name
 0x07000054  0x07FFDFFF    16768940  Local  R/W    heap
```

Listing 3: Memory mapping

The HEX value of starting address of the text memory area is computed which gives HEX(10000000)=16777216.
Then, this value is used to extract the text memory area from the memory core dump file along with the text area size which is 31300616. In order to complete extraction the ***dd*** tool on a unix-like system is used:

```
dd if=router.dump bs=1 skip=16777216 count=31300616 of=router.dump_main_text
```

Then the md5 hash of the **`router.dump_main_text`** file has to be computed.

```
root@kali:~/# md5sum router.dump_main_text
2f07329cbb22330a1618a2f7201b844b  router.dump_main_text
```

Repeating the same process with the same router platform, but with a known-good image and extracting again the text area memory can be used as a reference. Then the md5 hash is computed again.

```
root@kali:~/# md5sum router.cleandump_main_text
2f07329cbb22330a1618a2f7201b844b  router.cleandump_main_text
```

If hashes match as in this case the IOS image integrity of the device can be verified.

3.1.2 Extracting text memory area directly from CLI [9, 10]

A faster way to produce the text area file is to issue the following command from command line interface (CLI) and then compute the hash from a computer workstation.

```
router#copy system:memory/text ftp:
Address or name of remote host []? <FTP server ip  address>
Destination filename [text]? router_main_text
Writing router_main_text !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
<output suppressed>
```

Except for ftp there is also a possibility to use another more secure protocol like scp or https.

3.1.3 Computing the hash of text area from inside the router [9, 10]

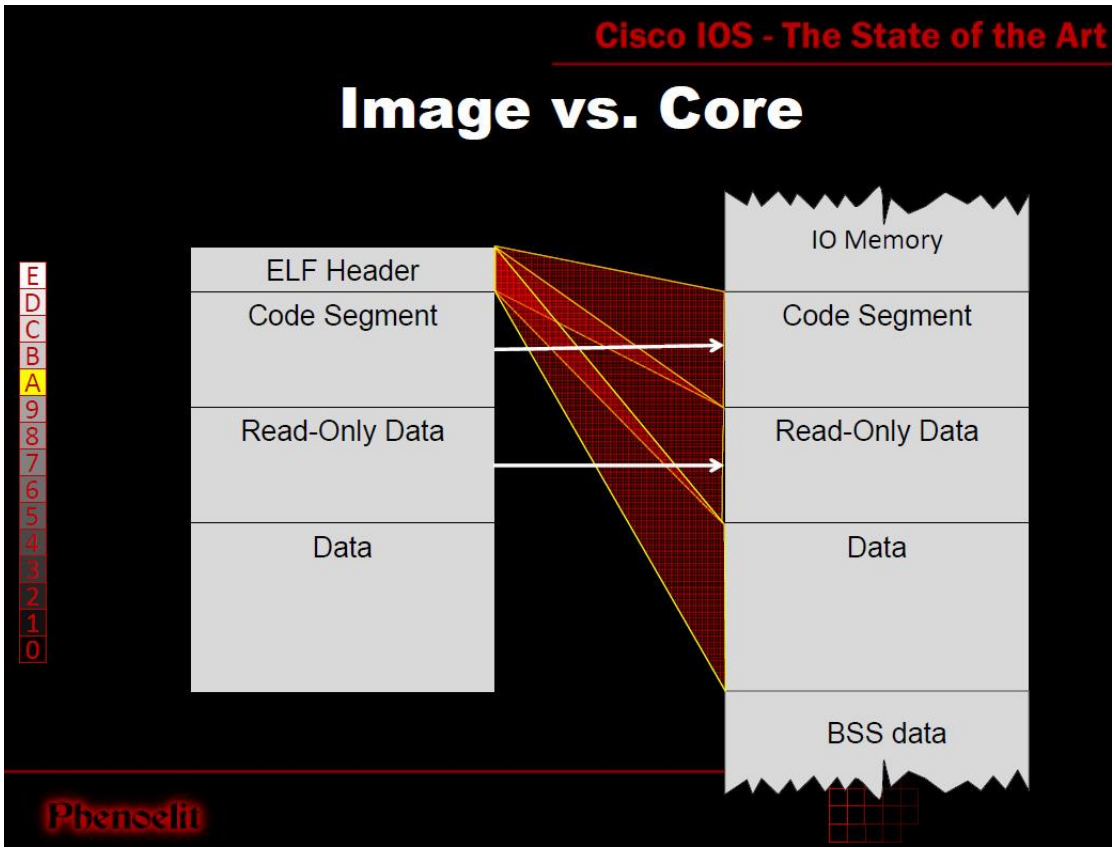
At last there is a more convenient way to compute the hash of the text area directly from CLI. Administrators may use the following command:

```
verify /md5 system:memory/text
```

Although this procedure produces the desired result in a fast and convenient way is considered the least secure one. It should be avoided as it is very easy for an attacker to preinstall this value within a possible compromised image.

3.2 Method of one memory dump and one known-good IOS image

Another method, which is convenient enough and easy to use, requires one memory dump and a known-good image. This method also relies on the fact that the text memory area contains the instructions to be executed and should not be overwritten. The main difference is that the reference text area memory hash is derived from a known-good image itself. This method has been described by Felix Lindner during a talk on CISCO forensics.



Slide form FX@25C3 talk on CISCO IOS Forensics and Exploits

Based on this method, Didier Stevens [4] has developed a tool to check the image integrity which is a part of a python toolset named "Network Appliance Forensic Toolkit". In order to use this script `naft-icd.py`, a memory dump is needed as described in previous paragraphs as well as a known-good IOS image. The command is issued with `checktext` argument.
e.g.

```
python naft-icd.py checktext router.dump router_ios_.bin
```

In case that the IOS image is compressed with a proprietary algorithm (as in 3750 switches) this method is not applicable because the tool cannot uncompress the image.

3.3 Proposed methods applicability

Newer versions of IOS have implemented the **Address Space Layout Randomization** (ASLR) technique. ASLR is a computer security technique used in protection against buffer overflow attacks. In order to prevent an attacker from reliably jumping to a particular exploited function in memory (for example), ASLR involves randomly arranging the positions of key data areas of a program, including the base of the executable and the positions of the stack, heap, and libraries, in a process's address space[2].

3.3.1 IOS ASLR enabled detection methods

Since CISCO is not able to provide users with a complete list of platform/IOS combinations where the ASLR feature is implemented, one has to discover it by

himself. It is tricky task and it requires rebooting the router and issuing the command `show region` twice.

```
Router1#show region
Region Manager:

      Start          End      Size(b)  Class  Media  Name
 0x16000000  0x17FFFFFF    33554432  Iomem  R/W    iomem:(iomem)
 0x60000000  0x75FFFFFF   369098752  Local  R/W    main
 0x6001B5B8  0x6487FFFF    75909704  IText  R/O    main:text
 0x6488BC40  0x6692125F    34166304  IData  R/W    main:data
 0x66921260  0x6742621F    11554752  IBss   R/W    main:bss
 0x67426220  0x75FFFFFF   247307744  Local  R/W    main:heap
 0x80000000  0x95FFFFFF   369098752  Local  R/W    main:(main_k0)
 0xA0000000  0xB5FFFFFF   369098752  Local  R/W    main:(main_k1)
 0xF6000000  0xF7FFFFFF    33554432  Iomem  R/W    iomem
```

show region before reboot

```
router1#show region
Region Manager:

      Start          End      Size(b)  Class  Media  Name
 0x16000000  0x17FFFFFF    33554432  Iomem  R/W    iomem:(iomem)
 0x60000000  0x75FFFFFF   369098752  Local  R/W    main
 0x6001EDF8  0x6487FFFF    75895304  IText  R/O    main:text
 0x6488F480  0x66924A9F    34166304  IData  R/W    main:data
 0x66924AA0  0x67429A5F    11554752  IBss   R/W    main:bss
 0x67429A60  0x75FFFFFF   247293344  Local  R/W    main:heap
 0x80000000  0x95FFFFFF   369098752  Local  R/W    main:(main_k0)
 0xA0000000  0xB5FFFFFF   369098752  Local  R/W    main:(main_k1)
 0xF6000000  0xF7FFFFFF    33554432  Iomem  R/W    iomem
```

show region after reboot

As it can be noticed, the starting address of the **main:text** memory area, the **main:data** and the **main:bss** area have been changed, which implies that the ASLR technique is implemented.

There is also the chance that the starting address of the **main:text** area remains the same but the **main:data** and the **main:bss** area have been changed.

The below example is taken from a 3845 CISCO router with c3845-adventerprisek9_ivs-mz.124-3j.bin IOS image.

```
Router2#sh region
Region Manager:

      Start          End      Size(b)  Class  Media  Name
 0x0FFFFE00  0x0FFFFFFF         512  Iomem  R/W    BCM region
 0x2D900000  0x2FFFFDFF    40893952  Iomem  R/W    iomem
 0x60000000  0x6D8FFFFF   227540992  Local  R/W    main
 0x6001105C  0x63040F77    50528028  IText  R/O    main:text
 0x63049EE0  0x6593FBFF    42949920  IData  R/W    main:data
 0x6593FC00  0x65EC833F     5801792  IBss   R/W    main:bss
 0x65EC8340  0x6D8FFFFF   128154816  Local  R/W    main:heap
 0x80000000  0x8D8FFFFF   227540992  Local  R/W    main:(main_k0)
 0xA0000000  0xAD8FFFFF   227540992  Local  R/W    main:(main_k1)
```

show region before reboot

```
Router2#sh region
Region Manager:

      Start          End      Size(b)  Class  Media  Name
 0x0FFFFE00  0x0FFFFFFF         512  Iomem  R/W    BCM region
 0x2D900000  0x2FFFFDFF    40893952  Iomem  R/W    iomem
 0x60000000  0x6D8FFFFF   227540992  Local  R/W    main
 0x6001105C  0x63040F77    50528028  IText  R/O    main:text
 0x6304AB20  0x6594083F    42949920  IData  R/W    main:data
```

```
0x65940840  0x65EC8F7F     5801792  IBss   R/W    main:bss
0x65EC8F80  0x6D8FFFFF   128151680  Local  R/W    main:heap
0x80000000  0x8D8FFFFF   227540992  Local  R/W    main:(main_k0)
0xA0000000  0xAD8FFFFF   227540992  Local  R/W    main:(main_k1)
```

show region after reboot

Another method to detect ASLR is to compute directly from CLI the hash of the main:text region provided that a known-good image is loaded to the router. An administrator can issue the command:

```
verify /md5 system:memory/text
```

record the hash value, reload the router, issue the same command and compare the hashes.
In the case of an ASLR image, there is not a tool or method publicly available to perform an IOS image integrity check and a verification process must be requested to the vendor.
In the following Table 1 the advantages and disadvantages of each method are summarized.

Method	Advantages	Disadvantages
2 Whole Memory Dumps	Larger Scope.	Time consuming. Possible Big impact on network operation. Not applicable on ASLR enabled IOS.
2 Text Area Memory Dumps	Fast deployment. Larger Scope.	Possible medium impact on network operation. Not applicable on ASLR enabled IOS.
Compute text area hash from CLI	Negligible impact on network operation. Larger Scope.	Least reliable method. Not applicable on ASLR enabled IOS.
1 Memory Dump 1 IOS image	Fast deployment. Easier to deploy. Low impact on network operation.	Smaller Scope as MZIP compressed IOS not supported. Not applicable on ASLR enabled IOS.

Table 1: IOS Integrity methods' comparison

3.3.2 Additional checks: call stacks and iosd boundaries [9, 10]

During typical operation, Cisco IOS processes should have the program counter (PC) and return address (RA) within the boundary of the text section as this section was identified by the output of the **show region** command. To confirm that the PC and RA are within the text section boundaries, network administrators must use the **show**

stacks *pid* command where the process ID (PID) can be obtained by using the **show processes** command.
The following example shows how to display the PID of the process running on the 3750 Cisco IOS device of the previous example where the text region starts at 0x01000000 and ends at 0x02DD9C07.

```
router#show processes
CPU utilization for five seconds: 21%/3%; one minute: 23%; five minutes: 22%
 PID QTy       PC Runtime(ms)   Invoked   uSecs    Stacks   TTY Process
   1 Cwe   2AE82D4        4747     47028     100  5460/6000     0 Chunk Manager
   2 Csp   1BCC0D0       63375   2469566      25  2588/3000     0 Load Meter
   3 Hwe   1FF48E0           0         1       0  8764/9000     0 Connection Mgr
   4 Lst   2AF6C40   151037105   6356998   23759  5676/6000     0 Check heaps
   5 Cwe   2AFEAE4       15453      5433    2844  5524/6000     0 Pool Manager
   6 Mst   1F7CF14           0         2       0  5560/6000     0 Timers
   7 Mwe   10277FC           0         1       0  5760/6000     0 AggMgr Process
   8 Mwe   1392F64           0         1       0 23672/24000    0 Crash writer
   9 Mwe   1D353FC    40272795  75256891     535  3060/6000     0 ARP Input
[Output omitted for brevity]

router#show stacks 9
Process 9:  ARP Input
  Stack segment 0x35A2CD4 - 0x35A4444
  FP: 0x35A43F0, RA: 0x2B03C80
  FP: 0x35A4430, RA: 0x1D35400
  FP: 0x35A4438, RA: 0x1BD4A68
  FP: 0x0, RA: 0x1BCB4E0
```

However this additional check is hard to accomplish in a manual manner considering the fact that often exist more than 300 processes running in a router.

3.3.3 Checking the ROM Monitor Information [9, 10]

The ROM monitor is a bootstrap program that initializes the hardware and boots the Cisco IOS Software. The booting location of the Cisco IOS Software is defined on the information stored in ROMMON (See APPENDIX A for a full booting process diagram). Although there is not a feasible way to check the integrity of the ROMMON, one can interrupt the booting process and check the information stored with the **set** command.

```
switch: set
BOOT=flash:/c2960-lanbasek9-mz.122-50.SE5/c2960-lanbasek9-mz.122-50.SE5.bin
CLEI_CODE_NUMBER=COMCU00ARB
MAC_ADDR=D4:D7:XX:XX:XX:XX
MODEL_NUM=WS-C2960-24TC-L
MODEL_REVISION_NUM=T0
MOTHERBOARD_ASSEMBLY_NUM=73-XXXXX-XX
MOTHERBOARD_REVISION_NUM=XX
MOTHERBOARD_SERIAL_NUM=FCQ1XXXXX
POWER_SUPPLY_PART_NUM=341-0097-03
POWER_SUPPLY_SERIAL_NUM=ALDXXXXXX
SDM_TEMPLATE_ID=0
SWITCH_PRIORITY=1
SYSTEM_SERIAL_NUM=FCQ15XXXXX
TAN_NUM=800-XXXXX-XX
TAN_REVISION_NUMBER=XX
VERSION_ID=V10
```

4 Memory Dump Analysis

In Network Appliance Forensic Toolkit there are some scripts available to analyze memory dumps. The most useful ones can provide information such as:

- show history: user issued commands from command line interface.
- show events: logging information which has been buffered
- show region: memory mapping
- Extract IPv4 packets: ip packets that were still in memory
- CW Strings: Information about IOS and hardware platform
- show Processes : running processes before memory dump

There are also some free tools available that have been developed by Recurity Labs. However, these tools are only available for specific platforms (CISCO 1700, CISCO 2600, and CISCO 2691).

5 Prevention [1]

In order to prevent infection either during runtime or to avoid loading a compromised image [8] into a networking device, there are some practices that the network administrators should follow.

5.1.1 Up-to-Date Software

IOS is an operating system under continuous development. CISCO is releasing software updates with bug fixes and new security features. Therefore, it is imperative that network administrators maintain their networks with up-to-date software and they are regularly informed about CISCO's advisories. The best available resource is Cisco's Security Intelligence Operations portal that can be reached at: http://tools.cisco.com/security/center/home.x

5.1.2 Implement Change Control

Change control is a mechanism through which changes being made to network devices are requested, approved, implemented, and audited. In the context of ensuring the authenticity of CISCO IOS software images used in the network, change control is relevant because it helps greatly when determining which changes have been authorized and which are unauthorized.

5.1.3 Limit Access to Devices

Once Authentication Authorization and Accounting AAA has been implemented to control which users can log in to particular network devices, access control should be implemented to limit from which IP addresses users may perform management functions on a network device. This access control includes multiple security features and solutions to limit access to a device:

- VTY access classes
- Infrastructure Access Control Lists (iACL)
- Simple Network Management Protocol (SNMP) access lists

5.1.4 Isolated Management VLAN

Workstations that are used for management purposes of network devices or IT equipment in general should operate in an isolated ***Management VLAN.*** This VLAN should also have no access to external networks either for incoming or outgoing communications.

5.1.5 Verifying IOS image integrity (Offline)

Every IOS image that is going to be installed in any network device should follow a verification procedure to ensure authenticity and integrity. It is also a good practice not to run preinstalled IOS as it may have been compromised during the supply chain. Network administrators should fully erase the flash and the NVram memory of the router and install an IOS that has been verified by them, prior to installing the device in a production environment. The first step is to download the latest IOS image with the desired feature set form CISCO's website (Fig.1).

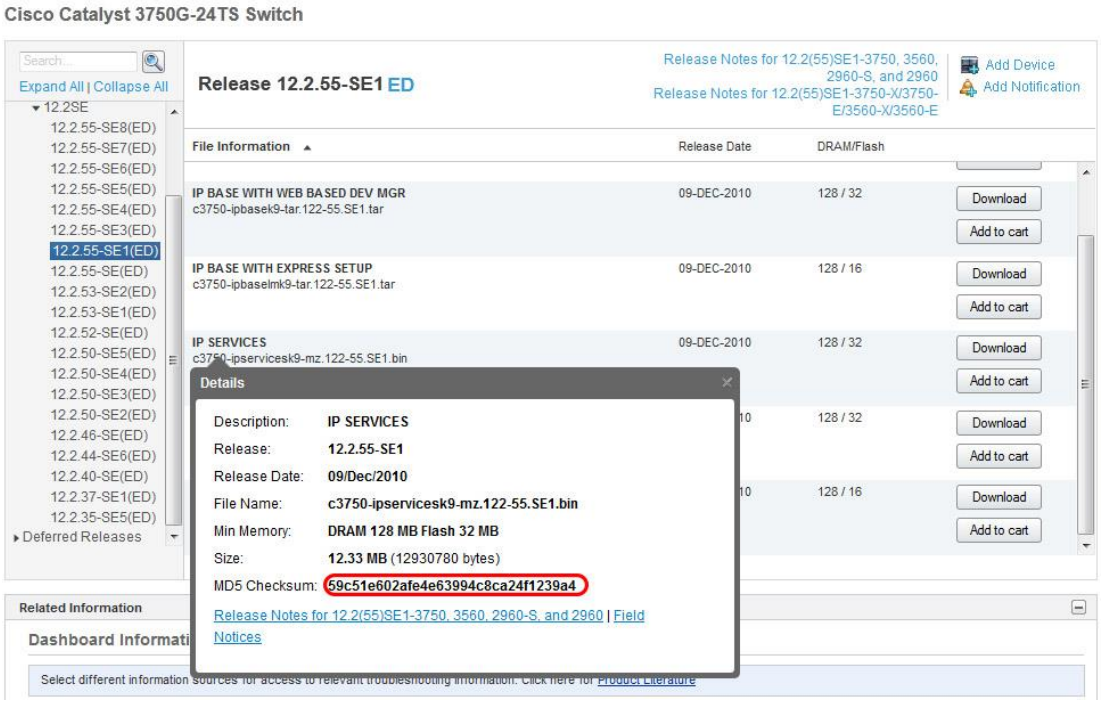


Figure 1: IOS image MD5 checksum

The MD5 checksum that is provided from the download page should be recorded. Running the hash function against (Fig.2) the downloaded file or against one that is on already saved in a storage server can verify the image.

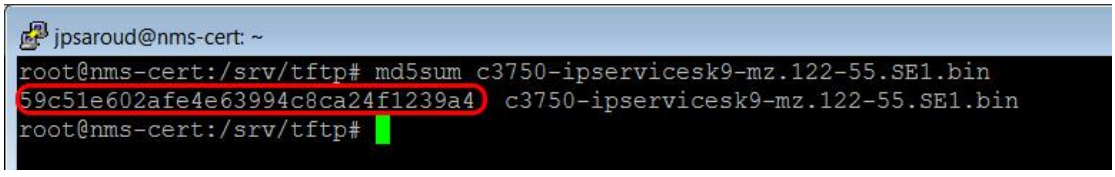


Figure 2: Validating a stored IOS image with MD5 checksum

In CISCO IOS Software Releases 12.2(4)T and 12.0(22)S, a MD5 File Validation feature was added that allows network administrators to calculate the MD5 hash of a CISCO IOS software image file that is loaded on a device. It also allows administrators to verify the calculated MD5 hash against that provided by the user. Once the MD5 hash value of the installed CISCO IOS image is determined, it can also be compared with the MD5 hash provided by CISCO to verify integrity of the image file.

```
router#verify /md5 flash:c3750-ipservicesk9-mz.122-55.SE1.bin
.....................................................................................
```

```
.....................................................................................
.....................................................
.....................................Done!
verify /md5 (flash:/c3750-ipservicesk9-mz.122-55.SE1.bin) =
59c51e602afe4e63994c8ca24f1239a4
```

Figure 3: Validating an IOS image with MD5 checksum stored in a network device

Network administrators can save time by downloading a compressed file from CISCO's web site that contains all the hash values of all IOS images. File can be reached at the following link:
http://www.cisco.com/c/dam/assets/about/security/resources/ioshashes.zip.

5.1.6 Verifying IOS image (online)

Network administrators can build a reference database with the hashes of the text area region as described in the first paragraphs related with their own equipment. During an initial networking device installation with a known-good IOS image, it is easy enough to compute the hash value through the CLI command and identify if the IOS image that is being used is also ASLR enabled. (See paragraphs 3.1.3 & 3.3.1). Periodically checks of the image integrity can be scheduled according to the initially developed methods and based on that reference database.

5.1.7 Centralized log management

For network administrators to overview events taking place on a network, a comprehensive logging structure using centralized log collection and correlation must be implemented - preferably using a SIEM platform. Additionally, standardized logging and time configuration must be deployed on all network devices to facilitate accurate logging. Furthermore, logging from the AAA functions in the network should be included in the centralized logging implementation. It is described shortly below along with the most important implementations that must exist in a centralized log management system.

Set up "ntp" with authentication to ensure synchronization between network devices with authentic time servers.

```
ntp authenticate
ntp authentication-key number md5 value
ntp trusted-key key-number
ntp server ip-address [version number] [key keyid] [source interface] [prefer]
```

Set up "timestamps" to ensure proper time format and zone for the syslog messages.

```
 service timestamps log datetime msec show-timezone
```

Setup the centralized "logging server(s)" where all syslog messages should be stored.

```
logging server ip
logging trap level
```

Setup the "login" facility to log successful or failed user logins.

```
login on-failure log
login on-success log
```

Setup "archive log" to log **configuration changes. [3]**

```
archive
 log config
logging enable
logging size 200
hidekeys
notify syslog
```

Setup "accounting" with TACACS+ to log **execution commands** [1]

```
aaa accounting exec default start-stop group tacacs
aaa accounting commands 0 default start-stop group tacacs
aaa accounting commands 1 default start-stop group tacacs
aaa accounting commands 15 default start-stop group tacacs
```

TACACS+ logs are preserved in a file managed by TACAS+ configuration settings. It is wise to index/store both syslog messages and tacacs+ log file into a SIEM system (Fig 4).

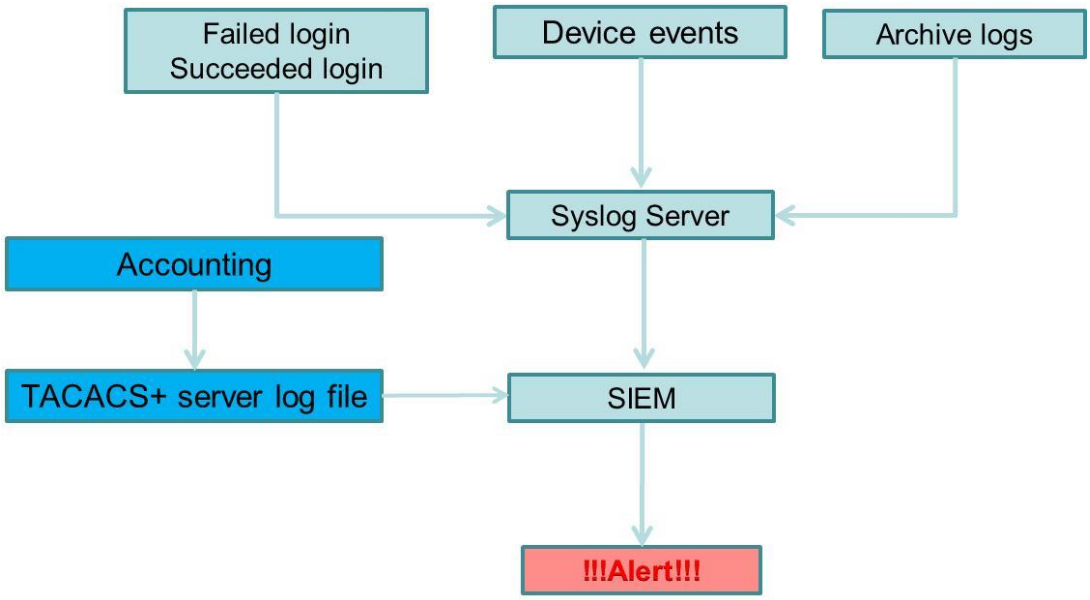


Figure 4: Centralized log management with SIEM

The latter will also allow a correlation procedure to take place as well as a more efficient monitoring process (Fig. 5).

5.1.8 Monitoring attributes

As it was stated before, some IOS features (such as GDB) may introduce a serious security risk. As long as there is no way to disable these features, network administrators should establish an alerting mechanism in case they are used.
The commands that should definitely raise an alert upon execution are:

- **gdb**
- **test**
- **tclsh**
- **copy**
- **reload**

The commands that should definitely raise an alert upon configuration change are:

- **service internals**
- **boot**

- **config-register**

The last command controls the booting process of the network device which is explained in detail in APPENDIX A.
Also the following commands may be used to connect to line cards or switch processors on products that support them. They are particularly important because after the CISCO IOS device is connected to a line card or switch processor the commands executed are not logged or authorized using the AAA server.[10]

- **attach**
- **remote**
- **ipc-con**
- **if-con**
- **execute-on**

For IOS XE the following commands must be also monitored:

- **attach**
- **hw-module**

as well the commands that can be used to connect to the Linux shell of the Cisco IOS XE.

- **platform shell**
- **request**

Common commands that should be monitor because they might be used during a reconnaissance phase of an upcoming attack are:

- **show platform**
- **show region**
- **show memory**

Last but no least the **do-exec** version of any of the previously mentioned commands while in configuration mode.

Important Note: While CISCO IOS allows command abbreviation, abbreviated commands such as **tes**, **rem,** etc should also be considered when examining logs.

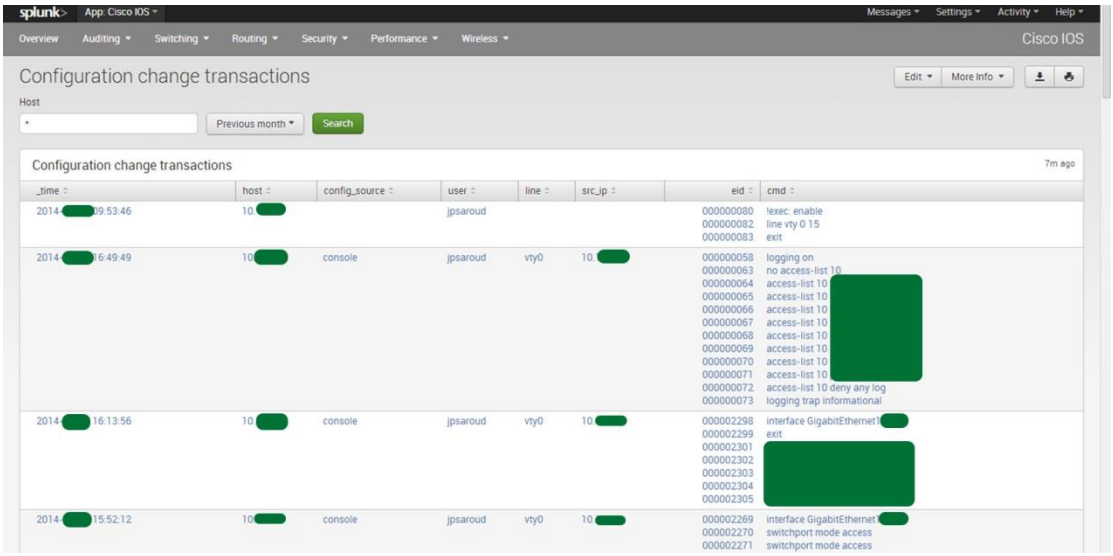


Figure 5: Monitoring configuration changes with SPLUNK

5.1.9 Alert from a device with a CISCO IOS Applet

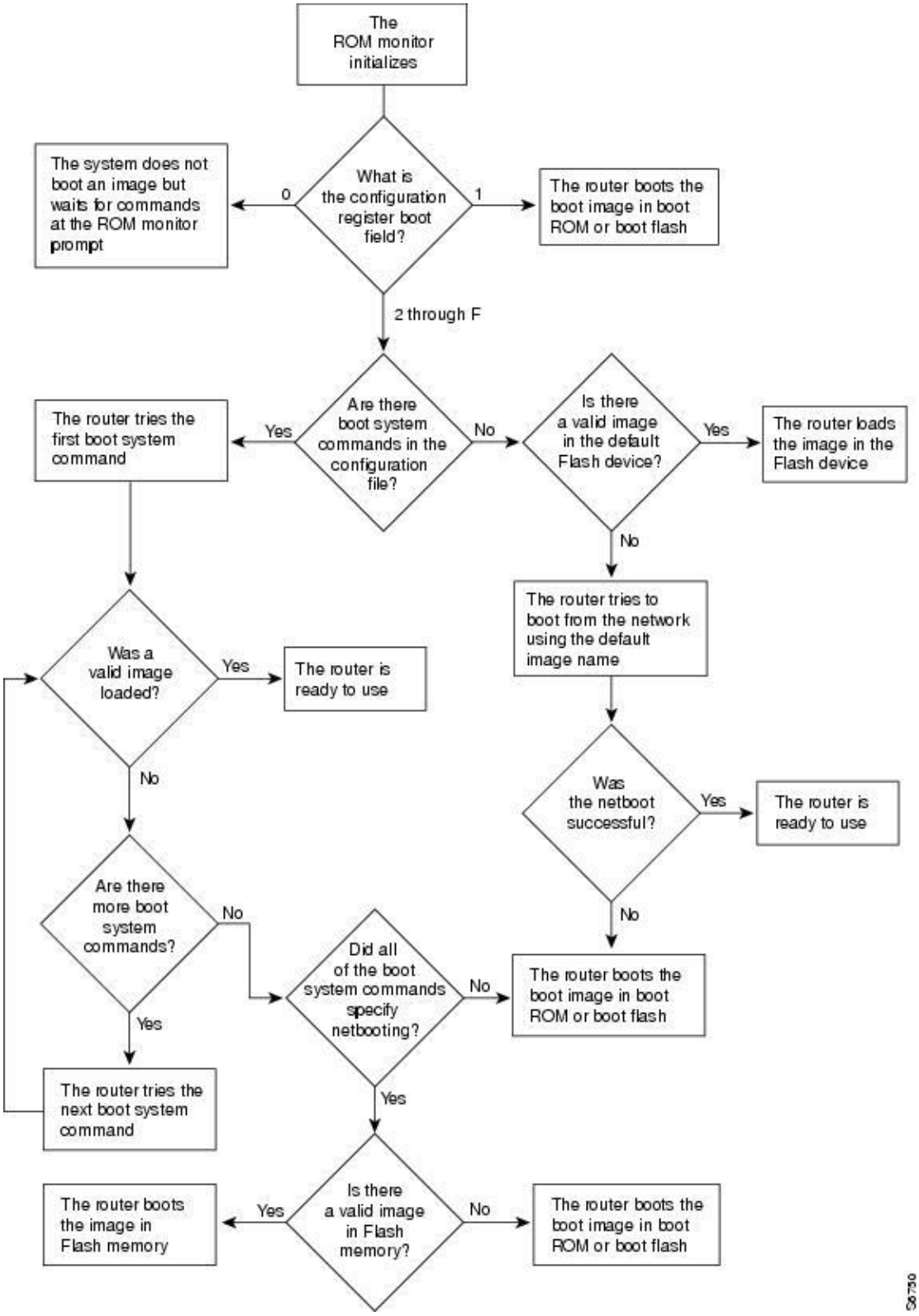
As mentioned before, some network devices have built-in scripting capabilities. Network administrators can take advantage of this feature and raise an alert (an email notification for example) from the device itself. In the following example, an email notification is send to a predefined email address upon successful ssh login.

```
event manager environment _email_to username@domain
event manager environment _email_server ip address
event manager environment _email_from device_name@domain
event manager applet login-ssh-ok
event syslog pattern "SEC_LOGIN-5-LOGIN_SUCCESS:"
action 1.0 mail server "$_email_server" to "$_email_to" from
"$_email_from" subject "$_event_pub_time: Login via SSH" body "$_syslog_msg"
 action 1.5 syslog msg "LOGIN SUCCESS - Mail Sent"
```

5.1.10 Restrict set of available commands

Another approach that network administrators may follow is to restrict their users to a set of available execution commands. Since these commands cannot be disabled from the device itself, it is possible to configure TACACS+ to deny execution of these commands given that TACACS+ is used for AAA. Configuration is specific to the used TACACS+ application.

APPENDIX A: CISCO IOS booting process



Bibliography

1. CISCO Guide to Harden CISCO IOS Devices, 2011

http://www.Cisco.com/c/en/us/support/docs/ip/access-lists/13608-21.html#usingaaa

2. Address space layout randomization (ASLR)

http://en.wikipedia.org/wiki/Address_space_layout_randomization

3. Log configuration commands entered on your CISCO router

http://blog.ipspace.net/2006/11/log-configuration-commands-entered-on.html

4. Network Appliance Forensic Toolkit | Didier Stevens,2014

http://blog.didierstevens.com/programs/network-appliance-forensic-toolkit/

5. Killing the myth of CISCO IOS rootkits:DIK (Da Ios rootKit), Sebastian 'topo' Muñiz, May 2008

http://www.coresecurity.com/files/attachments/Killing_the_myth_of_Cisco_IOS_rootkits.pdf

6. Fuzzing and Debugging CISCO IOS, Muniz Sebastian, Ortega Alfredo, March 2011

https://media.blackhat.com/bh-eu-11/Sebastian_Muniz/BlackHat_EU_2011_MunizOrtega_Cisco_iOS-WP.pdf

7. CISCO IOS rootkits and malware, Jason Nehrboss, 2012.

https://hakin9.org/exploiting-software-0412/

8. Rootkits on CISCO IOS Devices, 2008

http://www.Cisco.com/c/en/us/support/docs/csr/Cisco-sr-20080516-rootkits.html

9. CISCO IOS Software Integrity Assurance, 2014

http://www.cisco.com/web/about/security/intelligence/integrity-assurance.html

10. CISCO IOS XE Software Integrity Assurance, 2014

http://www.cisco.com/web/about/security/intelligence/ios-xe-integrity-assurance.html