Fuzzing and Debugging Cisco IOS

Muñiz, Sebastian
smuniz@groundworkstech.com
Groundworks Technologies

Ortega, Alfredo
aortega@groundworkstech.com
Groundworks Technologies

March 2, 2011

Abstract

We will present a tool which facilitates debugging and reverse engineering process of Cisco IOS by allowing the integration with most used existing debugging and disassembler tools such as GDB and IDA Pro. This solution consists of a modification that provides instrumentation capabilities for an existing hardware emulator called Dynamips. Among other things, our modification will enable the use of existing fuzzing tools and frameworks and complete analysis of boot-loading processes. It will debug the target IOS independently of pre-existing built-in GDB on IOS image; it will provide more reliability during debugging session and an isolated environment to reproduce attacks and analyze IOS malware.

1 Introduction

2 Cisco IOS architecture

Cisco IOS (originally called Internetwork Operating System) is a software used on most Cisco Systems routers and network switches which provides routing, switching, internetworking and telecommunication functions. Cisco IOS runs as a single binary image which is decompressed at boot-time (see main architecture at Fig. 1). Since it is an operative system that lacks process isolation, all the processes share the same memory space and there is no protection among them; which means that if memory is corrupted in one of the running processes, any other Operating System component might be affected, so tracking the problem will be a difficult task. The kernel also implements a cooperative scheduler. The scheduler is implemented as run-to-completion which means that processes have to relinquish their right to execute code and make a kernel call so that other processes can run. It is also worth mentioning that if a process runs for more than a predefined time, it will be terminated by a watchdog. Differentiation of process priority

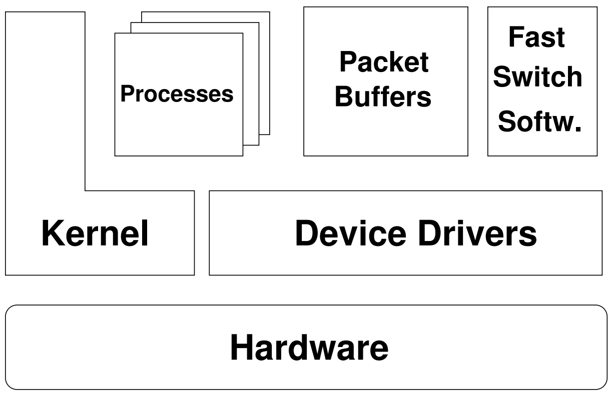

Figure 1: Cisco IOS process memory

levels is implemented so that higher priority processes get rescheduled before lower priority ones.

Recent versions of IOS make use of MMU (Memory Management Unit) on their CPUs to set strict permissions on image sections, i.e. .text section is marked with read-execute permissions. These protection mechanisms [9] have been present on major Operating Systems for several years. It is also worth noticing that recent versions of Cisco IOS perform a relocation of the main IOS image during boot process, making common remote attacks like buffer overflows -which rely on fixed addresses- impossible. This could be considered as a rudimentary implementation of ASLR (Address Space Layout Randomization) [9].

2.1 Built-in GDB server

The IOS built-in GDB server is the debugging interface for Cisco developers and support engineers which allows them to debug IOS processes. It also allows remote image diagnostic since it is capable of working over a Telnet session as well as over a Serial session established on the console port. The GDB client needed to interact with the built-in GDB server is similar to the standard GDB client with some minor adjustments. Cisco IOS uses a different RLE (Run Length Encoding) and GDB's maintainers dropped support for it a few years ago so if the user wants to use the standard client the code needs to be fixed. The built-in GDB server inside current Cisco IOS is capable of working in three different ways:

Process examination: Invoked with "gdb examine ¡pid¿" CLI command it allows memory and processor registers inspection but it cannot modify system values (either memory or registers values). The system execution continues normally during debugging so 'examine' mode can be executed over a Telnet session.

Process debugging: Invoked with “gdb debug ¡pid¿“ CLI command, it is used in situations where the device console port is not accessible. It works by catching unhandled exceptions on the specified process, setting it in a special state where it will not be rescheduled and then running the built-in GDB process to debug the failed process. IOS continues to run during process debugging, so it is possible to debug a process over a Telnet session with certain restrictions. The scheduler, an interrupt service routine or any process needed for debugging path (such as TCP/IP) cannot be debugged over this session. This debugging mode is capable of memory and processor registers modification so this is the best option for an attacker to remotely modify the device memory and insert special code to manipulate IOS functionality.

Kernel debugging: If the attacker gains physical access to a console port, he or she can execute the kernel debugger which is the preferred way to debug a device. In this mode the entire device execution is stopped during the exception, freezing all system processes. The privileged CLI command ”gdb kernel“ is used to initiate a debugging session using the serial port and it is not available through a VTY.

An unfortunate fact about the built-in GDB server is that when it tries to access specific regions of memory on certain devices, it does not catch the exception correctly and it reloads [1] the device. It is also discouraging the fact that the built-in GDB server performs certain actions (such as cache flushing) to setup the debugging environment that modifies Cisco IOS behavior thus making a shellcode debugging session behave differently than when tested on a device without the built-in GDB server activated.

3 Dynamips emulator

Dynamips [2] is a hardware emulation program created to emulate Cisco routers. Unlike other virtualization software, Dynamips emulates the underlying hardware and does not provide any access from guest Operating System (in this case Cisco IOS) to the host Operating System. It is able to run on Linux, Mac OS X and Windows and can emulate low-end and middle- end Cisco devices like 7200, 3600, 2691, 3725, 3745, 2600 and 1700 as well as some of their networking hardware without the need of an actual physical device. With Dynamips it is possible to emulate complex network topologies on a single PC because it has the capability to bridge the emulated device networking hardware with the hosts network device.

[1]Reload is the IOS terminology for reset

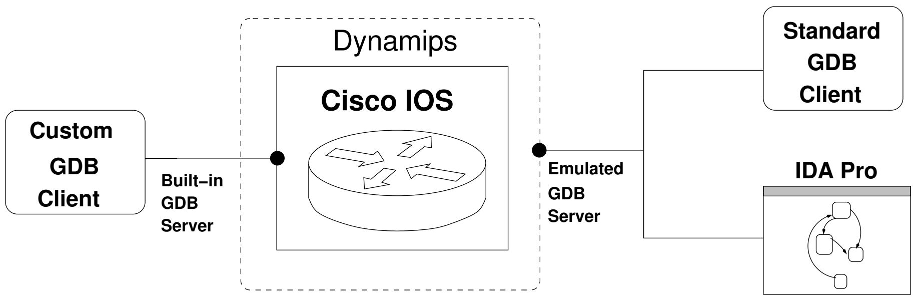

Figure 2: Dynamips modification

4 Dynamips modification

Dynamips modifications were used before to gain an insight of the inner workings of cisco IOS [10]. The main goal of our modification is to provide Dynamips the capability to interact with debuggers like GDB and IDA Pro using GDB's standard debugging protocol (not the modified Cisco's built-in GDB which needs a GDB client) and to allow the user to monitor and modify the running IOS with the minimum interference of the underlying Operating System. The bulk of the debugger is implemented in the files *gdb_cmd.c*, *gdb_server.c*, and *gdb_proto.c*, however, extensive modifications of original emulation functions exists in the ppc32* and mips64* source files to support devices of both PowerPC and MIPS architecture thus covering all the Cisco devices supported by Dynamips.

4.1 GDB Exception handling

In the event of an invalid program counter exception, the debugger will stop just before the execution, to make the offending instruction and code path analysis easier.

4.2 GDB remote debugging protocol and implemented commands

A subset of commands defined by the standard GDB remote debugging protocol was implemented. The definition is in the *gdb_cmd.h* file. This is a list of implemented commands on the current version of Dynamips GDB patch:

- set_cpu_reg (ppc/mips)
- get_cpu_reg (ppc/mips)
- set_cpu_regs (ppc/mips)
- get_cpu_regs (ppc/mips)

- proc_continue
- proc_step
- insert_breakpoint
- remove_breakpoint
- red_mem
- write_mem

Even this small subset can provide a comfortable debugging experience as GDB can use the available commands to implement most of the remaining protocol. No additional software is needed to directly debug an IOS image running on the Dynamips emulator, apart from the modification described in this article.

4.3 IDA Pro support

The commercial disassembler IDA Pro [3] is a de-facto standard on the reverse engineering community. It includes support for GDB's remote debugging protocol since version 5.5, allowing it to debug any GDB-aware device if the architecture of the device being debugged was supported. The advantage of using IDA Pro is seen when the code being debugged can be previously analyzed so that routines, structures propagation, loops and other interesting parts of the code can be recognized and manipulated to enhance the reverse engineering process. Using the included GDB client, IDA Pro can interact with our Dynamips modification as a regular GDB client. Reverse engineering of IOS is greatly enhanced using the previously mentioned IDA Pro features.

5 Shortcomings of self-checking routines

Analyzing a potentially malicious piece of code using tools which are not isolated from that malware will render unreliable results since those analysis tools might have been tampered to hide the underlying real behavior (Fig. 3(a)). This same scenario should be avoided on any other environment, i.e. analyzing and looking for a virus in a computer using an anti-virus installed on that very same computer.

In the case of Cisco IOS, the verifications for IOS image integrity should be done using tools like md5sum [5] or sha1sum [6] and Cisco's recommendations such as CLI verification command [7] needs a trusted device.

According to Cisco recommendations for rootkits [8], when updating an IOS image, it is necessary to maintain the trust chain to ensure IOS image integrity during this process. It also suggests on its best-practices to

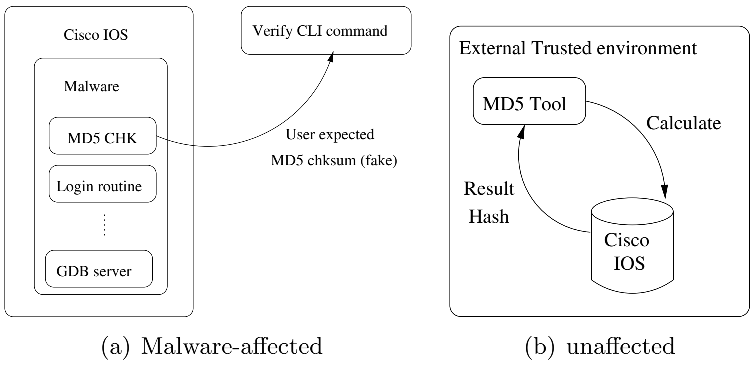

Figure 3: Built-in analysis (a) vs External analysis (b)

apply verification methods both in the device being updated (*verify* CLI command) as well as external verification tools.

In case of a run-time attack where the running image is modified either on memory or on the storage media (i.e. using a remote exploit), the supply chain integrity and verification method executing inside the attacked device to be analyzed are not effective to detect the modification performed by the attacker's code as it might have tampered with checking routines (Fig. 3(a)). Although in this case the offline verification is effective (Fig. 3(b)), this is probably not enough for a malware analyst as in the case of an infection that exists only in memory, Cisco does not provide any suitable detection method.

One possible solution to this problem is the modification of Dynamips presented in this article, which provides the ability to analyze memory, perform step by step execution in real-time on a totally isolated environment and hardly detectable by any process running within the emulated operative system.

An example would be an IOS image under analysis with the verified CLI command patch to always return to the expected MD5 signature (Fig. 3(a)). The analysis of a potentially malicious IOS image should be performed in a secure or virtual environment (Fig. 3(b)) instead of a physical device as it is done with Windows malware using virtual environments like Vmware [4].

6 Fuzzing/Vulnerability analysis

Fuzzing is a simple but effective technique often used on security reviews and vulnerability research [11].

We will present a typical black-box fuzzing session using the debugger and a very simple test-case generator. This fuzzer is implemented for demonstration purposes and it is not considered to be effective nor efficient, as there are specialized software available [12].

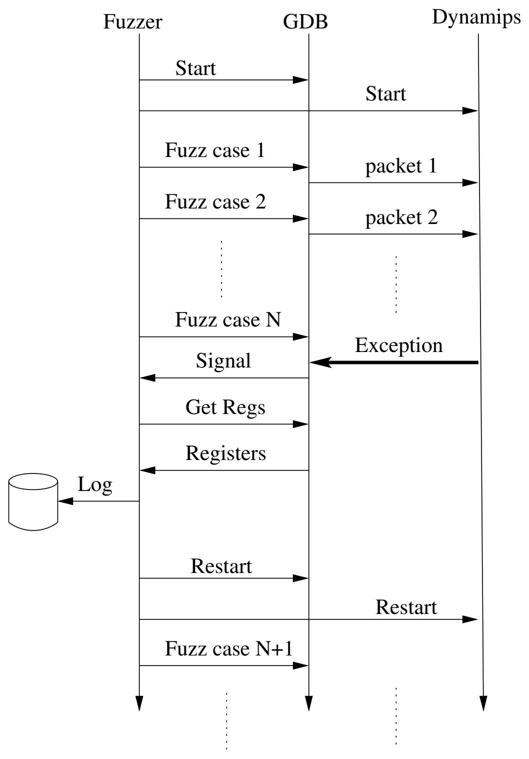

Figure 4: Time diagram of typical fuzzer session

The time diagram in Fig. 4 shows the fuzzing process. The demonstration fuzzer is a simple python script that connects with any ftp servers and runs every command with a very long string as the first parameter. This simple process is enough to uncover a very serious vulnerability present in the cisco FTP server [13]. This vulnerability has been fixed but the fuzzer will be run against an unpatched IOS image to trigger it.

The following is a log of a typical session of exploit development or vulnerability analysis. We start the Dynamips process with an emulated Cisco 17xx router and we activate the ftp server. On the first terminal we have the modified Dynamips output. It stops when the FTP bug is triggered:

```
Cisco Router Simulation Platform (version 0.2.8-RC2-amd64)
Copyright (c) 2005-2007 Christophe Fillot.
Build date: Feb 23 2011 20:25:28

IOS image file: ../C1700-EN.BIN

ILT: loaded table "mips64j" from cache.
ILT: loaded table "mips64e" from cache.
ILT: loaded table "ppc32j" from cache.
ILT: loaded table "ppc32e" from cache.
C1700 instance 'default' (id 0):
  VM Status  : 0
  RAM size   : 64 Mb
  NVRAM size : 32 Kb
  IOS image  : ../C1700-EN.BIN

Loading BAT registers
Loading ELF file '../C1700-EN.BIN'...
ELF entry point: 0x80008000

C1700 'default': starting simulation (CPU0 IA=0xfff00100), JIT disabled.
GDB Server listening on port 12345.
GDB Server: thread is now activated...
ROMMON emulation microcode.

Launching IOS image at 0x80008000...
Cisco IOS Software, C1700 Software (C1700-ENTBASEK9-M), Version 12.4(8a),
RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Wed 19-Jul-06 20:40 by prod_rel_team
Router>en
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ftp-server enable
Router(config)#
*Mar  1 00:00:44.911: %FTPSERVER-6-NEWCONN: FTP Server - new connection made.
-Process= "TCP/FTP Server", ipl= 0, pid= 47
```

On another terminal the fuzzer has generated the malformed MKD command that triggers the vulnerability:

```
Connected to 10.100.100.200.
220 Router IOS-FTP server (version 1.00) ready.
Name (10.100.100.200:alfred): anonymous
331 Password required for 'anonymous'.
Password:
230 Logged in.
Remote system type is Cisco.
ftp> mkd
(directory-name)
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
553 Access denied to
'/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA'
ftp>
```

Finally, on the third terminal the GDB client is attached to the emulator. An invalid memory access exception was captured and the fuzzer will log the registers and corresponding triggering test-case:

```
GNU gdb (GDB) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-unknown-linux-gnu
--target=powerpc-elf".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) target remote :12345
Remote debugging using localhost:12345
0xfff00100 in ?? ()
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0x80a397a4 in ?? ()
(gdb) x/i $pc
=> 0x80a397a4: br
(gdb) i r
r0             0x41414141 1094795585
r1             0x827e40a0 -2105655136
r2             0x82040000 -2113667072

.
.
.

r25            0x41414141 1094795585
r26            0x41414141 1094795585
r27            0x41414141 1094795585
r28            0x41414141 1094795585
r29            0x41414141 1094795585
r30            0x41414141 1094795585
r31            0x41414141 1094795585
pc             0x80a397a4 0x80a397a4
msr            0x0 0
cnd            0x0 0
lr             0x41414141 0x41414141
cnt            0x0 0
xer            0x0 0
mq             0x0 0
(gdb)
```

We can see that the Router is about to jump to the (attacker controlled address) 0x41414141 memory address.

7 Conclusion

We have presented a modification to the popular Dynamips Cisco emulator, which allows it to run as a GDB server and connect to multiple standard debugging tools. This modification does not include any additional analysis or post- processing, as the objective was to leverage the experience of the reverse engineer with standard debugging software packages. Further work may include the improvement GDB communication speed, and a faithful emulation of the Cisco Bootloader. This modification can be downloaded from the Groundworks Technologies Projects website [1].

References

[1] Groundworks Technologies, Dynamips GDB Server mod project, http://www.groundworkstech.com/projects/dynamips-gdb-mod

[2] Dynamips project, http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator, Retrieved January 2011.

[3] IDA Pro Disassembler and Debugger, http://www.hexrays.com/idapro/, Retrieved January 2011.

[4] VMWare virtualization software, http://www.vmware.com, Retrieved January 2011.

[5] GNU md5sum manual, http://www.gnu.org/software/coreutils/manual/html_node/md5sum-invocation.html, Retrieved January 2011.

[6] GNU sha1sum manual, http://www.gnu.org/software/coreutils/manual/html_node/sha1sum-invocation.html, Retrieved January 2011.

[7] MD5 File Validation, http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/cf_md5_ps6350_TSD_Products_Configuration_Guide_Chapter.html, Retrieved January 2011.

[8] Rootkits on Cisco IOS Devices, http://www.cisco.com/warp/public/707/cisco-sr-20080516-rootkits.shtml, Retrieved January 2011.

[9] Anley, Heasman, Lindner, Richarte, *The Shellcoder's Handbook: Discovering and Exploiting Security Holes,* ISBN: 978-0470080238, Wiley, 2nd Edition, 2007.

[10] Silvio Cesare, *Security Applications for Emulation*, Ruxcon, 2008.

[11] Ari Takanen, Jared D. DeMott, Charles Miller, *Fuzzing for Software Security Testing and Quality Assurance*, ISBN 978-1-59693-214-2, 2008.

[12] Eddington, *Developing Fuzzers with Peach 2.0*, Proceedings of CanSecWest Applied Security Conference, 2008.

[13] Cisco Security Advisory, *Multiple Vulnerabilities in the IOS FTP Server*, Advisory ID: cisco-sa-20070509-iosftp, 2009.