A dynamic analysis system for Cisco IOS based on virtualization

Yuxiang HAN, Shengli LIU, Xiaoyan SU, Zongli HU
Zhengzhou Institute of Information Science and Technology
Zhengzhou, China
e-mail: yuxianghan3@126.com

***Abstract*—**It is difficult to analyze and debug Cisco IOS with traditional debugging and disassembler tools such as GDB and IDA pro. These tools can't debug Cisco IOS in single step mode. This paper describes a dynamic analysis system based on a modification hardware emulator called Dynamips. Then, a new solution of inserting breakpoint with network events will be provided to analyze network communication procedures.

Keyword-IOS; virtualization; emulator; dynamic debugging system; breakpoint

I. INTRODUCTION

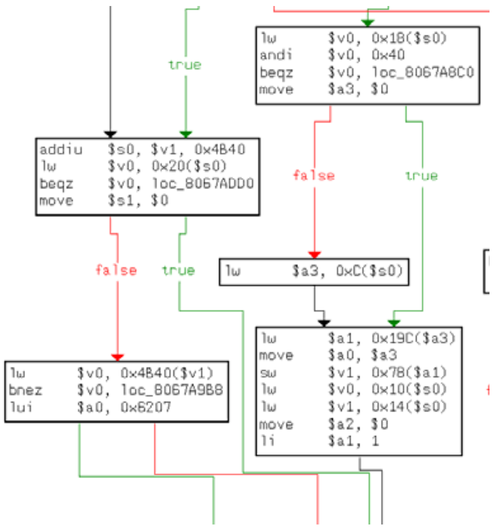
Cisco IOS (originally called Internetwork Operating System) is a non-open source operating system used on most Cisco router & switch equipments. Most of Cisco IOS images are standard ELF files. Cisco IOS is loaded by ROMMON (ROM Monitor) and decompressed to memory. Then, IOS image is launched at an address of 0x80008000. All the IOS processes share the same memory space, there is no isolation among different processes. This feature is the primary target of buffer overflow exploit, leads to all kinds of attack. Successful exploitation of software vulnerabilities in Cisco IOS distributed by different researchers and groups in the past [2]. Attackers can control routers and execute arbitrary code in some versions of Cisco IOS. Router security is essential to the whole network, but there is not any effective technique and assumption for Cisco IOS security analysis. New tools and assumptions are needed to analyze security mechanisms and operating mechanisms of Cisco IOS.

This paper will highlight a dynamic debugging and analysis system which is based on virtualization technology. If appropriate breakpoint is used in debugging system, IOS processes can be analyzed step by step. Meanwhile, the register and memory data can be set when needed. Breakpoint is the most important part of dynamic analysis techniques. But simple breakpoint can't work effectively when debugging network transit processes. An inserting breakpoint solution based on network protocols will be employed to solve this problem. The dynamic debugging model may be used in wide scale of other embedded systems.

II. TRADITIONAL ANALYSIS METHOD

The first observation about Cisco IOS analysis is that there are only two approaches published by researchers. Both of them are mentioned in the Black Hat. One is static analysis, the other is dynamic. Michael Lynn came to widespread attention in July 2005 following a controversy, informally known as "Ciscogate", that resulted from his research into a major security vulnerability of Cisco IOS, IDA pro was used to retrieve shellcode from IOS images. In Black Hat 2008, Gyan Chawdhary distributed the details of using IDA pro analysis, at the same time, GDB stub analysis method was been provided.

A. IDA pro analysis method



```
lw      $v0, 0x18($s0)
andi    $v0, 0x40
beqz    $v0, loc_8067A8C0
move    $a3, $0
```

```
addiu   $s0, $v1, 0x4B40
lw      $v0, 0x20($s0)
beqz    $v0, loc_8067ADD0
move    $s1, $0
```

```
lw      $a3, 0xC($s0)
```

```
lw      $v0, 0x4B40($v1)
bnez    $v0, loc_8067A9B8
lui     $a0, 0x6207
```

```
lw      $a1, 0x19C($a3)
move    $a0, $a3
sw      $v1, 0x78($a1)
lw      $v0, 0x10($s0)
lw      $v1, 0x14($s0)
move    $a2, $0
li      $a1, 1
```

Figure 1:IOS image analysis

IDA Pro is a Windows or Linux or Mac OS X hosted multi-processor disassembler and debugger, it can disassemble a series of executable files with different instruction sets, such as Intel, PowerPC and MIPS. Not only dynamic analysis, but also static. It can't support dynamic debugging for Cisco IOS, because Cisco router platform is needed.

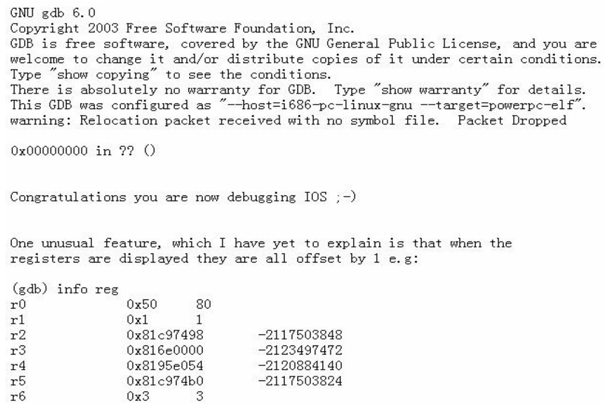
Decompressing the IOS image with standard Unix "unzip" tools or WinRAR. The uncompressed image is a standard ELF file. Generally, this process will be done by IOS loader at runtime. Using a hex editor to change the "e_machine" bit to 0x14.Load it in IDA pro using PPC instruction set.

Some IOS images may be encrypted, only part of image can be analyzed.

There is no symbol table for IOS images, therefore, all functions or processes would be distinguished relying on experience.

B. *GDB stub debugging*

The IOS built-in GDB stub is the debugging interface for Cisco developers and researchers which allows them to debug IOS processes[1]. Using Hyper Terminal to connect to Cisco router, the privileged CLI command "gdb kernel" is used to initiate a debugging session. A modification GDB will connect to the router via serial cable.



```
GNU gdb 6.0
Copyright 2003 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=powerpc-elf".
warning: Relocation packet received with no symbol file.  Packet Dropped

0x00000000 in ?? ()


Congratulations you are now debugging IOS ;-)


One unusual feature, which I have yet to explain is that when the
registers are displayed they are all offset by 1 e.g:

(gdb) info reg
r0             0x50      80
r1             0x1       1
r2             0x81c97498        -2117503848
r3             0x816e0000        -2123497472
r4             0x8195e054        -2120884140
r5             0x81c974b0        -2117503824
r6             0x3       3
```

Figure 2:GDB remote debugging

In this mode, the entire device execution is suspended, it must type "continue" to exit. Shellcode can be injected to memory for testing. It can disassemble specified memory with PPC instructions, but can't debug processes in dynamic mode, due to breakpoint inserting is not support.

III. DYNAMIC DEBUGGING SYSTEM

Traditional methods can do static analysis. However, when dynamic debugging is needed to analyze procedures or functions in IOS, it does not work. It's required to launch Cisco IOS when dynamic debugging. With the analysis of the structure of IOS images, IOS can't be launched in the regular PowerPC or MIPS platforms but only Cisco Routers. Since Cisco router models are quite different with high cost of equipments, hardware emulation technology might be a better choice. At first, a router emulation environment must be established. Then, remote debug module will be inserted into virtual machine instance, it is used for procedure of dynamic debugging controlling and user request processing. At last, communicate with remote debugger UI(User Interface) using debugging protocol.

A. *Modification dynamips emulator*

Dynamips[4] is a good performance hardware emulation program, which can emulate underlying hardware of Cisco routers. It is able to run on linux x86(or x86_64) and window platform. It also can emulate several types of Cisco routers. But the emulator can't debug the IOS at runtime. With the Cisco IOS is running on a layer called Hypervisor, which can monitor the running status and access the resources of virtual machines, it opens the door to dynamic debugging.

If required to analyze procedure of process in IOS, it should modify virtual CPU to support Single step debugging. The Control status of Virtual CPU consist of RUNNING, SUSPENDED and HALTED, and one status called SINGLESTEP should be inserted for dynamic debugging. Of course, Hypervisor should control all of the status at runtime.

B. *Dynamic debugging system*

To establish dynamic debugging system, much more work must be done. Dynamic debugging system mainly consists of two parts: One is Cisco router virtual machine, the other is remote debugger UI. Remote debug module will be inserted into virtual machine, which is the bridge between researchers and virtual machines. It will handle the user commands posted by communication module in Hypervisor, then, send response information of debugging to users. Remote debug module should have the ability to handle the users' error commands. To construct a virtual machine, the following work should be done:

1) Hardware analysis,analyzing the function and mechanism of hardware.

2) Hardware emulator, using the program to emulate hardware based on 1).

3) Instruction set translation, making virtual machine's instructions execute on host machine

4) Modules management, organizing all the hardware emulator, making the whole system running as real equipment.

5) Launching OS on virtual machine.

To transit various commands and response information, remote debug UI should be created. A subset of commands defined by users has been implemented. This is the list of implemented commands on the current dynamic debugging system:

show reg, show the values of Virtual CPU registers
set reg, Sets the values of Virtual CPU registers.
dump, show the instructions of specified memory.
read, read memory information of specified.
write, write a 32-bit integer into specified memory.
insert breakpoint , insert breakpoint into processed.
remove breakpoint, remove exist breakpoint.
suspend, suspend virtual CPU
single, enter step by step debugging mode
resume, quit from single step or suspend status.

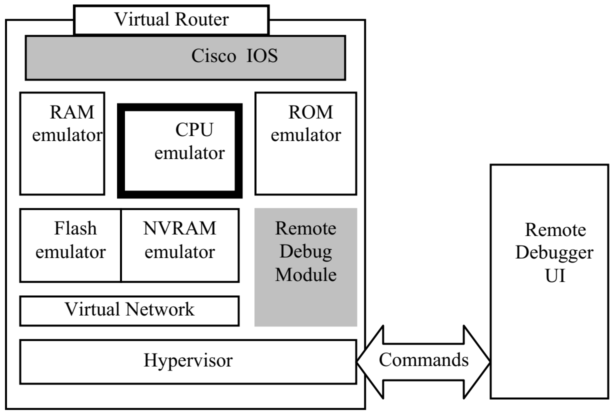


Figure 3 Dynamic Debugging System

IV. Breakpoint

The primary function of a router is to route packets. For each packet received by the router, it must determine whether or not it knows how to reach the destination network and, if so, where the packet must be sent to continue it on its way. Meanwhile, some network services provided to manage the equipments, such as http, telnet, tftp and so on. If an error has occurred during network transmission, it's difficult to detect, reproduce for analysis and debugging. In dynamic debugging procedure, breakpoint is one of the most basic and most important technology. Reasonable breakpoints can be set to analyze specified processes. Simple breakpoints, such as hardware breakpoint, conditional breakpoint, memory access breakpoint, do not work during network communication processes analysis. A new method of inserting breakpoint is present to solve the problem.

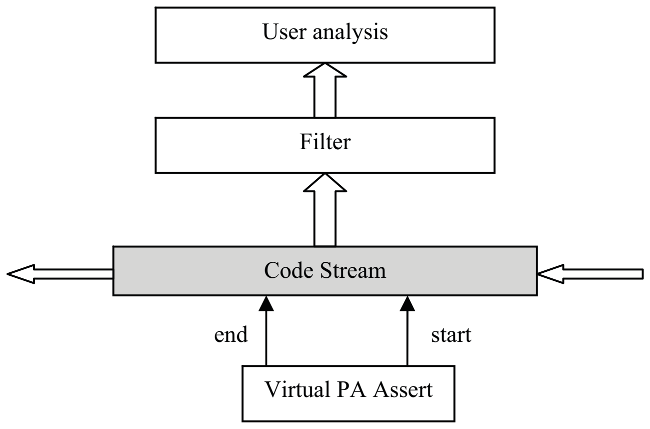


Figure 4 Code stream retrieved

Using dynamic debugging system to load Cisco IOS, inserting assert into virtual PA(Port Adapters) driver. The assert work with two mode: On is Code Stream Retrieve mode(CSR), and the other is Single Step Analysis(SSA) mode.

1)CSR: if the specified network event occurs, it will start to record code stream, until another end event occurring. Since Cisco IOS is multitasking operating system, record data may have redundancy, it needs to set a filter to drop redundancy information, and post the result for user analysis.

2)SSA: if the specified network event occurs, it will suspend the virtual CPU, and store registers and memory status, preparing for dynamic single step debugging.

V. Conclusion

As the essential equipment of network communication, Cisco router's security can not be ignored. Cisco IOS is the focus of security researchers. We have presented a dynamic debugging system for Cisco IOS analysis, and a new solution of inserting breakpoints. This system can load and analyze multiversion IOS. The CSR filter rules are not overall. Further work may include code stream filter rules, and research of embedded virtualization for system analysis.

[1] Gyan Chawdhary, Senior Consultant Varun Uppal, Senior Consultant.Cisco Shellcodes.Black Hat 2008.

[2] Felix 'FX' Lindner. Cisco IOS Router Exploitation. Black Hat 2009.

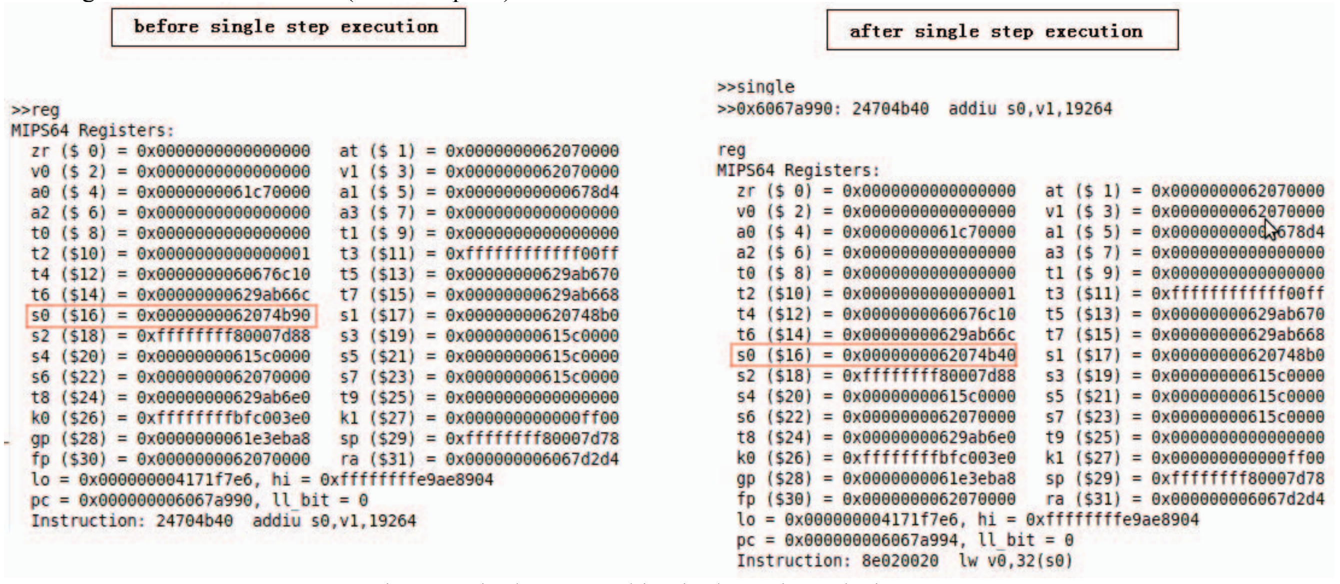
[3] Muñiz, Sebastian.Ortega, Alfredo. Fuzzing and Debugging Cisco IOS. BlackHat EU 2011

[4] Christophe Fillot. Cisco 7200 simulator. URL : www.ipflow.utc.fr/index.php/Cisco_7200_Simulator.

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

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

[7] Dr. Marc E. Fiuczynski.Virtual Machine Monitors.2009.



before single step execution

```
>>reg
MIPS64 Registers:
  zr ($ 0) = 0x0000000000000000   at ($ 1) = 0x0000000062070000
  v0 ($ 2) = 0x0000000000000000   v1 ($ 3) = 0x0000000062070000
  a0 ($ 4) = 0x0000000061c70000   a1 ($ 5) = 0x00000000000678d4
  a2 ($ 6) = 0x0000000000000000   a3 ($ 7) = 0x0000000000000000
  t0 ($ 8) = 0x0000000000000000   t1 ($ 9) = 0x0000000000000000
  t2 ($10) = 0x0000000000000001   t3 ($11) = 0xffffffffffff00ff
  t4 ($12) = 0x0000000060676c10   t5 ($13) = 0x00000000629ab670
  t6 ($14) = 0x00000000629ab66c   t7 ($15) = 0x00000000629ab668
  s0 ($16) = 0x0000000062074b90   s1 ($17) = 0x00000000620748b0
  s2 ($18) = 0xffffffff80007d88   s3 ($19) = 0x00000000615c0000
  s4 ($20) = 0x00000000615c0000   s5 ($21) = 0x00000000615c0000
  s6 ($22) = 0x0000000062070000   s7 ($23) = 0x00000000615c0000
  t8 ($24) = 0x00000000629ab6e0   t9 ($25) = 0x0000000000000000
  k0 ($26) = 0xffffffffbfc003e0   k1 ($27) = 0x000000000000ff00
  gp ($28) = 0x0000000061e3eba8   sp ($29) = 0xffffffff80007d78
  fp ($30) = 0x0000000062070000   ra ($31) = 0x000000006067d2d4
  lo = 0x000000004171f7e6, hi = 0xfffffffe9ae8904
  pc = 0x000000006067a990, ll_bit = 0
  Instruction: 24704b40  addiu s0,v1,19264
```

after single step execution

```
>>single
>>0x6067a990: 24704b40  addiu s0,v1,19264

reg
MIPS64 Registers:
  zr ($ 0) = 0x0000000000000000   at ($ 1) = 0x0000000062070000
  v0 ($ 2) = 0x0000000000000000   v1 ($ 3) = 0x0000000062070000
  a0 ($ 4) = 0x0000000061c70000   a1 ($ 5) = 0x00000000000678d4
  a2 ($ 6) = 0x0000000000000000   a3 ($ 7) = 0x0000000000000000
  t0 ($ 8) = 0x0000000000000000   t1 ($ 9) = 0x0000000000000000
  t2 ($10) = 0x0000000000000001   t3 ($11) = 0xffffffffffff00ff
  t4 ($12) = 0x0000000060676c10   t5 ($13) = 0x00000000629ab670
  t6 ($14) = 0x00000000629ab66c   t7 ($15) = 0x00000000629ab668
  s0 ($16) = 0x0000000062074b40   s1 ($17) = 0x00000000620748b0
  s2 ($18) = 0xffffffff80007d88   s3 ($19) = 0x00000000615c0000
  s4 ($20) = 0x00000000615c0000   s5 ($21) = 0x00000000615c0000
  s6 ($22) = 0x0000000062070000   s7 ($23) = 0x00000000615c0000
  t8 ($24) = 0x00000000629ab6e0   t9 ($25) = 0x0000000000000000
  k0 ($26) = 0xffffffffbfc003e0   k1 ($27) = 0x000000000000ff00
  gp ($28) = 0x0000000061e3eba8   sp ($29) = 0xffffffff80007d78
  fp ($30) = 0x0000000062070000   ra ($31) = 0x000000006067d2d4
  lo = 0x000000004171f7e6, hi = 0xfffffffe9ae8904
  pc = 0x000000006067a994, ll_bit = 0
  Instruction: 8e020020  lw v0,32(s0)
```

Figure 5 Single step tracking in dynamic analysis system