Fuzzing and Debugging Cisco IOS
Syscan 2011

Sebastian Muiz, Alfredo Ortega

Groundworks Technologies

April 28, 2011
Cisco IOS Architecture

Debugger internals

Dynamips modification
  - GDB support
  - IDA Pro support

Shortcomings of self-checking routines

Demos:
  - Malware analysis
  - Fuzzing example
Cisco IOS architecture

- Single binary image
- Shared single address space
- Cooperative priority-based scheduler

Figure: Cisco IOS process memory
Built-in GDB server

- Used by Cisco developers and support engineers
- Works over Telnet, SSH and Serial console
- Slightly different GDB protocol

<table>
<thead>
<tr>
<th></th>
<th>Examine</th>
<th>Debug</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Registers</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Write Registers</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Read Memory</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Write Memory</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Freeze OS</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Remote</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

Figure: GDB debugging modes
Dynamips emulator

- Created by Christophe Fillot
- Runs on Windows, Linux and Mac OS X.
- Equivalent to QEMU/Bochs
- Implements MIPS/PowerPC architecture and Cisco hardware
- Supports the following models:
  - (a) 7200
  - (b) 36XX
  - (c) 2691
  - (d) 3725
  - (e) 3745
  - (f) 26XX
  - (g) 17XX

Cisco IOS architecture
Debugger Architecture
Analyzing Pros and Cons
Use case: IOS malware
Use case: ROMMON debugging
Use cases: Fuzzer
Wrapping up

Virtual Machine Debugger internals

Virtual Machine Debugger internals

Dynamips

PowerPC

GDB Server

MIPS

Memory Controller

Special Hard

FPGA

PCI WIC

NM

GDB Protocol

CPU/Memory instrumentation
No JIT support
Supported commands
- Read/Write CPU Registers
- Read/Write Memory
- Set/Unset Breakpoints
Any standard GDB client supported

Figure: GDB Server embedding
Pros and Cons of Virtual Machine Debugger

Pros:
- Complete isolation (almost!)
- Cost-effective
- Controlled debugging environment
- Bug-hunter friendly

Cons:
- Not 100% exact emulation
- Not all models or hardware compatible
- Findings need double-check with physical device
- You need your own IOS images (legally).
Why isolation is good?

Using built-in GDB

- GDB Client
- Expected (fake) Bytes
- Read_Memory Request

Cisco IOS

- Malware
- Built-In GDB Stub
- Original memory Mirror

Dynamips GDB server

- GDB Client
- Malware

Cisco IOS

- Read_Memory
- Malware memory dump
- GDB Stub
- DYNAMIPS

Remember: NEVER analyze malware inside an infected host.
I don’t need this, I have the *verify* command

Cisco Response on IOS rootkits[^2]:

- Maintain chain of trust when verifying IOS images
- Verify IOS image in external host, or before booting it
- Use the MD5 File Validation command “verify” on **Loaded** image:

Using the MD5 File Validation Feature

“The MD5 File Validation feature, added in Cisco IOS Software Releases 12.2(4)T and 12.0(22)S, allows network administrators to calculate the MD5 hash of a Cisco IOS software image file that is loaded on a device.”

Shortcomings of self-checking routines

Malware-affected analysis
- Cisco IOS
- Malware
- MD5 CHK
- Login routine
- GDB server

Verify CLI command
User expected MD5 checksum (fake)

Clean analysis
- External Trusted environment
- MD5 Tool
- Calculate
- Result
Hash
Cisco IOS

Remember: NEVER verify code inside an infected host.
Use cases: IOS malware

- Demo: Backdoored IOS installation
- Not trivial to analyze (Many IOS variations)
- At least, possible:

Demo!
Use case: ROMMON debugging

- ROMMON: Cisco bootloader
- Very easy to verify and analyze (less variations)
- Read-only in some models
- Contains a basic but privileged debugger
- ROMMON itself can be debugged by Dynamips

Demo!

---

3 Felix 'FX' Lindner, 25c3, Cisco IOS - Attack & Defense
Fuzzing requirements

- Correct exception handling
- Reproducible test-cases
- Logging
- Desirable: Debugging environment (for post-analysis)
Fuzzing timing diagram

- **Fuzzer**
  - Start
  - Fuzz case N
  - Signal
  - Get Regs
  - Registers
  - Restart
  - Fuzz case N+1

- **GDB**
  - Start
  - Signal
  - Get Regs
  - Registers
  - Restart
  - Fuzz case N+1

- **Dynamips**
  - Start
  - Exception
  - Restart

- **Log**
Example fuzzer

- Attack surface via Protocol fuzzer (ftp)
- Trivial test-case generation (just an example!)
Fuzzer Demo

Demo!
Triggered Vulnerability

- Cisco Security Advisory: Multiple Vulnerabilities in the IOS FTP Server (cisco-sa-20070509-iosftp)
- 30 FTP commands, **remote code execution** on 16: (USER, CWD, DELE, RNFR, STOR, NLST, APPE, MKD, RMD, STOU, RETR, LIST, STAT, MDTM, SIZE, and HELP)
- Patched in 2007: **Completely remove all FTP server code**
How secure is this debugger?

Very.
Can be used in a production environment to analyze malicious code?

No
Dynamips contains emulation bugs.

Demo!
Future Development

- Honeypots
- Malware analysis Lab
- Exploit Dev (with certain restrictions)
- Duplicate exact memory behaviour (typical VMs problems)
- Secure host isolation (squash Dynamips bugs)
Questions?

Via email:

- aortega@groundworkstech.com
- smuniz@groundworkstech.com

Please download:


Published under the GNU General Public Licence (GPL)
The End

Thanks for listening!