Recurity Labs

# Developments in Cisco IOS Forensics

Felix 'FX' Lindner

BlackHat Briefings
Las Vegas, August 2008

*Invent & Verify*

Recurity Labs

# Agenda

- IP Routing Infrastructure and Cisco IOS
- Cisco IOS Internals
- Debugging and Post Mortem Analysis Today
- A New Analysis Approach
  - Proposal
  - Features
  - Challenges
- Public Offer
- Future Work

*Invent & Verify*

# IP Routing Infrastructure

- The Internet and corporate networks almost exclusively run on the Internet Protocol
  - IP Version 4 is still prevalent protocol
  - IP Version 6 coming up very slowly
- The design of IP requires intelligent nodes in the network to make routing decisions
  - This is a design principle of the protocol and cannot be changed
  - "Flat" networks have their own issues

# IP Infrastructure & Security

- All security protocols on top of IP share common design goals:
  - Guarantee end-to-end integrity (some also confidentiality) of the traffic
  - Detect modification, replay, injection and holding back of traffic
  - Inform the upper protocol layers
- None of them can recover from attacks rooted in the routing infrastructure
  - Security protocols cannot influence routing

Recurity Labs

# Infrastructure Monoculture

- Cisco Systems' routing platforms form the single largest population of networking equipment today
  - Equivalently distributed in the Internet core, government and corporate networks
  - Many different hardware platforms with different CPUs
  - Large investment sums bound to the equipment
  - Hard to replace
  - All run basically the same operating system
- Protecting this infrastructure is critical
- Therefore, in-depth analysis and diagnostics are of paramount importance

# Cisco IOS

- Cisco® Internetwork Operating System®
- Monolithic operating system
- Compile-time linked functionality – the 3 dimensional complexity of IOS
  - Platform dependent code
  - Feature-set dependent code
  - Major, Minor and Release version dependent code
- Several ***tens of thousands different*** IOS images used in today's networks
  - Over 10.000 still officially supported

# Inside Cisco IOS

- One large ELF binary
- Essentially a large, statically linked UNIX program
  - Loaded by ROMMON, a kind-of BIOS
- Runs directly on the router's main CPU
  - If the CPU provides virtual memory and privilege separation (for example Supervisor and User mode on MIPS), it will not be used

# Inside Cisco IOS

- Processes are rather like threads
  - No virtual memory mapping per process
- Run-to-completion, cooperative multitasking
  - Interrupt driven handling of critical events
- System-wide global data structures
  - Common heap
  - Very little abstraction around the data structures
  - No way to force abstraction

# The IOS Code Security Issue

- 12.4(16a) with enterprise base feature set consists of 25.316.780 bytes binary code!
  - This is a 2600 with PowerPC CPU
  - Not including 505.900 bytes firmware for E1T1 and initialization
- All written in plain C
- Sharing the same address space
- Sharing the same heap
- Sharing the same data structures
- Sharing millions of pointers

Recurity Labs

# The IOS Code Security Issue

- A single mistake in the most unimportant piece of code can influence anything on the system, including kernel, security subsystems and cryptographic code.
- Therefore, **everything** on IOS is a good target for remote code execution exploits in kernel context.

Invent & Verify

# Isn’t Cisco aware of that?

- Cisco recently started the distribution of the next generation IOS-XR
  - Commercial QNX microkernel
  - Real processes (memory protection?)
  - Concurrent scheduling
  - Significantly higher hardware requirements (as in Cisco 12000 !)
- People never use the latest IOS
  - Production corporate networks usually run on 12.1 or 12.2, which 12.5 is already available
  - Not even Cisco’s own engineers would recommend the latest IOS release to a customer
  - That only covers people actively maintaining their network, not everyone running one

# Just, how often are routers hacked?

- Keynote speaker Jerry Dixon at BlackHat Washington DC mentioned not updated routers as a cause for concern
  - Do you know how expensive that is?
- Old vulnerabilities like the HTTP level 16 bug are still actively scanned for
  - The router is used as a jump pad for further attacks
- TCL backdoors are commonly used
- Patched images are not rare
  - IOS images cost money
  - People will use images from anywhere
  - Patching images is not hard
- Lawful Interception is its own can of worms
  - The router's operator is not supposed to know that LI is performed
  - Who watches the watchers?

# And the future?

- Ever noticed attackers take on the target with the lowest efforts required and the highest return of invest?
  - Windows became just a lot harder
  - UNIXes are hardened, even OS X
  - Infected PCs leave obvious traces
- The question is not:
  "Will routers become a target?"
- The question should be:
  "Do we want to know when they did?"
- Check the speaking schedule: 3 IOS talks here, 2 of them on attack methods

# Summary – Part I

- A significant share of the Internet, governmental and corporate networks runs on:
  - one out of several tens of thousands of builds
  - of more or less the same code base
  - in a single process environment

... and we cannot bypass it, even if we could tell that it's compromised

Next question: **How can we even tell?**

# Error Handling and Recovery

- The software architecture of IOS dictates how exception handling has to be done
  - Remember, IOS is like a large UNIX process
  - What happens when a UNIX process segfaults?
- Upon an exception, IOS can only restart the entire system
  - Even on-board, scheduled diagnostic processes can only forcefully crash the system

Invent & Verify

# Crash Cause Evidence

- Reboot is a clean recovery method
- Reboot destroys all volatile evidence of the crash cause
  - Everything on the router is volatile!
  - Exception: startup configuration and IOS image
- Later IOS releases write an information file called “crashinfo”
  - Crashinfo contains very little information
  - Contents depend on what IOS thought was the cause of the crash

# Runtime Evidence

- Crashinfo is only written upon device crashes
- Successful attacks don't cause device crashes
- The available methods are:
  - Show commands
  - Debug commands
  - SNMP monitoring
  - Syslog monitoring

# Show Commands

- IOS offers a plethora of inspection commands known as the "show" commands
  - Requires access to the command line interface
- Geared towards network engineers
- Thousands of different options and versions
- Almost no access to code
  - 12.4 even limits memory show commands

# Debug Commands

- “debug” enables in-code debugging output
- Debug output has scheduler precedence
  - Too much debug output halts the router
  - Not an option in production environments
- Enabling the right debug output is an art
  - Turn on the wrong ones and you see very little
  - Turn on too many and the router stops working
  - Commands depend on the IOS version
- For debug commands to be useful, you have to know what you are looking for **before it happens**
  - Not very useful for security analysis

Recurity Labs

# SNMP and Syslog Monitoring

- Commonly accepted method for monitoring networking equipment
- SNMP depending on the implemented MIB
  - Geared towards networking functionality
  - Very little process related information
- Syslog is about as useful for security monitoring on IOS as it is on UNIX systems
- Both generate continuous network traffic
- Both consume system resources on the router
- Then again, someone has to read the logs.

Recurity Labs

# Summary – Part II

- Identifying compromised routers using today's tools and methods is hard, if not impossible.
- There is not enough data to perform any post mortem analysis of router crashes, security related or not.
- We cannot distinguish between a functional problem, an attempted attack and a successful attack on infrastructure running IOS.

Invent & Verify

# A (not so) New Approach

- We need the maximum amount of evidence
  - A full snapshot of the device is just enough
- We don’t need it continuously
  - We need it on-demand
  - We need it when the device crashes
- We need an independent and solid analysis framework to process the evidence
  - We need to be able to extend and adjust it

# Getting the Evidence

- Cisco IOS can write complete core dumps
  - Memory dump of the main memory
  - Memory dump of the IO memory
  - Memory dump of the PCI memory (if applicable)
- Core dumps are written in two cases
  - The device crashes
  - The user issues the “write core” command

# Core Dump Destinations

- IOS supports various destinations
  - TFTP server (bug!)
  - FTP server
  - RCP server
  - Flash file system (later IOS releases)
- Core dumps are enabled by configuration
  - Configuration commands do not differ between IOS versions
  - Configuration change has no effect on the router's operation or performance

# Core Dump Enabled Infrastructure

- Configure all IOS devices to dump core onto one or more centrally located FTP servers
  - Minimizes required monitoring of devices: A router crashed if you find a core dump on the FTP server
  - Preserves evidence
  - Allows crash correlation between different routers
- Why wasn't it used before?
  - Core dumps were useless, except for Cisco developers and exploit writers.

# Analyzing Core Dumps

Disclaimer:

- Any of the following methods can be implemented in whatever your preferred programming language is.
- This presentation will be centric to our implementation: Recurity Labs CIR.

# Core Dump Analyzer Requirements

- Must be 100% independent
  - No Cisco code
  - No disassembly based analysis
- Must gradually recover abstraction
  - No assumptions about anything
  - Ability to cope with massively corrupted data
- Should not be exploitable itself
  - Preferably not written in C

Invent & Verify

# The Image Blueprint

- The IOS image (ELF file) contains all required information about the memory mapping on the router.
  - The image serves as the memory layout blueprint, to be applied to the core files
  - We wish it were as easy as it sounds
- Using a known-to-be-good image also allows verification of the code and read-only data segments
  - Now we can easily and reliably detect runtime patched images

# Heap Reconstruction

- IOS uses one large heap
- The IOS heap contains plenty of meta-data for debugging purposes
  - 40 bytes overhead per heap block in IOS up to 12.3
  - 48 bytes overhead per heap block in IOS 12.4
- Reconstructing the entire heap allows extensive integrity and validity checks
  - Exceeding by far the on-board checks IOS performs during runtime
  - Showing a number of things that would have liked to stay hidden in the shadows ☹

# Heap Verification

- Full functionality of “CheckHeaps”
  - Verify the integrity of the allocated and free heap block doubly linked lists
- Find holes in addressable heap
  - Invisible to CheckHeaps
- Identify heap overflow footprints
  - Values not verified by CheckHeaps
  - Heuristics on rarely used fields
- Map heap blocks to referencing processes
- Identify formerly allocated heap blocks
  - Catches memory usage peaks from the recent past

*Invent & Verify*

# Process List

- Extraction of the IOS Process List
  - Identify the processes' stack block
    - Create individual, per process back-traces
    - Identify return address overwrites
  - Obtain the processes' scheduling state
  - Obtain the processes' CPU usage history
  - Obtain the processes' CPU context
- Almost any post mortem analysis method known can be applied, given the two reconstructed data structures.

# TCL Backdoor Detection

- TCL scripting is available on later Cisco IOS versions
- TCL scripts listening on TCP sockets
  - Well known method
  - Used to simplify automated administration
  - Used to silently keep privileged access to routers
  - Known bug:
    not terminated when the VTY session ends (fixed)
  - Simple TCL backdoor scripts published
- CIR can extract all TCP script chunks from IOS heap and dump them for further analysis
  - There is still some reversing work to do

# Random Applications

- Find occasional CPU hogs
- Detect Heap fragmentation causes
- Determine what processes where doing
- Finding attacked processes
  - See examples (Semi-DEMO)
- Research tool
  - Pointer correlation becomes really easy
  - Essential in a shared memory environment

# IOS Packet Forwarding Memory

- IOS performs routing either as:
  - Process switching
  - Fast switching
  - Particle systems
  - Hardware accelerated switching
- Except hardware switching, all use IO memory
  - IO memory is written as separate code dump
  - By default, about 6% of the router's memory is dedicated as IO memory
    - In real world installations, it is common to increase the percentage to speed up forwarding
- Hardware switched packets use PCI memory
  - PCI memory is written as separate core dump

# IO Memory Buffers

- Routing (switching) **ring buffers** are grouped by packet size
  - Small
  - Medium
  - Big
  - Huge
- Interfaces have their own buffers for locally handled traffic
- IOS tries really hard to not copy packets around in memory
- New traffic does not automatically erase older traffic in a linear way

# Traffic Extraction

- CIR dumps packets that were process switched by the router from IO memory into a PCAP file
  - Traffic addressed to and from the router itself
  - Traffic that was process switching inspected
    - Access List matching
    - QoS routed traffic
- CIR could dump packets that were forwarded through the router too
  - Reconstruction of packet fragments possible
  - Is it desirable?

# Advanced Traffic Extraction

- Writing core to a remote server uses IO memory
  - Overwrites part of the traffic evidence
- CIR can use a GDB link instead of a core dump
  - Serial GDB protocol allows direct access to router memory via the console
  - Uses Zynamics GDB debug link
- Disconnecting all network interfaces preserves IO and PCI memory contents
  - Using GDB halts the router
- All data is preserved – useful for emergency inspections

# Traffic Extraction Applications

- Identification of attack jump pad routers
- 0day identification against systems on segmented network interfaces
  - If you got the packet, you got the 0day
- Spoofing attack backtracking
  - One hop at the time, obviously
- LE detection

# Reality Check: March's Vulnerabilities

- "Cisco IOS Virtual Private Dial-up Network Denial of Service Vulnerability"
  - Memory exhaustion / leak
  - Visible by heap usage analysis
- "Cisco IOS User Datagram Protocol Delivery Issue For IPv4/IPv6 Dual-stack Routers"
  - "The show interfaces command can be used to view the input queue size to identify a blocked input interface."
  - CIR could output all the packets that are still in the queue, even allowing source identification
- "Vulnerability in Cisco IOS with OSPF, MPLS VPN, and Supervisor 32, Supervisor 720, or Route Switch Processor 720"
  - see above

# Challenges

- The analysis framework has to handle the complexity of the Cisco IOS landscape
  - Hardware platforms
  - Image versions
  - Any-to-Any relation!
- CIR is currently IOS feature set independent
- CIR successfully tested against IOS 12.1 – 12.4
- Official support starts with:
  - Cisco 2600
- Internal testing already covers:
  - Cisco 1700
  - Cisco 2691
  - Cisco 6200
- The platform is the major source of work, testing and verification

# Summary – Part III

- Writing core dumps is a viable method for obtaining IOS evidence when it is needed.
  - The evidence includes forwarded and received packets.
- An independent analysis framework can distinguish between bugs and attacks, enabling real forensics on IOS routers.
- Recurity Labs’ CIR already reliably identifies many types of attacks and IOS backdoors.
  - CIR is work-in-progress
  - CIR’s future depends on the feedback we receive from the community.

# Availability

1. CIR Online Service (free)
2. CIR Rootkit Detector (free)
3. CIR Professional (non-free)

# CIR Online

- An analysis framework's quality is directly related to the amount of cases it has seen
  - CIR needs a lot more food to grow up
  - We want to provide it to everyone while constantly developing and improving it
- Free Service: **http://cir.recurity-labs.com**
  - Processing on our servers
  - Always using the latest version
  - Right now, CIR Online runs in BETA state

# CIR Rootkit Detector

- Detection of image modification
- Detection of runtime code modification
- Support for all access layer platforms
- Freely available at http://cir.recurity-labs.com
- Currently in BETA state

# At the end, it's all up to you!

- We think CIR could be useful
  - For the networking engineer
  - For the forensics professional
  - To finally know the state of our infrastructure
- We know what we can do
- We need advise on where you want this tool to be in the future

Recurity Labs

# cir.recurity-labs.com

Felix ´FX´ Lindner
Head

fx@recurity-labs.com

Recurity Labs GmbH, Berlin, Germany
http://www.recurity-labs.com

*Invent & Verify*