We're Watching You:
An analysis of IP cameras through their firmware

Charles Monett
OISF - November 2016
Warning/Disclaimers

Please use this knowledge only for good and for your own devices.

Disclaimers:

• All trademarks/etc. used in this presentation are property of their respective owners.
• All testing performed in a controlled environment.
Outline

• Introduction/Key Points/Takeaways
• Example 1: Conference Camera + demos
• Example 2: Security Camera + demos
• Some good news
• Usage/Research
• Wrap-up/Q&A
Introduction

• Did product research for IP cameras
  • Was looking at more upmarket cameras (above Foscam)
  • Part of it involved looking at firmware updates
  • Was expecting a bit more resistance to modification

• Not just tearing apart firmware
  • Putting the knowledge to good use
Key Points & Takeaways

• Key points
  • Network appliance design is hard to get right
  • Sometimes we can use it for our advantage

• Takeaways
  • Introduction to tools/methods/processes
  • Relevant applications that highlight security issues
  • A greater understanding of IP cameras
Example 1: Conference Camera

Vaddio ClearVIEW HD-USB Slotcard

- Early-generation HD conference camera
- Runs Ångström distribution of Linux
- Powered by TI DaVinci DM368 platform
  - (ARM926EJ-S CPU)
- Provides HTTP/telnet/network streaming services
Issues:

• Network:
  • Cleartext administration interfaces (HTTP/telnet), no alternatives

• Firmware:
  • Can be modified (in entirety) while running
  • Firmware obfuscation is minimal (byte-reversal)
  • Can be updated with modified firmware
The update process

At System part of Administration menu:

• User uploads firmware

• Package is decoded and unpacked to scratch space.

• Bootloader update script is executed

• System update script is executed

• System verifies functionality, and:
  • If good, commits update.
  • If not good, reverts to existing firmware.
The firmware package

Relatively trivial unpacking. No binwalk needed.

A byte-reversed, base64 encoded zipfile containing:

- Bootloader
- Updated environment
- Support scripts
- Python Interpreter
- Other goodies
Extracting & re-packing firmware

• Extracting:
  • Undo byte-reversal
  • Uudecode file
  • Extract resulting zip into a directory

• Re-packing:
  • Create zip archive
  • Uuencode file
  • Redo byte-reversal

If all goes well, it will accept your changes.
Demonstration
Example 2: Security Camera

Canon VB-H41

• Pan/Tilt/Zoom IP camera
• Proprietary OS (Linux-based)
• Powered by DIGIC DV III Platform
  • (ARMV6TEJ-based CPU)
• SD slot for event recording
Issues:

• Network:
  • None (if running as intended)

• Firmware:
  • Default administrative account is root
  • Running software can be easily updated
  • Arbitrary tasks can be invoked with cron job
  • Easily unpacked, no apparent signature check in bootloader (?)

• Enough space available to run Debian in a chroot.
  • Remember that SD card slot?
  • Applications only limited by binutils
The firmware package

Courtesy of binwalk, we get the following:

• 128 bytes: Header (for this series)

• Remainder is a tarred CPIO archive containing:
  • Canon DryOS Bootloader (boot.bin)
  • Data (cmr.dat)
  • SquashFS filesystem (mtd4fs, ro) – core OS
  • JFFS2 'appfs' filesystem (main, mtd9fs, rw) – external apps
  • Linux Kernel (zImage)
  • MD5 sum of above items
Extracting firmware

- Extracting:
  - Remove header
  - Extract gunzip archive
  - Extract resulting cpio archive in a directory
  - Extract other filesystems
    - SquashFS (core OS):
      - unsquashfs mtd4fs.squashfs
    - JFFS2:
      - Extract/unpack to a loopback device
      - Use Jefferson (jffs2 extraction tool)
Demonstration
Good News

VB-H41:

• SSL is available (which raises the bar)
• Some parts of firmware resist modification.
• Some sanitization is performed (such as system logs)

HD-USB Slotcard:

• Obtaining root is not straightforward
• Outbound network traffic is restricted by default
• Subsequent generation products more protected from altered firmware
Usage

• For good/neutral:
  • Fix features (e.g. stepped Pan/Tilt)
  • Extend functionality to cross-platform clients
  • Ansible integration (depending on security model)

• For [not good]:
  • Unwanted surveillance
    • Redirect/Copy streams to external sources
  • Jumping-off point to other devices
    • Other accounts (crafted alert e-mail)?
    • Compromise other devices with the camera
Further research

• Obtain access without having to look over the wire
• Extract keys from other devices (via JTAG, TTL serial, etc.)
• Other firmware (Canon, AXIS, others)
• Addressing issues with Canon firmware:
  • Properly extracting squashfs
  • Building firmware package (squashfs/jffs2-appfs)
• NFS volume mounting off a camera (TI SDK kernel modules, perhaps?)
  • Stream straight to networked storage.
Questions?
Resources:

• Angstrom Distribution: http://www.angstrom-distribution.org/

• Binwalk: http://www.binwalk.org/

• SquashFS:
  • http://tldp.org/HOWTO/SquashFS-HOWTO/mksqoverture.html

• JFFS2 extraction:
  • https://github.com/sviehb/jefferson
  • http://linux-7110.sourceforge.net/howtos/netbook_new/x1125.htm

• Multistrap: https://wiki.debian.org/Multistrap

• TI DM365/368 SDK: http://www.ti.com/tool/linuxdv-sdk-dm36x

• Unpacking scripts: http://github.com/cm-code/firmware-scripts
Thank you.