IoT Security: Firmware Testing

Yue Duan
Outline

● Research paper:
  ○ A Large Scale Analysis of the Security of Embedded Firmwares
  ○ AVATAR: A Framework for Dynamic Security Analysis of Embedded Systems’ Firmwares
  ○ Firmalice - Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware
A Large Scale Analysis of the Security of Embedded Firmwares

Andrei Costin, Jonas Zaddach, Luca Bruno,
Aurélien Francillon, Davide Balzarotti

USENIX SEC 2014
Motivation

Routers

Based on the source code of the HTML pages and some Shodan search results, D-Link devices are likely affected:

- DIR-100
- DIR-120
- DI-624S
- DI-524UP
- DI-604S
- DI-604UP
- DI-604+
- TM-G5240

Additionally, several Planex routers also appear to use the same firmware:

- BRL-04R
- BRL-04UR
- BRL-04CW

You stay classy, D-Link.

Printers

Networked Printers at Risk

By Jimmy Shah on Dec 30, 2011

Multifunction printers (MFPs) have been common in offices for years. They let employees print, scan, and copy documents. Two separate talks at the 28th Chaos Communications Congress (28c3) show how attackers can infect these trusted office devices.

Hacking MFPs

In Andrei Costin’s presentation “Hacking MFPs,” he covered the history of printer and copier hacks from the 1960s to today. The meat of the talk concerned executing remote code on an MFP using crafted PostScript. Just printing a particular document can get code to run on the machine. Previous research proof of concepts have done exactly that, once with a specially designed Word document and once with a Java applet.
Motivation

- Each of above is a result of an individual analysis
- Manual and tedious efforts, Does not scale
Motivation

● Goal: Perform a large scale analysis to provide a better understanding of the problem
● Problems with Large Scale Analysis:
  ○ Heterogeneity of
    ■ Hardware, architectures, OSes
    ■ Users, requirements
    ■ Security goals
  ○ Manual analysis does not scale, it requires
    ■ Finding and downloading the firmwares
    ■ Unpacking and performing initial analysis
    ■ Re-discovering the same or similar bug in other firmwares
Existing Approaches

- Test on real devices [Bojinov et.al CCS’09]
  - accurate results
  - does not scale well

- Scan devices on the Internet
  - Large scale testing [Cui et.al ACSAC’10]
    - Can only test for known vulnerabilities
    - Blackbox approach
  - More is too intrusive [Census2012]
Proposed Approach

- Collect a large number of firmware images
- Perform broad but simple static analysis
- Correlate across firmwares
- Advantages:
  - no intrusive online testing
  - no device needed
  - scalable
- But many **challenges**
Challenge 1: Dataset

- No large scale firmware dataset yet
  - As opposed to existing datasets in security or other CS research areas
- We collected a subset of the firmwares available for download
- Many firmwares are not publicly available
  - Not intended to have an upgrade
  - Require product purchase and registration
- www.firmware.re project
Challenge 2: Firmware Identification

Clearly a firmware

Uncertain

Clearly not a firmware
Challenge 3: Unpacking & Custom Formats

- How to reliably unpack and learn formats?
- E.g., vendor provides a .ZIP 'firmware package'
  - .ZIP→.EXE + .PS
    - .EXE→self-extracting archive
      - Extract more or not?
      - Turns out to contain a printer driver inside
    - .PS→ASCII85 stream→ELF file that could be:
      - A complete embedded system software
      - An executable performing the firmware upgrade
      - A firmware patch
- Often, a firmware image→just 'data' binary blob
Approach to Unpacking & Custom Formats

- Used BAT (Binary Analysis Toolkit)
  - Extended it with multiple custom unpackers
  - Continuous development effort
- Often, a firmware image → just 'data' binary blob
  - File carving required
  - Brute force at every offset with all known unpackers
- Heuristics for detecting when to stop
Challenge 4: Results Confirmation

- An issue found statically
  - May not apply to a real-device
  - Cannot guarantee exploitability
  - E.g., vulnerable daemon present but never started

- Issue confirmation is difficult
  - Requires advanced analysis (static & dynamic)
  - Often requires real embedded devices
  - Does not scale well in heterogeneous environments
Crawler

- 759 K collected files, 1.8 TB of disk space
- FTP-index engines and GCSE
Unpacking

- 759 K total files collected
  - filtering
- 172 K filtered interesting files
  - random selection
- 32 K analyzed
  - successful unpacking
- 26 K unpacked (fully or partially)
  - unpacked files
- 1.7 M resulted files after unpacking
Static Analysis

- Correlation/clustering
  - Fuzzy hashes, Private SSL keys, Credentials
- Misconfigurations
  - Web-server configs, Credentials, Code repositories
- Data enrichment
  - Version banners
  - Keywords (e.g., telnet, shell, UART, backdoor)
Correlation

- Correlation via fuzzy-hashes (ssdeep, sdhash)
  - E.g., Vulnerability Propagation
Correlation

- SSL keys correlation + vulnerability propagation
Results

- 38 new vulnerabilities (CVE)
- Correlated them to 140 K online devices
- Affected 693 firmware files by at least one vuln
AVATAR: A Framework for Dynamic Security Analysis of Embedded Systems’ Firmwares

Jonas Zaddach, Luca Bruno, Aurélien Francillon, Davide Balzarotti

NDSS 2014
Embedded devices are everywhere

- Embedded devices are ubiquitous and diverse – but all of them run software
- Even if invisible, they are essential to your life
- Can operate for many years
  - Legacy systems, no (security) updates
- Have a large attack surface
  - Networking, forgotten debug interfaces, etc
Third Party Security Evaluation

- No source code available
- No toolchain available
- No documentation available
- Distinct tools (to flash and debug) for each manufacturer
Wishlist

- Typical PC security toolbox
  - advanced debugging techniques
    - tracing
    - fuzzing
    - tainting
    - symbolic execution
  - Integrated tools
    - IDA pro
    - GDB
Challenges

● Advanced dynamic analysis needs emulation

● Full emulation
  ○ unknown peripherals
  ○ firmware fails if peripherals are missing

● Integration
  ○ support multiple vendors and platforms
Avatar Overview

- Orchestrates execution between emulator and device.
- Forward peripheral accesses to the device under analysis.
Emulator

- S²E
  - LLVM
  - Qemu
  - Klee
  - Analysis plugins
    - Memory
    - Registers
    - CPU state

- Configuration
- Qemu GDB
- Remote Memory

Avatar
Bottlenecks

- Emulated execution is MUCH slower than execution on the real device
  - **memory access** forwarding through low-bandwidth channel

- Interrupts can saturate debug connection
Memory Access Optimization
Memory Access Optimization
Execute Code Snippets

Emulator

State

Code

Avatar

Device

State
Execute Code Snippets
Use Case: Hard Disk

- Recover bootloader protocol with symbolic execution
  - inject GDB stub
  - instrument flash loading
  - inject symbolic values for data read from serial port
  - keep track of which input leads into which code flow
Use Case: Hard Disk

- Search vulnerabilities in SMS decoding routine
  - connect through JTAG
  - execute on device until SMS decoding
  - replace SMS payload
    - with symbolic values
  - check for symbolic values in
    - program counter
    - load/store address
Firmalice Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware

Yan Shoshitaishvili, Ruoyu Wang, Christophe Hauser, Christopher Kruegel, Giovanni Vigna

NDSS 2015
The Rise of Firmware
Emergence of Backdoors


Heffner, Craig. "Reverse Engineering a D-Link Backdoor" /dev/ttys0 (2013).

Vanderbeken, Eloi. "TCP/32764 backdoor, or how linksys saved Christmas!" GitHub (2013).

Backdoor Discovery

Prompt

Authentication

Success

Failure

Backdoor e.g. strcmp()

Hard to find.

Missing!
Our Solution: Input Determinism

- Prompt
- Authentication
  - Backdoor e.g. strcmp()
  - Success
  - Failure
  - Hard to find.
  - Easier to find!
Input Determinism

Prompt

Authentication

Success

Failure

Backdoor e.g. strcmp()
Input Determinism

Required input?

→ Determinable!

Promp

Authentication

Success

Failure
Challenge

Prompt

Authentication

Hard to find.

Backdoor
e.g. strcmp()

Success

Failure

Easier to find, but how?
Firmalice

Inputs:

→ Firmware Sample
→ Security Policy

Challenges:

→ Large binary programs
→ Unrelated user input

Analysis Steps:

→ Static Analysis (backwards program slicing)
→ Dynamic Symbolic Execution
→ Authentication Bypass Check
Static Analysis

Control Flow Graph

Program Dependency Graph

Control Dependency Graph

Data Dependency Graph
Dynamic Symbolic Execution

Prompt

Backdoor

strcmpl()

Authentication

Success

<table>
<thead>
<tr>
<th>ID</th>
<th>Authenticated Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Path 1</td>
</tr>
</tbody>
</table>
Dynamic Symbolic Execution

Prompt

Authentication

Backdoor strcmp()

Success

<table>
<thead>
<tr>
<th>ID</th>
<th>Authenticated Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Path 1</td>
</tr>
<tr>
<td>2</td>
<td>Path 2</td>
</tr>
</tbody>
</table>
Dynamic Symbolic Execution

- Prompt
- Authentication
  - Backdoor strcmp() -> Success
- Path 1
- Path 2
- Path 3

<table>
<thead>
<tr>
<th>ID</th>
<th>Authenticated Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Path 1</td>
</tr>
<tr>
<td>2</td>
<td>Path 2</td>
</tr>
<tr>
<td>3</td>
<td>Path 3</td>
</tr>
</tbody>
</table>
Authentication Bypass

Path 1
input == ???

Path 2
input == ???

Path 3
input == "..."
Backdoor Example: 3S Vision N5072

- Linux embedded device.
- HTTP server for management and video monitoring.
- Security Policy
  - Authentication required for footage access
  - "Image-Type" header
- Backdoor
  - Hard-coded user credentials
  - Username: 3sadmin
  - Password: 27988303
Thank you!

Questions?