FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis

Mingeun Kim¹, Dongkwan Kim², Eunsoo Kim², Suryeon Kim³, Yeongjin Jang⁴, and Yongdae Kim²

¹The affiliated institute of ETRI
²KAIST
³Ministry of National Defense
⁴Oregon State University
IoT Devices are in danger

- 34.2 billion embedded devices will be in use in 2025*
  - Wireless routers, IP cameras, ...

- IoT Devices are an alluring target
  - Satori botnet using 0-days (Dec. 2017)
  - Crypto mining botnet (May. 2018)
  - ECHOBOT, a variant of Mirai (Dec. 2019)
  - New Mirai variant targeting Comtrend routers (July 2020)

- Many IoT devices are exposed to the Internet, especially their web interfaces
  - Shodan, ZoomEye
  - Over 30 exploits used in ECHOBOT target web services
  - Web service RCE (CVE-2020-10173) used for Mirai variants

Analyzing device firmware

❖ Statically analyze device firmware ➔ Many false positives
  – Crack default passwords or find backdoor strings: Costin et al. (SEC ’14), ...
  – Symbolic execution to find vulnerabilities: FIE (SEC’13), Firmalice (NDSS’15), ...

❖ User-level emulation
  – Emulate only the target program, not the entire environment
  – Utilize "chroot" on the firmware filesystem: Costin et al. (AsiaCCS ’16)
  ➔ Cannot reflect system-wide behavior (e.g., device initialization)

❖ System-level emulation
  – Emulating the entire environment, including the kernel
    - Firmadyne (NDSS’16), FirmPin (BLACKHAT US’18), Firm-AFL (SEC’19), ...
  ➔ Many approaches take this and analyze vulnerabilities

❖ Modeling accurate peripherals
  – MMIO, GPIO, DMA: Pretender (RAID’19), HALucinato (SEC’20), P2IM (SEC’20), ...
  ➔ Promising, but immature to support large-scale analysis

Is this sufficient to check vulnerabilities on a large-scale?
QEMU for a virtual environment
- Prebuilt kernel for hooking system calls (with Kprobe)
- Emulating target firmware twice
  - Collect system call logs for network interface setup
- NVRAM* library to wrap related functions

Firmadyne can emulate only 16% of firmware images for web services

*NVRAM (Non-Volatile RAM) stores configuration key-value pairs
Practical large-scale emulation for analyzing IoT devices ➔ Web services, typical attack targets

Randomness of embedded device implementation
➔ Difficulty of catching precise failure causes
➔ No need to be accurate for dynamic analysis
➔ Subtle efforts can address many failure cases
➔ Once implemented, such experience can build up
➔ Successful emulation of 892 firmware images!
Motivating example 1: CVE-2014-3936

❖ Target
  – D-Link DIR-505L

❖ Symptom
  – Fails to configure network connection
    ▪ Missing bridge interface to communicate with the host

❖ Possible causes
  – Access to unsupported peripherals
  – Missing NVRAM configuration value

❖ How to address
  – Run a single command that links the bridge interface

Test
  `brctl addif br0 eth0`
Motivating example 2: CVE-2017-5521

❖ Target
  – NETGEAR R6250

❖ Symptom
  – Fails to boot
    ▪ Diverse initializing program paths
  – Fails to run the web service
    ▪ Missing IOCTL functions

❖ Possible causes
  – Incorrect initializing program path
  – Missing kernel module

❖ How to address
  – Change the initializing program path to “/sbin/preinit”
  – Add IOCTL wrappers

Precompiled Custom Kernel (ARM, MIPS)
Library/Device Driver
Extracted Filesystem + Custom Binaries
Boot & Initialize  Network Setup  Web/CGI Daemons
Firmadyne
NETGEAR R6250
CVE-2017-5521
Test
Our approach

❖ Key observation
  – Emulating high-level behaviors can be sufficient to conduct dynamic analysis
  – Relatively easy and does not need to address the exact causes of emulation failures

❖ Arbitrated emulation
  – Ensures high-level conditions to run target programs by injecting interventions*
  – Focuses on emulating target program to conduct dynamic analysis

❖ Goal
  – Emulating web services in firmware for dynamic analysis (i.e., bug hunting) in a large scale
  – Targeting wireless routers and IP-cameras
    ▪ Popular attack targets and still have many vulnerabilities

❖ High-level conditions to analyze web services
  – A device should be booted without kernel panic
  – Its network should be reachable from the host
  – Its internal web services should be available

*Intervention: an intentionally added action

Check violation cases

➔ Boot environment
➔ Network configuration
➔ Library, device driver, etc.
FirmAE overview

Vendor Servers

Input Firmware

Filesystem

Analysis Container

Emulation Manager

Pre-Emulation

Final Emulation

1. Boot & Initialize
2. Network Setup
3. Library/Device Driver
4. Extracted Filesystem + Custom Binaries
5. Web/CGI Daemons

Precompiled Custom Kernel (ARM, MIPS)

Emulation DB
Crash DB

Checker

Fuzzer

Debug

Confirm

Parallelization

Arbitration
Systemization
Dynamic Analysis
Dataset building

❖ Firmware collection
  – Collect firmware from vendor servers
    ▪ Customized scraper based on Firmadyne’s + Manual download
  – Extract the filesystem
    ▪ Binwalk: Signature-based file search
  – Target architecture: ARMel, MIPSel, MIPSeb

❖ Dataset (1124 images)
  – AnalysisSet (526 images)
    ▪ Old images from 3 vendors to develop arbitrations
  – LatestSet (553 images)
    ▪ Latest images* from 8 vendors to check the effectiveness of arbitrations
  – CamSet (45 images)
    ▪ Latest images* to evaluate arbitrations in another, yet similar domain

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisSet</td>
<td>TP-Link</td>
<td>73</td>
</tr>
<tr>
<td></td>
<td>NETGEAR</td>
<td>274</td>
</tr>
<tr>
<td><strong>Sub Total</strong></td>
<td></td>
<td>526</td>
</tr>
<tr>
<td>LatestSet</td>
<td>D-Link</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>TP-Link</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>NETGEAR</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>TRENDnet</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>ASUS</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>Belkin</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Linksys</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>Zyxel</td>
<td>20</td>
</tr>
<tr>
<td><strong>Sub Total</strong></td>
<td></td>
<td>553</td>
</tr>
<tr>
<td>CamSet</td>
<td>D-Link</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>TP-Link</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>TRENDnet</td>
<td>13</td>
</tr>
<tr>
<td><strong>Sub Total</strong></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>1124</td>
</tr>
</tbody>
</table>

*Latest firmware images are checked as of Dec. 2018
FirmAE - Arbitration

- Analyze emulation failure cases and resolve them with arbitrations

Vendor Servers

Input Firmware

Pre-Emulation

Emulation Manager

Final Emulation

Pre-Emulation:
1. Boot & Initialize
2. Network Setup
3. Library/Device Driver
4. Extracted Filesystem + Custom Binaries
5. Web/CGI Daemons

Final Emulation:
- Precompiled Custom Kernel (ARM, MIPS)

Parallelization

Arbitration
Systemization
Dynamic Analysis
**Arbitration summary**

<table>
<thead>
<tr>
<th>Type</th>
<th>High-level Condition Violation</th>
<th>Intervention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot</td>
<td>Improper booting sequence</td>
<td>Identify the initializing program from the kernel of firmware image</td>
</tr>
<tr>
<td></td>
<td>Missing filesystem structure</td>
<td>Make necessary directories by extracting used paths from binaries</td>
</tr>
<tr>
<td></td>
<td>Improper booting sequence</td>
<td>Identify the initializing program from the kernel of firmware image</td>
</tr>
<tr>
<td>Network</td>
<td>Invalid IP alias handling</td>
<td>Fix routing rule to properly handle IP aliasing</td>
</tr>
<tr>
<td></td>
<td>No network information</td>
<td>Add sequence of commands to set up default network interface</td>
</tr>
<tr>
<td></td>
<td>Insufficient support of multiple network interfaces in QEMU ARM</td>
<td>Set a single network interface on QEMU ARM machine</td>
</tr>
<tr>
<td></td>
<td>Insufficient VLAN setup</td>
<td>Fix VLAN configuration on the host system</td>
</tr>
<tr>
<td></td>
<td>Blocked by rules in iptables</td>
<td>Flush the iptables rules</td>
</tr>
<tr>
<td>NVRAM</td>
<td>Unknown NVRAM default files</td>
<td>1. Search files that contain key names identified from pre-emulation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Initialize NVRAM with found default files</td>
</tr>
<tr>
<td></td>
<td>Crash due to returned NULL pointer</td>
<td>Return an empty string instead of NULL pointer</td>
</tr>
<tr>
<td>Kernel</td>
<td>Insufficient support of kernel module</td>
<td>1. Supplement IOCTL handler in the kernel, it can be different by architecture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. For generalization can be abstracted in LD_PRELOAD library as one function</td>
</tr>
<tr>
<td></td>
<td>Improper kernel version</td>
<td>Upgrade MIPS kernel version to the 4.1, but set ‘CONFIG_COMPAT_BRK’ to prevent old libc crashes</td>
</tr>
<tr>
<td>Others</td>
<td>Unexecuted web servers</td>
<td>Forcibly execute the web servers with appropriate configuration files</td>
</tr>
<tr>
<td></td>
<td>Timeout issues</td>
<td>Increase emulation timeout (Pre: 240s, Final: 360s)</td>
</tr>
<tr>
<td></td>
<td>Lack of tools for emulation</td>
<td>Add full-featured busybox to deal with insufficient command in firmware</td>
</tr>
</tbody>
</table>
Side-effects of arbitration

- Arbitrations may result in different behaviors against the original hardware
  - It has only slight effect on the security analysis of web services
  - We indeed found several vulnerabilities

- Examples
  - Returning empty string from NVRAM
    - As most values from NVRAM are used for configuration, this may direct the program to use the default value
    - Provides more chance to analyze programs than crashing due to NULL dereference
  - Changing network configuration
    - The network configuration can be different from the original environment
    - However, most vulnerabilities are independent to the network configuration (i.e., IP Address)
FirmAE - Systemization

- Vendor Servers
- Analysis Container
  - Input Firmware
  - Pre-Emulation
  - Emulation Manager
- Final Emulation
  - Boot & Initialize
  - Network Setup
  - Library/Device Driver
  - Web/CGI Daemons
  - Extracted Filesystem + Custom Binaries
  - Precompiled Custom Kernel (ARM, MIPS)

- Checker
- Fuzzer
- Debug
- Confirm
- Emulation DB
- Crash DB

Fully-automate and parallelize with containers

Parallelization

Arbitration  Systemization  Dynamic Analysis
Systemization

❖ Full automation
  – Apply interventions
    ▪ Analyze kernel and filesystem information
  – Check network and web server
    ▪ Use "ping" and "curl"
  – Further analyze vulnerabilities

❖ Parallelization with containers
  – Make entire firmware emulation/analysis abstract
  – Build an independent network environment
    ▪ Handle network collision from the hard-coded IP addresses

Host system
GET / HTTP/1.1
Host: 192.168.0.1

IP collision
IP 192.168.0.1
Guest system 1 (emulated firmware)
IP 192.168.0.1
Guest system 2 (emulated firmware)

Host system
Container1
GET / HTTP/1.1
Host: 192.168.0.1
IP 192.168.0.1
Guest system 2 (emulated firmware)

Container2
GET / HTTP/1.1
Host: 192.168.0.1
IP 192.168.0.1
Guest system 2 (emulated firmware)
Emulation results

❖ Emulation check
  – Network reachability
  – Web service availability

❖ Results (vs Firmadyne)
  – AnalysisSet
    ▪ 16.92% → 91.83%
  – LatestSet
    ▪ 16.64% → 69.08%
  – CamSet
    ▪ 4.44% → 60.00%
  – Total
    ▪ 16.28% → 79.36%

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor</th>
<th>Images</th>
<th>Net</th>
<th>Web</th>
<th>Net</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalysisSet</td>
<td>NETGEAR</td>
<td>73</td>
<td>26 (35.62%)</td>
<td>5 (6.85%)</td>
<td>73 (100%)</td>
<td>59 (80.82%)</td>
</tr>
<tr>
<td></td>
<td>TP-Link</td>
<td>274</td>
<td>86 (31.39%)</td>
<td>30 (10.95%)</td>
<td>259 (94.52%)</td>
<td>257 (93.80%)</td>
</tr>
</tbody>
</table>

| Sub Total    |        |        | 167 (31.75%) | 89 (16.92%) | 509 (96.77%) | 483 (91.83%) |

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor</th>
<th>Images</th>
<th>Net</th>
<th>Web</th>
<th>Net</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>LatestSet</td>
<td>TRENDnet</td>
<td>106</td>
<td>35 (33.02%)</td>
<td>23 (21.70%)</td>
<td>91 (85.85%)</td>
<td>63 (59.43%)</td>
</tr>
<tr>
<td></td>
<td>ASUS</td>
<td>107</td>
<td>27 (25.23%)</td>
<td>25 (23.36%)</td>
<td>63 (58.88%)</td>
<td>62 (57.94%)</td>
</tr>
<tr>
<td></td>
<td>Belkin</td>
<td>37</td>
<td>2 (5.41%)</td>
<td>2 (5.41%)</td>
<td>30 (81.08%)</td>
<td>22 (59.46%)</td>
</tr>
<tr>
<td></td>
<td>Linksys</td>
<td>55</td>
<td>13 (23.64%)</td>
<td>8 (14.55%)</td>
<td>48 (87.27%)</td>
<td>44 (80.00%)</td>
</tr>
<tr>
<td></td>
<td>Zyxel</td>
<td>20</td>
<td>3 (0.15%)</td>
<td>0 (0%)</td>
<td>18 (0.90%)</td>
<td>10 (50.00%)</td>
</tr>
</tbody>
</table>

| Sub Total    |        |        | 161 (29.11%) | 92 (16.64%) | 465 (84.09%) | 382 (69.08%) |

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Vendor</th>
<th>Images</th>
<th>Net</th>
<th>Web</th>
<th>Net</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamSet</td>
<td></td>
<td></td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>19 (73.08%)</td>
<td>17 (65.38%)</td>
</tr>
<tr>
<td></td>
<td>TP-Link</td>
<td>6</td>
<td>0 (0%)</td>
<td>0 (0%)</td>
<td>6 (100%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td></td>
<td>TRENDnet</td>
<td>13</td>
<td>2 (15.38%)</td>
<td>2 (15.38%)</td>
<td>10 (76.92%)</td>
<td>10 (76.92%)</td>
</tr>
</tbody>
</table>

| Sub Total    |        |        | 2 (4.44%) | 2 (4.44%) | 35 (77.78%) | 27 (60.00%) |

| Total        |        |        | 330 (29.36%) | 183 (16.28%) | 1009 (89.77%) | 892 (79.36%) |
Effectiveness of each arbitration

❖ How to check?
  – Remove each arbitration from full system
  – Check with web service availability

❖ Results
  – Boot & Network
    ▪ 30% affected
  – NVRAM (the most effective)
    ▪ 35% affected
  – Kernel
    ▪ 4.88% affected
  – Other
    ▪ 22.35% affected

❖ All arbitrations are necessary!
FirmAE - Dynamic Analysis

Dynamically analyze and find vulnerabilities with PoCs and a fuzzer.
Conducting dynamic analysis

- For the emulated web services,
  - Initialize webpages by clicking HTML buttons or calling JavaScript functions with Selenium
  - Collect website information from the filesystem
  - Perform dynamic analysis
    - 1-day analysis: RouterSploit (Known PoCs like Metasploit) + Customized PoC
    - 0-day analysis: Our simple fuzzer targets command injection and buffer overflow

- Customized syscall logs
  - Firmadyne's prebuilt kernel significantly helped analyzing the bugs

- Analyses to show the emulation indeed works!
  - 1-day analysis, vs Firmadyne (with AnalysisSet)
  - 1-day analysis, on latest images (with LatestSet)
  - 0-day analysis, on latest images (with LatestSet)
    - CVE hunting!
1-day analysis results on AnalysisSet (vs Firmadyne)

- Is FirmAE effective to reproduce vulnerabilities?

<table>
<thead>
<tr>
<th>Vulnerability Category</th>
<th>Firmadyne</th>
<th>FirmAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of POC</td>
<td># of Images (Unique)</td>
</tr>
<tr>
<td>Information leak</td>
<td>2</td>
<td>0 (0)</td>
</tr>
<tr>
<td>Command injection</td>
<td>9</td>
<td>10 (6)</td>
</tr>
<tr>
<td>Password disclosure</td>
<td>2</td>
<td>4 (3)</td>
</tr>
<tr>
<td>Authentication bypass</td>
<td>2</td>
<td>0 (0)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>15</strong></td>
<td><strong>14 (9)</strong></td>
</tr>
</tbody>
</table>
### 1-day and 0-day analysis results on LatestSet

- Is FirmAE effective to find new/unpatched vulnerabilities?

<table>
<thead>
<tr>
<th>Type</th>
<th>Vulnerability Category</th>
<th># of Vulns</th>
<th># of Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-day</td>
<td>Information leak in PHP</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Information leak in CGI</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Command injection in UPnP</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Command injection in SOAP CGI</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Command injection in HNAP</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Command injection with backdoor (32764)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Path traversal</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td><strong>Sub Total</strong></td>
<td></td>
<td><strong>11</strong></td>
<td><strong>72</strong></td>
</tr>
</tbody>
</table>

| 0-day      | Command injection in HNAP              | 6          | 13           |
|            | Command injection in CGI               | 1          | 3            |
|            | Buffer overflow in HNAP                | 1          | 1            |
|            | Buffer overflow in CGI                 | 4          | 6            |
| **Sub Total** |                                           | **12**    | **23**       |
| **Total**  |                                           | **23**    | **95**       |
Responsible disclosure

❖ D-Link
  – HNAP (Command injection, Buffer overflow)
    ▪ SetClientInfoDemo – Deprecated page, but can be identified from filesystem
    ▪ All vulnerabilities are patched by the vendor

❖ ASUS
  – BOF: Hall of fame (Dec 2019)
    ▪ Reported on Apr 2019
    ▪ Confirmed on Jan 2020

❖ Belkin
  – Buffer overflow (P1, 40pts from Bugcrowd)
  – Two years passed, no more progress :( 

❖ For more details
  – https://github.com/pr0v3rbs/CVE
Discussion

❖ Improving emulation rates
  – Developing other arbitration techniques
  – Defining more NVRAM default values and IOCTL functions
  – Investigating other devices types such as Network Attached Storage (NAS)
  – Adopting promising peripheral modeling techniques

❖ Applying promising analysis techniques
  – Static + Dynamic analysis
  – Targeting other services
    ▪ UPNP, SOAP-CGI, DHCP, and so on

❖ Developing a honeypot
  – Honware (Vetterl et al., Electronic Crime Research’19)
Conclusion

❖ What we have done
- Proposed arbitrated emulation and investigated failure cases
- Developed its prototype, FirmAE
- Boosted emulation rate from 16.28% (Firmadyne's) to 79.36% (FirmAE) for 1,124 devices
- Found 23 new bugs (11 1-days and 12 0-days) affecting 95 unique latest devices

❖ Lessons learned
- Many failure cases can be easily resolved by arbitrating the high-level behaviors of firmware
- This is sufficient for dynamic analysis
- Emulating diverse embedded devices is challenging, which requires manual efforts

❖ To support community, we release our source code:
- https://github.com/pr0v3rbs/FirmAE
Thank You! Any Questions?

rla5072@nsr.re.kr