SRLFuzzor: An Automatic Fuzzing Framework for Physical SOHO Router Devices to Discover Multi-Type Vulnerabilities

{Yu Zhang, Wei Huo, Kunpeng Jian, Ji Shi, Haoliang Lu, Longquan Liu, Chen Wang, Dandan Sun}¹,², Chao Zhang³, Baoxu Liu¹,²
Institute of Information Engineering, Chinese Academy of Sciences¹
School of Cyber Security, University of Chinese Academy of Sciences²
Institute for Network Science and Cyberspace, Tsinghua University³
Outline

Introduction
- Security of the SOHO Router is Important
- How to Find the Vulnerability
- Challenge of Fuzzing the SOHO Router

Example —— NTP configuration

SRFuzzer —— Our Solution
- Seed Generation
- Seed Mutation
- Exceptional Behavior Triggering and Monitoring
- Power Control

Evaluation
- Experiment Overview
- Analysis of Issues
- Performance of Monitors
- Comparison with Popular Fuzzers

Discussion

Summary

Q&A
Security of the SOHO Router is Important

SOHO routers are in prominent position in nowadays life.

- Smart phone, personal computer, camera, printer, etc

SOHO routers are one of the essential exploiting targets by adversaries.

- VPNFilter infected at least 500,000 devices in at least 54 countries [1]
# Find Vulnerabilities in Routers

Fuzzing is popular in discovering vulnerabilities of IoT devices

<table>
<thead>
<tr>
<th>Technique</th>
<th>Andrei Costin et al.</th>
<th>FIRMADYNE</th>
<th>IoTfuzzer</th>
<th>Muench et al.</th>
<th>FIRM-AFL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulation</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Multi-Type Vulnerability</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Zero-Day Detection</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Coverage-guide</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
Challenge of Fuzzing the SOHO Router

1. How to generate initial seeds?
   - many vendors, few standards
   - various implementations
   - rich information in seeds

2. How to fuzz dedicated systems?
   - obtain code coverage without emulation is difficult
   - while emulation is limited among the various devices
   - “zombie” state

3. How to trigger and monitor multi-type vulnerabilities as many as possible?
   - silent memory corruptions
   - not only memory corruptions
   - decrease the false positive and missing
Example——NTP configuration

• CONF-READ communication model
  ➢ GET request is a READ operation
  ➢ POST request is a CONF operation

• KEY-VALUE data model
  ➢ ntpserver1=time.test1.com

• Several different phases to trigger multi-type vulnerabilities

User views the NTP configuration "ntpserver1" with value "cn.pool.ntp.org", etc.
User modifies the value of "ntpserver1" to "time.test1.com" and submit the configuration.
User views the new configuration after handling of backend.

Web Server
  Applications
  Networking Service
  Platform Specific Services
  Dedicated Operating System
  Typical Architecture of SOHO router

GET request
POST request

NVRAM
Database File
Config File
Example——NTP configuration

- 2 functions to handle the variable ntpserver1
  - A command injection vulnerability in conf_ntpserver1() function
    - Data type inconsistency
  - A stack-based overflow vulnerability in read_ntpserver1() function
    - Length limitation inconsistency in 2 related functions
  - The memory corruption can cause crash, what about the command injection, XSS and info disclosure?

Raw Request

POST /apply.cgi/?NTP_debug.htm HTTP/1.1
Host: 192.168.66.1
Connection: keep-alive
Content-Length: 209
submit_flag=ntp_debug&conflict_wanlan=n&ntpserver1=ntpserver2=time.test1.com&ntpserver2=time.test2.com&ntpadjust=0&hidden_ntpserver=GMT8&hidden_dstflag=0&hidden_select=33&dif_timezone=0&time_zone=GMT-8&ntp_type=0&pri_ntp=

Example — NTP configuration

1. `int conf_ntpserver1(char * input){
   char buf[0x100];
   char * ntp = read_from_request("ntpserver1", input);
   if(strlen(ntp) > 0x80)
      return -1;
   //use variable "ntp" to build config command.
   sprintf(buf, "/usr/bin/config ntpserver=%s.", ntp);
   //command injection occurs.
   system(buf);
   return 0;
}

2. `int read_ntpserver1(){
   //the length of info is no more than 0x80.
   char info[0x50];
   char * ntp = get_config("ntpserver1");
   //stack-based overflow occurs.
   sprintf(info, "ntpserver=%s", ntp);
   return 0;
}
SRFuzzer

- Fuzz the physical devices directly and automatically
- Trigger multi-type vulnerabilities with KEY-VALUE data model and CONF-READ communication model
- Generate information and monitor it when triggering exceptional behaviors
- Use smart plug to restore the device from “zombie” state
- Modular design and well extendibility
Seed Generation

- **Crawler**
  - General crawler
  - Passive crawler
- **KEY-VALUE parser with labeling**
  - Variable string
  - Fixed string
  - Number

**Raw Request**

```
POST /apply.cgi?/NTP_debug.htm HTTP/1.1
Host: 192.168.66.1
Connection: keep-alive
Content-Length: 209

submit_flag=ntp_debug&conflict_wanlan=&ntpserver1=time.test1.com
&ntpserver2=time.test2.com&ntpadjust=0&hidden_ntpserver=GMT8&
hidden_dstflag=0&hidden_select=33&dif_timezone=0&time_zone=GMT-
8&ntp_type=0&pri_ntp=
```

**Seed**

```
<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>submit_flag</td>
<td>ntp debug</td>
<td>fixed str, variable str</td>
</tr>
<tr>
<td>conflict_wanlan</td>
<td>variable str</td>
<td></td>
</tr>
<tr>
<td>ntpserver1</td>
<td>time.test1.com</td>
<td>variable str</td>
</tr>
<tr>
<td>hidden_dstflag</td>
<td>0</td>
<td>number, variable str</td>
</tr>
<tr>
<td>hidden_select</td>
<td>33</td>
<td>number, variable str</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

**URL:** http://FUZZING_IP/apply.cgi?/NTP_debug.htm
**METHOD:** POST
**Tuple SET:**
# Seed Mutation

<table>
<thead>
<tr>
<th>Key</th>
<th>Original Value</th>
<th>Attribute</th>
<th>Mutation Rule</th>
<th>Example of Mutated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntpserver1</td>
<td>time.test1.com</td>
<td>variable string</td>
<td>Overflow</td>
<td><code>time.test1.comtime.test1.com...</code> (repeat 20 times)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NULL-Pointer dereference</td>
<td>(empty value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Command Injection</td>
<td><code>time.test1.com&quot;;wget http://192.168.1.2/ntpserver1;</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Stored XSS</td>
<td><code>time.test1.com&quot;;&lt;script&gt;alert('xss_ntpserver1')&lt;/script&gt;</code></td>
</tr>
<tr>
<td>submit_flag</td>
<td>ntp_debug</td>
<td>fixed string</td>
<td>fixed string</td>
<td>ntp_debug</td>
</tr>
<tr>
<td>hidden_dstflag</td>
<td>0</td>
<td>number</td>
<td>number</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>variable string</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Exceptional Behavior Triggering and Monitoring

- A CONF operation for the first step
- A READ operation after a CONF operation
- Three typical monitoring mechanisms
  - Response-based monitor
  - Proxy-based monitor
  - Signal-based monitor

Here should be an explanation on how to determine if a vulnerability is triggered.
Power Control

- Use an extra hotpot to connect the Smart Plug and Fuzzing Node
- Use Mi Smart Plug and python-miio package in practice
## Experiment Overview

- We selected 10 devices from 5 different popular vendors
- We obtained 101 unique issues, 97 of which were assigned vulnerability IDs
- We manually crafted the PoCs for all unique issues

<table>
<thead>
<tr>
<th>ID</th>
<th>Vendor</th>
<th>Product</th>
<th>Firmware Version</th>
<th>Architecture</th>
<th>Signal-based Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NETGEAR</td>
<td>Orbi</td>
<td>V15.03.05.19 (6318)_CN</td>
<td>ARM32 (LE)</td>
<td>Device Feature, Serial Port</td>
</tr>
<tr>
<td>2</td>
<td>NETGEAR</td>
<td>Insight Managed Smart Cloud Wireless Access Point</td>
<td>WAC505-510_firmware_V5.0.5.4</td>
<td>ARM32 (LE)</td>
<td>Not Support</td>
</tr>
<tr>
<td>3</td>
<td>NETGEAR</td>
<td>WNDR-4500v3</td>
<td>WNDR4500v3-V1.0.0.50</td>
<td>MIPS32 (BE)</td>
<td>Device Feature, Serial Port</td>
</tr>
<tr>
<td>4</td>
<td>NETGEAR</td>
<td>R8500</td>
<td>R8500-v1.0.2.100, R8500-V1.0.2.116</td>
<td>ARM32 (LE)</td>
<td>Device Feature, Serial Port</td>
</tr>
<tr>
<td>5</td>
<td>NETGEAR</td>
<td>R7800</td>
<td>R7800-V1.0.2.44, R7800-V1.0.2.46</td>
<td>ARM32 (LE)</td>
<td>Device Feature, Serial Port</td>
</tr>
<tr>
<td>6</td>
<td>TP-Link</td>
<td>TL-WVR900G</td>
<td>V3.0_170306</td>
<td>MIPS32 (BE)</td>
<td>Not Support</td>
</tr>
<tr>
<td>7</td>
<td>Mercury</td>
<td>Mer450</td>
<td>MER1200GV1.0</td>
<td>MIPS32 (BE)</td>
<td>Not Support</td>
</tr>
<tr>
<td>8</td>
<td>Tenda</td>
<td>G3</td>
<td>V15.11.0.5</td>
<td>ARM32 (LE)</td>
<td>Existed Vulnerability</td>
</tr>
<tr>
<td>9</td>
<td>Tenda</td>
<td>AC9</td>
<td>V15.03.05.19</td>
<td>ARM32 (LE)</td>
<td>Existed Vulnerability</td>
</tr>
<tr>
<td>10</td>
<td>Asus</td>
<td>RT-AC1200</td>
<td>RT-AC1200-3.0.0.4.380.9880</td>
<td>MIPS32 (LE)</td>
<td>Device Feature</td>
</tr>
</tbody>
</table>
Analysis of Issues

- 101 confirmed issues
  - 48 memory corruption
  - 39 command injection
  - 9 stored XSS
  - 5 info disclosure
- 67.33% are triggered in CONF
- 32.67% are triggered in READ
- Device specificity
  - TP-Link TL- WVR900G

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>CONF</th>
<th>READ</th>
<th>SUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEM</td>
<td>CMD</td>
<td>MEM</td>
</tr>
<tr>
<td>Orbi</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Insight</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WNDR- 4500v3</td>
<td>6</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>R8500</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R7800</td>
<td>0</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>TL- WVR900G</td>
<td>0</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>Mer450</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>G3</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AC9</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RT-AC1200</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUM</td>
<td>31</td>
<td>37</td>
<td>17</td>
</tr>
</tbody>
</table>
Analysis of Issues

- 97 assigned IDs (43 CVEs + 52 PSVs + 2 CNVDs)

Severity Statistic Based on CVSS Score

- Critical: 25.77%
- High: 58.76%
- Medium: 15.46%
- Low: More than 1/4

Impact Statistic

- EoP: 84.54%
- Scripts Execution In Browser: 9.27%
- DoS: 5.15%
- Information Disclosure: 1.03%

web management -> root shell
### Performance of Monitors

- 77.23% confirmed issues are caught by the general mechanism (response-based and proxy-based monitor).
- Signal-based monitor can catch the silent memory corruption.
- Device rebooted 6.8 times on average:
  - Handle requests in one process
  - Handle requests in subprocesses
  - Backend is developed on top of OpenWRT

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>MEM</th>
<th>CMD</th>
<th>XSS</th>
<th>INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orbi</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Insight</td>
<td>0</td>
<td>N/A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WNDR-4500v3</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>R8500</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>R7800</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>TL-WVR900G</td>
<td>0</td>
<td>N/A</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>Mer450</td>
<td>0</td>
<td>N/A</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>G3</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AC9</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RT-AC1200</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>SUM</strong></td>
<td>25</td>
<td>23</td>
<td>39</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Reboot Times**

![Reboot Times Chart](chart.png)
Comparison with Popular Fuzzers

• Selected 7 devices randomly
• More memory corruption issues than customized boofuzz by 53.57%
• More command injection issues than Commix by 25.81%
• Similar performance with wfuzz on XSS detection

“time-related” injection monitoring technique missed some issues

one mutated input cannot bypass the backend validity check

Memory Corruption

Missing the issues occurring in READ op or subprocesses

Command Injection

Stored XSS
Discussion

• Limitation of the scope
  ➢ More types of device
  ➢ More types of protocol

• Vulnerability of severity
  ➢ More critical issues

• Research on data inconsistency
  ➢ To find more issues and help vendors to harden their products.

• Monitoring
  ➢ Make the efficient monitoring mechanism more general
Summary

- We present SRFuzzer for physical SOHO routers to automatically discover multi-type vulnerabilities
- We reveal the root cause of the different types of vulnerability as data inconsistency
- We obtain 97 assigned vulnerability IDs by fuzzing 10 popular real-world devices
THANKS