Are you watching TV now? Is it real?:
Hacking of smart TV with 0-day

Hack in paris 2017

Security Analysis aNd Evaluation(SANE) Lab.

Jongho Lee, Mingeun Kim*, Seungjoo Kim**

hellsonic@korea.ac.kr, pr0v3rbs@kaist.ac.kr, skim71@korea.ac.kr
CIST (Center for Information Security Technologies), Korea University
*KAIST
**Corresponding Author
Contents

1. Project introduction
2. webOS structure and analysis
3. webOS smart TV Vulnerability analysis
4. webOS smart TV Hacking Scenario & Demonstration
1. Project introduction

- Project topics
1. Project introduction

Project topics

WikiLeaks says CIA hacked Samsung smart TVs

30 Comments / Share / Tweet / Stumble / Email

Last Updated Mar 8, 2017 11:12 AM EST

WikiLeaks says Samsung smart TVs were hacked to enable spying on consumers.

In a trove of documents released Tuesday, WikiLeaks included code that it says shows the CIA worked with U.K. intelligence officials to turn microphones in TVs into listening devices.

Samsung smart TVs have microphones so viewers can make voice commands, such as requests for movie recommendations. The commands typically aren’t transmitted outside the home unless users activate the feature. If the TV is off, there’s no listening being done.

But WikiLeaks claims that documents it obtained show that through a program called Weeping Angel, the target TV appears to be off when it is actually on -- and listening.

WikiLeaks says the audio goes to a covert CIA server rather than a party authorized by Samsung. In such cases, audio isn’t limited to TV commands but could include everyday
“Making hacking scenarios and demonstrate it, for smart TV through analysis of webOS vulnerability.”
2. webOS structure and analysis

- webOS History
- Web SW Platform VS Native SW Platform
- webOS features
- webOS Smart TV Emulator analysis
- webOS Smart TV structure
2. webOS structure and analysis

webOS History

- **2008**
  - Selected as Palm's next generation OS

- **2009**
  - Presentation of Palm Pre based on webOS (CES 2009)
  - Released Palm Pre
  - Competitor launch (Motorola DROID, iPhone 3GS)
  - Launched "low-cost smartphone" Palm Pixi

- **2010**
  - HP take over Palm
  - Attempt to mount webOS on HP products

- **2011**
  - Released Touch Pad
  - HP WebOS external sales announcement

- **2012**
  - Open webOS project announcement
  - Rubinstein resigned two days before webOS Project announcement

- **2013**
  - It is being developed as an operating system for mobile and smart TV and Internet of things
2. webOS structure and analysis

Web SW Platform VS Native SW Platform

Native SW Platform

- Apple iOS
- Android
- Ubuntu OS
- C, C++, Java

- Relatively fast
- Device-specific functions to increase efficiency
- Application and development dependency affected by SW and HW platforms
- Difficulty of support multi-platform

Web SW Platform

- Firefox OS
- Chrome OS
- webOS
- HTML, JavaScript

- Relatively slow
- Application can be developed based on web standards
- Web engine alleviates platform dependencies
- High upgrade flexibility
- Convenience of code portability
2. webOS structure and analysis

webOS features

Application is runs in browser that uses HTML, JavaScript, HTTP, etc.

webOS Features

Development environment based on Enyo Web App Framework

Increase of system scalability (Embedded-optimized Luna Bus system)

Maximize the use of JavaScript (Node.js service framework)

Convenient Development

Dependency Mitigation

Code Portability
2. webOS structure and analysis

webOS features

Luna Bus

- Used to communicate apps and services.
- Built-in apps that use services that restrict access due to security problems use Private Bus.
- The 3rd party application registers with the public bus to prevent access to the internal API that use the private bus.
2. webOS structure and analysis

webOS features

webOS structure

- /usr/sbin/sam
  - Processes that run non-web-based applications

- /usr/bin/WebAppMgr
  - Chrome-based QTWebEngine program that runs webOS app

- /usr/sbin/appinstallld
  - Service process that creates and manages files when installing Application

- /usr/sbin/activitymanager
  - A service process that manages all activities (apps, services, networks, etc.) running on the device

- /usr/palm/nodejs/unified_service_server.js
  - Nodejs local server to run the webOS app
2. webOS structure and analysis

webOS smart TV Emulator analysis

Emulator analysis

Introduction to webOS TV Emulator

Getting Familiar with Emulator UI

This is the screenshot of webOS TV emulator launched on the computer.
## Why Analyze Emulator?

- **Has a similar environment with real TV**
  - Was provided for application testing

- **Rooting easily (maybe?)**
  - We can attack, even virtual disk!
  - We can predict attack vectors for real TV

<table>
<thead>
<tr>
<th>sam</th>
<th>WebAppMgr</th>
<th>appinstalld</th>
<th>activitymanager</th>
<th>Node.js service</th>
</tr>
</thead>
</table>

2. webOS structure and analysis

webOS smart TV Emulator analysis

Partitions of webOS Emulator

- The mount order is partition 1 through 4.
  - Partition 2 has a script that decrypts Partition 3.
  - On encrypted Partition 3, there will be a script to execute the important daemon.
2. webOS structure and analysis

webOS smart TV Emulator analysis

Partition 2 configuration file analysis

- Analysis of service configuration file executed at emulator boot time

```
Partition 3 decryption program

/home/root/openV

Partition 3 decryption program found.
```
2. webOS structure and analysis

webOS smart TV Emulator analysis

Partitions of webOS Emulator

- Partition 3 decrypted with the decryption key found by strace

root@kali:~# echo "814c501c60289307aaedf3c83286983425f5654d67875f0f4454433e35d48f" > /mnt/sda2/home/root/key
root@kali:~# cryptsetup luksOpen /dev/sda3 enc < /mnt/sda2/home/root/key
root@kali:~# mkdir -p /mnt/sda3
root@kali:~# mount -o nosuid /dev/mapper/enc /mnt/sda3
2. webOS structure and analysis

**webOS smart TV Emulator analysis**

Modify the dropbear option inside Partition 3

Modify the dropbear option in 12.sh

```bash
exec /usr/sbin/dropbear -w -g -B -F -d /var/lib/dropbear/dropbear_dss_host_key -r /var/lib/dropbear/dropbear_rsa_host_key
```

```bash
exec /usr/sbin/dropbear -B -F -d /var/lib/dropbear/dropbear_dss_host_key -r /var/lib/dropbear/dropbear_rsa_host_key
```

```bash
/home/root/init.enc/12.sh
```
2. webOS structure and analysis

webOS smart TV Emulator analysis

Emulator root shell

root shell
2. webOS structure and analysis

**webOS smart TV structure**

**Application Sandbox**

webOS 3.5 is the first Smart TV platform to attain Cybersecurity Assurance Program Certification.

*World’s First UL Cybersecurity Certified webOS Smart TV*

‘Cybersecurity Assurance Program UL 2900-1 Certificate’

Each application was sandboxed.

```
/web
  - /bin
  - /boot
  - /var/palm/jail
  - /dev
  - /etc
  - /home
  - /lib
  - /opt
  - /proc
  - /run
  - /sbin
  - /sys
  - /tmp
  - /usr
  - /var
  - ...
```

```
/var/palm/jail/{app}
  - /bin
  - /dev
  - /etc
  - /home
  - /lib
  - /media
  - /mnt
  - /opt
  - /proc
  - /sys
  - /tmp
  - /usr
  - /var
```
2. webOS structure and analysis

webOS smart TV structure

Other difference

- Remote Control
  - Different binary
  - Different work process

- ssh daemon
  - Openssh

- FrameBuffer
  - Can’t access with root privilege

- Architecture
  - ARM
3. webOS smart TV
Vulnerability analysis

- Attack vector
- Vulnerability
3. webOS smart TV Vulnerability analysis

**Attack vector**

**Key Process**

- **appinstallld**: Daemon receiving application install command
- **sam**: Helps to run non-web-based applications.
- **ls-hubd**: Daemons for communication between applications
- **bsa_server**: Bluetooth server for communication with remote control
- **upnpd**: HTTP protocol as a daemon for plug & play communication
  - Pass the command through
- **ss.apiadapter**: Daemon that sends and receives commands when remote control of mobile phone is connected
3. webOS smart TV Vulnerability analysis

**Attack vector**

**Analysis of input vector as hacking path**

- **ls-hubd**
  - Commands in Luna protocol for application communication
  - Vulnerability due to xml tampering. (BOF, Command Injection)

- **ss.apiadapter**
  - Modify command received from remote control
  - Replay attack, hidden service?

- **upnpd**
  - Ability to modulate XML, HTTP header information when connecting to other devices with smartphone
  - Possible memory vulnerability (BOF)

- **WebAppMgr / Nodejs server**
  - Page modifiable by user creating application.
  - Running applications on the Node.js server.
    - Vulnerability in Chrome browser possible
    - Node.js server vulnerability attack target

- **bsa_server**
  - The remote control modulates the transmitted Bluetooth signal and attacks the running server
    - Possible replay attack vulnerability
    - Possible BOF memory corruption vulnerability
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **Process luna protocol data centrally**
  - Transfers commands to a system or other apps

- **Maybe has a vulnerabilities about...**
  - Command injection

![Image](image_url)

- **ls-hubd**

  Commands in Luna protocol for application communication
  → Vulnerability due to xml tampering. (BOF, Command Injection)
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **Communicate with mobile remote control**
  - Receives general TV functions (WSS)
  - Receives touchpad inputs (WS)

- **Maybe has a vulnerabilities about...**
  - Command injection
  - Replay attack by capturing the packet
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **upnpd**
  
  Ability to modulate XML, HTTP header information when connecting to other devices with smartphone
  
  → BOF, CSRF, command injection

- **Broadcasts to announce the presence of TV**
  - Sends and receives upnp packets

- **Maybe has a vulnerabilities about...**
  - CSRF
  - Command Injection
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **WebAppMgr / Nodejs server**
  - Page modifiable by user creating application.
  - Running applications on the Node.js server.
  - Vulnerability in Chrome browser possible
  - Node.js server vulnerability attack target

- **Runs the webOS application**
  - Uses QtWebEngine based on chrome

- **Maybe has a vulnerabilities about...**
  - Chrome
  - Nodejs
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **bsa_server**
  - The remote control modulates the transmitted Bluetooth signal and attacks the running server
    - Possible replay attack vulnerability
    - Possible BOF memory corruption vulnerability

- **Communicates with remote control using BT**
  - Used for air-mouse and sending voice data

- **Maybe has a vulnerabilities about...**
  - Replay attack by using captured BT data
3. webOS smart TV Vulnerability analysis

**Attack vector**

Analysis of input vector as hacking path

- **To find attack vectors...**
  - More expensive -> more functions
  - More functions -> more attack vectors

- **Other modules...**
  - Voice inputs from Bluetooth remote control
  - Camera? (maybe webOS doesn’t support)
Vulnerability with Incorrect Permission

Access to physical memory with prisoner privilege
3. webOS smart TV Vulnerability analysis

Vulnerability

Vulnerability with Incorrect Permission
3. webOS smart TV Vulnerability analysis

Vulnerability

Vulnerability with Incorrect Permission

/bin/ping.iputils
Usage: ping [-aAbBdDfhlNnOqrRUvV] [-c count] [-i interval] [-l interface]
[-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
[-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
[-W deadline] [-W timeout] [hop1 ...] destination
hexdump -C /dev/mem | grep "2d e9 f0 4f ad f2 64 4d"
hexdump: 224621c0 2d e9 f0 4f ad f2 64 4d 82 4d 06 46 0f 46 01 f0 |-..O..dM.M.F.F..|
25eda20 b2 8f 00 00 96 92 00 00 2d e9 f0 4f ad f2 64 4d |....V...-.O..dM|
/dev/mem: Bad address

Checking physical memory after binary execution
3. webOS smart TV Vulnerability analysis

Vulnerability

Vulnerability with Incorrect Permission

Run the program → Program is loaded into memory → Memory modulation → Running a Modulated Program
3. webOS smart TV Vulnerability analysis

Vulnerability

Vulnerability with Incorrect Permission

```
/media/developer $ /bin/ping.iputils
Usage: ping [-aAbBdDfhLn0qrRUVvV] [-c count] [-i interval] [-l interface]
             [-m mark] [-M pmtdisc_option] [-l preload] [-p pattern] [-Q tos]
             [-s packetsize] [-S smdbuf] [-t ttl] [-T timestamp_option]
             [-w deadline] [-W timeout] [hop1 ...] destination

/media/developer $ ./devmem_root
[*] ping main addr = 0x232ed1c0
2d e9 f0 4f ad f2 64 4d 82 4d 6 46 f 46 1 f0

[*] overwritten with shellcode
```

Physical Memory Modulation ➔
Obtain root authority
3. webOS smart TV Vulnerability analysis

Vulnerability

DirtyCOW Vulnerability

Vulnerability to write different contents to read-only area using Race Condition while executing Copy-On-Write function of Linux

Vulnerability

DirtyCOW Vulnerability

CVE-2016-5195 code fix

Check your TV's kernel code

Check for possible DirtyCOW vulnerability
3. webOS smart TV Vulnerability analysis

Vulnerability

DirtyCOW Vulnerability

1. Find the binary that has `setuid` attribute

2. Make the binary that executes the shell
3. webOS smart TV Vulnerability analysis

Vulnerability

DirtyCOW Vulnerability

① Find the binary that has setuid attribute

② Make a binary that executes the shell

Overwrite setuid-binary with the shell-binary to Get root privileged shell!
4. webOS smart TV
Hacking Scenario & Demonstration

- Hacking Scenario
- Hacking Demonstration
4. webOS smart TV Hacking Scenario & Demonstration

Hacking Scenario

- Force installation of malicious app
- Pirate broadcast
- Invasion of privacy
- Remote control

Hacker
Hacking Scenario

Pirate broadcasting

1. Smart TV remote control
2. Force installing the Developer Mode.
3. Obtain root authority
4. Pirate broadcasting and screen modulation
4. webOS smart TV Hacking Scenario & Demonstration

Hacking Scenario

Exporting information

1. Smart TV remote control
2. Force installing the Developer Mode.
3. Obtain root authority
4. Network packet capture, log collection
5. Collecting and exporting information

Enterprise

Person

Hacker
4. webOS smart TV Hacking Scenario & Demonstration

Hacking Scenario

Screen Modulation & Pirate Broadcasting

```
pr0v3rbs@pr0v3rbs-ralz: /mnt/c/Users/pr0v3rbs

/home/root # id
uid=0(root) gid=0(root) groups=0(root),10(wheel),506(pulse-access),509(se),777(crashd)
/home/root # ls -al /dev/fb0
crw------ 1 root root 29, 0 Dec 31 1969 /dev/fb0
/home/root # cat /dev/fb0 2>&1
cat: read error: No such device
/home/root # echo 11 > /dev/fb0 2> result
/home/root # cat result
bash: write error: No such device
/home/root#
```
4. webOS smart TV Hacking Scenario & Demonstration

Hacking Scenario

Screen Modulation & Pirate Broadcasting
Hacking Scenario

Screen Modulation & Pirate Broadcasting

Check source code of FB device from webOS kernel source
→ FB input/output routine found

Create a kernel driver and successfully modifies the screen
Hacking Demonstration
Acknowledgement

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2015-0-00403) supervised by the IITP (Institute for Information & communications Technology Promotion)
JongHo Lee (hellsonic)

Age: 27

2016
• Belluminar WCTF 2016 Beijing / 1st Place
• HITCON CTF 2016 Taipei Final / 1st Place
• SECCON CTF 2015 Tokyo Final / 1st Place

2015
• SECCON CTF 2014 Tokyo Final / 1st Place
• DEFCON CTF 2015 Las Vegas / 1st Place
• HITCON CTF 2015 Taipei Final / 1st Place

JongHo Lee is a Master's Degree student at the Graduate School of Information Security, Korea University. He is in charge of information security at RaonSecure. He has won in many CTFs (including DEFCON, HITCON, SECCON, etc...) and has many real-world hacking experiences. He is also a mentor and adviser in various national information security areas.
MinGeun Kim is a Master's Degree student at the Graduate School of Information Security, KAIST. He worked as a windows kernel driver developer for 3 years at a company called ESTsoft, which developed the ALYac antivirus. He has various CTF experiences and some real-world hacking experience. He research various IoT vulnerabilities at KAIST.
## Author

**Han-byeon Ji (onestar)**
- Security Consultant, RAONSECURE
- ghb202@gmail.com

**Ui-seong Park (zairo)**
- Associate Research Engineer, RAONSECURE
- park.uiseong@gmail.com

**Gyeong-sik Song (secretpack)**
- KITRI BoB
- secretpack@gmail.com

**Jin-woo Lee (unknown84)**
- Kookmin univ.
- rkske9254@gmail.com

**Sang-jun Song (s0ngsari)**
- Associate Research Engineer, THEORI
- s0ngsari@theori.io
Prof. Seungjoo Gabriel Kim received his B.S, M.S and Ph.D. from Sungkyunkwan University (SKKU) of Korea, in 1994, 1996, and 1999, respectively. Prior to joining the faculty at Korea University (KU) in 2011, he served as Assistant & Associate Professor at SKKU for 7 years. Before that, he served as Director of the Cryptographic Technology Team and the (CC-based) IT Security Evaluation Team of the Korea Internet & Security Agency (KISA) for 5 years. He is currently a Professor in the Graduate School of Information Security Technologies (CIST). Also, He is a Founder and Advisory director of hacker group, HARU and an international security & hacking conference, SECUINSIDE. Prof. Seungjoo Gabriel Kim’s research interests are mainly on cryptography, Cyber Physical Security, IoT Security, and HCI Security. He is a corresponding author.
Thank you

hellsonic@korea.ac.kr
pr0v3rbs@kaist.ac.kr