Hacking Cell Phone Embedded Systems

Keegan Ryan – RECON 2017
The Target
The Target
The Target

Meriac (2010), Churchill
Legacy ICLASS

- Introduced in 2007
- Broken in 2010
  - Master key on every reader
  - Security of card reader broken
  - Protocol reverse engineered
- New version of iCLASS released, but many still use Legacy iCLASS
- Uses ISO15693

Nexus S

- Introduced in 2010
- One of earliest to support NFC, including ISO15693
- Android source code available
- Cheap
Nexus S

- Try Android app first
- Transceive raw bytes
- CRC added automatically, but we don’t want a CRC
- Not added by libraries
- Not added by kernel
- Must be added by NFC controller chip
PN544

- Separate from Nexus S CPU
- Powered by host or external field
- Supports ISO 15693, Mifare, FeliCa
- Supports firmware upgrades
- Uses 80C51MX Processor
Investigating the PN544
Firmware Recovery

- **PHDNLD_CMD_READ**
- Pull from update file
- Code signing
  - Protected with SHA1 and RSA-1024
  - Introduced after first devices shipped

```
tree 08510355fb6f70462288c28e03fafc99ae9ee7e9
parent df82c4dd7c6d5ad232b5628edf73aa9ea3f8c2c0 [diff]
```

Patch to add Secure Download Mechanism in the libnfc

This patch permit to support secure download update and also to avoid locking states in case of download failures

Change-Id: 198aa80976a67b18562ddcff4d085ed415dac4933

- Need a device never updated past Gingerbread
Reverse Engineering

• Look for strings. There aren’t any.
• Look for CRC constants. They don’t exist.
• Look for usage of the XOR instruction. No help.
• Just start reversing until we find something useful.
Reverse Engineering

- Reverse commonly called functions
- Find switch function
- Find command switching
- Trace known command IDs through code
Reverse Engineering

```c
/* ETSI HCI Specific RF Reader Gates */
phHciNfc_RFReaderAGate = 0x13,
phHciNfc_RFReaderBGate = 0x11,

/* Proprietary Reader Gate */
phHciNfc_ISO15693Gate = 0x12,
phHciNfc_RFReaderFGate = 0x14,
phHciNfc_JewelReaderGate = 0x15,

/* ETSI HCI Card RF Gates */
phHciNfc_CETypeBGate = 0x21,
phHciNfc_CETypeBPrimeGate = 0x22,
phHciNfc_CETypeAGate = 0x23,
phHciNfc_CETypeFGate = 0x24,

/* NFC-IP1 Gates */
phHciNfc_NFCIP1InitRFGate = 0x30,
phHciNfc_NFCIP1TargetRFGate = 0x31,
```

; End of function switch_by_gate

Libnfc-nxp
Problem:
Problem: Missing Code

- PATCH_TABLE
- FW_CODE
- ???
- PATCH_CODE
- EEPROM/CFG
Problem: Missing Code

- PATCH_TABLE
- FW_CODE
- KERNEL_CODE
- EEPROM/CFG
- PATCH_CODE
Kernel Recovery

- We understand and can modify \texttt{FW\_CODE}
- \texttt{FW\_CODE} doesn’t have access to kernel
- We can modify \texttt{PATCH\_CODE}
- Don’t know how to trigger \texttt{PATCH\_CODE}
- Want to maximize chances of executing our code
Kernel Recovery

PATCH_CODE
Kernel Recovery

PATCH_CODE
Kernel Recovery
Problem: Missing Code

- PATCH_TABLE
- FW_CODE
- KERNEL_CODE
- EEPROM/CFG
- PATCH_CODE
Problem: Missing Code

- PATCH_TABLE
- EEPROM/CFG
- FW_CODE
- KERNEL_CODE
- PATCH_CODE
Reverse Engineering Kernel

- Look for strings. Still aren’t any.
- Look for CRC constants. Still don’t exist.
- Look for usage of the XOR instruction. No help.
- CRC creation is done by hardware
- Still not impossible, but we need a new approach
Wireless Protocols
SDR Setup

- Signal Source
- Antenna
- Upconverter
- Radio
SDR Setup

Scope Plot

Counts

Time (ms)

Ch1  Ch2  Ch3

Persistence
Analog Alpha: 0.0994

Axes Options
Secs/Div:
Counts/Div:
Y Offset:
T Offset:

Channel Options
Coupling: AC
Marker: None

<\s> 10 01 10 00 01 00 00 00...
Transfer Speed

- ISO15693 has two modes:
  - Slow (1.65 kbps)
  - Fast (26.48 kbps)
- Nexus S uses slow mode
- ICLASS only uses fast mode
Problem: Transfer Speed

- Capability probably exists, but is unused.
- Find transmission code
- Loads settings from EEPROM/CFG
- Only uses one set of values
- Swap around values in EEPROM/CFG
- Fast mode!
case NXP_MIFARE_RAW:
{
    if (p_pipe_info->param_length < RDR_A_MIFARE_RAW_LENGTH)
    {
        status = PHNFCSTVAL(CID_NFC_HCI,
                            NFCSTATUS_INVALID_PARAMETER);
    }
    else
    {
        /*
         * Buffer shall be updated with
         * TO - Time out (1 byte)
         * Status - b0 to b2 indicate valid bits (1 byte)
         * Data (with CRC) - params received from this function
         */
    }
}

case NXP_MIFARE_CMD:
{
    /*
     * Buffer shall be updated with
     * Cmd - Authentication A/B, read/write (1 byte)
     * Addr - Address associated with Mifare cmd (1 byte)
     * Data - params received from this function
     */
Problem: Checksum Generation

- FW_CODE Command Handler
  - ISO15693 Setup (CRC)
  - MIFARE Setup (CRC)
  - MIFARE Setup (No CRC)
- RF Transmit

Find differences here

Apply difference here
Patching the Kernel

- PATCH_TABLE
- FW_CODE
- KERNEL_CODE
- PATCH_CODE
- EEPROM/CFG
Exploitation
Patching Checksum Generation

- PATCH_TABLE
- FW_CODE
- KERNEL_CODE
- EEPROM/CFG
- PATCH_CODE
Putting It All Together

- PATCH_TABLE
- FW_CODE
- EEPROM/CFG
- KERNEL_CODE
- PATCH_CODE
Demo
Demo

keegan@keegan-VirtualBox:~/magic_demo$ ./run_demo.sh
Future Research

What can be done with a hacked NFC controller?

- Surreptitiously read a badge
- Information storage
- Information exfiltration
Future Research

- What other embedded systems do we carry everywhere?
  - Bluetooth
  - USB controller
  - Baseband radio
  - Camera
  - Fingerprint reader
- What could you make these systems do?
Bypassing Firmware Signing?

```c
if (*flag == 0xa55a)  
doInsecureDownload();
else  
doSecureDownload();
```
**Bibliography**


