Supply Chain Attack & Modern APT Malware Reversing

Inndy & Bletchley @ CyCraft
Whoami

► C.K Chen/Bletchley

► Senior Researcher in CyCraft
  • PHD from DSNSLab, NCTU
  • Research about reverse engineering, malware analysis and APT investigation

► CTF Player
  • Founder of BambooFox CTF Team in NCTU
  • Participate DEFCON final 2018 and 2016
  • Bug Bounty- Our team discover several vulnerabilities in Synology, QNap

► CHROOT member
  • Legend and Best private hacker group in Taiwan

► AIS3 lecturer & mentor, Hacker College
  • Education framework design(~2019)
  • Lecturer

► Reviewer of HITCON, HITB
Whoami

► Inndy Lin
  • Cyber Security Researcher at CyCarrier
  • Reverse Engineering Hobbyist
  • Learn to make online game hack since high school

► Speaker of HITCON, HITCON Training and LINE Becks

► I gave a lot of talk and training in Taiwan local security community
Outline

1. 1. Supply Chain Security
2. 2. Shadow Hammer Operation
3. 3. Web Storage Operation
4. 4. Threat Investigating Process
APT Against Taiwan

Taiwan suffers from state-sponsored APT every day
- BlackTech (PLEAD)
- Winnti Group (Winnti)
- APT 27 "LStudio" (Elise)

In the talk today, we will introduce 2 APT attacks we have investigated.
New Trend in APT Tactics

- Supply Chain Attack
- Synthesis normal program for malicious intent
In side your program, do you know where is every component come from?

Every step here is possible to be compromised
Malicious event-stream backdoor (2019)

A Post-Mortem of the Malicious event-stream backdoor

Last week the impossible happened. A malicious package, `stream`, was published to npm and was later added to a dependency in the widely used `event-stream` package by user `username`. Someone, and I believe it was a bot, uploaded hundreds of these malicious code to the npm registry. The exploit leverages the `stream` package to perform a denial-of-service attack by sending malicious data to the victim's system. As a result, it becomes difficult for the victim to access the application.

What is the event-stream package?

This event-stream package is vulnerable to a malicious payload. It requires all of the vulnerable versions of a client-side script running on the victim’s system.

Ruby strong_password Backdoor (2019)

Backdoor discovered in Ruby strong_password library

CI

Update

Dispatch

Loader

Executing
XcodeGhost (2015)

Malware-Laced Xcode Tool Used to Infect iOS Apps

21 września 2015

Malware that targets and successfully infects unmodified iOS devices is comparatively few and far between, with the few that does manage to create a site usually affecting jailbroken phones. Simply put, when malware manages to get through Apple's App Store policies and curation process, it's a big deal. While the iOS ecosystem is comparatively considered "safer" in the sense that there are simply far more threats designed to target other mobile operating systems, it's not perfect, and some still manage to fall through the cracks.
CCleaner Attack (2018)

**Timeline—How & When?**
CCleaner Supply-Chain Malware Attack

Last year, the popular system cleanup software CCleaner suffered a massive supply-chain malware attack of all times, when hackers compromised the company's servers for more than a month and replaced the original version of the software with the malicious one.
APTs Utilize Supply Chain Attacks

While most organizations gradually enhance their security, adversarial actors try to compromise the weakest point of a partner/supply chain first.

- ASUS Shadow Hammer (2019)
  - Discovered by Kaspersky

- ASUS Web Storage (2019)
  - We discovered this operation in the same time as ESET
Synthesis normal program for malicious intent

- To prevent security products' detection, APT groups often utilize some administrator tools for lateral movement.

- Use the built-in functionalities to achieve their malicious intent.

- Avoid being blocked or detecting, make investigation harder.
Widely Abused Applications

**Remote Execution**
- WMI
- PSEXEC
- WinRM
- Assess Management System

**Download**
- Update.exe
- Powerpnt.exe
- Bitsadmin
- Certutil.exe

**File Operation**
- Esentutl.exe
- Expand.exe

**Bypass**
- Regsvr32.exe
- Advpack.dll
- bash.exe

Check LOLBAS for more "useful" binary
[https://lolbas-project.github.io/](https://lolbas-project.github.io/)
Shadow Hammer
Shadow Hammer Overview

- A redditor notices that ASUS is publishing a critical update without any description via ASUS Live Update
- Kaspersky lab discovered malicious update on 2019/01
First Stage Dropper

- Sample: Setup.exe - 55a7aa5f0e52ba4d78c145811c830107
- This sample has the valid signature and certificate from ASUS
  - AV/EDR may trust singed executables which causes detection evasion
- The develop cycle may be compromised, or the certificate has been stolen
  - So attacker can somehow inject malicious payload without breaking signature
- Most of the program remains identical to the original setup.exe
  - It would take some time to locate malicious payload
Injected Code

Injected code was located at the end of .text section

Should go to __crtCorExitProcess normally
Shellcode Loader Behavior

- Shellcode was encrypted using Winnti's stream cipher algorithm and stored in PE resource under "EXE" type
- It uses GetModuleHandle(NULL) to get module base address
- Then uses precomputed offset to access resource and import table
  - Reduce references in static analysis tool, make it more stealth
- Shellcode will be executed in same process after decryption
void __stdcall decrypt(BYTE *input, int len, BYTE *output) {
    DWORD s1, s2, s3, s4;
    int i = 0; s1 = s2 = s3 = s4 = *(DWORD *)input;
    do {
        s1 += (s1 >> 3) - 0x11111111;
        s2 += (s2 >> 5) - 0x22222222;
        s3 += 0x33333333 - (s3 << 7);
        s4 += 0x44444444 - (s4 << 9);

        output[i] = (s4 + s3 + s2 + s1) ^ input[i];
    } while (++i < len);
}
First Stage Dropper

- Some of samples even contains entire PE structure
- PDB path in embedded PE
  - D:\C++\AsusShellCode\Release\AsusShellCode.pdb

(Tue.) Jul. 17th, 2018

Debug data of embedded PE from f2f879989d967e03b9ea0938399464ab
Second Stage Shellcode

This shellcode was generated from C++ compiler

Compiler prologue and epilogue structure

```
3 00000010: push ebp
4 00000011: mov ebp, esp
5 00000013: sub esp, 0x10
6 00000016: push ebx
7 00000017: mov ebx, dword ptr [ebp + 8]
8 0000001A: mov eax, dword ptr [ebx + 0x3c]
9 0000001D: mov eax, dword ptr [eax + ebx +
```

```
23 0000003F: test eax, eax
24 00000041: jg 0x4d
25 00000043: xor eax, eax
26 00000045: pop edi
27 00000046: pop esi
28 00000047: pop ebx
29 00000048: leave
30 00000049: ret
```
Shellcode Import Table

- Locating kernel32.dll via traverse through PEB->Ldr->InInitializationOrderModuleList
- Search LoadLibraryExW by function name hash
- Find other API by LoadLibraryExW and function name hash
- Make an import table for further usage

```c
imp_name_hash[15] = 0x9ACB1212;
imp_name_hash[16] = 0x87B21B7C;
imp_name_hash[17] = 0xD19124AF;
imp_name_hash[18] = 0xE8BAA2FA;
imp_name_hash[19] = 0x3D840FA5;
import_dlls[4] = U_wininet;
imp_table_idx = 0;
mod_idx = 0;
while ( 1 )
{
    curr_module = imp_table->LoadLibraryExW(import_dlls[mod_idx], 0, 8);
    imp_idx = 0;
    if ( imp_count_per_dll[mod_idx] > 0 )
        break;
```
Targeted Attack

- Enumerate MAC addresses by GetAdaptersAddresses
- Compute MD5 hash of MAC address
  - MD5Init, MD5Update, MD5Final from NTDLL
- Hardcoded MAC address hashes
  - Millions of users were affected, but only 600+ users are targeted
- When target matched:
  - Download shellcode from asushotfix[.]com and execute it
- Otherwise, write date to ..../idx.ini
Target MD5 Hash Dump and Decrypt

- Implement a DLL to hook MD5Init
- Search pattern of structure on the stack
- Break the 606 unique MD5 hashes
  - It takes ~4 hours with one GTX-1080Ti
- My friend told me that they used 4 GTX-1080Ti with less than an hour

```c
typedef struct {
    DWORD count; // 1 or 2
    BYTE mac1_md5[16];
    DWORD __padding1; // always 0
    BYTE mac2_md5[16];
    DWORD __padding2; // always 0
} MAC_RECORD;
```
Statistics about MAC addresses

268 ASUSTek COMPUTER INC.
159 Intel Corporate
101 AzureWave Technology Inc.
33 Liteon Technology Corporation
13 Hon Hai Precision Ind. Co., Ltd.
 7 BizLink (Kunshan) Co., Ltd
 6 <Unknown>
 5 AMPAK Technology, Inc.
 3 VMware, Inc.
 3 TwinHan Technology Co., Ltd
 2 REALTEK SEMICONDUCTOR CORP.
 2 Chicony Electronics Co., Ltd.
 1 TP-LINK TECHNOLOGIES CO., LTD.
 1 HUAWEI TECHNOLOGIES CO., LTD
 1 GOOD WAY IND. CO., LTD.
 1 Digital Data Communications Asia Co., Ltd
Shadow Hammer Conclusion

- We can't trust certificates blindly
- Millions people was affected, but only 600+ users targeted
- Attacker really knows the victim very well, even has a list of their MAC addresses
- Even now, victims are still unknown
PLEAD APT Group

Our system monitored that PLEAD APT group launch attacks simultaneously targeting several government organizations on 2019/4/24.

These attacks were started via ASUS WebStorage Update, and rarely detected by antivirus that time.

ESET was the first vendor to disclose this operation.
  - According to ESET, the update program is vulnerable to MITM attack.

Reused TSCookie malware was identified by our system.
PLEAD APT Activities

2019-04-24
Asus WebStorage AsusWSPanel.exe was invoked

Asus Webstorage Update.exe
WebStorage 2.4.3.612
2.4.3.612

%APPDATA%\Roaming\WebStorage\Asus Webstorage Update.exe
update.asuswebstorage.com ssmailer.com

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\ctfmon.exe
Point to TSCookie C2
159.117.79.7

InvokerUtility.exe
Onenote.exe (With valid certificate)
scsie.exe.dll
Samples We Got

- Initial malware distributed by ASUS Webstorage Update system
- The payload which downloaded by initial malware
- Others are the samples for persistency which captured by our system, from Taiwan government agencies
Initial malware spreaded by Asus Webstorage Update

ctfmon.exe in Startup folder with Excel icon

Second stage malware dropped from Asus Webstorage Update.exe
### Autorsu
- **Company**: Microsoft Corporation
- **Owner Name**: BUILTIN\Administrators
- **File MD5**: 071a8895b4a18796da5d867744934fd8
- **File Size**: 60 KB (61440 Bytes)
- **Create Time**: 2019-04-30 15:24:55
- **Last Access**: 2019-04-30 15:24:55
- **Last Write**: 2019-04-30 15:24:57
- **Time Stamp**: 2019-04-25 11:19:57
- **Autors**: 
  - HKEY_LOCAL_MACHINE\SYSTEM\CURRENTCONTROLSET\SERVICES\MSISCSI\n
### EXE (GUI)
- **Owner Name**: BUILTIN\Administrators
- **File MD5**: 64dftc7ba5a02e022f2f4e76443325
- **File Size**: 44 KB (45056 Bytes)
- **Create Time**: 2019-04-30 15:15:41
- **Last Access**: 2019-04-30 15:15:41
- **Last Write**: 2019-04-30 15:15:42
- **Time Stamp**: 2019-04-30 15:00:13

**Reference File**
Yet Another Malware with Signature

OneNote.exe signed with certificate from a Beijing company

OneNote.exe placed in a folder named "EPSON"?
ASUS Webstorage Update.exe

- Download update.asuswebstorage[.]com.ssmailer[.]com/fav.ico
- Split and decrypt data
- Drop files to
  - %appdata%\Microsoft\Windows\Start Menu\Programs\Startup\ctfmon.exe
  - %appdata%\Microsoft\Windows\Start Menu\Programs\Startup\slui.exe
  - %TEMP%\DEV + hex(GetTickCount() & 0xffff) + .TMP
ctfmon.exe

- TSCookie, a backdoor frequently appears in Japan APT operation
- Read data from Resource/PNG/143 and decrypt it with RC4
- Shellcode loads the DLL payload in memory
  - File-less attack leaves less trace, which can evade detection
ctfmon.exe - Obfuscation Technique

- Insert tons of useless function call to increase the difficulty of reversing
  - GetLastError
  - GetTickCount
  - printf(""")
  - Junk functions

- You will see hundreds of call instruction in debugger and decompiled output

- Use IDAPython to perform some magic and clean the useless function calls
if ( v11 ) {
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    if ( !NumberOfBytesRead )
        return -1;
    v13 = (int)v9 + v24 - 128;
    v14 = v24 - 128;
    *(DWORD *)(v13 + 124) = 0x5D765A92; // part of RC4 key
    rc4((int)v9, v14, v13, 128);
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    sub_40D70F(&unk_43824C);
    GetLastErrorCode();
    return -1;
    res_namea = LoadResource_0(module, v8);
    modulea = LockResource_0(res_namea);
    v11 = SizeofResource_0(v7, v9);
    if ( v11 < 0x800 || v11 > 0x4FD800 )
        FreeResource(res_namea);
        result = -2;
    }
}
else {
    lpMem = malloc_ensure(v11 + 1024);
    qmemcmp(lpMem, modulea, v11);
    FreeResource(res_namea);
    v12 = (int)lpMem + v11 - 128;
    *(DWORD *)(v12 + 124) = 1568037522;
    rc4((int)lpMem, v11 - 128, v12, 128);
        {v13 = lpMem[3];
        *a5 = v13;
        v14 = malloc_ensure(v13 + 128);
        *a4 = v14;
        memset(v14, 0, *a5);
        qmemcmp(*a4, lpMem + 10, *a5);
        *a7 = lpMem[4];
        v15 = malloc_ensure(*a7 + 128);
        *a6 = v15;
        memset(v15, 0, *a7);
        qmemcmp(*a6, (char *)lpMem + *a5 + 40, *a7);
        free(lpMem);
        free(v14);
        free(v15);\}
if ( v11 )
{
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    if ( !NumberOfBytesRead )
        return -1;
    v13 = (int)v9 + v24 - 128;
    v14 = v24 - 128;
    *(DWORD *)(v13 + 124) = 0x5D765A92; // part of RC4 key
    rc4((int)v9, v14, v13, 128);
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    sub_40D70F(&unk_43824C);
    GetLastError();
    return -1;
    res_namea = LoadResource_O(module, v8);
    modulen = LockResource_O(res_namea);
    v11 = SizeofResource_O(v7, v9);
    if ( v11 < 0x880 || v11 > 0x4FD800 )
        FreeResource(res_namea);
        result = -2;
    else
        lpMema = malloc ensured(v11 + 1024);
        qmemcpy(lpMema, modulea, v11);
        FreeResource(res_namea);
        v12 = (int)lpMema + v11 - 128;
        *(DWORD *)(v12 + 124) = 1568037522;
        rc4((int)lpMema, v11 - 128, v12, 128);
            v13 = lpMema[3];
            *a5 = v13;
            v14 = malloc ensured(v13 + 128);
            *a4 = v14;
            memset(v14, 0, *a5);
            qmemcpy(*a4, lpMema + 10, *a5);
            *a7 = lpMema[4];
            v15 = malloc ensured(*a7 + 128);
            *a6 = v15;
            memset(v15, 0, *a7);
            qmemcpy(*a6, (char *)lpMema + *a5 + 40, *a7);
scsiexe.dll

- Installed as service main DLL
- Encrypted shellcode
- InvokeUtility.exe has same structure and payload
Decrypted Shellcode

- Shellcode will decrypt its own body by standard RC4
- Other PLEAD backdoor may use modified RC4
- These two PLEAD backdoor have same payload but different key and C2 config
ASUS WebStorage Conclusion

- MITM could be very serious problem, especially in the update utility
- Additional data appended to the known file format could be a signal of abnormal
Malware Tricks
Modern Attackers are Lazy Smart

- Modern attackers don't write their own shellcode manually anymore
- They use compiler to generate shellcode, and shellcode is good for file-less attack!
- sRDI - Convert your DLL into shellcode
  - [https://github.com/monoxgas/sRDI](https://github.com/monoxgas/sRDI)
- MemoryModule - Parse and load PE module from memory without touching disk
  - So, you don't need LoadLibrary API anymore
  - [https://github.com/fancycode/MemoryModule](https://github.com/fancycode/MemoryModule)
IDAPython is your best friend

I have published some of my idapython script snippets

https://github.com/Inndy/idapython-cheatsheet
How to Find Malicious Payload in 1 Sec

- Windows system call number will change from time to time
  - Must use system call through NTDLL
  - Shellcode must locate DLL in memory from PEB structure and use Windows API
  - Malware usually will not list all the functions in the import table

- Set break points on...
  - Kernel32!VirtualAllocEx, Kernel32!VirtualProtectEx
  - Kernel32!LoadLibraryExW, Kernel32!GetProcAddress

- Not deep enough?
  - ntdll!NtProtectVirtualMemory
  - ntdll!NtAllocateVirtualMemory(Ex)
  - ntdll!LdrLoadDll
  - ntdll!LdrGetProcedureAddress
APT Investigation Process
Investigating Procedure

The process of proactively and iteratively formulating and validating threat hypotheses based on security relevant observation's and domain knowledge.
**APT investigation process**

**A**  
**Malware Analysis**  
For the special malware, the manual reverse will be conducted.

**B**  
**Threat Intelligence**  
Integrate several intelligences to track the potential source of attackers, which APT groups.

**C**  
**Threat Hunting**  
Regularly diagnosis end points to discover potential attacks.

**D**  
**Threat Investigation**  
Correlate the discovered events to depict the whole story line of APT. (causality)

**E**  
**Feedback**  
Feedback the information to victim to improve their security.
Threat Hunting System

Instead of passive detecting attacks, threat hunting aims to proactive & regular investigate if any attacker already conceals in your network.

- Network
- File
- Process
- Registry
- Memory
- ....

In our company

- Xensor
- Cycarrier
Threat Hunting System

Instead of passive detecting attacks, threat hunting aims to proactive & regular investigate if any attacker already conceals in your network.

► Network
► File
► Process
► Registry
► Memory
► ....
Threat Hunting System

EDR system, e.g. our Xensor EDR

Monitor what happens in a single end point
Threat Investigation

- Correlate the events and illustrate the whole attack storyline
- We use some machine learning for that
- After EDR conducts investigation on end points, threat investigation system (our Cycarrier) correlates the events and provides a platform for easily investigation
Cycarrier for Web Storage
Threat Intelligence System

- Previous 2 steps focus on intelligence from internal, it provides information for current APT attack.

- When we need to connect it to some APT groups, to investigate who is the attacker, the external information is indispensable
  - Who uses these malware
  - Who owns/operates these domain
  - Are there any APT group utilize the same network infrastructure
## Domain Activate Hunting

### asushotfix.com

<table>
<thead>
<tr>
<th>Name</th>
<th>IP Address</th>
<th>Resource Records Type</th>
<th>First Seen</th>
<th>Last Seen</th>
</tr>
</thead>
<tbody>
<tr>
<td>asushotfix.com</td>
<td>107.161.23.204</td>
<td>Address Mapping records</td>
<td>2018/05/03 18:35</td>
<td>2018/10/25 21:10</td>
</tr>
<tr>
<td>asushotfix.com</td>
<td>141.105.71.116</td>
<td>Address Mapping records</td>
<td>2018/05/23 07:16</td>
<td>2018/10/25 21:10</td>
</tr>
<tr>
<td>asushotfix.com</td>
<td>192.161.187.200</td>
<td>Address Mapping records</td>
<td>2018/05/03 18:35</td>
<td>2018/05/14 02:15</td>
</tr>
<tr>
<td>asushotfix.com</td>
<td>209.141.38.71</td>
<td>Address Mapping records</td>
<td>2018/05/03 18:35</td>
<td>2018/05/14 02:15</td>
</tr>
<tr>
<td>asushotfix.com</td>
<td>107.161.23.204</td>
<td>Address Mapping records</td>
<td>2018/05/03 18:43</td>
<td>2018/05/15 16:46</td>
</tr>
<tr>
<td>asushotfix.com</td>
<td>209.141.38.71</td>
<td>Address Mapping records</td>
<td>2018/05/03 18:43</td>
<td>2018/09/04 03:46</td>
</tr>
</tbody>
</table>

Our CTI shows that 103.28.46.4 (update.asuswebstorage.com.ssmailer.com) is highly suspicious.
Operation ShadowHammer Timeline

- **May 3, 2018**: Domain Registration Date Record by CyberTotal
- **Jan 27, 2019**: Kaspersky Lab. Confirm Problem.
- **Apr 8-11, 2019**: Announced is SAS about Operation ShadowHammer.
- **Mar 25, 2019**: Kaspersky Lab. Announce Report
Take Away

- While your software becomes more complex, it nearly impossible to establish a trusted/secure supply chain. Therefore, attackers will target the weakness of supply chain.
- We illustrate 2 APT operations to show the real world APT targeting supply chain security.
  - Shadow Hammer Operation
  - Web Storage Operation
- Modern APT malware tricks (for both offensive and defensive)
- Overview of our investigating procedure
THANK YOU

ck.chen@cycraft.com

Inndy.lin@cycarrier.com