Guloader Deobfuscation using Ghidra

By irfan_eternal

Published: 2023-07-23 · Archived: 2026-04-05 23:16:46 UTC

Hi all, Today we will be Analyzing Guloader Shellcode using Ghidra. Our Objective is to Identify some Anti-analysis and Obfuscation techniques used by Guloader and Defeat it using Automation. People who would like to follow along can download the sample from here . The File was was seen on 2023-05-11

The Shellcode is using API hashing to hide API's being called. For Each API Resolving it first resolves LdrLoadDll add 5 to it's address to avoid any Hooking done by EDR .Adding 5 to the API address is to avoid the classic 5 Byte Hook.Most EDR replaces first first bytes with a jump to EDR's code. It then use this address to Load the DLL. after Loading the DLL . it resolves the hash of the API it needs to call

```
dword __fastcall resolveAPIFunc(dword DlllName,dword ApI_hash)

{
  undefined4 in_EAX;
  int extraout_ECX;
  undefined4 unaff_EBX;
  int unaff_EBP;
  int unaff_ESI;
  undefined4 unaff_EDI;
  undefined2 *DllName2;
  dword dllName;

  *(undefined4 *)(unaff_EBP + 0x212) = unaff_EBX;
  *(undefined4 *)(unaff_EBP + 0x22d) = in_EAX;
  *(dword *)(unaff_EBP + 0x1df) = ApI_hash;
  *(undefined4 *)(unaff_EBP + 0x1c2) = 0x5c618f9c;
  *(uint *)(unaff_EBP + 0x1c2) = *(uint *)(unaff_EBP + 0x1c2) ^ 0xe289b05a;
  *(uint *)(unaff_EBP + 0x1c2) = *(uint *)(unaff_EBP + 0x1c2) ^ 0x26e75f13;
  *(int *)(unaff_EBP + 0x1c2) = *(int *)(unaff_EBP + 0x1c2) + 0x67f0a464;
  if (*(int *)(unaff_EBP + 0x22d) == *(int *)(unaff_EBP + 0x1c2)) {
                    /* ntdll */
    resolveAPiLibrary(0x9cda770c);
                    /* ldrloaddll */
    resolveHashedApi's(0x55db1585);
                    /* anti-hooking for LdrLoadDll */
    *(int *)(unaff_EBP + 0xc) = unaff_ESI + 5;
  }
  DllName2 = *(undefined2 **)(unaff_EBP + 0x164);
  *(dword *)(DllName2 + 2) = DlllName;
  returnstrlength(DlllName);
  *DllName2 = (short)extraout_ECX;
  *(undefined4 *)(unaff_EBP + 0x235) = unaff_EDI;
  DllName2[1] = (short)(extraout_ECX + 2);
  *(undefined2 **)(unaff_EBP + 0x17f) = DllName2;
  dllName = *(dword *)(unaff_EBP + 0x17f);
  *(int *)(unaff_EBP + 0x24f) = extraout_ECX + 2;
  LdrLoadDll(0,0,(char)dllName,(char)unaff_EBP + 'h');
  resolveHashedApi's(ApI_hash);
  return dllName;
}
```

API Hashing function is the same as we see in the wild. it goes to the Export Directory of the DLL Loaded and performs Hashing of all API name's till the hashes match .if the hashes match it stores the Address of the API

Hashing Algorithm : In the past Guloader was using just DJB2 hash . Now it xors the result of djb2 hash with a hardcoded value to perform API Hashing

I wrote a python script to identify the Hashes of the API's used by the past Guloaders to identify the Functionalities of the Code

```
 val = 0x1505
APIstrings = ["NtGetContextThread", "RtlAddVectoredExceptionHandler", "NtAllocateVirtualMemory", "DbgUIRemoteBreakIn", "LdrLoadDll", "Dbg
             "NtResumeThread", "NtProtectVirtualMemory", "CreateProcessInternal", "GetLongPathNameW", "Sleep", "NtCreateThreadEx", "WaitF
             "RegCreateKeyExA","RegSetValueExA", "NtQueryInformationProcess", "InternetOpenA", "InternetSetOptionA", "InternetOpenUrlA",

for APIstring in APIstrings:
    val  = 0x1505
    for ch in instring:
        val += ((val <<5))
        val &= 0xFFFFFFFF
        val += ord(ch)
        val &= 0xFFFFFFFF
        val ^= 0x8131A1
    print(APIstring+" :  "+hex(val))
```

Before Calling the API it performs some Checks to make sure it is not being Debugged

It compares the first few bytes of the API address with a) CC b) INT 3 c) UD2 to check if it’s being Debugged. if this is not the case it calls the API

Control Flow Obfuscation using Vectored Exception Handling

To Obfuscate the Control Flow it uses RtlAddVectoredExceptionHandler to register a vectored Exception Handler. If the Exception raised is any of below three cases it changes the EIP by an XOR operation

1. EXCEPTION_ACCESS_VIOLATION while accessed memory Address is 0
2. EXCEPTION_SINGLE_STEP (Single Stepping)
3. EXCEPTION_BREAKPOINT (Software Break Point - CC)

If it is EXCEPTION_BREAKPOINT it calculates the value of the EIP by this expression EIP = EIP + *(EIP+1) ^ 6A (Changes with sample)

If it is EXCEPTION_ACCESS_VIOLATION or EXCEPTION_SINGLE_STEP it calculates the value of the EIP by this expression EIP = EIP + *(EIP+2) ^ 6A (Changes with sample) .

In the Vectored Exception Handler function it also checks if any Dynamic anyalsis is being performed by Checking if any hardware breakpoint is set using ContextRecord a member of _EXCEPTION_POINTERS

To Understand Vectored Exception in Detail i would suggest to read this article

All the Important strings including the C2 URL is Encrypted using XOR. The Encrypted strings are created by Performing a lot of Mathematical Expression(not Hardcoded).After Encryption String is Fully Created . The first Dword contains the length of the Encrypted string. then what follows is the Encrypted string. The Address of the Encrypted string is given as parameter to the String Decrpyion preperation Function. It store the Encrypted String length in a Varaible. And changes the First Dword to a Dummy Value . After that increments the Address of Encrypted String by a dword now it points to the Actual start of the String. Encrypted_string and the Encrypted String length is given an paramerers to the next Function called which is a wrapper around the string decryption function

The wrapper function passes the Encrypted_string, Encrypted String length and the Key. The key is actually stored in the return address of the wrapper function

The Decrytion Algorithm is a Simple XOR were Encrypted string is calculated by Mathematical Expression and key is stored in the the return address of the wrapper function

Analyzing the shellcode dynamically will be very tiring because it has multiple checks for Anti-Analysis . Static Analysis will also take much time due to Control Flow Obfusucation. So I wrote 2 Scripts to help in this Analysis . one is to Deobfuscate the Control Flow and the other is to Decrypt the Strings

Please Follow the Below Steps to Reduce your time on Guloader

1. Import the Shellcode to Ghidra
2. Disassemble (Key Binding D) the Start of the Shell code
3. Run the Guloader_deobfusucate.py script Providing the Decryption Key and key for EIP modification
4. Run the Gu_string_decryption.py script Providing the Decryption Key

NOTE: For Both Cases Provide the keys as Hex String with out Ox Example if the key is 0x6A Provide it as 6A. My Scripts are not the best way to Achieve this . Note to Self need to improve to write better code

After these Steps the Control Flow will be Deobfusucated and Decrypted Strings and Payload key will be printed in the Console . If you have a closer look at the below image you will see the C2 URl will not be starting with http:// this is to prevent it from XOR Bruteforcing

I Have only Checked one sample with the scripts .Feel free to use it with Other Guloader Shell Code and let me know if you are able to see the Decrypted strings

File SHA1 : 992d98aa6f31ae6f8f42fac9866a19c2a2f879be

1. SonicWall
2. CheckPoint
3. AnyRun
