Unloading the GuLoader

By VIPRE Labs

Published: 2020-05-20 · Archived: 2026-04-05 18:25:10 UTC

We recently came across a spike of spam email samples containing GuLoader. This malware was discovered last year in 2019 and became more popular among cyber criminals during the coronavirus outbreak. GuLoader is usually attached to a spam email related to bill payments, wire transfers or COVID malspam (you can see a detailed analysis of the COVID malspam here). GuLoader is written in VB5/6 and compressed in a .rar/.iso file. We can see on the graph below the increase of GuLoader which our customers have received:

Figure 1.0 Data collected from January to April 2020 showing the increase in GuLoader related samples

Figure 2.0 Spam emails containing GuLoader

Guloader is popular for distributing Remote Access Trojan (*RAT*) tools. These allow the attackers to control, monitor, or steal information from the infected machine. This malware downloader utilizes cloud hosting services (*Microsoft OneDrive or Google Drive*) to keep its payload encrypted.

Dig Deeper Inside of GuLoader

Analyzing the GuLoader sample, the malware is indeed a VB5/6 executable. Also, a compiled Visual Basic sample can be recognized by an imported DLL called *MSVBVM60.DLL*.

Figure 3.0 GuLoader sample written in VB5/6 and the msvbvm60.dll

Analyzing further, we've found the malware's encrypted malicious code. This malware allocates virtual memory and decrypts the encrypted malicious code using XOR.

Figure 4.0 The decryption routine

Figure 5.0 The encrypted malicious code (left) and the decrypted malicious code in allocated virtual memory (right).

The decrypted code will be in virtual memory 0x350000. Checking this memory in memory map, it has read, write, and execute (*RWE*) access. We've now dumped the decrypted code to conduct analysis.


The dumped memory and the familiar strings that were found in the decrypted code

Figure 6.0 The dumped memory and the familiar strings that were found in the decrypted code

Checking the strings on the decrypted code, we can see clearly the cloud hosting service URL that stores the encrypted payload (*hxxps://drive[.]google[.]com/uc?export=download&id=19sVk-ZTWHVl3_ITBst1x51qX2L28yNlw*). We can also see familiar DLLs like wininet.dll and APIs like InternetOpenA, InternetOpenUrlA, InternetSetOptionA etc. The wininet.dll contains internet related functions like InternetOpenA and these functions will probably be used to connect to the URL that contains the encrypted payload.

Analyzing what's inside of the decrypted code, we can see that the malware will find the GetProcAddress function in kernel32.dll because GetProcAddress is important in finding and calling other API functions. In order to do this, the malware will first access the Process Environment Block (*PEB*) -> LDR data -> InMemoryOrderModuleList and then get the address of the module kernel32.dll.


Accessing the PEB and getting the address of kernel32.dll

Figure 7.0 Accessing the PEB and getting the address of kernel32.dll

After obtaining the address of kernel32.dll and finding GetProcAddress in kernel32.dll, the malware will resolve the following series of APIs:

- LoadLibraryA
- TerminateProcess
- EnumWindows
- NtProtectVirtualMemory
- NtSetInformationThread
- NtAllocateVirtualMemory
- DbgBreakPoint
- DbgUiRemoteBreakin

After this, we ran into some anti-analysis techniques. The anti-analysis was used by malware authors to make it more difficult to analyze the malware.

Here are some of the techniques we encountered:

- An anti-debugger that hides the thread from the debugger. In order to perform this, the API NtSetInformationThread is needed. They set the second parameter (*ThreadInformationClass*) to 0x11 which is equivalent to ThreadHideFromDebugger. It will hide the thread from the debugger so it can't be easily debugged. For example, the thread will continue to run, but the debugger will not be able to receive any events related to the thread.


Calling of NtSetInformationThread to hide the thread from the debugger

Figure 8.0 Calling of NtSetInformationThread to hide the thread from the debugger

- Thread attach in a debugger can be seen in the thread window. On *figure 9.0*, we can see the before and after the thread is hidden from the debugger. The before part is where we can see the main thread and its thread ID which is 11DC. The after part is where the main thread is hidden from the debugger.

   Before and after the hiding of thread

Figure 9.0 Before and after the hiding of thread

- There's another technique that will first call the *NtProtectVirtualMemory* function to set the permission of ntdll's .text section as *PAGE_EXECUTE_READWRITE*. The ntdll.dll contains the following APIs, *DbgBreakPoint* and *DbgUiRemoteBreakin*, that will be used to perform anti-attach. The malware prevents the debugger from attaching to a process by hooking the *DbgBreakPoint* and *DbgUiRemoteBreakin* functions. For example, it will patch *DbgBreakPoint* and *DbgUIRemoteBreakin* functions that will trigger the process to exit or to designate an unknown location. Like in *figure 12.0*, *DbgUIRemoteBreakin* will call *0x00000000* address and exit.

   Calling of NtProtectVirtualMemory to set the permission of NTDLL.DLL

Figure 10.0 Calling of NtProtectVirtualMemory to set the permission of NTDLL.DLL

 Patching of DbgBreakPoint and DbgUiRemoteBreakin for anti-attach technique

Figure 11.0 Patching of DbgBreakPoint and DbgUiRemoteBreakin for anti-attach technique

 Before and after patching the DbgUlRemote Breakin function

Figure 12.0 Before and after patching the DbgUlRemote Breakin function

GuLoader will create a folder in the C:\Users directory and the created folder contains a copy of the malware itself. It will also achieve persistence by modifying the registry key HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce

 The created folder containing the created malware copy

Figure 13.0 The created folder containing the created malware copy

Now GuLoader implements process hollowing:

 The child process (for this sample it's RegAsm.exe) downloads and decrypts the encrypted payload

The child process (for this sample it's *RegAsm.exe*) downloads and decrypts the encrypted payload from a cloud hosting service, and maps the decrypted payload into memory to execute.

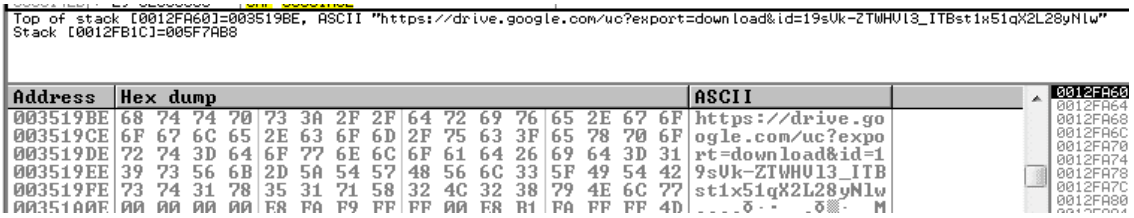

Figure 15.0 The cloud hosting service storing the encrypted payload

The common GuLoader payloads are Formbook, NetWire, Remcos, Lokibot etc.

IOCs:

URLs

- hxxps://onedrive[.]live[.]com/download?cid=1491235303209D1A&resid=1491235303209D1A!109&authkey=ACw2GiM8jfgliBs
- hxxps://drive[.]google[.]com/uc?export=download&id=1EQ7DIlAk9lk2E52DQLELmB02ADqw-62s
- hxxps://drive[.]google[.]com/uc?export=download&id=19sVk-ZTWHVl3_ITBst1x51qX2L28yNlw

Samples

- IMG and ISO Files
  - 466a8de97917fdbc706ccad735ef08a4b049f802d01a03e4f611f75a132e4839
  - 7aadacc7c5bb0c0319f8943d3c65ef2d41d49b1c470210e70e250dd665f167fe
- EXE Files
  - 503f94f00304bc18900c3494f2da5bcb1d8a103a0b15ce00bbdaeb5dfd8d9b7b
  - cbffd8f471de9728610b1edd4519f65399a8e64e46177e1178685ef6b081065b

VIPRE detects and prevents this kind of malware and associated infections.

Analysis by #Farrallel
