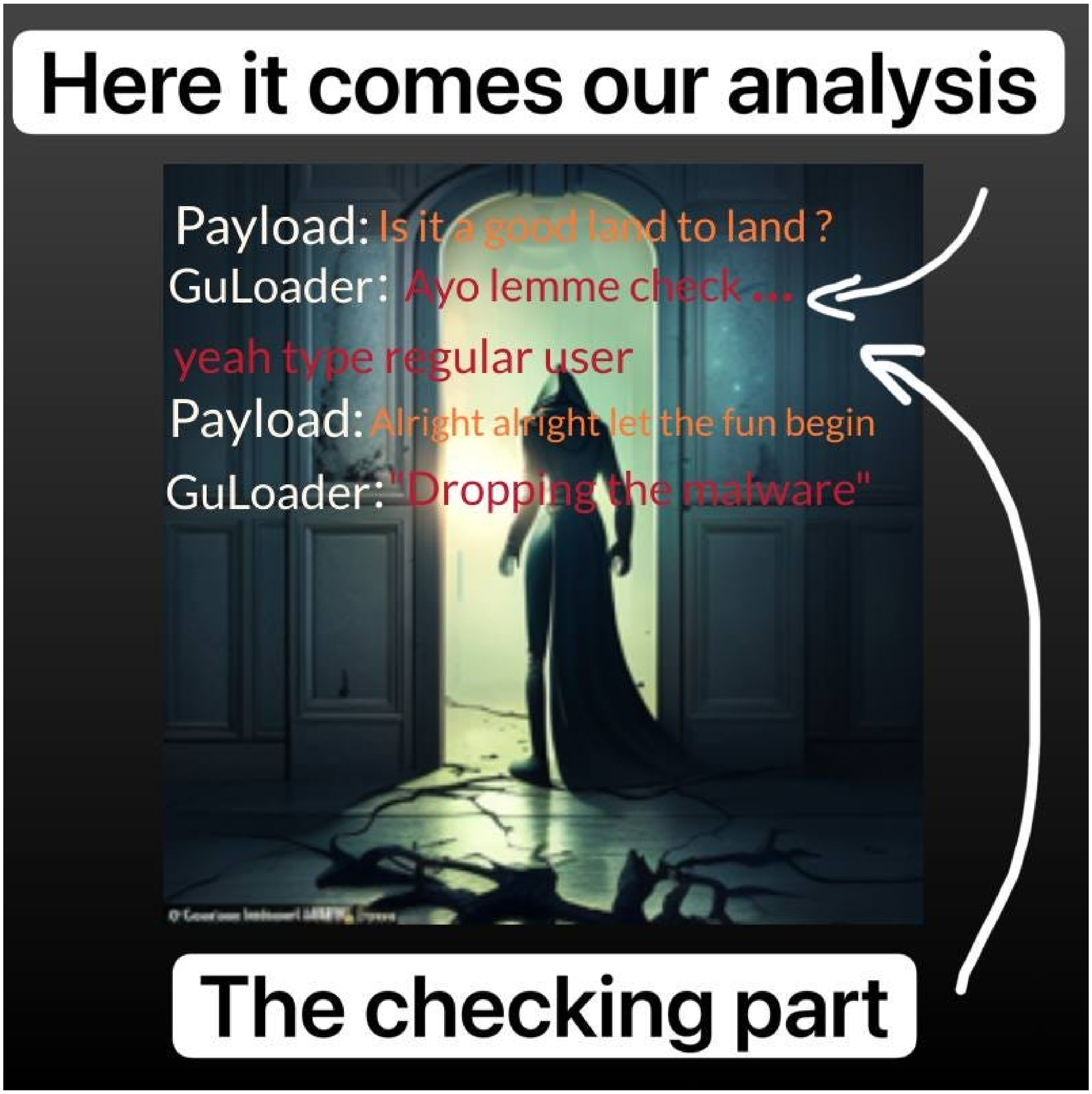
Analyzing GuLoader

medium.com/@ZainWare/analyzing-guloader-42c1d6a73dfa

(0xA1N-MalZaMes-ZainWare) March 11, 2023


Here it comes our analysis
Payload: Is it a good land to land ?
GuLoader: Ayo lemme check ...
yeah type regular user
Payload: Alright alright let the fun begin
GuLoader: "Dropping the malware"
The checking part

(0xA1N-MalZaMes-ZainWare)

Mar 11

.

11 min read

GuLoader [figure 1]

So we need to know what does GuLoader do for living, but before that let us discuss and talk about its history.

GuLoader history:

GuLoader also known as CloudEyE and vbdropper. First seen in the wild was in December 2019 and it's delivered via spam emails (Malspam) as an attachment fooling users and acting like a legit mail.

Here's some examples:

Malspam using DHL theme to push GuLoader, MalwareBytes [Figure 2]
another Malspam, CrowdStrike [Figure 3]
another Malspam, PCrisk [Figure 4]
From AhnLab [Figure 5]

> GuLoader got its name as Google Drive is frequently used as its download URL. (ASEC analysis team)

GuLoader is written in VB6(wrapper)containing the shellcode that gives GuLoader the needed flexibility to be heavily obfuscated to evade and escape from AVs and that gives GuLoader low detection on VT.

Guloader has a lot of capabilities that we will discuss in a moment, After doing its work it drops a lot of different malwares like RATs, stealers and ransomwares (ex: Formbook, Remcos, Lokibot, NanoCore, AgentTesla, Arkei/Vidar, NetWire, Hakbit, etc.) using clouds like google drive(not always), at 2020(when GuLoader distributed and was so active) using cloud services was a trend to deliver your malware and from a legitimate website was a big benefit (still a good thing for adversaries of course).

At 2020 Check Point exposed GuLoader as it's related to CloudEye (CloudEyE is an italian security software company intended for "*Protecting windows applications from cracking, tampering, debugging, disassembling, dumping*").

here's the link

We will discuss GuLoader's career through the past years 2020, 2021, 2022, and the latest Italian e-commerce attack(Feb 2023) using NSIS.

A lot of its capabilities will continue through years and GuLoader's author will modify some code implementations for them and some interesting techniques will be added.

So let's start our analysis…

At 2020 (GuLoader's birth)

According to Lawrence Abrams he said that first discovered this spam campaign is MalwareHunterTeam. I tried to get an early sample of it and the earliest sample i got(my starting point) from Jan 15 2020:

md5:cf3e7341f48bcc58822c4aecb4eb6241

GuLoader once executed will download an encrypted file from google drive and decrypt it then inject it(not always) into wininit.exe(lgeitimate windows process) and evade detection.They also impersonated WHO in Covid-19 pandemic.

GuLoader as WHO (Malwarebytes) [Figure 6]

Process injection:

How does it excute?

GuLoader uses process hollowing but Here shellcode doesn't unmap memory code of legitimate processes; instead it uses the NtCreateSection:

By calling CreateProcessInternalW to create the legitimate process RegAsm.exe(C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe) or if it doesn't find it will loop(in the same path) to find MSbuild.exe or RegSvcs.com(they all a part of.NET framework maybe will distribute .NET malware) and create it in susbended state CREATE_SUSPENDED(0x00000004) then will use zwOpenFile to open a file handle to mstsc.exe or msvbvm60.dll then will call NtCreateSection for this fila handle to create a section of memory with the desired access parameter then this section mapped in the legitimate process that in suspended state (ex: RegAsm.exe) using NtMapViewOfSection with base address 0x400000 instead of a normal high load address then will call NtWriteVirtualMemory to write the shellcode in this new section. Now after writting the shellcode in this newly allocated memory GuLoader needs to excutes it by using NtSetContextThread to change the context of the only thread running in the targeted process that is still in a suspended state. now this context change sets the EIP register to the address that points to the begining of the shellcode that will make the shellcode runs by using NtResumeThread. The shellcode will do some anti debugging, analysis, vm and some other interesting techniques.that's what we will talk about next.

Anti AVs:

GuLoader is wrapped in a VB6 and changing in every campaign to evade AV detections and inside it the shellcode and
the downloaded payloads are encrypted with a hard coded 4 bytes XOR key embedded inside the malware to make it difficult for AVs so it won't identify the payload as malicious.

Anti Debugging:

GuLoader does hooks for functions DbgBreakPoint and DbgUiRemoteBreakin. And patches these two APIs by replacing the INT3 opcode of DbgBreakPoint with opcode 90 (NOP or no operation to do nothing) and for DbgUiRemoteBreakin it replaces the first few bytes with a dummy call to cause a crash.

Crash message [Figure 7]
and another thing that GuLoader does is to hide its running thread from debuggers by calling NtSetInformationThreadpushing with the value (0x11) as the second parameter and that value is ThreadHideFromDebugger.
Also GuLoader checks for both hardware and software breakpoints by checking DR0-DR7 registers (that registers responsible for hardware breakpoints by debuggers)

```
//Hardware breakpoint checkscmp dword ptr ds:[eax+4],0jne Crash_fncmp dword ptr ds:
[eax+8],0jne Crash_fntest bl,blcmp dword ptr ds:[eax+C],0jne Crash_fncmp dword ptr
ds:[eax+10],0jne Crash_fntest ecx,ecxcmp dword ptr ds:[eax+14],0jne Crash_fncmp dword
ptr ds:[eax+18],0jne Crash_fncmp ah,ah
```

[eax+4] = DR 0
[eax+8] = DR 1
[eax+C] = DR 2
[eax+10] = DR 3
[eax+14] = DR 4
[eax+18] = DR 5

and this Crash_fn that responsible for showing the crash message box[Figure 7].

and the software breakpoints by looking for 0xCC (INT3), 0x3C and 0xB0F and if a breakpoint is found it will cause a crash (same as the message above)[Figure 7].

```
pop eaxtest bx,bxmov bl,byte ptr ds:[eax]cmp bl,CC                  //CC je
Crash_fnmov bx,word ptr ds:[eax]cmp bx,3CD               //3CDje Crash_fncmp
edx,ebxmov bx,word ptr ds:[eax]cmp bx,B0F              //B0Fje Crash_fn// I didn't
find assembly so i chose less cuz it's making the code colorful. :)
```

this Crash_fn that responsible for showing the crash message box[Figure 7].

Anti Sandboxing and VMs (virtualization):

GuLoader is using EnumWindows API to count all the windows on the screen and if it’s less than 12 (by checking EAX value if less than 12)it calls TerminateProcess.

```
call eax        // EAX containing <user32.EnumWindows>pop eaxcmp eax,C      // 12jge
TerminateProcess_fn
```

Using CPUID:

calling CPUID then do

bt ecx,1F // 31

cause if it’s running inside a debugger or vm the 31st bit will be set to one

(by default it’s zero)

and then showing the crash message box[Figure 7].

calling CPUID will generate a VM exit to the VM manager and that takes much more time than running in a physical machine.

Using RDTSC:

RDTSC(Read Time-Stamp Counter) is the same and takes so much time and slower if it’s in a vm but in GuLoader it’s using:

mov ecx,186A0

(that’s 100,000 times). Storing it in ECX it’s the number of times EDI will be incremented with.

then decrement ecx and then compare it with 0 if not zero will return again till ecx becomes zero

and we said that edi will be incremented, it’ll be incremented till it comares it with

cmp edi,68E7780

(that’s 110,000,000 times). cause if it reachs that number will enter the loop again and it will be an infinte loop.

As you can see GuLoader choosing specific time and numbers for that so of course these numbers mean something. I guess he estimated everything in a virsualized environment.

GuLoader also uses instruction hammering we have seen this techniques in so many malware samples by executing massive amounts of delay that exceeds the execution time on a sandbox or any analyzer so the payload execution is never reached. GuLoader also uses ZwQueryVirtualMemory to lacate any pages that contains any strings that related to VMs(scaning the process's memory)with the handle 0xffffffff(to retrieve the base address of every page) and if it finds any , will create the Crash message box[Figure 7].

Api resolving:

GuLoader uses a **DJB2** hash algorithm for resolving APIs to hide them from the IAT(import address table). When it needs any windows API it resolves it first as it first generates the hashs for every function in Kernel32.dll then when it needs any API it'll call it by its hash value.Like if it needs CreateProcessInternalW will use 9688DA44.

DJB2

```
unsigned long
hash(unsigned char *str)
{
  unsigned long hash = 5381;
  int c;



  while (c=*str++)
    hash = ((hash << 5) + hash) + c;

  return hash;}
```

using 5381 in this alogrithm cause fewer collisions and better avalanching

using 33 never explained

using (hash << 5) + hash that's bit shifting and it's faster for CPU than performing hash*32 and that's shifting it five bits to left as multiplying it with 2^5 (32) and adding it to itself means it becoming 33.

Entering Heaven:

Guloader is a 32-bit application and if it's running on 64-bit systems could trick the operating system into executing 64-bit code, despite initially declaring itself as 32-bit process. How does it happen? by calling 32-bit ntdll.dll then wow64.dll (windows on windows) and that will call wow64cpu.dll and then call wow64Transition and that's the heaven's gate and then performs a far jmp instruction that switches into 64-bit and use ntdll.dll but the 64 version of it.How do you know if it's using Heaven's gate or not?
by using FS:[0xC0] register value(containing a pointer to FastSysCall in wow64) to see if the system is x64 or not so if it's a 64 it will use heaven's gate.

(mov ebx, dword ptr FS:[C0]).

in this year 2020, when CheckPoint exposed CloudEyE at June 8 2020,GuLoader was boomin at June 9:

Malware Bazzar[Figure 8]
CheckPoint[Figure 9]
Here you can see it was active at that month and at June 10 Sebastiano said:

Service Suspension temporarily[Figure 10]
but CheckPoint came again after this Message:

Taking down The top malicious dropper of 2020 said by CheckPoint[Figure 11]
Resume_service Malwarebytes[Figure 12]
So i think that’s enough for 2020 let’s discuss 2021.

Number of GuLoader samples submitted to MalwareBazaar during Jan-Oct 2021(deepinstict)[Figure 13]
2021’s code is the same as 2020 plus some features. I’ll discuss these features as the previous ones i already talked about above.

Anti VMs:

GuLoader searches for Qemu emulator and virtualizer

C:\Program Files\Qemu-ga\qemu-ga.exe

and also searching for qga (Qemu gues agent).

C:\Program Files\qga\qga.exe

Checking the installed softwares by using MsiEnumProductsA, MsiGetProductInfoA.

The MsiEnumProductsA function enumerates through all the products currently installed.
The MsiGetProductInfoA function returns product information for published and installed products.

by using OpenSCManagerA GuLoader opens a specific service control manager database and checks for running processes by using EnumServicesStatusA.

OpenSCManagerA establishes a connection to the service control manager on the specified computer and opens the specified service control manager database.

EnumServicesStatusA function Enumerates services in the specified service control manager database. The name and status of each service are provided, along with additional data based on the specified information level. Examples of some services GuLoader is looking for:VMware tools, VMware snapshot provider.

Checking windows drivers by using EnumDeviceDrivers and GetDeviceDriverBaseName.

EnumDeviceDrivers receives the list of load addresses for the device drivers.

GetDeviceDriverBaseName Retrieves the base name of the specified device driver.

Examples of some drivers GuLoader is looking for:vm3dmp.sys, vm3dmp_loader.sys, vmmouse.sys and vmusbmouse.sys.

Anti analysis:

GuLoader uses a lot of useless and dummy calls like pushfd followed by popfd, using mov edx,edx or repeating some instructions or doing some arithmetic calculations for nothing. And using opaque predicate.

Opaque predicates are conditions that always(constant) true or false and obfuscate it to hide that they are constant and fool the disassembler that there's two paths but in fact there's only one path.

All used APIs is hashed by DJB2 as we know but before using any of them GuLoader XOR the shellcode then calling them and then xoring it again to return the instructions back. At this process the disassembler treats the xored instructions as data and doesn't define it as functions.

```
some code instructions then followed by
;...............................................       db       db    random hex
db
```

2022

In 2022 Guloader started to use NSIS.

What is NSIS used for?

> NSIS (Nullsoft Scriptable Install System) is a tool that **allows programmers to create such installers for Windows**. It is released under an open source license and is completely free for any use. NSIS creates installers that are capable of installing, uninstalling, setting system settings, extracting files, etc.

trellix[Figure 14]

Anti debugging:

like any animal that lives in caves when searching for food he goes out and then return again to its cave not to be noticed.GuLoader when it uses any data will first decrypt it and then encrypt it again covering itself from being noticed. Another example:the first 40 bytes of the shellcode is responsible for decrypting the code and after doing its job all the 40 bytes changed to NOPs(90 No operation) like it's destroying its key.

By using VEH(Vectored Exception Handling a function to handle all exceptions for the application) GuLoader waste researchers time by raising expections and distrub the control flow of the sc pointing and jumping to other instructions handling it by using RtlAddVectoredExcepitionHandler API. Another use of Exception handler GuLoader uses it for software breakpoints and hardware breakpoints if an exception is raised by any of them will continue without jumping and excutes some invalid instructions and if non of them are set it'll jump to a new value and continue.

By using NtQueryInformationProcess GuLoader checks for a presence of a remote debugger and by using 0x7 as the second parameter (ProcessDebugPort) by checking the return values if it's non-zero it means the process is being debugged.

Beside using RegAsm.exe(for process injection) GuLoader also used caspol.exe and aspnet_compiler.exe. And after that it calls NtOpenFile on C:\Windows\syswow64\iertutil.dll(I already explained how process injection happens in GuLoader above).

Conclusion:

GuLoader doesn't targeting a specific nation or a specific industry but most of its attacks is E-Commerce.

[Figure 15]
It's just a service everyone can buy it(with minimum 100$) and do whatever he wants (if really GuLoader is part of CloudEyE).

GuLoader started long ago and still active and will remain active because it uses polymorphic Shellcode that always change and its DJB2 algorithm XORing the hashes everytime with a different key so no need for a table to save its API hashes because it's changing everytime and also using NSIS and improving it over time. Using a lot of anti debugging it's trivial and well known but with all these techniques it's pain in the ass and time consuming for reversing it.

Additional reading & Resources

GuLoader? No, CloudEyE.

Malware Analysis: GuLoader Dissection Reveals New Anti-Analysis Techniques and Code Injection Redundancy

GuLoader: Peering Into a Shellcode-based Downloader

GuLoader: The NSIS Vantage Point

Defeating Guloader Anti-Analysis Technique

GuLoader: A fileless shellcode based malware in action

[Down]loaded by GuLoader Malware | DeepInstinct

The evolution of GuLoader

Spoofed Saudi Purchase Order Drops GuLoader: Part 1

Spoofed Saudi Purchase Order Drops GuLoader — Part 2

GuLoader: The RAT Downloader

Dissecting the new shellcode-based variant of GuLoader (CloudEyE)

Dancing With Shellcodes: Cracking the latest version of Guloader

Playing with GuLoader Anti-VM techniques

and so many others but i don’t remember :(