# GuLoader: The RAT Downloader

blog.morphisec.com/guloader-the-rat-downloader

- Tweet
-

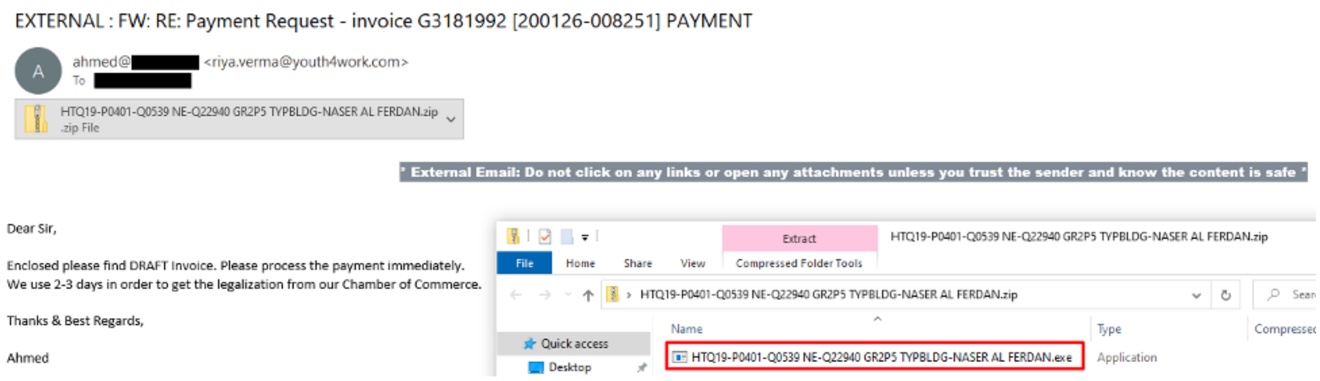
Guloader is a downloader that has been widely used from December 2019. Several security researchers have identified the downloader in the wild, signifying that it has quickly gained popularity among threat actors. When it first appeared, GuLoader was used to download Parallax RAT, but has been applied to other remote access trojans and info-stealers such as Netwire, FormBook, and Tesla.

Recently, the Morphisec Labs team noticed the downloader spreading in a targeted phishing email against a major bank. Although Parallax RAT was among the first malwares used with GuLoader, we noticed this particular campaign had Remcos RAT as the final payload. GuLoader is considered one of the most advanced downloaders, written in Visual Basic, and this often makes it difficult to scan for static analysis.

In this blog post, we are going to cover GuLoader's internals. This includes its shellcode, evasion techniques, and its delivery mechanism.

## Technical Analysis: (2d6720ae866875b42601951eb3c9421fa2d5b7f0)

The first stage in this campaign is an email that claims it's a payment invoice. This email contains a ZIP file attachment; as with other phishing emails, the goal is to get the target to download the attachment and open the file. The email appears as part of a chain, which makes it more likely for the target to open the attachment when it's received.



EXTERNAL : FW: RE: Payment Request - invoice G3181992 [200126-008251] PAYMENT

ahmed@[redacted] <riya.verma@youth4work.com>
To [redacted]

HTQ19-P0401-Q0539 NE-Q22940 GR2P5 TYPBLDG-NASER AL FERDAN.zip
.zip File

* External Email: Do not click on any links or open any attachments unless you trust the sender and know the content is safe *

Dear Sir,

Enclosed please find DRAFT Invoice. Please process the payment immediately.
We use 2-3 days in order to get the legalization from our Chamber of Commerce.

Thanks & Best Regards,

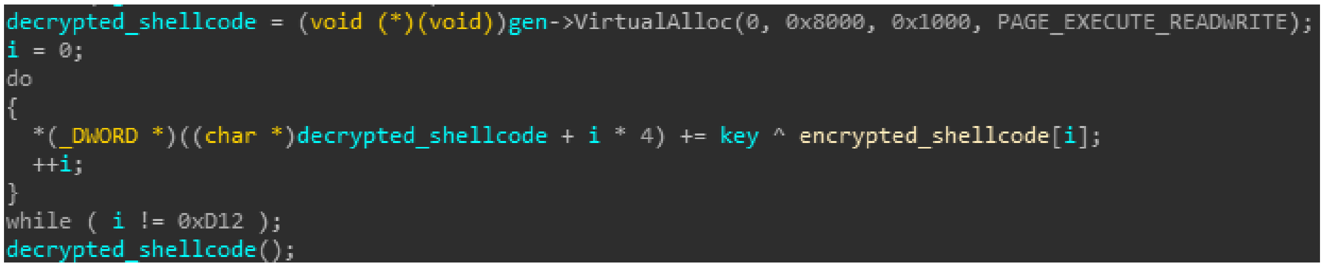
Ahmed

HTQ19-P0401-Q0539 NE-Q22940 GR2P5 TYPBLDG-NASER AL FERDAN.zip

HTQ19-P0401-Q0539 NE-Q22940 GR2P5 TYPBLDG-NASER AL FERDAN.exe Application

*Figure 1: The email pretends to be a payment request.*

The ZIP file attachment contains a VB6 executable that stores an encrypted shellcode. The shellcode is XORed with a 4 byte key and then executed. Jason Reaves (@sysopfb) wrote an unpacker for this stage, allowing us to see the URLs for GuLoader.



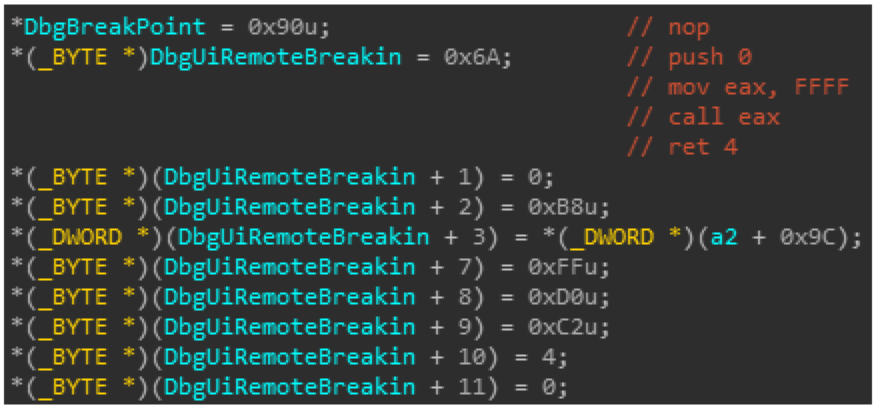
```
decrypted_shellcode = (void (*)(void))gen->VirtualAlloc(0, 0x8000, 0x1000, PAGE_EXECUTE_READWRITE);
i = 0;
do
{
  *(_DWORD *)((char *)decrypted_shellcode + i * 4) += key ^ encrypted_shellcode[i];
  ++i;
}
while ( i != 0xD12 );
decrypted_shellcode();
```

*Figure 2: Decrypting and executing the shellcode.*

## GuLoader Shellcode

**GuLoader** has some strong evasive characteristics in its shellcode, with several anti-debugging tricks in order to make it harder for defensive solutions to analyze it. These include:

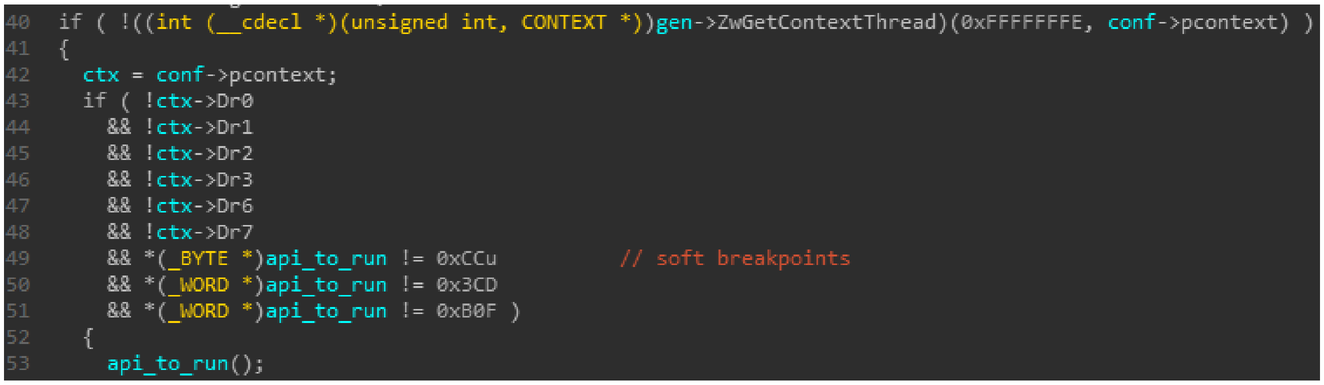
**Anti Attach**: In order to prevent a debugger from attaching to the process, the malware's authors hook DbgBreakPoint and DbgUiRemoteBreakin. Attackers usually hook those functions with a jump to the "ExitProcess" function. In this case though, it is just nop's or it jumps to an invalid address to crash the program.



```
*DbgBreakPoint = 0x90u;                         // nop
*(_BYTE *)DbgUiRemoteBreakin = 0x6A;            // push 0
                                                // mov eax, FFFF
                                                // call eax
                                                // ret 4
*(_BYTE *)(DbgUiRemoteBreakin + 1) = 0;
*(_BYTE *)(DbgUiRemoteBreakin + 2) = 0xB8u;
*(_DWORD *)(DbgUiRemoteBreakin + 3) = *(_DWORD *)(a2 + 0x9C);
*(_BYTE *)(DbgUiRemoteBreakin + 7) = 0xFFu;
*(_BYTE *)(DbgUiRemoteBreakin + 8) = 0xD0u;
*(_BYTE *)(DbgUiRemoteBreakin + 9) = 0xC2u;
*(_BYTE *)(DbgUiRemoteBreakin + 10) = 4;
*(_BYTE *)(DbgUiRemoteBreakin + 11) = 0;
```

*Figure 3: Ntdll function hook.*

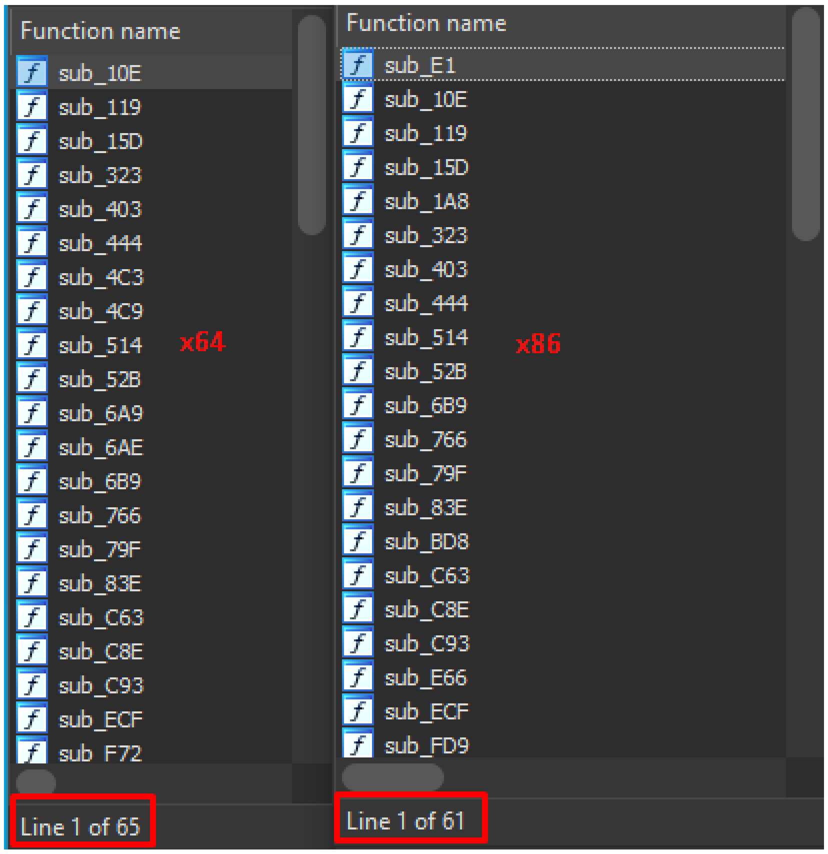
The shellcode wraps several API calls that researchers put breakpoints on with a function that detects if soft breakpoints or hardware breakpoints have been set. It does that by checking if the DR* registers are set to a value other than 0, and checks whether bytes of the called API is 0xCC, 0x3CD or 0xB0F. If it detects hardware breakpoints or soft breakpoints it will jump to an invalid address to crash the program.



```
if ( !((int (__cdecl *)(unsigned int, CONTEXT *))gen->ZwGetContextThread)(0xFFFFFFFE, conf->pcontext) )
{
  ctx = conf->pcontext;
  if ( !ctx->Dr0
    && !ctx->Dr1
    && !ctx->Dr2
    && !ctx->Dr3
    && !ctx->Dr6
    && !ctx->Dr7
    && *(_BYTE *)api_to_run != 0xCCu          // soft breakpoints
    && *(_WORD *)api_to_run != 0x3CD
    && *(_WORD *)api_to_run != 0xB0F )
  {
    api_to_run();
```

*Figure 4: Check hardware breakpoints and soft breakpoints before running an API call.*

- Calls the NtSetInformationThread with the ThreadHideFromDebugger (0x11) information class.
- **Anti static and dynamic analysis** -- Mixed x86\x64 code. The same shellcode can be read as x64 bit and x86 bit code. Though the shellcode only works as x86 bit, this is used to hamper static and dynamic analysis by pushing junk instruction that doesn't affect the original flow of the shellcode.

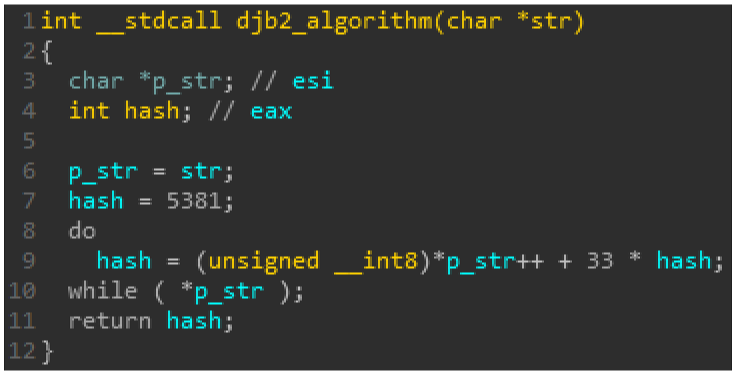

Function name
sub_10E
sub_119
sub_15D
sub_323
sub_403
sub_444
sub_4C3
sub_4C9
sub_514
sub_52B
sub_6A9
sub_6AE
sub_6B9
sub_766
sub_79F
sub_83E
sub_C63
sub_C8E
sub_C93
sub_ECF
sub_F72
x64
Line 1 of 65

Function name
sub_E1
sub_10E
sub_119
sub_15D
sub_1A8
sub_323
sub_403
sub_444
sub_514
sub_52B
sub_6B9
sub_766
sub_79F
sub_83E
sub_BD8
sub_C63
sub_C8E
sub_C93
sub_E66
sub_ECF
sub_FD9
x86
Line 1 of 61

*Figure 5: Same shellcode read as x64 bit and x86 bit.*

## GuLoader Functionality

Guloader uses the djb2 hash function in order to resolve API calls by hash. It does that to resolve GetProcAddress. Then, it will be using the same djb2 hash function to retrieve other API calls.

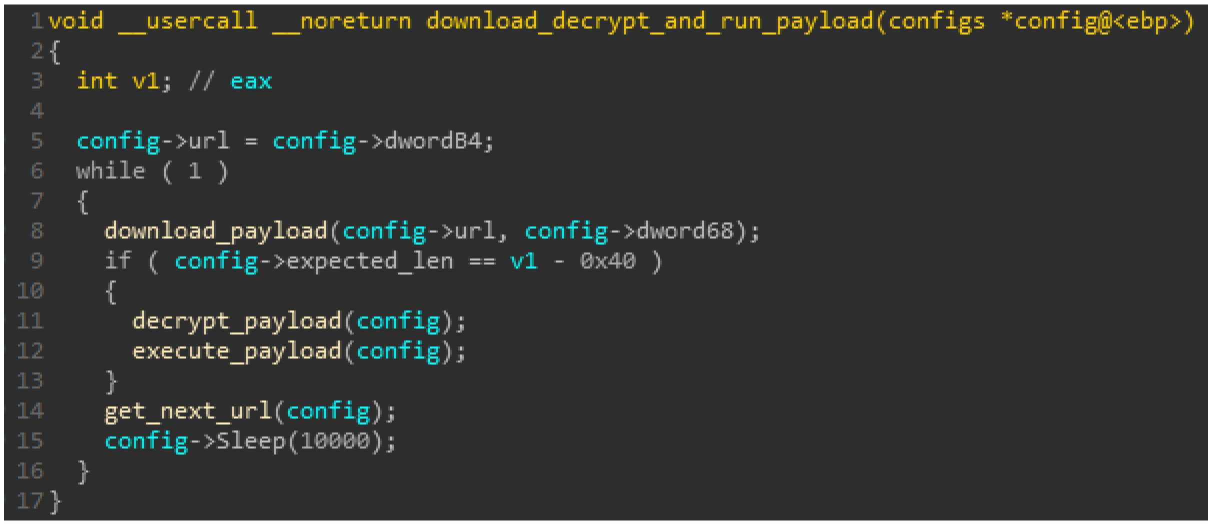


```
int __stdcall djb2_algorithm(char *str)
{
  char *p_str; // esi
  int hash; // eax

  p_str = str;
  hash = 5381;
  do
    hash = (unsigned __int8)*p_str++ + 33 * hash;
  while ( *p_str );
  return hash;
}
```

*Figure 6: djb2 hash algorithm.*

GuLoader uses a variation of a process hollowing injection in order to run its next stage. It spawns a new suspended process of itself (CreateProcessInternalW), and will then map “msvbvm60.dll” over the child process at 0x400000. Other variations of *Guloader malware* creates “RegAsm.exe” as a child process and map “mstsc.exe.” Next, it will inject the shellcode into the child process, changing the entry point to the shellcode’s entry point. The process is then resumed and the child process downloads the encrypted payload and overwrites the image at 0x400000 with the decrypted one. The decrypted payload, in our example, is Remcos RAT.



```
void __usercall __noreturn download_decrypt_and_run_payload(configs *config@<ebp>)
{
  int v1; // eax

  config->url = config->dwordB4;
  while ( 1 )
  {
    download_payload(config->url, config->dword68);
    if ( config->expected_len == v1 - 0x40 )
    {
      decrypt_payload(config);
      execute_payload(config);
    }
    get_next_url(config);
    config->Sleep(10000);
  }
}
```

*Figure 7: Function that downloads, decrypts, and runs the payload.*

The loader then iterates through the URLs; there are two URLs in this sample but, at the time, one of them wasn't active. Then, it will download the final payload in its encrypted form and will decrypt it using the XOR key (0x239 length in this sample), which is stored within the shellcode.

Next, the loader will overwrite the content loaded at 0x400000 with the decrypted code and jump to the entry point.

## Conclusion

GuLoader is appearing more frequently as a malware loader in the wild. It's one of the most advanced downloaders currently in use, and often downloads its payload from cloud hosting platforms. This campaign was a regular URL, however.

Morphisec blocks GuLoader execution, protecting its customers from this and other evasive malware to ensure that their critical infrastructure is secure. This protection extends to remote employees as well, and we encourage everyone to take advantage of our free trial to see how we protect critical systems against cyberattack.

IOCs:

Samples:

- 907b9784966bdbfc5447943747d3aed7445727d0
- 2d6720ae866875b42601951eb3c9421fa2d5b7f0
- fa601abbdeb159eccbcee527ebc6019619a9e442
- 12dd8f1cf43b51ec779363de98636c6e35fc1b4b

- 117d9701b22abb5530de2063698bcf75f2a90577

URLs:

- hxxp://arabianbrother[.]com/a/6.bin
- hxxp://arabianbrother[.]com/a/3.bin
- hxxp://ntaryan[.]com/a/6.bin
- hxxp://ntaryan[.]com/a/3.bin
- hxxps://drive.google[.]com/uc?export=download&id=12dw-X7CxidMNeST2IlWkZAaJAha5TUFG
- hxxps://drive.google[.]com/uc?export=download&id=1xz02BCj0obD4UPgs0CMtu_6GXxCEYXz
- hxxps://drive.google[.]com/uc?export=download&id=1xm_RKeKAUaH1QnWB_RZw4nMtdq7jK_PX