OilRig Uses Updated BONDUPDATER to Target Middle Eastern Government

23,860
people reacted

1

9 min. read

SHARE

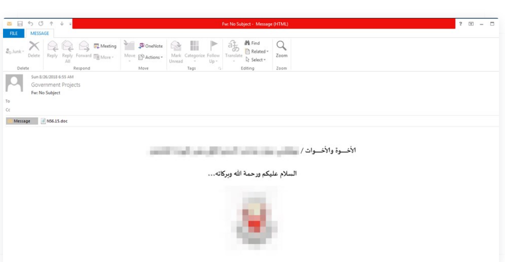
By Kyle Wilhoit and Robert Falcone
September 12, 2018 at 4:00 PM
Category: Unit 42
Tags: BONDUPDATER, OilRig

The OilRig group has been active since at least mid-2016, and continues their attack campaigns throughout the Middle East, targeting both governmental agencies and businesses on an almost routine basis. Often preferring homegrown tools and malware, OilRig continually modifies their malware and tools to accomplish their objectives. In August 2018, Unit 42 observed OilRig targeting a government organization using spear-phishing emails to deliver an updated version of a Trojan known as BONDUPDATER. BONDUPDATER is a PowerShell-based Trojan first discovered by FireEye in mid-November 2017, when OilRig targeted a different Middle Eastern governmental organization.

The BONDUPDATER Trojan contains basic backdoor functionality, allowing threat actors to upload and download files, as well as the ability to execute commands. BONDUPDATER, like other OilRig tools, uses DNS tunneling to communicate with its C2 server. During the past month, Unit 42 observed several attacks against a Middle Eastern government leveraging an updated version of the BONDUPDATER malware, which now includes the ability to use TXT records within its DNS tunneling protocol for its C2 communications.

A New Series of Attacks

In mid-August, the Oilrig threat group sent what appeared to be a highly targeted phishing email to a high-ranking office in a Middle Eastern nation. The email had no subject and what initially drew our attention to this attack was the content of the spear phishing email.



Sun 8/26/2018 6:55 AM
Government Projects
Fw: No Subject
To
Cc
Message
N56.15.doc

[redacted] / الأخـــوة والأخـــوات

السلام عليكم ورحمة الله وبركاته...

Figure 1. Spear phishing email sent by the Oilrig threat group

Attached to the email was a malicious document named “N56.15.doc” (SHA256: 7cbad6b3f505a199d6766a86b41ed23786bbb99dab9cae6c18936afdc2512f00) which contained a macro that attempted to install a new version of the BONDUPDATER Trojan.

SECURITY WARNING Macros have been disabled. Enable Content

[redacted] / الأخـــوة والأخـــوات

السلام عليكم ورحمة الله وبركاته...

Figure 2. Microsoft Word document with embedded macros and PowerShell

Updated BONDUPDATER

The spear-phishing email had an attached Microsoft Word document that contained a macro responsible for installing a new variant of BONDUPDATER. The macro begins this installation process by creating two files on the system at the following location:

```
C:\ProgramData\WindowsAppPool\AppPool.vbs (SHA256:
c0018a2e36c7ef8aa15b81001a19c4127ad7cd21ae410c1f854e5dadfa98b322)
```

```
C:\ProgramData\WindowsAppPool\AppPool.ps1 (SHA256:
d5c1822a36f2e7107d0d4c005c26978d00bcb34a587bd9ccf11ae7761ec73fb7)
```

The macro finishes by running the dropped VBScript “AppPool.vbs” file by running “wscript C:\ProgramData\WindowsAppPool\AppPool.vbs”. When first executed, the “AppPool.vbs” file will create the following scheduled task to execute every minute, which provides BONDUPDATER persistence and the ability to continually run on the system as the Trojan does not have a main loop to carry out its functionality:

```
cmd.exe /C schtasks /create /F /sc minute /mo 1 /tn “\WindowsAppPool\AppPool” /tr
“wscript /b “C:\ProgramData\WindowsAppPool\AppPool.vbs””
```

After creating the scheduled task, the VBScript will execute the “AppPool.ps1” script dropped by the macro using the following command line:

```
PowerShell.exe -exec bypass -file C:\ProgramData\WindowsAppPool\AppPool.ps1
```

Subsequent executions of “AppPool.vbs” will check for the existence of a file named “quid”, which, if present, will also use the above PowerShell script to run “AppPool.ps1”. The PowerShell script creates the “quid” file upon its first execution to avoid creating multiple scheduled tasks.

The “AppPool.ps1” file is a PowerShell script that is a variant of the BONDUPDATER payload. BONDUPDATER, like other OilRig payloads, uses DNS tunneling to communicate with its C2 server. This variant of the BONDUPDATER Trojan has a new lock file, which involves creating a lock file at the following location which will write the current PowerShell process identifier (PID) to this file:

```
C:\ProgramData\WindowsAppPool\lock
```

The purpose of this lock file is to only have one instance of the Trojan running at one time. However, it also uses this lock file to determine how long the PowerShell process has been executing, as it checks the creation time of the file lock against the current time to determine if the PowerShell process has been running for more than ten minutes. If it has, the script will stop the process based on the PID in the lock file and delete the lock file. Future executions of the PowerShell script will fully execute as the lock file will no longer exist on the system. This suggests the threat actors may have experienced issues with this Trojan running for extended periods in the past, likely related to the communication loops that we will discuss later.

This BONDUPDATER variant also creates the following file that it will use to store a unique identifier for the system, which is generated by obtaining a random number between 10 and 99 and appending the first 8 characters of a generated GUID:

```
C:\ProgramData\WindowsAppPool\quid
```

The BONDUPDATER Trojan then creates several folders that it will use to store files it receives from the C2 server and files that it generates or gathers to send to the C2 server. The folder names are:

```
 C:\ProgramData\WindowsAppPool\files
```

```
C:\ProgramData\WindowsAppPool\<unique ID from quid file>\receivebox
```

```
C:\ProgramData\WindowsAppPool\<unique ID from quid file>\sendbox
```

```
C:\ProgramData\WindowsAppPool\<unique ID from quid file>\done
```

Even though the script creates the “files” folder, it does not appear that the Trojan uses these folders within the code. The Trojan uses the “receivebox” folder to store files obtained from the C2 server, while the “sendbox” folder is used to store files that the Trojan will upload to the server. The Trojan uses the filenames of the file received from the C2 server stored in the “receivebox” folder to determine how to handle the file, which is explained below.

The BONDUPDATER sample retains its original command handling and C2 communication functionality. This process involves communicating with the C2 server to receive a file and using a character in the filename as the command. The Trojan’s command handler checks the trailing character of the filename to process the file contents, which can be seen in Table 1.

Trailing Character/Command	Purpose	Description
0	Execute command	Reads the contents of the file and runs them as a command with “cmd.exe”. The output of the command is saved to a file whose name starts with “proc” and is stored in the “sendbox” folder, which the Trojan will send to the C2 server.
1	Download file	Reads the contents of the file for a path to a file to download. Copies the specified file to a file in the “sendbox” folder for the Trojan to send to the C2 server.
Any other character	Upload file	Used to store a file on the system. The file is moved to the “done” folder, which stores the file for future use. The Trojan writes “200<>[path to stored file]” to a file in the “sendbox” folder to notify the C2 that the file was downloaded successfully.

Table 1 Commands available in BONDUPDATER and their purpose

After handling the command, BONDUPDATER will send files it saved in the “sendbox” folder to the C2 server, after which it terminates and relies on the scheduled task to run again in the future.

As discussed above, the BONDUPDATER Trojan uses a DNS tunneling protocol to receive files from the C2 server for processing. This particular BONDUPDATER sample includes two different variations of the DNS tunneling protocol, one using DNS A records, and one using DNS TXT records to transmit data from the C2 to the Trojan. The use of TXT records for C2 communications appears to be a new feature to the BONDUPDATER Trojan.

The DNS tunneling protocol uses carefully crafted subdomains sent to the authoritative name server of the C2 domain, which in this specific sample was the domain “withyourface[.]com”. The Trojan generates subdomains differently when it sends data to the C2 compared to when it is looking to receive data from the C2, regardless of which DNS tunneling protocol is used for communications. The format of the generated domains for both sending and receiving starts with the previously generated GUID created to uniquely identify the system. However, the Trojan inserts a part number value and an action type character into this GUID string at random offsets. The part number value is a three-digit string that corresponds to the chunk of data the Trojan is attempting to transmit. The action type is a single character that notifies the C2 the type of communication the Trojan is carrying out. The two static characters “C” and “T” in the subdomain surround two digits, which help the C2 server find the part number and action type mixed in within the GUID string at random offsets.

Sending data format

```
<GUID with part number and action character><sequence number><between 1 and 7
random characters>C<index of part number><index of action>T.<data chunk>.
<filename>.<c2 domain>
```

Receiving data format

```
<GUID with part number and action character><sequence number><between 1 and 7
random characters>C<index of part number><index of action>T.<c2 domain>
```

Depending on whether the C2 communications use DNS A or TXT records, different action types are used when generating the subdomains to notify the C2 what format to use to respond. Table 2 shows action types used in outbound requests and the purpose of the request.

Action Type	A/TXT Communications	Description
M	A/TXT	Initial beacon
0	A	Provide filename
1	A	Provide data
W	TXT	Provide filename
D	TXT	Provide data
P	TXT	TXT method failed, notify C2 to switch to A records

Table 2 Action types used within the DNS tunneling protocol and their purpose

For example, the Trojan will begin communicating with its C2 server by sending an initial beacon with the action character “M”. This initial beacon will follow the receiving format, as seen in the following example:

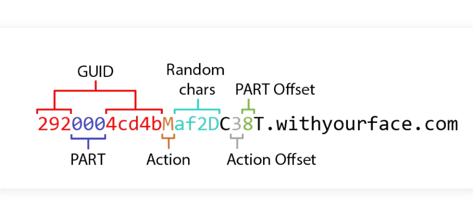


Figure 3. Example domain for the C2 beacon with its format explained

The offset to the part number and action type in the GUID above may not seem correct at first, as the action type does not appear to be at offset 8 in the string. However, this is correct, as the action type is inserted at offset 8 before the part number was inserted at offset 3. Therefore, we believe the C2 server just adds the two offsets together to find the actual location of the action type in the string.

Original Data Transfer using DNS Tunneling

The original data transfer process within BONDUPDATER looked for specific A records within answers to DNS queries, shown in Table 3. The Trojan processed the A records in the C2 response to obtain a filename, which specifically looks for an IPv4 address within the A record with “24.125” in the first two octets. The Trojan appends the remaining two octets of this A record to the string “rcvd” and uses this as the filename to save future data to in the “receivebox” folder. This IP address also instructs the Trojan to treat following DNS A records as data. The following DNS A records are split on the “.” and each of the first three octets are treated as data, whereas the fourth octet is used as a counter to obtain the correct chunk of data. Lastly, the Trojan looks for the A record “1.2.3” as a signal to write the provided data to the specified file, which is then subjected to the previously mentioned file-based command handler.

IP Address	Description
24.125.\d.\d	Sets the filename “rcvd\d\d” in the “receivebox” to store data for processing
\d.\d.\d.\d	First three \d are treated as data and the fourth is used to keep track of the sequence
1.2.3.\d	Instructs the Trojan to write the data to the file and begin processing it for commands.

Table 3 IP Addresses and their meanings within the original data transfer process in BONDUPDATER

New Data Transfer using DNS Tunneling

This BONDUPDATER sample has a new method to obtain files from the C2 server using a series of DNS TXT queries. This method follows a similar process as the original method but uses DNS TXT results to obtain a filename and the data to write to the file. Once data is written to the file system, this method uses the same command handler as the original method to process the contents of the file based on the trailing character of the filename, as seen in Table 1.

The C2 can initiate the new command handling functionality by responding to the initial beacon with a DNS A record of “99.250.250.199”. The script will enter a loop attempting to communicate with its C2 every 50 milliseconds, looking for a series of responses with specific characters that the script will use as instructions to determine how to handle the result of the TXT record, which can be seen in Table 4. The Trojan will split each TXT record from the C2 response on the character “>”, with the data to the left of the “>” character used as the instruction and the data to the right as the data.

Instruction	Description
N	Idle. Set action type of next query to “W”
S	Receive data from C2. Decode data portion as base64. Sets the action type of future queries to the C2 to “D”.
S000s	Use data to as a portion of the filename to save data to. The data is appended to the string “rcvd”, which will be saved in the “receivebox” folder. Sets the action type of future queries to the C2 to “D”.
E	Write bytes provided by the “S” command to the file resulting from the “S000s” command. The breaks the loop for the script to process the downloaded file.
C	Cancel communications by exiting the loop.

Table 4 Instructions within the new data transfer process in BONDUPDATER and their meanings

While we have not seen the C2 use this TXT record-based communications, we believe the process would involve the C2 providing a TXT record with the “S000s” instruction to set the filename to which data is saved. The C2 would then provide a series of “S” instructions to provide data, followed by the “E” instruction to write that data to the file, which would then be subjected to the command handler.

Conclusion

As expected, OilRig is continuing their onslaught of attacks well into 2018 with continued targeting in the Middle East. Sometimes developing new tools, OilRig also often uses what has worked in the past, including developing variants of previously used tools and malware. This reduces development time and capitalizes on previous versions of the tool and its success.

Oilrig is a highly diverse and very resourceful threat actor, employing a litany of methods and tools to compromise victims, but Palo Alto Networks customers are protected from this OilRig attack and BONDUPDATER by:

- AutoFocus customers can track this Trojan with the Bondupdater_Docs tag
- All known BONDUPDATER document samples are marked with malicious verdicts in WildFire
- All known BONDUPDATER document C2 domains have DNS signatures and are classified as Command and Control

Indicators

BONDUPDATER C2

withyourface[.]com

BONDUPDATER Dropper Docs

7cbad6b3f505a199d6766a86b41ed23786bbb99dab9cae6c18936afdc2512f00

c0018a2e36c7ef8aa15b81001a19c4127ad7cd21ae410c1f854e5dadfa98b322

d5c1822a36f2e7107d0d4c005c26978d00bcb34a587bd9ccf11ae7761ec73fb7

Get updates from Palo Alto Networks!

Sign up to receive the latest news, cyber threat intelligence and research from us

Email address

Subscribe

☐ I'm not a robot reCAPTCHA Privacy - Terms

By submitting this form, you agree to our Terms of Use and acknowledge our Privacy Statement.

© 2020 Palo Alto Networks, Inc. All rights reserved.