The OilRig Campaign: Attacks on Saudi Arabian Organizations Deliver Helminth Backdoor

By Robert Falcone, Bryan Lee

Published: 2016-05-26 · Archived: 2026-04-05 14:10:44 UTC

In May 2016, Unit 42 observed targeted attacks primarily focused on financial institutions and technology organizations within Saudi Arabia. Artifacts identified within the malware samples related to these attacks also suggest the targeting of the defense industry in Saudi Arabia, which appears to be related to an earlier wave of attacks carried out in the fall of 2015. We have grouped these two waves of attacks into a campaign we have named ‘OilRig’.

In recent OilRig attacks, the threat actors purport to be legitimate service providers offering service and technical troubleshooting as a social engineering theme in their spear-phishing attacks. Earlier OilRig attacks appear to use fake job offers as a social engineering theme. The campaign appears highly targeted and delivers a backdoor we have called ‘Helminth’. Over the course of the attack campaign, we have observed two different variations of the Helminth backdoor, one written in VBScript and PowerShell that was delivered via a macro within Excel spreadsheets and the other a standalone Windows executable.

Clayslide: Excel Macros Install Helminth Script

In May 2016, Unit 42 began researching attacks that used spear-phishing emails with attachments, specifically malicious Excel spreadsheets sent to financial organizations within Saudi Arabia. We observed spear-phishing emails sent between May 4 and May 12 of this year that delivered these malicious Excel spreadsheets, which we are tracking as ‘Clayslide’. ClaySlide documents contain malicious macros that display decoy content within the spreadsheet and installs a variant of a Helminth backdoor. FireEye also reported on these attacks in a May 22 blog post.

The macro within Clayslide samples installs the Helminth script, which is composed of a VBScript called ‘update.vbs’ and a PowersShell script called ‘dns.ps1’. The purpose of the VBScript is to send network beacons to its command and control server using HTTP requests and will either download a file or run a batch script provided within the HTTP response. The VBScript also uploads the output of the provided batch scripts to the command and control (C2) server, which provides threat actors a functional remote shell to the system.

The PowerShell script has similar capabilities to the VBScript, but instead of using HTTP for communications it uses a series of DNS queries to send and receive data from the server. This communication channel relies on the C2 server responding to DNS queries with IP addresses that the PowerShell script will parse treat as data to construct a batch script to execute on the system. The script specifically looks for the IP address “33.33.x.x” to mark the beginning of the batch script transfer. The script will continue sending additional DNS requests and use the octets of the resolving IP addresses as characters to write to the batch script. The script continues to write data

to the batch script until it receives the IP address “35.35.35.35”, which notifies the script to stop saving data to the file and to run the batch script.

Please reference the Appendix for more detailed information on the Clayslide delivery documents and the Helminth script variant.

Discovery of Executable Helminth Variant

Additional samples were discovered in WildFire exhibiting the same DNS-based C2 behavior as the script variant of Helminth; however, many of these samples were found to be Windows executable, instead of the previously observed VBScript and PowerShell combination. These samples were found to contain the same functionality as the previously mentioned Helminth samples. Figure 1 shows the code within the VBScript version of Helminth checking resolving IP addresses for the 35.35.35.35 IP address to stop appending data to a batch script before executing it, while Figure 2 shows the same functionality within the executable version of the Trojan.

```
elseif ($mydata.Equals('35.35.35.35'))
{
    $global:myflag = 0
}
```

Figure 1 Helminth dns.ps1 PowerShell script looking for 35.35.35.35 IP address

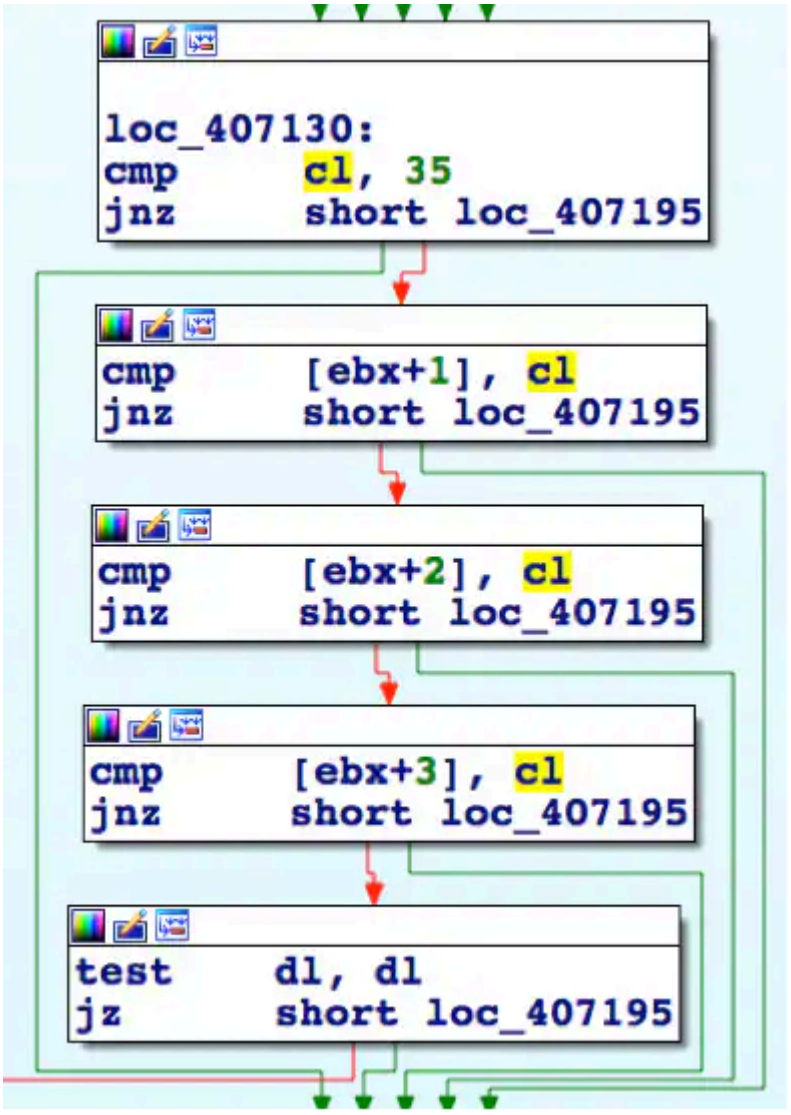


Figure 2 Helminth executable looking for the 35.35.35.35 IP address

This suggests that the threat actors developed the executable variant of Helminth as a standalone option whose installation does not rely on a macro within an Excel spreadsheet. This also suggests that the threat actors purposely used the same communication methods across both variants with the intention to use the same command and control server application. This variant of the Trojan is also where we obtained its name, as several of these payloads had the following debug symbol path that suggests the malware author called this project 'Helminth':

E:\Projects\hlm updated\Helminth\Release\Helminth.pdb

Please reference the Appendix for additional details on the Helminth executable variant.

Delivery of Windows Executable Helminth Variant

Unit 42 does not have detailed targeting information associated with attacks delivering the executable variant of the Helminth Trojan, however, we found a Zip archive created in August 2015 that may have been used by the threat actors to deliver the Helminth Trojan. This Zip file was hosted at the following location:

hxxp://minfosecu.doosan[.]com/data/joboffer.zip

The Zip archive is encrypted with an unknown password, but we know it contains two files named joboffer.chm and thumb.db. The thumb.db file in the archive has the same name and file size (368128 bytes) as a dropper Trojan we track as 'HerHer' (SHA256: fb424443ad3e27ef535574cf7e67fbf9054949c48ec19be0b9ddfbfc733f9b07) that installs a known Helminth executable sample. The joboffer.chm file is a compiled HTML file that we believe loads and executes the 'thumb.db' file as a payload, but we cannot be absolutely sure as we do not have the password required to extract the files from the archive.

The decoy opened by the Helminth sample installed by 'thumb.db' (seen in Figure 3) is a dialog box associated with HTML help, which further strengthens our theory that the joboffer.chm ran the sample. This decoy suggests that the threat actors wanted to open the HTML help dialog after installing the Helminth Trojan, as the joboffer.chm file is effectively a standalone HTML file. We believe that the threat actors employed social engineering to underplay the situation and provide a different legitimate job offer if the victim responded with concerns of malicious activity.

Figure 3 A Helminth sample displays this dialog box if provided 'w' on the command line

The executable variant of Helminth is installed with a dropper Trojan that we are tracking as the HerHer Trojan. This Trojan has two objectives: installing embedded Trojans and displaying either a fake error prompt or a fake "trubleshooting" (the malware author misspelled this word in each sample) utility. Figure 4 is an example of the fake error prompt displayed by the HerHer Trojan.

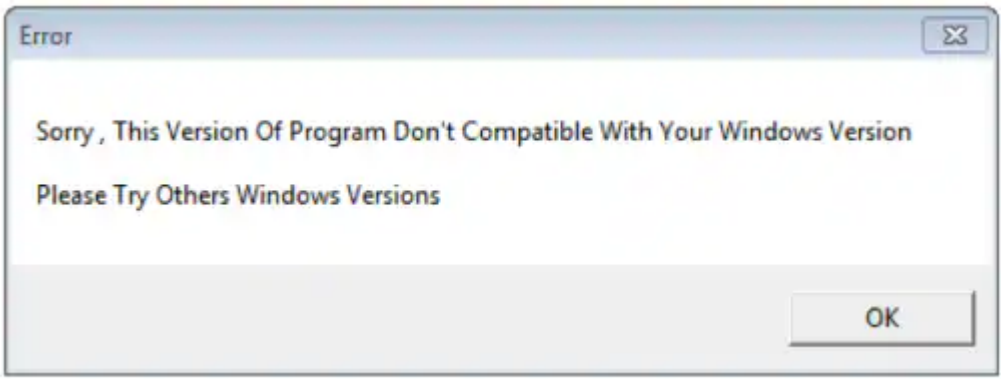


Figure 4 Fake Error Prompt Displayed by the HerHer Trojan

The Helminth executable variant is very similar in functionality to its script-based counterpart, as it also communicates with its C2 server using both HTTP and DNS queries. The major difference in capabilities between the two variants is that the executable version comes with a module that Helminth uses to log keystrokes and the clipboard contents to exfiltrate to the C2 server.

Helminth executable samples send artifacts within network beacons to its C2 server that the Trojan refers to as a 'Group' and 'Name'. We extracted the group and name values from the Helminth executable samples to determine their purpose. It appears that the group values hardcoded into the malware is associated with the targeted organization, as several are Saudi Arabian organizations within the telecommunications and defense industries. This suggests that the threat actors are not only focused on financial organizations, as their target set could include other industries as well.

The name values hardcoded into the Helminth samples are also interesting, as a majority of the names are related to famous philosophers, such as 'Plato' (Greek philosopher), 'Arasto' (Persian and Urdu for Greek philosopher Aristotle), and 'ALAfghani' (Jamal ad-Din al-Afghani, Islamic Philosopher). Other name values embedded in samples contain other Persian words, such as 'Nafti' (نفتی) that translates to 'oily', which led us to name this campaign OilRig).

Helminth Infrastructure

Examining the known infrastructure of the collected sample set of Helminth provides several interesting findings in regards to the adversary's tactics. The variants leveraging malicious macros embedded in Excel documents all share the same command and control server of go0gie[.]com. The executable variants, on the other hand, used a variety of domains:

checkgoogle[.]org
mydomain1110[.]com
kernel[.]ws
mydomain1607[.]com
mydomain1609[.]com

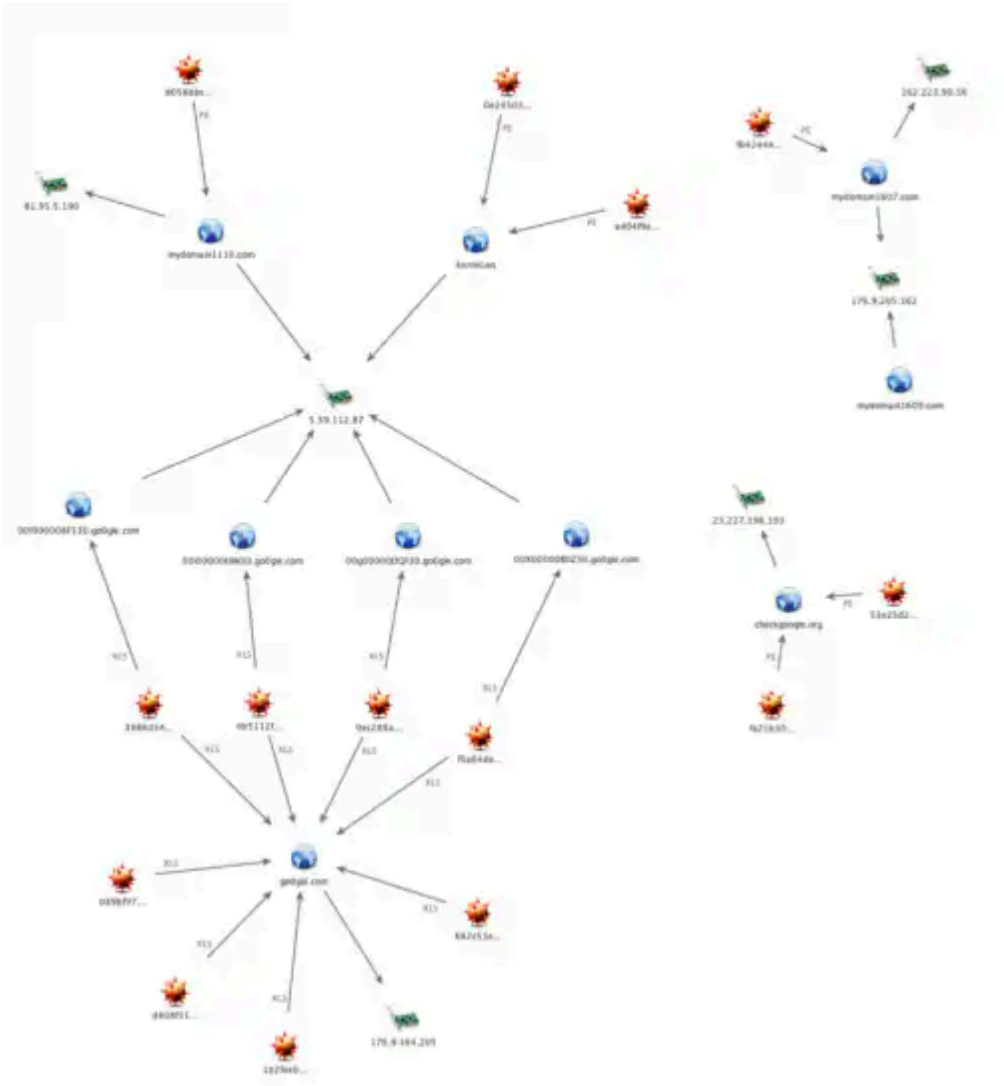


Figure 5 Helminth C2 Infrastructure

Each sample of the weaponized Excel document variant used a unique command and control domain to retrieve a bot ID, using the following format:

00000000<base 36 of a random number smaller than 46655>30.go0gie[.]com

Each of these domains, however, resolved to the same IP address of 5.39.112.87. This IP is observed as the resolution for two domains in use by the portable executable variants, kernel[.]ws and mydomain1110[.]com. Judging by compile timestamps of the executables and last saved timestamps of the weaponized documents, it is likely the adversary is recycling a previously created C2 server at 5.39.112.87 for the newer macro based variant. The other C2 domains and IPs observed in use by the previous portable executable samples did not have shared infrastructure with the newer macro variants, although there is tactical overlap via the naming scheme of the domains.

Historical WHOIS data reveals additional findings, potentially alluding to an Iranian-based operator. From a timeline perspective, a new domain was registered almost in consecutive months, beginning in July 2015. Each of the domains's WHOIS data contained registrant information that was either reused, or was closely related to

previously used information. For example, the domains mydomain1607[.]com and mydomain1609[.]com used the exact same registrant information. The email address edmundj@chmail[.]ir and the geolocation of Tehran, Iran, being of note. Kernel[.]ws and checkgoogle[.]org used very similar email addresses, andre_serkisian@yahoo[.]com and andre.serkisian@chmail[.]ir, respectively. The registrant information for kernel[.]ws also provided a geolocation of Tehran, IR and the email provider for the address used in checkgoogle[.]org was the same used for mydomain1607[.]com and mydomain1609[.]com, chmail.ir. The mydomain1110[.]com domain did not appear to reuse any of the previously observed WHOIS data artifacts, but did still give a geolocation of Tehran in addition to the use of an email address linked to other domains thematically similar to the know command and control domains and are potentially related.

Although there is heavy use of Iranian-based artifacts within the WHOIS registrant information, it is important to remember that this data is easily falsified. At face value, however, taking into account the registrant information and the use of Persian language in the samples are compelling indicators that the operators may indeed be based out of Iran.

Conclusion

While researching the OilRig campaign, we have seen two waves of targeted attacks on Saudi Arabian organizations in which a group of threat actors delivered the Helminth Trojan as a payload. The two waves of attacks used separate variants of the Helminth Trojan, specifically a script and executable variant of the Trojan.

The two variants of Helminth use almost identical command and control protocols, which allows the threat actors to maintain consistent infrastructure throughout the campaign to manage the compromised hosts, regardless of the Helminth variant used in the attack.

The two variants of Helminth do require different delivery methods, with the script variant relying on an Excel spreadsheet for delivery, while the executable variant is more traditional in the fact that it can be installed without a delivery document. We speculate that the executable variant involves threat actors socially engineering the victim into running the payload, rather than installing the payload as the result of successful exploitation of a vulnerability. The multiple delivery methods suggest this threat group is capable of adapting their procedures to suit the current operation in the overarching campaign.

Palo Alto Networks customers are protected from the Helminth Trojan and can gather additional information using the following tools:

- WildFire detection of all known samples as malicious
- All Helminth C2 domains have DNS signatures created and are identified as malicious in PAN-DB.
- AutoFocus tags Clayslide, Helminth and HerHerDropper.

Appendix

Clayslide Delivery Documents

At first, Clayslide spreadsheets display a worksheet called “Incompatible” that contains instructions for the user to manually enable macros (as seen in Figure 6), as macros are disabled in Excel by default. This is an attempt to trick the user into running the embedded macro to install the Trojan, which does not require any vulnerability

exploitation. Figure 6 shows the “Protected View” alert in Excel informing the user that there is an embedded macro that may cause harm to the system.

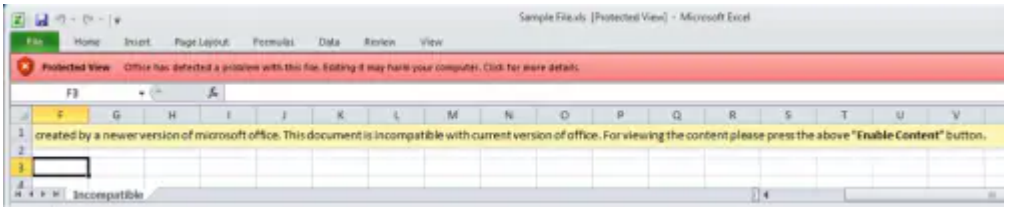


Figure 6 Clayslide spreadsheet showing the Incompatible worksheet with instructions to enable macros and Excel displaying its Protected View alert message

Before the user can enable the macros in accordance with the instructions displayed in the spreadsheet, the user must click the red bar displayed by Protected View and click the “Edit Anyway” button, as seen in 7.

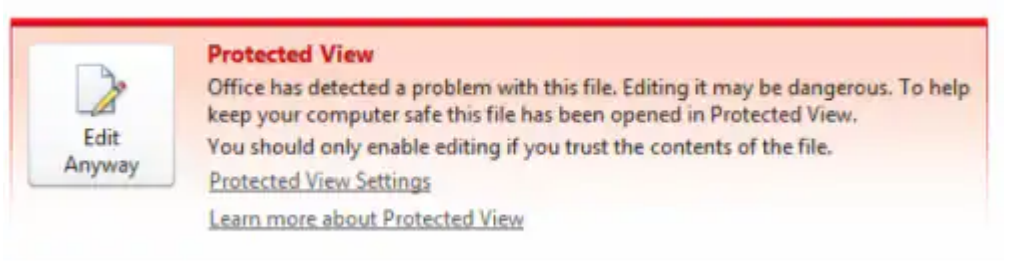


Figure 7 Protected View further mentioning the potential danger with editing the spreadsheet in the ClaySlide sample

After clicking the “Edit Anyway” button, Excel displays another security warning bar alerting that the spreadsheet contains macros, as seen in Figure 8. The “Enable Content” button mentioned within the instructions displayed within the Clayslide spreadsheet is now presented to the user.

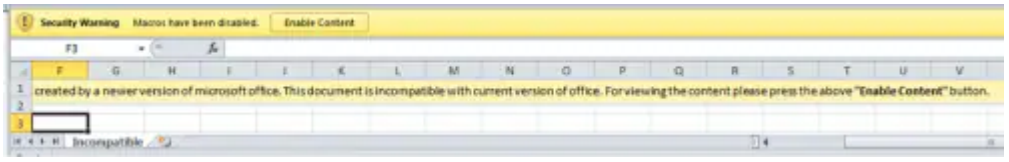
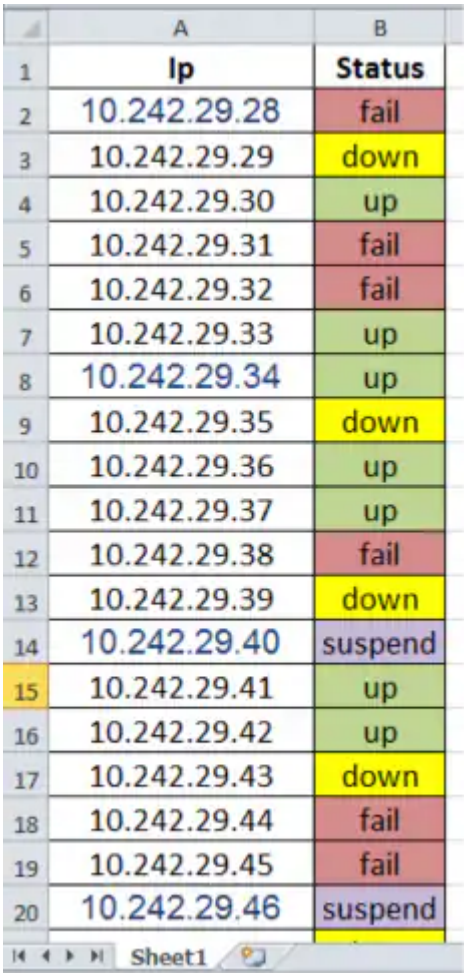


Figure 8 Excel security warning with the Enable Content button mentioned in Incompatible worksheet

If the user clicks the “Enable Content” button, the macro hides the “Incompatible” worksheet and makes hidden worksheets visible that displays decoy content to minimize the victim’s suspicions of malicious behavior taking place. Figure 9 below shows the decoy content displayed by macros within a Clayslide sample, specifically showing the status of internal network IP addresses that fit with the service provider social engineering theme used throughout the attack campaign. Figure 9 also shows that the “Incompatible” worksheet is no longer visible, as the decoy content is displayed in a worksheet called “Sheet1”.



	A	B
1	Ip	Status
2	10.242.29.28	fail
3	10.242.29.29	down
4	10.242.29.30	up
5	10.242.29.31	fail
6	10.242.29.32	fail
7	10.242.29.33	up
8	10.242.29.34	up
9	10.242.29.35	down
10	10.242.29.36	up
11	10.242.29.37	up
12	10.242.29.38	fail
13	10.242.29.39	down
14	10.242.29.40	suspend
15	10.242.29.41	up
16	10.242.29.42	up
17	10.242.29.43	down
18	10.242.29.44	fail
19	10.242.29.45	fail
20	10.242.29.46	suspend

Sheet1

Figure 9 Decoy content displayed after enabling macros within a Clayslide sample

After displaying the decoy content, the macro begins installing the script variant of the Helminth Trojan to the system. The process used by the macro to install this variant of Helminth begins with the creation of the following files and folders:

%PUBLIC%\Libraries\update.vbs
%PUBLIC%\Libraries\dns.ps1
%PUBLIC%\Libraries\up
%PUBLIC%\Libraries\dn
%PUBLIC%\Libraries\tp

The malicious macro finishes the installation process by creating a scheduled task that is responsible for running the two scripts at regular intervals, as the scripts themselves do not have the ability to continually run after the initial execution. The following code snippet within the macro creates a scheduled task named “GoogleUpdateTaskMachineUI” that will run the update.vbs script every three minutes:

```
wss.Run "schtasks /create /F /sc minute /mo 3 /tn " & Chr(34) & "GoogleUpdateTaskMachineUI" & Chr(34) & " /tr " & wss.ExpandEnvironmentStrings("%PUBLIC%") & "\Libraries\update.vbs", 0
```

Helminth Script Variant

The script variant of the Helminth Trojan consists of a VBScript and PowerShell script named update.vbs and dns.ps1. We aptly named this variant the script version, as we found another version of this Trojan that we will discuss later in this Appendix. The update.vbs script is responsible for reaching out to its command and control (C2) server using HTTP requests to the following two URLs:

hxxp://go0gIe.com/sysupdate.aspx?req=<random number>%5Cdwn&m=d
hxxp://go0gIe.com/sysupdate.aspx?req=<random number>%5Cbat&m=d

The C2 server will respond to the HTTP requests to the "bat&m=d" URL with a batch script that update.vbs will save to the "dn" folder and execute. The output of the downloaded batch script is saved to a text file in the "up" folder and uploaded to the C2 server via an HTTP POST request to the following URL:

hxxp://go0gIe.com/sysupdate.aspx?req=<random number>%5Cupl&m=u

Palo Alto Networks WildFire observed commands provided by the C2 server for the known Helminth samples. The commands, as seen below, show that the threat actors are attempting to do initial information gathering on the system, including available user accounts, username, computer name, running tasks, services, network services and if remote desktop is enabled.

whoami & hostname & ipconfig /all & net user /domain 2>&1 & net group /domain 2>&1 & net group "domain admins" /domain 2>&1 & net group "Exchange Trusted Subsystem" /domain 2>&1 & net accounts /domain 2>&1 & net user 2>&1 & net localgroup administrators 2>&1 & netstat -an 2>&1 & tasklist 2>&1 & sc query 2>&1 & systeminfo 2>&1 & reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1

The update.vbs concludes by running the dns.ps1 PowerShell script. The dns.ps1 script is also responsible for communicating with the C2 server, but it uses DNS queries to send data to the server. The DNS queries sent by this script are queries to subdomains on the same domain as the C2 server, which contains system information or the contents of files from the system. The subdomain of the DNS request that acts as the initial C2 beacon has the following structure:

00000000<base 36 of a random number smaller than 46655>30

The dns.ps1 script checks the response to this DNS query and uses the first octet of the resolving IP address as an identifier for the compromised system. The script then uses this identifier in a follow up DNS request to a subdomain with the following structure:

00<identifier>00000<base36 of a random number smaller than 46655>30

The C2 server will respond to these DNS queries with IP addresses that the script will parse and eventually treat as data to construct a batch script to execute on the system. The script specifically looks for the IP address "33.33.x.x" to mark the beginning of the batch script transfer. Upon receipt of this IP address, the script uses the last two octets of this IP address as a filename for the batch file that it saves to the "tp" folder that was initially created by the macro. Once the batch file name is obtained, the script will continue sending additional DNS requests and use the octets of the resolving IP addresses as characters to write to the batch script. The script

continues writing characters to the batch script until it receives the IP address “35.35.35.35” that notifies the script to stop saving data to the file and to run the batch script.

The output of the downloaded batch file is saved to “%PUBLIC%\Libraries\tp\<batch filename>.txt”. The script will then upload the output of this batch file by including the data in a sequence of DNS queries. The exfiltrates the output of the batch script by splitting up the data within the text file into chunks up to 23 bytes and sends the data within a series of DNS queries that have the following structure:

00<identifier><filename of batch file without its extension><base36 of sequence number><base36 of a random number smaller than 46655><up to 23 bytes of data from batch script output>

Both the update.vbs and dns.ps1 both provide a fully functional remote shell to the actors, which allow the actor to carry out any activities on the compromised system they wish.

Helminth Executable Variant

The executable variant of Helminth is installed with a Trojan that we are tracking as the HerHer Trojan. The HerHer Trojan saves several files to the file system upon execution to install the Helminth Trojan to the system.

- %APPDATA%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Certificate Managment.lnk
- %APPDATA%\Roaming\Microsoft\Windows\Start Menu\Programs\Certificate.ico
- %APPDATA%\Roaming\Microsoft Temperary\adbmanager.exe
- %APPDATA%\Roaming\Microsoft Temperary\adbtray.exe
- %APPDATA%\Local\Temp\acro\Users\config.txt
- %PUBLIC%\Libraries\~Windows\wintrust.hlm

The “Certificate Managment.lnk” shortcut uses the “Certificate.ico” file for its icon, as seen in Figure 10.

Figure 10 Icon file used

Additionally, it has a comment of ‘herher’, which is basis of the dropper’s name. Helminth relies on the following shortcut for persistence, as it runs the Trojan each time the system starts using the following command line:

"C:\Users\Rick James\AppData\Roaming\Microsoft Temperary\adbmanager.exe" q 1

The ‘adbmanager.exe’ and ‘adbtray.exe’ files are the actual Helminth Trojan, both of which are the same executable. The reason for two different filenames is currently unknown. The Helminth Trojan requires arguments

on the command-line to execute properly (‘q’ in the analyzed sample as seen in the ‘Certificate Managment.lnk’ shortcut), one of which will run the Trojan’s functional code and the other can open a dialog box as a decoy.

The Helminth Trojan begins by creating a mutex named ‘[username]ver4.1’ and writes its embedded configuration as ciphertext to the following file:

%APPDATA%\Local\Temp\acro\Users\config.txt

The Trojan will later decrypt the contents of this file using the RC4 algorithm, using the MD5 hash of ‘f246b23d-c2d6-45f2-b268-dec30d9adaad’ as the key. We decrypted the configuration file dropped by Helminth and found the structure of the configuration file is ‘IsAlive,[sleep interval]\r\n[C2 domain]’. For example, one Helminth sample had the following data within the “config.txt” file:

IsAlive,30
checkgoogle.org

The Helminth executable variant is able to run batch scripts provided by the C2 server, which is very similar to the script version of this Trojan. The executable variant has one additional capability that is not present in the script version, which involves the ability to log keystrokes via a supplemental keylogger module.

Helminth loads its keylogger module of the Trojan by loading the wintrust.hlm file dropped by the HerHer Trojan as a DLL and calling its exported function named ‘Initialize’. The keylogger that creates a window named ‘kk’ to monitor both the clipboard and keystrokes and to save the data in cleartext to the file ‘%TEMP%/acro/Users/[GUID from CoCreateGuid]kk.tmp’. The keylogger saves the keystrokes and the name of the Window visible while the keys were typed to this file in the following structure:

####T####[Window Name]####ET####
[logged keystrokes]

The wintrust.hlm keylogger logs the contents of the clipboard to the same file, but the clipboard contents do not follow a header that specifies the window name like the other logged keystrokes. The clipboard contents are logged to the file in the following format:

<<< Clipboard ---> [contents of clipboard]>>>

Helminth Exe C2 Communications

The Helminth executable is able to communicate with its C2 server via HTTP and via DNS queries in very similar ways to the Helminth script variant. In fact, the DNS beacons follow the same structure and sequence as the script variant of Helminth discussed in the previous section. The main difference between the beacons sent from the two variants of Helminth is the data included within the beacon, as the script variant does not send any system information within the beacons, whereas the executable version sends system and malware specific information within both the HTTP and DNS beacons.

Helminth executables include the system and malware information within HTTP beacons in the “Cookie” field of the request. Helminth structures the beacon data as follows:

Function=F1; ID=[MD5 of Computer and Username]; Group=[Hardcoded in Malware]; Name=[Hardcoded in Malware]; Service=0;

The Trojan will encrypt this data using RC4 and the MD5 hash of “f246b23d-c2d6-45f2-b268-dec30d9adaad” as the key and encode the encrypted data using base64. Figure 11 shows a Helminth HTTP beacon with the Cookie field containing the base64 data.

Figure 11 Helminth HTTP C2 beacon

Helminth sends data within DNS beacons differently than the HTTP beacons and includes additional information as well. The data within the DNS beacons follows the structure:

<path of folder containing keylogger module>
<path of folder containing key logs and batch script output files>
<group name hardcoded in malware>
<name hardcoded in malware>
<computer name>
<user name>
<sleep interval>
<C2 domain>

The Trojan does not encrypt the data sent via DNS beacons, rather it converts the ASCII characters into their hexadecimal values and includes these values in cleartext. The DNS beacons sent from the Helminth executable have the following structure, which is very similar to the script version:

00<identifier>01<sequence number><up to 24 hexadecimal values of the ASCII data>

Figure 12 shows an example of the DNS beacons sent from a Helminth executable.

Figure 12 Helminth DNS C2 beacon

The encoded data within the DNS beacons displayed in Figure 12 decode to the following:

C:\Users\Public\Libraries\~Windows\
C:\Users\Administrator\AppData\Roaming\Microsoft\Internet Explorer\Users\
[redacted company name]
Plato
[Computer Name redacted]
Administrator

30
kernel.ws
