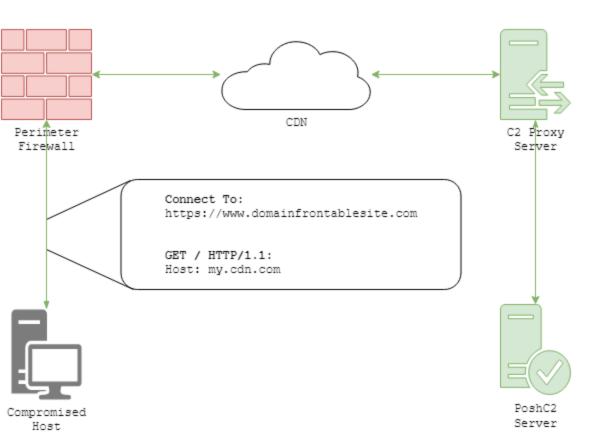
Detecting PoshC2 – Indicators of Compromise

labs.nettitude.com/blog/detecting-poshc2-indicators-of-compromise/

Rob Bone



As a counterpart to the <u>release of PoshC2 version 6.0</u> we are providing a list of some of its Indicators of Compromise (IoCs), particularly as used out-of-the-box, as well as some other effective methods for detecting it in your environment.

We also introduce the new PoshC2 Detections GitHub repository at <u>https://github.com/nettitude/PoshC2_IOCs</u> that will be continually updated as development continues, in order to assist blue teams with detecting PoshC2, particularly when used by less sophisticated attackers that do not alter or configure it to change the default IoCs. We encourage the community to contribute to and help update and improve this repository.

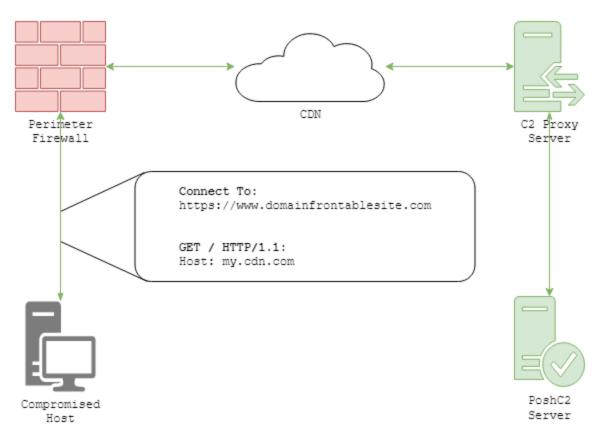
It is worth noting that PoshC2 is open source, so while these are IoCs for PoshC2 if used it its default state, ultimately it can be altered either through configuration or by changing the source code itself. The default configuration is subject to change, however where possible the location of that value is pointed out to the reader in order to allow these values to be monitored and updated, in addition to providing the GitHub repository.

Communications

June 17, 2020

One way to detect PoshC2 is to focus on the communications. Compromised hosts have to communicate with the C2 server somehow in order to pick up tasks and return task output. This is unavoidable and PoshC2 can currently only do this through the use of HTTP(S) requests.

That isn't to say this this makes it easy to detect; a huge amount of HTTP traffic is present in most environments and the flexibility of the protocol allows for traffic to be hidden and routed through legitimate websites using techniques such as Domain Fronting and reverse proxies.



An example of HTTP comms when domain fronting.

Something very helpful for catching C2 communications is SSL Inspection. By being able to inspect SSL traffic leaving the perimeter, the contents of that traffic can be checked and statistics acquired for detecting C2 communications. Without this, network defenders are largely blind, particularly against domain fronted communications which travel via legitimate third-party websites.

If SSL Inspection is implemented then the HTTP traffic can be viewed, and while PoshC2 encrypts the contents of the HTTP bodies, the HTTP URLs and headers can still be viewed.

URLs

PoshC2 has two different ways of generating URLs to use for communications. Operators can either use a static list of URLs or provide a wordlist from which random URLs will be generated, and these files are stored at resources/urls.txt and

resources/wordlist.txt respectively, with the static URL list being the default option. These URLs are then loaded into the database when the project is first created, and a random URL is chosen by each implant each time it beacons. The default URL list is below:

/adsense/troubleshooter/1631343/ /adServingData/PROD/TMClient/6/8736/ /advanced_search?hl=en-GB&fg= /async/newtab?ei= /babel-polyfill/6.3.14/polyfill.min.js= /bh/sync/aol?rurl=/ups/55972/sync?origin= /bootstrap/3.1.1/bootstrap.min.js?p= /branch-locator/search.asp?WT.ac&api= /business/home.asp&ved= /business/retail-business/insurance.asp?WT.mc_id= /cdba?ptv=48&profileId=125&av=1&cb= /cisben/marketg?bartype=AREA&showheader=FALSE&showvaluemarkers= /classroom/sharewidget/widget_stable.html?usegapi= /client_204?&atyp=i&biw=1920&bih=921&ei= /load/pages/index.php?t= /putil/2018/0/11/po.html?ved= /qqzddddd/2018/load.php?lang=en&modules= /status/995598521343541248/guery= /TOS?loc=GB&hl=en&privacy= /trader-update/history&pd= /types/translation/v1/articles/ /uasclient/0.1.34/modules/ /usersync/tradedesk/ /utag/lbg/main/prod/utag.15.js?utv= /vfe01s/1/vsopts.js? /vssf/wppo/site/bgroup/visitor/ /wpaas/load.php?debug=false&lang=en&modules= /web/20110920084728/ /webhp?hl=en&sa=X&ved= /work/embedded/search?oid= /GoPro5/black/2018/ /Philips/v902/

While these URLs were originally copied from legitimate requests, if you see several of them being repeated to a site, particularly if they do not seem relevant to that site and if the response does not make sense, then it could be PoshC2 beacon traffic.

HTTP Responses

PoshC2 also has static HTML responses that it responds with. The default is six HTTP 200 responses and one 404 response. These are stored in files at resources/responses/ and also loaded into the database when the server is first created. The server responds with a

random 200 response to POST requests that do not error or require a specific response, and with the single 404 response to all unexpected URLs or when the C2 server errors. Other responses return context relevant data, such as tasks, implant code and so on.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
The requested URL was not found on this server.
<hr>
<address>Apache (Debian) Server</address>
</body></html>
```

This static HTML file then at resources/responses/404_response.html is an IoC and if returned from a webserver that you are investigating is suggestive of PoshC2. Similarly, the 200_response* files in the same directory are IoCs if returned by POST requests.

Note however, that it is recommended that operators change these files before creating a PoshC2 project, as is the use of a C2 proxy, so as with the other indicators the absence of this particular response is not unexpected for a PoshC2 installation.

SSL Certificate

PoshC2 by default creates a self-signed certificate for its HTTP server, the values for which are stored in poshc2/server/Config.py file. These values are not in the 'normal' configuration file config.yml and are less documented and are therefore harder to change.

```
Cert_C = "US"
Cert_ST = "Minnesota"
Cert_L = "Minnetonka"
Cert_O = "Pajfds"
Cert_OU = "Jethpro"
Cert_CN = "P18055077"
Cert_SerialNumber = 1000
Cert_NotBefore = 0
Cert_NotAfter = (10 * 365 * 24 * 60 * 60)
```

An experienced operator will not expose their C2 server to the internet, but will instead use a proxy server with a valid certificate and filter firewall traffic to the C2 server that is not from that proxy, however if these steps are not taken and a certificate with the below values is presented then it is another strong indicator that PoshC2 is in use, likely by less sophisticated adversaries.

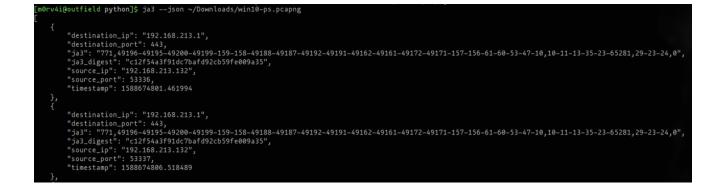
Certificate)
General Details Certification Pat	th	
Show: <all></all>	~	
Field	Value	^
🗒 Signature hash algorithm	sha 1	
🔲 Issuer	P18055077, Jethpro, Pajfds,	
🔲 Valid from	04 May 2020 13:46:24	
🔲 Valid to	02 May 2030 13:46:24	
Subject	P18055077, Jethpro, Pajfds,	
Public key	RSA (2048 Bits)	
Public key parameters	05 00	5
	1686d8cf855f2594165b98111	*
CN = P18055077 OU = Jethpro O = Pajfds L = Minnetonka S = Minnesota C = US		
	Edit Properties Copy to File	
	0	К

The issuer data can be viewed using Chrome and the default values suggest this is a PoshC2 server.

JA3 Signatures

Another method for signaturing C2 traffic without SSL Inspection is to fingerprint the *Client Hello* packet that initialises the TCP connection. The specific bytes that make up the packet are dependent upon the type of connection being formed and the underlying packages and methods used to do so.

As these packets are sent before encryption has been established they are cleartext and can be intercepted and read. It turns out that the packet contents are quite unique and can be fingerprinted. Salesforce have an <u>excellent blog post on JA3 fingerprinting</u>, but essentially the details are extracted from the packets that make up this TCP handshake and hashed to create a quick and easy signature, that can be used to identify not only that PoshC2 is in use, but also the specific implant type.



The current Windows 10 PoshC2 signatures are below:

PowerShell: c12f54a3f91dc7bafd92cb59fe009a35 Sharp: fc54e0d16d9764783542f0146a98b300

JA3 fingerprinting has been incorporated into security products such as Splunk and Darktrace to provide quick and easy identification of C2 traffic, and can be utilised with these fingerprints. These fingerprints are available and will be maintained in the <u>PoshC2</u> <u>Detections GitHub repository</u>.

Telemetry

Another mechanism for detecting C2 traffic, with or without SSL Inspection, is through the use of telemetry.

PoshC2's default values for beacon time and jitter in the **config.yml** file are 5 seconds and a 20% jitter, as can be seen below, which is both a fast beacon rate and a small jitter, making it relatively easy to detect using this method.

DefaultSleep: "5s" Jitter: 0.20

With these values the implants are going to beacon every 4-6 seconds by default, with an average of 5 seconds.

After capturing some traffic, filtering by the C2 server host and only checking TLS Client Hello packets, we can see that the TLS connection is created roughly every five seconds, confirming what we expect.

	st == 192.168.213.1	&& ssl.handshake.type == 1			
	Time	Source	Destination	Protocol Length Info	
ſ	18 4.601666	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
·	44 9.649968	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
	69 13.715747	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
	94 19.775269	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	L21 24.823069	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
	46 28.886231	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	158 33.916277	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	L83 38.973428	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	208 43.007701	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	233 49.025732	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
1	259 54.056501	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
3	309 60.068042	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
3	335 64.105470	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
3	363 69.168948	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
3	392 75.192406	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
4	118 80.223900	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
4	156 85.239605	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
4	468 89.271020	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
5	506 95.340147	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
5	518 99.385810	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
5	543 103.442443	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
5	581 108.489625	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
5	593 112.543649	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
6	518 118.603521	192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
			192.168.213.1	TLSv1.2 361 Client Hello	
	556 123.638279	192.168.213.132	192.100.215.1	105VI.2 Sol click hello	
6	556 123.638279 581 128.692775	192.168.213.132 192.168.213.132	192.168.213.1	TLSv1.2 361 Client Hello	
((Fra	me 18: 361 byte	192.168.213.132 s on wire (2888 bits	192.168.213.1), 361 bytes captured	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F	> 84A4AF27D14}, id 0
fra Eth Int Tra Tra	me 18: 361 byte ernet II, Src: V ernet Protocol V nsmission Contro nsport Layer Se TLSv1.2 Record 1	192.168.213.132 s on wire (2888 bits WWaare_c4:cf:34 (00: Version 4, Src: 192.: ol Protocol, Src Port curity Layer: Handshake Prot : Handshake (22)	192.168.213.1), 361 bytes capture 0c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VMware_c0:00:08 (00:50:56:c0:00:08)	
fra Eth Int Tra Tra	me 18: 361 byte ernet II, Src: " ernet Protocol " nsmission Contr nsport Layer Se TLSV1.2 Record I Content Type: Version: TLS Length: 302	192.168.213.132 s on wire (2888 bits WWaare_c4:cf:34 (00: Version 4, Src: 192.: ol Protocol, Src Port curity Layer: Handshake Prot : Handshake (22)	192.168.213.1), 361 bytes capture 0c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWware_c0:00:08 (00:50:56:c0:00:08) 168.213.1	
fra Eth Int Tra Tra	me 18: 361 byte ernet II, Src: " ernet Protocol " nsmission Contr nsport Layer Se TLSV1.2 Record I Content Type: Version: TLS Length: 302	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00: Version 4, Src: 192.: ol Protocol, Src Pori curity Layer: Handshake Prof : Handshake (22) 1.2 (0x0303)	192.168.213.1), 361 bytes capture 0c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWware_c0:00:08 (00:50:56:c0:00:08) 168.213.1	
((Eth Int Tra Y	me 18: 361 byte ernet II, Src: ' ernet Protocol ' nsmission Contr nsport Layer Se TLSV1.2 Record I Content Type: Version: TLS Length: 302 Handshake Pro	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00: Version 4, Src: 192.) ol Protocol, Src Por curity Layer: Handshake Prot : Handshake (22) 1.2 (0x0303) ptocol: Client Hello	192.168.213.1), 361 bytes capture 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4 cocol: Client Hello	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWware_c0:00:08 (00:50:56:c0:00:08) 168.213.1	
fra Eth Int Tra Tra 30	me 18: 361 byte: me 18: 361 byte: ernet II, Src: ' ernet Protocol ' nsmission Contrr nsport Layer Set Content Type: Version: TLS Length: 302 Handshake Pro 20 14 2d 25 00 03 5e 00 2b 40	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (000: Version 4, Src: 192.: al Protocol, Src Por- curity Layer: Handshake Prot : Handshake (22) 1.2 (0x0303) btocol: Client Hello 00 16 03 03 01 22 0 75 18 47 08 0 a 05	192.168.213.1), 361 bytes captures 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19: t: 50122, Dst Port: 4 tocol: Client Hello 10 00 01 2a 03 -5 d 65 b0 d8 c4 -5	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307	
(Fra Eth Int Tra Tra 30 40 50	<pre>581 128.692775 me 18: 361 byte ernet II, Src: ' ernet Protocol ' nsmission Contry nsport Layer Se TLSv1.2 Record I Content Type: Version: TLS Length: 302 Handshake Pro 20 14 2d 25 00 03 5e b0 2b 40 46 24 8f a a 9d</pre>	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00: Version 4, Src: 192.; ol Protocol, Src Por curity Layer: Handshake (22) 1.2 (0x0303) otocol: Client Hello 00 16 03 03 01 22 0 75 18 47 08 08 08 0 05 56 86 65 C5 95	192.168.213.1), 361 bytes captured 2c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: - t: cocol: Client Hello 1 00 01 2a 03 5 bd 65 b0 dB e4 1 3 81 6a 64 87 48 5 5 5 5 5 5 5 	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307	
(Fra Eth Int Tra Tra 30 40 50 50	<pre>881 128.692775 me 18: 361 byte: ernet II, Src: ' ernet Protocol ' nsmission Contrr nsport Layer Se TLSv1.2 Record I Content Type: Version: TLS Length: 302 > Handshake Pro 20 14 2d 25 00 03 5e b0 2b 40 46 24 8f aa 9d 54 00 00 2a c0</pre>	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00:: Version 4, Src: 192.: ol Protocol, Src Pori curity Layer: Handshake Prof : Handshake (22) 1.2 (0x0303) btocol: Client Hello 00 16 03 03 01 22 05 18 47 d8 0a d8 c db 56 88 66 5c 59 0 2c c0 2b c0 30 c0 2	192.168.213.1), 361 bytes captures 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19: t: 50122, Dst Port: 4 tocol: Client Hello 1000 01 2a 039 1000 01 2a 039 36 55 b0 d6 e4 -^ 381 6a f5 48 f\$ f00 9f 00 9e f	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307 	
Eth Int Tra 70	<pre>881 128.692775 me 18: 361 byte: ernet II, Src: ' ernet Protocol ' nsmission Contr. nsport Layer Se: TLSV1.2 Record I Content Type: Version: TLS Length: 382 > Handshake Pro 20 14 2d 25 00 03 5e b0 2b 40 46 24 8f aa 9d 54 00 00 2a c0 06 24 c0 23 c0</pre>	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00: Version 4, Src: 192.; ol Protocol, Src Por curity Layer: Handshake Prot : Handshake (22) 1.2 (0x0303) ptocol: Client Hello 00 16 03 03 01 2e db 56 88 6e 5c 59 2c c0 2b c0 08 c0 2; 28 c0 27 c0 08 c6 2;	192.168.213.1), 361 bytes captures 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: - t: 50122, Dst Port: - t: 50122, Dst Port: - t: 50122, Dst Port: - t: 50122, Dst Port: - 50100 01 2a 035 04 65 b0 d8 e4 3 81 6a f5 48 F\$- f*0 9f 00 9e 9c 01 4 c0 13 -5;**	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307	
() Fra Eth Int Tra Tra 30 40 50 60 70 80	<pre>581 128.692775 me 18: 361 byte: ernet II, Src: ' ernet Protocol ' nsmission Contre sport Layer Se: TLSv1.2 Record I Content Type: Version: TLS Length: 302 > Handshake Pro 20 14 2d 25 00 03 5e b0 2b 40 03 5e b0 2b 40 03 5e b0 2b 40 03 5e 00 2b 40 06 04 00 02 a c0 06 04 00 92 00</pre>	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00:: Version 4, Src: 192.: ol Protocol, Src Pori curity Layer: Handshake Prof : Handshake (22) 1.2 (0x0303) btocol: Client Hello 00 16 03 03 01 22 05 18 47 d8 0a d8 c db 56 88 66 5c 59 0 2c c0 2b c0 30 c0 2	192.168.213.1), 361 bytes captures 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4 t: 50122,	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307 	
fra Eth Int Tra Tra	<pre>881 128.692775 me 18: 361 byte: ernet II, Src: ' ernet Protocol ' nsmission Contrr nsport Layer Se TLSV1.2 Record I Content Type: Version: TLS Length: 302 > Handshake Pro 20 14 2d 25 00 03 5e b0 2b 40 46 24 8f aa 9d 46 24 8f aa 9d 46 24 8f aa 9d 60 46 00 02 ac 00 00 9d 00 9c 00 00 9d 00 9c 00 00 9d 00 9c 00 00 70 00 a0 00</pre>	192.168.213.132 s on wire (2888 bits Whware_c4:cf:34 (00: Version 4, Src: 192.: ol Protocol, Src Por curity Layer: Handshake Prot : Handshake (22) 1.2 (0x0303) btocol: Client Hello 00 16 03 03 01 22 0 05 18 47 08 0a 05 04 56 88 65 55 95 22 c0 2b c0 30 02 28 c0 27 c0 08 c0 30 00 35 00 55	192.168.213.1), 361 bytes captures 3c:29:c4:cf:34), Dst 168.213.132, Dst: 19 t: 50122, Dst Port: 4 tocol: Client Hello 1 00 01 2a 03 +5 +5 +5 +6 06 9f 09 ef +9 c0 14 c0 13 +5 +7 +7 00 18 00 05 +5 +7 +7 +7 +7 +7 +7 +7 +7 +7 +7	TLSv1.2 361 Client Hello (2888 bits) on interface \Device\NPF_{3C2891A5-9C77-4A87-93AD-F VWare_c0:00:08 (00:50:56:c0:00:08) 168.213.1 3, Seq: 1, Ack: 1, Len: 307 	

We can see the difference between each session initialisation is between 4 and 6 seconds.

By exploring this small sample of data, we can determine that the average time between the requests is 5.05 seconds, with a standard deviation of 0.81 or 16%, close to the 5 second beacon with the 20% jitter we expect from PoshC2's defaults.

Security products that can provide telemetry data, e.g. Splunk can be used to detect C2 traffic in this way by checking for repetitive beacons at a relatively fixed frequency, whether it's at the above defaults for PoshC2 or at any frequency, given that it is configurable.

PowerShell Implant

💰 *Ethernet0

PoshC2 has three implant types; PowerShell, C#, and Python, with the latter being a lightweight implant type mostly intended for compromising *nix hosts. The PowerShell Implant runs by loading System.Management.Automation.dll, the engine behind PowerShell.exe, into the desired process and executing PowerShell commands directly using this DLL.

 \times

Task 00054 (R2B2) issue 05/2020 07:43:44) get-process -id 1808	d against implant 12 on host BEEROCLOCK\bob @ BEEROC <mark>117/19</mark> fl ★	419 PoshC2 v6.0 (3ec4308 2020-05-04 10:47:58)
Task 00054 (R2B2) retur 5/05/2020 07:43:45)	rned against implant 12 on host BEEROCLOCK\bob @ BEEROCLOCK	(0 [12]: Seen:05/05/2020 07:43:28 PID:1808 Ss BEEROCLOCK\bob @ BEEROCLOCK (AMD64) PS Select ImplantID or ALL or Comma Separated List (Enter to refresh):: 12
Name Id PriorityClass FileVersion HandleCount WorkingSet PagedMemorySize VirtualMemorySize VirtualMemorySize VirtualMemorySize VirtualMemorySize VirtualMemorySize VirtualMemorySize VirtualMemorySize VatalProcessorTime SI Handles VM WS PM PM PM PM PM PAth Company CPU ProductVersion Description ProductVersion Description Product SasePriority ExitCode HasExited	<pre>: netsh : 1808 : Normal : 10.0.18362.1 (WinBuild.160101.0800) : 603 : 806232448 : 80621568 : 80621568 : 719328320 : 00:00:06.8906250 : 1 : 603 : 22244028551168 : 80232448 : 80621568 : 306251568 : 30628 : G:\Windows\system32\netsh.exe : Microsoft Corporation : 6.90625 : 10.0.18362.1 : Network Command Shell : Microsoft Windows* Operating System : Process : 8 : 6 : 6 : 8 : 6 : 8 : 6 : 8 : 8 : 6 : 8 : 8 : 8 : 6 : 8 : 8 : 8 : 8 : 8 : 8 : 8 : 8 : 8 : 8</pre>	BEEROCLOCK\bob @ BEEROCLOCK (PID:1808) PS 12> get-process -id 1808 fl * BEEROCLOCK\bob @ BEEROCLOCK (PID:1808) PS 12>
Hastrico Exitime Handle SafeHandle MachineName MainNindowHandle MainNindowTitle MainNodule	<pre>. Parsa : : 1832 : Microsoft.Win32.SafeHandles.SafeProcessHandle : . : 0 : : System.Diagnostics.ProcessModule (netsh.exe)</pre>	

The PowerShell implant supports full PowerShell execution from any process (here netsh.exe) by loading System.Management.Automation.dll.

While this does not utilise **PowerShell.exe**, bypassing many restrictions and controls, it is still subject to PowerShell specific constraints such as Constrained Language mode and ScriptBlock logging. The latter, in particular, can be used to detect PoshC2 and any other malicious PowerShell commands, whether via **PowerShell.exe** or otherwise.

Note, however, that ScriptBlock logging was only enabled in PowerShell v5.0 and above, so if PowerShell v2.0 is available on the target then a downgrade attack can be performed and PowerShell 2.0 used, bypassing Constrained Language mode, ScriptBlock logging, AMSI and other controls added in v5.0.

PS Logging

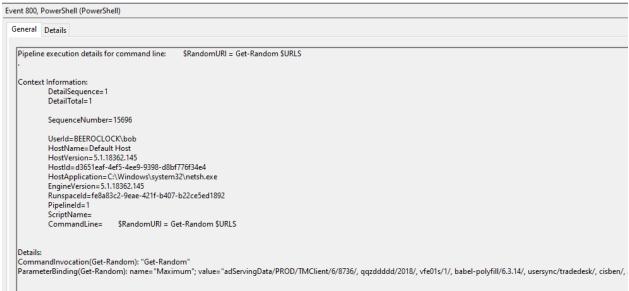
Enabling ScriptBlock logging and Transcript Logging for PowerShell allows logging of processes running PowerShell commands.

There are multiple IoCs here, not least that the Host Application is not **PowerShell.exe**, as should be expected, but instead **netsh.exe**.

****** Windows PowerShell transcript start Start time: 20200504164342 Username: BEEROCLOCK\bob RunAs User: BEEROCLOCK\bob Configuration Name: Machine: BEEROCLOCK (Microsoft Windows NT 10.0.18362.0) Host Application: C:\Windows\system32\netsh.exe Process ID: 7772 PSVersion: 5.1.18362.145 PSEdition: Desktop PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.18362.145 BuildVersion: 10.0.18362.145 CLRVersion: 4.0.30319.42000 WSManStackVersion: 3.0 PSRemotingProtocolVersion: 2.3 SerializationVersion: 1.1.0.1 ***** ****** Command start time: 20200504164342 ****** PS>\$o = IEX \$c | Out-String ******

PowerShell transcripts provide valuable information. Alerting on Host Applications that are not PowerShell.exe is a good way to find PowerShell implants from any C2 framework.

Similarly in the Event Viewer for ScriptBlock logging:



PowerShell ScriptBlock logging can reveal a wealth of information, here including the beacon URLs for PoshC2 and again netsh.exe as a host process.

This also includes the 'command line' of the ScriptBlock being executed, which includes the beacon URLs from which a random element is being chosen, therefore listing all the C2 URLs that are in use by this implant.

Aside from the wealth of information ScriptBlock logging provides for any threat hunter, an event command line specific to PoshC2 that can be used to identify the PowerShell implant is below:

Event 800, PowerShell (PowerShell)
General Details
Pipeline execution details for command line: \$ServerClean = Get-Random \$ServerURLS
Context Information:
DetailSequence=1
DetailSequence=1
SequenceNumber=15698
UserId=BEEROCLOCK\bob
HostName=Default Host
HostVersion=5.1.18362.145
HostId=d3651eaf-4ef5-4ee9-9398-d8bf776f34e4
HostApplication=C:\Windows\system32\netsh.exe
EngineVersion=5.1.18362.145 RunspaceId=fe8a83c2-9eae-421f-b407-b22ce5ed1892
Pipelineld=1
ScriptName=
CommandLine= \$ServerClean = Get-Random \$ServerURLS
Details:
CommandInvocation(Get-Random): "Get-Random"
ParameterBinding(Get-Random): name="Maximum"; value="https://192.168.213.1:443, https://192.168.213.1:443"

The \$ServerClean variable is specific to PoshC2 and a clear loC.

CommandLine= \$ServerClean = Get-Random \$ServerURLS

This ScriptBlock is part of the beaconing process and will be repeated frequently in the Event Viewer making it easy to identify.

System.Management.Automation.dll

As mentioned earlier, the PowerShell implant does not function by invoking PowerShell.exe, but instead by loading System.Management.Automation.dll into the implant process.

This means that this is an IoC, and if we find a process that should not have this DLL loaded, particularly if it is an unmanaged code binary (so not .NET) then it is highly likely that this is a process that has been injected into by a PowerShell implant, PoshC2 or otherwise.

System.Mana	agement.Automation.
Property	Value
File Name	C:\Program Files (x86)\Reference Assemblies\Microsoft\WindowsPo
File Type	Portable Executable 32 .NET Assembly
File Info	Microsoft Visual Studio .NET

System.Management.Automation.dll, the engine behind PowerShell, is a .NET library.

netsh.exe	<u>]</u>
Property	Value
File Name	C:\Windows\System32\netsh.exe
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0

netsh.exe is a C++ binary and should not be loading .NET libraries.

We can see this DLL loaded into an implant process using Process Hacker:

📧 netsh.exe (1808) Properties

_

Handles	.NET ass			NET performance	GF	-		and Network	Comm	-
General	Statistics	Perform	nance	Threads	Token		Modules	Memory	Environ	ment
Name				Base address	Size	Descrip	tion			^
schannel.dll				0x7ffbc464	544 kB	TLS / SS	SL Security	Provider		
sechost.dll				0x7ffbc7c3	604 kB	Host fo	r SCM/SDDI	L/LSA Lookup APIs		
secur32.dll				0x7ffbb81f	48 kB	Securit	y Support P	rovider Interface		
SHCore.dll				0x7ffbc657	676 kB	SHCOR	E			
shell32.dll				0x7ffbc74e	6.89 MB	Window	vs Shell Con	nmon Dll		
shlwapi.dll				0x7ffbc66d	328 kB	Shell Lig	ght-weight l	Utility Library		
SortDefault.r	hls			0x298fedc	3.21 MB					
sspicli.dll				0x7ffbc52b	188 kB	Security	y Support P	rovider Interface		
System.Conf	iguration.Install.	ni.dll		0x7ffba0d5	180 kB	.NET Fr	amework			
System.Conf	iguration.ni.dll			0x7ffb98f4	1.2 MB	System	.Configurat	ion.dll		
System.Core	.ni.dl			0x7ffb9908	10.46 MB	.NET Fr	amework			
System.Data	.dll			0x7ffb9c23	3.41 MB	.NET Fr	amework			
System.Data	.ni.dll			0x7ffba0f2	9.44 MB	.NET Fr	amework			
System.Dired	toryServices.ni.	dll		0x7ffba18f	1.4 MB	.NET Fr	amework			
System.Mana	agement.Automa	ation.ni.dll		0x7ffb932c	32.4 MB	System	.Manageme	ent.Automation		
System.Mana	agement.ni.dll			0x7ffba1ab	1.4 MB	.NET Fr	amework			
System.ni.dll				0x7ffb9c7e	12.44 MB	.NET Fr	amework			
System.Num	erics.ni.dll			0x7ffba189	324 kB	.NET Fr	amework			
System.Tran	sactions.dll			0x7ffba0d8	316 kB	.NET Fr	amework			
System.Tran	sactions.ni.dll			0x7ffba0dd	876 kB	.NET Fr	amework			
System.Xml.r	ni.dll			0x7ffb9869	8.67 MB	.NET Fr	amework			
ucrtbase.dll				0x7ffbc62d	0.98 MB	Microso	ft® C Runt	ime Library		
ucrtbase_dr(0400.dll			0x7ffba926	756 kB	Microso	ft® C Runt	ime Library		
umpdc.dll				0x7ffbc539	64 kB					
user32.dll				0x7ffbc673	1.58 MB	Multi-Us	ser Window	s USER API Client D	LL	
userenv.dll				0x7ffbc528	148 kB	Useren	v			
vcruntime 14(0_dr0400.dll			0x7ffba932	88 kB	Microso	ft® C Runt	ime Library		
version.dll				0x7ffbbdb3	40 kB	Version	Checking a	nd File Installation I	Libraries	
vmhgfs.dll				0x65140000	144 kB	VMware	e HGFS Prov	/ider		
win32u.dll				0x7ffbc587	132 kB	Win32u	I.			
windows.sto	rage.dll			0x7ffbc5aa	7.5 MB	Microso	ft WinRT St	torage API		
winhttp.dll				0x7ffbbccb	960 kB	Window	vs HTTP Ser	vices		~
winnlsres.dll				<						>

System.Management.Automation.dll loaded into netsh.exe.

Similarly we can view the .NET Assemblies in the process:

netsh.exe:1808 Properties

_

Image	Performar	nce	Perfor	mance Graph	Disk and Networ	k	GPU Graph	Threads	TCP/IP
Security	/	Envi	ronment	.N	ET Assemblies		.NET Performance	2	Strings
Structure				Flags	Path				
□ CLR v4.	0 20219 0			ConcurrentGC					
	omain: Share	dDoma	in	Shared					
	scorlib	uDoma			C:\WINDOWS\asse	mblv\N	ativelmages v4.0	30319 64\m	scorlib\ce65e
	omain: Defau	ltDoma	in	Default, Exec			direiningeo_r-rie		000000
	nonymously H			Dynamic	Anonymously Hosted	Dynan	nicMethods Assem	ıblv	
	Microsoft Microsoft Management Infrastr Microsoft PowerShell Comman Microsoft PowerShell Comman Microsoft PowerShell Comman Microsoft PowerShell Console Microsoft PowerShell Security			-	Microsoft				
М	icrosoft.Mana	gemen	t.Infrastr	Native	C:\WINDOWS\asse	mbly\N	ativelmages_v4.0	.30319_64\M	icrosoft.Mf49f
					C:\WINDOWS\asse				
					C:\WINDOWS\asse				
					C:\WINDOWS\asse				
				Native	C:\WINDOWS\asse				
				Native	C:\WINDOWS\asse				
	icrosoft.WSM	an.Mar	nagement		C:\WINDOWS\asse				
_	ystem Confirm			Native Native	C:\WINDOWS\asse	-			
	ystem.Configu ystem.Configu		ootall	Native	C:\WINDOWS\asse C:\WINDOWS\asse				
	ystem.Coringu ystem.Core	ration.i	ISLOII	Native	C:\WINDOWS\asse				
	vstem.Data			Native	C:\WINDOWS\asse				
	stem.Director	vServi	es	Native	C:\WINDOWS\asse				
	vstem.Manage			Native	C:\WINDOWS\asse				
S	stem.Manage	ement./	Automati	Native	C:\WINDOWS\asse				
S	stem.Numerio	CS		Native	C:\WINDOWS\asse	mbly\N	ativelmages_v4.0	.30319_64\S	ystem.Numerio
S	ystem.Transad	ctions		Native	C:\WINDOWS\asse				
S	ystem.Xml			Native	C:\WINDOWS\asse	mbly\N	ativelmages_v4.0	.30319_64\S	ystem.Xml∖e4
				<					>
							_	_	Cancel

We can view the .NET Assemblies in the netsh.exe process and see PowerShell is loaded.

This tab shouldn't even be present for a genuine **netsh.exe** process as it is unmanaged code!

Handles GPU				Disk and Network			Comment			
General	Statistics	Performance	Threa	ads Tok	ien M	odules	Memory	Environ	men	
Name			Ba	se address	Size	Descripti	on		^	
netsh.e	netsh.exe			7ff6585	140 kB	3 Network Command Shell				
activeds.	dll		0x7	7ffb8e43	276 kB	ADs Rou	ter Layer DLL			
adsldpc.d			0x2	7ffb8b64	264 kB	ADs LDAP Provider C DLL				
advapi32	.dll		0x	7ffbc68d	652 kB	Advanced Windows 32 Base API				
authfwcf	g.dll		0x7ffba1dc 540 kB W			Windows Defender Firewall with Ac				
bcrypt.dl	I		0x7ffbc58a 152 kB Windows			ndows Cryptographic Primitives				
bcryptpri	mitives.dll		0x	0x7ffbc5a2 512 kB Windows			Windows Cryptographic Primitives			
cabinet.d			0x2	0x7ffbbe7a 164 kB Mid			3 Microsoft® Cabinet File API			
cfgmgr32	2.dll		0x	0x7ffbc622 296 kB Config			Configuration Manager DLL			
combase.	.dll		0x	0x7ffbc6d4 3.21 MB Microsof			Microsoft COM for Windows			
crypt32.c	III		0x	0x7ffbc58d 1.29 MB Crypto A			Crypto API32			
cryptbase	e.dll		0x	0x7ffbc4dd 48 kB Base cry			ase cryptographic API DLL			
cryptsp.c			0x	7ffbc545	92 kB	Cryptographic Service Provider API			PI	
dhcpcmor	nitor.dll		0x	0x7ffbaefe 32 kB			ent Monitor Dl			
dhcpcsvc	dll		0x2	0x7ffbbe06 112 k			B DHCP Client Service			
dhcpcsvc	6.dll		0x2	7ffbbe08	88 kB	DHCPv6	/6 Client			
dnsapi.dl					0x7ffbc495 808 kB DNS Clier			ent API DLL		

The module is not loaded in a genuine netsh process and the .NET tabs are not available.

An operator can be smarter about their migration by injecting into .NET processes, however it is still unlikely that a legitimate process (that isn't PowerShell.exe) would load
System.Management.Automation.dll, so this is a great IoC to look out for in your
environment.

This can be implemented at scale by searching for loaded modules in an EDR product, for example in CarbonBlack with a query of:

```
modload:System.Management.Automation* AND -process_name:powershell.exe AND
-process_name:powershell_ise.exe
```

This searches for process with System.Management.Automation.dll or System.Management.Automation.ni.dll (the Native Image version) loaded into a process when that process is *not* PowerShell.exe or PowerShell_ISE.exe. Other legitimate exclusions may need to be added for your particular environment.

Q modload:system.management.automation* and -process_name:powers	shell.exe and -process_name:powershell_ise.exe
+ ADD SEARCH TERMS C RESET SEARCH GROUP BY PROCESS	
Results	
Process	Endpoint
rundll32.exe c:\windows\syswow64\rundll32.exe	win7-client1
c:\windows\system32\rundll32.exe	win7-client5
rundll32.exe c:\windows\system32\rundll32.exe	win7-client5
c:\windows\system32\svchost.exe	win2016-s1
rundll32.exe c:\windows\system32\rundll32.exe	win2016-s1
posh64.exe \\tsclient\share\posh64.exe	win10-client11

Here we can see various processes that have loaded System.Management.Automation.dll into memory that are likely implants.

C# Implant

The C# or Sharp implant is PoshC2's 'next gen' implant, written (unsurprisingly) in C#. The key difference from a detection perspective is that this implant does not require loading **System.Management.Automation.dll** in order to function. Most of the functionality of the C# implant is custom-written and while it *can* load **System.Management.Automation.dll** in order to execute PowerShell, this is an operator decision and is by no means necessary.

A similar process to the above can be applied, but is a little harder to implement. .NET processes load the mscoree.dll library which is the core library behind the .NET CLR (Command Language Runtime), so again any unmanaged code processes that are loading this library could have been migrated into. The issue here is finding these processes, as it's not as simple as searching for just 'module loaded and process name is not powershell.exe'.

A blacklist can be created of common target processes that are unmanaged, always available and should not be loading the .NET runtime, and these can be monitored, as well as noting this during manual triage.

📧 netsh.exe (1600) Properties

Handles	.NET asse	.NET assemblies		.NET performance GP		PU Disk and Netwo		Comment			
General	Statistics	tics Performance		Threads	Token	Modules Memory Envir					
Name				Base address	Size	Description		^			
MpClient.dll				x7ffbb59b	936 kB	Client Interfa	ice				
MpOAV.dll				x7ffbb882	280 kB	IOfficeAntiVir	us Module				
msasn1.dll				x7ffbc543	72 kB	ASN. 1 Runtin	ne APIs				
mscoree.dll				x7ffba950	400 kB	Microsoft .NE	T Runtime Executio	n Engine			
mscoreei.dll				x7ffba93c	676 kB	Microsoft .NE	T Runtime Executio	n Engine			
mscorlib.ni.c	dli			x7ffb9d5a	22 MB	Microsoft Cor	nmon Language Ru	ntime Class Libr			
mskeyprote	ct.dll			x7ffbb4ed	84 kB	Microsoft Key Protection Provider					
msvcp_win.e	dll			x7ffbc57d	632 kB	Microsoft® C	Runtime Library				
msvcrt.dll				x7ffbc83e	632 kB	Windows NT	CRT DLL				
mswsock.dll)x7ffbc4c0	412 kB	Microsoft Wir	ndows Sockets 2.0 S	Service Provider			
ncrypt.dll				x7ffbc4ed	152 kB	Windows NCr	ypt Router				
ncryptsslp.d	HII)x7ffbb4f2	148 kB	Microsoft SC	nannel Provider				

×

A C# implant loads mscoree.dll if it is not already present in the process – another IoC if the process is supposed to be unmanaged.

Note that the PowerShell implant will also load this library as it is also a .NET process, however the presence of System.Management.Automation.dll marks it as an implant that can run PowerShell.

The C# implant has the ability to load compiled binaries into memory over C2 and run them, which is extremely powerful. There are indicators of this however, as a new virtual runspace is created for each module and their namespace is listed in the AppDomain to which they are loaded.

Viewing the .NET Assemblies in Process Explorer or Process Hacker for example then reveals what modules have been run.

General	Statistics	Performan	ce	Threads	Token	Modul	es	Memory	Environm	nent
Handles	.NET asse	emblies	.NE	[performance	GPU		Disk a	nd Network	Comme	ent
Structure				ID	Flags		Path			
✓ CLR v4.0	.30319.0			6	CONCURRENT_GC,					
Y AppD	omain: DefaultD	omain		29948734	Default, Execu	utable				
C	ore			29953121			Core			
Ci	CustomMarshalers			29953183	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
dr	opper_cs			29948742			drop	per_cs		
M	crosoft			29948739			Micro	soft		
Se	atbelt			29953122			Seat	belt		
Sł	harpUp			29953121			Shar	pUp		
S	/stem			29948739	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
S	/stem.Configura	tion		29948742	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
S	/stem.Core			29948742	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
Sy	/stem.Managem	ent		29953122	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
Sy	/stem.Web.Exte	insions		29953183			C:\W	INDOWS Micros	oft.Net\asse	embl
Sy	/stem.Xml			29948742	Native		C:\W	INDOWS Micros	oft.Net\asse	embl
Y AppD	omain: SharedD	omain		14071868	Shared					
m	scorlib			29948737	DomainNeutra	l, Native	C:\W	INDOWS Micros	oft.Net\asse	embl

In the C# implant loaded modules have their namespace visible in the AppDomain, making it clear what has been loaded into the implant.

Above we can see the *Core* and *dropper_cs* modules that make up the core functionality of the C# implant, as well as some native modules from Microsoft that are required to run. These modules will always be present in the PoshC2 C# implant and are a clear IoC. We also see <u>Seatbelt</u> and <u>SharpUp</u>, two common C# offensive modules from SpectreOps, and we can surmise that they have been on the target.

In General

Migration

We have used netsh.exe as the example implant process in this post and that is for a good reason. The default migration process for PoshC2 in the C# and PowerShell implants is C:\Windows\System32\netsh.exe, so when the migrate or inject-shellcode commands are used and a specific process ID or name is not set, then a new netsh.exe process is spawned and migrated into.

This itself is a common IoC for PoshC2, as **netsh.exe** is not typically frequently run in environments, and certainly not on most end user's hosts. Therefore, if there is a sudden uptick in the number of these processes being run in the environment or if several are running on a host then it could be worth investigating.

Persistence

PoshC2 has three quick persistence commands available to the PowerShell implant. Each of these installs a PowerShell.exe one liner payload to the registry in the key at HKCU\Software\Microsoft\Windows\CurrentVersion\themes with a name Wallpaper777, Wallpaper555 or Wallpaper666, depending on the command being run.

This payload is then triggered by either:

- A registry key at HKCU\Software\Microsoft\Windows\CurrentVersion\run with the name IEUpdate
- A Scheduled Task, also with the name IEUpdate
- A shortcut file placed at %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\IEUpdate.lnk

All of these can be alerted upon and used to determine that the adversary using PoshC2, in addition to alerting on the invocation of PowerShell.exe with encoded parameters, and so on.

Binary payloads

Some of the more common payloads that are dropped on targets are the PoshC2 executables and DLLs that can be run using rundll32.exe.

For the DLLs, there are different versions for PowerShell and Sharp implants across versions 2 and 4 of PowerShell and x86 and x64 bit architectures, however all the DLLs have a single entry-point common to all: VoidFunc.

Member		Offset		Size		Value		
Characteristics		0001	3A20	Dword		0000000		
TimeDateStamp		00013A24		Dword		FFFFFFF		
MajorVersion		00013A28		Word		0000		
MinorVersion		00013A2A		Word		0000		
Name		00013A2C		Dword		00014852		
Base		00013A30		Dword		0000001		
NumberOfFunctions		00013A34		Dword		0000001		
NumberOfNames		00013A38		Dword		00000001		
AddressOfFunctions		00013A3C		Dword		00014848		
Ordinal Function R		VA Name Ordi		inal Name RVA			Name	
(nFunctions)	Dword		Word		Dword		szAnsi	
00000001	0000175C		0000		0001485F		VoidFunc	

All the DLL payloads have the same single entry point of VoidFunc.

This entry-point is hard-coded in PoshC2 and cannot be changed without hacking the compiled binary itself.

For the common executable and DLL payloads we've also added Yara rules for detecting them. These are based on signaturable parts of the binaries that will not change across different installs of PoshC2, for example with different comms options. These are also available in the new PoshC2 Detections GitHub repository.

In Summary

We've looked at a few different detections for catching PoshC2 when used out-of-the-box. Using these in your environment will help protect against the less sophisticated users of PoshC2 in addition to further understanding how the tool works.

Any new detections or amendments can be added to the

<u>https://github.com/nettitude/PoshC2_IOCs</u> repository and we encourage the community to add their own detections or rules and configurations for other security tools to help build a centralised data store for everyone.