FIREEYE

Network Evasion Now you see me, now you don't

James Anderson

What we're covering

- Network Evasion
 - Current State
 - Abusing HTTP
 - External Communications
 - Detection and Evasion

James Anderson

- Red Team Consultant at Fireeye
- Previously Work
 - Reverse Engineering
 - Security Engineer
- Big fan of Board Games, D&D and Hack the box challenges.





Evasion

"Subtle and insubstantial, the expert leaves no trace; divinely mysterious, he is inaudible. Thus he is the master of his enemy's fate."

~Sun Tzu, The Art of War

Network Evasion: bypass an information security device like a firewall or intrusion detection prevention system in order to deliver an exploit, attack or other form of malware to a target network or system without detection.

- To achieve network evasion a set of techniques, tools practices is used to conceal the true activity to human and automated resources
 - Obfuscation
 - Encryption
 - Stenography





7

Source: Juniper Networks

- A World with ubiquitous monitoring
 - Host based EDR products
 - Network Port spanning
 - Machine Learning
 - AMSI Group Policy Settings
 - NGFW



Easy to detect scan traffic (nmap)

- IDS/IPS can easily to detect
- Org's are getting better at detecting password spraying
- Multiple failed attempts may send you to a shunned portal

- Domain Fronting
 - Beacons through a high reputation cloud provider
 - Uses the host header that points to a subdomain entry that is a CDN entry to the actual server.



()

https://cobaltstrike.com

- Domain Fronting gave us the best method of external communications we could ever ask for.
- While it is still around the window is closing
- Lateral movement techniques are being signature.
- SSL decryption.



 Some techniques, ideas, OPSEC and resources of how to remain stealthy in a high security environment

Public Platforms

P2C2 (Public Platform Command and Control) CRUD (Create Read Update Destroy)

3





SCANNING

Port Scanning

Three things matter when you are looking at initial scanning

5

- Timing between requests
- Requests per IP
- Packet Fragmentation
- Slowing you scans down is a must
- Being able to manipulate your source IP's will help

Packet Fragmentation

- Many security devices rely on sessionized data
 - Initiate a connection (handshake)
 - Pass some data
 - Close the connection



6

Image source cisco

Packet Fragmentation

The MTU size of a link determines whether there is a need to fragment a datagram into smaller units or not



Packet Fragmentation

 Fragmented Packets to avoid session detection



Port Scanning

Nmap

 Timing options T1: Sneaky(waits 15 seconds) T0: Paranoid (waits 5 minutes) 9

- Packet Fragmentation (-f)
- <u>https://nmap.org.book.man-bypass-firerwalls-</u> <u>rds.html</u>

Port Scanning

- Proxycannon (from Shellntel) <u>https://github.com.Shellntel/scr</u> <u>ipts</u>
 - Can spin up 20 Amazon EC2 instances to proxy scans through
 - Can rotate public WAN IP of nodes



Proxycannon

ront@ip-172-31-40-33:~/proxycannon-mg/modes/aws#

 \cap



HTTP PIPELINING

HTTP Pipelining – DigiNinja

- In the early days each object requested by a client was done in its own TCP connection
- If a page had two images and one JavaScript library, then there would be four connections,
- Pipelining allows for multiple requests at the same time to occur
 - Part of HTTP/1.0, HTTP/1.1



https://digi.ninja/blog/pipelining.php

- Multiple HTTP requests are sent on a single TCP connection
 - Technique superseded by HTTP2
 - HTTP pipelining is not enabled by default in modern browsers
 - Support still exists in most servers and, more importantly, most CDNs.



https://digi.ninja/blog/pipelining.php

Lets start with an example request

GET /pipeline/page1.php HTTP/1.1
Host: vuln-demo.com

```
HTTP/1.1 200 OK
Date: Sat, 14 Sep 2019 03:48:34
GMT
Server: Apache
...
Content-Length: 14
```

- Two requests
 - Page1.php
 - Page2.php

GET /pipeline/page1.php HTTP/1.1
Host: vuln-demo.com

GET /pipeline/page2.php HTTP/1.1 Host: vuln-demo.com

 Burp Doesn't handle this type of request well. Injects content length header

- Send Carrol C +	Target: https://vuln-demo.com 🖉 🕧
Request	Response
Raw Parama Headers Hex	Raw Headers Hex Render
üff /pipeline/pagel.php HTTP/ll Hatt: vuln-demo.com GET /pipeline/page2.php HTTP/ll Hust: vuln-demo.com	<pre>* HTTP/1.1 200 0K * Date: 1wt, 14 Sep 2010 03:48:34 GMT * Strict Transport Security: max.ege=63072000 Upgrade: h2,h2c Connection: Upgrade Vary: Accept Excoding X-Content Type-Optione: nosniff Cache-Control: max.ege=0, no-cache, no-store, must-revulidate Prages: no cache Expires: Wed, 11 Jan 1004 05:00:00 GMT X-XSS-Protection: 0; Acceps-Control.Allew-Origin: * Acceps.control.Allew-Origin: * Acceps.control.Allew-Origin, X-Requested-With, Content-Type: Access-Control.Allew-Origin Content-Type: test/html; charset=UTF_8 Cantent-Length: 14 This is page 1</pre>
1 Type a search benn 0 m	atches 🕐 💌 + 💌 Type a search term O matches
Done	559 bytes 632 millin

These can both be sent, will be processed by the server, and then the responses sent back in order.

GET /pipeline/page1.php HTTP/1.1
Host: vuln-demo.com

GET /pipeline/page2.php HTTP/1.1
Host: vuln-demo.com



- Do you require the header: Connection: keep-alive ?
- "keep-alive" enables persistent connections which is a different thing to pipelining

9

- Persistent connections keep the TCP connection open between requests but enforce the original rule of waiting for any previous requests to return before making new ones
- In HTTP 1.0, persistence had to be activated with the "keep-alive" header, in HTTP 1.1, persistence is assumed unless a connection is requested to be closed with the Connection: close header.

Sending the two requests

(echo -e "GET /pipeline/page1.php HTTP/1.1\r\nHost: vuln-demo.com\r\n\r\nGET /pipeline/page2.php HTTP/1.1\r\nHost: vuln-demo.com\r\n\r\n"; sleep 5) | openssl s_client - connect vuln-demo.com

HTTP/1.1 200 OK Date: Fri, 08 Mar 2019 20:42:47 GMT Server: Apache Expires: Wed, 11 Jan 1984 05:00:00 GMT Access-Control-Allow-Origin: https://vuln-demo.com Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Access-Control-Allow-Origin Content-Length: 14 Keep-Alive: timeout=5, max=100 Content-Type: text/html; charset=UTF-8

This is page 1HTTP/1.1 200 OK Date: Fri, 08 Mar 2019 20:42:47 GMT Server: Apache Expires: Wed, 11 Jan 1984 05:00:00 GMT Access-Control-Allow-Origin: https://vuln-demo.com Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Access-Control-Allow-Origin Content-Length: 14 Content-Type: text/html; charset=UTF-8 This is page 2DONE

Sending the two requests

HTTP/1.1 200 OK Date: Fri, 08 Mar 2019 20:42:47 GMT Server: Apache Expires: Wed, 11 Jan 1984 05:00:00 GMT Access-Control-Allow-Origin: https://vuln-demo.com Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Access-Control-Allow-Origin Content-Length: 14 Keep-Alive: timeout=5, max=100 Content-Type: text/html; charset=UTF-8

This <mark>is</mark> page 1

HTTP/1.1 200 OK

Date: Fri, 08 Mar 2019 20:42:47 GMT

Server: Apache Expires: Wed, 11 Jan 1984 05:00:00 GMT Access-Control-Allow-Origin: https://vuln-demo.com Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Access-Control-Allow-Origin Content-Length: 14 Content-Type: text/html; charset=UTF-8 This is page 2DONE

Response 1

Response 2

Two requests, two responses

- Pipeline worked!

- Can we use it through a CDN?
 - AWS Cloudfront
 - Actual domain fastpackagedomain.com
 - Fronted Domain. d1sdh26o01490vk5.cloudfront.net



3 2

Via Cloudfront AWS

\$ (cat pipe2 ; sleep 5) | openssl s_client -connect fronted.fastpackagedomain:443 fronted.fastpackagedomain | grep "<title>"

depth=2 C = US, O = Amazon, CN = Amazon Root CA 1 verify return:1 depth=1 C = US, O = Amazon, OU = Server CA 1B, CN = Amazon verify return:1 depth=0 CN = fronted.fastpackagedomain verify return:1 <title>FastPackage - Delivered</title> <title>Eronted Vuln Demo</title>

DONE

- Could be used as a covert channel for communications
- Would require SSL decryption to interpret the payload.
- Some IDS/IPS devices might only parse the first request leaving the second request concealed.



We need to be QUIC-ER

QUIC

Wait... Firewall was blocking all TCP? 3

6
Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes
▼ Frame	100.0	9511	100.0	8361781	540 k	0	0
▼ Ethernet	100.0	9511	100.0	8361781	540 k	0	0
 Internet Protocol Version 6 	0.1	7	0.0	915	59	0	0
 User Datagram Protocol 	0.0	1	0.0	153	9	0	0
DHCPv6	0.0	1	0.0	153	9	1	153
Internet Control Message Protocol v6	0.1	6	0.0	762	49	6	762
 Internet Protocol Version 4 	99.7	9478	100.0	8359708	540 k	0	0
 User Datagram Protocol 	99.7	9478	100.0	8359708	540 k	0	0
Teredo IPv6 over UDP tunneling	0.1	6	0.0	762	49	0	0
QUIC (Quick UDP Internet Connections)	98.5	9365	99.7	8337858	538 k	9365	8337858
Domain Name System	1.1	107	0.3	21088	1362	107	21088
Address Resolution Protocol	0.3	32	0.0	1920	124	32	1920

3

7

CI.

```
User Datagram Protocol, Src Port: 63786 (63786), Dst Port: 443 (443)
~QUIC (Quick UDP Internet Connections)
>Public Flags: 0x0d
 CID: 1464692183167920367
 Version: Q030
 Sequence: 1
 Message Authentication Hash: 870608f6bf34b710f976e324
Private Flags: 0x00
STREAM (Special Frame Type) Stream ID:1, Type: CHLO (Client Hello)
 Frame Type: STREAM (Special Frame Type) (0xa0)
  Stream ID: 1
  Data Length: 1024
  Tag: CHLO (Client Hello)
  Tag Number: 27
  Padding: 0000
                                                                            .
```

What type of traffic is this?

Wireshark · Follow SSL Stream (tcp.stream eq 78) · wireshark_pcapng_eth0_20160720152239_p5EQBz	
PRI * HTTP/2.0	
SM	
<pre>/%. A.A.RKRVG. W. yc\$./QS. X.?.c.0. a.< "}</pre>	+<. 3 .gG1 mnt
9 # #811*) -0-(0%()/ C	



9

	90	00	00	07	3a	6d	65	74	68	6f	64	00	00	00	03	47	:met	hodG	
0910	45	54	00	00	09	0a	3a	61	75	74	68	6f	72	69	74	79	ET:a	uthority	
0820	88	88	60	10	77	77	77	2e	66	61	63	65	62	61	61	6b	WWW.	facebook	
0835	2e	63	6f	6d	09	00	89	97	3a	73	63	68	65	6d	65	88	.com	:scheme.	
0840	88	80	65	68	74	74	70	73	00	88	.00	05	3a	70	61	74	https	:pat	
0850	68	80	00	69	14	21	45	6c	65	63	74	72	69	63	Ze	42	h/E1	ectric.B	
6966	72	65	61	6b	66	61	73	74	2f	88	00	60	19	75	70	67	reakfast	/upg	
0070	72	61	64	65	2d	69	6e	73	65	63	75	72	65	2d	72	65	rade-ins	ecure-re	
0880	71	75	65	73	74	73	00	00	00	91	31	60	00	00	9a	75	quests	1u	
0090	73	65	72	2d	61	67	65	6e	74	88	00	00	85	4d	61	7a	ser-agen	tMoz	
09a0	69	6c	6c	61	2f	35	2e	30	20	28	58	31	31	3b	20	4c	illa/5.0	(X11; L	
0000	69	6e	75	78	20	78	38	36	51	36	34	29	20	41	70	70	inux x86	64) App	
090±0	6c	65	57	65	62	4b	69	74	2f	35	33	37	2e	33	36	20	leWebKit	/537.36	
0000	28	4b	48	54	4d	4c	2c	20	6c	69	6b	65	20	47	65	63	(KHTML,	like Gec	
0846	6b	6f	29	20	55	62	75	6e	74	75	20	43	68	72	6f	6d	ko) Ubun	tu Chrom	
8978	69	75	6d	21	35	31	2e	30	2e	32	37	30	34	2e	37	39	1um/51.0	.2704.79	
0100	20	43	68	72	6f	6d	65	2f	35	31	2e	30	2e	32	37	30	Chrome/	51.0.270	
0110	34	2e	37	39	20	53	61	66	61	72	69	21	35	33	37	2e	4.79 Saf	ar1/537.	
0120	33	36	00	00	00	06	61	63	63	65	70	74	88	00	99	4a	36ac	ceptJ	
0128	74	65	78	74	21	68	74	6d	6c	2c	61	70	78	6c	69	63	text/htm	1, applic	
0140	61	74	69	6f	6e	2f	78	68	74	6d	6c	2b	78	6d	6c	2c	ation/xh	tml+xml,	
0150	61	70	70	6c	69	63	61	74	69	61	6e	21	78	6d	6c	3b	applicat	ion/xml;	
G160	71	3d	30	2e	39	2c	69	6d	61	67	65	2f	77	65	62	70	q=0.9, im	age/webp	
0170	2c	2a	21	2a	3b	71	3d	30	2e	38	00	00	88	01	61	63	,*/*;q=0	.8ac	
0188	63	65	70	74	2d	65	6e	63	6f	64	69	6e	67	00	00	00	cept-enc	oding	
9198	17	67	7a	69	79	2c	20	64	65	66	6c	61	74	65	2c	20	.gzip, d	eflate,	
@1a6	73	64	63	68	2c	20	62	72	00	88	88	0f	61	63	63	65	sdch, br	acce	
0106	70	74	20	6c	61	6e	67	75	61	67	65	00	00	00	0e	65	pt-langu	agee	
01c0	6e	2d	55	53	2c	65	6e	3b	71	3đ	30	2e	38				n-US,en;	q=0.8	
Frame	e (3	63 E	yte	s)	0	ecr	ypt	ed S	SL d	ata	(26	8 by	ytes	6)	De	ecom	pressed Hea	ader (461 byte	5

• What's going on here?

4

- Quick UDP Internet Connections (QUIC) is a new protocol created by Google to make the web faster and more efficient
 - Enabled by default in Chromium and used by a growth list of sites



	Http1/1	Http1/1	Http1/1	Http1/1	
HTTP/2	HTTP/2	HTTP/2	HTTP/2	HTTP/2	
Connection	Stream	Stream	Stream	Stream	
QUIC TLS	QUIC	QUIC	QUIC	QUIC	
Connection Session	Stream	Stream	Stream	Stream	
UDP	QUIC	QUIC	QUIC	QUIC	
Connection	Packet	Packet	Packet	Packet	

CI.



Source: cloudflare

- Unlike the TCP protocol, QUIC requires 0-RTT in the handshake compared to 1-3 roundtrip TCP + TLS trips
- This ensures security for anyone using the protocol
- Invalidates the possibility of a man-in-the-middle attack
 - A lot of inspection mechanisms don't support QUIC Protocol.

- Quick Example of using QUIC
 - Most stable version of QUIC is written in go by Lucas Clemente
 - <u>https://github.com/lucas-clemente/quic-go</u>

QUIC HTTP/3 - SERVER

package main

import (

"fmt"

"io/ioutil"

"log"

"time"

"net/http"

"github.com/lucas-clemente/quic-go/h2quic" "github.com/lucas-clemente/quic-go/internal/protocol" quic "github.com/lucas-clemente/quic-go"

type Page struct { Title string Body []byte

func (p *Page) save() error {
 filename := p.Title + ".txt"
 return ioutil.WriteFile(filename, p.Body, 0600)

func loadPage(title string) (*Page, error) {
 filename := title + ".txt"
 body, err := ioutil.ReadFile(filename)
 if err != nil {
 return nil, err

return & Page{Title: title, Body: body}, nil

func viewHandler(w http.ResponseWriter, r *http.Request) {
 title := r.URL.Path[len("/view/"):]
 p, _ := loadPage(title)
 fmt.Fprintf(w, "<h1>%s</h1><div>%s</div>", p.Title, p.Body)

func main() {
versions := protocol.SupportedVersions
http.HandleFunc("/view/", viewHandler)
server := h2quic.Server{
 Server: &http.Server{Addr: ":443"},
QuicConfig: &quic.Config{Versions: versions, IdleTimeout: 30000 * time.Millisecond},

log.Fatal(server.ListenAndServeTLS("fullchain.pem", "privkey.pem"))

QUIC HTTP/3 - CLIENT

package main

nport (
"bytes"
"flag"
"fmt"
"io"
"net/htt
العامية ال

quic "github.com/lucas-clemente/quic-go" "github.com/lucas-clemente/quic-go/h2quic" "github.com/lucas-clemente/quic-go/internal/protocol"

func main() {

urls := flag.String("url", "https://127.0.0.1:443/", "URL")
flag.Parse()

versions := protocol.SupportedVersions
roundTripper := &h2quic.RoundTripper{
 QuicConfig: &quic.Config{Versions: versions, IdleTimeout: 30000 * time.Millisecond},

defer roundTripper.Close()
hclient := &http.Client{
 Transport: roundTripper,

rsp, err := hclient.Get(*urls)
rsp.Header.Add("User-Agent", "UnkL4b")
if err != nil {
 panic(err)

body := &bytes.Buffer{}
_, err = io.Copy(body, rsp.Body)
if err != nil {
 panic(err)
}
fmt.Printf("%s", body.Bytes())

2605:6001:e7c... QUIC

2607:f8b0:400... OUIC

94 443

101 64178

443

64178

						Ethernet: anD								
	0	AL		000	11									
	ALC: NO. TO.	-	- Inthe Intel		come :									
quie							i creation and a second						El - Lerner	4 C.
n a Time Time delta basitité	10.000.000.000.000	Destination	Pretocel	Length - Source Port	Destrution	Pert CID	PET a QUE	Ster MAX STRMS	- orfo		1210-010		And the second second	
73 2,076150 0,043715000 26051	filte:40031c8011	258516881	1400 OUTC	98 443	61757	15345671032972174127	23844		Payload (Encrypted),	PKN1 2	2014	5877651977174127	
104 8.590301 0.514151000 2005:	60011e7c917500:-	260711850	:488. QUIC	251 64178	443	13898777283338313855	159		Paylant (Encrypted),	PHN 1	59, CID: 138	00777283330313095	
105 8.601401 8.011100000 2005:	6001:e7c9:7500:_	2687:1808	:408_ QUIC	258 64178	443	13890777283338313895	168		Payload (Encrypted),	PNN: 1	60, CID: 130	00777283330313095	
105 8,614305 8.012905000 2687:	f8b8:4003:c87::-	2685:6881	ie7c. QUIC	135 443	64178		8449		Payload (Encrypted),	PIONI B	449		
107 8.614836 8.000530000 2607: 188 8.614636 8.000530000 2605.	780014803100711.	200510001	ierc. QUIC	79 443	64178	11050777202110311055	8705		Payload (Encrypted),	PROVI B	705 83 / 735- 138	00777261110111005	
109 8.643428 8.028492008 2087:	7858:4803:087::-	2005:6001	te7c. QUIC	04 443	64178	13030111203330313035	8961		Payload (Encrypted),	PION: B	961	10111203330323033	
110 8.762638 8.119210008 2685:	6001:07c9:7500:	2687: 1858	:488. OUIC	101 64178	443	13000377283338313005	162		Payload /	Fer runted)	and in the	65 PTD: 130	007772833389111005	
Destination	Protocol	Length	Source Port	Destinatio	on Port	CID		PKT #	QUIC Ve	r MAX S	STRM	S Info	<u> </u>	_
Destination 2607: f8b0: 400.	Protocol . QUIC	Length 85	Source Port 61752	Destinatio	on Port	CID 13345871632972	174127	PKT #	QUIC Ve	r MAX !	STRMS	S Info Paylo	ad (Encryp	ted)
Destination 2607: f8b0: 400. 2605: 6001: e7c.	Protocol . QUIC . QUIC	Length 85 98	Source Port 61752 443	Destinatio 443 61752	on Port	CID 13345871632972	174127	PKT # 249 23044	QUIC Ve	er MAX :	STRMS	S Info Paylo Paylo	ad (Encryp ad (Encryp	ted), ted),
Destination 2607: f8b0: 400. 2605: 6001: e7c. 2607: f8b0: 400.	Protocol . QUIC . QUIC . QUIC	Length 85 98 251	Source Port 61752 443 64178	Destinatio 443 61752 443	on Port	CID 13345871632972 13090777283330	1 74127 313095	PKT # 249 23044 159	QUIC Ve	r MAX :	STRMS	S Info Paylo Paylo Paylo	ad (Encryp ad (Encryp ad (Encryp	ted), ted),
Destination 2607:f8b0:400. 2605:6001:e7c 2607:f8b0:400 2607:f8b0:400	Protocol . QUIC . QUIC . QUIC . QUIC . QUIC	Length 85 98 251 250	Source Port 61752 443 64178 64178	Destinatio 443 61752 443 443	on Port	CID 13345871632972 13090777283330 13090777283330	174127 313095 313095	PKT # 249 23044 159 160	QUIC Ve	r MAX :	STRMS	S Info Paylo Paylo Paylo Paylo	ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp	ted), ted), ted),
Destination 2607: f8b0: 400. 2605: 6001: e7c. 2607: f8b0: 400. 2607: f8b0: 400. 2605: 6001: e7c.	Protocol QUIC QUIC QUIC QUIC QUIC QUIC	Length 85 98 251 250 135	Source Port 61752 443 64178 64178 443	Destinatio 443 61752 443 443 64178	on Port	CID 13345871632972 13090777283330 13090777283330	174127 313095 313095	PKT # 249 23044 159 160 8449	QUIC Ve	r MAX :	STRMS	S Info Paylo Paylo Paylo Paylo Paylo	ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp	ted), ted), ted), ted),
Destination 2607: f8b0: 400. 2605: 6001: e7c. 2607: f8b0: 400. 2607: f8b0: 400. 2605: 6001: e7c. 2605: 6001: e7c.	Protocol QUIC QUIC QUIC QUIC QUIC QUIC QUIC	Length 85 98 251 250 135 79	Source Port 61752 443 64178 64178 443 443	Destinatio 443 61752 443 443 64178 64178	on Port	CID 13345871632972 13090777283330 13090777283330	174127 313095 313095	PKT # 249 23044 159 160 8449 8705	QUIC Ve	r MAX S	STRMS	S Info Paylo Paylo Paylo Paylo Paylo Paylo	ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp ad (Encryp	ted), ted), ted), ted), ted),

13090777283330313095

8961

162

Payload (Encrypted),

Pavload (Encrypted).

- Frame 70: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0
- Ethernet II, Src: Apple_69:2d:cf (3c:07:54:69:2d:cf), Dst: HonHaiPr_dd:c3:7c (70:77:81:dd:c3:7c)
- Internet Protocol Version 6, Src: 2605:6001:e7c9:7500:4d66:8f11:bfc7:77d9, Dst: 2607:f8b0:4003:c00::bd
- User Datagram Protocol, Src Port: 61752, Dst Port: 443
- QUIC (Quick UDP Internet Connections)
 - Public Flags: 0x0c
 -0 = Version: No
 -0. = Reset: No
 - 11.. = CID Length: 8 Bytes (0x3)
 - ..00 = Packet Number Length: 1 Byte (0x0)
 - .0.. = Multipath: No
 - 0... = Reserved: 0x0

CID: 13345871632972174127

Packet Number: 249

Payload: 5f97bce069fcb3280197e6d8d4

Because the QUIC transport stream does not allow Firewall to perform a deep packet inspection, there is an impact in both reporting and network security that allows attackers to abuse the protocol and avoid detection of malicious actions just changing the **version** in the public header.

QUIC Structure



QUIC Public Header

Public Flags (8 bits)	Connection ID (64 bits)
	QUIC version (32 bit) (Optional)
Div	ersification nonce (32 bytes) (Optional)
Pa	cket Number (8, 16, 32 or 48)

5 2

QUIC Version

- The QUIC specification reserves from 0x00000001 to 0x0000ffff for standardized versions of the protocol
- How some IDS interpret a connection with the QUIC protocol
- What If we change the header in our comms

Version	Owner	Notes
0x00000000	n/a	This value is reserved as invalid
02222222	IETE	Values meeting this pattern ((x&0x0f0f0f0f)==0x0a0a0a0a) are reserved for ensuring that version negotiation remains
	Drivato	vidble.
0x50435130		Picoauic internal test version
0x5130303[1-9]	Google	Google QUIC 01 - 09 (Q001 - Q009)
0x5130313[0-9]	Google	Google QUIC 10 - 19 (Q010 - Q019)
0x5130323[0-9]	Google	Google QUIC 20 - 29 (Q020 - Q029)
0x5130333[0-9]	Google	Google QUIC 30 - 39 (Q030 - Q039)
0x5130343[0-9]	Google	Google QUIC 40 - 49 (Q040 - Q049)
0x51474f[0-255]	quic-go	"QGO" + [0-255]
0x91c170[0-255]	quicly	"qicly0" + [0-255]
0xabcd000[0-f]	Microsoft	WinQuic
0xf10000[00-ff]	IETF	QUIC-LB
0xf123f0c[0-f]	Mozilla	MozQuic
	Faceboo	
0xfaceb00[0-f]	k	mvfst
0xff000001	IETF	draft-ietf-quic-transport-01
0xff000002	IETF	draft-ietf-quic-transport-02
0xff000003	IETF	draft-ietf-quic-transport-03
0xff000004	IETF	draft-ietf-quic-transport-04
0xff000005	IETF	draft-ietf-quic-transport-05
0xff000006	IETF	draft-ietf-quic-transport-06
0xff000007	IETF	draft-ietf-quic-transport-07
0xff000008	IETF	draft-ietf-quic-transport-08
0xff000009	IETF	draft-ietf-quic-transport-09
0xff00000a	IETF	draft-ietf-quic-transport-10
0xff00000b	IETF	draft-ietf-quic-transport-11
0xf0f0f0f[0-f]	ETH Zürich	Measurability experiments

UnkL4b - unkl4b.github.io

 The tests consist of blocking the QUIC protocol in Fortinet AppControl and running the client to close communication with a server in the cloud that is accepting only the protocol in the Q309 version

0x5130333[0-9] Google Google QUIC 30 - 39 (Q030 - Q039)

Fortigate logs in Splunk

Blocked when identified as QUIC

12.000 PM	Apr 1 18:10:12 date=2019- ventlime=1554153011 appid=40169 srcip= tgoing* policyid=37 sessionid= 5	-04-01 time=10.10:11 devname=""" de «172.25.48.17 dstip= 8 srcport=5657 \$7 applist="VLAN" appcat="Network.Serv	<pre>wid=" 06" logid=" "I dstport=443 srcintf=" sce" app="QUIC" action="block"</pre>	705" type="utm" subtype="ap 'srcintfrole="lan" dstintf="port3 incidentserialno="721 esg="	p-ctrl" eventtype="app-ctrl-all" level="a " dstintfrole="wan" proto=17 service="udp Wetwork.Service: QUIC," apprisk="low"
	heat - Technol & America - fortinet American	hype = factinet			
				0.00	
-01	time=18:10:11 devnam	me=" " devid	=" stport=443 srcin	06" logid="1	705" type="utm" sub
-01 2.20 app:	time=18:10:11 devnam 6.48.17 dstip= list="VLAN" a	me=" devid 8 srcport=56571 d appcat="Network.Service	=" stport=443 srcir " app="QUIC" act	06" logid="1 htf=" " src: ion="block" incid	705" type="utm" sub intfrole="lan" dstint dentserialno= 7

5 5

Passed with the header version change

1	Time	Event
3	4/1/19 9:11:27.000 PM	Apr 1 18:11:27 date=2019-04-01 time=18:11:27 devname=" " devide" 06" logid="00000000011" type="forward" level="marning" vd="root" eventtime=1554153087 src ip=172:25:48:17 arcport=35433 srcintfrumknom=-0 srcintfrumknom=-0 srcintfru="winde" is disport=443 distintf="port" distintf="marning" vd="root" eventtime=1554153087 src otor17 action="ip=come" policyid=7 poli
		e=2019-04-01 time=18:11:27 devname=""devid="06" logid="000000001 rcintf=unknown-0 srcintfrole="undefined" d tip d=37 policytype="policy" service="udp/443" appcat="unscanned" (rscore=5 craction=262144 crl e:36" srcserver=0
		sourcetype = fortinet

J 56 ©2019 FireEye

- New/Unique Protocols are harder for IDS/IPS to track
- For those that to have signatures, changing the public header can effectively mask it again.

HTTP2

- Merlin is a cross-platform post-exploitation HTTP/2 Command & Control server
 - https://github.com/Ne0nd0g/merlin





Traffic Normalization IDS/IPS and Machine Learning

Normalizing Traffic

- Signature based NIDS
 - Look for pre-defined patters of previously known attacks
 - Doesn't require training phase
 - Highly available and popular
 - Can't catch new attacks

6 0

Normalizing Traffic

- Bypassing this form of signature
 - Not hard but not super easy
 - Change traffic elements
 - Don't match with any signatures

- Build a statistical model describing the normal network traffic and flagging the abnormal traffic
- Requires training phase
- Uses math, machine learning and some more sophisticated methods
- Might catch on previously unseen activity.





63 ©2019 FireEye

- Evasion
 - Pre-Training

6

4

- Post-Training



- Pre Training
 - Sending Malicious requests to the system.

11 Internet of Shit Retweeted



Computer Facts @computerfact

concerned parent: if all your friends jumped off a bridge would you follow them? machine learning algorithm: yes.

2:20 PM · Mar 15, 2018

V

DARKTRACECISCO



- Machine Learning Algorithms
 - Supervised machine learning algorithms
 - Can apply what has been learned in the past to predict future events using labelled examples.

6

- Unsupervised machine learning algorithms:
 - Used when the information used to train is neither marked nor classified.
- Semi-supervised machine learning algorithms:
 - Makes use of unlabelled data for training with a blend of less labelled data and a lot of unlabelled data.

- Basic features of individual TCP connections

Feature Name	Description	Туре
duration	length (number of seconds) of the connection	continuous
protocol type	type of the protocol (tcp, udp)	discrete
service	network service of the destination. (http, telnet ssh)	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
	1 if connection is from/to the same host port; 0	
land	otherwise	discrete
wrong fragment	number of "wrong" fragments	continuous
urgent	number of urgent packets	continuous

Common Machine Learning Algorithms

6 9

- K means clustering
- Bayes Network
- Random Forest Classifier
- Multi-Layer Perceptron (MLP)

Machine Learning Algorithms

 $Accuracy_{ave} = \frac{TP + TN}{TP + FN + FP + TN}$

- No model is perfect
- We want to sit in the slim but possible False negative category

- KDD Cup Data
 - Labelled as normal or as an attack
 - 4898431 instances with 41 attributes Table VI. Average Accuracy Rate

Machine Learning Classifiers	Correctly classified Instances	Incorrectly classified Instances	Accuracy Rate
j48	55865	4135	93.10%
Random Forest	56265	3735	93.77%
Random tree	55345	5655	90.57%
Decision table	55464	4536	92.44%
MLP	55141	4859	91.90%
Naive Bayes	54741	5259	91.23%
Bayes Network	54439	5561	90.73%

- How do we evade detection?
 - Normalize your behaviour so you look like everything else
 - Avoid similar methods of connection that ML would use as a characteristic.


Polymorphic Blanding Attack

 Polymorphic Blending attack: Creating attack packets which match to a normal traffic profile



STANDARD TRAFFIC

http Smtp

Blending

- Get traffic capture data of traffic and define normal behaviour of users
 - Which User agents are most common
 - Which ports are used, what kind of server headers are there?
- Alter comms channel to reflect the same pattern
- What hosts is this machine communicating
- <u>https://github.com/tearsecurity/first</u> order



Blending in

Sometimes your life depends on it

Blending

=== Top 10 Port Statistics ===

Port 443: 1677/5937 (28.25%) Port 58471: 1107/5937 (18.65%) Port 80: 536/5937 (9.03%) Port 58457: 454/5937 (7.65%) Port 54674: 341/5937 (5.74%) Port 57859: 228/5937 (3.84%) Port 54119: 157/5937 (2.64%) Port 58408: 155/5937 (2.61%) Port 53: 124/5937 (2.09%) Port 58403: 80/5937 (1.35%)

=== Top 10 Server Headers ===

Server: PWS/8.3.1.0.4: 9/36 (25.00%) Server: RocketCache/2.2: 5/36 (13.89%) Server: nginx: 5/36 (13.89%) Server: NetDNA-cache/2.2: 4/36 (11.11%) Server: None: 3/36 (8.33%) Server: nginx/1.8.1: 2/36 (5.56%) Server: cafe: 1/36 (2.78%) Server: Microsoft-IIS/7.5: 1/36 (2.78%) Server: Cloudflare-nginx: 1/36 (2.78%) Server: Microsoft-IIS/10.0: 1/36 (2.78%)

=== Top 10 User-Agent Headers ===

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36: 29/32 (90.62%) User-Agent: Google Chrome/63.0.3239.84 Mac OS X: 3/32 (9.38%)

Blending

- Talk to hosts on the same protocol
- Don't talk to hosts that it that this host isn't talking to







Machine Learning

- How do we evade detection?
 - Normalize your behaviour so you look like everything else
 - Don't use the same methods of connection that ML would use as a characteristic.



The quieter you are the more you hear Listen for the packets you want

Listening for packets

Can we listen for packets

- Does require local admin
- Direct traffic to a wrong node and collect the traffic

 Windows starting in XP cannot send data on a raw socket but it can listen 8

 \cap

Alternatively, you may use WinPcap drivers



All analysis is based off the idea of hosts talking to each other



- What if instead we sent to a random machine in the network and just listened on a raw socket?
- Listening for traffic that is deliberately sent to the wrong host





The wrong host is likely to drop the packets

- Invalid port
- Wasn't expecting data.



Finally mask the traffic similar to how previous packets of data have appeared being sent to the host.



- To send tasking the controller needs to transmit tasking encapsulated in valid TCP network traffic
- Typically requires connecting to an (open) listening port on the victim
- Send raw packets, skips the triple handshake.



REDSALT

 A loader that decodes a second-stage loader that decodes another payload containing a backdoor capable of listening for commands using a raw socket or obtaining commands from a URL or file.



The Future?

The game keeps changing and you have to change with it

Wrapping up

Defences are improving.

- Defence in depth continually adds layers



Wrapping up

 Blend in so defenders can't distinguish between you an attacker and legitimate activity 8

9

- Machine learning

- Has a margin of error, we need to sit under it.
- Other Technologies not covered
 - WCF
 - Packet Stuffing
 - Serverless Computing



Thank you

"Be extremely subtle even to the point of formlessness. Be extremely mysterious even to the point of soundlessness. Thereby you can be the director of the opponent's fate."

~Sun Tzu, The Art of War





Packet Fragmentation is back?

To Encrypt or not Encrypt?



Markov Obfuscation

Cylance Spear Team

