

Compression Oracle Attacks on VPN Networks

Nafeez
Defcon 26



About

Nafeez - @sketpic_fx

Interested in AppSec and writing software

Maker @ assetwatch.io, Attacker Surface Discovery as a Service

Overview

Compression Side Channel and Encryption

History of attacks

VPNs and how they use compression

Demo - Voracle Tool

How to find if your "VPN" is vulnerable

Way forward

Data Compression

LZ77

Replace redundant patterns

102 Characters

Everything looked dark and bleak, everything looked gloomy, and everything was under a blanket of mist

89 Characters

Everything looked dark and bleak, (-34,18)gloomy, and (-54,11)was under a blanket of mist

Data Compression

Huffman Coding

Replace frequent bytes with shorter codes

Char ⇄	Freq ⇄	Code ⇄
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010

Data Compression

DEFLATE - LZ77 + Huffman Coding

ZLIB, GZIP are well known DEFLATE libraries

Compression Side Channel

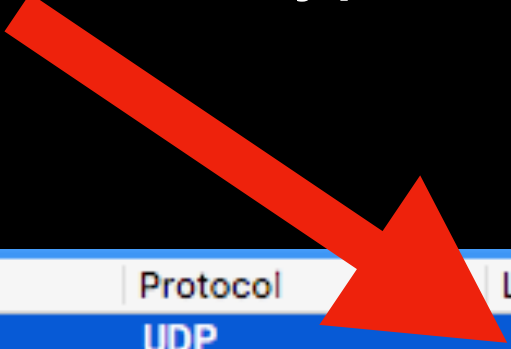
First known research in 2002

**Compression and Information Leakage of
Plaintext**

John Kelsey, Certicom

The Side Channel

Length of encrypted payloads



Destination	Protocol	Length	Info
162.243.9.106	UDP	118	54452 → 443 Len=76
162.243.9.106	UDP	123	54452 → 443 Len=81
162.243.9.106	ISAKMP	158	IKE_AUTH MID=02 Initiator
162.243.9.106	UDP	119	54452 → 443 Len=77

Compression Oracle

Chosen Plain Text Attack

Brute force the secret byte by byte

Force a compression between the chosen byte and the existing bytes in the secret

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=**1**

Cookie: secret=637193 -some-data- (-34,15)**1**

Encrypted Payload Length = 43

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=2

Cookie: secret=637193 -some-data- (-34,15)2

Encrypted Payload Length = 43

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=3

Cookie: secret=637193 -some-data- (-34,15)3

Encrypted Payload Length = 43

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=4

Cookie: secret=637193 -some-data- (-34,15)4

Encrypted Payload Length = 43

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=5

Cookie: secret=637193 -some-data- (-34,15)5

Encrypted Payload Length = 43

Compression Oracle

Cookie: secret=637193 -some-data- Cookie: secret=6

Cookie: secret=637193 -some-data- (-34,16)

Encrypted Payload Length = 42

**How can we convert this
into an attack using
browsers?**

Back in 2012

The CRIME attack



Ingredients

Attacker on the data path can sniff packet length

Browser attaches cookies as part of any cross-domain request

Attacker controls HTTP request body

You get!

Chosen Plain Text attack using browsers

TIME Attack 2013

Tal Be'ery, Amichai Shulman

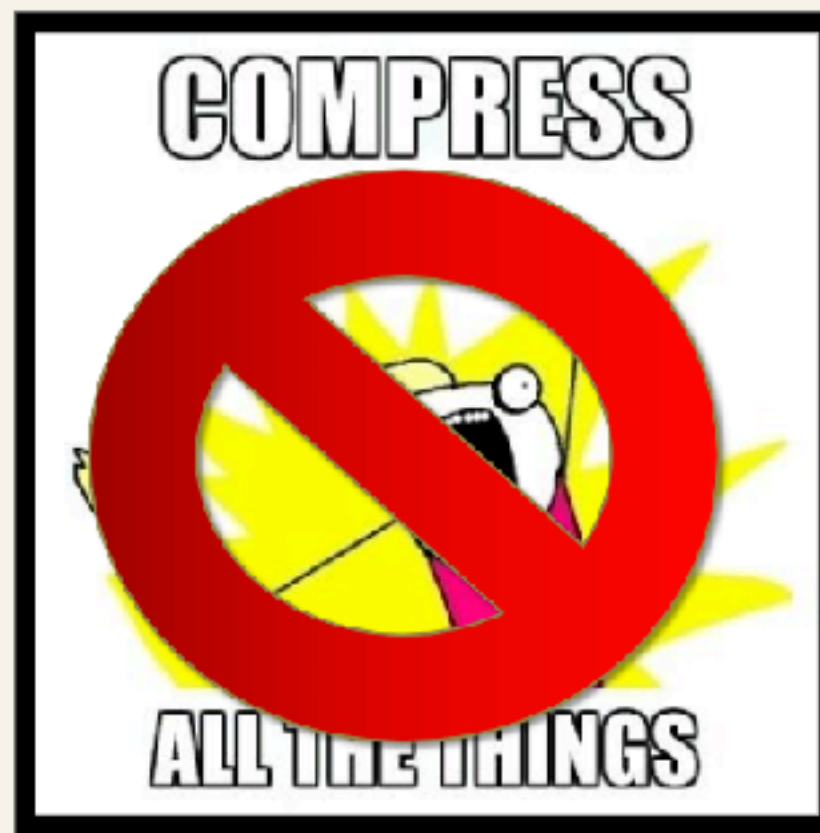
Timing side channel purely via browsers

Extending CRIME to HTTP Responses

BREACH Attack 2013

Angelo Prado, Neal Harris, Yoel Gluck

**A CRIME AGAINST THE
RESPONSE BODY**



breach
SSL, GONE IN 30 SECONDS

blackhat
USA 2013

BreachAttack.com

So far

CRIME style attacks have been mostly targeted on HTTPS

Researchers have possibly explored all possible side channels to efficiently leak sensitive data

Lets talk VPNs

TLS VPNS

IPSEC

L2TP/ PPTP

**TLS VPNs are pretty
common these days**



Hotspot Shield



PUREVPN

TunnelBear



ExpressVPN



privateinternetaccess

**What do most of these
SaaS VPNs have in
common?**



Compression

**Almost all VPNs support
compression by default**

OpenVPN Client Configuration (*.OVPN)

```
remote-cert-tls server
```

```
#mute 10000
```

```
auth-user-pass
```



```
comp-lzo
```

```
verb 3
```

```
pull
```

```
fast-io
```

```
cipher AES-256-CBC
```

```
auth SHA512
```

```
<ca>
```

```
-----BEGIN CERTIFICATE-----
```

```
MIIExDCCA6ygAwIBAgIJAPyaiSxcR5IvMA0GCSqGSI
```

OpenVPN Compression Algorithms

LZO

LZ4

-LZ77 Family-

High level overview

Authentication & Key Negotiation (Control Channel)

Data Channel Encryption

High level overview

Authentication & Key Negotiation (Control Channel)

Data Channel Compression

Data Channel Encryption

Compress everything

UDP

TCP

Bi-Directional

We have a **compress** then
encrypt on all of data channel

CRIME + BREACH on VPN Networks

**Existing TLS channel are
safe**

Things are safe, if the underlying
app layer already uses HTTPS / TLS.

 Secure | <https://www.google.com>

```
ssh user@website.com
```

Things might go bad, if the VPN is helping you to encrypt already encrypted data

ⓘ Not Secure www.bbc.com

DNS	74	Standard query 0x4ddc
DNS	74	Standard query 0xc3a7

ⓘ Not Secure corporate-network.internal.net

Lets see how this attack works on an
HTTP website using an **encrypted VPN**

Given a **HTTP Website** through VPN, Can
we leak **Sensitive Cookie Data** from a
Cross-Domain Website?

Ingredients

- VPN Server and Client has **compression** turned on by default
- VPN User using a **vulnerable browser**
- Visits **attacker controlled website**

Vulnerable Browser?

Yes, the browser plays a huge role in how it sends plain HTTP requests.

**Browser needs to send HTTP
requests in single TCP Data Packet**



Google Chrome splits HTTP packets into Header and Body

So we can't get the compression window in the same request

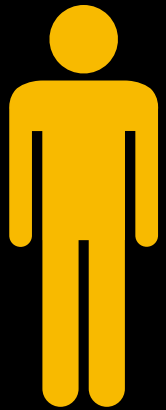


Mozilla Firefox sends them all
in a single TCP data packet

Now we get the compression
window in the same request

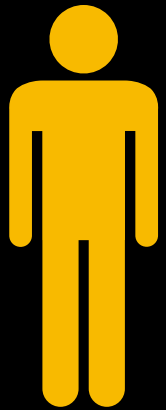
Attack Setup

VPN User



Attack Setup

VPN User

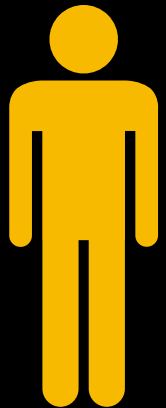


Vulnerable Browser



Attack Setup

VPN User



HTTP WebApp



Vulnerable Browser

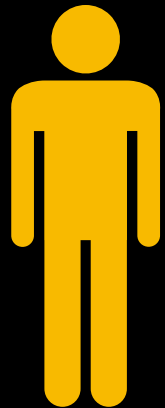


Attack Setup

Trusted VPN with Compression



VPN User



HTTP WebApp



Vulnerable Browser



Attack Setup

Trusted VPN with Compression

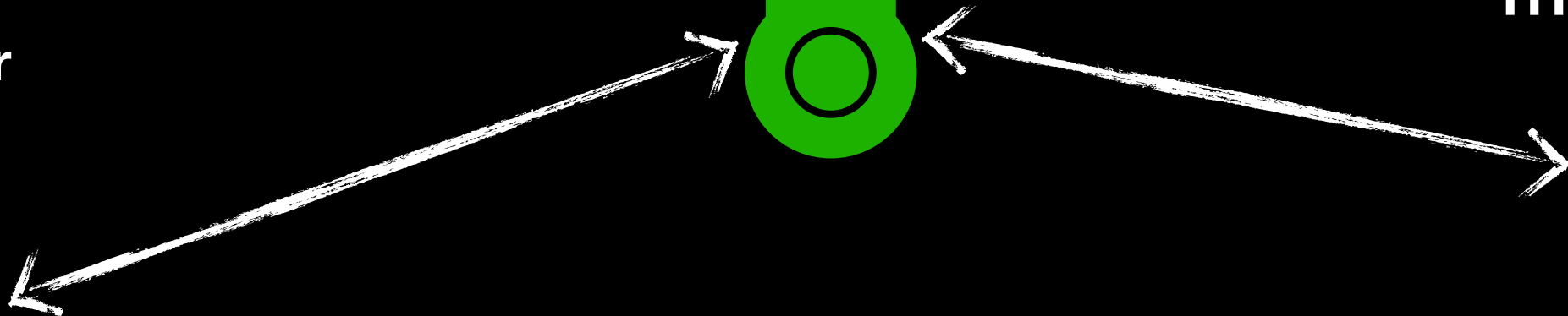
VPN User



HTTP WebApp



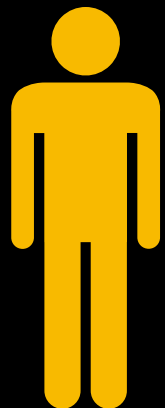
Vulnerable Browser



Attack Setup

Trusted VPN with Compression

VPN User



HTTP WebApp



Vulnerable Browser

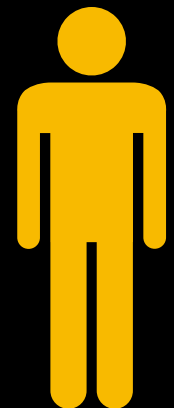


Attacker

Attack Setup

Trusted VPN with Compression

VPN User



HTTP WebApp



Vulnerable Browser



attacker.com

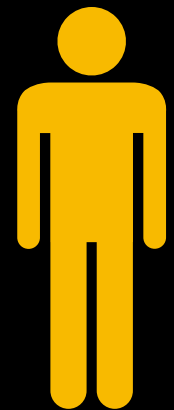


Attacker

Attack Setup

Trusted VPN with Compression

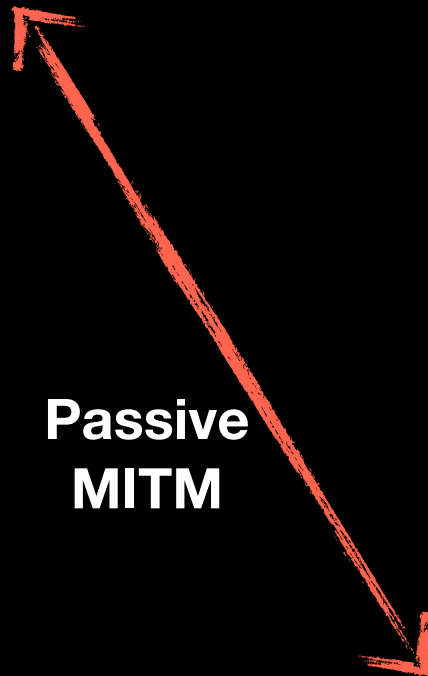
VPN User



HTTP WebApp



Passive
MITM



attacker.com Attacker



Vulnerable Browser



Attack Setup

Trusted VPN with Compression

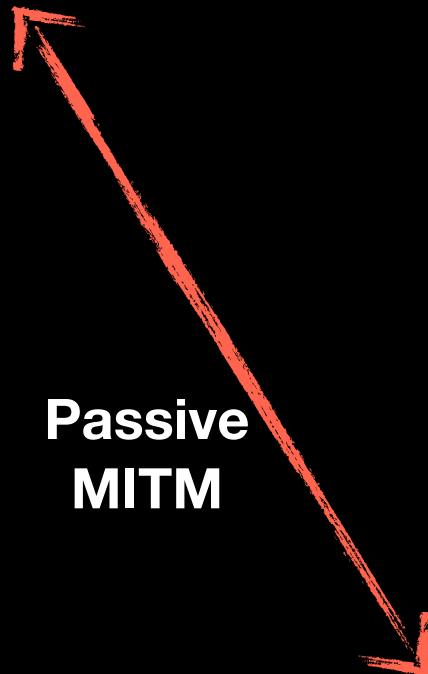
VPN User



HTTP WebApp



Passive
MITM



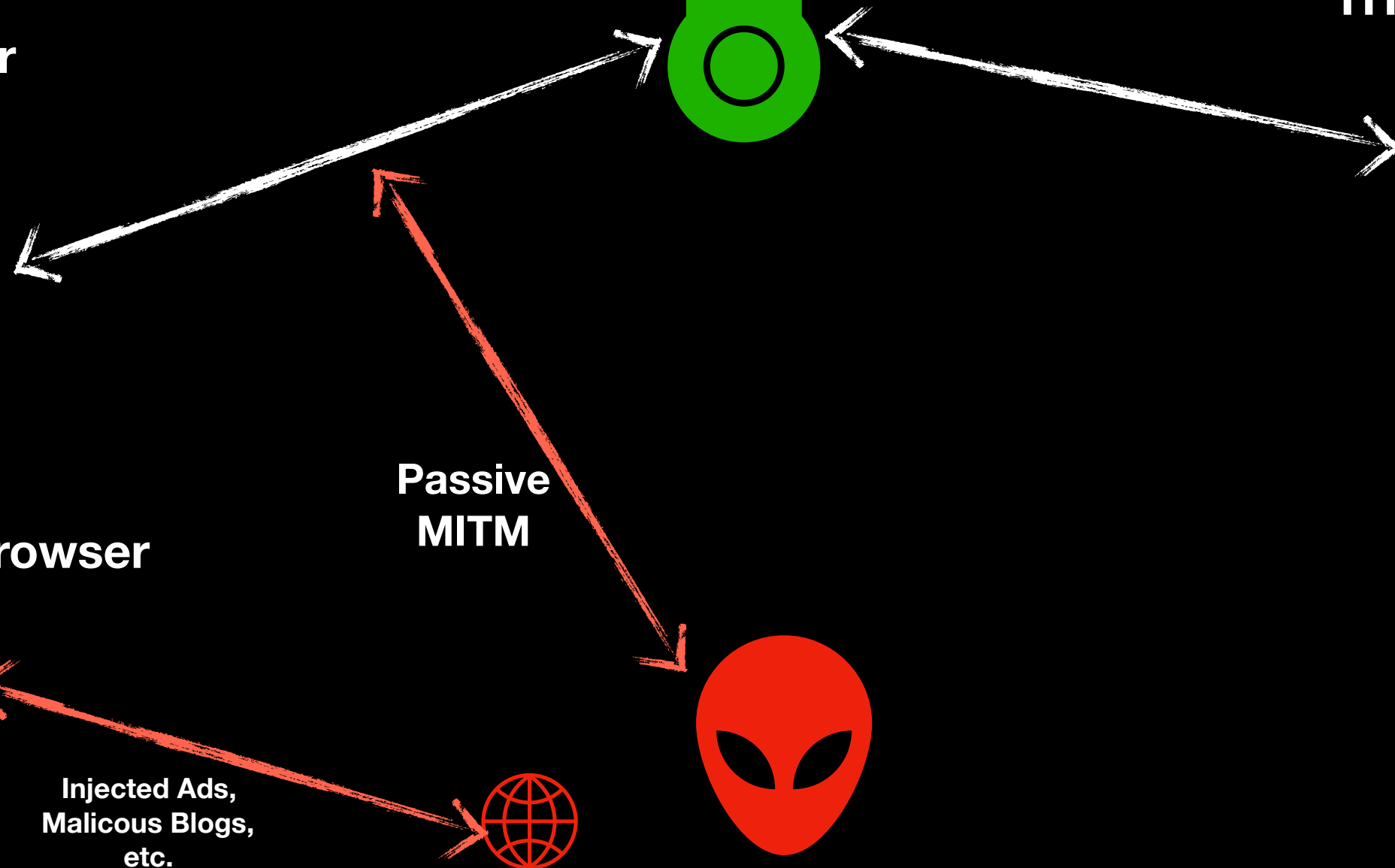
Vulnerable Browser



Injected Ads,
Malicious Blogs,
etc.



attacker.com Attacker



Attack Setup

Trusted VPN with Compression

VPN User



HTTP WebApp



Passive
MITM

Can Observe VPN
Data packet Lengths



attacker.com Attacker

Vulnerable Browser



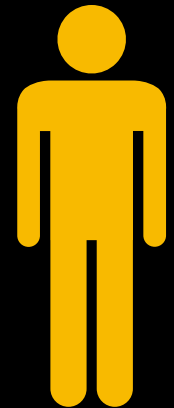
Injected Ads,
Malicious Blogs,
etc.



Attack Setup

Trusted VPN with Compression

VPN User



HTTP WebApp



Passive
MITM

Can Observe VPN
Data packet Lengths

Can Send Cross
Domain requests to
the HTTP WebApp

Injected Ads,
Malicious Blogs,
etc.

attacker.com Attacker



Vulnerable Browser

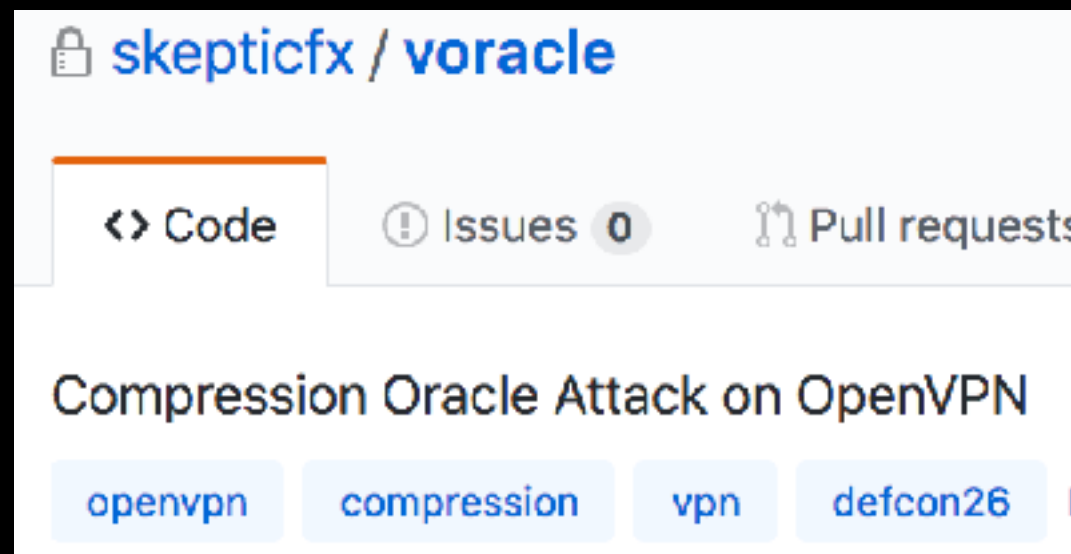


Attacker can now conduct CRIME Style attacks on HTTP requests and responses



Demo

Voracle



<https://github.com/skepticfx/voracle>

**How to tell if your VPN is
vulnerable?**

Ingredients

Wireshark

Terminal with Curl

Connected to your VPN under test

Observe VPN Payload Length

Curl and Observe Length

```
curl -s -o /dev/null -X POST http://website.com  
-d "--some-data-- Secret=37346282;  
--blah-- Secret=1 Secret=1"
```

Length = x

Curl and Observe Length

```
curl -s -o /dev/null -X POST http://website.com  
-d "--some-data-- Secret=37346282;  
--blah-- Secret=2 Secret=2"
```

Length = x

Curl and Observe Length

```
curl -s -o /dev/null -X POST http://website.com  
-d "--some-data-- Secret=37346282;  
--blah-- Secret=3 Secret=3"
```

Length = x-1

Curl and Observe Length

```
curl -s -o /dev/null -X POST http://website.com  
-d "--some-data-- Secret=37346282;  
--blah-- Secret=1 Secret=1"
```

Length = x

Fix?

**Fixing Compression is an
interesting problem**

Selectively disable Compression

- HPACK in HTTP2

Remember when **SPDY** was
vulnerable to CRIME?

HPACK selectively disabled header
compression for sensitive fields

HPACK: Header Compression for HTTP/2

draft-ietf-httpbis-header-compression-latest

7.1.3 Never-Indexed Literals

Implementations can also choose to protect sensitive header fields by not compressing them and instead encoding their value as literals.

<https://http2.github.io/http2-spec/compression.html>

**For VPNs, Disable
compression completely for
any plain text transactions**

Turning compression
off by default is
opinionated.

OpenVPN chose to warn
the implementors more
explicitly to turn off data
Compression.

<https://github.com/OpenVPN/openvpn/commit/a59fd147>

OpenVPN / openvpn

<> Code

Pull requests 32

Projects 0

Insights

man: add security considerations to --compress section

As Ahamed Nafeez reported to the OpenVPN security team, we did not sufficiently inform our users about the risks of combining encryption and compression. This patch adds a "Security Considerations" paragraph to the --compress section of the manpage to point the risks out to our users.

Signed-off-by: Steffan Karger <steffan@karger.me>


Acked-by: Gert Doering <gert@greenie.muc.de>

Message-Id: <1528020718-12721-1-git-send-email-steffan@karger.me>

URL: <https://www.mail-archive.com/openvpn-devel@lists.sourceforge.net/msg16919.html>

Signed-off-by: Gert Doering <gert@greenie.muc.de>

master

 syzz authored and cron2 committed on Jun 3



TunnelBear

turned off compression entirely



TunnelBear

Hi,

Thanks for the report.

As discussed via email, we have now removed compression support on our OpenVPN servers.
Would you be able to verify that your attack is no longer possible with the TunnelBear client?

Thanks

**Its time, everything
moves to HTTPS**

Takeaway

If you are using VPNs to access plain text websites over the internet, its time to move them to HTTPs.

Most corporates using VPN still allow plain text HTTP websites, because they think VPN protects them.

Thank you!



@skeptical_fx

