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

<table>
<thead>
<tr>
<th>Char</th>
<th>Freq</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>7</td>
<td>111</td>
</tr>
<tr>
<td>a</td>
<td>4</td>
<td>010</td>
</tr>
<tr>
<td>e</td>
<td>4</td>
<td>000</td>
</tr>
<tr>
<td>f</td>
<td>3</td>
<td>1101</td>
</tr>
<tr>
<td>h</td>
<td>2</td>
<td>1010</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
<td>1000</td>
</tr>
<tr>
<td>m</td>
<td>2</td>
<td>0111</td>
</tr>
<tr>
<td>n</td>
<td>2</td>
<td>0010</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>162.243.9.106</td>
<td>UDP</td>
<td>118</td>
<td>54452 → 443 Len=76</td>
</tr>
<tr>
<td>162.243.9.106</td>
<td>UDP</td>
<td>123</td>
<td>54452 → 443 Len=81</td>
</tr>
<tr>
<td>162.243.9.106</td>
<td>ISAKMP</td>
<td>158</td>
<td>IKE_AUTH MID=02 Initiator</td>
</tr>
<tr>
<td>162.243.9.106</td>
<td>UDP</td>
<td>119</td>
<td>54452 → 443 Len=77</td>
</tr>
</tbody>
</table>
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

A new TLS show was aired today

starring Juliano Rizzo and Thai Duong
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
Timing side channel purely via browsers

Extending CRIME to HTTP Responses
A CRIME AGAINST THE RESPONSE BODY

COMPRESS

ALL THE THINGS
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.
What do most of these SaaS VPNs have in common?
Compression
Almost all VPNs support compression by default.
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-----
MIIExDCCA6ygAwIBAgIJAPlAyISxCcR5IvMA0GCSqGSI
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.
Things might go bad, if the VPN is helping you to encrypt already encrypted data.

Not Secure: www.bbc.com

<table>
<thead>
<tr>
<th>DNS</th>
<th>74 Standard query 0x4ddc</th>
</tr>
</thead>
</table>

Not Secure: corporate-network.internal.net
Let's 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

Vulnerable Browser

HTTP WebApp
Attack Setup

- VPN User
- Vulnerable Browser
- Trusted VPN with Compression
- HTTP WebApp
Attack Setup

Trust VPN with Compression

VPN User

Vulnerable Browser

attacker.com

HTTP WebApp

Attacker
Attack Setup

- VPN User
- Vulnerable Browser
- Attacker
- HTTP WebApp

Trusted VPN with Compression
Passive MITM

attacker.com
Attacker
Attack Setup

Trusted VPN with Compression

VPN User

Passive MITM

Can Observe VPN Data packet Lengths

HTTP WebApp

Vulnerable Browser

Attacker

attacker.com

Injected Ads, Malicious Blogs, etc.
Attack Setup

- **VPN User**
- **Vulnerable Browser**
- **Trusted VPN with Compression**
- **Attacker**
- **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**
- attacker.com

---

**Can Observe VPN Data packet Lengths**
Attacker can now conduct CRIME Style attacks on HTTP requests and responses
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
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
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.

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.
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 <stefan@karger.me>
Acked-by: Gert Doering <gert@greenie.muc.de>
Message-Id: <1528020718-12721-1-git-send-email-stefan@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

syzzer authored and cron2 committed on Jun 3
turned off compression entirely

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, it's time to move them to HTTPS.

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

@skeptic_fx