Compression Oracle
Attacks on VPN Networks
- Nafeez
Nafeezz

AppSec research, static analysis tools, writing code

Maker @ assetwatch.io - Simple & Transparent Attack Surface Discovery

@sketpic_fx
Overview

Compression Side Channel and Encryption

History of attacks

VPNs and how they use compression

Voracle attack

How to find if your "VPN" is vulnerable

Way forward
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>

https://en.wikipedia.org/wiki/Huffman_coding
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>
Plain Text Data

Compress

Encrypt

Encrypted Data + Data Length
Plain Text Data

Add Attacker Controlled Bytes

Encrypt

Encrypted Data + Data Length

Compress
Plain Text Data

Add Attacker Controlled Bytes

Compress

Encrypt

Observe Encrypted Traffic

Encrypted Data + Data Length
Compression Oracle Attack

- Chosen Plain Text Attack
- Brute force the secret byte by byte
- Force a compression using the chosen byte and the existing bytes in the secret
secret=637193-some-app-data;

Compress

Encrypt

secret=637193-some-app-data; secret=1

Data Length

Encrypted Length = 30

secret=1
secret=637193-some-app-data;

Data Length

Compress

Encrypt

secret=637193-some-app-data; secret=3

secret=3

Encoded Length = 30
secret=637193-some-app-data;

Compress

Encrypt

Data Length

secret=637193-some-app-data;secret=5

secret=5

Encrypted Length = 30
secret=637193-some-app-data;

secret=6

Compress

Encrypt

Data Length

Encrypted Length = 29

Compression increased by 1 byte

More Compression, Smaller Length
How can we convert this into a real world attack on browsers?
Plain Text Data

Add Attacker Controlled Bytes

Compress

Encrypt

Observe Encrypted Traffic

Encrypted Data + Data Length
Add Attacker Controlled Bytes

Browser Sends Cross-Domain requests with Cookies attached

Attacker can send simple HTTP POST requests cross-domain with his own data

Observe Encrypted Traffic

MITM. People do this all the time
Back in 2012

Juliano Rizzo, Thai Duong

The CRIME attack

A new TLS show was aired today

starring Juliano Rizzo and Thai Duong
"We believe"

- TLS compression may resurrect in the near future
  - "Browsers are not the only TLS clients!"

- HTTP gzip may be a bigger problem than both SPDY and TLS compression
  - If you control the network, then a XSRF token is as good as, if not better, a session cookie.

- Remember: compression is everywhere.
TIME Attack 2013

Tal Be'ery, Amichai Shulman

Timing side channel purely via browsers, using TCP window sizes.

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

There are more - HEIST, Practical Developments to BREACH
So, what's new today?
VPN Tunnels
TLS VPNs are pretty common these days
What do most of these SaaS VPNs have in common?
OpenVPN
High level overview

Authentication & Key Negotiation (Control Channel)

Data Channel Compression

Data Channel Encryption
Compress everything

UDP

TCP

Bi-Directional
OpenVPN Compression Algorithms

LZO

LZ4

-LZ77 Family-
We have a **compress** then **encrypt** on all of data channel
VORACLE Attack
Under a VPN, HTTP WebApps are still insecure!
Things are safe, if the underlying app layer already uses an encryption channel.
Things **might go bad**, if the VPN tunnel is helping you encrypt already non-encrypted data.
Let's see how this attack works on an HTTP website using an encrypted VPN.
Requirements

- VPN Server and Client has Compression enabled by default
- Attacker can observe VPN traffic
- VPN User visits attacker.com
Attack Setup

VPN User
Attack Setup

VPN User

Browser
Attack Setup

VPN User

Browser

HTTP WebApp
Attack Setup

Trusted VPN with Compression

VPN User

Browser

HTTP WebApp
Attack Setup

Trusted VPN with Compression

VPN User

Browser

HTTP WebApp
Attack Setup

Trusted VPN with Compression

VPN User

Browser

Attacker.com

Attacker

Passive MITM

HTTP WebApp
Attack Setup

Trusted VPN with Compression

VPN User

Browser

Passive MITM

Injected Ads, Malicious Blogs, etc.

Attacker.com

Attacker

HTTP WebApp
Attacker Setup

Trusted VPN with Compression

VPN User

Browser

Passive MITM

Can Observe VPN Data packet Lengths

Injected Ads, Malicious Blogs, etc.

attacker.com

Attacker

HTTP WebApp
Attack Setup

VPN User

Browser

Trusted VPN with Compression

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

HTTP WebApp
Attacker can now conduct Compression Oracle attacks on HTTP requests and responses
<table>
<thead>
<tr>
<th><strong>Browser</strong></th>
<th>Mozilla Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VPN Client</strong></td>
<td><a href="https://github.com/OpenVPN/openvpn3">https://github.com/OpenVPN/openvpn3</a></td>
</tr>
<tr>
<td><strong>VPN Server</strong></td>
<td>OpenVPN Server</td>
</tr>
<tr>
<td><strong>WebApp</strong></td>
<td><a href="http://insecure.skepticfx.com">http://insecure.skepticfx.com</a></td>
</tr>
<tr>
<td><strong>Attack Goal</strong></td>
<td>Steal <code>sessionId</code> cookie from a cross-domain website</td>
</tr>
</tbody>
</table>
Voracle

https://github.com/skepticfx/voracle
Attack Challenges

- No Server Name Indication (SNI) or TLS certificates.
- VPN traffic is too chatty. Everything goes through it.

Hard to determine attacker's own traffic.
Also

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.
Detecting Voracle in your VPN
If your VPN provider is using OpenVPN - take a look at your client configuration.
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-----
MIIExDCCA6ygAwIBAgIJAyJAPyaiSxcR51vMAOGCSqGSI
Or you can test this dynamically by triggering compression and observing the length
DIY Voracle Detection

- Fire up Wireshark
- Connect to your VPN under test
- Send a few Curl requests with compression
- 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  ✓  More Compression, Smaller Length
Fix?
Fixing Compression is an interesting problem
Remember when SPDY was vulnerable to CRIME?
HPACK in HTTP/2 selectively disables 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 entirely for all plain text transactions
Turning compression off by default is opinionated.
OpenVPN chose to warn the implementors more explicitly to turn off data Compression.

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>
Acknowledged-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>

syzzer authored and cron2 committed on Jun 3

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

**EndUsers & Website owners** - If you are using VPN to access plain text websites over the internet, it's time to move them to HTTPs.

**VPN Providers** - Explicitly state what your VPN protects against. If you are claiming your VPN tunnel protects against plain text web apps, ensure you do not compress them.
Thank you!

@skeptic_fx

nafeez@assetwatch.io