black hat USA 2018

AUGUST 4-9, 2018

MANDALAY BAY / LAS VEGAS

Compression Oracle Attacks on VPN Networks

- Nafeez

Nafeez

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

Data Compression

LZ77

Replace redundant patterns

102 Characters

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

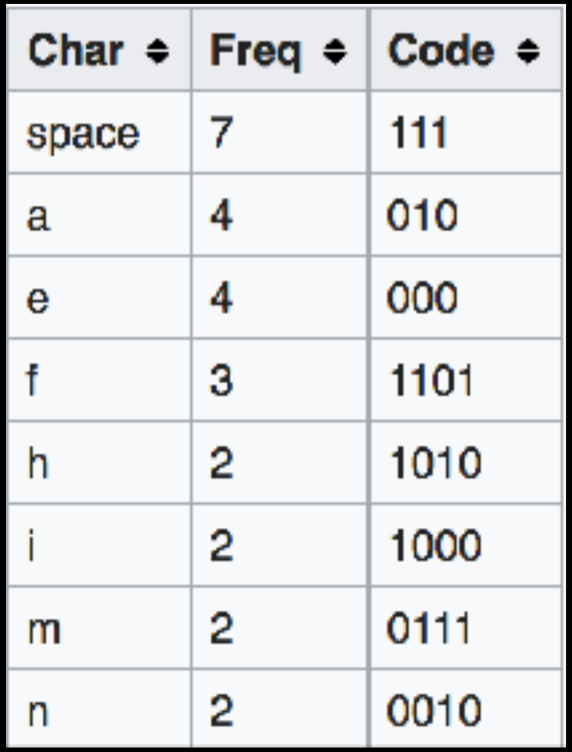
89 Characters

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

https://en.wikipedia.org/wiki/Huffman_coding

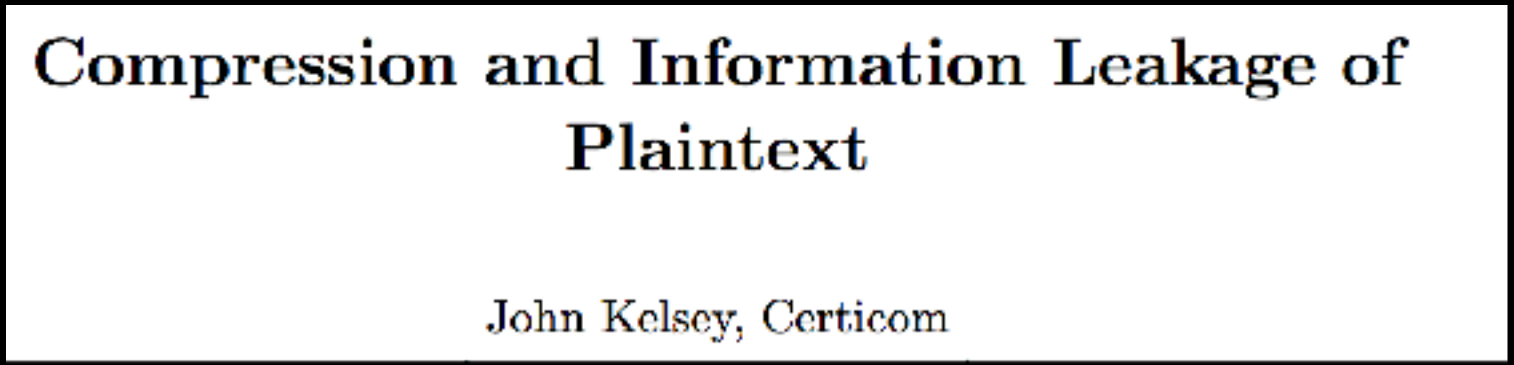
Data Compression

DEFLATE - LZ77 + Huffman Coding

ZLIB, GZIP are well known DEFLATE libraries

Compression Side Channel

First known research in 2002

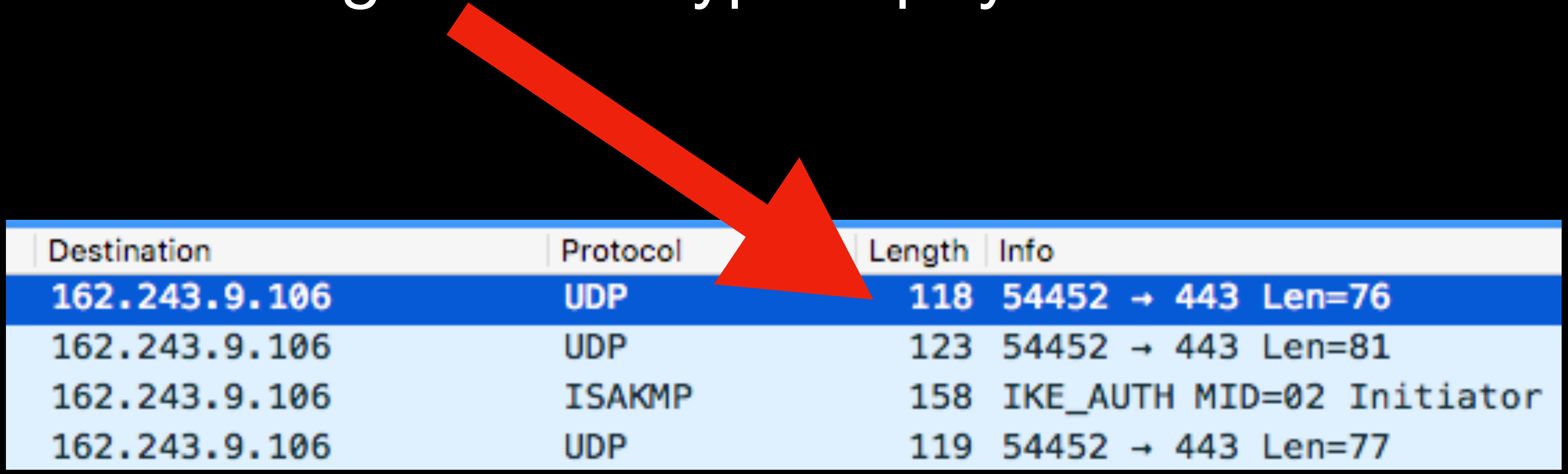


Compression and Information Leakage of Plaintext

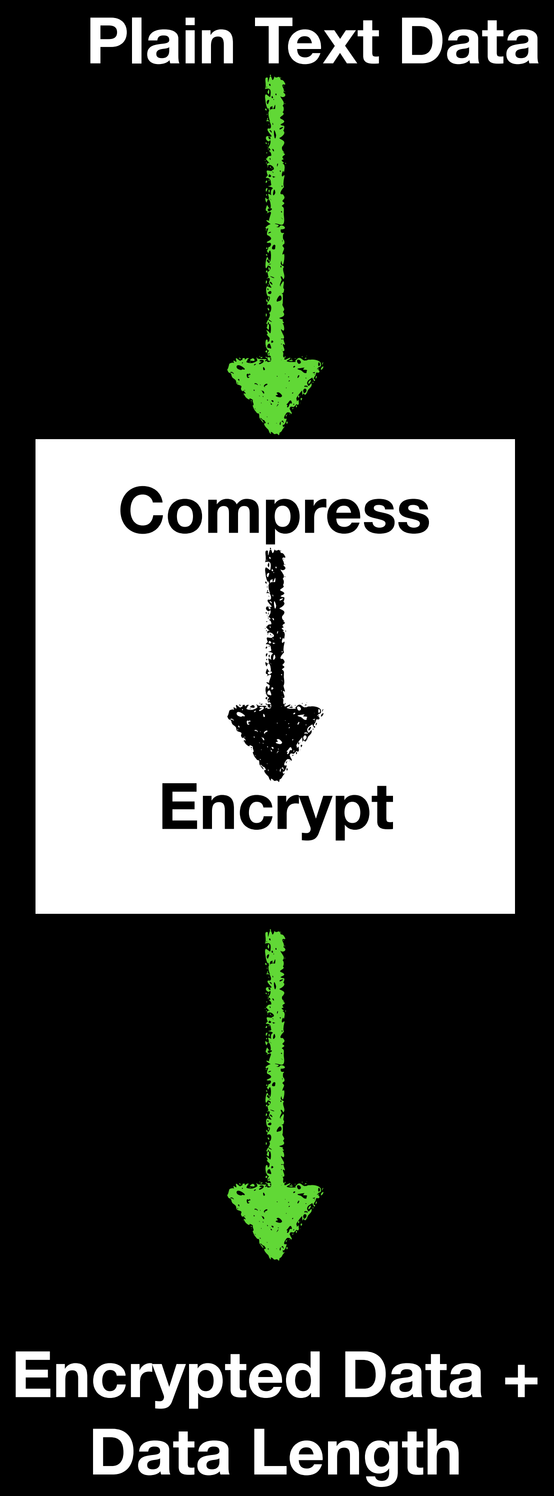
John Kelsey, Certicom

The Side Channel

Length of encrypted payloads



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

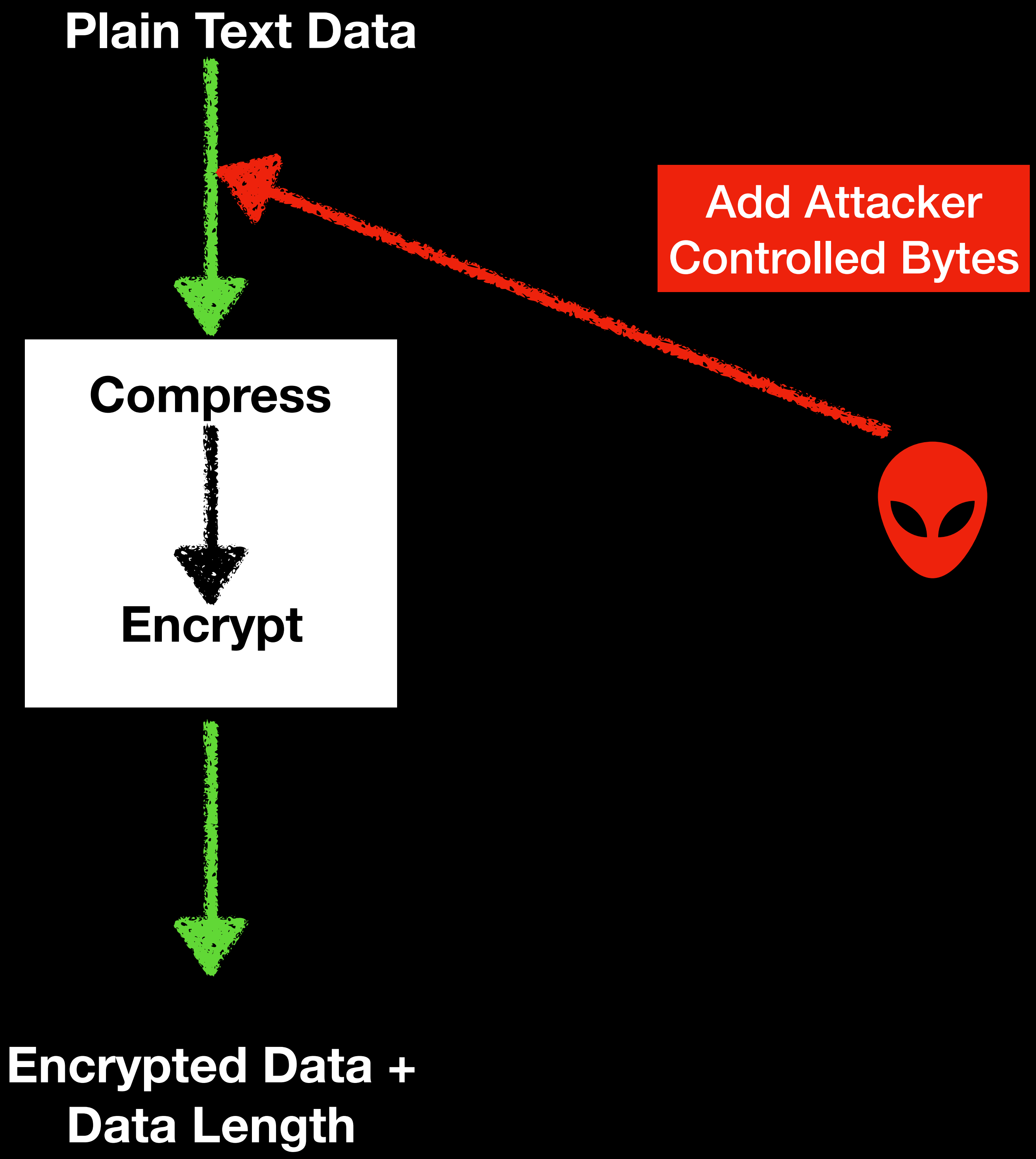


Plain Text Data

Compress

Encrypt

Encrypted Data +
Data Length



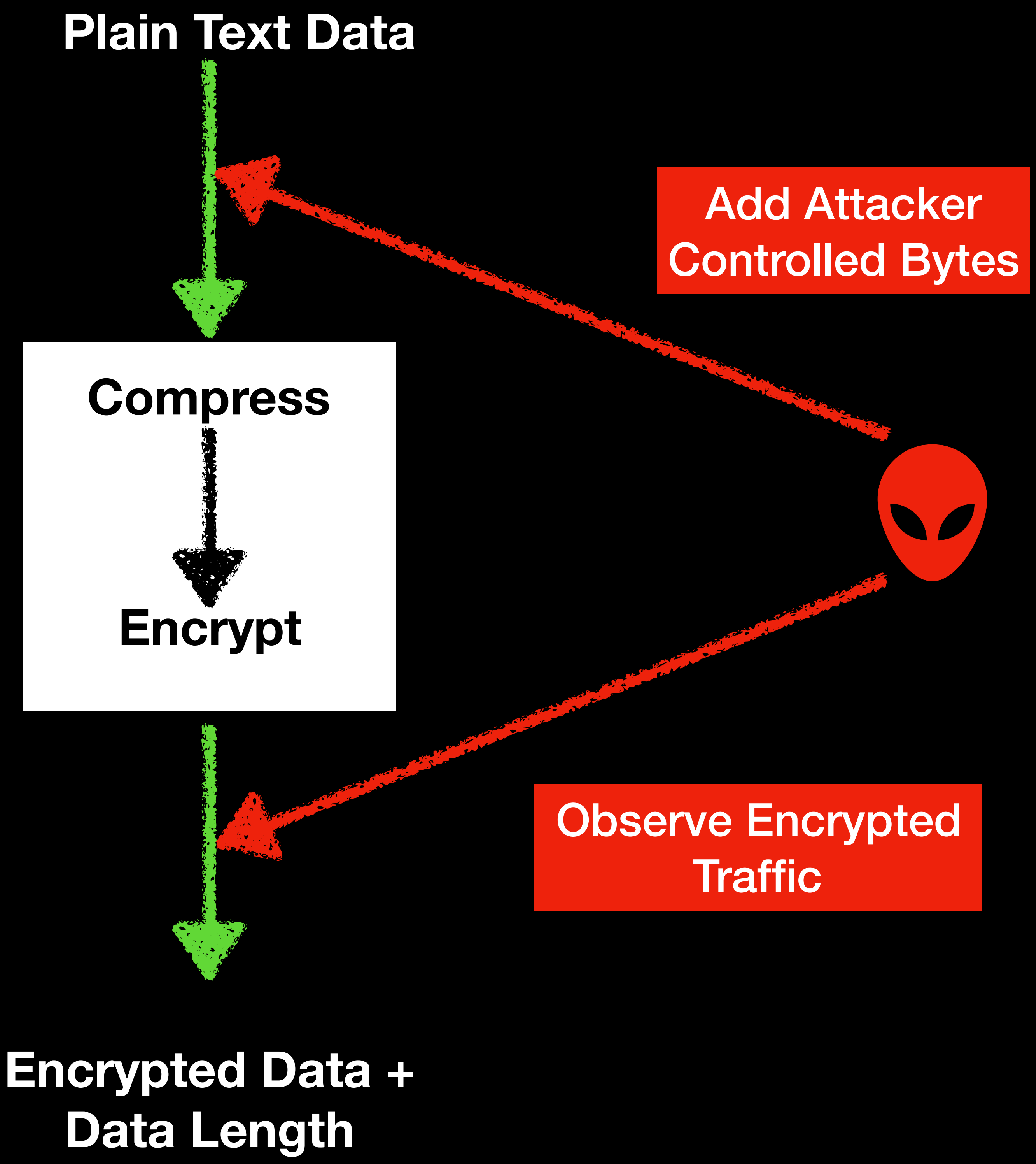
Plain Text Data

Add Attacker Controlled Bytes

Compress

Encrypt

Encrypted Data + Data Length



Plain Text Data

Add Attacker Controlled Bytes

Compress

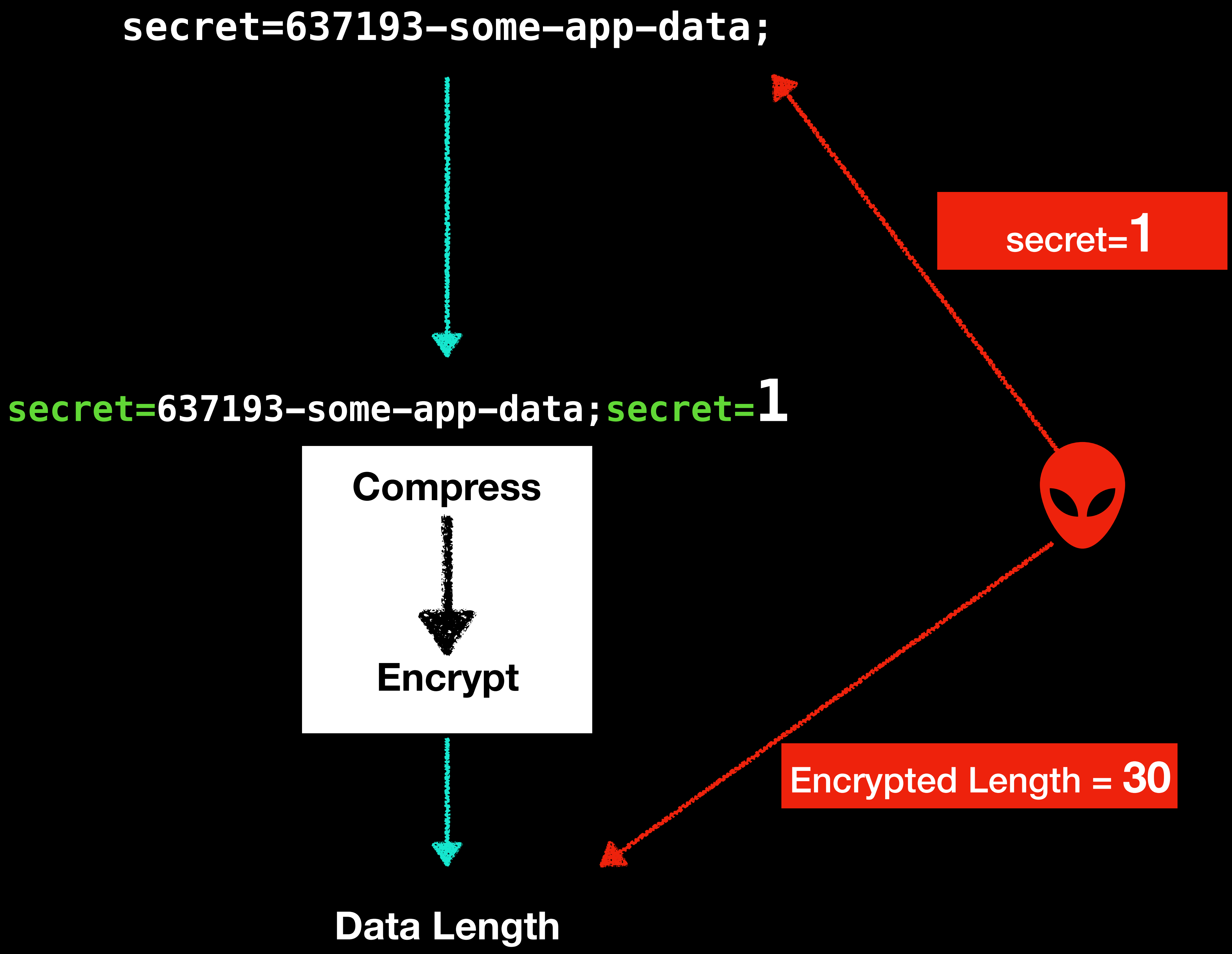
Encrypt

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;`

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

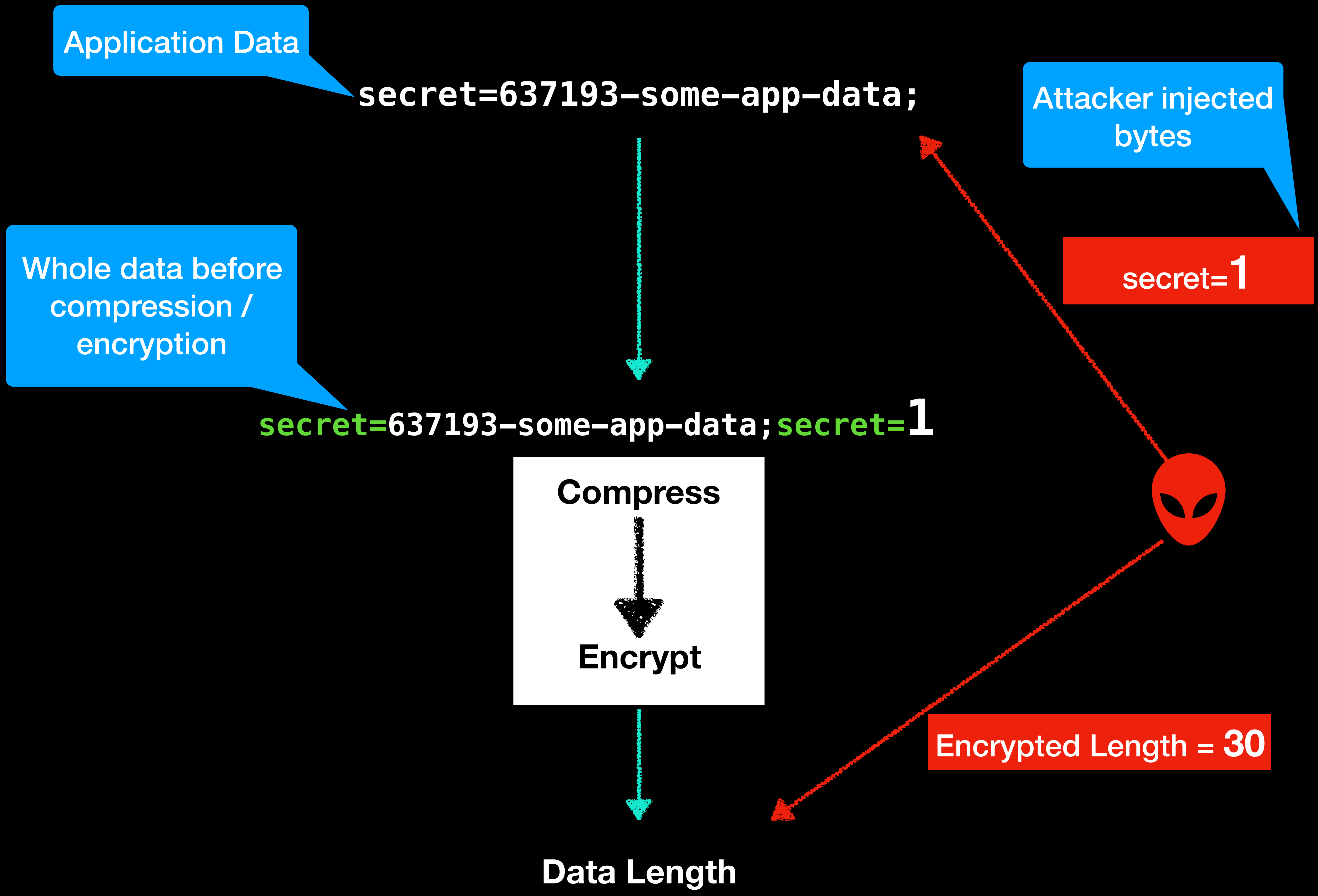
Compress

Encrypt

Data Length

secret=1

Encrypted Length = 30



Application Data

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

Attacker injected bytes

secret=1

Whole data before compression / encryption

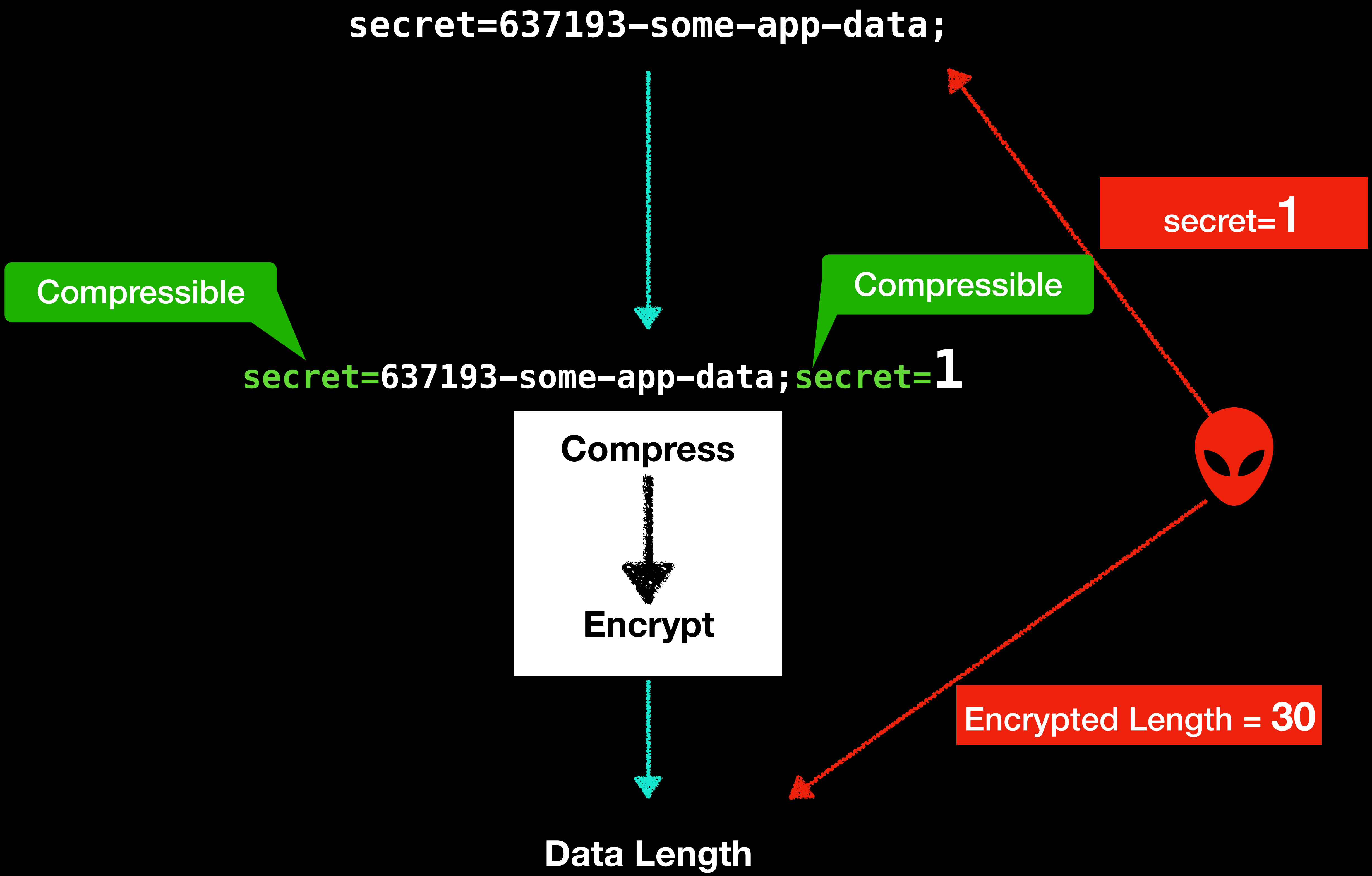
`secret=637193-some-app-data;secret=1`

Compress

Encrypt

Encrypted Length = 30

Data Length



secret=637193-some-app-data;

Compressible

Compressible

secret=1

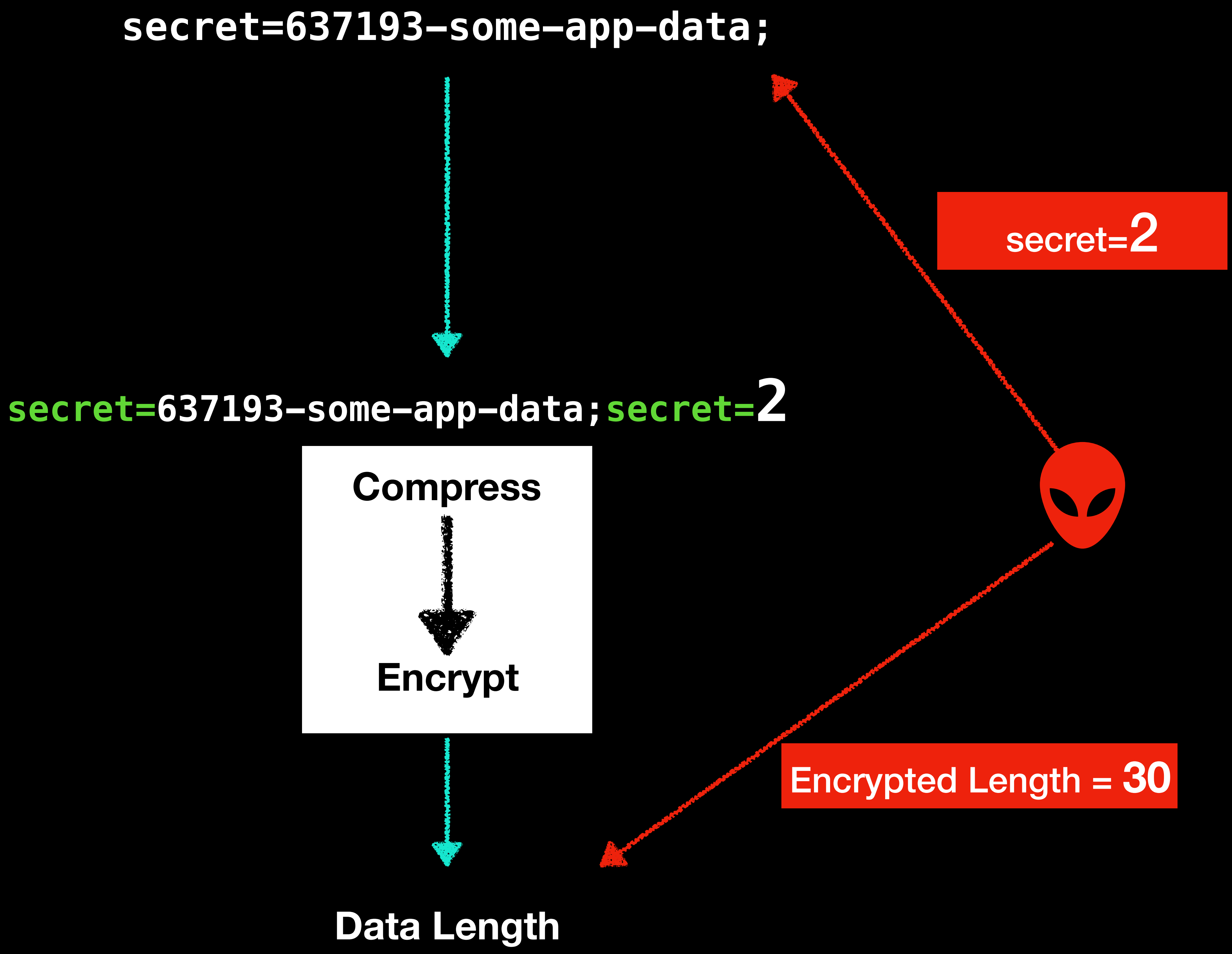
secret=637193-some-app-data;secret=1

Compress

Encrypt

Encrypted Length = 30

Data Length



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

`secret=637193-some-app-data;secret=2`

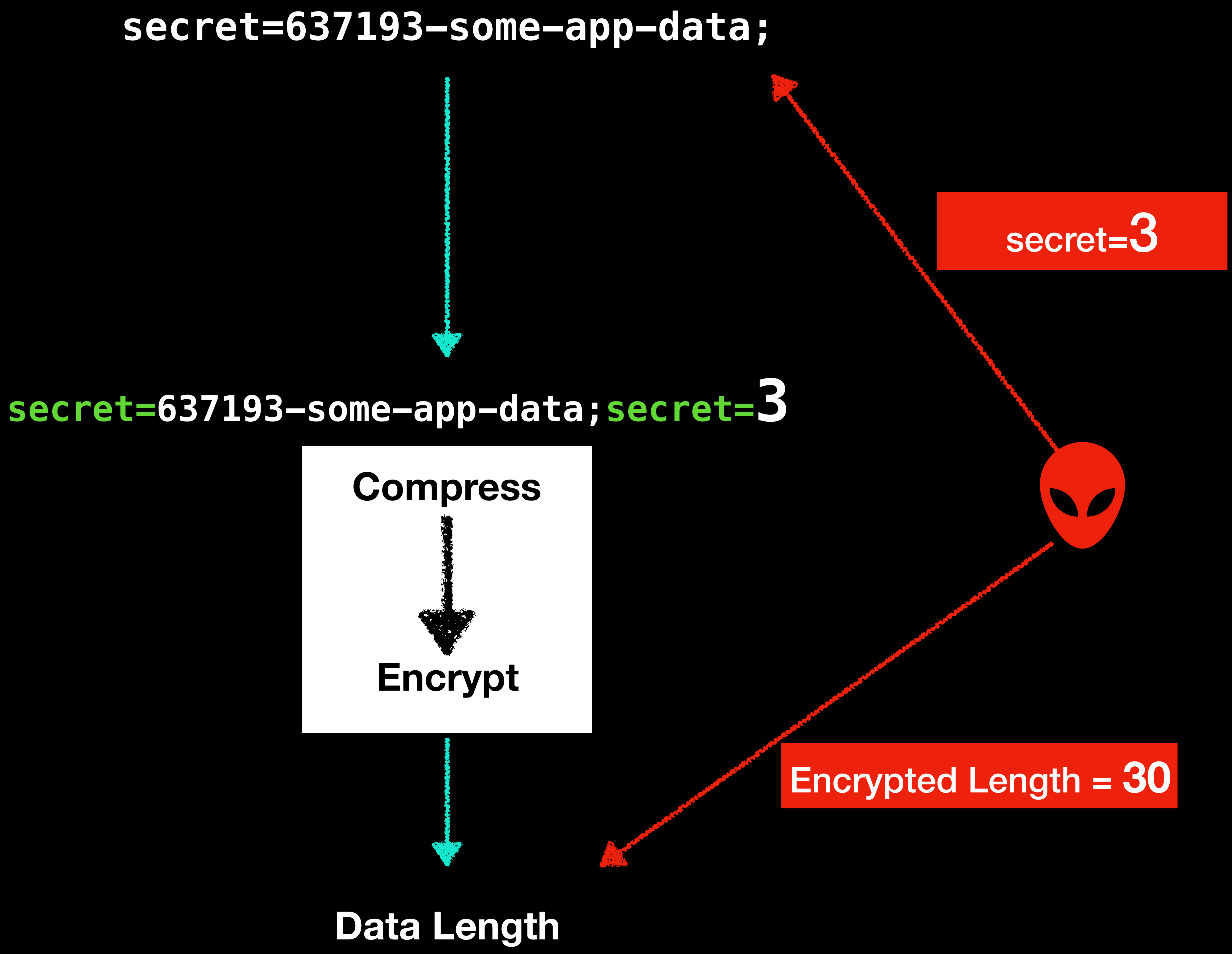
Compress

Encrypt

Data Length

secret=2

Encrypted Length = 30



secret=637193-some-app-data;

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

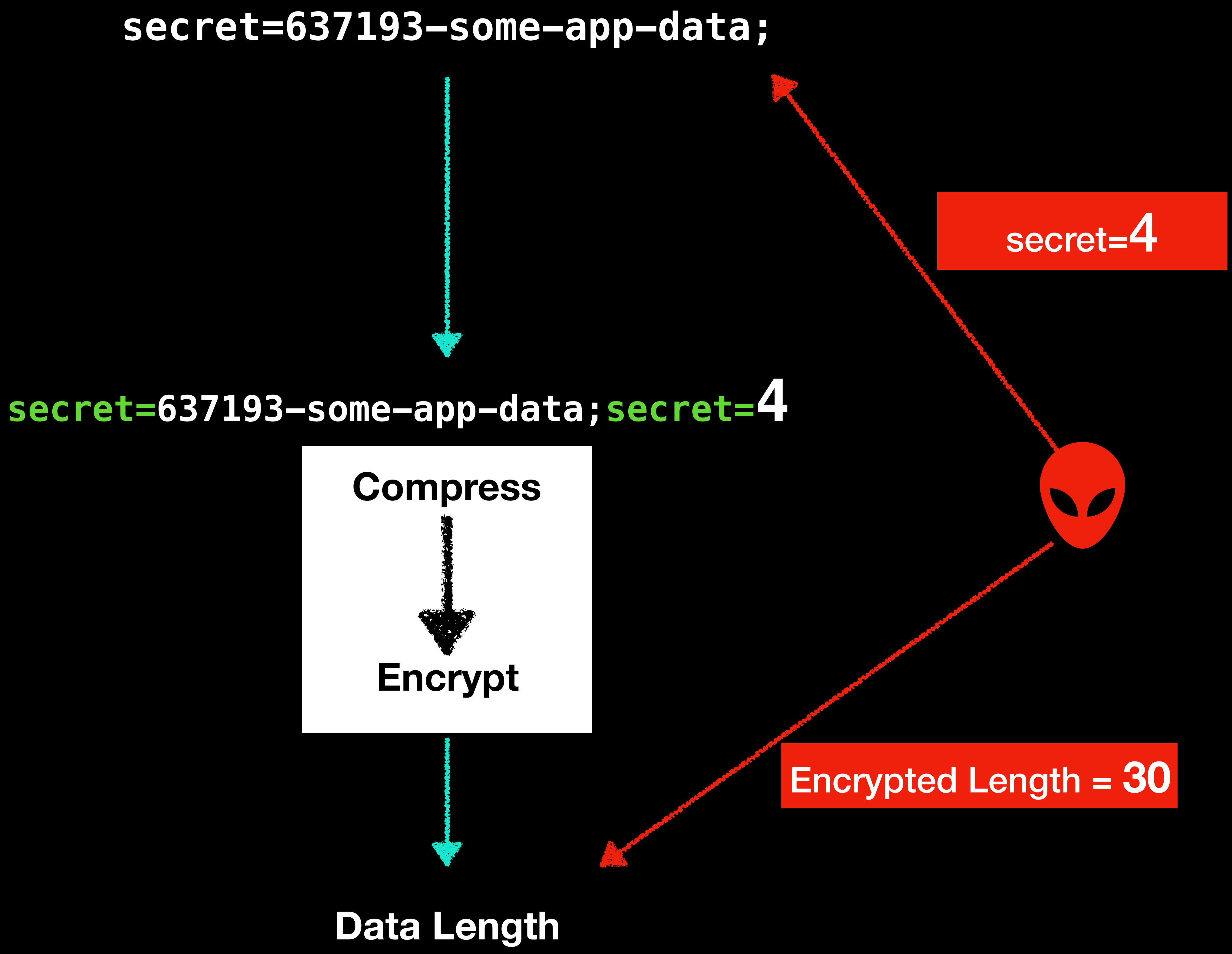
Compress

Encrypt

Data Length

secret=3

Encrypted Length = 30



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

`secret=637193-some-app-data;secret=4`

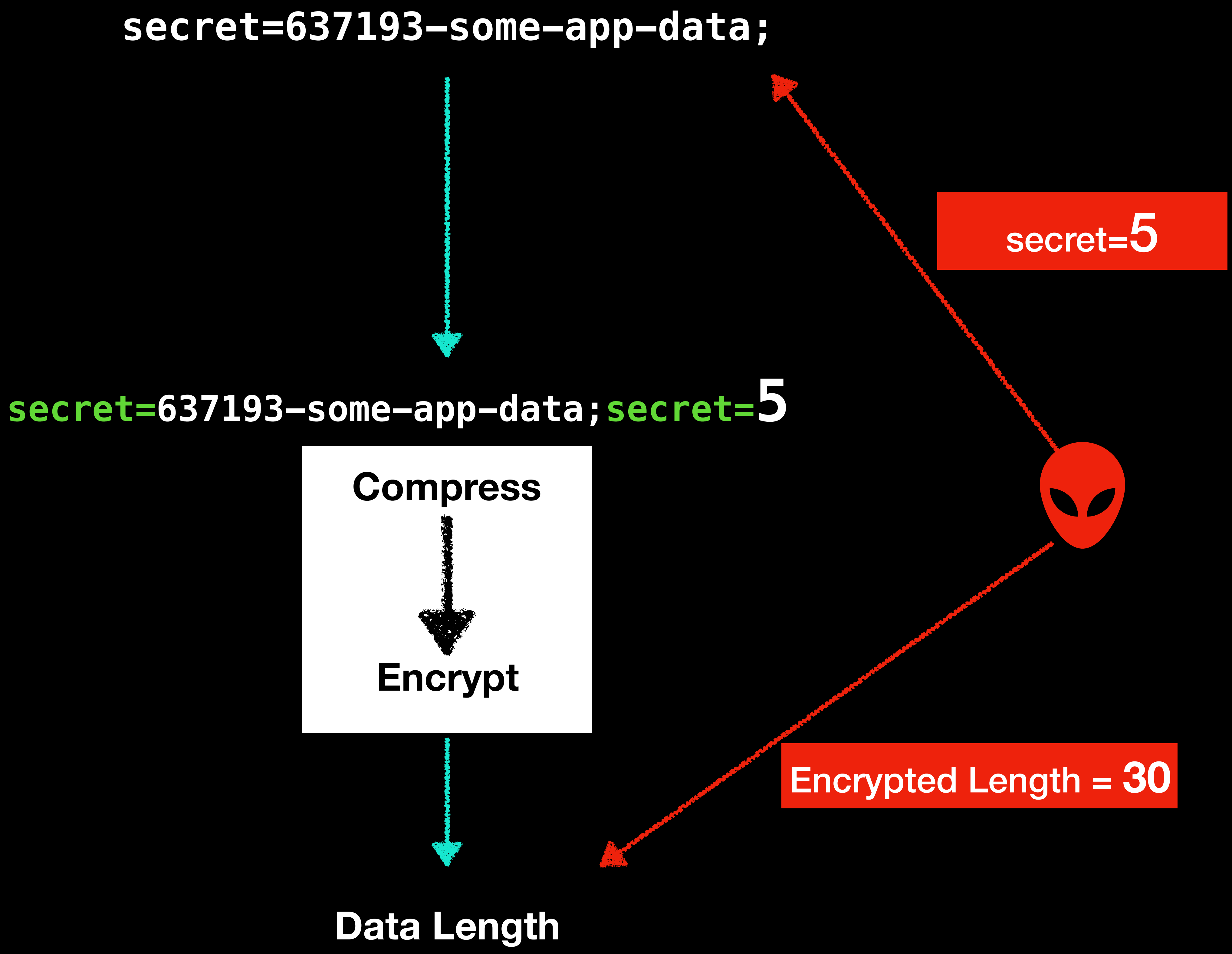
Compress

Encrypt

Data Length

secret=4

Encrypted Length = 30



secret=637193-some-app-data;

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

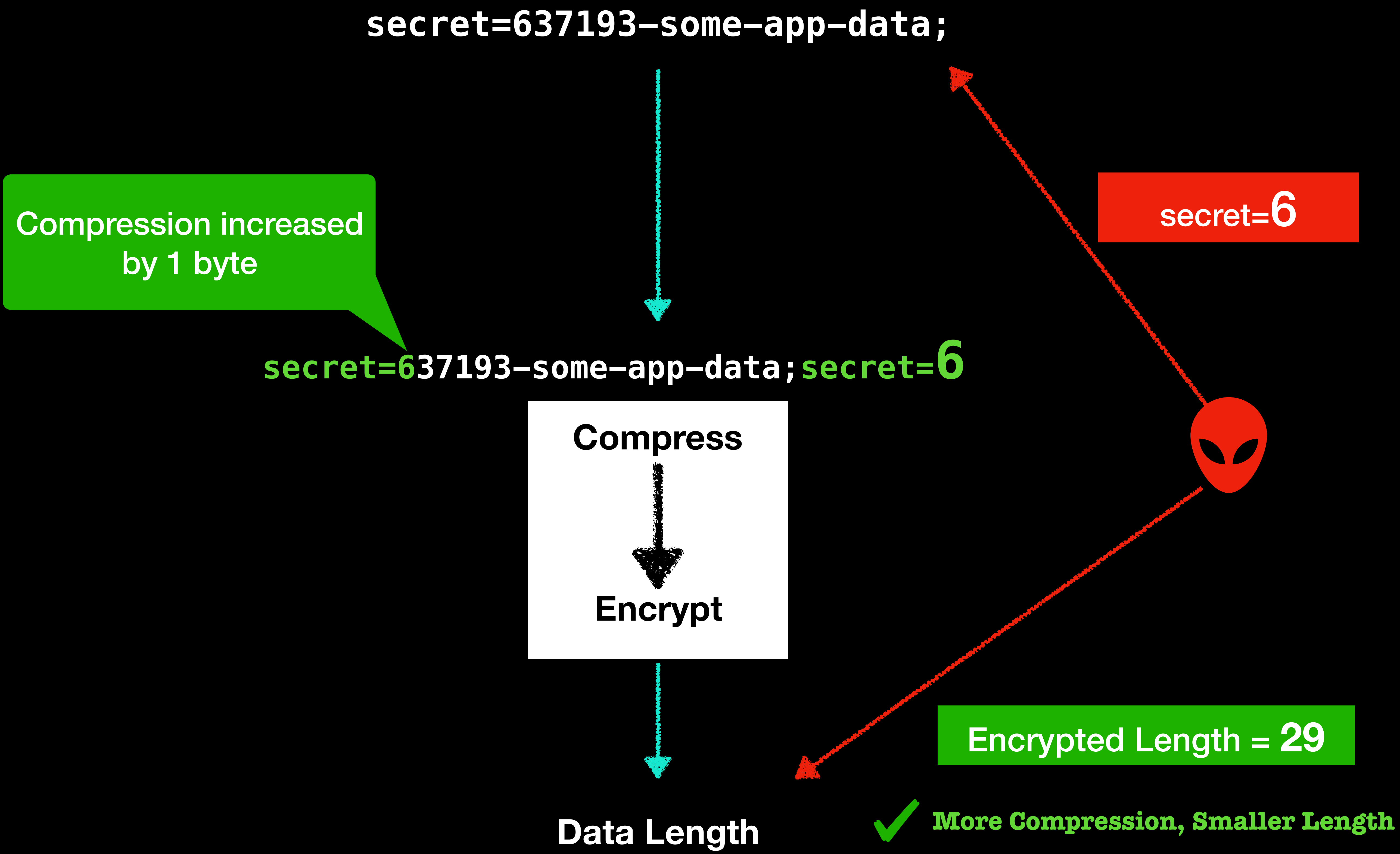
Compress

Encrypt

Data Length

secret=5

Encrypted Length = 30



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

Compression increased by 1 byte

`secret=637193-some-app-data;secret=6`

Compress

Encrypt

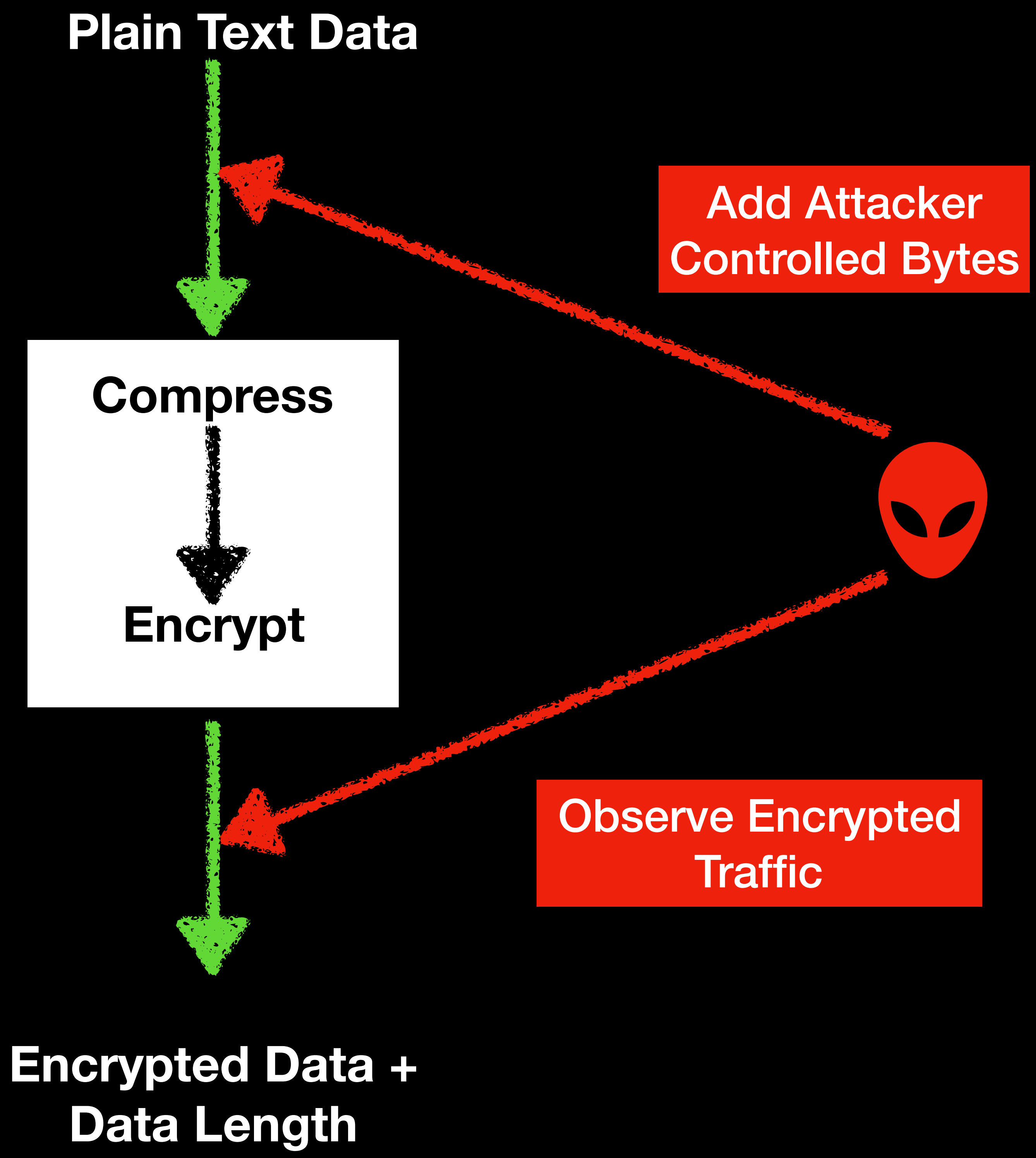
Data Length

secret=6

Encrypted Length = 29

✓ More Compression, Smaller Length

How can we convert this into a real world attack on browsers?



Plain Text Data

Compress

Encrypt

Encrypted Data +
Data Length

Add Attacker
Controlled Bytes

Observe Encrypted
Traffic

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

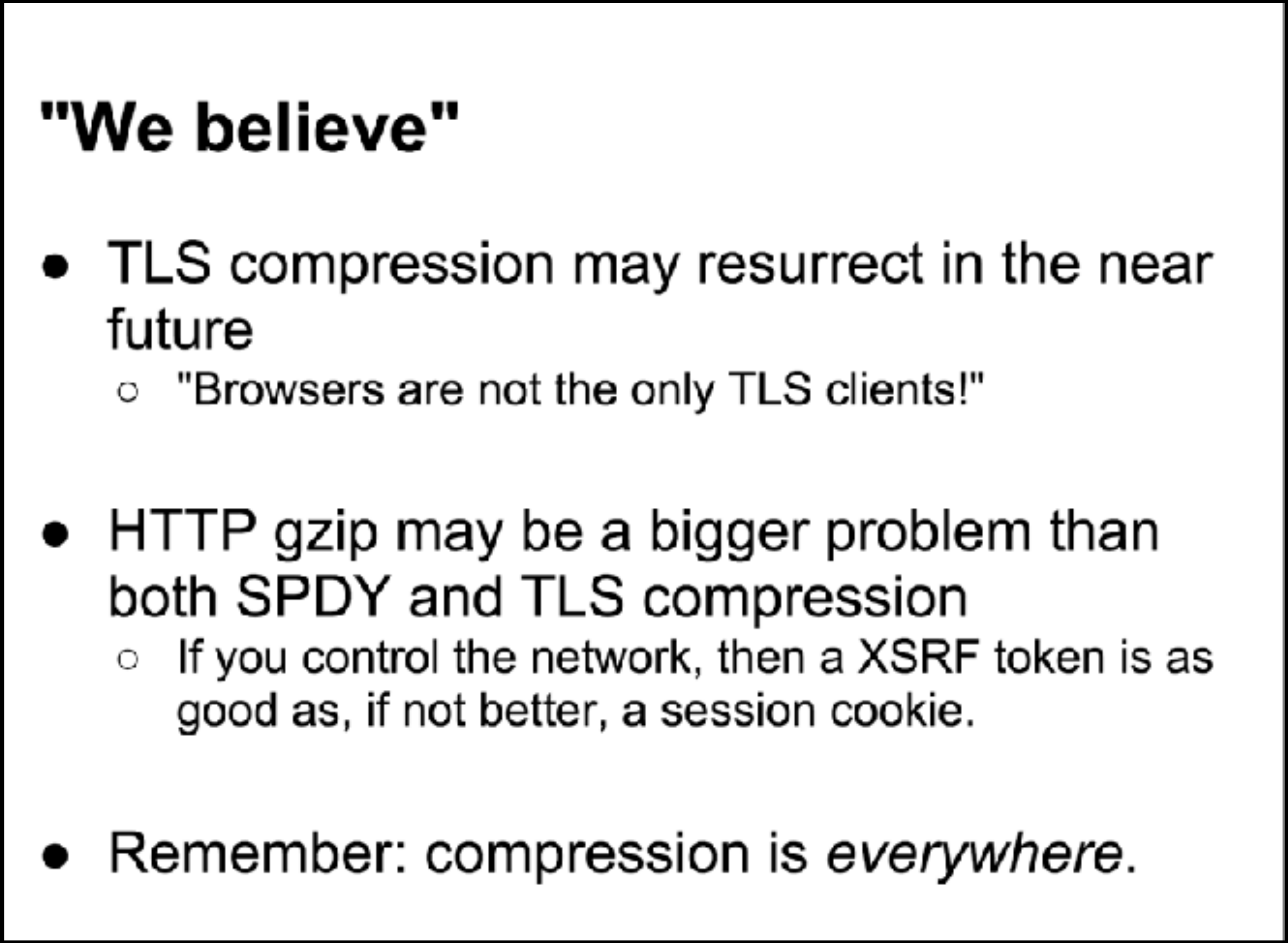
Juliano Rizzo, Thai Duong

The CRIME attack

A new TLS show was aired today
starring Juliano Rizzo and Thai Duong

EkoParty 2012

CRIME, 2012



"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*.

www.ekoparty.org/archive/2012/CRIME_ekoparty2012.pdf

TIME Attack 2013

Tal Be'ery, Amichai Shulman

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

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

black hat 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

There are more - HEIST, Practical Developments to BREACH

So, whats new today?

VPN Tunnels

TLS VPNs are pretty common these days

NordVPN

Hotspot Shield

PUREVPN

TunnelBear

ExpressVPN

privateinternetaccess

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-

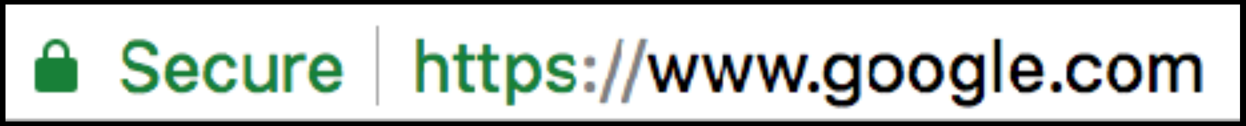
We have a compress then encrypt on all of data channel

VORACLE Attack

Under a VPN, HTTP WebApps are still insecure !

✓

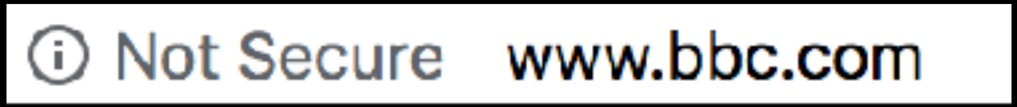
Things are safe, if the underlying app layer already uses an encryption channel.


🔒 Secure | https://www.google.com

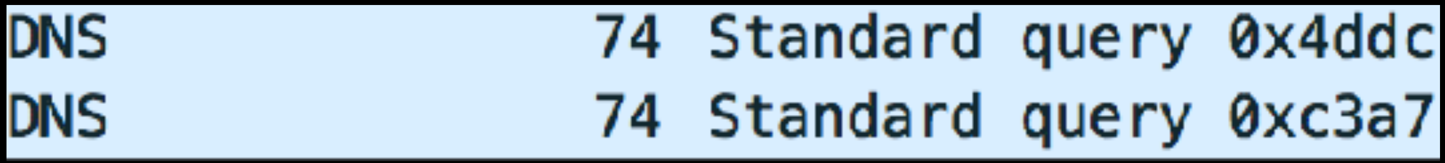
```
ssh user@website.com
```

✗

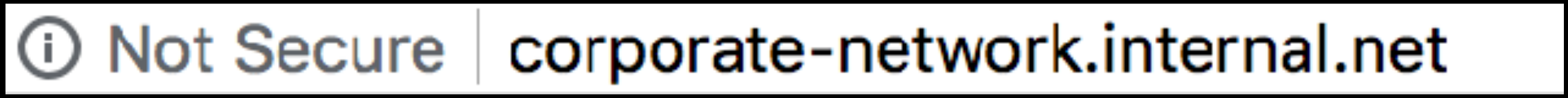
Things might go bad, if the VPN tunnel is helping you encrypt already non-encrypted data



ⓘ Not Secure www.bbc.com



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



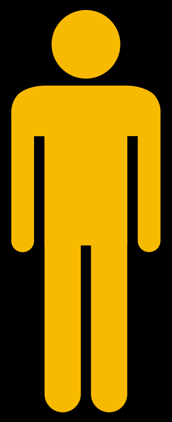
ⓘ Not Secure | corporate-network.internal.net

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

Requirements

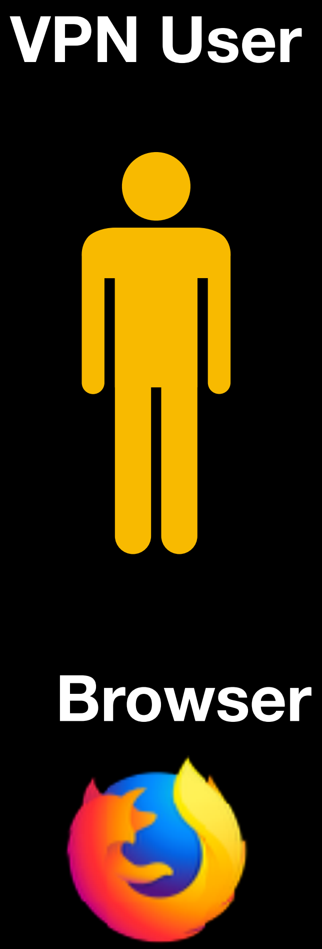
- ✓ VPN Server and Client has Compression enabled by default
- ✓ Attacker can observe VPN traffic
- ✓ VPN User visits attacker.com

Attack Setup

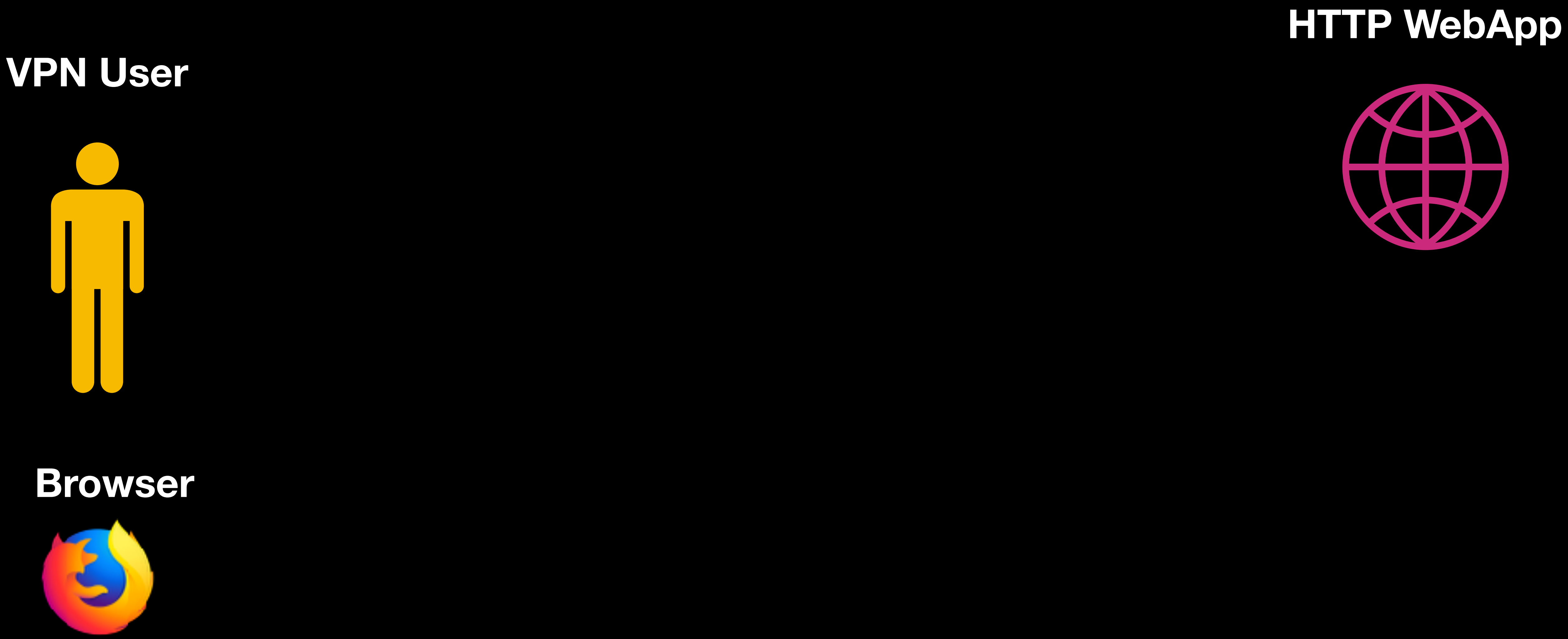


VPN User

Attack Setup



Attack Setup

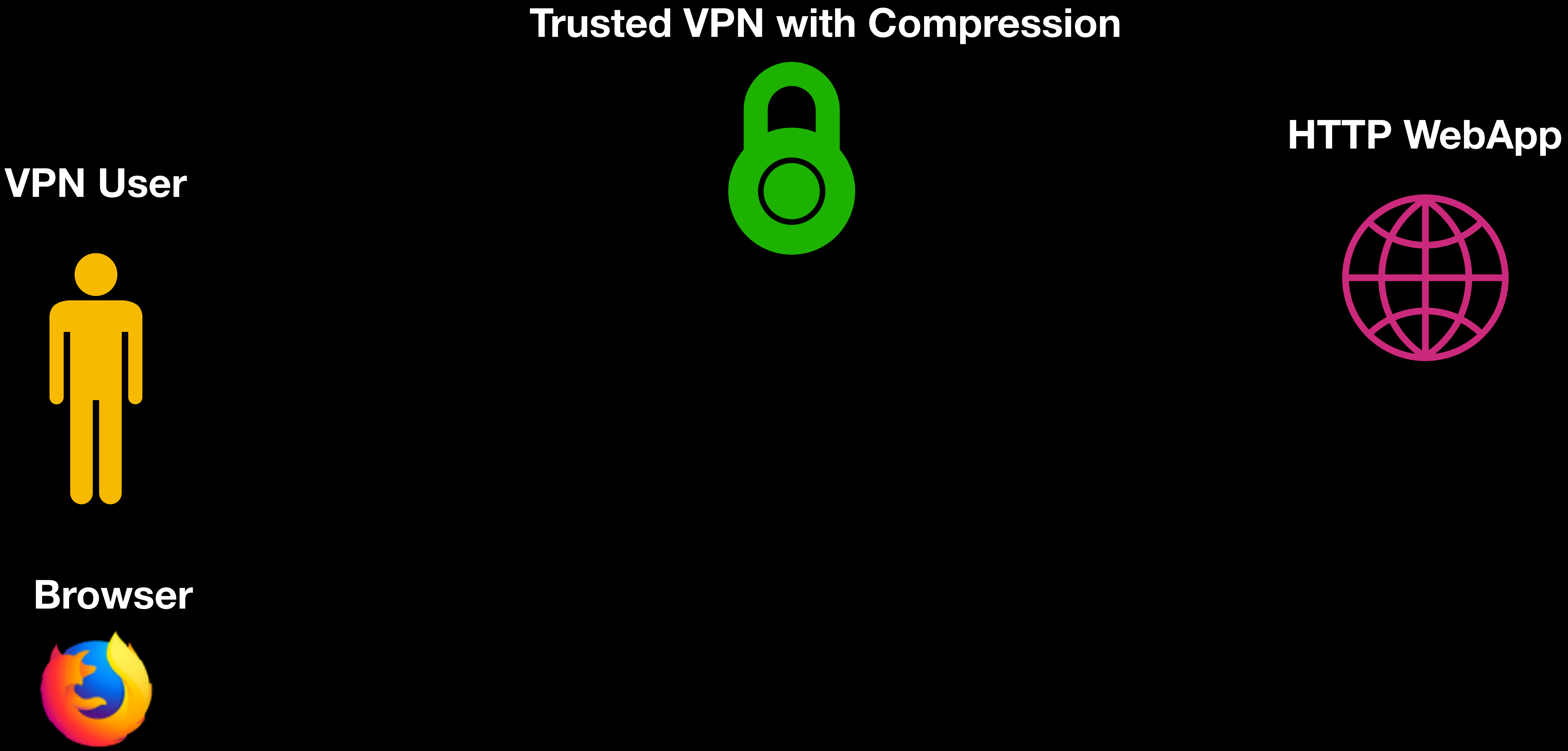


VPN User

Browser

HTTP WebApp

Attack Setup



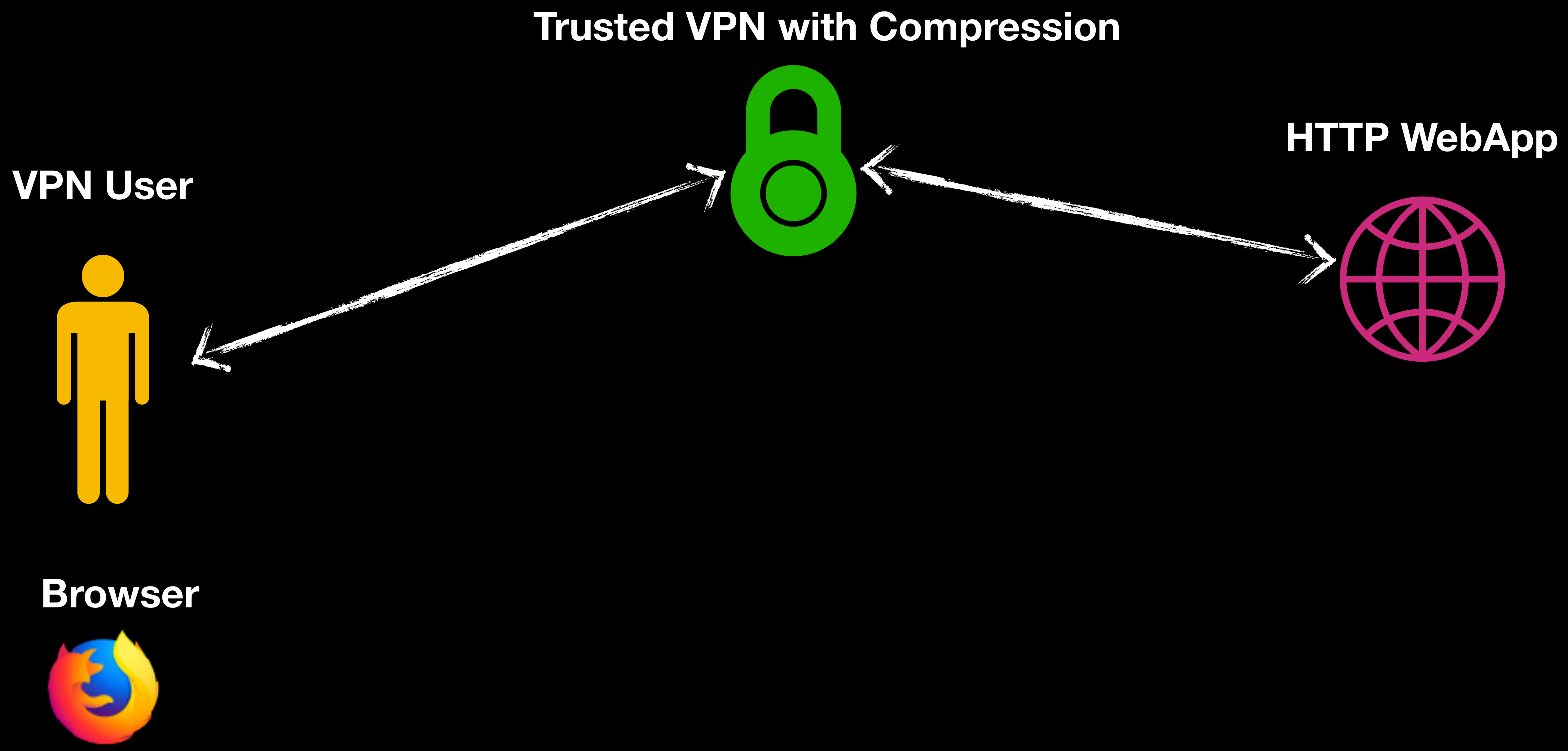
Trusted VPN with Compression

HTTP WebApp

VPN User

Browser

Attack Setup



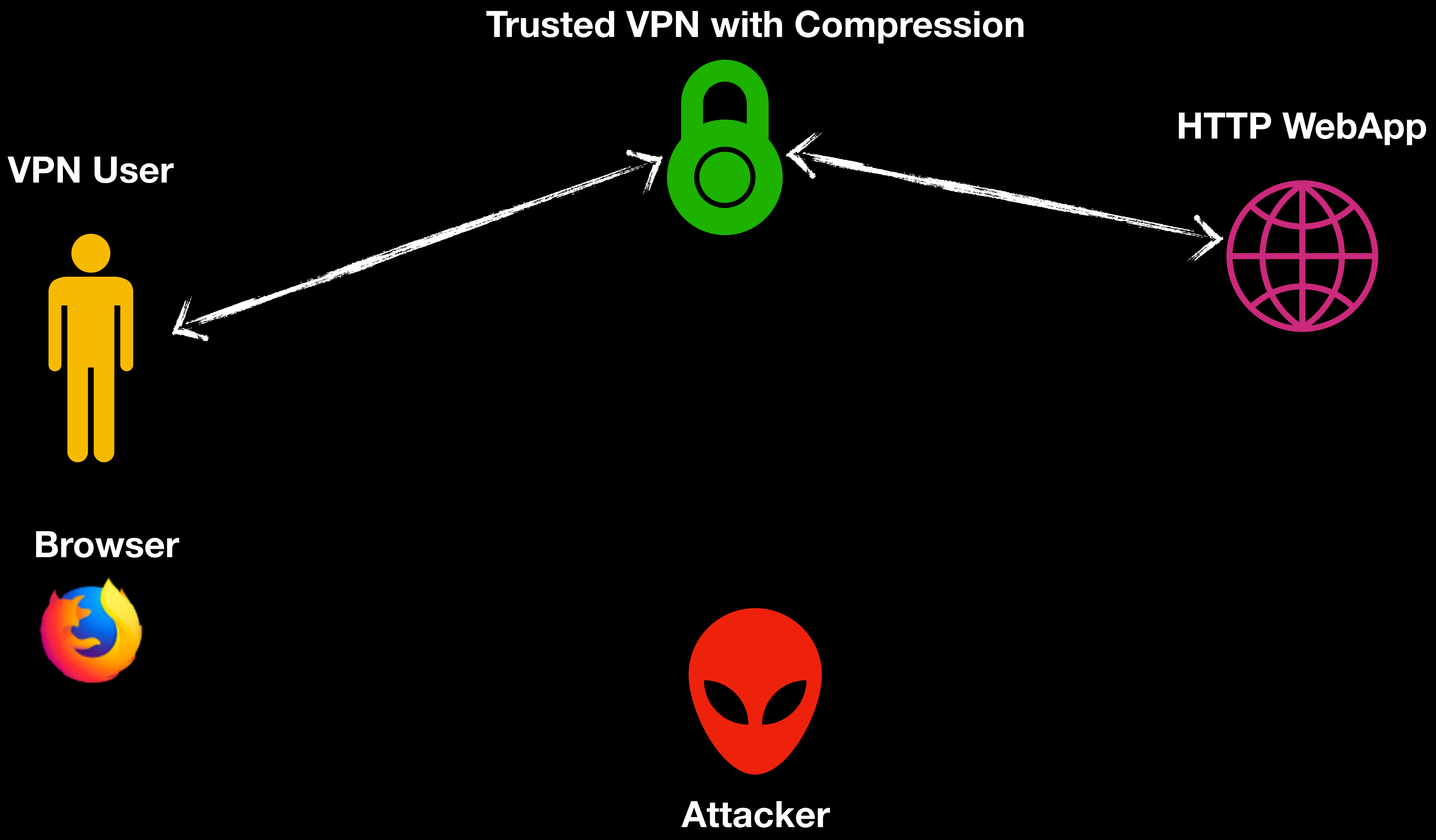
Trusted VPN with Compression

HTTP WebApp

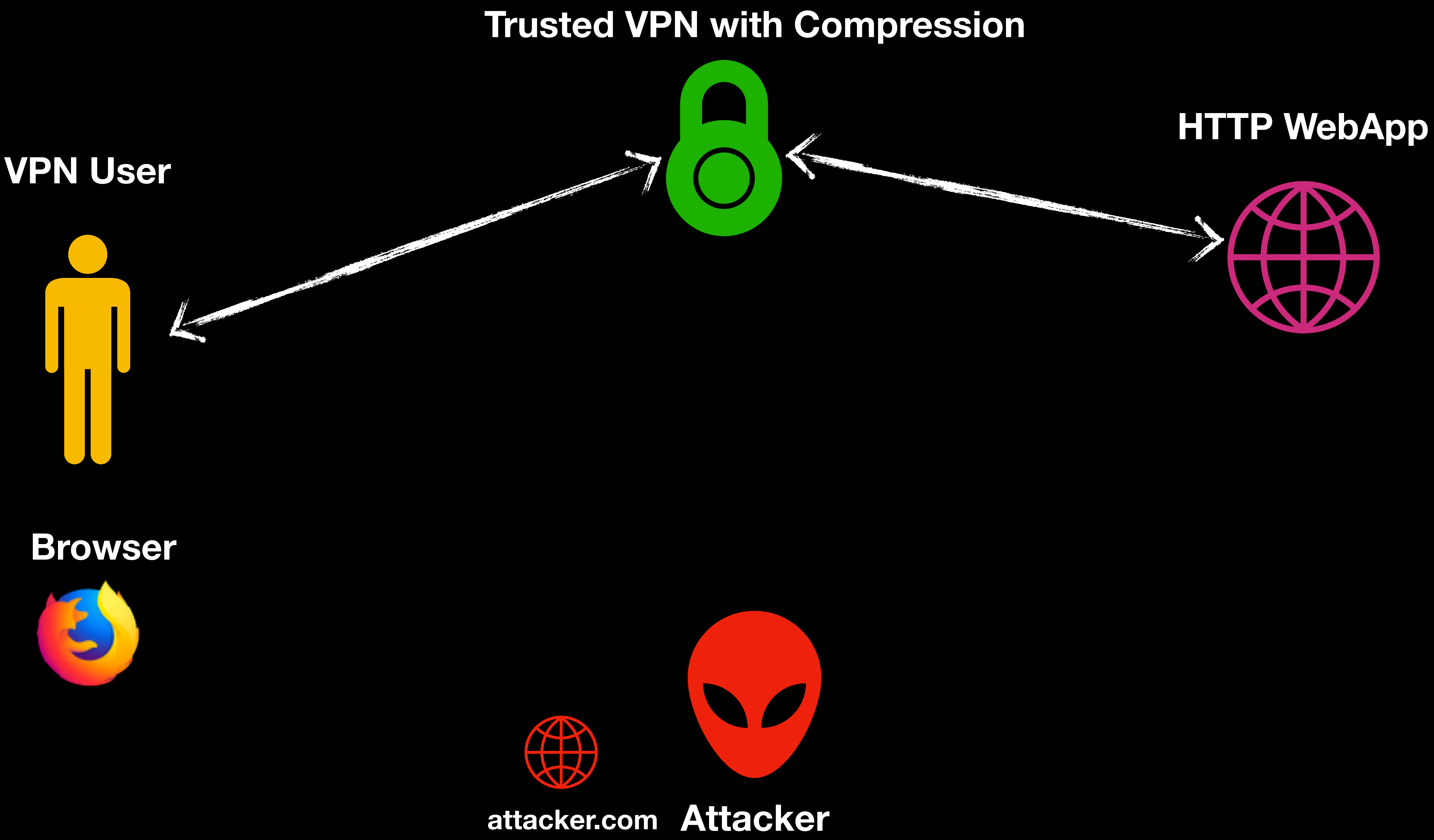
VPN User

Browser

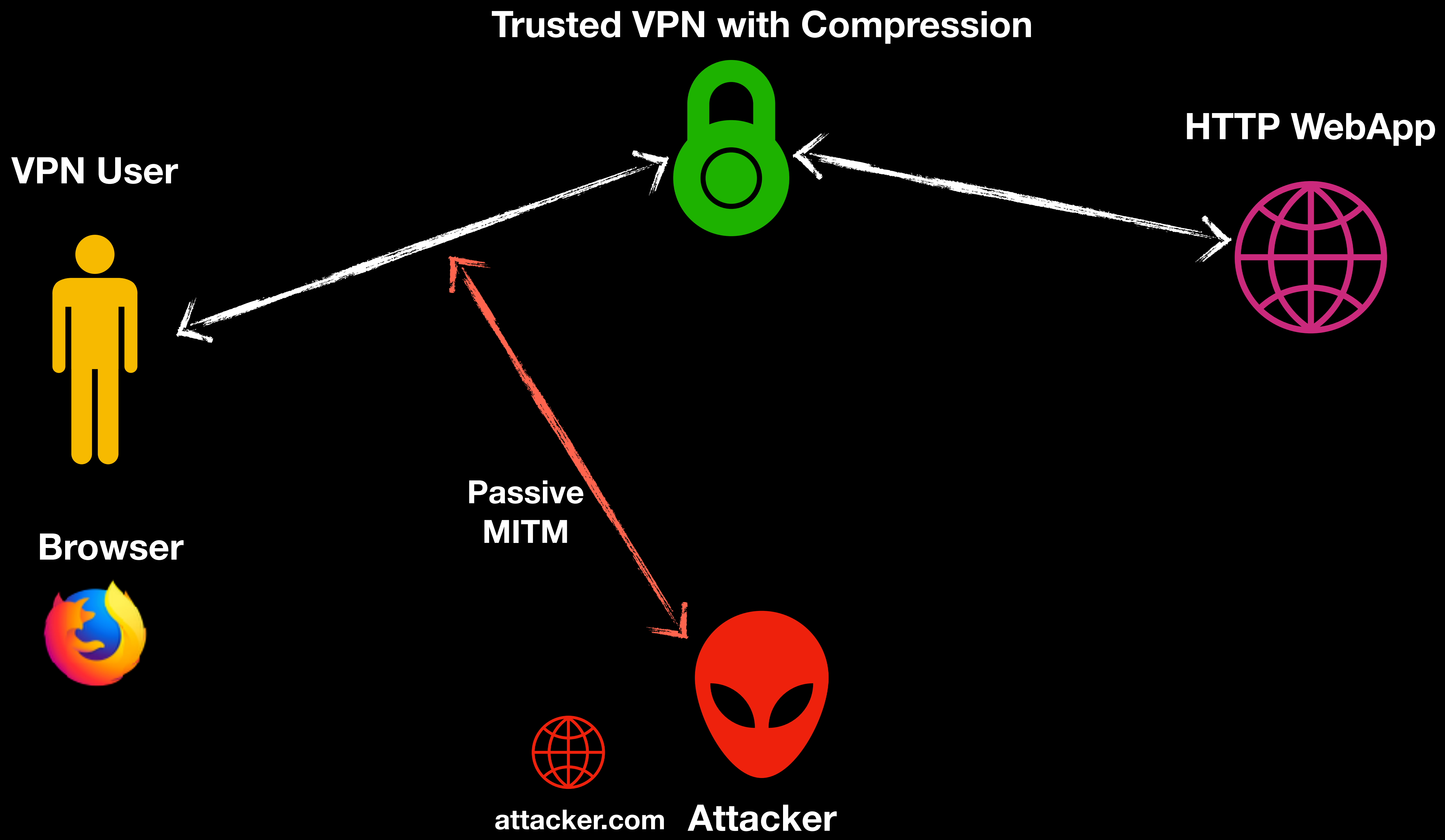
Attack Setup



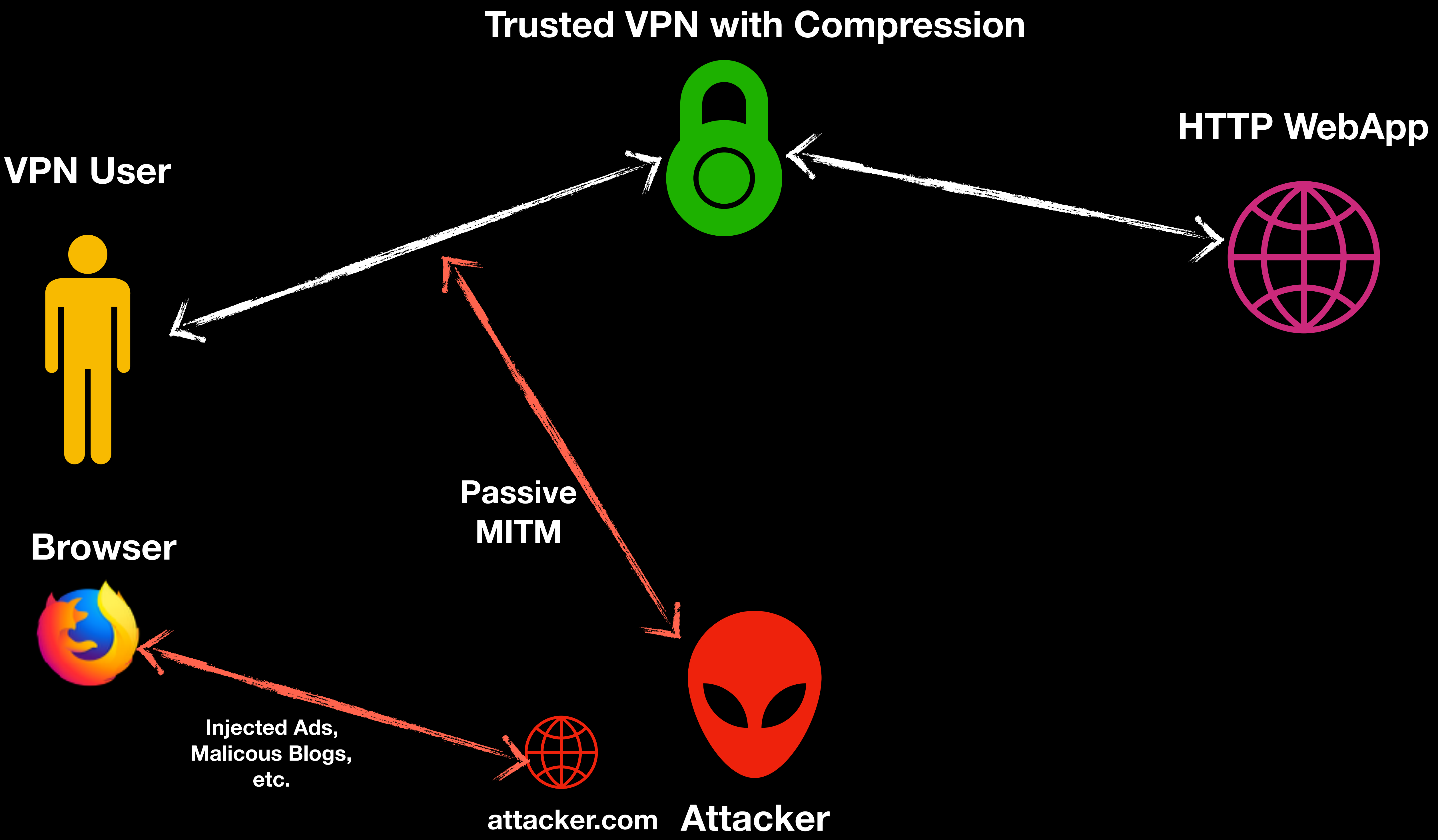
Attack Setup



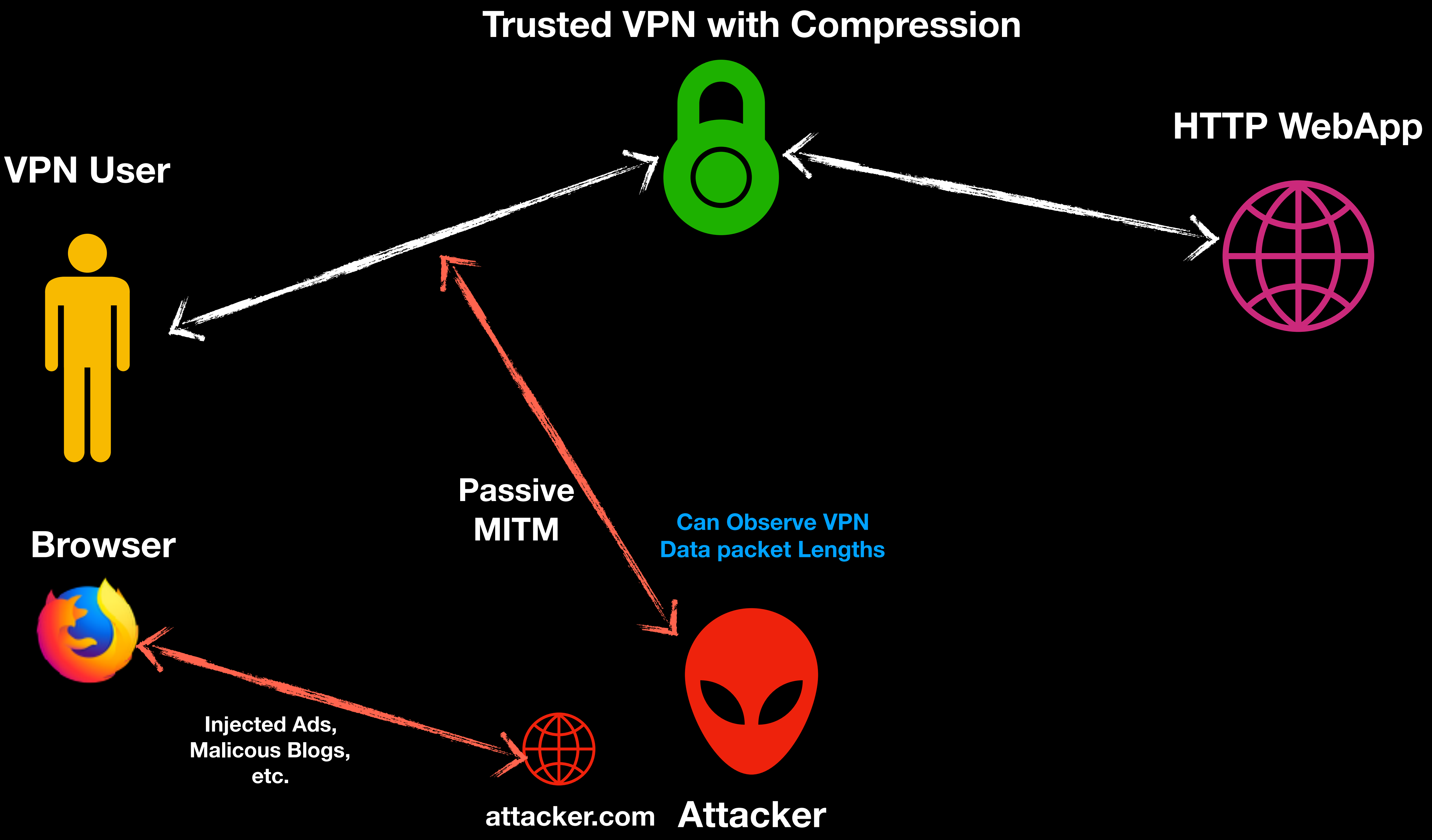
Attack Setup



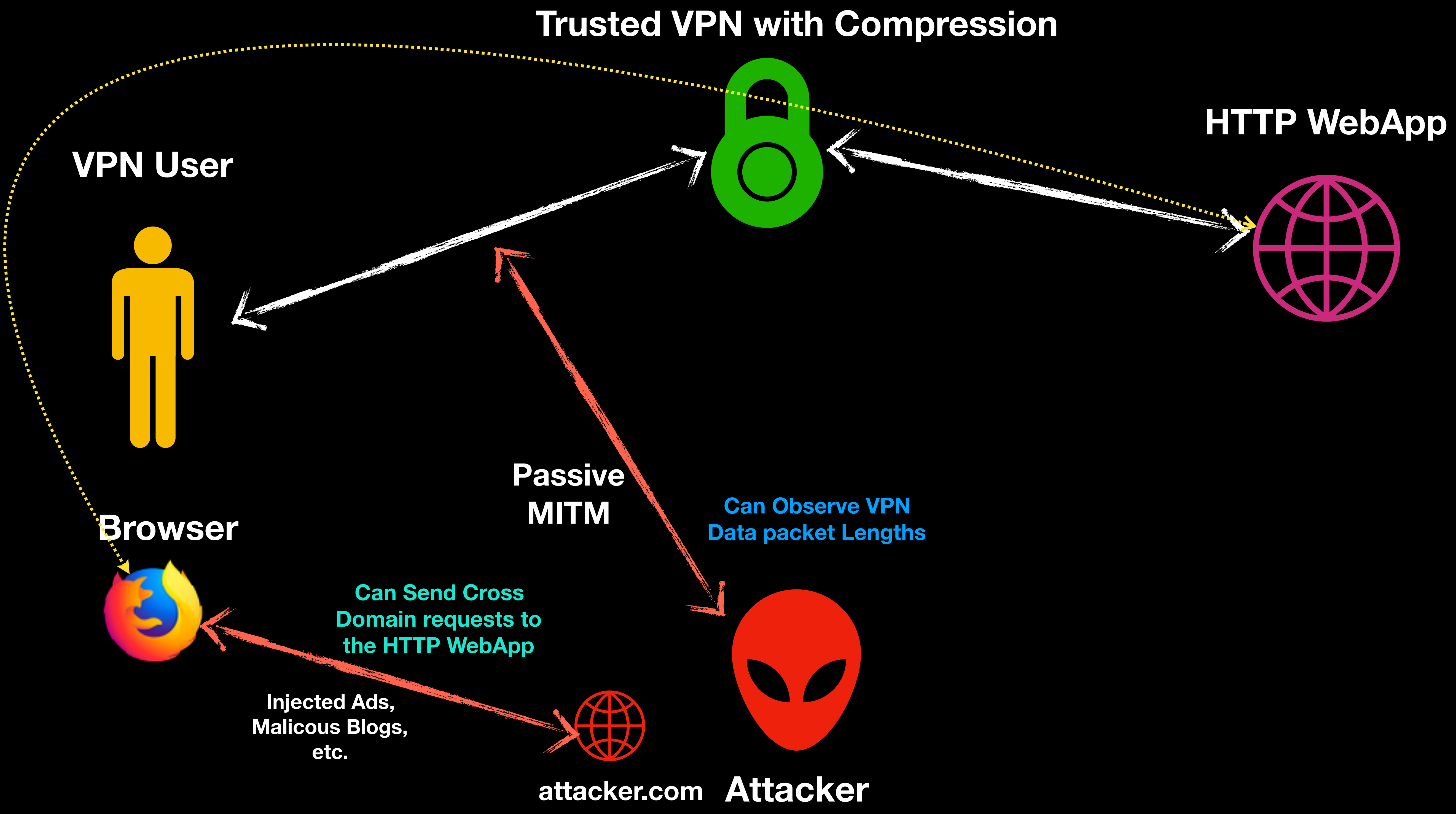
Attack Setup



Attack Setup



Attack Setup

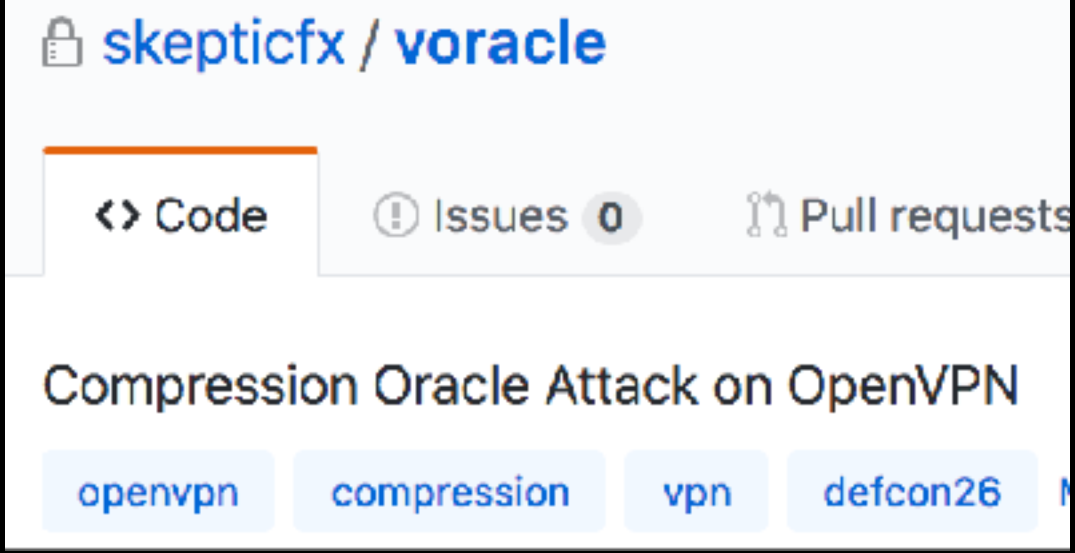


Attacker can now conduct Compression Oracle attacks on HTTP requests and responses

Demo

Browser	Mozilla Firefox
VPN Client	https://github.com/OpenVPN/openvpn3
VPN Server	OpenVPN Server
WebApp	http://insecure.skepticfx.com
Attack Goal	Steal sessionId cookie from a cross-domain website

Voracle



skepticfx / voracle

Compression Oracle Attack on OpenVPN

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

✗

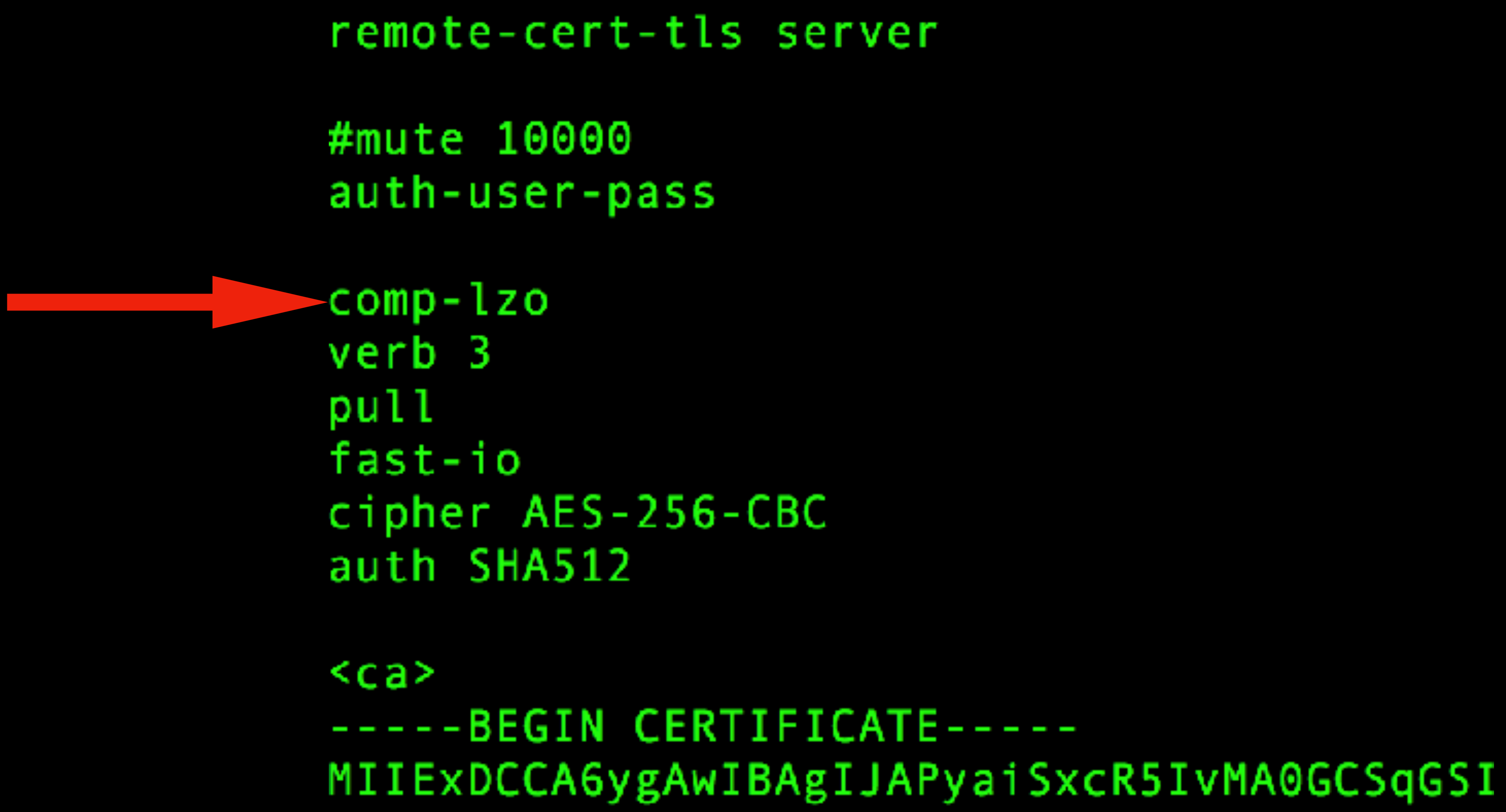
Mozilla Firefox sends them all in a single TCP data packet

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.

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
```

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**

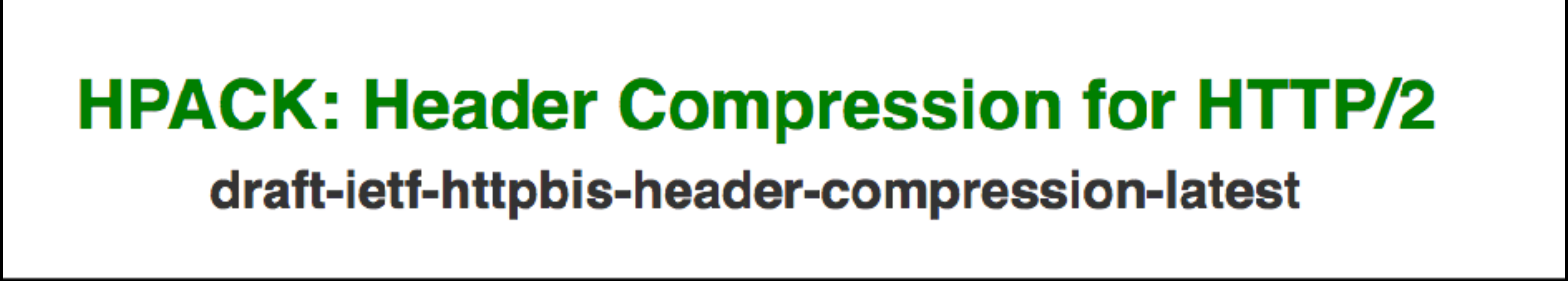
Fix?

Fixing Compression is an interesting problem

Remember when SPDY was vulnerable to CRIME?

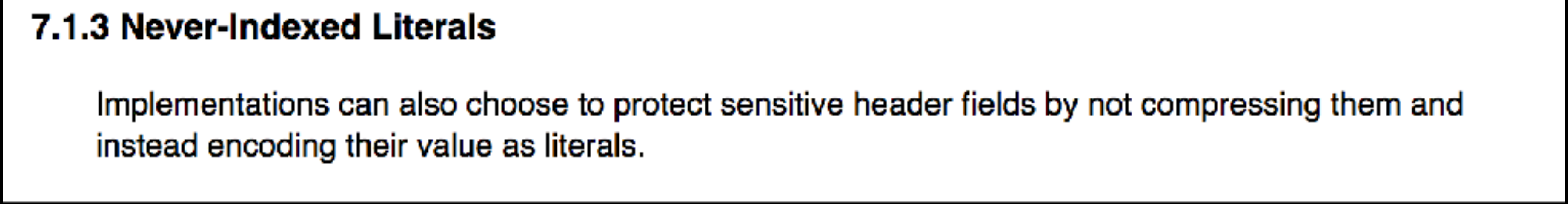
HPACK in HTTP/2

selectively disables header compression for sensitive fields



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

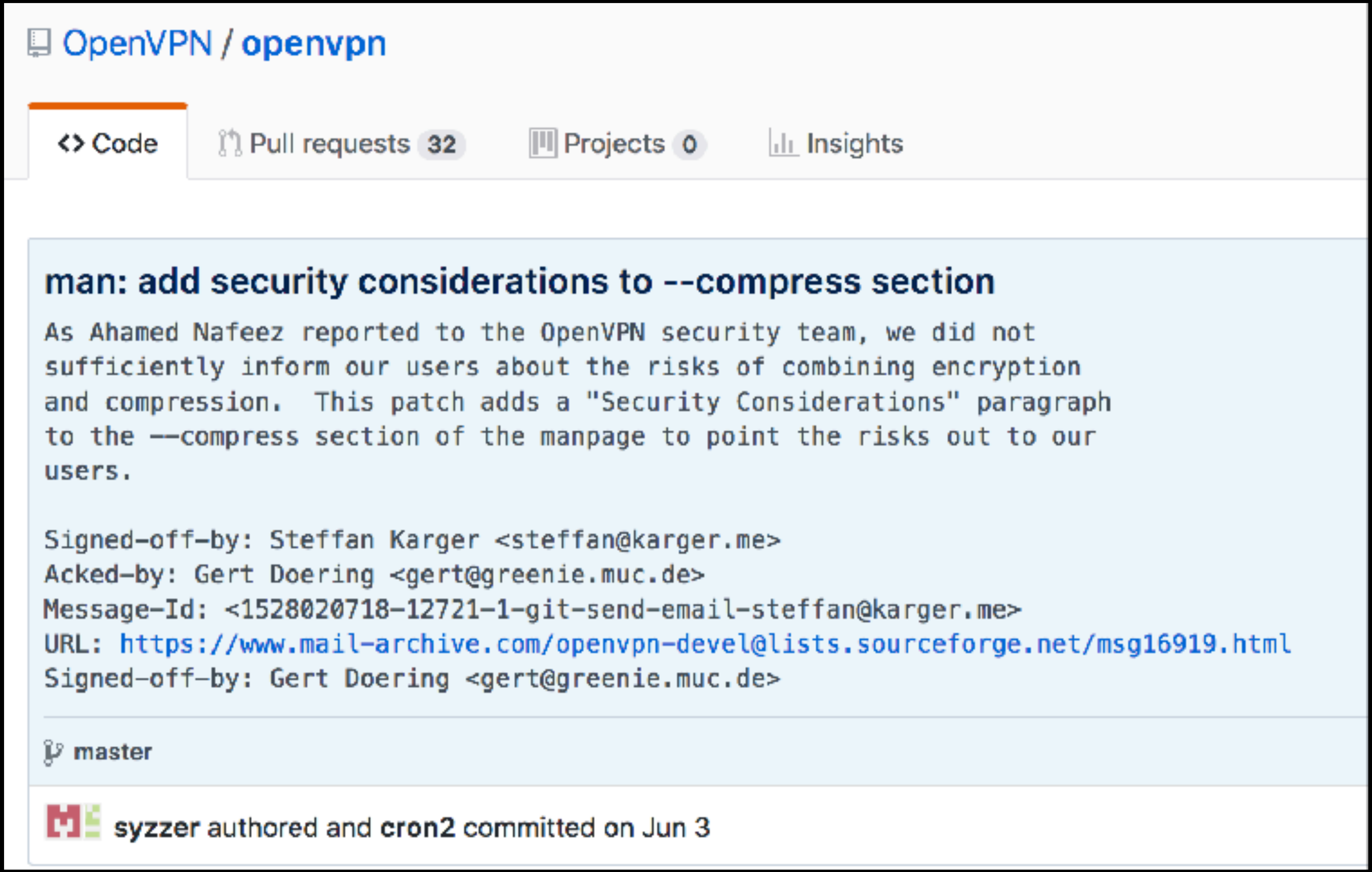
cf-nocompress

https://blog.cloudflare.com/a-solution-to-compression-oracles-on-the-web/

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.



man: add security considerations to --compress section

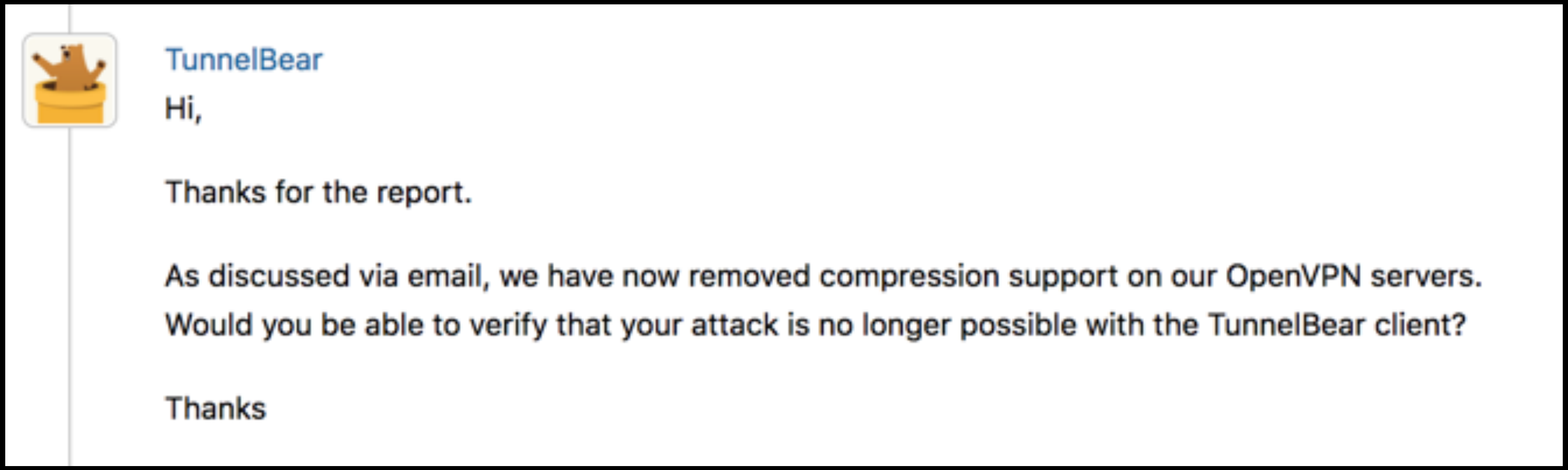
```
As Ahamed Nafeez reported to the OpenVPN security team, we did not
sufficiently inform our users about the risks of combining encryption
and compression.  This patch adds a "Security Considerations" paragraph
to the --compress section of the manpage to point the risks out to our
users.

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

syzzer authored and **cron2** committed on Jun 3

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

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

EndUsers & Website owners - If you are using VPN to access plain text websites over the internet, its 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