Getting TLS Right

Zack Tollman

WIRED

@tollmanz

TLS is hot right now

We implement TLS

poorly

SSL Pulse

Reviews SSL/TLS sites in Alexa's Top 300k sites

https://www.trustworthyinternet.org/ssl-pulse/

474 are vulnerable to heartbleed

21.0% use *weak ciphers*

47.3% support

SSLv3

38.3% do no support

Forward Secrecy

97.3% do not use
HSTS

83.6% are
insecure

*“**misconfiguration** errors are undermining the potential security”*

Kranch & Bonneau (2015)

http://www.internetsociety.org/sites/default/files/01_4_0.pdf *“developers who **should** be in the best position to **understand** these new tools”*

Kranch & Bonneau (2015)

http://www.internetsociety.org/sites/default/files/01_4_0.pdf *“industry-wide* ***configuration problem*** *with the deployment of DHE key exchange"*

Huang, Adhikarla, Boneh, & Jackson (2014)

http://www.w2spconf.com/2014/papers/TLS.pdf

Why?

Why?

Why?

*Unless you are a cryptographer, this **stuff is hard***

Copying and pasting is easy

```
ssl_protocols       TLSv1 TLSv1.1 TLSv1.2;

ssl_certificate     /path/to/public.crt;
ssl_certificate_key /path/to/private.key;

ssl_prefer_server_ciphers    on;

ssl_ciphers    ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-
RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-
GCM-SHA384…;
```

https://github.com/igrigorik/istlsfastyet.com/blob/master/nginx/includes/ssl.conf

Knowing what you are doing is hard

TLS Basics

Transport Layer Security

```
SSLv2
SSLv3
TLSv1.0
TLSv1.1
TLSv1.2
```

SSLv2	1995
SSLv3	1996
TLSv1.0	1999
TLSv1.1	2006
TLSv1.2	2008

SSLv2	1995	PHP Tools
SSLv3	1996	PHP/FI (2.0)
TLSv1.0	1999	PHP 3.0
TLSv1.1	2006	PHP 5.2
TLSv1.2	2008	PHP 5.2.8

SSLv2	1995	MITM
SSLv3	1996	POODLE
TLSv1.0	1999	BEAST
TLSv1.1	2006	
TLSv1.2	2008	

Provides authentication, encryption, integrity, and key exchange

Authentication

Encryption

Integrity

Key exchange

Compromise of any of these, compromises the whole system

Cipher Suites

Combination of algorithms for authentication, encryption, integrity and key exchange

ECDHE-RSA-AES128-GCM-SHA256

ECDHE-RSA-AES128-GCM-SHA256

Key Exchange

Certificate signing algorithm (authentication)

ECDHE-**RSA**-AES128-GCM-SHA256

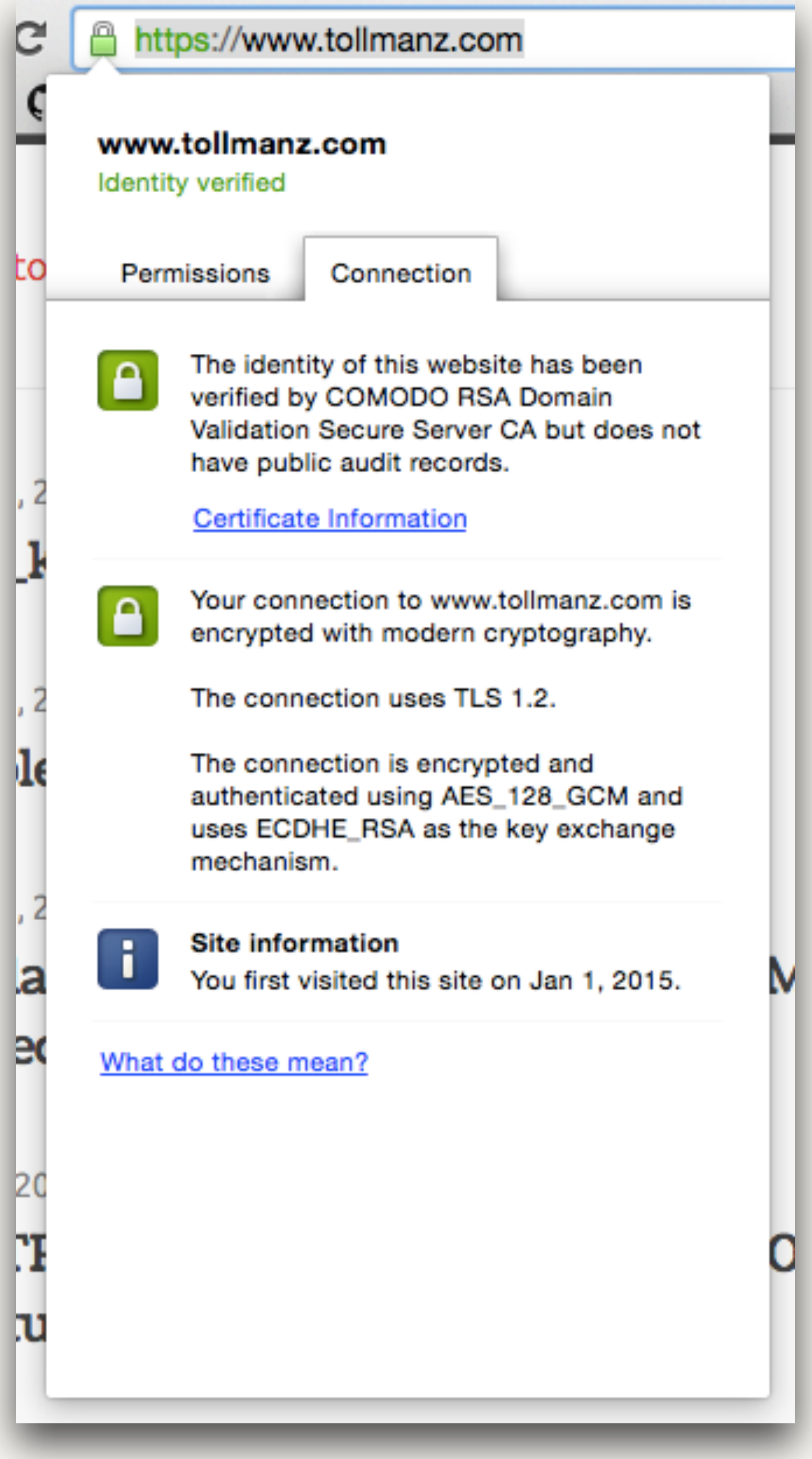
ECDHE-RSA-**AES128-GCM**-SHA256

Cipher (Encryption)

Message authentication code (integrity)

ECDHE-RSA-AES128-GCM-**SHA256**

```
ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-
GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-
SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH
+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-
AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-
AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-
AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-
SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-
SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-
SHA256:AES256-SHA256:AES128-SHA:AES256-
SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!
EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-
CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-
SHA
```



https://www.tollmanz.com

www.tollmanz.com

Identity verified

Permissions | Connection

The identity of this website has been verified by COMODO RSA Domain Validation Secure Server CA but does not have public audit records.

Certificate Information

Your connection to www.tollmanz.com is encrypted with modern cryptography.

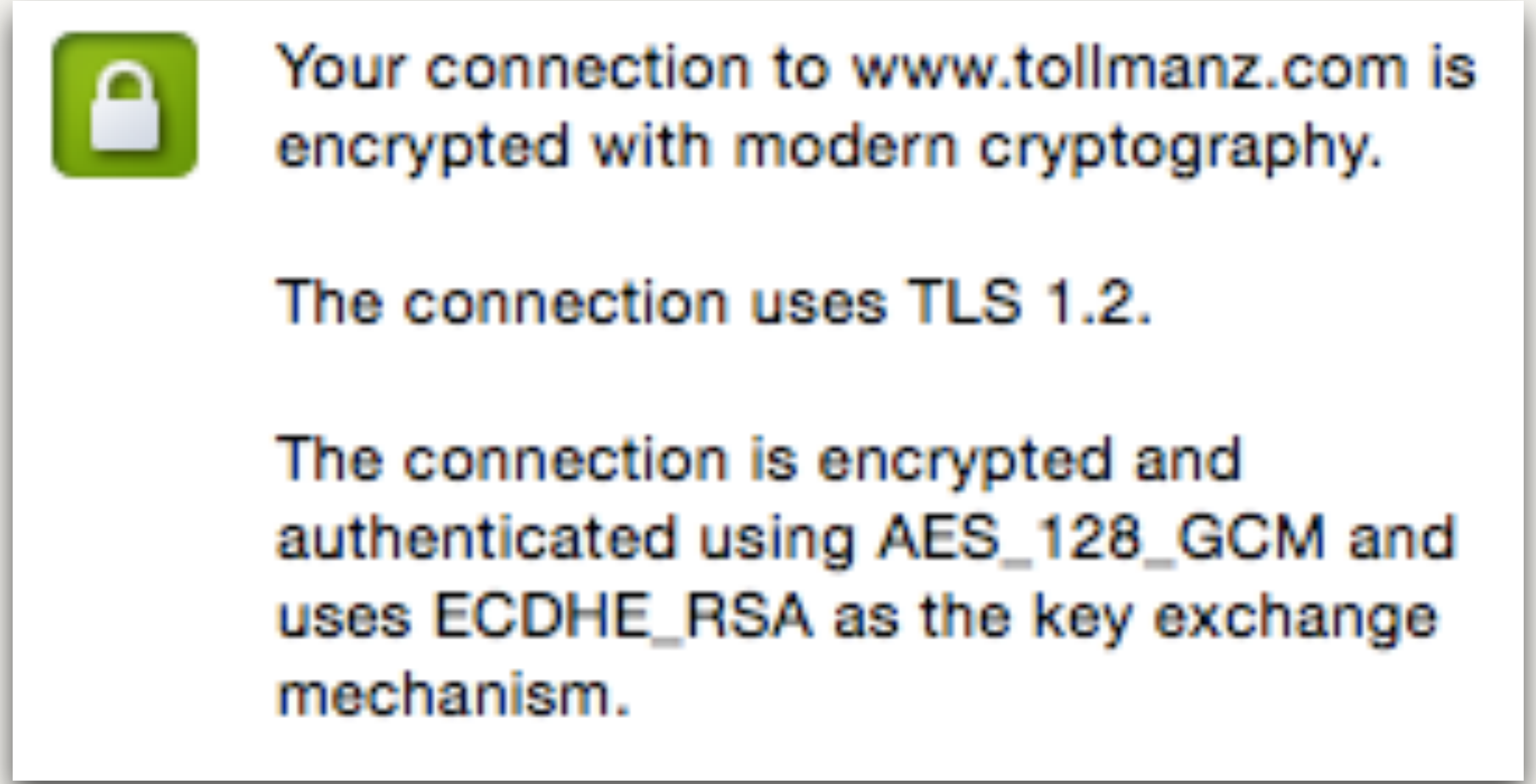
The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.

Site information

You first visited this site on Jan 1, 2015.

What do these mean?



Your connection to www.tollmanz.com is encrypted with modern cryptography.

The connection uses TLS 1.2.

The connection is encrypted and authenticated using AES_128_GCM and uses ECDHE_RSA as the key exchange mechanism.

TLS Handshake

Client presents supported cipher suites

Server chooses suite to use

Certificate sent to client

Verified with signing algorithm to authenticate the certificate

ECDHE-**RSA**-AES128-GCM-SHA256

RSA is the most widely supported signing mechanism

Recommendation

RSA for Certificate Authentication

but ECDSA will be the new hotness

Key exchange

Negotiate the key for encryption and decryption

ECDHE-RSA-AES128-GCM-SHA256

Preferring Ephemeral Diffie Hellman algorithms give you Perfect Forward Secrecy

Guarantees a different key for each connection

RSA uses the
same key
for each connection

Recommendation

ECDHE for Key Exchange

Server is verified and keys are negotiated

Key is used by encryption algorithm

ECDHE-RSA-**AES128-GCM**-SHA256

Advanced Encryption Standard (AES) is the only real option

Other ciphers have known weaknesses

Can choose between 128 and 256 bit encryption

Recommendation

AES-128-GCM for encryption

but watch for ChaCha20

Encrypted messages are signed to guarantee integrity

SHA-256 and SHA-384 are the two practical options

Recommendation

SHA-256 for MAC

but watch for Poly1305

So…huh?

Use Mozilla's guide

https://wiki.mozilla.org/Security/Server_Side_TLS

HTTP Strict Transport Security

SSL Stripping

http://www.thoughtcrime.org/software/sslstrip/

What if HTTP variant was never accessed?

HSTS blocks browser from HTTP version of site

Recommendation

Set HSTS headers

Set HSTS only after mixed content issues are resolved

Content Security Policy

Mixed content warnings are ***bad***

Whitelist assets loaded on your site

*Whitelist only **HTTPS** assets*

Use `report-only` *variant*

Current recommendation

Use CSP headers

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com
```

```
Content-Security-Policy-Report-
Only:
    default-src ’self’ https:;
    font-src https://
fonts.gstatic.com;
    img-src ’self’ https:;
    style-src ‘self’ https:
https://fonts.googleapis.com;
    script-src ’self’ https:
https://ssl.google-analytics.com;
    report-uri /beacon.php
```

HTTPS Mixed Content Detector Plugin for WordPress

Do your homework

Make good decisions

Maintain your TLS config
like you maintain your code

The Code Book

Simon Singh

High Performance Browser Networking (TLS Chapter)

Ilya Grigorik

Bulletproof SSL and TLS

Ivan Ristic

SSL and TLS: Designing and Building Secure Systems

Eric Rescorla

Zack Tollman

WIRED

@tollmanz

tollmanz.com/mwphp15