The cost of legacy

An introduction to TLS, PKI and x509

MODIO
Secure & efficient energy information

Introductions

"Release Dictator"

Systems nerd
Security fellow
Free Software Fan
Opinionated Unixbeard
Social Justice Aficionado

D.S. Ljungmark
Modio AB
https://twitter.com/spidler
https://github.com/Spindel/

Slides

https://www.modio.se/pages/presentations.html

MODIO
Secure & efficient energy information

TLS is a collection of

ciphers, hashes

MACS, curves

algorithms, certificates

negotiations, extensions

exchange formats

SSLv2 (not standard)

- The first SSL from 1995

the year after Python 1.0

- Deprecated for 20 years

SSLv3 (not standard)

- Released in 1996

the year before Python 1.5

- Deprecated for 17 years

The Crypto Wars

- Munitions
- Export restrictions
- Weakened crypto

TLS v1.0: The first standard

- Released in 1999
- Pre-dating Python 2.0
- No export crypto

TLS v1.1

- Released in 2006
- Before Python 2.4

TLS v1.2 (Current day)

- Released in 2008
- Same year as Python 2.5

- Extension support

Cracks in the shell

RFC 7457

BEAST

CRIME, BREACH

POODLE (SSLv3)

RC4

Logjam, FREAK

GHOST

DROWN

MD5, SHA1

Cleaning out...
...or not

- SSLv2 shot in the neck (2016)
- SSLv3 taken out back (2016)
- TLSv1.0, with broken parts, still alive
- TLSv1.1, with broken parts, same

The future: TLSv1.3
(not released)

Fixing security

Removes broken… “stuff”

Limits compatibility

TLS is not enough...

- DUAL EC DRBG
- More bans on crypto

...but it is such a perfect place to start

The problem is key distribution

- Walk to your bank
- Leap of Faith

  (Trust on first sight)
- The web of Trust (GPG)

Relevant PKI Models

- Trust the Government
- Pay for trust
- Assume tech is fair
- Trust something else

x509 certificates

1. Imagine JSON
2. With types and tags
3. from the 80’s

Certificate contents

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            11:21:a2:25:ba:04:02:d7:91:85:48:54:c8:ba:60:68:6a:9b
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256
- G2
        Validity
            Not Before: Dec 10 23:22:05 2015 GMT
            Not After : Dec 10 22:46:04 2016 GMT
        Subject: C=US, ST=California, L=San Francisco, O=Wikimedia Foundation, Inc., CN=*.wi
kipedia.org
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:cb:db:d0:46:41:79:b8:d3:6e:2e:c8:5c:32:d3:
                    f7:82:44:c4:5b:6d:36:51:1b:e6:8f:29:34:92:fe:
                    7c:32:65:8f:23:fd:18:82:b2:35:40:13:fd:7a:c1:
                    ce:c8:3a:da:52:19:93:10:0b:aa:59:1a:f0:f3:8b:
                    ef:dc:7b:0b:fd
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
```

Signature paths

How PKI works (theory)

1. Vendor distributes CA collection to client
2. Server distributes cert and intermediate
3. Client validates the collection

The weakness of PKI

- Paid trust
- Mispaid trust
- Too big to fail
- Forced trust

PKI: Too big to fail

Symantec

Türktrust

~~DigiNotar~~

PKI: Forced trust

- Employers
- Remote admin apps
- Governments*

(*Kazakhstan)

Domain Validation

vs.

Extended Validation

Verification is hard

Randall Munroe, https://xkcd.com/1181/

MODIO
Secure & efficient energy information

The most common failures

“Yey, I got a cert!”

Python 2.7.6

“I didn’t bother to check the signature”

Python, again

“I didn’t bother to check the expiration date”

Python, once more

“I didn’t bother to check the domain name”

It’s tedious now

Deprecation, nigh impossible

SHA1 deprecation has failed

MD5 deprecation almost failed

RC4 is still around

TLS: The defaults suck

```
ssl_certificate /etc/letsencrypt/live/do01.skuggor.se/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/do01.skuggor.se/privkey.pem;
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_session_tickets off;
ssl_dhparam /etc/ssl/private/2048.pem;
ssl_protocols TLSv1.2;
ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-
AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-
SHA256:ECDHE-RSA-AES128-SHA256';
ssl_prefer_server_ciphers on;
add_header Strict-Transport-Security max-age=15768000;
ssl_stapling on;
ssl_stapling_verify on;
ssl_trusted_certificate /etc/letsencrypt/live/do01.skuggor.se/fullchain.pem;
resolver 8.8.8.8;
```

TLS: The defaults suck

SSLContext - Broken defaults

use create_default_context() factory!

Vendors shipping broken tools

OS X: Deprecated OpenSSL

OS X: ships OpenSSL 0.9.8

Python: Bundles outdated OpenSSL

Picking a cipher

ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256

MODIO
Secure & efficient energy information

Legacy: The old configuration

```
PROTOCOL.SSLv23(deprecated)
+ OP.NO_SSLv2
+ OP.NO_SSLv3
```

vs.

```
PROTOCOL.TLS
```

Documentation never dies

~~ssl.wrap_socket()~~

```
ctx = ssl.create_default_context()
ctx.options |= ssl.OP_NO_TLSv1
ctx.options |= ssl.OP_NO_TLSv1_1
ctx.wrap_socket()
WTF!?
```

Upgrading is hard

Python 2.7 is the Windows XP of programming languages

The cost of compatibility

Supporting TLS 1.0 in a new service?

- Do you also run it on Python 1.6?

The Fixes

Configuration:

- Mozilla SSL configurator tool

Documentation:

- ...no easy fixes

Let’s Encrypt

- Trust the machine

MODIO
Secure & efficient energy information

Let’s Encrypt!

```
$ apt-get install letsencrypt
$ letsencrypt certonly -d example.com
```

Let’s Encrypt!

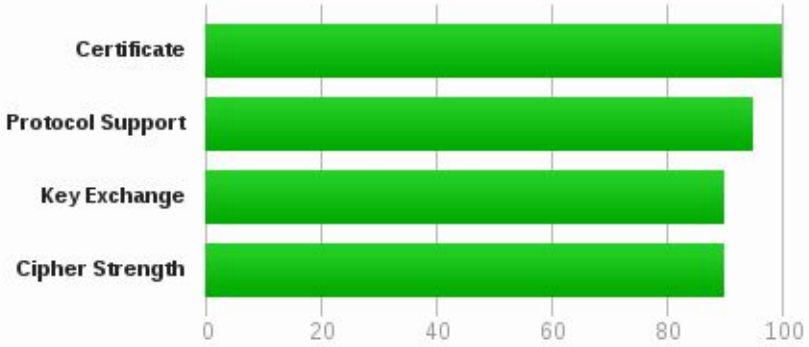
```
import cherrypy

class HelloWorld(object):
        @cherrypy.expose
        def index(self):
        return "Hello world!"

if __name__ == '__main__':
        server_config={
        'server.socket_host': '0.0.0.0',    'server.socket_port': 443,
        'server.ssl_module': 'builtin',
        'server.ssl_certificate': '/etc/letsencrypt/live/do01.skuggor.se/fullchain.pem',
        'server.ssl_private_key': '/etc/letsencrypt/live/do01.skuggor.se/privkey.pem',
        'server.ssl_certificate_chain': '/etc/letsencrypt/live/do01.skuggor.se/fullchain.pem',
        }
        cherrypy.config.update(server_config)
        cherrypy.quickstart(HelloWorld())
```

CherryPy, Ubuntu 16.04

Summary

Overall Rating

A-



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

The server does not support Forward Secrecy with the reference browsers. Grade reduced to A-. MORE INFO »

That’s it for now

Thank You!

Tomorrow, 15:00
“Eliminating application passwords using TLS”

Slides
https://www.modio.se/pages/presentations.html

MODIO
Secure & efficient energy information