TLS “secrets“
What everyone forgot to tell you...

Florent Daignière – Matta Consulting Ltd

Blackhat USA

July 2013

Layout

Who am I?

- Technical Director of a boutique security consultancy firm in London, UK
- One of the few Tiger Scheme trainers
- One of the core developers behind Freenet
- The guy who got a pwnie award last year for exposing the Most Epic FAIL!

Layout

A bit of history...

Versions of the protocol

- ~~SSLv2 : released 1995~~
- SSLv3 : released 1996
- TLSv1 : released 1999
- TLSv1.1 : released 2006
- TLSv1.2 : released 2008

Unless you are stuck with IE6, you are unlikely to be using SSL!

A bit of history...

Versions of the protocol

- ~~SSLv2 : released 1995~~
- SSLv3 : released 1996
- TLSv1 : released 1999
- TLSv1.1 : released 2006
- TLSv1.2 : released 2008

Unless you are stuck with IE6, you are unlikely to be using SSL!

Most likely you are using Transport Security Layer...
Good; this is what my talk is about!

What bad excuses do people find Not to use/deploy SSL?

We are in 2013... but ‘performance‘ seems to remain number one

What bad excuses do people find Not to use/deploy SSL?

We are in 2013... but 'performance' seems to remain number one

Let's look into it...

- Handshaking is expensive (more on this later)
- If there's a high-packet loss it adds significant amount of latency (more round trips)

What bad excuses do people find Not to use/deploy SSL?

We are in 2013... but 'performance' seems to remain number one

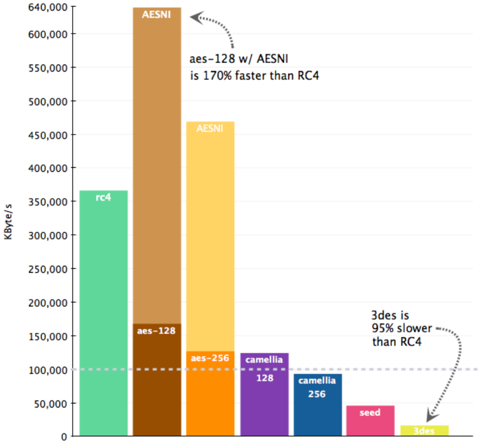
Let's look into it...

- Handshaking is expensive (more on this later)
- If there's a high-packet loss it adds significant amount of latency (more round trips)

Volume doesn't matter... it's symmetric encryption that modern processors do at several times wire-speed!

Performance of symmetric encryption

Cipher choice is of paramount importance!



Performance of the Handshake

No silver bullet. Asymmetric cryptography is expensive.
Whether it's RSA / DSA / ECDSA doesn't make much difference
Keysize does... but it would be unwise to optimize too much...

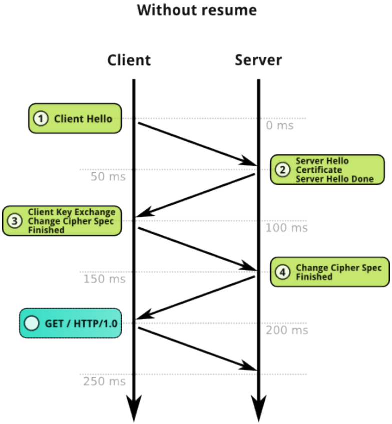
Performance of the Handshake

No silver bullet. Asymmetric cryptography is expensive.
Whether it's RSA / DSA / ECDSA doesn't make much difference
Keysize does... but it would be unwise to optimize too much...

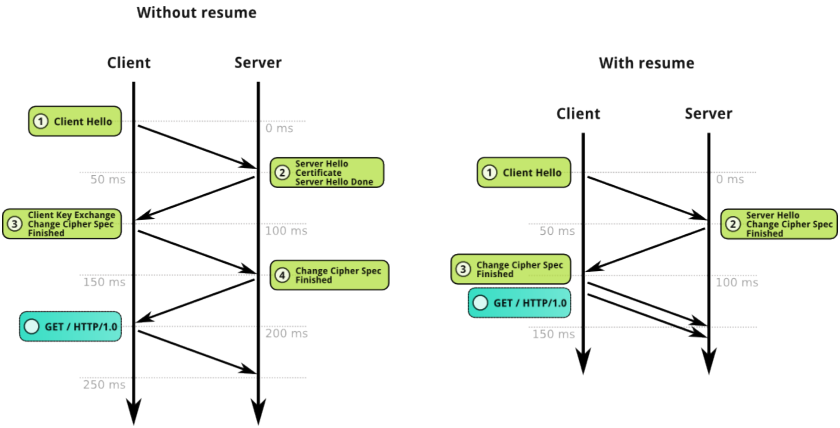
The solution?

Handshake once... and resume sessions (using an abbreviated handshake) where possible!

SSL Session resumption



SSL Session resumption



How does it work?

For SSL and basic TLS

You get a session-id... that you present on each re-connection

```
Terminal
File  Edit  View  Search  Terminal  Help
---
SSL handshake has read 1979 bytes and written 442 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: CBA4FFFE0B8C5532ABA9DE52B4804140ED2C7CC91460C62A7DAF201859029561
    Session-ID-ctx:
    Master-Key: FA38301254FE485F4F943E17EB0651704D123B0B9018E45E52164197B251BA0E
21408532105C4E25938E80FF9DB32E22
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1373561258
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
```

TLS Session tickets - RFC 5077

What if we made it stateless?

- Store an arbitrary-sized, encrypted blob stored client-side

TLS Session tickets - RFC 5077

What if we made it stateless?

- Store an arbitrary-sized, encrypted blob stored client-side

RFC to the rescue!

4. Recommended Ticket Construction

This section describes a recommended format and protection for the ticket. Note that the ticket is opaque to the client, so the structure is not subject to interoperability concerns, and implementations may diverge from this format. If implementations do diverge from this format, they must take security concerns seriously. Clients MUST NOT examine the ticket under the assumption that it complies with this document.

The server uses two different keys: one 128-bit key for Advanced Encryption Standard (AES) [AES] in Cipher Block Chaining (CBC) mode [CBC] encryption and one 256-bit key for HMAC-SHA-256 [RFC4634].

The ticket is structured as follows:

```
struct {
    opaque key_name[16];
    opaque iv[16];
    opaque encrypted_state<0..2^16-1>;
    opaque mac[32];
} ticket;
```

Here, key_name serves to identify a particular set of keys used to protect the ticket. It enables the server to easily recognize tickets it has issued. The key_name should be randomly generated to avoid collisions between servers. One possibility is to generate new random keys and key_name every time the server is started.

The actual state information in encrypted_state is encrypted using 128-bit AES in CBC mode with the given IV. The Message Authentication Code (MAC) is calculated using HMAC-SHA-256 over key_name (16 octets) and IV (16 octets), followed by the length of

RFC 5077 - what does it look like?

For SSL and basic TLS

You get a blob... that you present on each re-connection

Terminal

File Edit View Search Terminal Help

```
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES128-GCM-SHA256
    Session-ID: E9DC5C892D78E0F45D04385AA302A1BE0EEFC34840A75CDF06E4AD06E6CEE2FC
    Session-ID-ctx:
    Master-Key: 122614C9FA1901141B021FBE1CD3C726EB34E33A716B8CA6C5C9FCEBE28D662A
4FD9788178E16BABD8BD1CAF3BCFDA71
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 100800 (seconds)
    TLS session ticket:
    0000 - d7 bf 2b f9 fb b1 71 c1-31 ea 5d 98 09 15 0c 83   ..+...q.1.].....
    0010 - df b5 88 09 fd 84 45 e4-e7 e1 dc f8 3e 94 6a 6b   ......E.....>.jk
    0020 - 04 6f 64 6f 6f 15 f9 ce-e8 83 96 27 13 5e 7c 3c   .odoo......'.^|<
    0030 - 7d c0 7f 56 10 7f 7f 5e-24 62 23 f7 76 19 b8 61   }..V...^$b#.v..a
    0040 - 56 e7 db 99 56 e7 c4 29-a0 e4 da c7 5b de b5 89   V...V..)....[...
    0050 - 87 3b ae 7f 5e f2 39 5c-46 83 37 0b 4f 27 42 f5   .;..^.9\F.7.O'B.
    0060 - 7d c4 42 84 2a cf 22 30-2b 6b 8c 76 d0 a0 3f 1a   }.B.*."0+k.v..?.
    0070 - 4c cc a6 3c 8b cc b5 7d-84 9e 7a b7 52 59 78 06   L..<...}..z.RYx.
    0080 - b2 52 e2 4a 0f 55 8e 6a-f6 e6 c9 d8 18 b6 54 13   .R.J.U.j......T.
    0090 - 80 4d 82 fb                                       .M..

    Start Time: 1373561537
    Timeout   : 300 (sec)
```

Layout

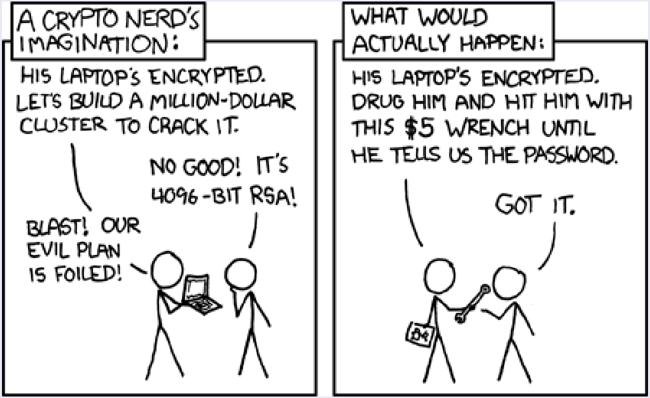
1. Introduction
   - Who am I?
   - Secure Socket Layer
   - Forward secrecy
2. Where it all goes wrong...
   - Chosen extracts of the RFC
   - OpenSSL's case
   - What about applications?
   - With the tin-foil hat on
3. Here comes the Tool
4. Conclusion

What is forward secrecy?

What is forward secrecy?

- Attacker cannot decrypt a conversation even if he records the entire session and subsequently steals their associated long-term secrets
- The session keys are not derivable from information stored after the session concludes

Why would you want forward secrecy?



Where do you have no forward secrecy? (whereas you should!)

Where do you have no forward secrecy? (whereas you should!)

- Browsing the internet (more on this later)
- WiFi (WPA-PSK / WPA-EAP-tunnel)
- Cell phones (2G/3G/4G)
- ... everywhere?

How do you get Forward Secrecy?

How do you get forward secrecy?

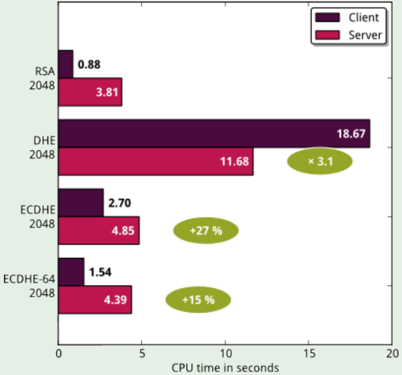
Using a Diffie-Hellman construct!

How do you get Forward Secrecy?

How do you get forward secrecy?

Using a Diffie-Hellman construct!

How much does it cost?



Layout

Chosen extracts of the RFC

5. Security Considerations

5.5. Ticket Protection Key Management

A full description of the management of the keys used to protect the ticket is beyond the scope of this document. A list of RECOMMENDED practices is given below.

- o The keys should be generated securely following the randomness recommendations in [RFC4086].
- o The keys and cryptographic protection algorithms should be at least 128 bits in strength. Some ciphersuites and applications may require cryptographic protection greater than 128 bits in strength.
- o The keys should not be used for any purpose other than generating and verifying tickets.
- o The keys should be changed regularly.
- o The keys should be changed if the ticket format or cryptographic protection algorithms change.

"beyond the scope of this document"?!?

Chosen extracts of the RFC (cont)

5. Security Considerations

5.6. Ticket Lifetime

The TLS server controls the lifetime of the ticket. Servers determine the acceptable lifetime based on the operational and security requirements of the environments in which they are deployed. The ticket lifetime may be longer than the 24-hour lifetime recommended in [RFC4346]. TLS clients may be given a hint of the lifetime of the ticket. Since the lifetime of a ticket may be unspecified, a client has its own local policy that determines when it discards tickets.

"The ticket lifetime may be longer than the 24-hour..."

Layout

1. Introduction
   - Who am I?
   - Secure Socket Layer
   - Forward secrecy
2. Where it all goes wrong...
   - Chosen extracts of the RFC
   - OpenSSL's case
   - What about applications?
   - With the tin-foil hat on
3. Here comes the Tool
4. Conclusion

OpenSSL won't keep you safe!

How do they do it?

- Tickets are enabled by default
- Encrypted using AES128-CBC
- Keys are stored in the SSL_CTX
- No rekeying

OpenSSL won't keep you safe!

How do they do it?

- Tickets are enabled by default
- Encrypted using AES128-CBC
- Keys are stored in the SSL_CTX
- No rekeying

What does it mean?

- No point in using anything fancier than AES128-CBC!
- Your PFS interval is the program's lifetime!

Layout

What about applications?

nginx

PFS interval is the program lifespan

Haha, but I use Apache!

What about applications?

nginx

PFS interval is the program lifespan

Haha, but I use Apache!

Apache HTTPd

PFS interval is :
* pre r1200040 the program lifespan
* post r1200040 the user is in charge of key management!

Vendors don't care; do you?

What about 'sensitive' applications?

Tor's case

Yes, Tor is affected.

Ephemeral long-term keys (rotating certificates)

... that's the PFS interval, unless ...

What about 'sensitive' applications?

Tor's case

Yes, Tor is affected.
Ephemeral long-term keys (rotating certificates)
... that's the PFS interval, unless ...
You keep a circuit alive on the relay you target.
In which case, you can keep the SSL_CTX in memory forever

What about 'sensitive' applications?

Tor's case

Yes, Tor is affected.
Ephemeral long-term keys (rotating certificates)
... that's the PFS interval, unless ...
You keep a circuit alive on the relay you target.
In which case, you can keep the SSL_CTX in memory forever

1) Connect to all relays you want to bust
2) Repeat (but don't rinse) every MAX_SSL_KEY_LIFETIME_INTERNAL (2h)
3) Bust the operators/relays, get the keys, decrypt the traffic.

What about 'sensitive' applications?

Tor's case

Yes, Tor is affected.
Ephemeral long-term keys (rotating certificates)
... that's the PFS interval, unless ...
You keep a circuit alive on the relay you target.
In which case, you can keep the SSL_CTX in memory forever

1) Connect to all relays you want to bust
2) Repeat (but don't rinse) every MAX_SSL_KEY_LIFETIME_INTERNAL (2h)
3) Bust the operators/relays, get the keys, decrypt the traffic.

One layer of the onion is gone; two to go!

Layout

How does that affect me?

Website	seconds	1h	24h	48h
www.facebook.com	Y	Y	N	N
www.google.com	Y	Y	Y	N
www.youtube.com	Y	Y	Y	N
www.wikipedia.org	Y	Y	N	N
www.twitter.com	N			
www.wikileaks.org	N			
www.yahoo.com	N			
www.fbi.gov	N			
www.royal.gov.uk	N			

Wouldn't having the key of tickets be convenient?

Key management

How would someone go about stealing the secret?

Well, it depends on who you are I guess.

Key management

How would someone go about stealing the secret?

Well, it depends on who you are I guess.

If you are the government

You just ask politely...
And should your request be politely declined...
you use a PRISM to “see“ it through the interwebz! ;)

Key management

How would someone go about stealing the secret?

Well, it depends on who you are I guess.

If you are the government

You just ask politely...
And should your request be politely declined...
you use a PRISM to “see“ it through the interwebz! ;)

If you are not the government

You can ask your mate who is in the planet-alignment-business to give you one of his “useless“ memory disclosure bugs.
Odds are he has plenty, as it's now pretty much required to get reliable exploitation.

Key management

If you don't have a mate doing exploitation...

Well, you must be LEO then.

Key management

If you don't have a mate doing exploitation...

Well, you must be LEO then.
Jokes aside, you can do forensics and my tool can probably help you.

Demo

Demo time...

...

How does it work?

Demo

Demo time...

...

How does it work?

Using and abusing PTRACE to extract the master encryption key;

Demo

Demo time...

...

How does it work?

Using and abusing PTRACE to extract the master encryption key;
Allowing to decrypt the session tickets sent over the wire...

Demo

Demo time...

...

How does it work?

Using and abusing PTRACE to extract the master encryption key;
Allowing to decrypt the session tickets sent over the wire...
Which in turn contain the Master Session Key allowing to
derive the key used to decrypt the cipher text and
recover the plaintext.

Conclusion and take-aways

If you are an auditor

You shouldn't focus on getting people to use a cipher strength providing more than 128 bits of security.

If you are a pentester

You should learn to use and abuse SSL to bypass "intermediary" devices preventing you from doing your job.

If you are a end-user

You might want to reconfigure your clients and disable RFC5077 support.

References

- https://tools.ietf.org/html/rfc5077
- http://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077.html
- https://www.eff.org/deeplinks/2011/11/long-term-privacy-forward-secrecy
- http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html
- http://zombe.es/post/4078724716/openssl-cipher-selection
- https://issues.apache.org/bugzilla/show_bug.cgi?id=50869
- https://httpd.apache.org/docs/trunk/mod/mod_ssl.html#sslsessiont
- https://trac.torproject.org/projects/tor/ticket/7139

Any questions?

Thank you!

I blog at http://blog.trustmatta.com
and tweet at @nextgens1
You can find the source-code of the tool at
https://github.com/nextgens/

Important!

Please don't forget to fill in the feedback form!