SSL Checklist for Pentesters

Jerome Smith

BSides MCR, 27th June 2014

whoami

```
# whoami
jerome
```

- Pentester
- Author/trainer
  - Hands-on technical
  - Web application, infrastructure, wireless security
- Security projects
  - Log correlation
  - Dirty data
  - Incident response exercises
- Sysadmin
- MSc Computing Science (Dist)
- www.exploresecurity.com | @exploresecurity

Introduction

- Broad review of SSL/TLS checks
  - Viewpoint of pentester
  - Pitfalls
  - Manually replicating what tools do (unless you told the client that *SSL Labs* would be testing them ☺)
  - Issues to consider reporting (but views are my own)
- While SSL issues are generally low in priority, it's nice to get them right!
- I'm not a cryptographer: this is all best efforts

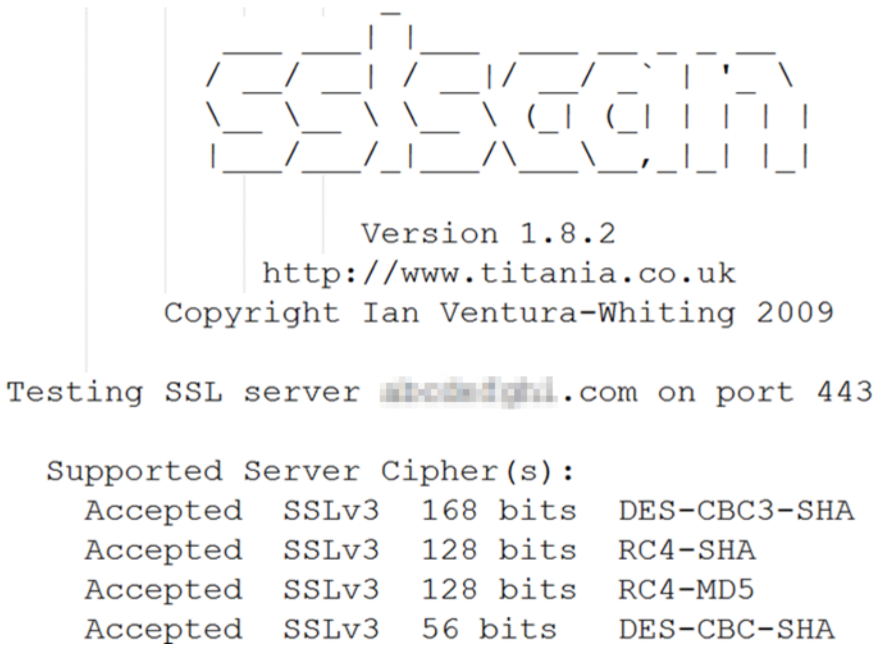
freedom from doubt

SSLv2

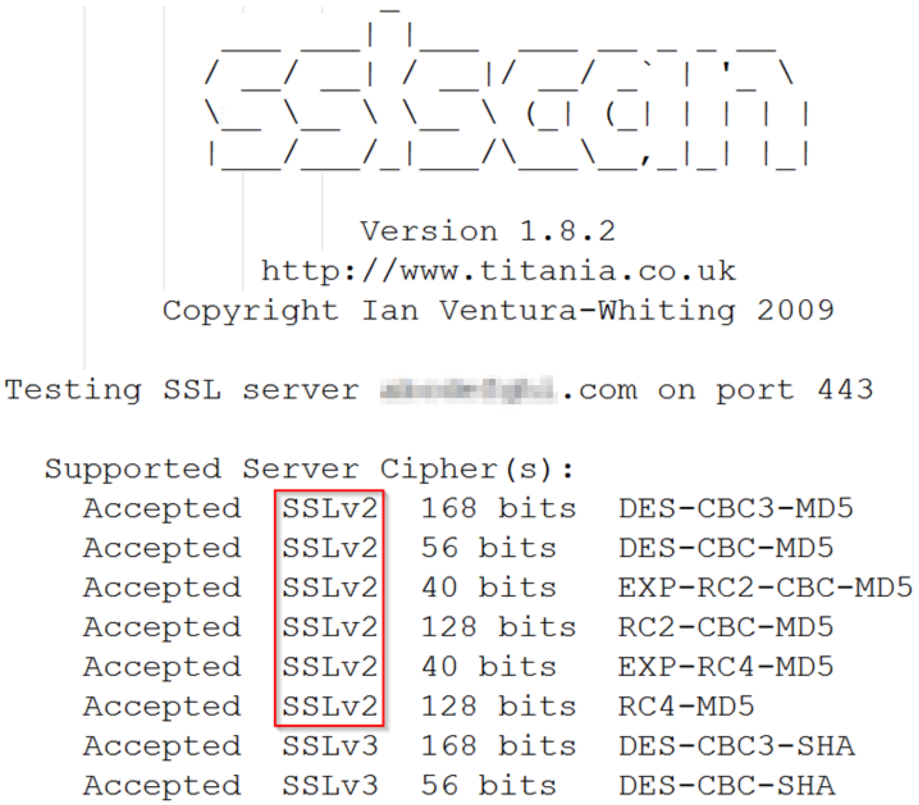
- Flawed, e.g. no handshake protection → MITM downgrade
- Modern browsers do not support SSLv2 anyway
  - Except for IE but it's disabled by default from IE7
  - That mitigates the risk these days
  - http://en.wikipedia.org/wiki/Transport_Layer_Security#Web_browsers
- OpenSSL 1.0.0+ doesn't support it
  - Which means SSLscan won't find it
  - General point: tools that dynamically link to an underlying SSL library in the OS can be limited by what that library supports

SSLv2

- Same scan on different OpenSSL versions:



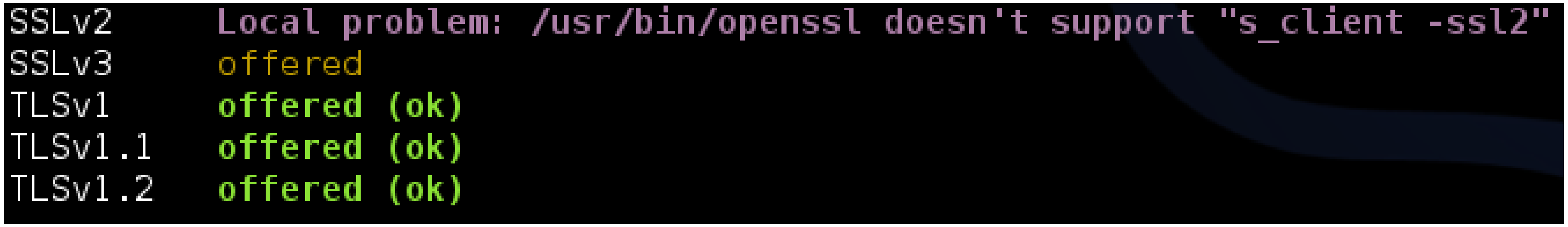
```
                   _
           ___ ___| |___  ___ __ _ _ __
          / __/ __| / __|/ __/ _` | '_ \
          \__ \__ \ \__ \ (_| (_| | | | |
          |___/___/_|___/\___\__,_|_| |_|

                  Version 1.8.2
             http://www.titania.co.uk
        Copyright Ian Ventura-Whiting 2009

Testing SSL server [redacted].com on port 443

  Supported Server Cipher(s):
    Accepted  SSLv3  168 bits  DES-CBC3-SHA
    Accepted  SSLv3  128 bits  RC4-SHA
    Accepted  SSLv3  128 bits  RC4-MD5
    Accepted  SSLv3  56 bits   DES-CBC-SHA
```



```
                   _
           ___ ___| |___  ___ __ _ _ __
          / __/ __| / __|/ __/ _` | '_ \
          \__ \__ \ \__ \ (_| (_| | | | |
          |___/___/_|___/\___\__,_|_| |_|

                  Version 1.8.2
             http://www.titania.co.uk
        Copyright Ian Ventura-Whiting 2009

Testing SSL server [redacted].com on port 443

  Supported Server Cipher(s):
    Accepted  SSLv2  168 bits  DES-CBC3-MD5
    Accepted  SSLv2  56 bits   DES-CBC-MD5
    Accepted  SSLv2  40 bits   EXP-RC2-CBC-MD5
    Accepted  SSLv2  128 bits  RC2-CBC-MD5
    Accepted  SSLv2  40 bits   EXP-RC4-MD5
    Accepted  SSLv2  128 bits  RC4-MD5
    Accepted  SSLv3  168 bits  DES-CBC3-SHA
    Accepted  SSLv3  56 bits   DES-CBC-SHA
```

SSLv2

- `testssl.sh` warns you

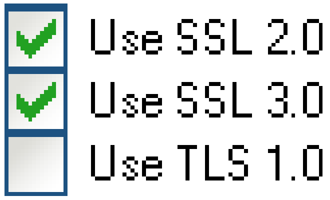


```
SSLv2      Local problem: /usr/bin/openssl doesn't support "s_client -ssl2"
SSLv3      offered
TLSv1      offered (ok)
TLSv1.1    offered (ok)
TLSv1.2    offered (ok)
```

  - It can work with any installed OpenSSL version
- OpenSSL <1.0.0 `s_client -ssl2` switch
  - More on this later
- Recompile OpenSSL
  - http://blog.opensecurityresearch.com/2013/05/fixing-sslv2-support-in-kali-linux.html
- SSLyze 0.7+ is statically linked
  - Watch out for bug https://github.com/iSECPartners/sslyze/issues/73

SSLv3

- SSLv3 RFC is in fact “historical”
  - TLS began in 1999 as the standardised version of SSL
  - The SSL protocol hasn’t been updated since 1996
- We really shouldn’t be running it
  - Everything supports TLSv1.0 by default
  - Oh, except IE6



[✓] Use SSL 2.0
[✓] Use SSL 3.0
[ ] Use TLS 1.0

  - Should we take the lead and begin to flag it in pentests as obsolete?

TLSv1.1 & v1.2

- We really should be running it
  - They've been around since 2006 and 2008 respectively
- Latest versions of browsers support them (platform dependent)
  - http://en.wikipedia.org/wiki/Transport_Layer_Security#Web_browsers
  - http://en.wikipedia.org/wiki/Comparison_of_TLS_implementations
- Check support – and report their absence?
  - Missing out on more robust design and better ciphers
  - Again, trying to push in the right direction
- SSLscan isn't designed to check for these versions
  - `openssl s_client -tls1_1` or `-tls1_2` switch (from v1.0.1)

TLSv1.1 & v1.2

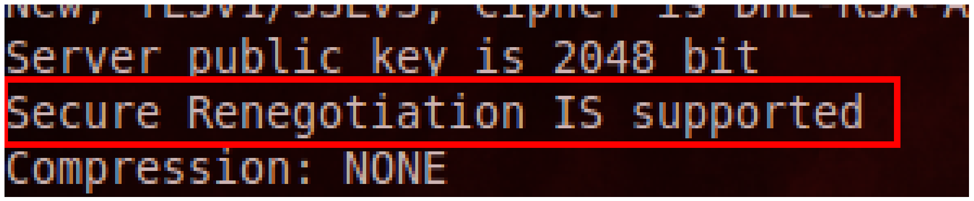
- Not immune
- While the protocol handshake is protected, browsers have fall-back mechanisms or performance tricks (e.g. False Start) that could be abused
  - To be fair, Google later abandoned False Start
  - http://blog.cryptographyengineering.com/2012/04/so-long-false-start-we-hardly-knew-ya.html
- A MITM attacker could trigger a protocol downgrade
  - Possibly all the way down to SSLv3
  - https://www.imperialviolet.org/2013/10/07/chacha20.html
  - http://www.carbonwind.net/blog/post/Random-SSLTLS-101%E2%80%93SSLTLS-version-rollbacks-and-browsers.aspx
- But benefits far outweigh this annoyance

Renegotiation Checks

- Insecure renegotiation flaw CVE-2009-3555
- Both insecure and secure renegotiation may be supported
  - Especially if multiple servers are behind the hostname
- Manual check

```
openssl s_client -connect site:port
HEAD / HTTP/1.0
R
```

  - If no error, renegotiation is supported: whether it's insecure or secure will depend on the OpenSSL version
  - Add a final CRLF to prove the request completes
- OpenSSL 0.9.8m+ won't renegotiate insecurely
  - Conversely v0.9.8k and older won't renegotiate securely
  - So how can BT5R3's OpenSSL 0.9.8k state this?:

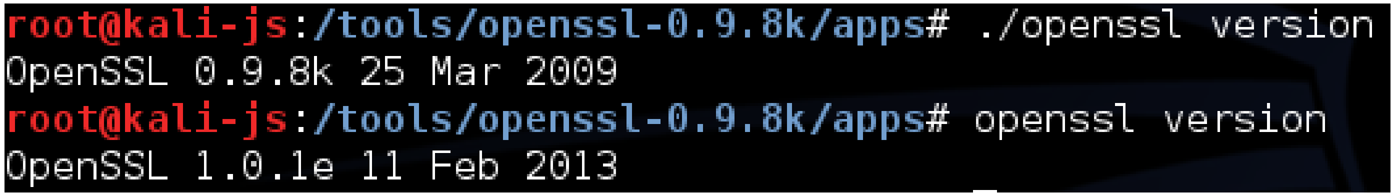


```
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
```

OpenSSL

- It's useful to have an older OpenSSL around
  - But you don't want it to clash with your main version
- Download e.g. https://www.openssl.org/source/openssl-0.9.8k.tar.gz to /tools
- Run

```
      cd /tools
      tar -xzf openssl-0.9.8k.tar.gz
      cd openssl-0.9.8k
      ./config --prefix=/tools/openssl-0.9.8k --
openssldir=/tools/openssl-0.9.8k
      make
```



```
root@kali-js:/tools/openssl-0.9.8k/apps# ./openssl version
OpenSSL 0.9.8k 25 Mar 2009
root@kali-js:/tools/openssl-0.9.8k/apps# openssl version
OpenSSL 1.0.1e 11 Feb 2013
```

Renegotiation Checks

- Client-initiated renegotiation
  - Wholly different issue to secure vs insecure
  - Potential DoS attack
    - http://www.thc.org/thc-ssl-dos/
    - Although renegotiation isn't a prerequisite, it helps
  - If client-initiated renegotiation is disabled, insecure renegotiation is not exploitable
    - So clients may do this to "fix" CVE-2009-3555
  - Only really needed for client certificate authentication
- So what does it mean if renegotiation works?

	<= 0.9.8k	>= 0.9.8m
Insecure	Yes	No
Client-initiated	Yes	Yes

Certificate Checks

Public key size

- <1,024-bit – vulnerable (ish)
  - RSA-768 was factored
    - http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-challenge-numbers.htm
  - 512-bit Google key cracked to spoof email
    - "72 hours using Amazon Web Services for £47"
    - http://www.wired.co.uk/news/archive/2012-10/25/google-email
- 1,024-bit – upgrade to 2,048-bit
  - Any larger increases overhead with no real security benefit (yet)

Valid certificate chain

- Note that different browsers hold different sets of root CAs
- Tip: don't report "the certificate was signed by an untrusted root CA *PortSwigger*" ☺

Certificate Checks

Expiry date

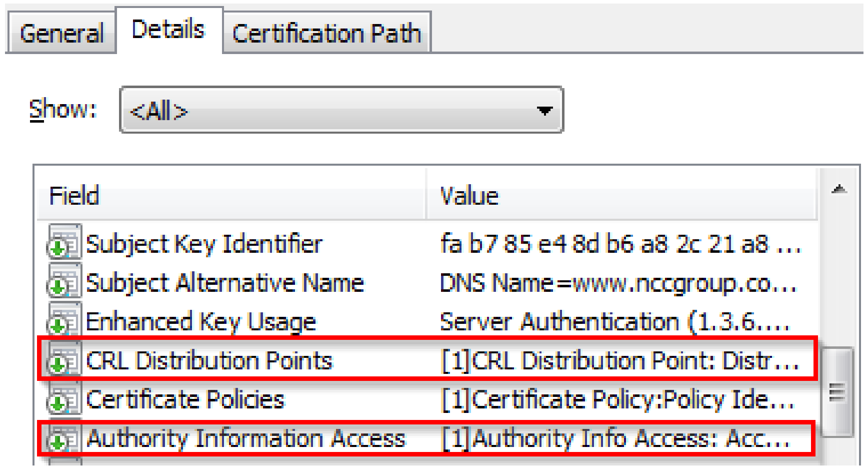
- Warn of imminent expiry

Signature

- Hashed using MD5 (certificate spoofing 2008)

Revocation

- Does the certificate hold CRL/OCSP (AIA) fields?



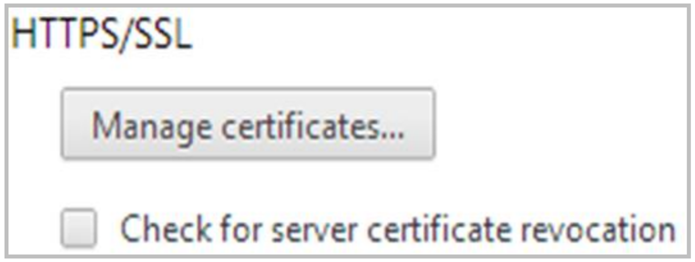
Field	Value
Subject Key Identifier	fa b7 85 e4 8d b6 a8 2c 21 a8 ...
Subject Alternative Name	DNS Name=www.nccgroup.co...
Enhanced Key Usage	Server Authentication (1.3.6....
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Certificate Policies	[1]Certificate Policy:Policy Ide...
Authority Information Access	[1]Authority Info Access: Acc...

nccgroup
freedom from doubt

Certificate Checks

Revocation (cont)

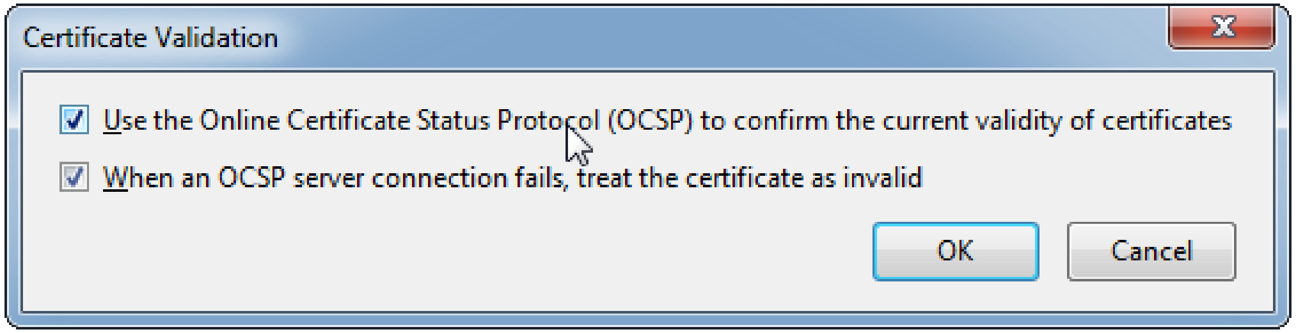
- Are those fields valid?
  - Is your browser checking them?
  - Chrome doesn't by default →
  - Does it fail open?
  - Revocation check for a pentest requires hard fail
  - Firefox Tools | Options | Advanced | Certificates



HTTPS/SSL

Manage certificates...

☐ Check for server certificate revocation



Certificate Validation

☑ Use the Online Certificate Status Protocol (OCSP) to confirm the current validity of certificates

☑ When an OCSP server connection fails, treat the certificate as invalid

OK Cancel

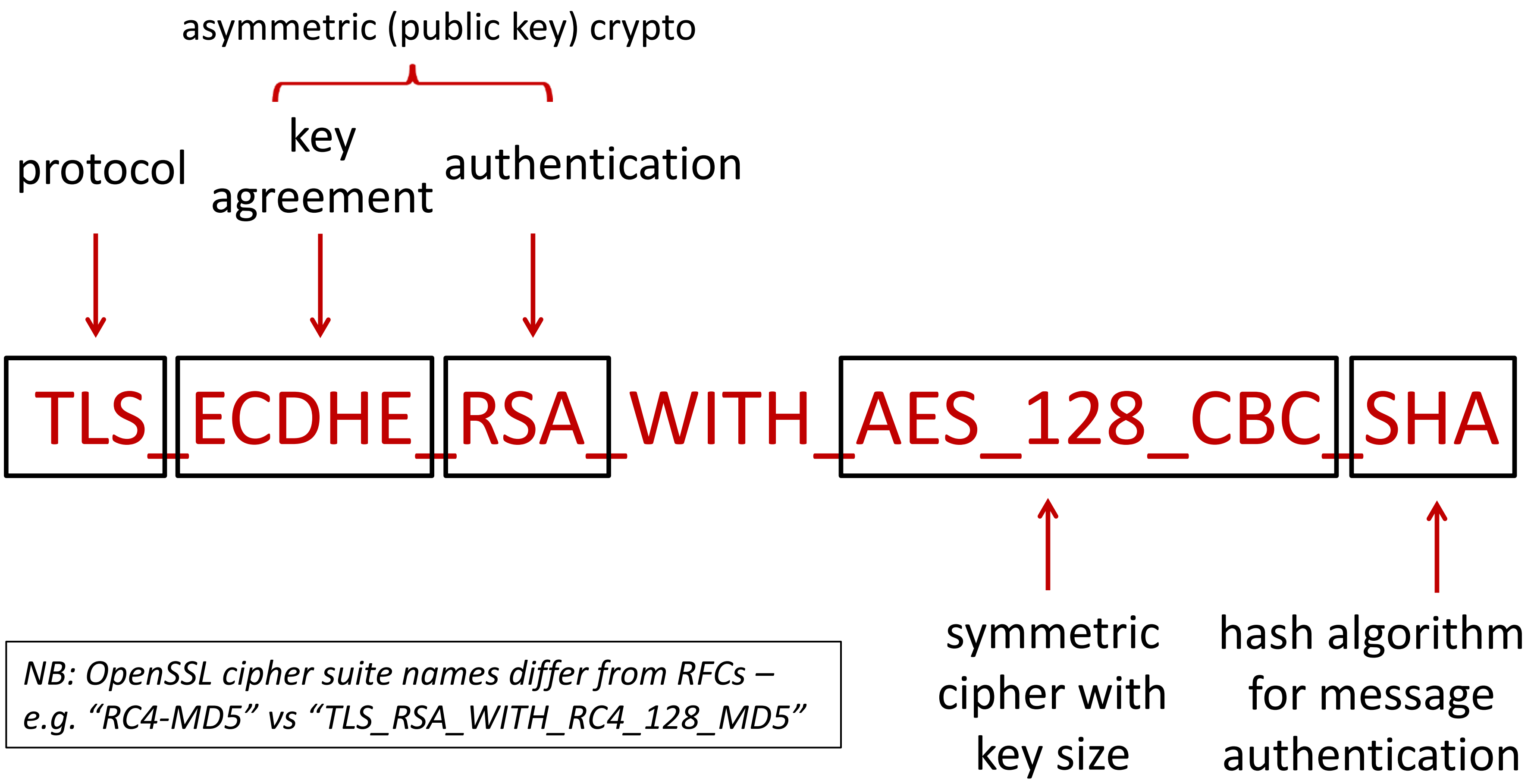
  - Ah, but what about the full certificate chain?

Certificate Checks

Certificate Subject

- Valid for all resource requests?
  - Users often miss off www – is the certificate still valid?
- Wildcard certificate
  - *.domain.com not valid for https://domain.com
  - Encourages more widespread use of powerful certificate
  - Provides SSL confidence to links exploiting other flaws (especially if wildcard DNS enabled)
- Subject Alternative Names
  - Can be used for other domains, not just the host's
  - Some tools may not check this field correctly → false positive "certificate name does not match hostname"

Cipher Suites



asymmetric (public key) crypto
protocol
key agreement
authentication
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
symmetric cipher with key size
hash algorithm for message authentication

NB: OpenSSL cipher suite names differ from RFCs – e.g. "RC4-MD5" vs "TLS_RSA_WITH_RC4_128_MD5"

Cipher Suites

- <128-bit keys can be brute-forced (ish)
  - Not trivial: decent hardware < 1 day for DES
    - e.g. RIVYERA S3-5000
    - http://www.voltage.com/blog/crypto/rivyera-from-sciengines/
  - 3DES provides an effective key strength of 112 bits
    - http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf
    - And it's relatively slow
- Less likely to see:
  - Anonymous Diffie-Hellman (lacks authentication)
  - NULL cipher suites (lacks encryption)
  - "Export" ciphers (unlike with beer, this label is bad)
  - Unlikely to be supported by browser anyway

Cipher Suites

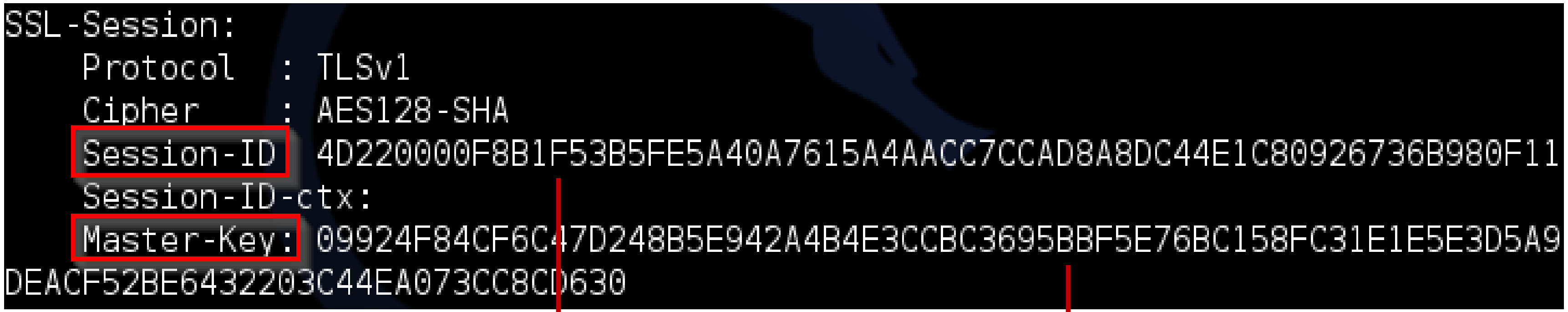
- Server preference
  - Client sends list of cipher suites in order of preference
  - Server will choose the first one it supports unless it has a preference
- `openssl s_client -cipher` switch
  - Pick specific cipher suites (sent in order) or groups
  - `man ciphers` or https://testssl.sh/openssl-rfc.mappping.html
  - Replicate server preference check by switching order of ciphers
- (Perfect) Forward secrecy
  - Without it, private key compromise means previous traffic can be read
  - Look for "ephemeral" in the key agreement part, e.g. DH**E**, ECDH**E**
  - Adds a cost (ECDHE is faster than DHE)
- Latest cipher suites – require TLSv1.2
  - AES-GCM (slow), ChaCha20/Poly1305 (new)
  - https://www.imperialviolet.org/2013/10/07/chacha20.html

RC4

- First bytes of ciphertext are cryptographically weak
- http://www.isg.rhul.ac.uk/tls/RC4biases.pdf
  - To attack cookies, it would take 2,000 hours (short of 3 months)
  - "It would be incorrect to describe the attacks as being a practical threat to TLS…*today*" (my emphasis)
- "Attacks always get better, they never get worse"
  - Sensible to phase out RC4 ciphers
  - Even Microsoft has done it with KB2868725
- IETF draft advisory
  - http://tools.ietf.org/html/draft-popov-tls-prohibiting-rc4-02
  - "TLS clients MUST NOT include RC4 cipher suites"
  - "TLS servers MUST NOT select an RC4 cipher suite"

Revealing SSL/TLS

- Most of the handshake is in clear text
- To decrypt traffic in Wireshark, run with OpenSSL:



```
SSL-Session:
    Protocol  : TLSv1
    Cipher    : AES128-SHA
    Session-ID  4D220000F8B1F53B5FE5A40A7615A4AACC7CCAD8A8DC44E1C80926736B980F11
    Session-ID-ctx:
    Master-Key: 09924F84CF6C47D248B5E942A4B4E3CCBC3695BBF5E76BC158FC31E1E5E3D5A9
DEACF52BE6432203C44EA073CC8CD630
```

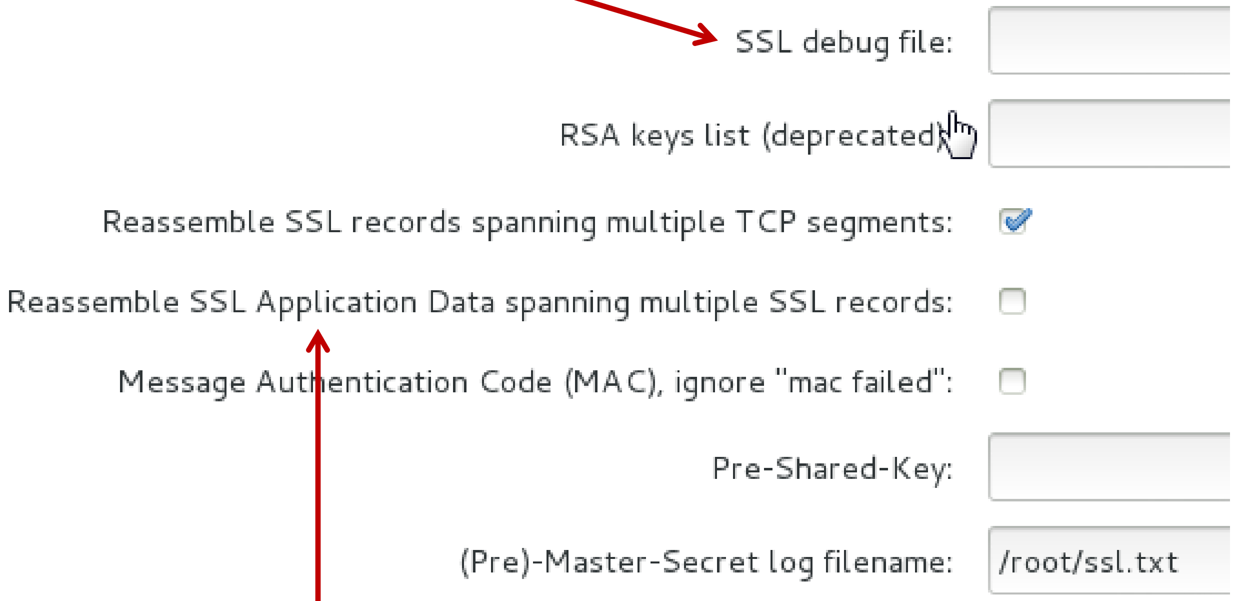
  - Create text file:

```
RSA Session-ID:<Session-ID> Master-Key:<Master-Key>
```

  - The NSS SSL/TLS stack (Chrome & Firefox) can auto-create this file: set the environment variable SSLKEYLOGFILE to the path of a text file
- Wireshark
  - Edit | Preferences | Protocols | SSL
  - Set “(Pre-)Master-Secret log filename” to file above

Revealing SSL/TLS

- Wireshark SSL preferences
  - Everything you (n)ever wished to know



SSL debug file:	
RSA keys list (deprecated):	
Reassemble SSL records spanning multiple TCP segments:	✓
Reassemble SSL Application Data spanning multiple SSL records:	☐
Message Authentication Code (MAC), ignore "mac failed":	☐
Pre-Shared-Key:	
(Pre)-Master-Secret log filename:	/root/ssl.txt

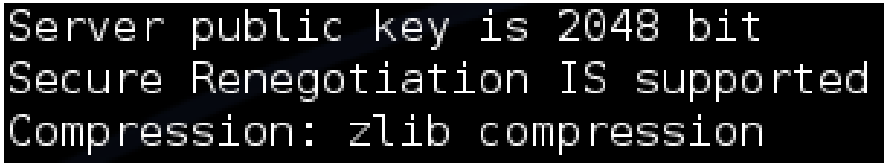
  - "Records" are a sub-layer – and if you see a starting record of data that's empty or has one byte, that's because of...

BEAST

- A client-side attack but it's nice to help your users
  - TLS 1.0 or less with block ciphers in CBC mode = vulnerable server
- All recent major browsers have a patch
  - Apple finally woke up in Nov 2013
  - But not every user will be running the latest version
- TLSv1.1 and v1.2 aren't vulnerable to BEAST
  - But recall browser downgrade attack
- Alternative is to prefer RC4
  - But we've said not to use it!
  - RC4 flaws are systemic: BEAST attack surface will diminish
- So should we let it go now?
  - Confusing for client to have both RC4 and BEAST reported?
  - https://community.qualys.com/blogs/securitylabs/2013/09/10/is-beast-still-a-threat

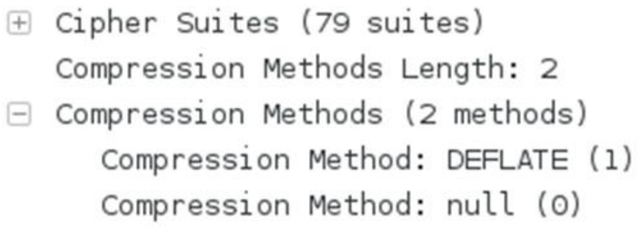
CRIME

- Targets SSL compression (again, client-side attack)
  - Only Chrome really supported it, it's now disabled
  - So it's unlikely to be exploitable
- Check for compression on server with OpenSSL



```
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: zlib compression
```

  - Don't rely on "Compression: NONE" message until you've checked your OpenSSL version supports compression
  - Look at Client Hello:

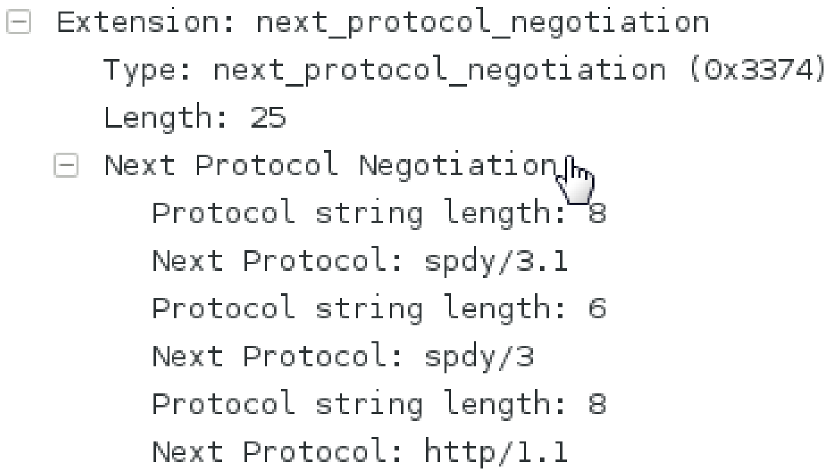


```
⊞ Cipher Suites (79 suites)
  Compression Methods Length: 2
⊟ Compression Methods (2 methods)
    Compression Method: DEFLATE (1)
    Compression Method: null (0)
```

  - To enable, build with `./config zlib zlib-dynamic`

CRIME

- SPDY uses compression to “make the web faster”
  - It sits between HTTP and TLS: it’s similarly vulnerable
  - BREACH: HTTP has native compression – same issue
- `openssl s_client -nextprotoneg NULL`
  - Connection will fail but look at Server Hello:

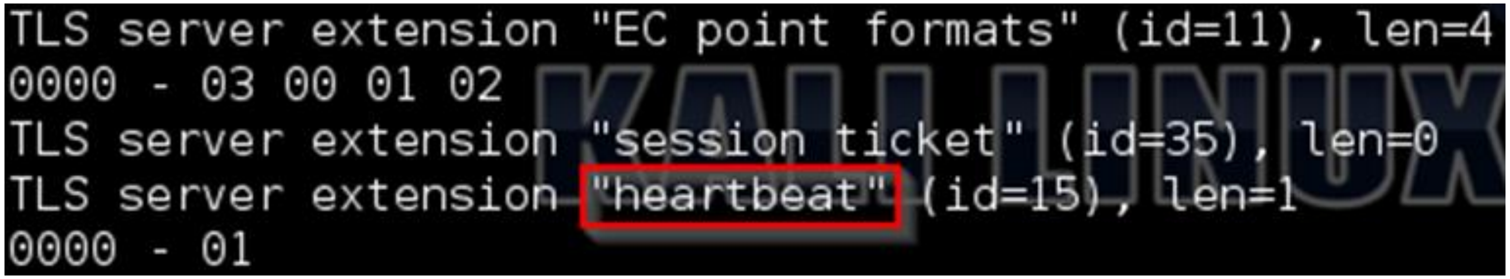


```
⊟ Extension: next_protocol_negotiation
    Type: next_protocol_negotiation (0x3374)
    Length: 25
  ⊟ Next Protocol Negotiation
      Protocol string length: 8
      Next Protocol: spdy/3.1
      Protocol string length: 6
      Next Protocol: spdy/3
      Protocol string length: 8
      Next Protocol: http/1.1
```

  - Future TLS extension “Application Layer Protocol Negotiation”

Heartbleed

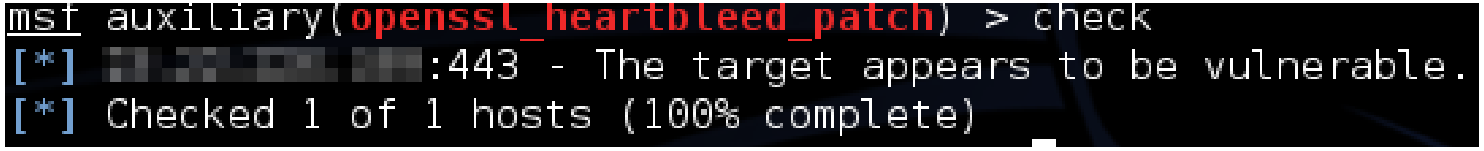
- OpenSSL 1.0.1 - 1.0.1f (and 1.0.2-beta1)
- PoC and first-gen tools raced out
  - Testing could lead to compromise of sensitive data and/or potentially crash the service
  - https://blog.mozilla.org/security/2014/04/12/testing-for-heartbleed-vulnerability-without-exploiting-the-server/
- Vulnerability analysis vs pentesting
  - More prone to false negatives
- Easy to check if Heartbeat enabled
  - Connect using OpenSSL 1.0.1+ with `s_client -tlsextdebug`



```
TLS server extension "EC point formats" (id=11), len=4
0000 - 03 00 01 02
TLS server extension "session ticket" (id=35), len=0
TLS server extension "heartbeat" (id=15), len=1
0000 - 01
```

Heartbleed

- Obviously `openssl s_client` can't be used to test
- Tools
  - *heartbleeder* from Titanous
  - MDSec's `heartbleed -s <target> -p 443 -f out` **`-t 0`**
  - Metasploit
    - Core *openssl_heartbleed* module is greedy even using "check"
    - Try the module from the previous Mozilla article

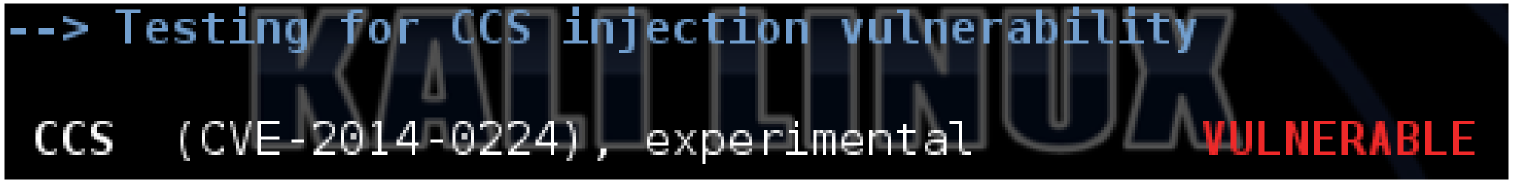


```
msf auxiliary(openssl_heartbleed_patch) > check
[*] [redacted]:443 - The target appears to be vulnerable.
[*] Checked 1 of 1 hosts (100% complete)
```

  - HP iLO/iLO2 products locked up (not vulnerable anyway!)
- Nice GUI tool from CrowdStrike (aggressive)

Change Cipher Spec (CCS)

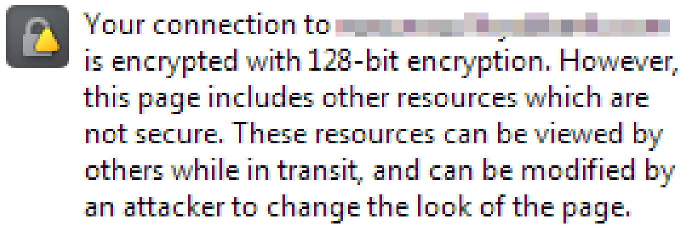
- CVE-2014-0224 MITM attack to force weak keys
- Only exploitable if:
  - Server uses OpenSSL <1.0.1h
  - Client uses OpenSSL <1.0.1h, <1.0.0m, <0.9.8za
    - So that's only Android as far as browsers are concerned
- Tools
  - Metasploit *openssl_ccs* module
  - `./testssl.sh --ccs`

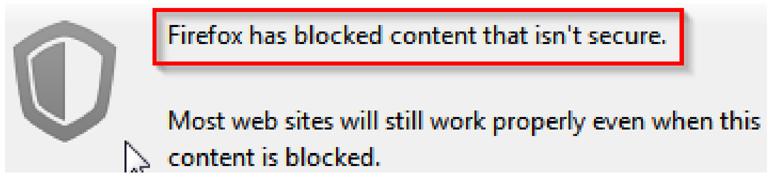


```
--> Testing for CCS injection vulnerability

 CCS   (CVE-2014-0224), experimental          VULNERABLE
```

Web Applications

- Mixed secure and non-secure content
  - Read session cookies, data etc.
  - Edit non-secure resources, e.g. JavaScript
  - Browser errors reduce confidence – or may refuse to load content


Your connection to [redacted] is encrypted with 128-bit encryption. However, this page includes other resources which are not secure. These resources can be viewed by others while in transit, and can be modified by an attacker to change the look of the page.


Firefox has blocked content that isn't secure.

Most web sites will still work properly even when this content is blocked.

- Cacheable HTTPS
  - Non-sensitive content marked as "public" improves performance
  - Check for pages with sensitive data

```
                Pragma: no-cache
                Cache-Control: no-store, no-cache
```

  - http://palizine.plynt.com/issues/2008Jul/cache-control-attributes/

Web Applications

- Sensitive cookies `secure` (but site must expect this…)
- Redirect back to HTTP following HTTPS
- Login over HTTPS but HTTP pre-auth session cookie re-used
- HTTP Strict Transport Security (HSTS) – a safety net
  - Convert all insecure links to secure ones (blocks SSLstrip)
  - Ensures SSL cert warnings cannot be ignored and access blocked
  - Set by a response header `Strict-Transport-Security`
  - Supported by recent browser versions (oh, except IE)
  - If a site is fully HTTPS (and is likely to remain so), why not use it?
  - https://www.leviathansecurity.com/blog/the-double-edged-sword-of-hsts-persistence-and-privacy/ – a fun privacy/tracking issue abusing HSTS

20: Finished

Questions?

www.exploresecurity.com | [@exploresecurity](https://twitter.com/exploresecurity)

nccgroup
freedom from doubt