SSL / TLS

- SSL: secure sockets layer
- TLS: transport layer security
- same protocol design, different crypto
- TLS is evolution of SSL
- standard for Internet security

The primary goal of the TLS protocol is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream.

SSL / TLS Guarantees

- **end-to-end (E2E)** secure communications in the presence of a **network attacker**
  - communication protections from originating client to **intended** server
  - no need to trust intermediaries
  - attacker **completely** owns the network
- against threats
  - eavesdropping
    - encryption
  - manipulation (injection, MITM)
    - integrity, replay protection
  - impersonation
    - server always authenticated
    - client optionally

Imagine you're checking your email

Imagine you're checking your email
at an Internet cafe

Imagine you're checking your email
at an Internet cafe
where the router is hacked

Imagine you're checking your email
at an Internet cafe
where the router is hacked
and it goes through an evil ISP

Imagine you're checking your email
at an Internet cafe
where the router is hacked
and it goes through an evil ISP
and you're in a country that wants to spy on you

Imagine you're checking your email
at an Internet cafe
where the router is hacked
and it goes through an evil ISP
and you're in a country that wants to spy on you
can you be secure?

History

- SSL 1.0 (internal Netscape design, 1994)
- SSL 2.0 (Netscape, 1994, flaws)
- SSL 3.0 (Netscape, 1996)
- TLS 1.0 (standardized, 1999, based on SSL 3.0)
- TLS 1.1 (2006)
- TLS 1.2 (2008)
- TLS 1.3 (2018, fixed many issues)

SSL / TLS in Network Stack

- application
- SSL / TLS
- transport (TCP)
- (inter)network
- link
- physical

TLS in Network Stack
it provides security to **any**
application that uses TCP

TLS in Network Stack
it provides security to **any**
application that uses TCP
API similar to “socket” interface
so it is easy to add security to an app

TLS Basics

- has two protocols: handshake and record
- handshake uses public-key crypto to establish shared secrets (session keys) between client and server
- record uses secret session keys to communicate between client and server while protecting confidentiality, integrity, and authenticity of data

TLS Handshake Protocol

- runs between client and server
  - e.g., web browser and website
- negotiate version of the protocol and cryptographic algorithms
  - client and server talk about what they can do and choose the best option
- authenticate the server and client
  - use **digital certificates** to learn each other's public keys
  - public keys used to sign the messages in the protocol
  - parties can authenticate each other
  - typically only the server is authenticated

TLS Connection

- B(rowser) $\rightarrow$ (S)erver: SYN
- S $\rightarrow$ B: SYN ACK
- B $\rightarrow$ S: ACK
- B $\rightarrow$ S: Hello. My random number is R_B. I support:
  - TLS+RSA+AES128+SHA256, or
  - SSL+DH+3DES+MD5, or ...
- S $\rightarrow$ B: My random number is R_S. Let's use TLS+RSA+AES128+SHA256
- S $\rightarrow$ B: Here's my cert
- around 2–3 KiB of data

Browser sends the list of supported ciphers in the clear.
Is this a problem?

Server makes the selection of the cipher to use.
Is this a problem?

At this point the Browser validates the certificate.

For RSA TLS

- browser constructs **pre-master secret** PS
- $B \rightarrow S$: $\{\mathrm{PS}\}_{S_{PK}}$
- PS encrypted with server's public key
  - PS and both client and server randomness used to derive four keys:
    - two integrity keys (I_B, I_S)
    - two encryption keys (K_B, K_S)
    - separate keys for purpose is good practice
- what does this approach lack?

RSA TLS

- browser and server exchange a MAC on their dialog until now
- $B \rightarrow S$: MAC(dialog, I_B)
- $S \rightarrow B$: MAC(dialog, I_S)
- all subsequent messages are then encrypted with the negotiated cipher
- all messages are sequentially numbered to prevent **replay attacks**
- per-direction encryption keys prevent **reflection attacks**
  - replay of a message back to the source

Alternative: Diffie-Hellman

- after $S \rightarrow B$: Here's my cert
- $S \rightarrow B$: $\{g, p, g^a \bmod p\}_{S_K}$
- $B \rightarrow S$: $g^b \bmod p$
- both servers compute PS $= g^{ab} \bmod p$
- both servers derive C_B, C_S, I_B, I_S
  - using PS and both client and server randomness
- $B \rightarrow S$: MAC(dialog, I_B)
- $S \rightarrow B$: MAC(dialog, I_S)
- all subsequent messages are then encrypted with the negotiated cipher
- all messages are sequentially numbered to prevent **replay attacks**

TLS Protocol again using its nomenclature

TLS Handshake Protocol

- $B \rightarrow S$: ClientHello
- $S \rightarrow B$:
  - ServerHello
  - Certificate
  - ServerKeyExchange
  - CertificateRequest
  - ServerHelloDone

SSL Handshake Protocol (con't)

- $B \rightarrow S$:
  - Certificate
  - ClientKeyExchange
  - CertificateVerify
- client and server switch to the negotiated cipher
- $B \rightarrow S$:
  - Finished
    - MAC of dialog
- $S \rightarrow B$:
  - Finished
    - MAC of dialog

DTLS: Datagram TLS

- a UDP-based variant of TLS
  - not as widely used
  - but still standardized
- provides the security of TLS without needing guarantees of TCP

Ciphersuite Downgrade Attack

Ciphersuite Downgrade Attack

How do the browser and the server decide on which cipher suite to use?

Who ultimately controls that decision?

Ciphersuite Downgrade Attack

How do the browser and the server decide on which cipher suite to use?

Who ultimately controls that decision?

How is it protected against a MITM-attacker?

Ciphersuite Downgrade Attack

How do the browser and the server decide on which cipher suite to use?

Who ultimately controls that decision?

How is it protected against a MITM-attacker?

What is an attack that can happen?

Version Downgrade Attack

Version Downgrade Attack
Suppose both Alice and Bob use TLS 1.3

Version Downgrade Attack

Suppose both Alice and Bob use TLS 1.3

And attacker wants them to use 1.1 (why?)

Version Downgrade Attack

Suppose both Alice and Bob use TLS 1.3

And attacker wants them to use 1.1 (why?)

And why would want to allow a client to use 1.1?

Version Downgrade Attack

Suppose both Alice and Bob use TLS 1.3

And attacker wants them to use 1.1 (why?)

And why would want to allow a client to use 1.1?

What in the protocol protects it?

Version Downgrade Attack

Suppose both Alice and Bob use TLS 1.3

And attacker wants them to use 1.1 (why?)

And why would want to allow a client to use 1.1?

What in the protocol protects it?

MAC is done using a MAC algorithm decided by adversary!

Goal: The cryptographic parameters should be the same on both sides and should be the same as if the peers had been communicating in the absence of an attack.

Solution: make it conspicuous if both parties preferred a higher TLS version

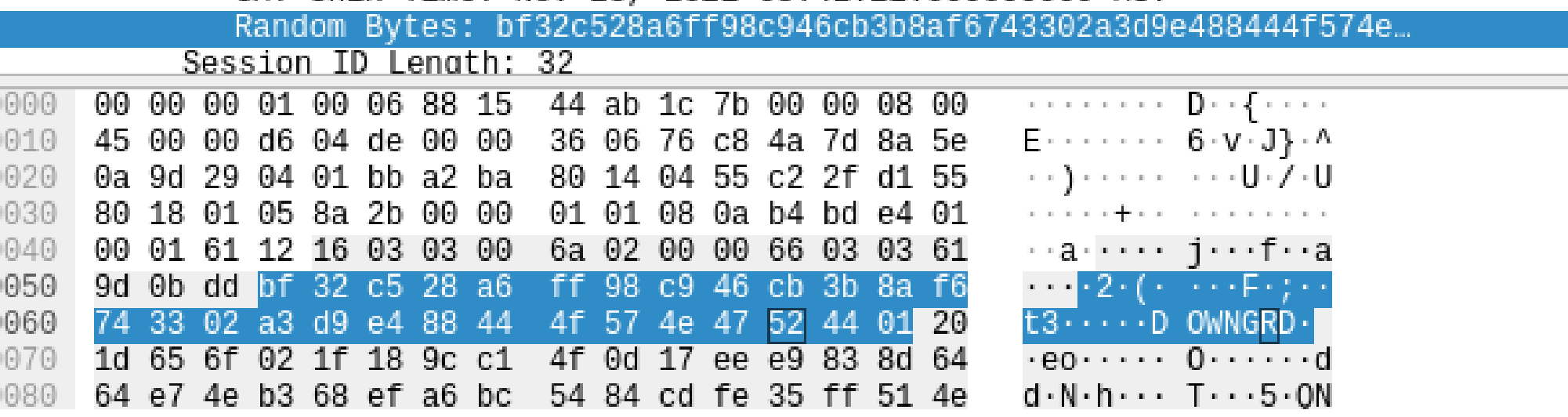
Solution: make it conspicuous if both parties preferred a higher TLS version

If server supports higher version than client asks for server puts DOWNGRDx at end of server randomness

Solution: make it conspicuous if both parties preferred a higher TLS version

If server supports higher version than client asks for server puts DOWNGRDx at end of server randomness

x indicates the version that was asked by the client


GMT Unix Time: Nov 23, 2021 08:42:21.000000000 MST
Random Bytes: bf32c528a6ff98c946cb3b8af6743302a3d9e488444f574e…
Session ID Length: 32
000 00 00 00 01 00 06 88 15 44 ab 1c 7b 00 00 08 00 D {
010 45 00 00 d6 04 de 00 00 36 06 76 c8 4a 7d 8a 5e E 6 v J} ^
020 0a 9d 29 04 01 bb a2 ba 80 14 04 55 c2 2f d1 55) U / U
030 80 18 01 05 8a 2b 00 00 01 01 08 0a b4 bd e4 01 +
040 00 01 61 12 16 03 03 00 6a 02 00 00 66 03 03 61 a j f a
050 9d 0b dd bf 32 c5 28 a6 ff 98 c9 46 cb 3b 8a f6 2 (F ;
060 74 33 02 a3 d9 e4 88 44 4f 57 4e 47 52 44 01 20 t3 D OWNGRD
070 1d 65 6f 02 1f 18 9c c1 4f 0d 17 ee e9 83 8d 64 eo O d
080 64 e7 4e b3 68 ef a6 bc 54 84 cd fe 35 ff 51 4e d N h T 5 QN

Why does this work? If client is using older TLS they won't look at randomness.

Why does this work? If client is using older TLS they won't look at randomness.

Why can't the adversary also change the DOWNGRD?

TLS 1.3 Changes

- (as seen) downgrade is made conspicuous
- remove choices in cipher suites and Diffie-Hellman groups
- remove RSA and force forward secrecy
- 1RTT and 0RTT modes

Structure of this message:

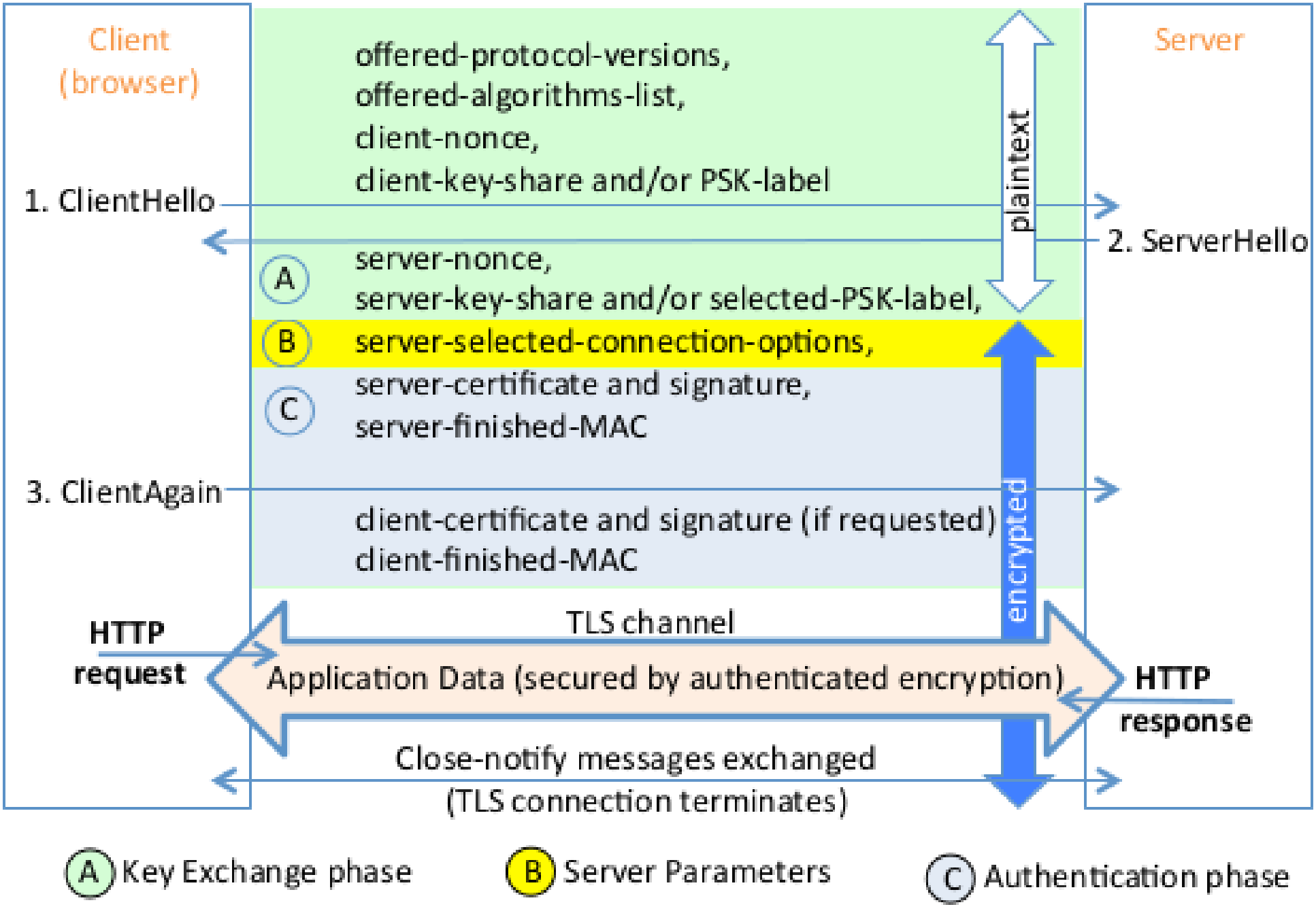
```
   struct {
       ProtocolVersion legacy_version = 0x0303;    /* TLS v1.2 */
       Random random;
       opaque legacy_session_id_echo<0..32>;
       CipherSuite cipher_suite;
       uint8 legacy_compression_method = 0;
       Extension extensions<6..2^16-1>;
   } ServerHello;
```

legacy_version: In previous versions of TLS, this field was used for version negotiation and represented the selected version number for the connection. Unfortunately, some middleboxes fail when presented with new values. In TLS 1.3, the TLS server indicates its version using the "supported_versions" extension (Section 4.2.1), and the legacy_version field MUST be set to 0x0303, which is the version number for TLS 1.2. (See Appendix D for details about backward compatibility.)

```
DHE-DSS-AES128-SHA256   TLSv1.2 Kx=DH       Au=DSS  Enc=AES(128)  Mac=SHA256
ECDHE-ECDSA-CAMELLIA128-SHA256 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=Camellia(128) Mac=SHA256
ECDHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=Camellia(128) Mac=SHA256
DHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA256
DHE-DSS-CAMELLIA128-SHA256 TLSv1.2 Kx=DH       Au=DSS  Enc=Camellia(128) Mac=SHA256
ADH-AES128-SHA256       TLSv1.2 Kx=DH       Au=None Enc=AES(128)  Mac=SHA256
ADH-CAMELLIA128-SHA256  TLSv1.2 Kx=DH       Au=None Enc=Camellia(128) Mac=SHA256
ECDHE-ECDSA-AES256-SHA  TLSv1 Kx=ECDH     Au=ECDSA Enc=AES(256)  Mac=SHA1
ECDHE-RSA-AES256-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-RSA-AES256-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA1
DHE-RSA-CAMELLIA256-SHA SSLv3 Kx=DH       Au=RSA  Enc=Camellia(256) Mac=SHA1
DHE-DSS-CAMELLIA256-SHA SSLv3 Kx=DH       Au=DSS  Enc=Camellia(256) Mac=SHA1
AECDH-AES256-SHA        TLSv1 Kx=ECDH     Au=None Enc=AES(256)  Mac=SHA1
ADH-AES256-SHA          SSLv3 Kx=DH       Au=None Enc=AES(256)  Mac=SHA1
ADH-CAMELLIA256-SHA     SSLv3 Kx=DH       Au=None Enc=Camellia(256) Mac=SHA1
ECDHE-ECDSA-AES128-SHA  TLSv1 Kx=ECDH     Au=ECDSA Enc=AES(128)  Mac=SHA1
ECDHE-RSA-AES128-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-RSA-AES128-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-DSS-AES128-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(128)  Mac=SHA1
DHE-RSA-SEED-SHA        SSLv3 Kx=DH       Au=RSA  Enc=SEED(128) Mac=SHA1
DHE-DSS-SEED-SHA        SSLv3 Kx=DH       Au=DSS  Enc=SEED(128) Mac=SHA1
DHE-RSA-CAMELLIA128-SHA SSLv3 Kx=DH       Au=RSA  Enc=Camellia(128) Mac=SHA1
DHE-DSS-CAMELLIA128-SHA SSLv3 Kx=DH       Au=DSS  Enc=Camellia(128) Mac=SHA1
AECDH-AES128-SHA        TLSv1 Kx=ECDH     Au=None Enc=AES(128)  Mac=SHA1
ADH-AES128-SHA          SSLv3 Kx=DH       Au=None Enc=AES(128)  Mac=SHA1
ADH-SEED-SHA            SSLv3 Kx=DH       Au=None Enc=SEED(128) Mac=SHA1
ADH-CAMELLIA128-SHA     SSLv3 Kx=DH       Au=None Enc=Camellia(128) Mac=SHA1
```

```
ECDHE-ECDSA-NULL-SHA    TLSv1 Kx=ECDH     Au=ECDSA Enc=None       Mac=SHA1
ECDHE-RSA-NULL-SHA      TLSv1 Kx=ECDH     Au=RSA  Enc=None       Mac=SHA1
AECDH-NULL-SHA          TLSv1 Kx=ECDH     Au=None Enc=None       Mac=SHA1
NULL-SHA256             TLSv1.2 Kx=RSA      Au=RSA  Enc=None       Mac=SHA256
ECDHE-PSK-NULL-SHA384   TLSv1 Kx=ECDHEPSK Au=PSK  Enc=None       Mac=SHA384
ECDHE-PSK-NULL-SHA256   TLSv1 Kx=ECDHEPSK Au=PSK  Enc=None       Mac=SHA256
ECDHE-PSK-NULL-SHA      TLSv1 Kx=ECDHEPSK Au=PSK  Enc=None       Mac=SHA1
RSA-PSK-NULL-SHA384     TLSv1 Kx=RSAPSK   Au=RSA  Enc=None       Mac=SHA384
RSA-PSK-NULL-SHA256     TLSv1 Kx=RSAPSK   Au=RSA  Enc=None       Mac=SHA256
DHE-PSK-NULL-SHA384     TLSv1 Kx=DHEPSK   Au=PSK  Enc=None       Mac=SHA384
DHE-PSK-NULL-SHA256     TLSv1 Kx=DHEPSK   Au=PSK  Enc=None       Mac=SHA256
RSA-PSK-NULL-SHA        SSLv3 Kx=RSAPSK   Au=RSA  Enc=None       Mac=SHA1
DHE-PSK-NULL-SHA        SSLv3 Kx=DHEPSK   Au=PSK  Enc=None       Mac=SHA1
NULL-SHA                SSLv3 Kx=RSA      Au=RSA  Enc=None       Mac=SHA1
NULL-MD5                SSLv3 Kx=RSA      Au=RSA  Enc=None       Mac=MD5
PSK-NULL-SHA384         TLSv1 Kx=PSK      Au=PSK  Enc=None       Mac=SHA384
PSK-NULL-SHA256         TLSv1 Kx=PSK      Au=PSK  Enc=None       Mac=SHA256
PSK-NULL-SHA            SSLv3 Kx=PSK      Au=PSK  Enc=None       Mac=SHA1
```

```
TLS_AES_256_GCM_SHA384   TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256   TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
```


Client
(browser)
Server
offered-protocol-versions,
offered-algorithms-list,
client-nonce,
client-key-share and/or PSK-label
plaintext
1. ClientHello
2. ServerHello
A
server-nonce,
server-key-share and/or selected-PSK-label,
B
server-selected-connection-options,
C
server-certificate and signature,
server-finished-MAC
3. ClientAgain
client-certificate and signature (if requested)
client-finished-MAC
encrypted
TLS channel
HTTP
request
Application Data (secured by authenticated encryption)
HTTP
response
Close-notify messages exchanged
(TLS connection terminates)
A Key Exchange phase
B Server Parameters
C Authentication phase

TLS Stripping Attacks

TLS Stripping Attacks
AKA HTTP Downgrade Attacks

How do we access HTTPS sites?

How do we access HTTPS sites?

302 redirects

clicking links

manually typing https://

HTTPS everywhere browser extension

Bank of America | Home | Personal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.bankofamerica.com/index.jsp

Google

Most Visited

Locations • Contact Us • Help • Sign In • En Español

Bank of America

Search

PERSONAL | SMALL BUSINESS | CORPORATE & INSTITUTIONAL | ABOUT BANK OF AMERICA

Online Banking

Easy. Secure. Free.

Enroll View demo | Learn more

Enter Online ID:

Save this Online ID

Password:

Where do I enter my Passcode?

Sign In

Forgot or need help with your ID?

Reset Passcode

Your Privacy & Security

Report suspicious email

Norton 360 - Free Trial

Products & Services

Checking
Savings & CDs
Credit cards
Mortgage
Refinance
Home equity
Auto loans
IRAs
Investment Services

Manage Your Accounts

Fees and processes
Order Check Card
Online Investing

Online Banking >
Viewing your accounts
Accessing credit cards
Bill Pay
Tracking your expenses

Achieve Your Goals

Keep the Change®
Buying a home
Searching for a home
Retirement Center
Planning for college
Student loans
Purchasing a car
Consolidating debt
Small Business Online

Find: Previous Next Highlight all Match case

Done

Stripper Attacks

Stripper Attacks
instead of attacking the TLS connection
with, e.g., a bad cert

Stripper Attacks
instead of attacking the TLS connection
with, e.g., a bad cert
we attack the HTTP connection instead

Stripper Attacks
instead of attacking the TLS connection
with, e.g., a bad cert
we attack the HTTP connection instead
e.g., we stop it from using HTTPS

Stripper Attacks

instead of attacking the TLS connection

with, e.g., a bad cert

we attack the HTTP connection instead

e.g., we stop it from using HTTPS

and hope user doesn't notice

HTTPS is reached by 302s or links, both from HTTP.

Stripper Attacks

- $B \rightarrow E \rightarrow S$: HTTP GET
- $B \leftarrow E \leftarrow S$: webpage
  - E replaces
    - from <a href="https://...">
    - to <a href="http://...">
    - from 302 Found\r\nLocation: https://
    - to 302 Found\r\nLocation: http://
  - E remembers all these changes

Stripper Attacks

if B requests an HTTP URL that E stripped
then E establishes an HTTPS request to S
to get content and follow other links.

Stripper Attacks

if B requests an HTTP URL that E stripped

then E establishes an HTTPS request to S

to get content and follow other links.

E forwards the HTTPS traffic to B over HTTP

Stripper Attacks

if B requests an HTTP URL that E stripped

then E establishes an HTTPS request to S

to get content and follow other links.

E forwards the HTTPS traffic to B over HTTP

E continues to strip any HTTP links.

The Result

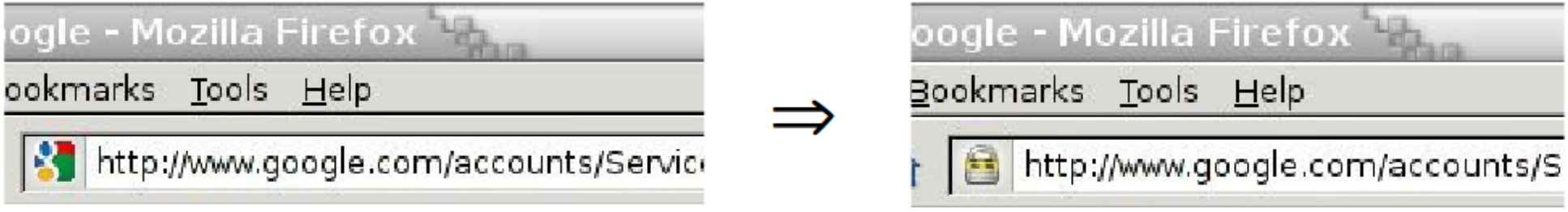
- server cannot tell the difference
  - server does not authenticate client
  - everything looks like a TLS client requested it
- client didn't display any warnings
  - no bad cert, etc.
  - just a normal HTTP website

How can the client realize this is happening?

How can the client realize this is happening?
They notice the lock icon is missing.

How can the client realize this is happening?
They notice the lock icon is missing.
Neutralization of HTTPS helps detect this now.

Trick to get the lock icon:
E does everything the same
except she changes the favicon


ogle - Mozilla Firefox
ookmarks Tools Help
http://www.google.com/accounts/Servic
oogle - Mozilla Firefox
Bookmarks Tools Help
http://www.google.com/accounts/S

Gmail: Email from Google - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.google.com/accounts/ServiceLogin?service=

Google

Most Visited Getting Started Latest Headlines

Gmail by Google BETA

Welcome to Gmail

A Google approach to email.

Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient, and useful. And maybe even fun. After all, Gmail has:

Less spam
Keep unwanted messages out of your inbox with Google's innovative technology.

Mobile access
Read Gmail on your mobile phone by pointing your phone's web browser to **http://gmail.com/app**. Learn more

Lots of space
Over 7290.462157 megabytes (and counting) of free storage so you'll never need to delete another message.

Sign in to Gmail with your Google Account

Username:

Password:

Remember me on this computer.

Sign in

I cannot access my account

Sign up for Gmail

About Gmail New features!

Done

Online Payment, Merchant Account - PayPal - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.paypal.com/

Google

Most Visited Getting Started Latest Headlines

Sign Up | Log In | Help | Security Center

Search

PayPal

U.S. English

Home Personal Business Products & Services Developers

Get Started Send Money Request Money Sell on eBay Developers

Account login

Email address

PayPal password

Log In

Forgot your email address or password?

New to PayPal? Sign up.

Top questions

- Why use PayPal when I have credit cards?
- What can I do with PayPal?
- Is PayPal free to use?

Pay With: VISA MasterCard DISCOVER BANK

Pay online

- Learn how PayPal works.
- Shop without exposing your financial information.
- Send money to friends and family around

Get paid online

- Accept payments for your eBay listings.
- Start accepting credit cards on your website.
- See all the ways to get paid online.

Done

How can we fix this?

Firefox Add-ons

Explore Extensions Themes More...

This add-on is not compatible with your version of Firefox.

Restart Required

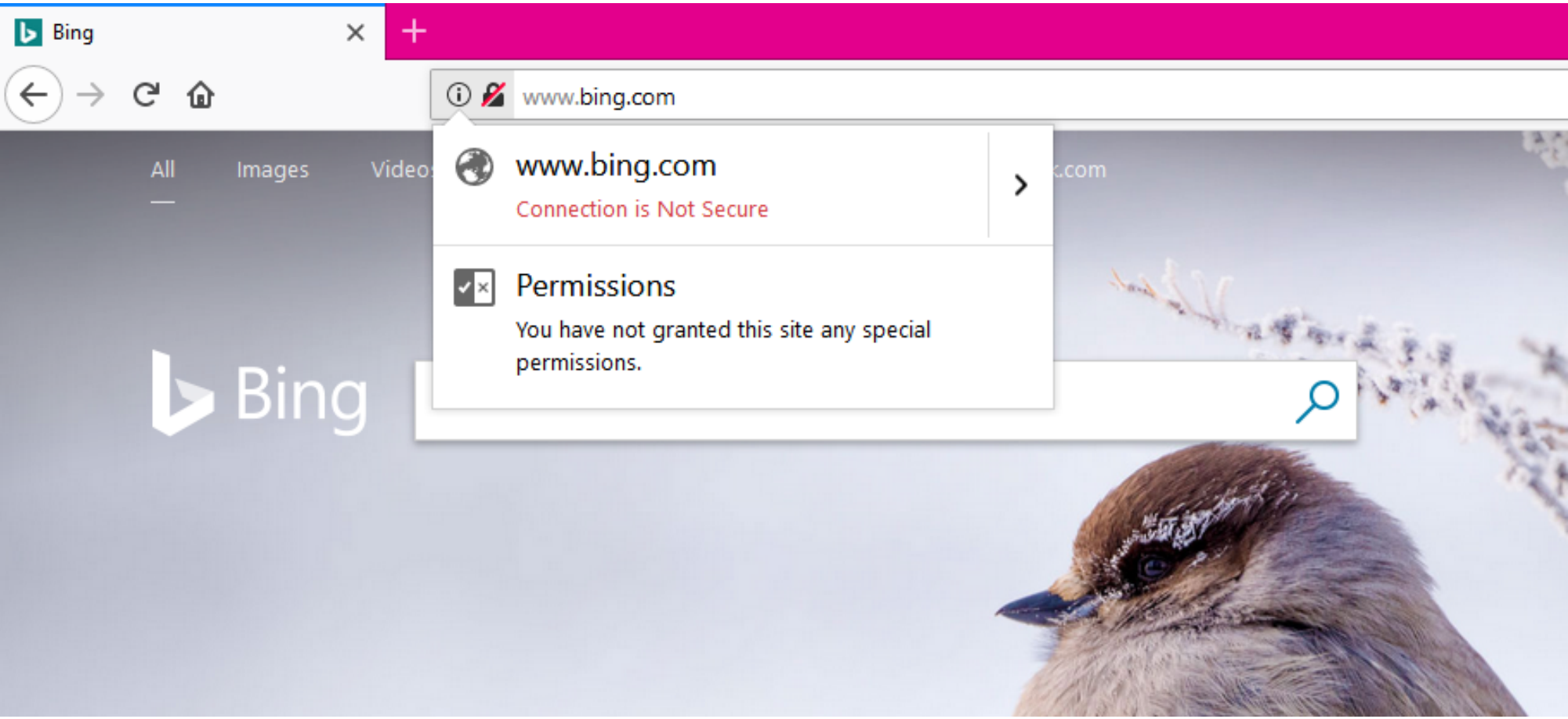
Not compatible with Firefox Quantum

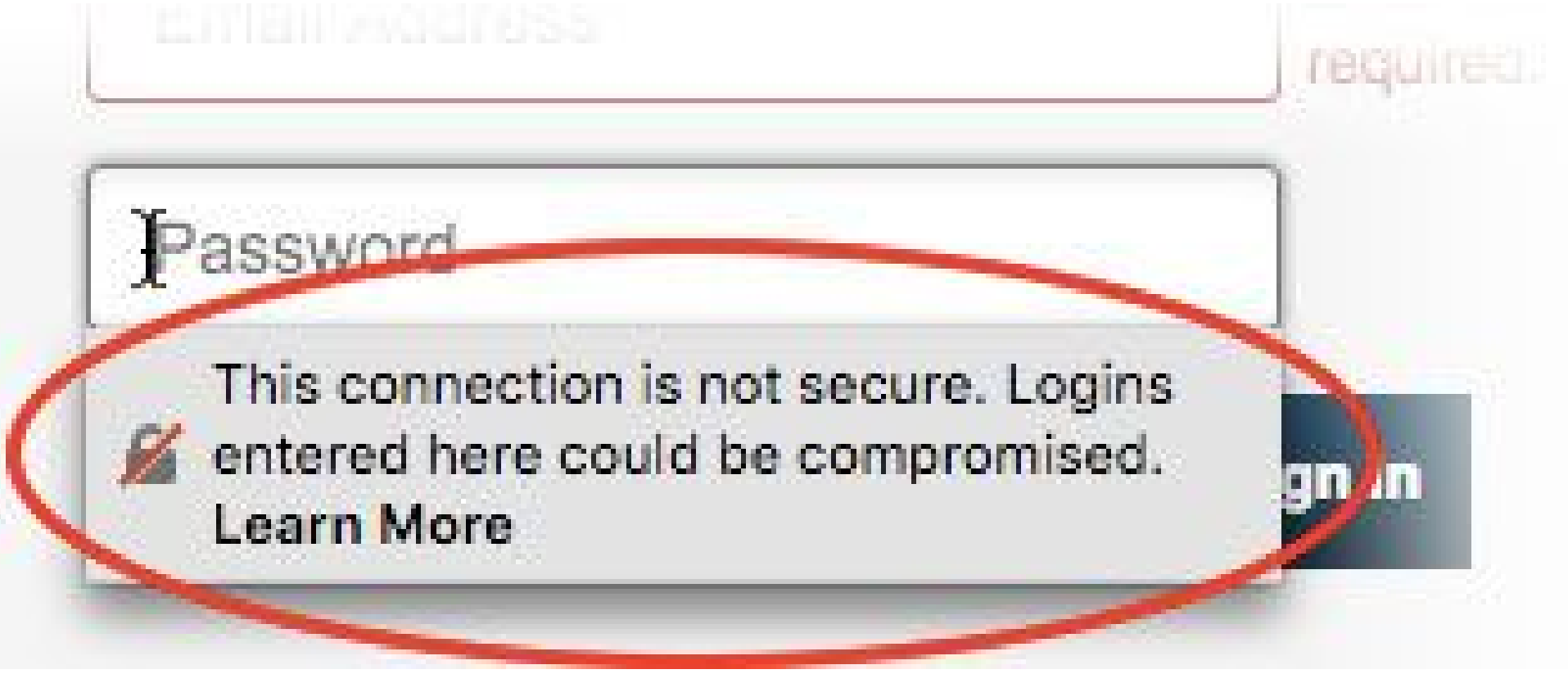
HTTP Nowhere

by Chris Wilper

Block unencrypted web traffic for added security.

+ Add to Firefox


Bing
www.bing.com
All
Images
Video
www.bing.com
Connection is Not Secure
Permissions
You have not granted this site any special permissions.
Bing


Email address
required
Password
This connection is not secure. Logins entered here could be compromised.
Learn More

What about non-browser HTTP

What about non-browser HTTP
e.g., apps that download resources over HTTP won't necessarily be rendered in a browser

Example Non-Browser HTTP

- mobile app traffic
- wget, curl, apt, pip, yum, npm, git clone
- stunnel
- HTTP-based RPCs or APIs

PROPOSED STANDARD
Errata Exist
Internet Engineering Task Force (IETF)
Request for Comments: 6797
Category: Standards Track
ISSN: 2070-1721

J. Hodges
PayPal
C. Jackson
Carnegie Mellon University
A. Barth
Google, Inc.

November 2012

HTTP Strict Transport Security (HSTS)

Abstract

This specification defines a mechanism enabling web sites to declare themselves accessible only via secure connections and/or for users to be able to direct their user agent(s) to interact with given sites only over secure connections. This overall policy is referred to as HTTP Strict Transport Security (HSTS). The policy is declared by web sites via the Strict-Transport-Security HTTP response header field and/or by other means, such as user agent configuration, for example.

Status of This Memo

HTTP Strict Transport Security

- puts a Strict-Transport-Security header in the reply
- includes a “max-age” to say how long it should be there
- client refuses HTTP for any page that has this header stored
- what trust model applies here?
- other issues?

```
       University of Calgary, Department of Computer Science
       This system is for use by authorized CPSC users only.
**************************************************************************
            TECH       https://cpsc.ucalgary.ca/tech/
            SUPPORT    it@ucalgary.ca
            CONTACT    403-210-9300
**************************************************************************
  This system will be rebooted as required to apply security updates
**************************************************************************

MISSING PANEL MENU? NO TASK BAR? Open terminal ctrl+alt+t, then run:
         gsettings reset-recursively org.cinnamon

Last login: Mon Sep 22 14:53:22 2025 from 198.48.151.21
joel.reardon@rsx2:~$ ll .wget-hsts
-rw-r--r-- 1 joel.reardon profs 267 Jan 31  2022 .wget-hsts
joel.reardon@rsx2:~$
```