Software and Web Security 2

MitM attacks on HTTPS sessions

Much of this material can also be seen in
DEFCON 2009 presentation by Moxie Marlinspike;
see URL on course page

MitM (Man-in-the-Middle) attacks

- MitM attack: attacker gets between the browser and the web server, eg by
  - setting up a wifi access point
  - luring victim to his website and passing on traffic to another site
- HTTPS (ie HTTP over TLS/SSL) should protect against this...

- There are different ways in which this can go wrong
  - on the protocol/network level
  - security flaws in the browser or at the Certificate Authority (CA),
    - in the software or the human/organisation

- One particular attack is SSL stripping, invented by Moxie Marlinspike, and implemented in **`sslstrip`** tool

(Aside: name confusion on SSL vs TLS)

- Is it SSL, TLS, SSL/TLS, or TLS/SSL?

- SSL (Secure Socket Layer) developed at Netscape.
  Version 1.0 never released.
  Version 2.0 had several security flaws

- SSL 3.0 renamed to TLS, of which versions 1.0, 1.1, and 1.2 exist

Reminder: weaknesses/attacks on HTTPS already discussed in week 3

- Implementation flaws in various TLS implementations
  - such as HeartBleed, iOS goto bug, or FREAK

- Software failing to do the right certificate checks

- Certificate Authorities being hacked

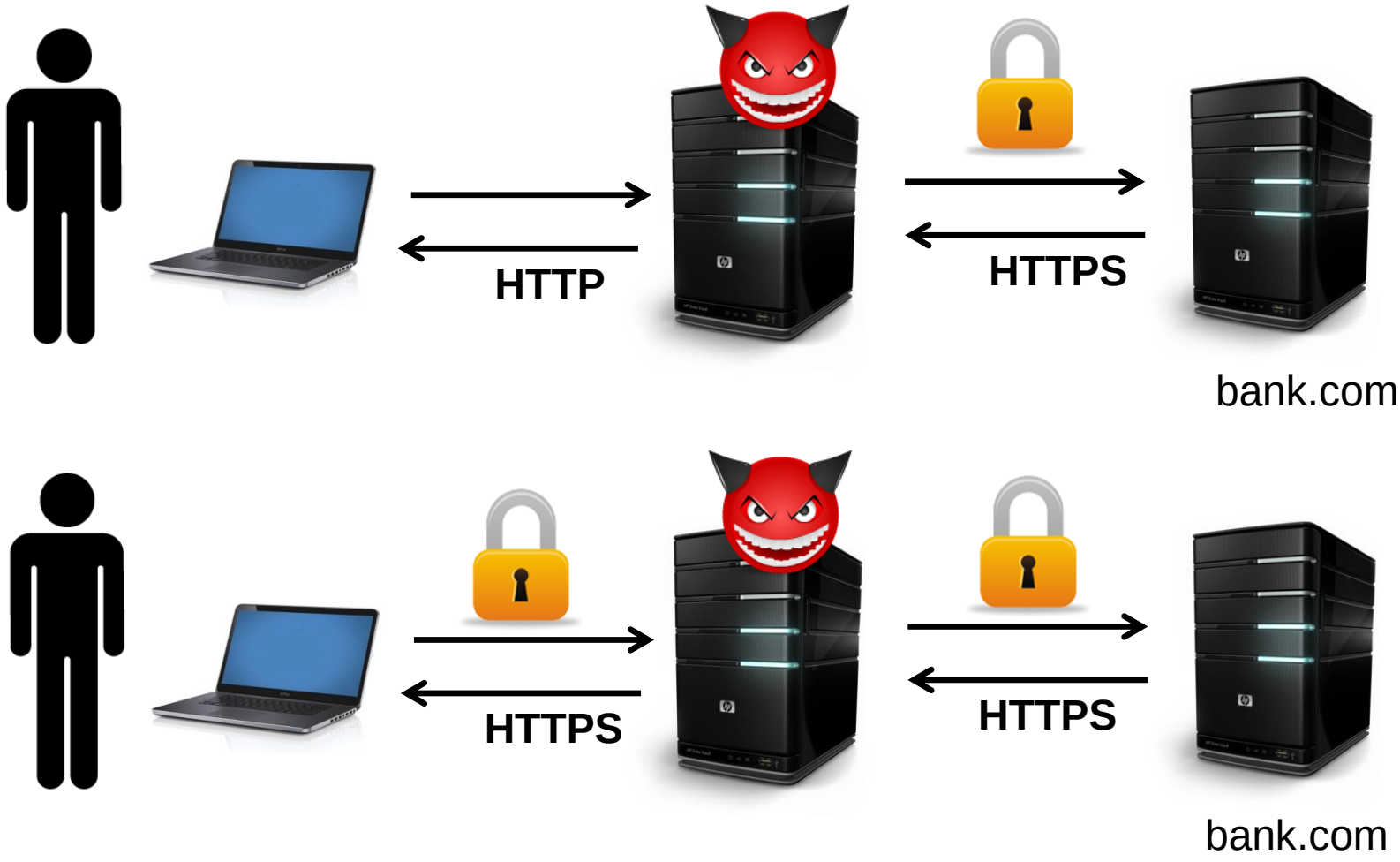
DigiNotar
Internet Trust Services

- Software in webbrower displaying URLs incorrectly
  - eg `http://paypal.com%01%00@mafia.com`

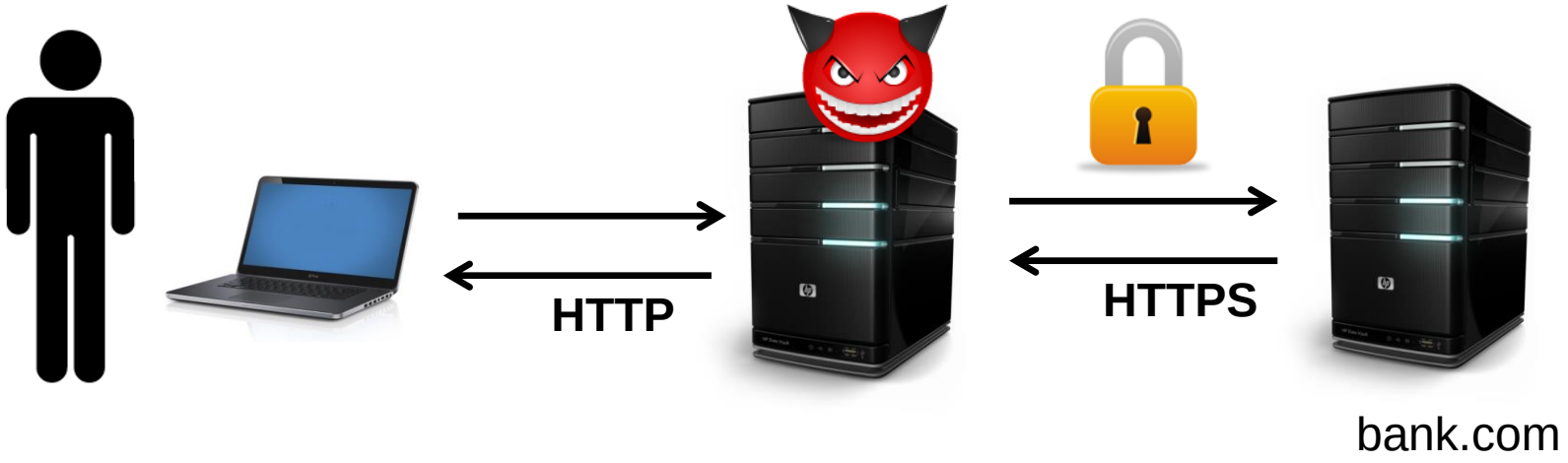
- Human user of the web browser not checking things correctly
  - but punycode representation of URLs is a countermeasure against user being confusing by strange character sets

SSL stripping

Two variants of SSL stripping

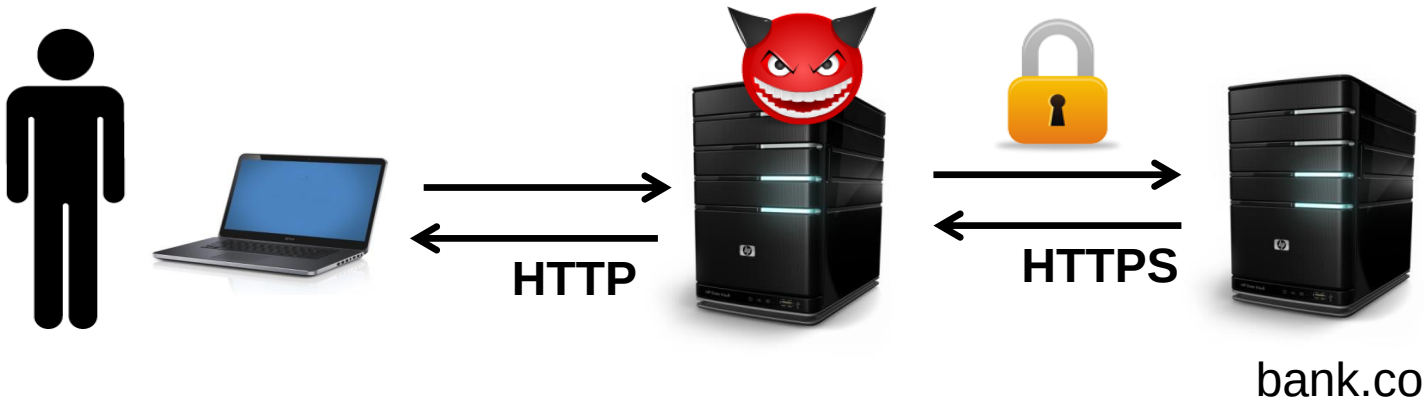


SSL stripping (1)



SSL stripping (1) HTTP + HTTPS

The idea: the attacker forces the browser to fall back to an HTTP session, and hope the user won't notice the missing s

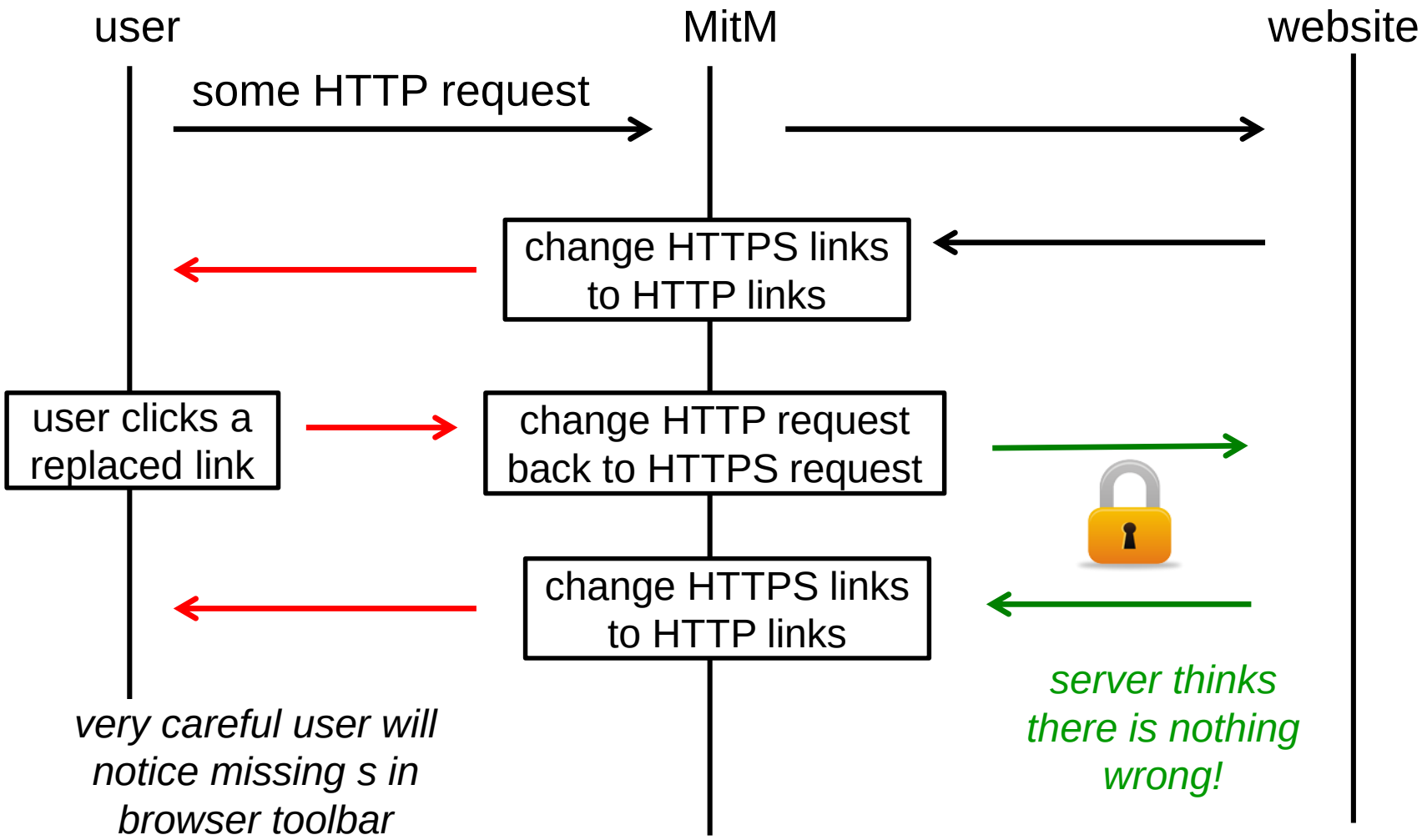


When can the attacker do this? If the user

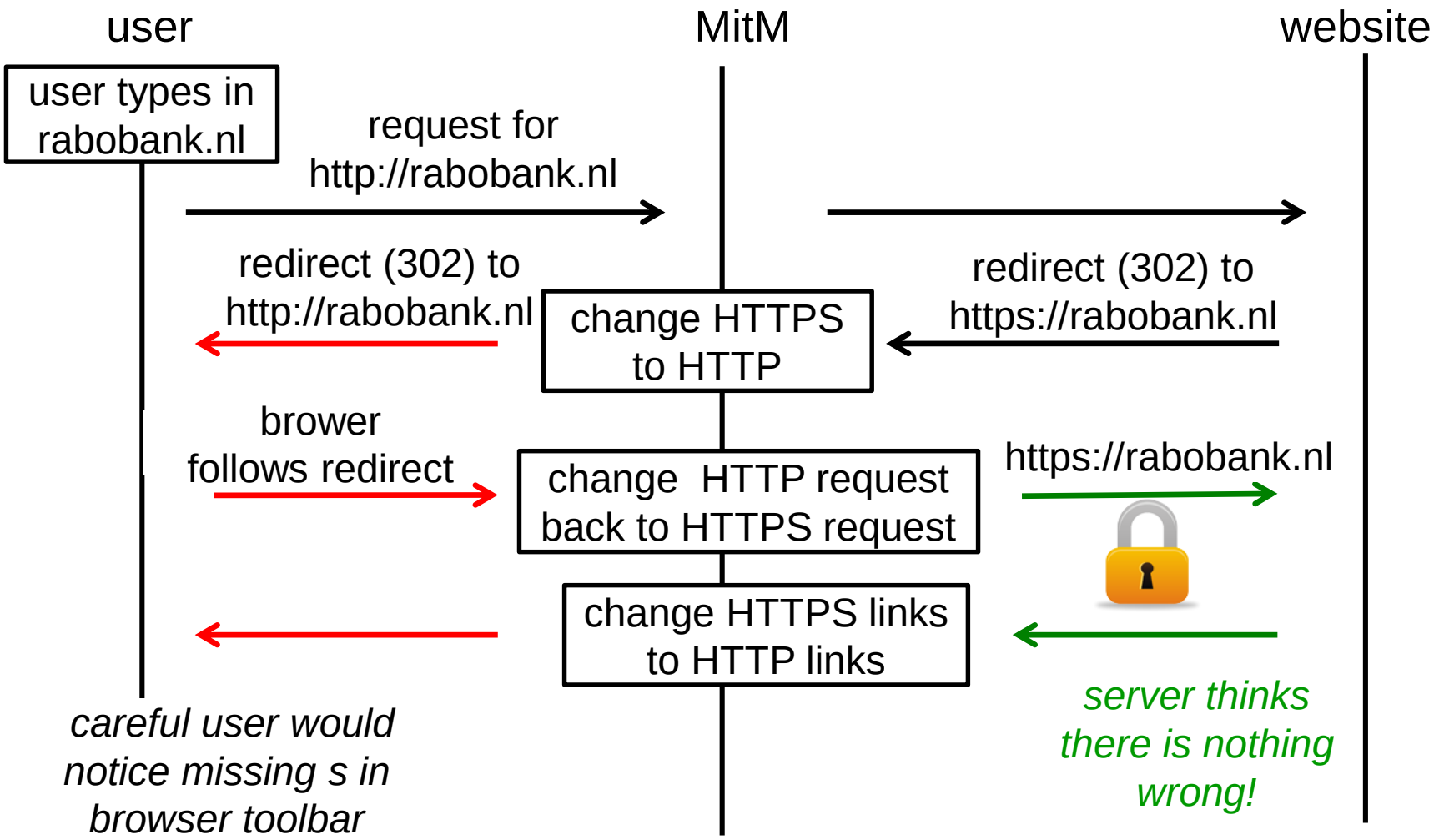
a) types in rabobank.nl, without https in front of it

b) begins a HTTPS session by clicking on a link in a webpage that was retrieved with HTTP

MitM attack on start of HTTPS session (b)

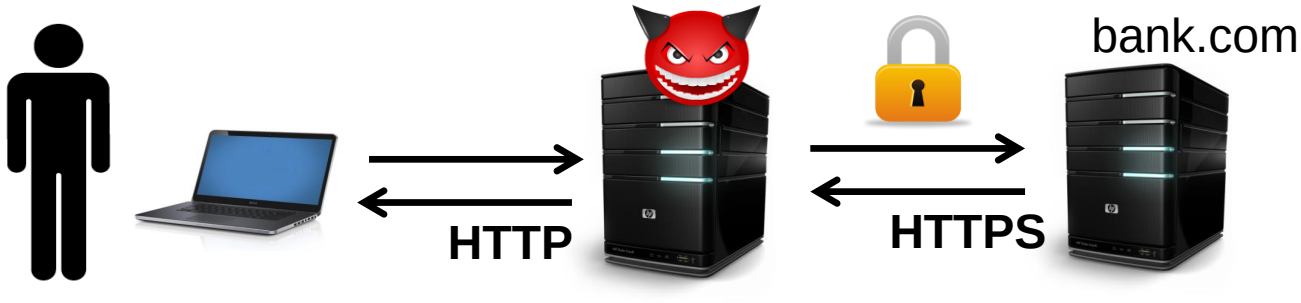


MitM attack on starting of HTTPS session (a)



SSL stripping (1)

- The MitM attacker
  - strips S from HTTPS in links in traffic from server to user
  - puts this S back in traffic from the user to the server
- The result



- The attacker can now intercept a username and password that the user sends (typically in a POST request)
- After intercepting this information, the attacker could stop the MitM attack, so that a secure tunnel between user & server is established
  - and the user can then no longer see anything wrong!

Some problems & fixes

- Secure cookies won't be sent by the client's browser over HTTP

  Solution: remove the secure bit from Set Cookie instructions when forwarding traffic from the server to user

Similarly, the attacker can

- strip content encodings (eg gzip) to simplify having to parse these
- strip **`if-modified`** headers to prevent the web browser from reusing cached pages

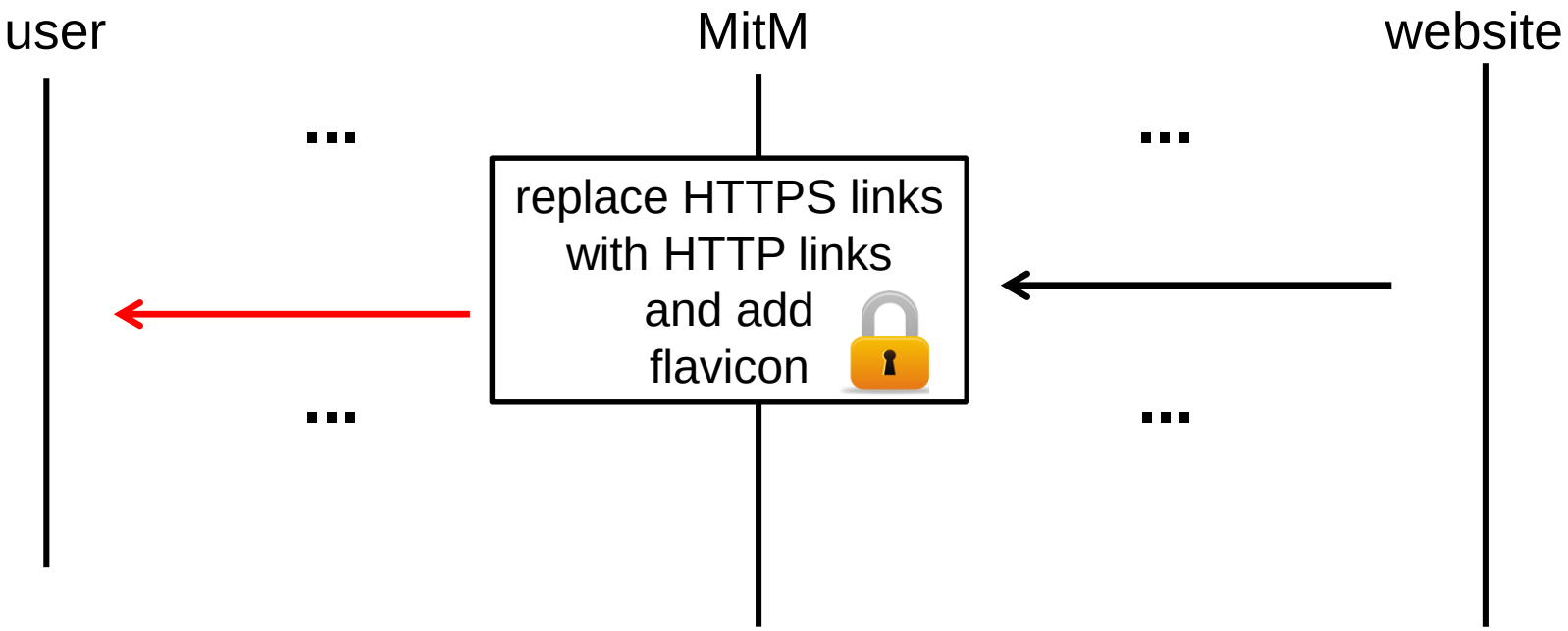
Spotting this attack?

A careful user can spot this attack

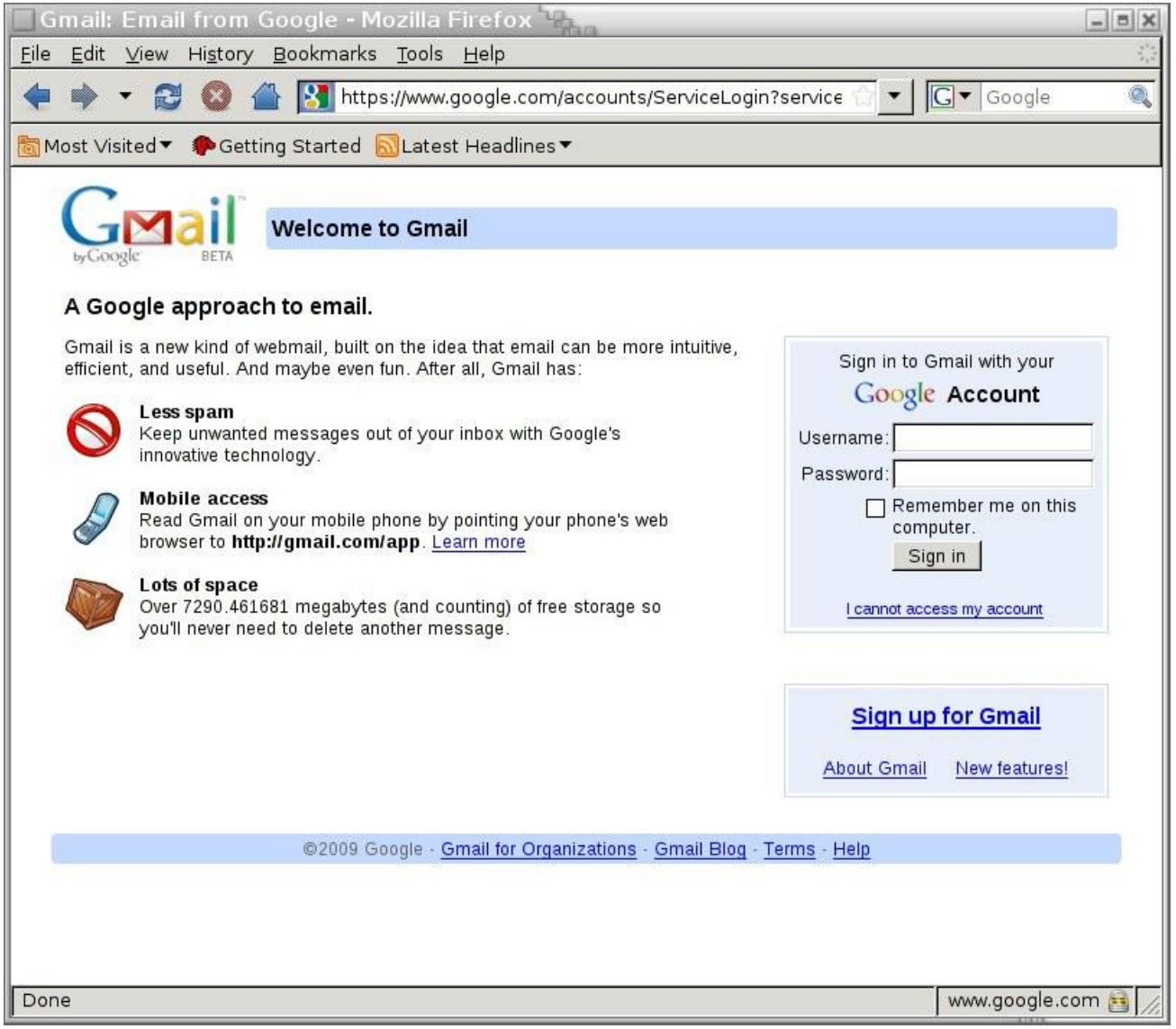
- the URL misses the s in https
- the little lock is missing in the browser corner 

Funny improvement: the attacker can add  as flavicon

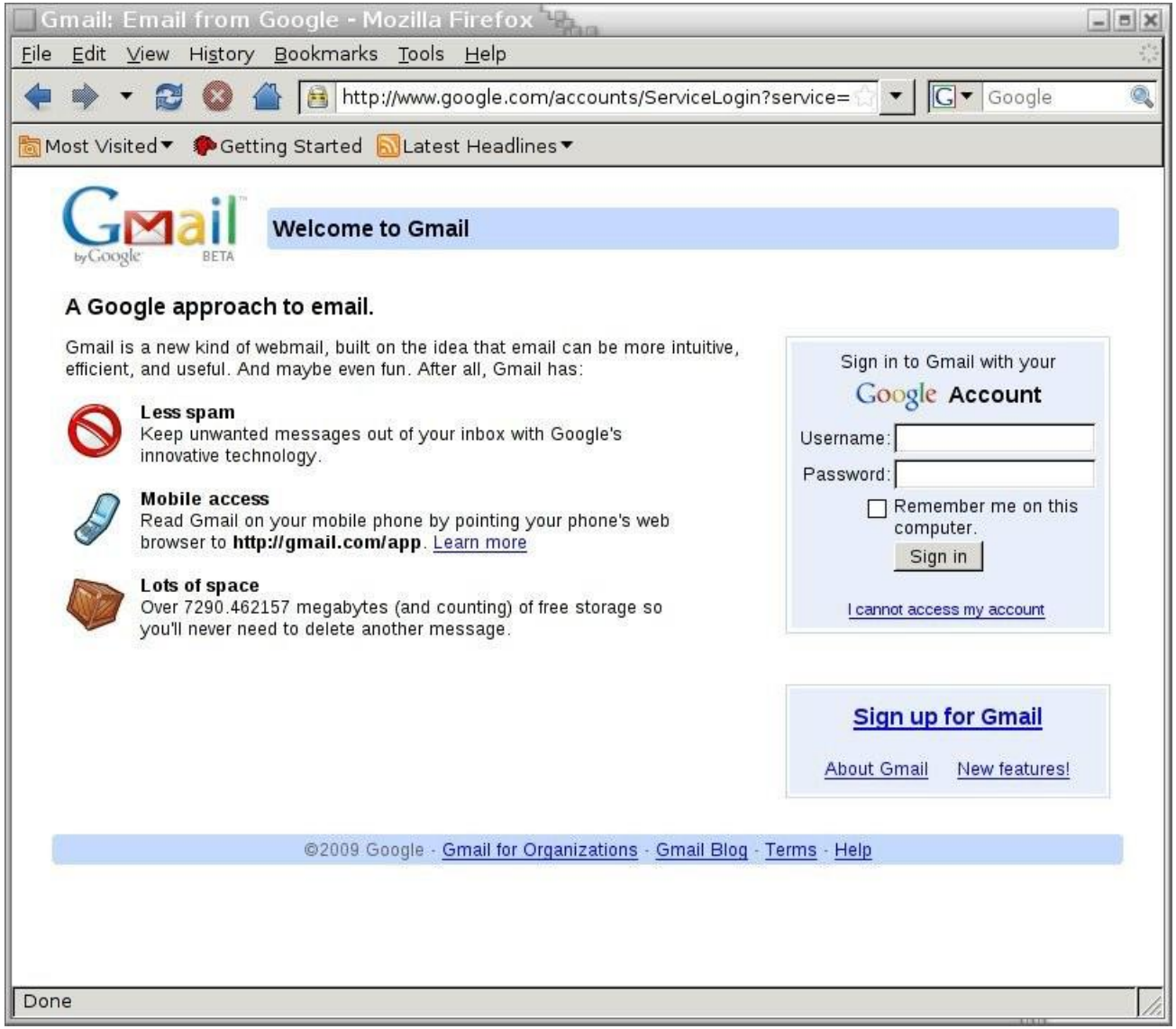
Improvement: adding flavicon



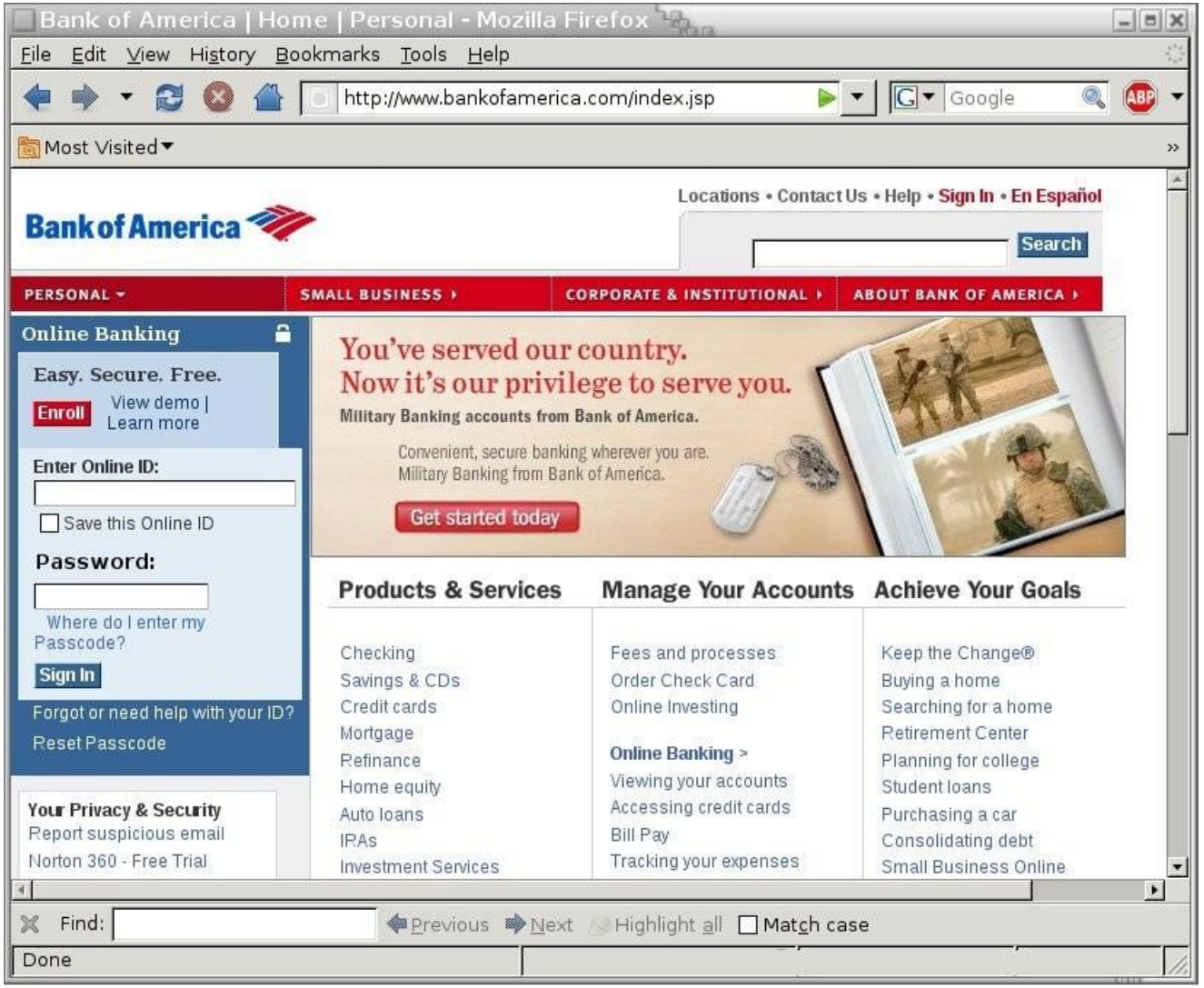
The original secure site



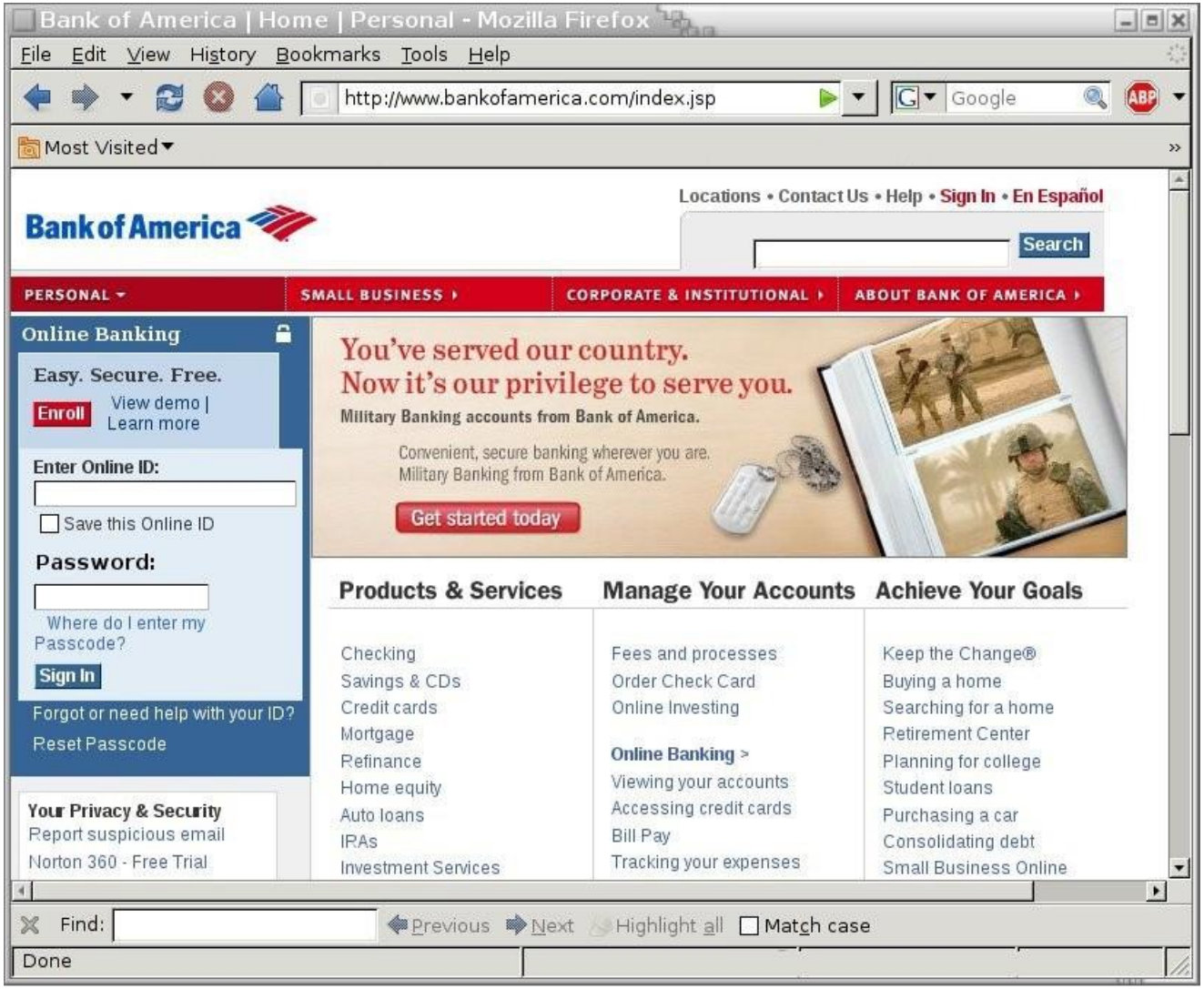
SSL stripped version

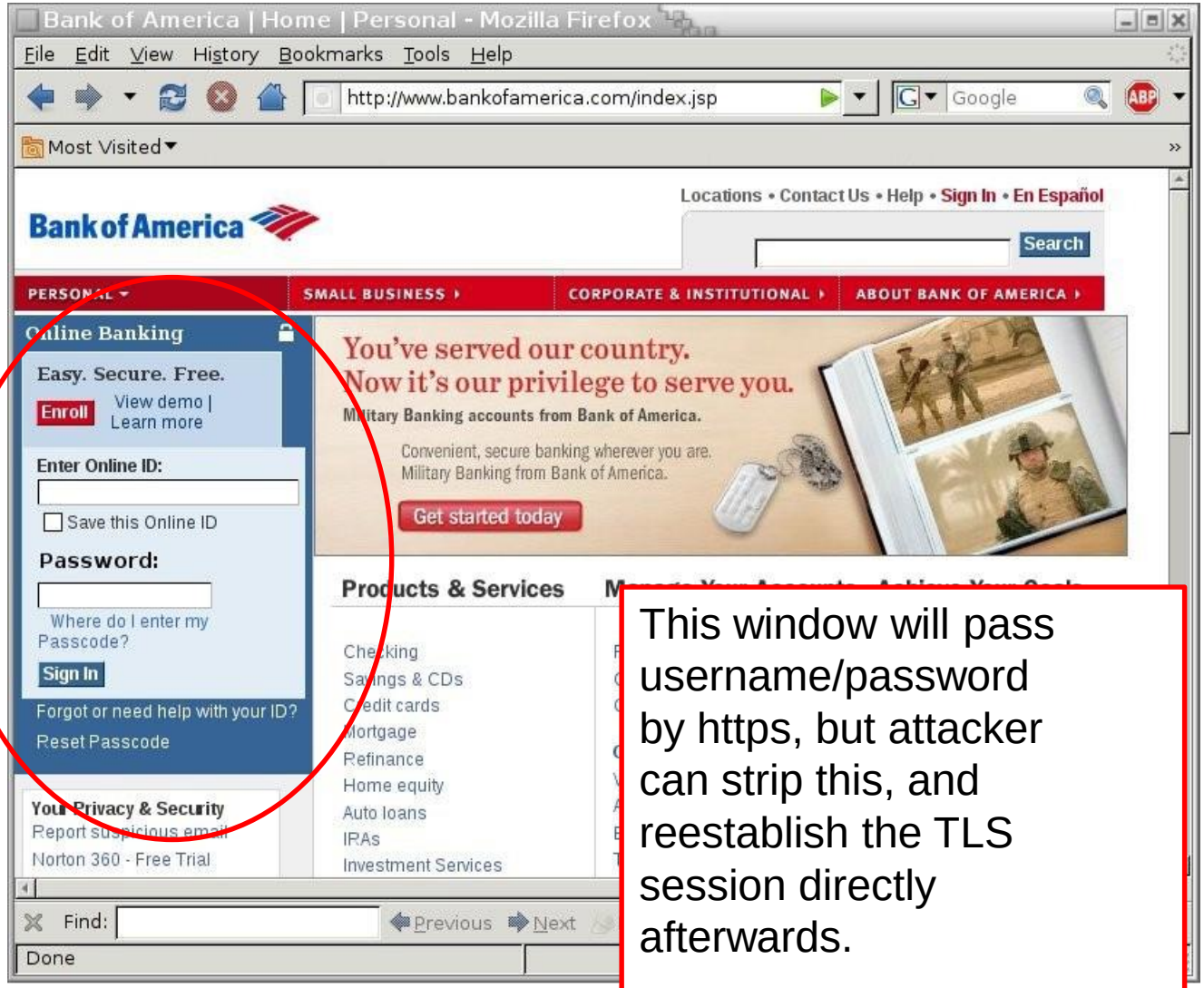


The original secure site



The SSL stripped version





This window will pass username/password by https, but attacker can strip this, and reestablish the TLS session directly afterwards.

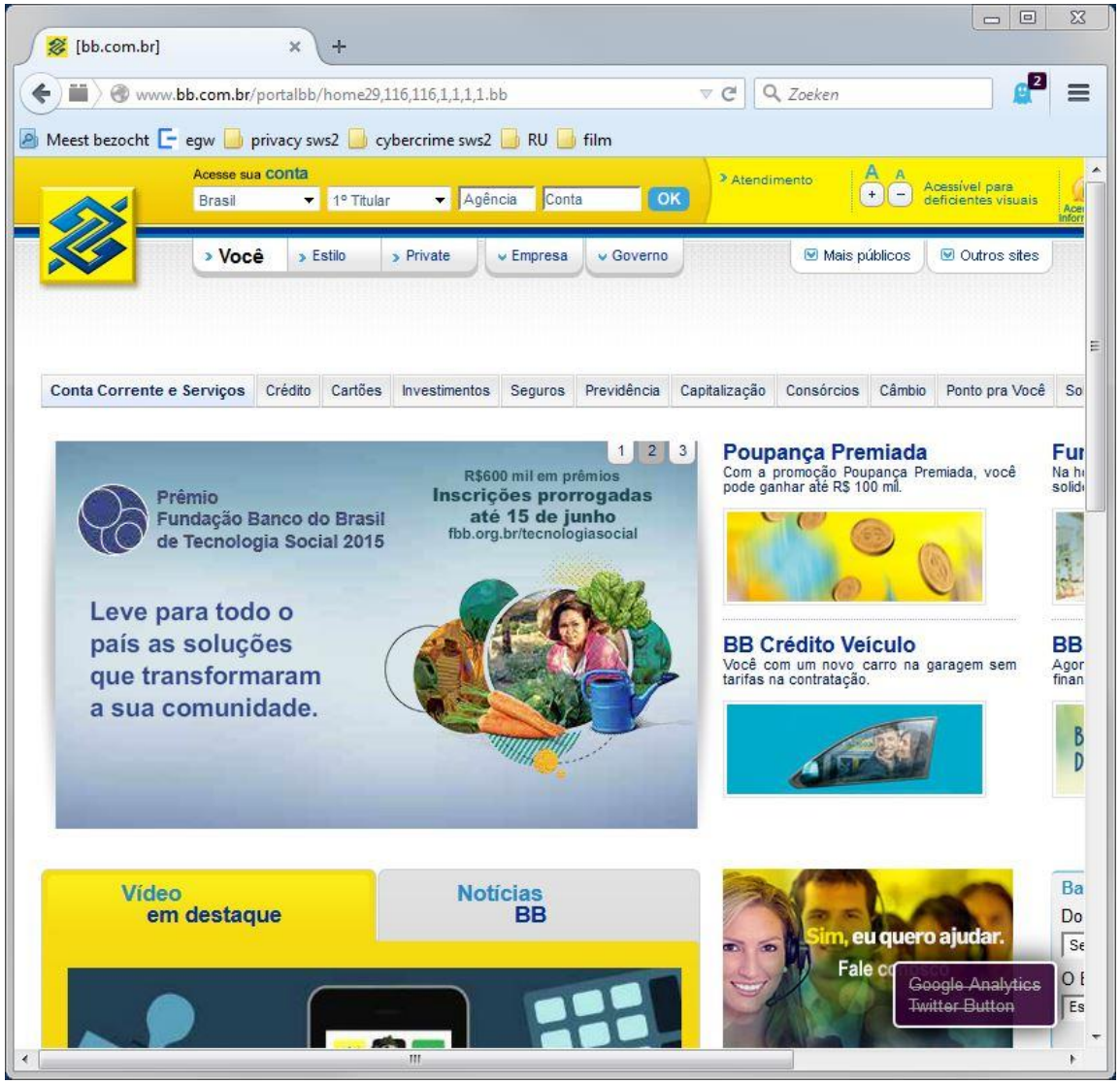
Can the user still spot this?

Moral

Moral of the last examples:

- Never use https for a frame inside a http page
- Never issues https requests from an http page

This still happens...



(old) news

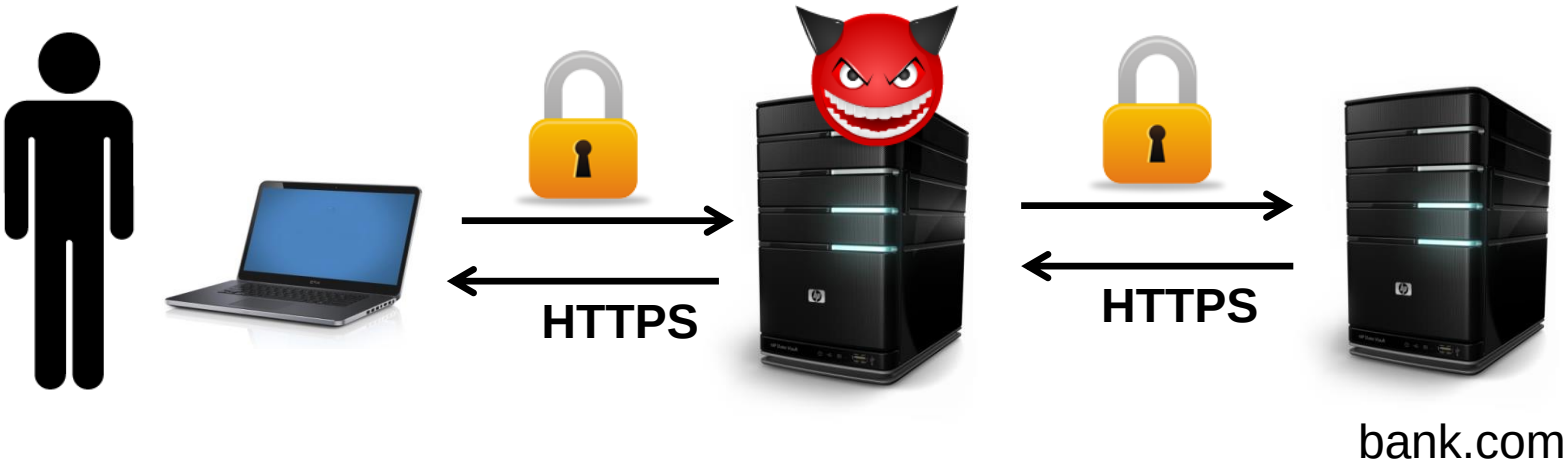
- http://kassa.vara.nl/tv/afspeelpagina/fragment/schokkend-nieuws-gevaarlijk-lek-in-internetbankieren-ontdekt/speel/1/
- http://webwereld.nl/beveiliging/82658-geld-stelen-via-hotspots-kon-door-lek-in-internetbankieren

Schokkend nieuws: gevaarlijk lek in internetbankieren ontdekt

Trefwoorden:
Internetbankieren, Gehackt, Banken, Onveilig, Nieuws
Datum: za 24 mei 2014, 19:07
Categorie: Computers & Internet
Reacties: 0

SSL stripping (improved)

Can we improve things? Ideally we want to get



so the user cannot notice he is not having a TLS session?

For this, we have to to trick the browser into setting up a TLS tunnel to the attacker, believing it to be bank.com

SSL stripping (2): HTTPS+HTTPS

Different ways for attacker to set up TLS tunnel to himself from victim

1. Use a self-signed certificate for bank.com
   - but warnings will scare most users away ☹
2. Attacker can buy domain name that looks like bank.com with international characters
   - browser using puny-code may reveal this to user ☹
3. Attacker can redirect to mafia.com, for which he has a certificate
   a) and hope the user does not notice the mafia.com in address bar
   b) better, use characters that look like / and ? to make URL that looks like the bank's, eg **`https://bank.com/Somelongname?.mafia.com`**
      - browser that highlights domain part of URL may warn user ☹

SSL stripping (2): HTTPS+HTTPS

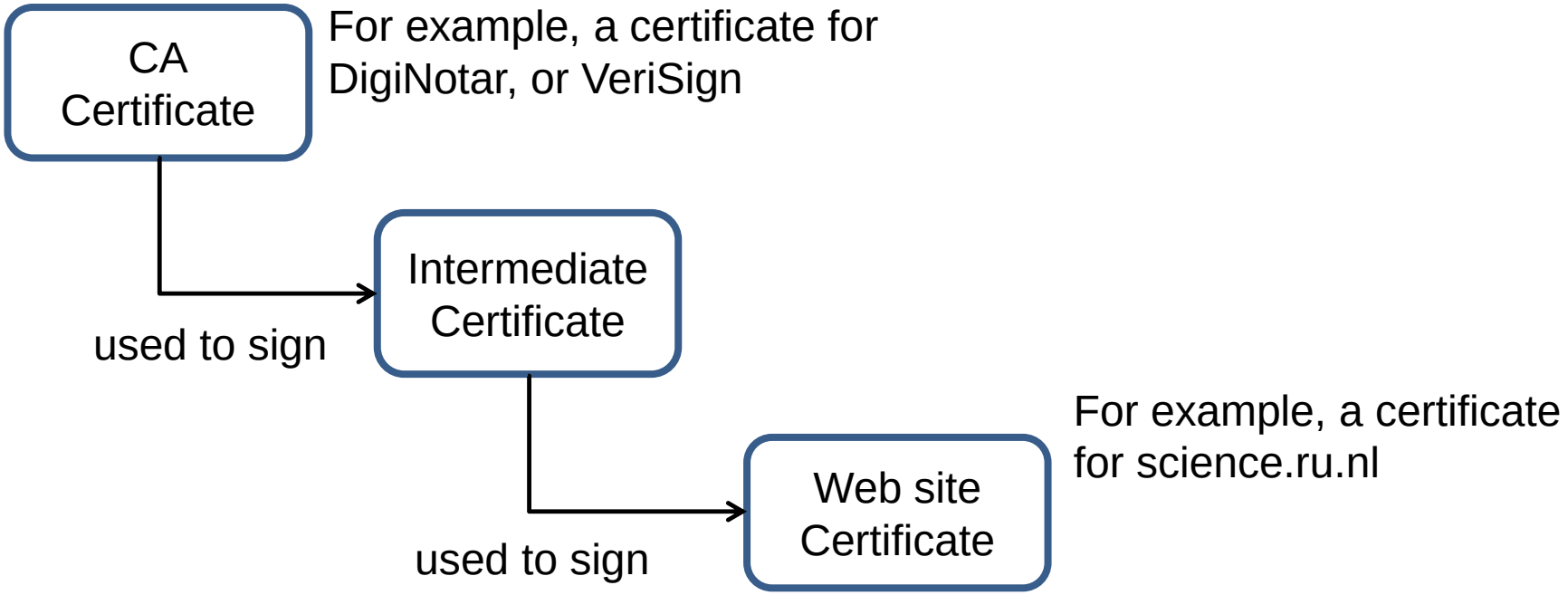
Different ways for attacker to set up TLS tunnel to himself from victim

4. older TLS implementations in browsers had a bug that allowed attackers to create certificate for any site

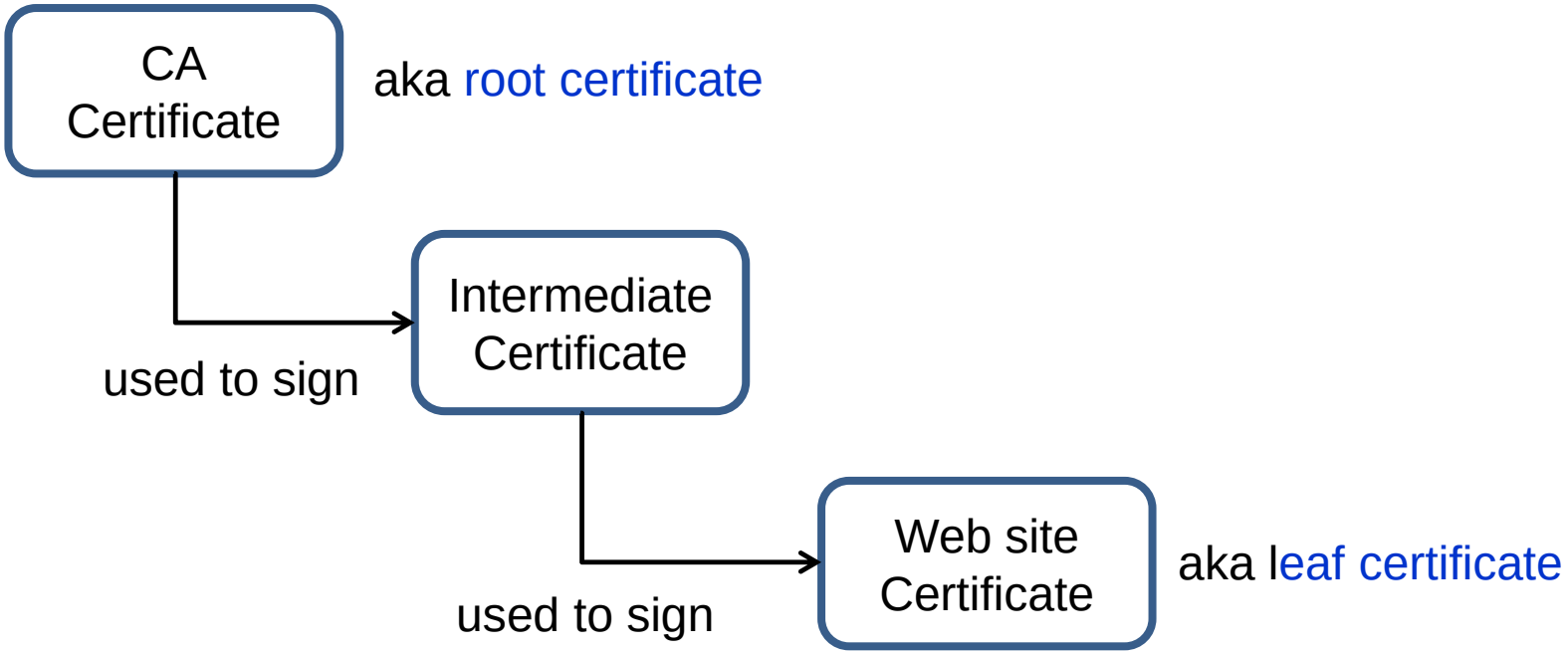
   by extending the certificate chain,

   – incorrectly but without the browser noticing

Certificates are chained



Certificates are chained



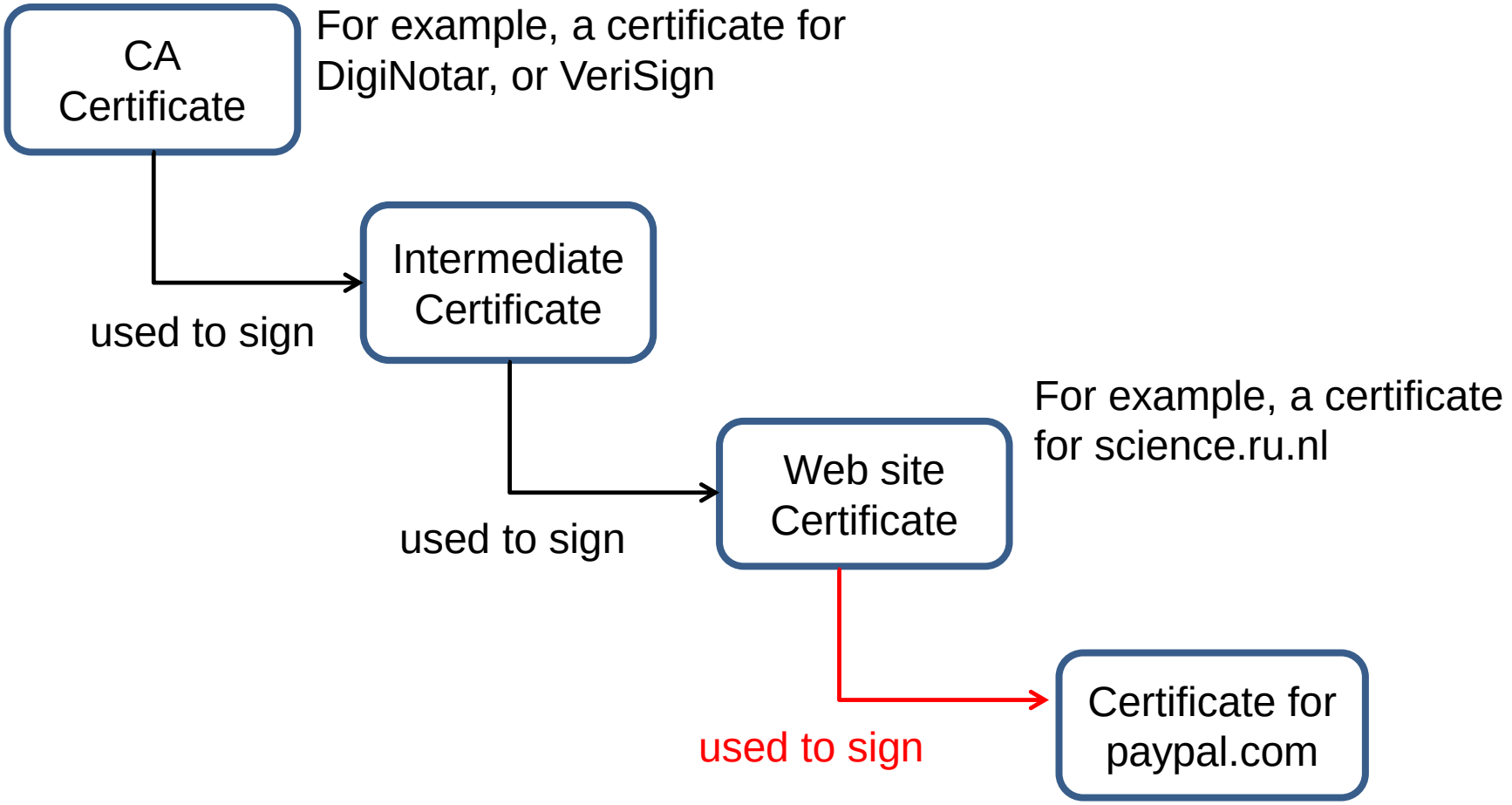
How would you implement checking these chains?

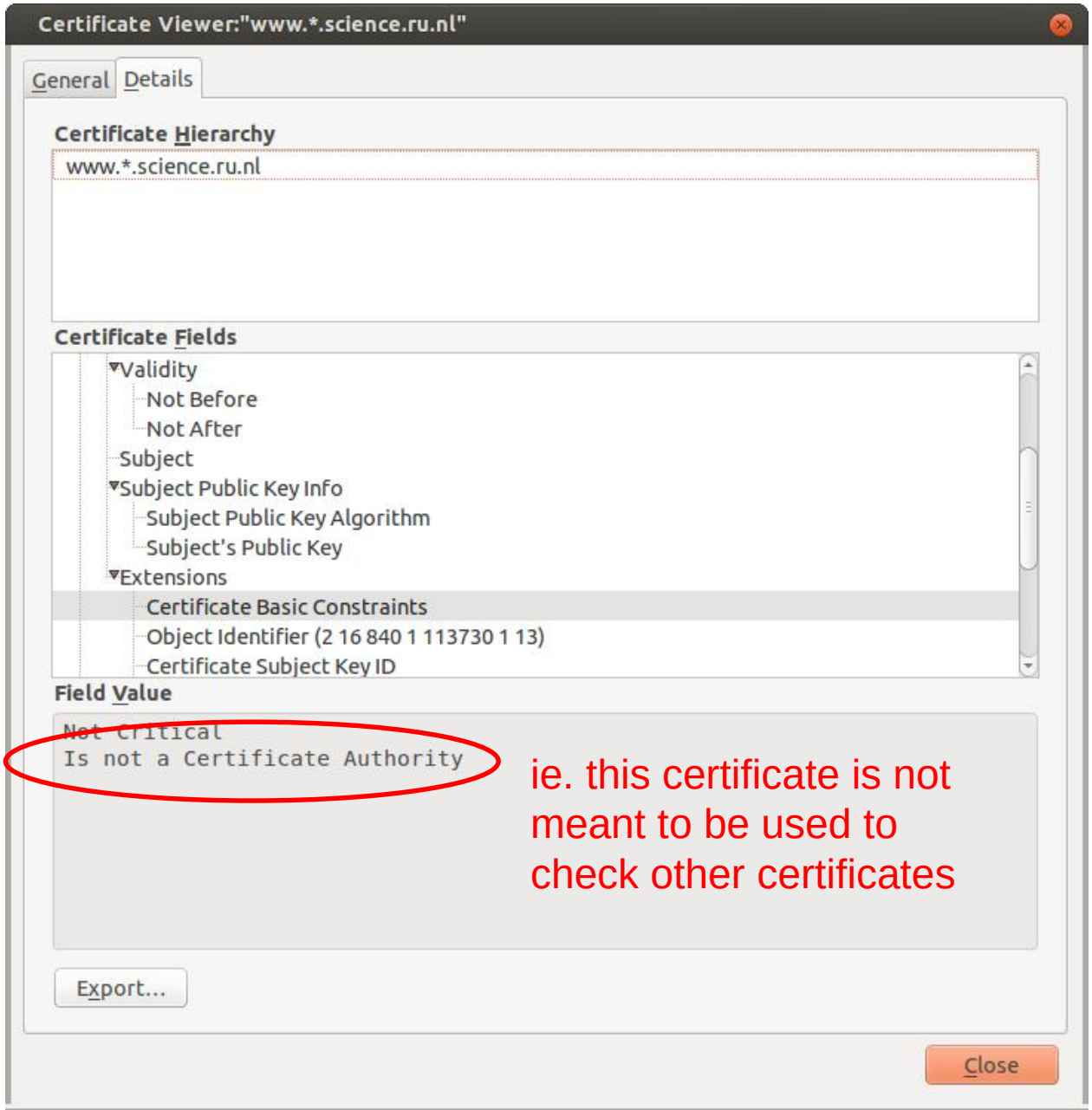
Eg in a web-browser or TLS/SSL library

1. Check that the leaf node as the name of the site you are connecting to
2. Check that the leaf node hasn't expired
3. Check the signature
4. If the signing certificate is in the list of root CAs, then stop.
5. Otherwise, go up the chain, and start again with 2

CA = Certificate Authority

Oops...


CA Certificate
For example, a certificate for DigiNotar, or VeriSign
used to sign
Intermediate Certificate
used to sign
Web site Certificate
For example, a certificate for science.ru.nl
used to sign
Certificate for paypal.com


Certificate Viewer:"www.*.science.ru.nl"
General
Details
Certificate Hierarchy
www.*.science.ru.nl
Certificate Fields
Validity
Not Before
Not After
Subject
Subject Public Key Info
Subject Public Key Algorithm
Subject's Public Key
Extensions
Certificate Basic Constraints
Object Identifier (2 16 840 1 113730 1 13)
Certificate Subject Key ID
Field Value
Not Critical
Is not a Certificate Authority
ie. this certificate is not meant to be used to check other certificates
Export...
Close

Problems checking certificate chains (historic)

Two problems with certificate chains

1. Some CAs did not set CA=FALSE in Basic Constraints

2. Some browsers did not not check it, and allowed leaf certificates to be used to sign other certificates

Such bugs should now all be gone...

But*: history repeats itself...*

Countermeasures to SSL stripping

- HSTS (HTTP Strict Transport Security)

  Server declares *"I only talk HTTPS"*

  ```
  HTTP(S) Response Header:
  Strict-Transport-Security: max-age=15768000;
  includeSubDomain
  ```

  This would stop the browser from *ever* issuing an HTTP request to that domain.

- use HTTPS Everywhere browser plugin

- Giving that CAs may not be trustworthy anyway, Chrome will now check for suspicious certificates issued for google.com

Just when you think it's all over…

Newer attacks on software handling X509 certificates:

When talking to a CA to request certificates, an attacker may try

- including a null character in the Common Name for which he request a certificate, eg.

  `paypal.com[null]mafia.com`

  – Different libraries interpret this differently!
  Which does the CA use and which does the browser use?
- create confusing certificates with multiple Common Names, eg.

  `paypal.com,mafia.com`

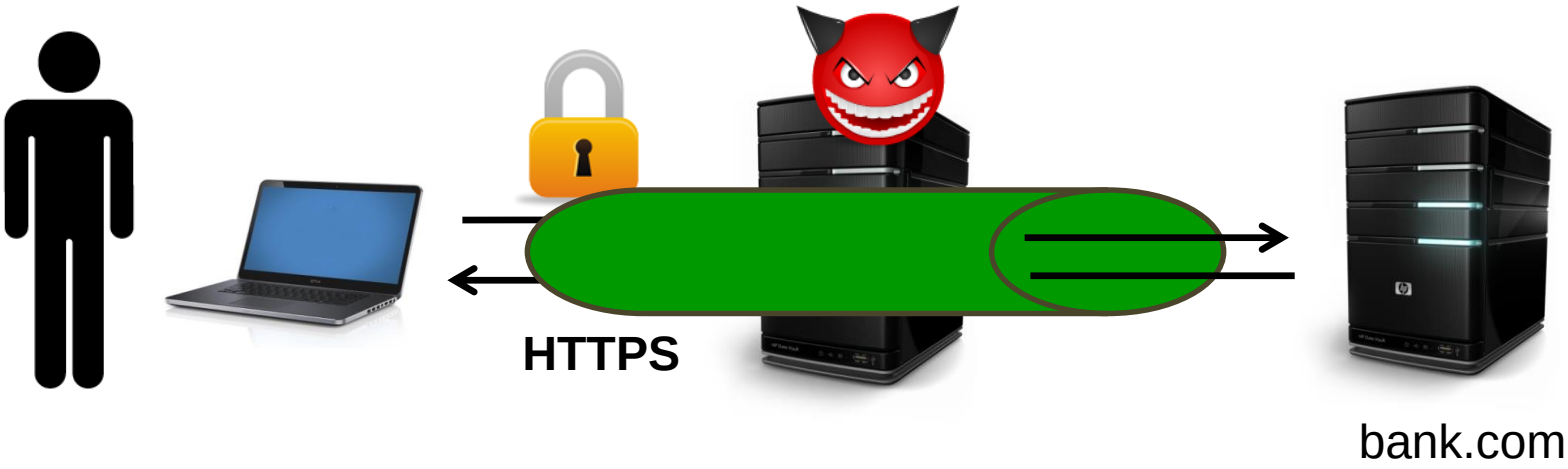
  – Internet Explorer will respect all names in the list, Firefox will only respect the last one, and how has the CA interpreted this?
- a SQL injection attack in the Certificate Signing Request to the CA
- …

Other attacks on sessions (out of scope for this course)

- Cryptographic attacks on TLS
  - Inherent cryptographic weakness in TLS/SSL is that encryption and MAC-ing are done the wrong way around
    - you should first encrypt, and then MAC
  - Some cryptographic primitives used may be weak
    - eg using MD5 as hash function in certificates
- attacks on DNS
  - eg DNS cache poisoning
  - will be discussed in the Networking Security course next semester

Advanced cookie stealing: without breaking HTTPS tunnel

Even without breaking the HTTPS tunnel, an attacker may be able to steal cookies



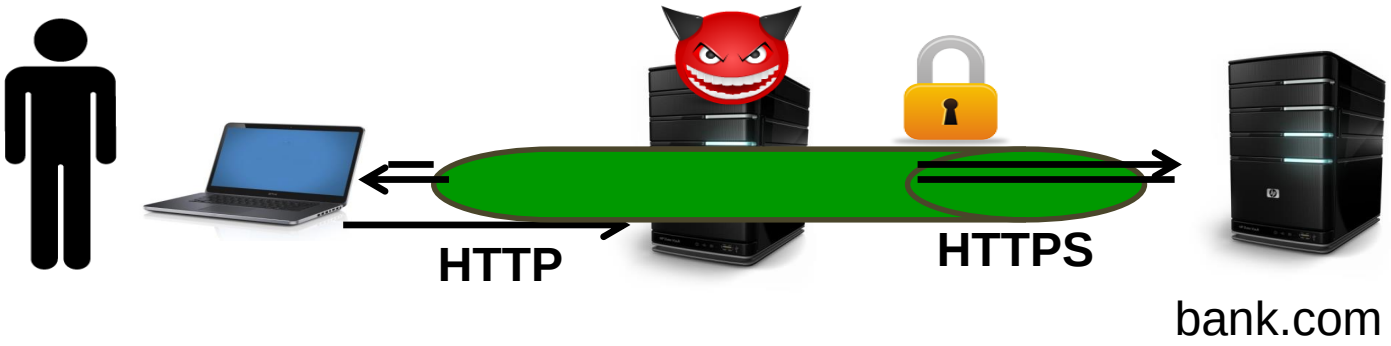
namely if bank.com fails to set the secure flag for its cookie.

(Recall: the secure flag means the cookie is only sent over HTTPS)

Alternative MitM attack: stealing HTTPS cookies

Attack steps

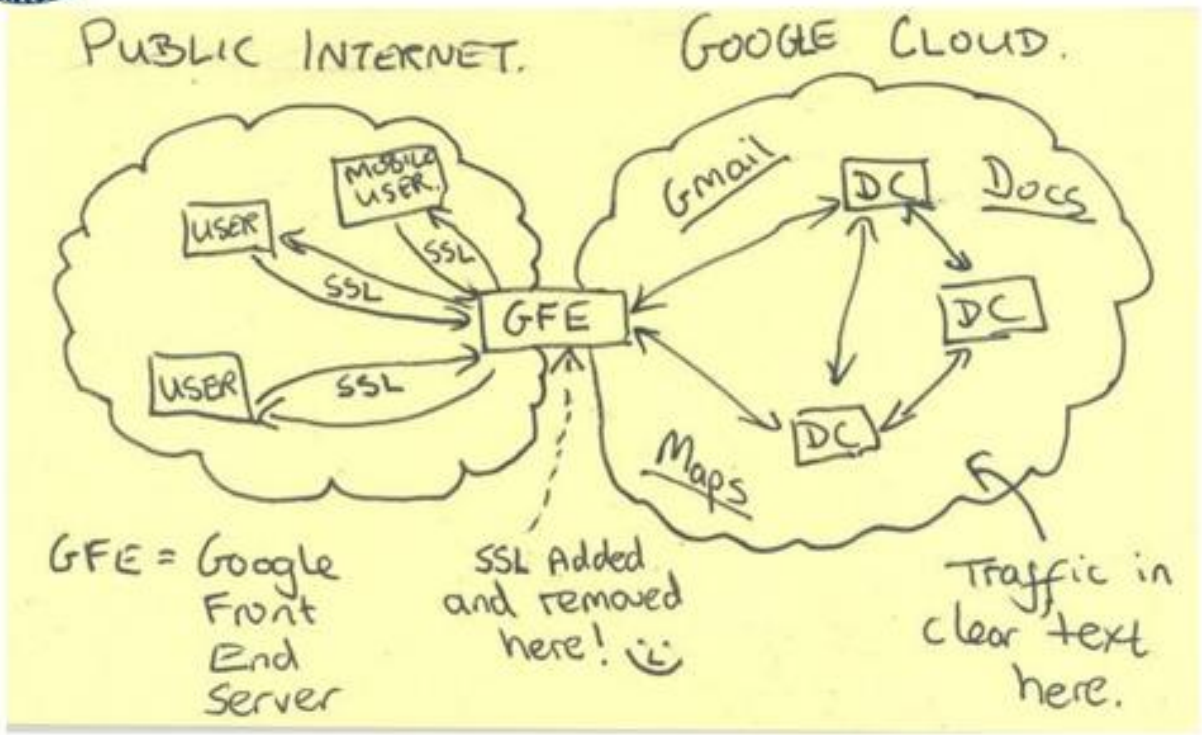
1. user logs on to HTTPS://bank.com
2. server sets session ID for bank.com in cookie
   – which is encrypted in HTTPS-traffic
3. user ask for an unencrypted HTTP request (eg for http://nu.nl)
4. MitM attacker replies with a redirect to http://bank.com
5. Browser follows redirect and sends the bank's cookie over HTTP
6. Bingo! Attacker has the cookie



Still, using TLS is a hurdle to an attacker

TOP SECRET//SI//NOFORN

Current Efforts - Google



TOP SECRET//SI//NOFORN

To watch

- For another explanation of SSL stripping, see the video of Moxie Marlinspike's presentation at DEFCON

  This also presents

  – an improvement, where session cookies that already exists when the MitM attack starts are invalidated to force users to re-enter username & password
  – experimental results of running SSL-stripping at a Tor exit node