



Web Security — New Browser Security Technologies

Tobias Gondrom

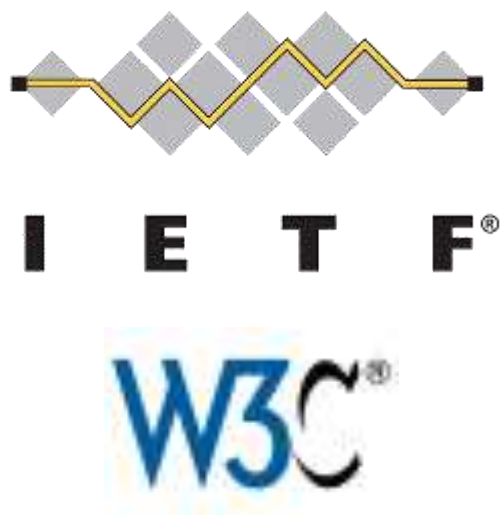
OWASP London

OWASP Global Industry Committee

Chair of IETF Web Security WG

tobias.gondrom@gondrom.org

Tobias Gondrom



- 12 years information security experience (Global Head of Security, CISO, CTO)
- 10 years application development experience
- Information Security & Risk Management, CISO Research and Advisory, Managing Director, Thames Stanley Ltd.
- Author of Standards on Digital Signatures and Secure Archiving
- Chair of IETF Web Security Working Group <http://datatracker.ietf.org/wg/websec/charter/> Member of the IETF Security Directorate
- Invited expert at W3C WebAppSec WG
- London OWASP chapter board member OWASP Global Industry Committee www.owasp.org





Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When

Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When



The Past: CA breaches

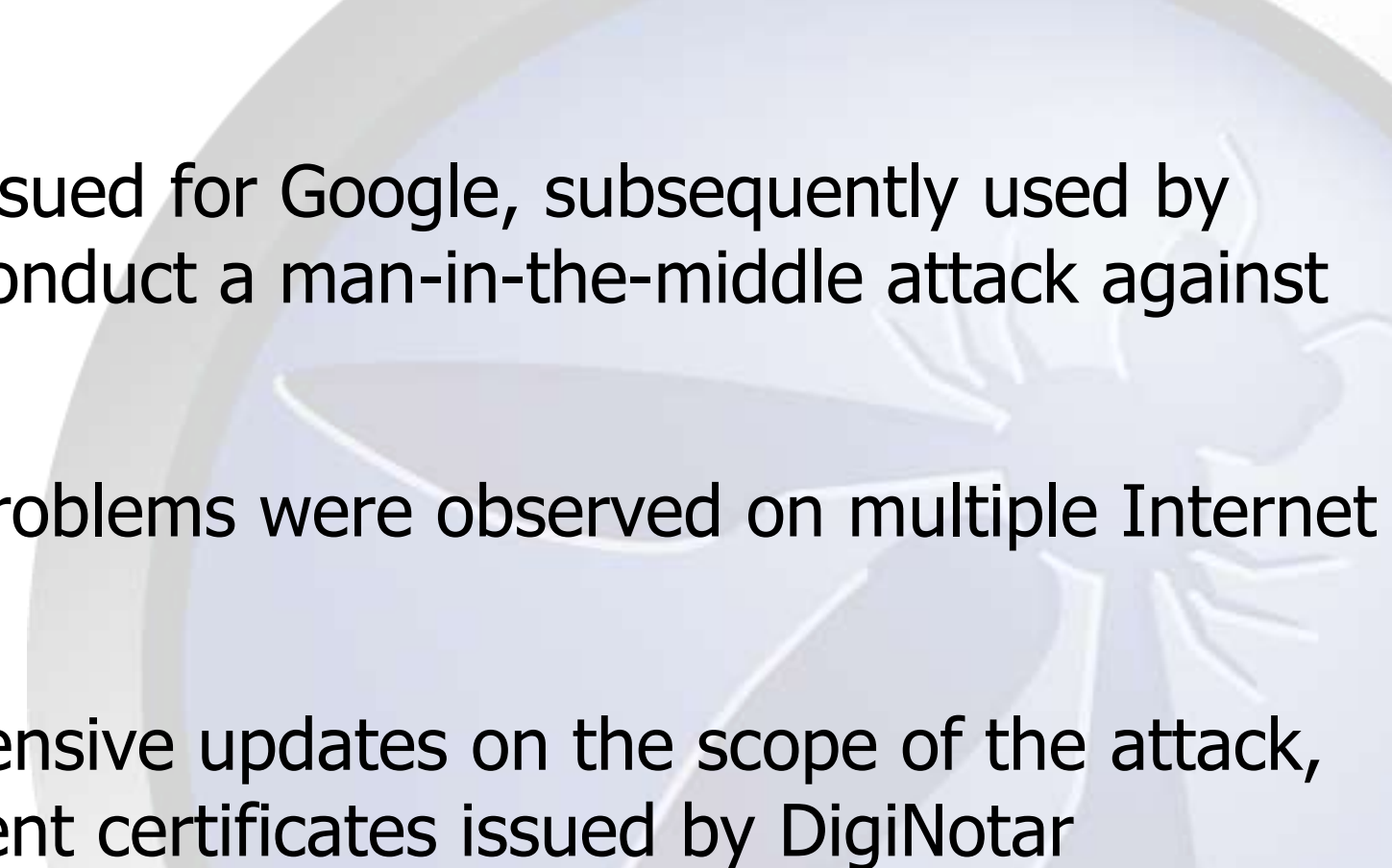
March 15th 2011: Comodo breach

- Nine fake certificates for seven domains were issued: mail.google.com, login.live.com, www.google.com, login.yahoo.com (three certificates), login.skype.com, addons.mozilla.org, and global trustee
- Hacked several times afterwards



The Past: CA breaches

June (?) 2011: DigiNotar breach

- Discovered on June 19th
 - July 10, 2011: wildcard cert issued for Google, subsequently used by unknown persons in Iran to conduct a man-in-the-middle attack against Google services
 - August 28, 2011, certificate problems were observed on multiple Internet service providers in Iran
 - Tor Project has published extensive updates on the scope of the attack, including a list of 531 fraudulent certificates issued by DigiNotar
- 

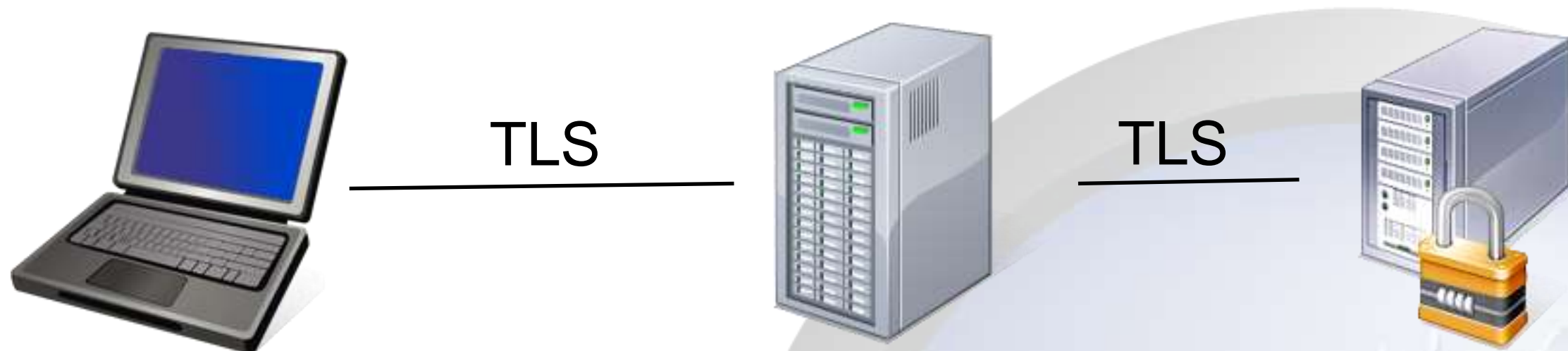


The Past: CA breaches

June (?) 2011: DigiNotar breach

- All browser vendors remove trust of DigiNotar swiftly, e.g. August 30, 2011: Mozilla removed DigiNotar certificates from their list of trusted CAs (via patches etc.)
- September 20, 2011 – DigiNotar filed for bankruptcy
- Remark: Google Chrome users were protected from this attack because Chrome was able to detect the fraudulent certificate due to pinning.
- Statements have appeared that the DigiNotar attacker is the same person who attacked Comodo earlier
- The attacker claims to be an individual Iranian who has chosen to help the government monitor individuals' communications. Additionally, he claims to have compromised four additional as-yet-unspecified certificate authorities.

MITMA - TLS attack



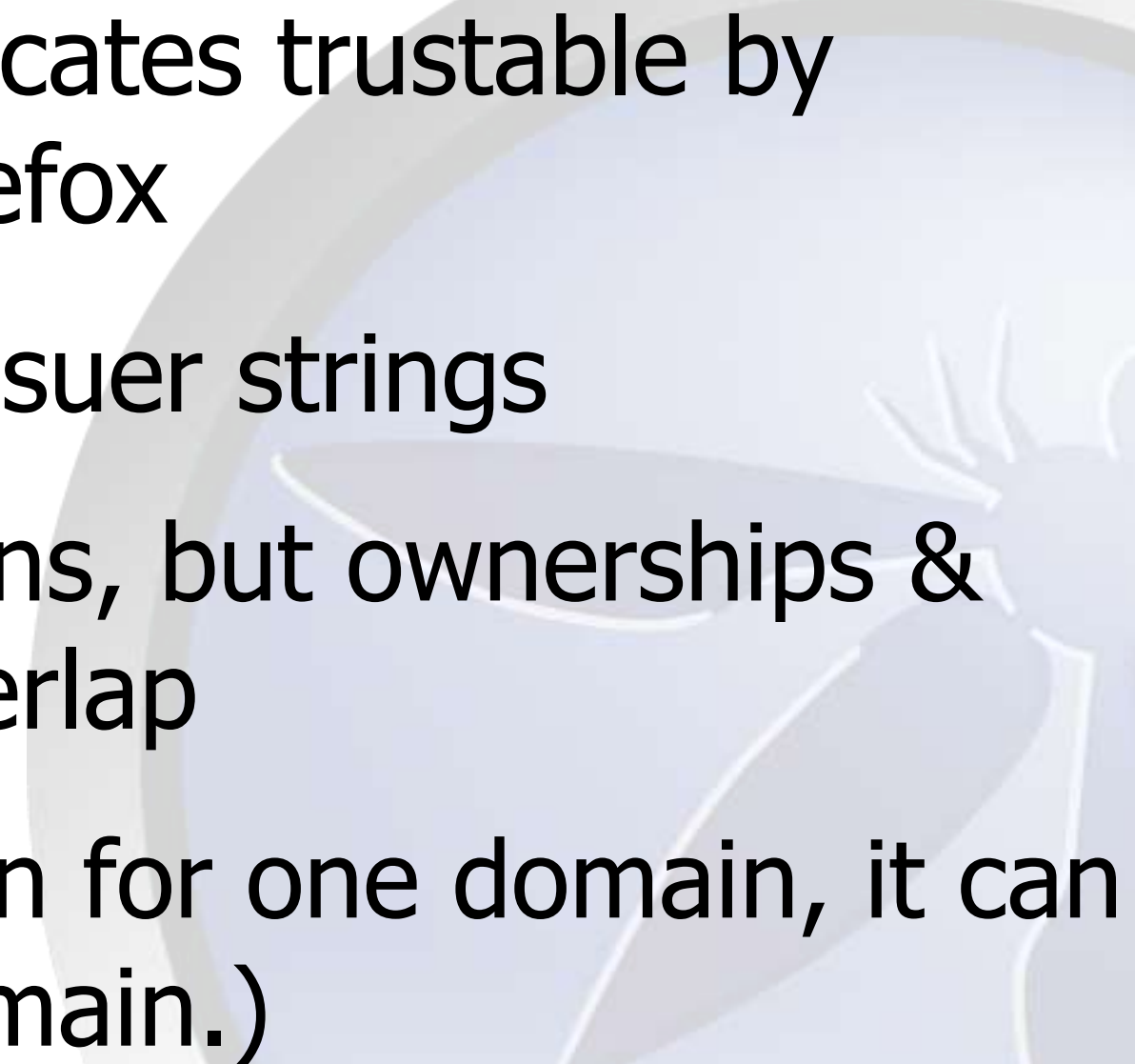
Attacker replaced Server cert with own compromised cert and could read all communication (incl. passwords) in the clear

The situation

- Browsers trust CA certificates for all domains equally (any trusted CA can sign for any identity, true or fake, e.g. google.com, paypal.com, ...)
- hundreds of CAs
- From 46 countries/jurisdictions
- If a single one is broken, all TLS/SSL domains are prone to attacks



From EFF: SSL Observatory

- 1,482 CA Certificates trustable by Windows or Firefox
 - 1,167 distinct issuer strings
 - 651 organizations, but ownerships & jurisdictions overlap
 - (If a CA can sign for one domain, it can sign for any domain.)
- 

Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When

OWASP Top 10 – Insufficient Transport Layer Protection

A1: Injection

A2: Cross-Site Scripting (XSS)

A3: Broken Authentication and Session Management

A4: Insecure Direct Object References

A5: Cross Site Request Forgery (CSRF)

A6: Security Misconfiguration

A7: Failure to Restrict URL Access

A8: Insecure Cryptographic Storage

A9: Insufficient Transport Layer Protection

A10: Unvalidated Redirects and Forwards

What's the problem

- Some are not using / not mandating TLS/SSL
 - Relies on trust relationships (trust on first use / trusted source)
 - Weak channel protection
 - Authentication & leakage of credentials
- => Today, Web Applications try to fix this on the Application level with little support of the underlying infrastructure

A9 – Insufficient Transport Layer Protection

Transmitting sensitive data insecurely

- Failure to identify all sensitive data
- Failure to identify all the places that this sensitive data is sent
 - On the web, to backend databases, to business partners, internal communications
- Failure to properly protect this data in every location

Typical Impact

- Attackers access or modify confidential or private information
 - e.g, credit cards, health care records, financial data (yours or your customers)
- Attackers extract secrets to use in additional attacks
- Company embarrassment, customer dissatisfaction, and loss of trust
- Expense of cleaning up the incident
- Business gets sued and/or fined

Still not using SSL?

Now:
Redirect to https
before login



Firesheep downloaden

Firesheep is een gratis add-on voor de [Firefox](#) webbrowser waarmee iedereen een niet-versleuteld Wifi-netwerk kan scannen en wachtwoorden kan onderscheppen van andere gebruikers op dat netwerk.

Na installatie van Firesheep verschijnt er in de Firefox browser een nieuwe sidebar. Nadat u een connectie heeft gemaakt met een onbeveiligd netwerk klikt u op de knop "Start Capturing". Wanneer andere personen die verbonden zijn met dit netwerk inloggen op een website waarmee Firesheep bekend is worden de naam en foto van die gebruikers in de sidebar weergegeven. Wanneer u dubbelklikt op een persoon logt u in als die gebruiker op bijvoorbeeld Facebook, Twitter of Flickr.

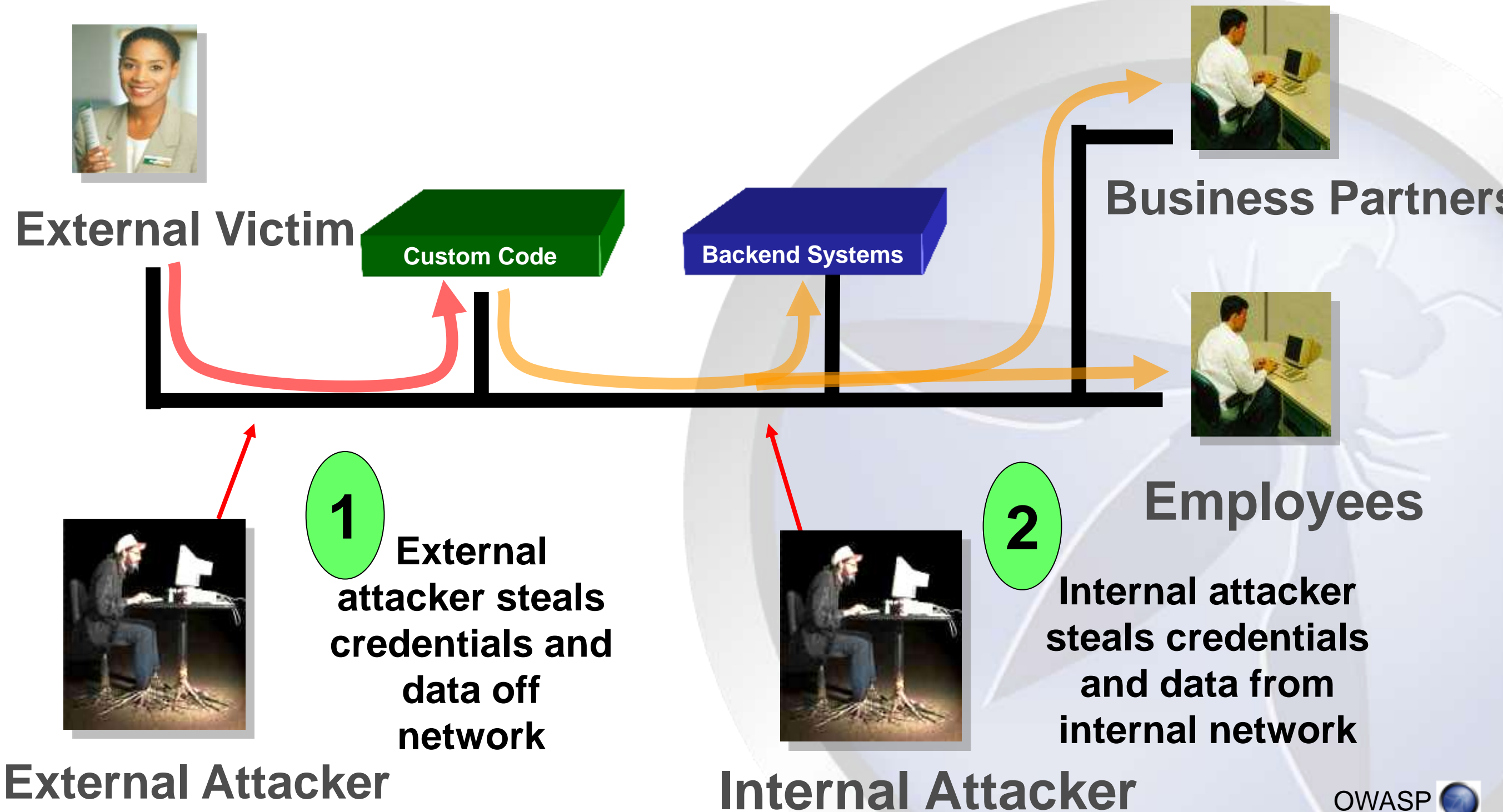
Met het openbaar maken van deze software wil de ontwikkelaar, Eric Butler, de aandacht vestigen op de gevaren van niet-versleutelde websites.

Mozilla, de ontwikkelaar van [Firefox](#), heeft aangegeven de Firesheep add-on niet te zullen blokkeren.

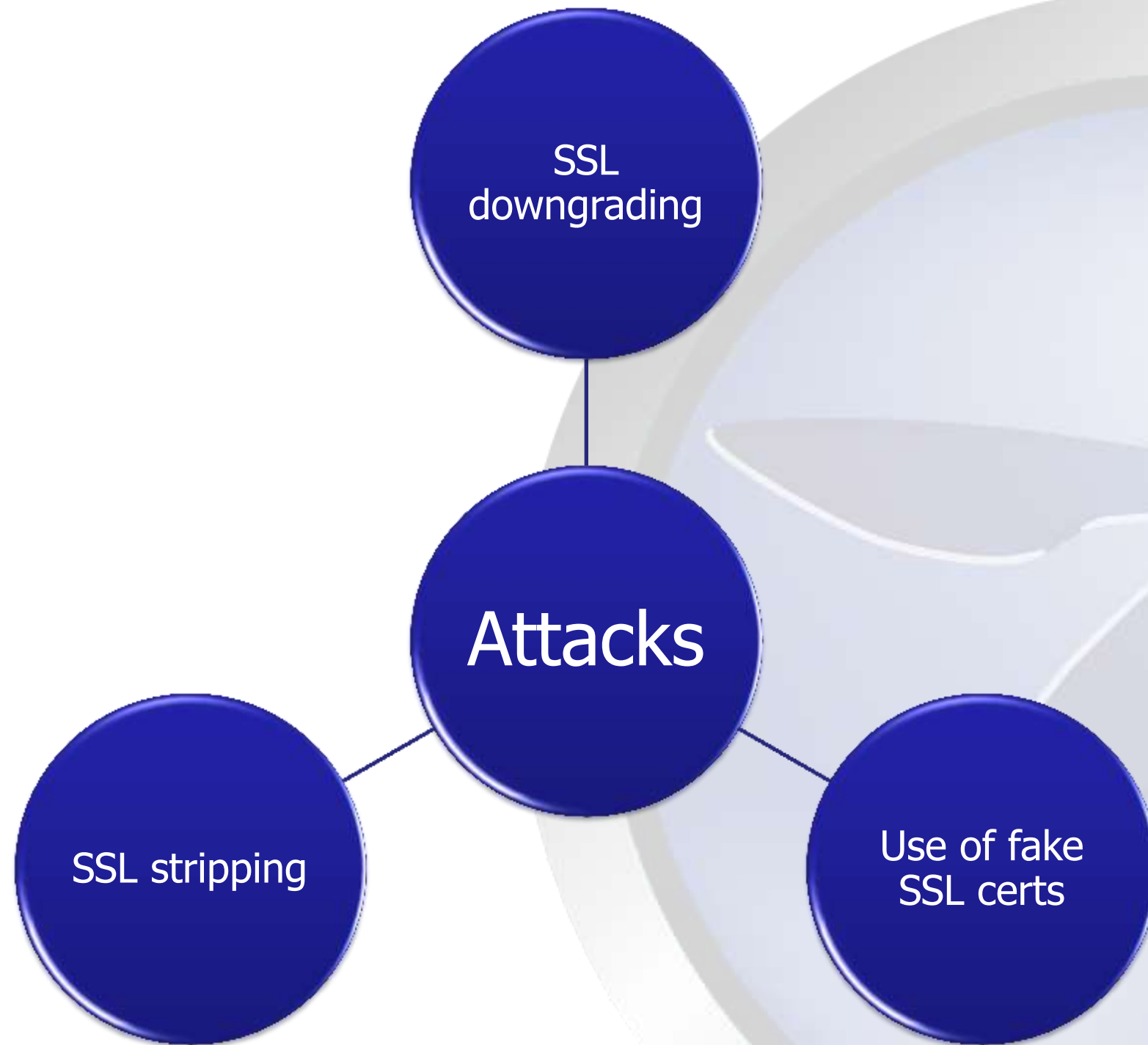
Firesheep heeft de volgende kenmerken:

- gratis [Firefox](#) add-on
- gebruikersnamen en wachtwoorden onderscheppen van andere gebruikers op een openbaar Wifi netwerk
- werkt voor onder andere accounts op Facebook, Twitter, Google, Flickr, Amazon en Bit.ly
- [open source](#)
- beschikbaar voor Windows en Mac

Insufficient Transport Layer Protection



Common attack vectors



Moxie's SSL Strip



user



MitM



Data centre

Terminates
SSL

Changes
https to http

Normal
https to the
server

Acts as
client

Moxie's SSL Strip



user



MitM



Data centre

Secure cookie?

Encoding, gzip?

Cached content?

Sessions?

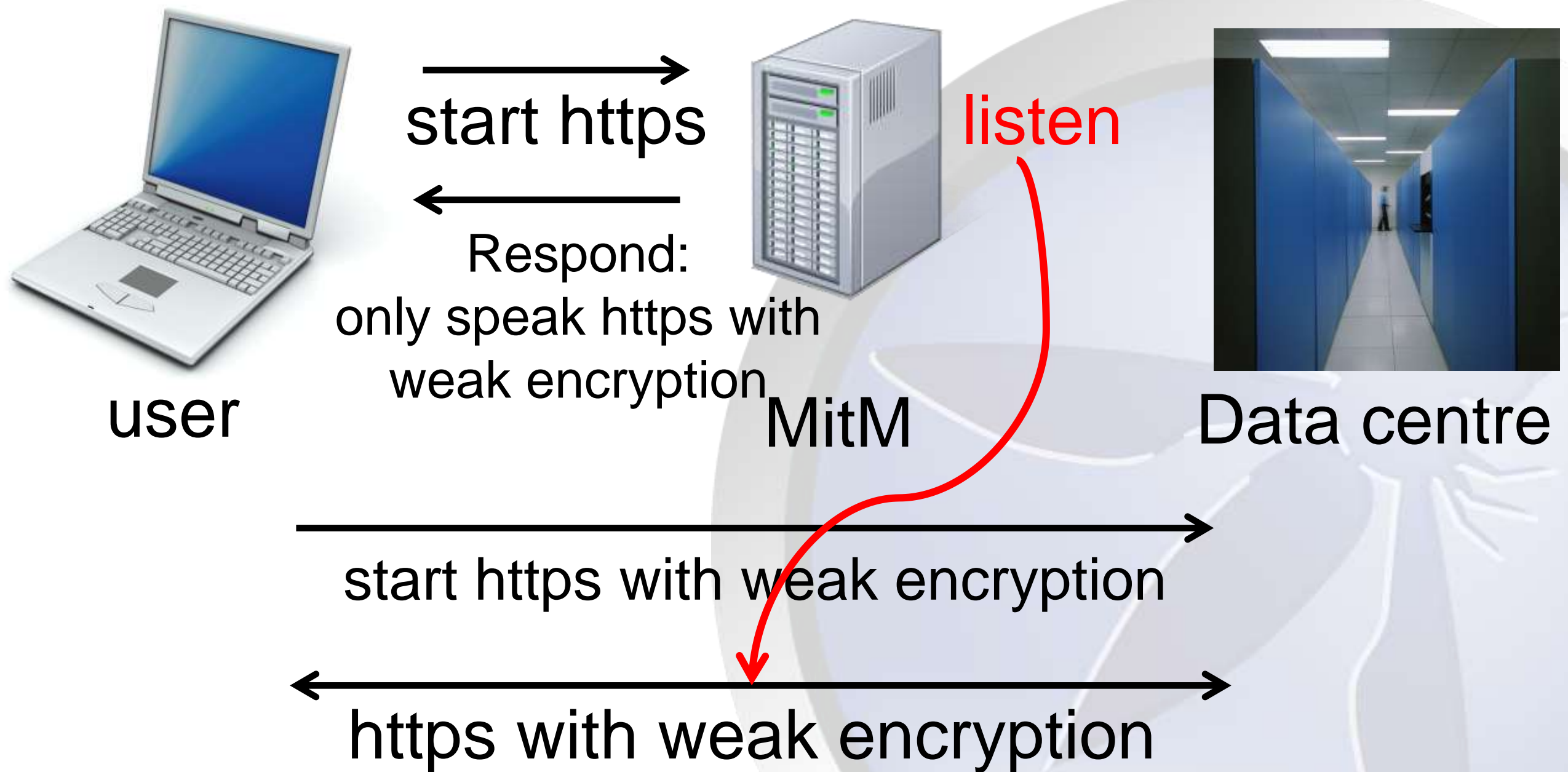
Strip the secure attribute off all cookies.

Strip all encodings in the request.

Strip all if-modified-since in the request.

Redirect to same page, set-cookie expired

SSL downgrading



A9 – Avoiding Insufficient Transport Layer Protection

Use the mechanisms correctly

- Use TLS on **all** connections with sensitive data
- Use standard strong algorithms (disable old SSL algorithms)
- Manage keys/certificates properly
- Verify SSL certificates before using them
- Use proven mechanisms when sufficient
 - E.g., SSL vs. XML-Encryption

See: http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet for more details

Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When

Who – Introducing the Players



- OWASP
 - Top Ten
 - Browser Security Day at OWASP Summit



- IETF
 - Web Security WG
- Browser Vendors
- Secure Web-sites of critical information and payment systems (e.g. paypal, google, ebay, ...)
- Security Researchers and Plug-in developers for browsers

What's been done / what's coming

- Secure the Channel:
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
 - TLS cert pinning in DNSSEC

HSTS - Secure Channels: Strict Transport Security

- Server declares **"I only talk TLS"**
- Example:
HTTP(S) Response Header:
Strict-Transport-Security: max-age=15768000;
includeSubDomains
- Header can be cached and also prevents leakage via subdomain-content through non-TLS links in content
- Weakness: "Trust on first use"
 - Possible pre-loaded HSTS in browsers
- Already first deployments

Cert Pinning (1)

draft-ietf-websec-key-pinning-01

- Server identities tend to be long-lived, but clients have to re-establish the server's identity on every TLS session.
- How could Google/Chrome be resilient to DigiNotar attack?
- Google built-in in Chrome "preloaded" fingerprints for the known public keys in the certificate chains of Google properties. Thereby exposed the false *.google.com certificate DigiNotar signed.

Cert Pinning (2)

But....

.....preloading does not scale, so we need something dynamic:

=> Could use an HTTP header

i.e. transmit the SHA1 or SHA256 hash of the Subject Public Key Info structure of the X.509 certificate. (You could pin to end entity, intermediary, root. Select your degree of precision.)

Cert Pinning - Syntax

```
Header add Public-Key-Pins "max-  
age=10000; pin-  
sha1=\"ObT42aoSpAqWdY9WfRfL7i0H  
sVk=\"; pin-  
sha1=\"hvfkN/qlp/zhXR3cuerq6jd2Z7g=\n\""
```


Cert Pinning - parameters

- List at least 2 certs: 1 live pin (a hash of an SPKI in the current cert chain) and at least one backup pin (a hash of an SPKI not in the current cert chain).
- Clients remember the most recently seen set of pins for max-age seconds after it was most recently seen.
- Clients drop TLS connections if not using the listed certs.

Cert Pinning – possible problems

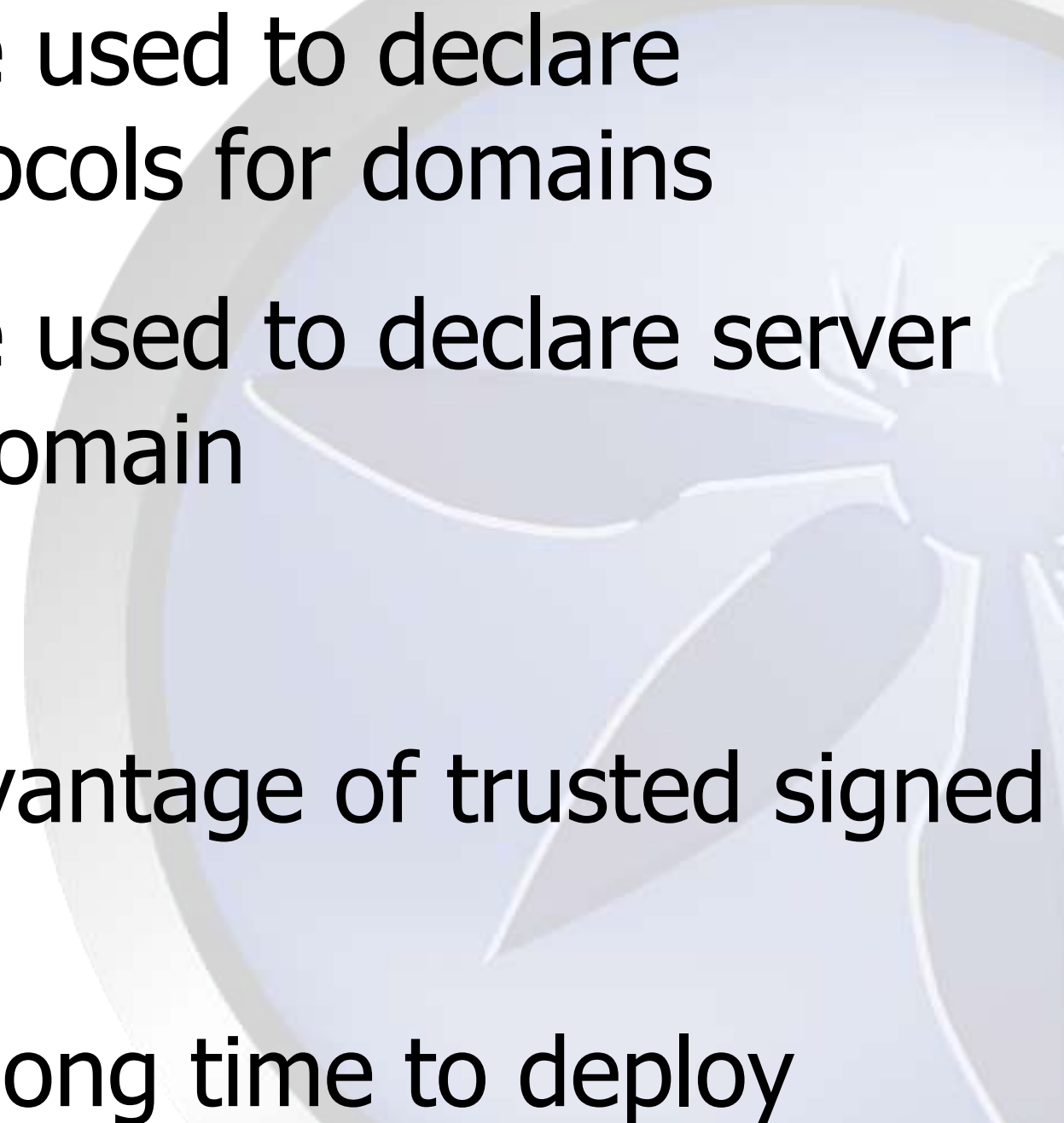
Possible Problems:

- Bootstrap – “trust on first use”
 - Pre-loaded browser
- Servers might accidentally "brick" themselves (pin for a long time to an SPKI which is later lost, for example) – reason why backup cert is mandatory
- Attackers with ISP capabilities / man-in-the-middle access may try to “brick” domains for users even when outside of their reach (imagine: Iranian travelling abroad and no longer able to access Google, etc.)
 - Recovery / cache flush mechanisms



Other Methods:

Secure Channels: DNSSEC for TLS

- DNSSEC can be used to declare supported protocols for domains
 - DNSSEC can be used to declare server certificate for domain
 - Advantage: Advantage of trusted signed source
 - Disadvantage: long time to deploy
- 

Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When



New Protection against XSS and Clickjacking

X-Frame-Options and CSP

Common Threats from XSS and Clickjacking:

- Bootstrap – “trust on first use”
 - Recovery / cache flush mechanisms

A2 – Cross-Site Scripting (XSS)

Occurs any time...

- Raw data from attacker is sent to an innocent user's browser

Raw data...

- Stored in database
- Reflected from web input (form field, hidden field, URL, etc...)
- Sent directly into rich JavaScript client

Virtually every web application has this problem

- Try this in your browser – `javascript:alert(document.cookie)`

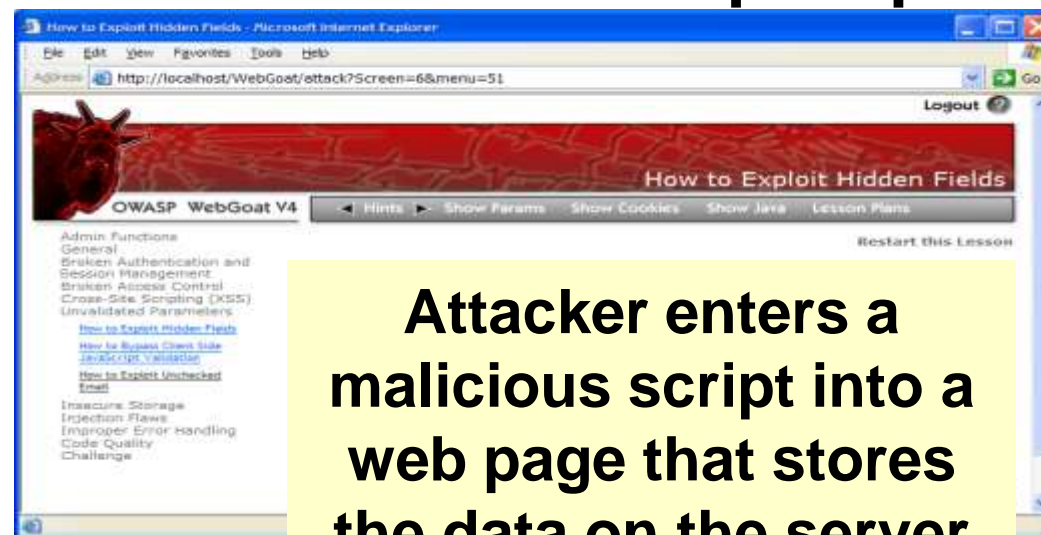
Typical Impact

- Steal user's session, steal sensitive data, rewrite web page, redirect user to phishing or malware site
- Most Severe: Install XSS proxy which allows attacker to observe and direct all user's behavior on vulnerable site and force user to other sites

Cross-Site Scripting Illustrated

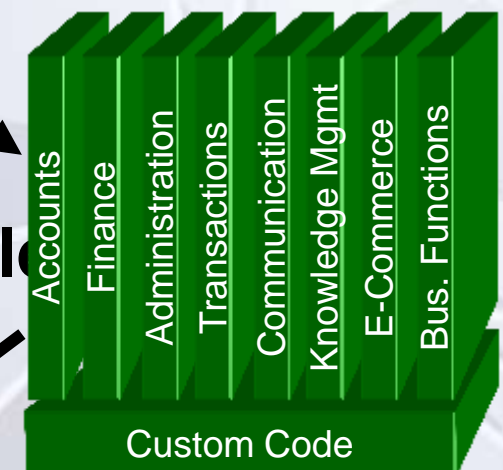
1

Attacker sets the trap – update my profile



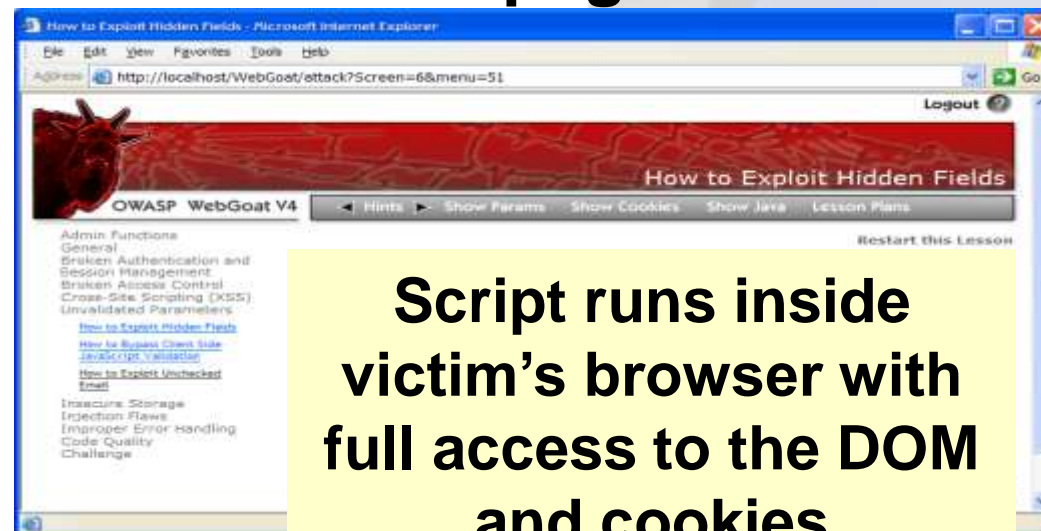
Attacker enters a malicious script into a web page that stores the data on the server

Application with stored XSS vulnerability



2

Victim views page – sees attacker profile



Script runs inside victim's browser with full access to the DOM and cookies

3

Script silently sends attacker Victim's session cookie

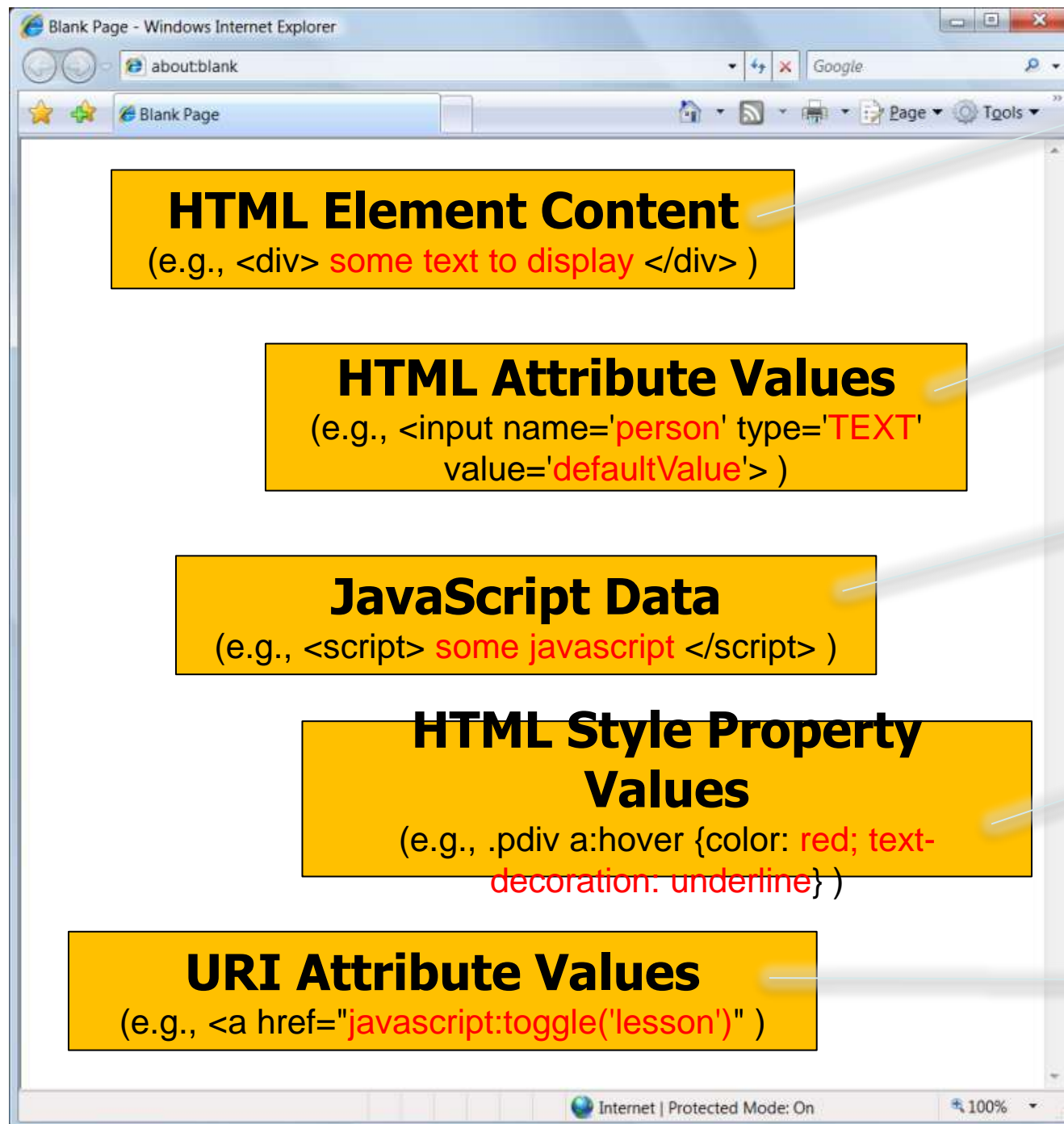
A2 – Avoiding XSS Flaws

Recommendations

- Eliminate Flaw
 - Don't include user supplied input in the output page
- Defend Against the Flaw
 - Primary Recommendation: Output encode all user supplied input (Use OWASP's ESAPI to output encode:
<http://www.owasp.org/index.php/ESAPI>)
 - Perform 'white list' input validation on all user input to be included in page
 - For large chunks of user supplied HTML, use OWASP's AntiSamy to sanitize this HTML to make it safe
See: <http://www.owasp.org/index.php/AntiSamy>
- **Use Content Security Policy!**

References: For how to output encode properly, read the new
[http://www.owasp.org/index.php/XSS_\(Cross Site Scripting\) Prevention Cheat Sheet](http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

Safe Escaping Schemes in Various HTML Execution Contexts



#1: (&, <, >, ") → &entity; (', /) → &#xHH;
ESAPI: encodeForHTML()

#2: All non-alphanumeric < 256 → &#xHH
ESAPI: encodeForHTMLAttribute()

#3: All non-alphanumeric < 256 → \xHH
ESAPI: encodeForJavaScript()

#4: All non-alphanumeric < 256 → \HH
ESAPI: encodeForCSS()

#5: All non-alphanumeric < 256 → %HH
ESAPI: encodeForURL()

ALL other contexts CANNOT include Untrusted Data

Recommendation: Only allow #1 and #2 and disallow all others

See: [www.owasp.org/index.php/XSS_\(Cross_Site_Scripting\)_Prevention_Cheat_Sheet](http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)
for more details

New Protection against XSS and Clickjacking

X-Frame-Options and CSP

- `"Content-Security-Policy:" 1#policy`
- `"Content-Security-Policy-Report-Only:" 1#policy`

`directive-name = "default-src"`

`directive-value = source-list`

Example:

```
Content-Security-Policy: default-src 'self';  
script-src example.com
```


New Protection against XSS and Clickjacking

X-Frame-Options and CSP

`script-src`

`object-src`

`style-src`

`img-src`

`media-src`

`frame-src`

`font-src`

`connect-src`

Frame-Options - History

X-Frame-Options widely deployed/used to prevent Click-jacking

- Running code and (some) consensus by implementers in using X-FRAME-OPTIONS

HTTP-Header:

- DENY: cannot be displayed in a frame, regardless of the site attempting to do so.
- SAMEORIGIN: can only be displayed if the top-frame is of the same "origin" as the page itself.



Frame-Options – Example Use-Cases

A.1. Shop

- An Internet Marketplace/Shop link/button to "Buy this" Gadget, wants their affiliates to be able to stick the "Buy such-and-such from XYZ" IFRAMES into their pages.

A.2. Confirm Purchase Page

- Onlineshop "Confirm purchase" anti-Click-Jacking page. The Confirm Purchase page must be shown to the end user without possibility of overlay or misuse by an attacker.

Frame-Options

In EBNF:

```
Frame-Options = "Frame-Options" ":"  
"DENY" / "SAMEORIGIN" / ("ALLOW-FROM"  
" : "URI)
```

- **DENY**: The page cannot be displayed in a frame, regardless of the site attempting to do so.
- **SAMEORIGIN**: can only be displayed in a frame on the same origin as the page itself.
- **ALLOW-FROM**: can only be displayed in a frame on the specified origin

Protection against Clickjacking with Frame-Options in CSP1.1

```
directive-name    = "frame-options"

directive-value   = 'deny' / 'self' ['top-only']
                    / 1*1<host-source> ['self'] / 1*1<host-
                    source> 'top-only'
```


Web Security – New Browser Security Technologies

- Past Attacks/Breaches
- Insufficient Transport Layer Protection
- Solutions
 - HSTS - HTTP Strict Transport Security
 - Cert Pinning
- New Protection against XSS and Clickjacking
 - X-Frame-Options and CSP
- When

When - Timeframes

HSTS Strict Transport Security – **now**

Cert Pinning Q2 2013

TLS in DNSSEC – 201?

X-Frame-Options – **now** (will be informational RFC in Q2 2013)

CSP 1.0 – **now** - published as a W3C Candidate Recommendation. – Q4 2012

CSP 1.1 – Q? 2013

Join the discussion

Ideas / feedback / participation welcome

IETF Websec:

<http://tools.ietf.org/wg/websec/charters>

W3C WebAppSec:

<http://www.w3.org/2011/webappsec/>

Or drop me an email:

tobias.gondrom@gondrom.org



Questions?





Thank you

