Recent Web Security Technology

Lieven Desmet – iMinds-DistriNet, KU Leuven
Lieven.Desmet@cs.kuleuven.be

SecAppDev Leuven 2014 (13/02/2014, Leuven)

DistriNet
iMinds KU LEUVEN
SEVENTH FRAMEWORK PROGRAMME
STREWS

About myself: Lieven Desmet

@lieven_desmet

- Research manager at KU Leuven
  - (Web) Application Security

- Active participation in OWASP
  - Board member of the OWASP Belgium Chapter
  - Co-organizer of the academic track on past OWASP AppSec Europe Conferences

- Program director at SecAppDev

iMinds-DistriNet, KU Leuven

- Headcount:
  - 10 professors
  - 65 researchers
- Research Domains
  - Secure Software
  - Distributed Software
- Academic and industrial collaboration in 30+ national and European projects

https://distrinet.cs.kuleuven.be

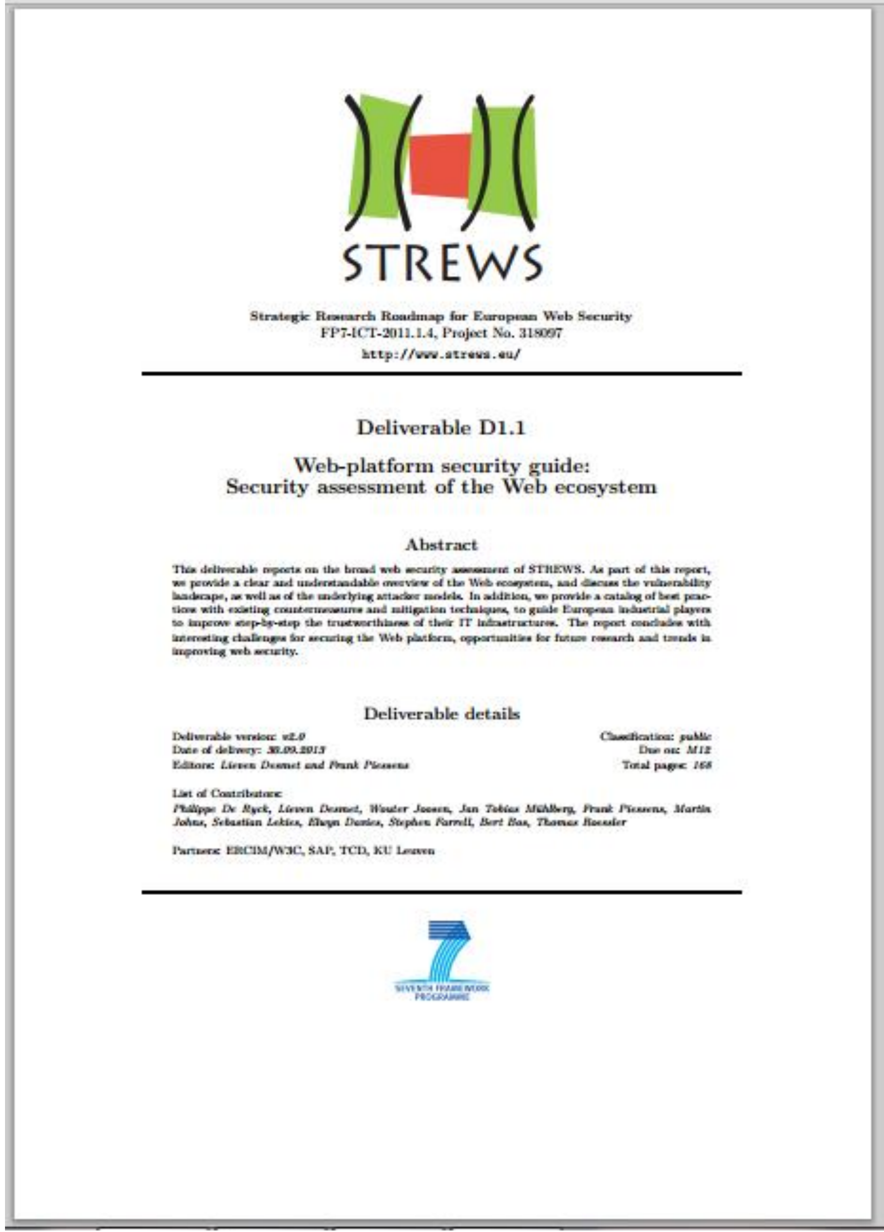
Web Application Security Team

- Web Session management
  - Session hijacking, fixation, SSL stripping, CSRF,...
  - CSRF protection: CsFire
    - 50K downloads
    - Available for Firefox and Chrome
- Web Mashup Security
  - Secure integration of 3rd party JavaScript
  - Information Flow Control for JavaScript
- Various Web Security Assessments
  - HTML5 security analysis for ENISA
  - Large scale assessments of security state-of-practise

Web-platform Security Guide

- Web security overview
  - Vulnerabilities
  - Mitigation techniques
  - Recent research and standardization activities
  - Best practices
- Bundled in 169 pages
  - EU FP7 project STREWS
  - Freely downloadable

http://www.strews.eu/images/STREWS-D1.1-final.pdf



STREWS

Strategic Research Roadmap for European Web Security
FP7-ICT-2011.1.4, Project No. 318097
http://www.strews.eu/

Deliverable D1.1

Web-platform security guide:
Security assessment of the Web ecosystem

Abstract

This deliverable reports on the broad web security assessment of STREWS. As part of this report, we provide a clear and understandable overview of the Web ecosystem, and discuss the vulnerability landscape, as well as of the underlying attacker models. In addition, we provide a catalog of best practices with existing countermeasures and mitigation techniques, to guide European industrial players to improve step-by-step the trustworthiness of their IT infrastructures. The report concludes with interesting challenges for securing the Web platform, opportunities for future research and trends in improving web security.

Deliverable details

Deliverable version: *v2.0*	Classification: *public*
Date of delivery: *30.09.2013*	Due on: *M12*
Editors: *Lieven Desmet and Frank Piessens*	Total pages: *168*

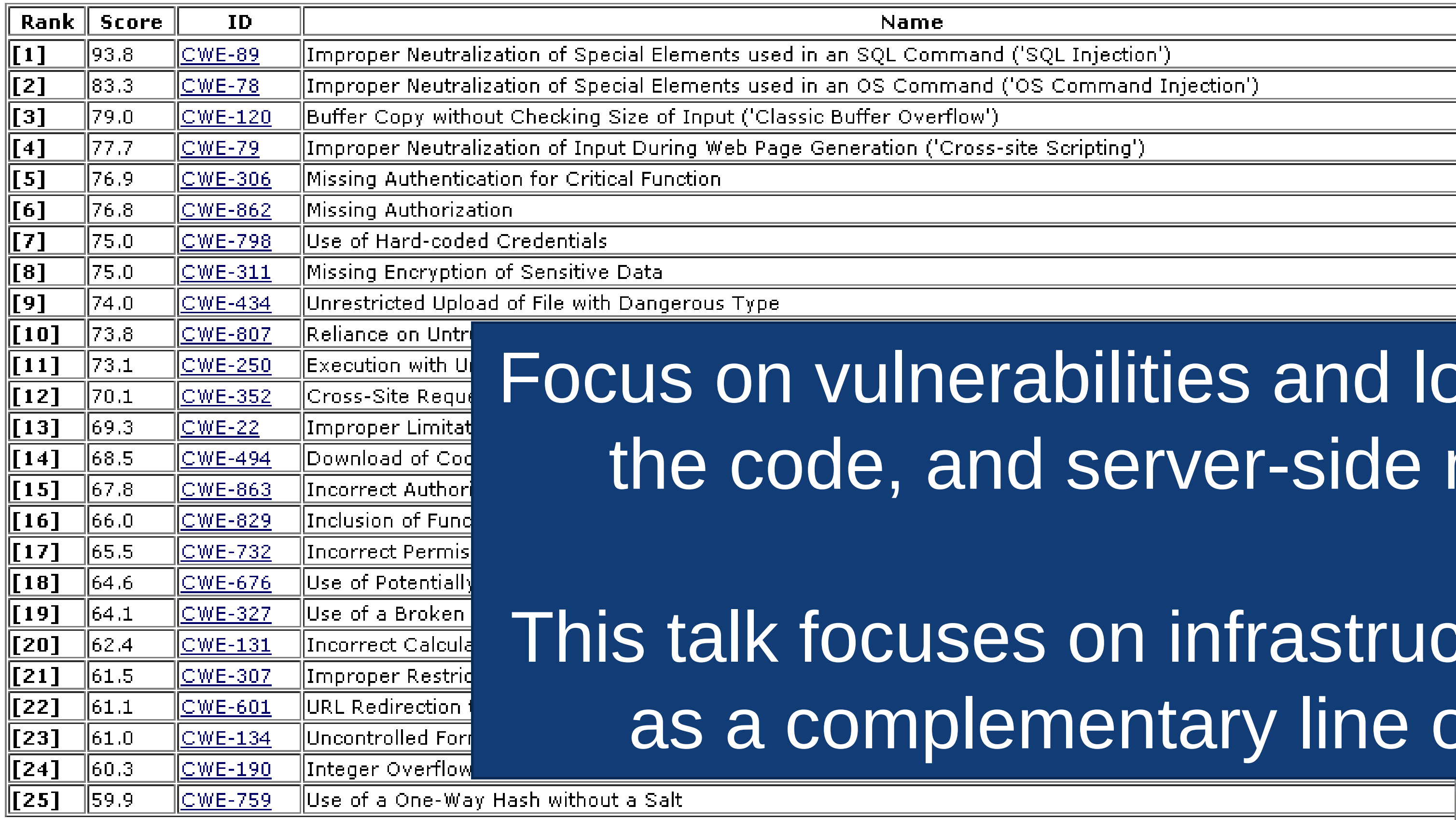
List of Contributors:
Philippe De Ryck, Lieven Desmet, Wouter Joosen, Jan Tobias Mühlberg, Frank Piessens, Martin Johns, Sebastian Lekies, Elwyn Davies, Stephen Farrell, Bert Bos, Thomas Roessler

Partners: ERCIM/W3C, SAP, TCD, KU Leuven

Recent Web Security Technology

Server-side security policies, enforced by the browser

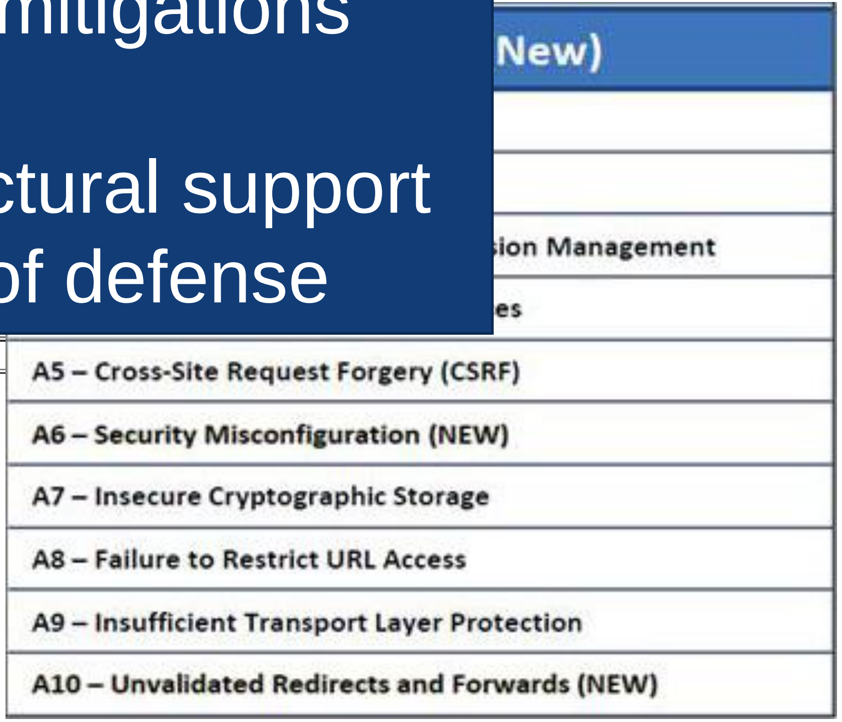
Sans Top 25 - OWASP Top 10



Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untr
[11]	73.1	CWE-250	Execution with U
[12]	70.1	CWE-352	Cross-Site Reque
[13]	69.3	CWE-22	Improper Limitat
[14]	68.5	CWE-494	Download of Cod
[15]	67.8	CWE-863	Incorrect Author
[16]	66.0	CWE-829	Inclusion of Func
[17]	65.5	CWE-732	Incorrect Permis
[18]	64.6	CWE-676	Use of Potentially
[19]	64.1	CWE-327	Use of a Broken
[20]	62.4	CWE-131	Incorrect Calcula
[21]	61.5	CWE-307	Improper Restric
[22]	61.1	CWE-601	URL Redirection t
[23]	61.0	CWE-134	Uncontrolled For
[24]	60.3	CWE-190	Integer Overflow
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Focus on vulnerabilities and logical flaws in the code, and server-side mitigations

This talk focuses on infrastructural support as a complementary line of defense



New)

ion Management

es

A5 – Cross-Site Request Forgery (CSRF)

A6 – Security Misconfiguration (NEW)

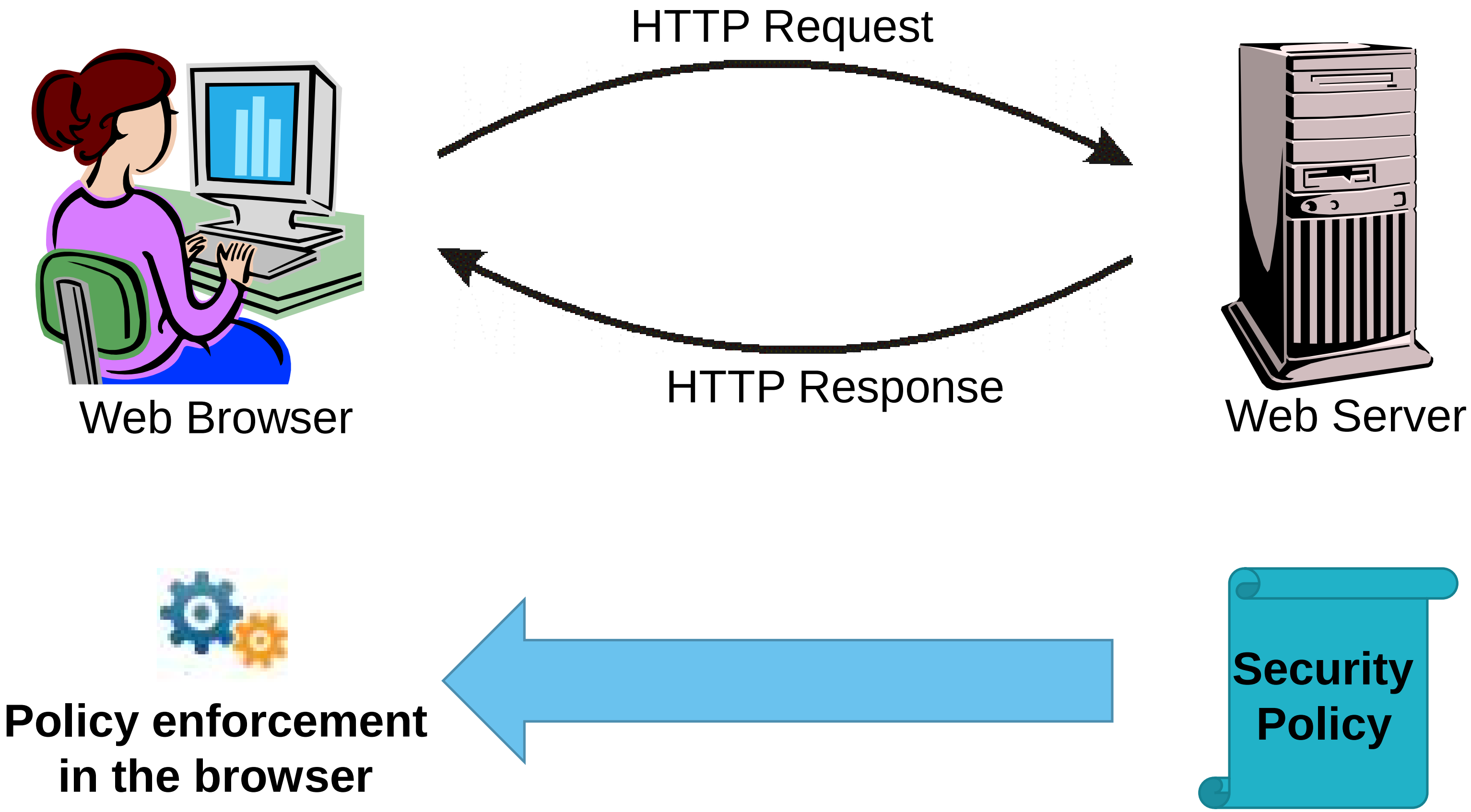
A7 – Insecure Cryptographic Storage

A8 – Failure to Restrict URL Access

A9 – Insufficient Transport Layer Protection

A10 – Unvalidated Redirects and Forwards (NEW)

Recent security technology on the web



HTTP Request

Web Browser

HTTP Response

Web Server

Policy enforcement in the browser

Security Policy

Overview

- Introduction
- Securing browser-server communication
- Mitigating script injection attacks
- Framing content securely
- Example security architecture: Combining CSP & Sandbox
- Wrap-up

Introduction

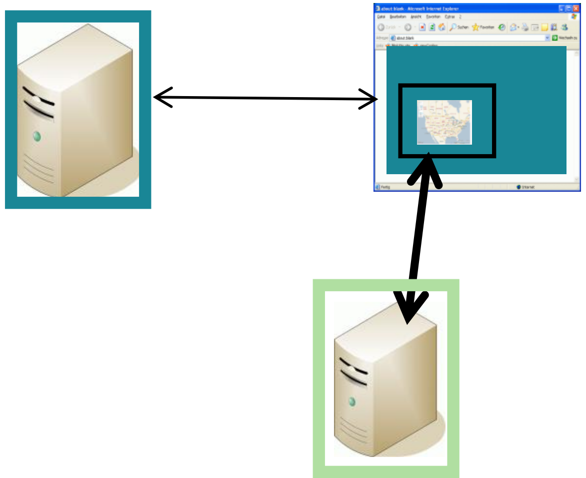
Overview

- Basic security policy for the web:
  - Same-Origin Policy
- What does it mean for scripts running on your page?
- What does it mean for frames included in your page?

Two basic composition techniques

Script inclusion

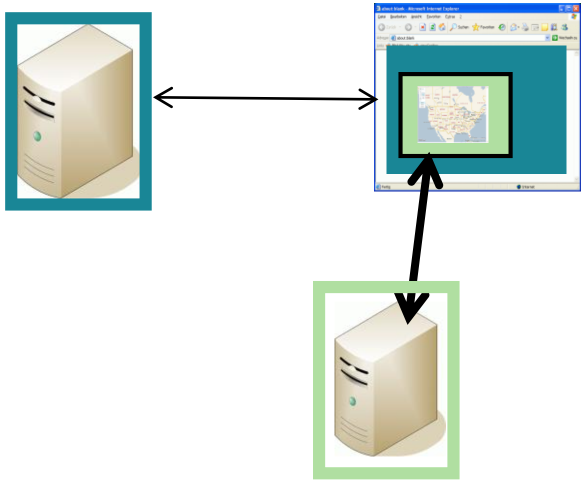
```
<html><body>
…
<script src=“http://3rdparty.com/script.js”></script>
…
</body></html>
```



3rd party

Iframe integration

```
<html><body>
…
<iframe src=“http://3rdparty.com/frame.html”></iframe>
…
</body></html>
```



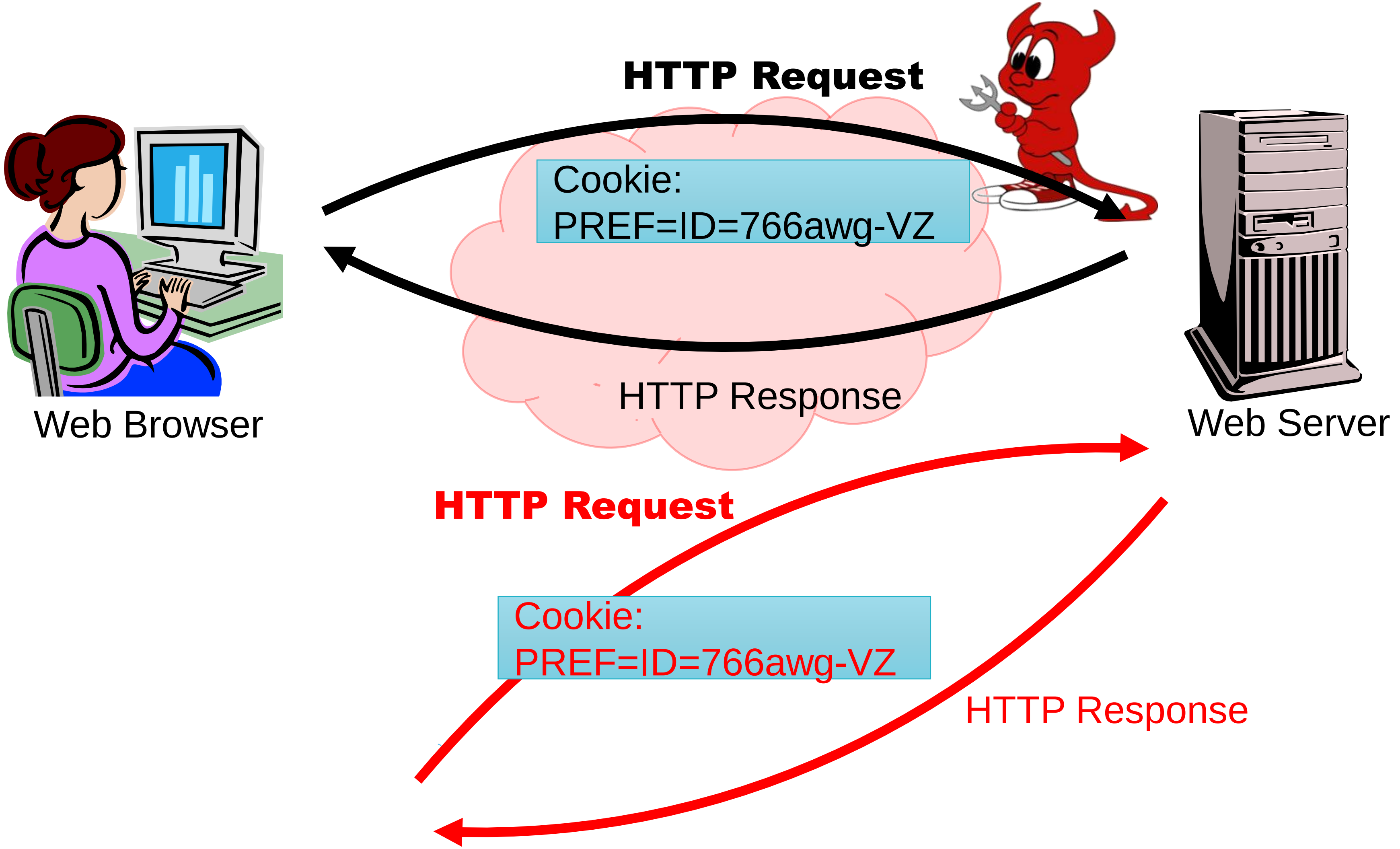
3rd party

Securing browser-server communication

Overview

- Attacks:
  - Session hijacking
  - SSL Stripping
- Countermeasures:
  - Use of SSL/TLS
  - Secure flag for session cookies
  - HSTS header
  - Public Key Pinning

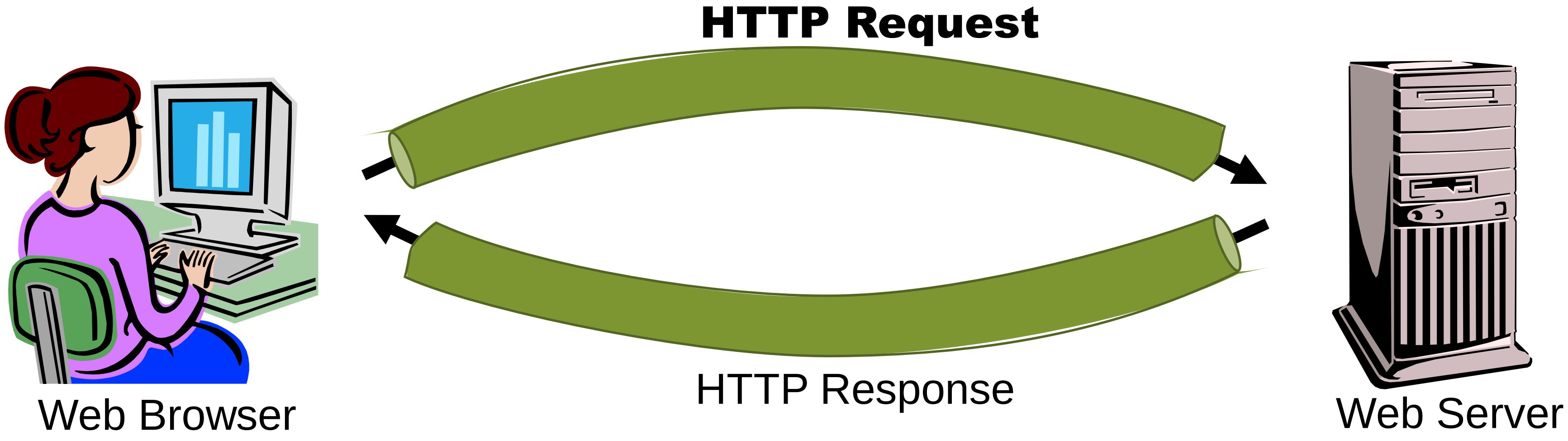
Network attacks: Session hijacking



Cookie:
PREF=ID=766awg-VZ

Cookie:
PREF=ID=766awg-VZ

HTTPS to the rescue...



Problem cured?

- TLS usage statistics:
  - 0.78% of active domains use TLS (with valid SSL certificate)
  - For Alexa top 1 million: 27.86% use TLS

Internet SSL Survey 2010, Qualys

- Remaining problems:
  - Mixed use of HTTPS/HTTP and session cookies
  - SSL Stripping attacks

Mixed use of HTTPS/HTTP

- Cookies are bound to domains, not origins
- By default, cookies are sent both over HTTPS and HTTP
- Any request to your domain over HTTP leaks the (session) cookies…

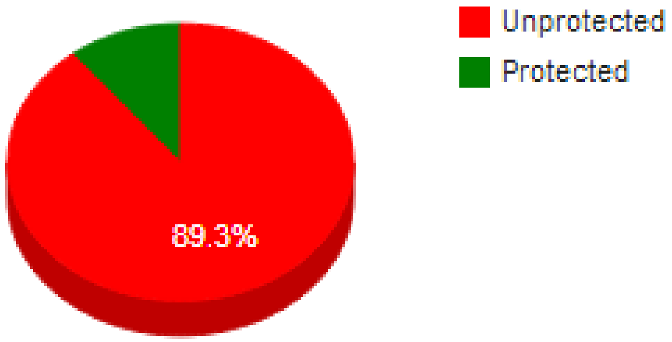
Secure flag for cookies

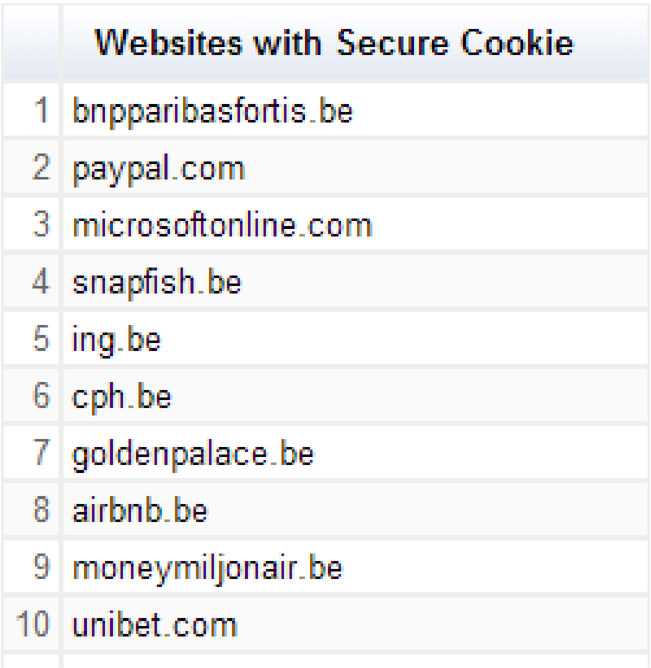
- Issued at cookie creation (HTTP response)
  - Set-Cookie: PREF=766awg-VZ; Domain=yourdomain.com; Secure
- If set, the cookie is only sent over an encrypted channel
- Should be enabled by default for your session cookies!

Secure flag: state-of-practice

- Browser compatibility
  - All recent browsers support the secure flag for cookies

- Usage statistics



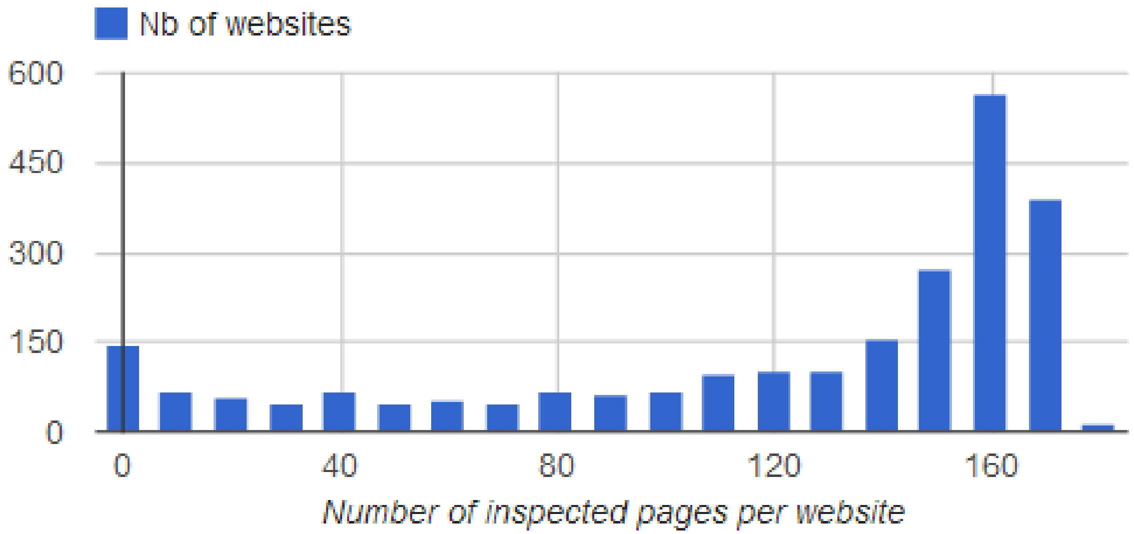


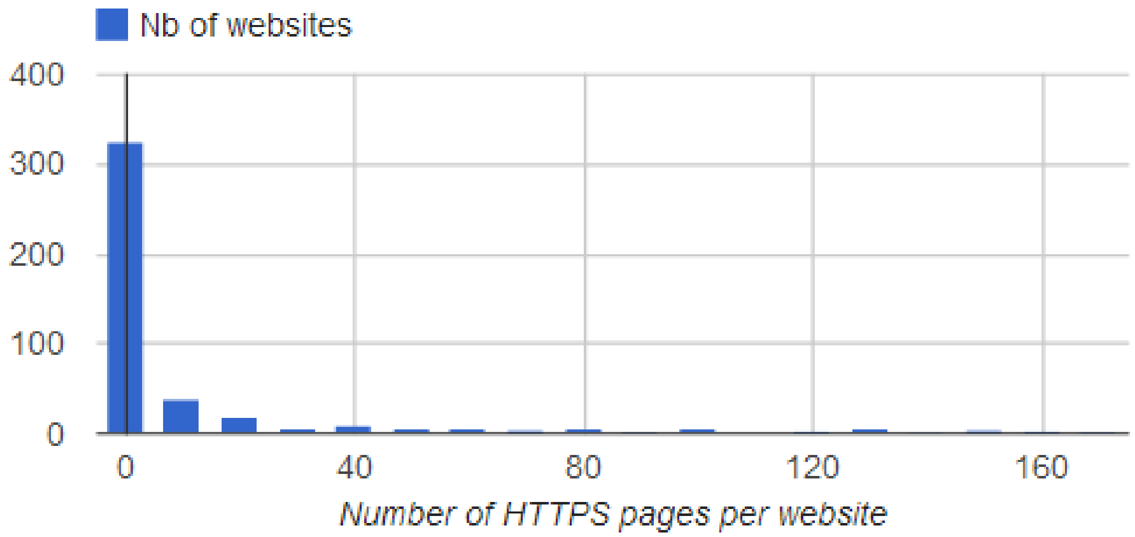
	Websites with Secure Cookie
1	bnpparibasfortis.be
2	paypal.com
3	microsoftonline.com
4	snapfish.be
5	ing.be
6	cph.be
7	goldenpalace.be
8	airbnb.be
9	moneymiljonair.be
10	unibet.com

Own experiment on top 2500 websites, visited from Belgium (Alexa)

Some background on this experiment

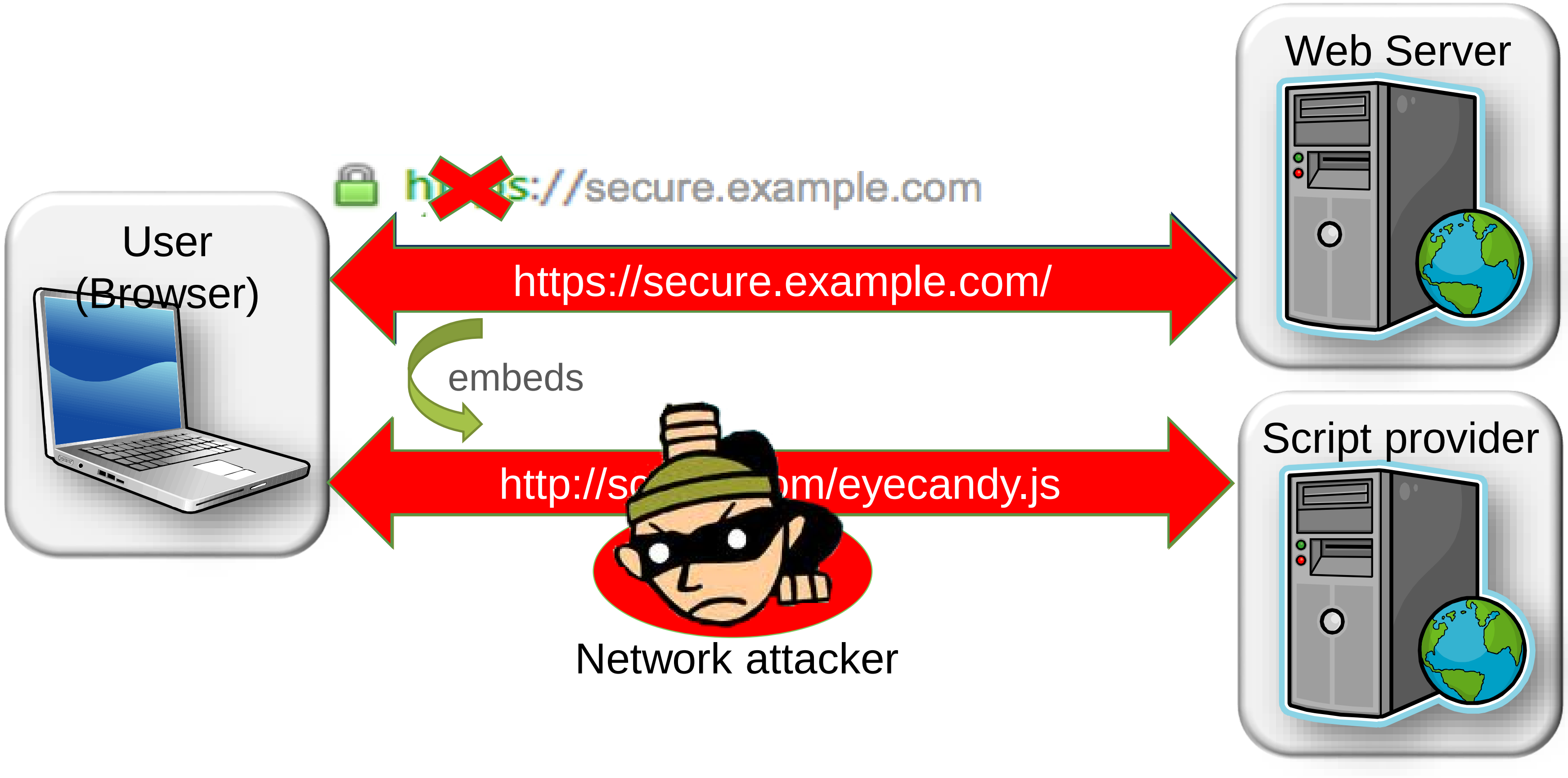
- Number of inspected domains: 2449
- Total number of inspected pages: 302855
- Average number of pages per domains: 123
- 18,25% of domains serve HTTPS pages





Mixed content inclusions:

TLS-enabled sites under attack



Source: *Ping Chen et. al.* A Dangerous Mix: Large-scale analysis of mixed-content websites. *ISC 2013*

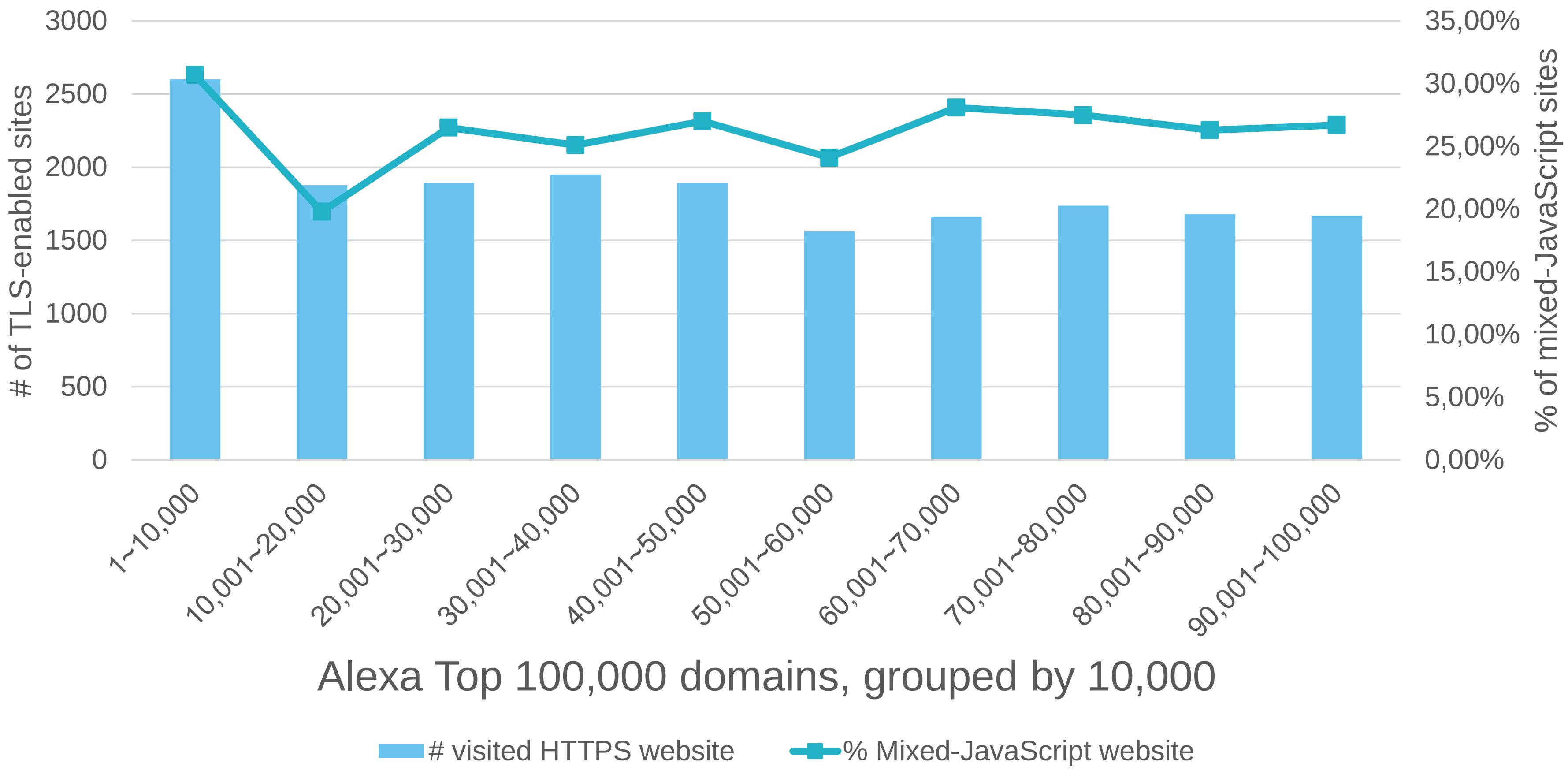
Mixed content inclusions:

Large scale assessment of the state-of-practice

- Alexa Top 100,000 domains
- Crawled over 480,000 pages belonging to the Alexa top 100,000
- Discovered:
  - 18,526 TLS-protected sites
  - 7,980 sites have mixed content (43% of the sites)
  - 150,179 scripts are included over HTTP (26% of the sites)

<u>Source:</u> *Ping Chen et. al.* A Dangerous Mix: Large-scale analysis of mixed-content websites. *ISC 2013*

Distribution of mixed-JavaScript sites
across the top Alexa Top 100,000

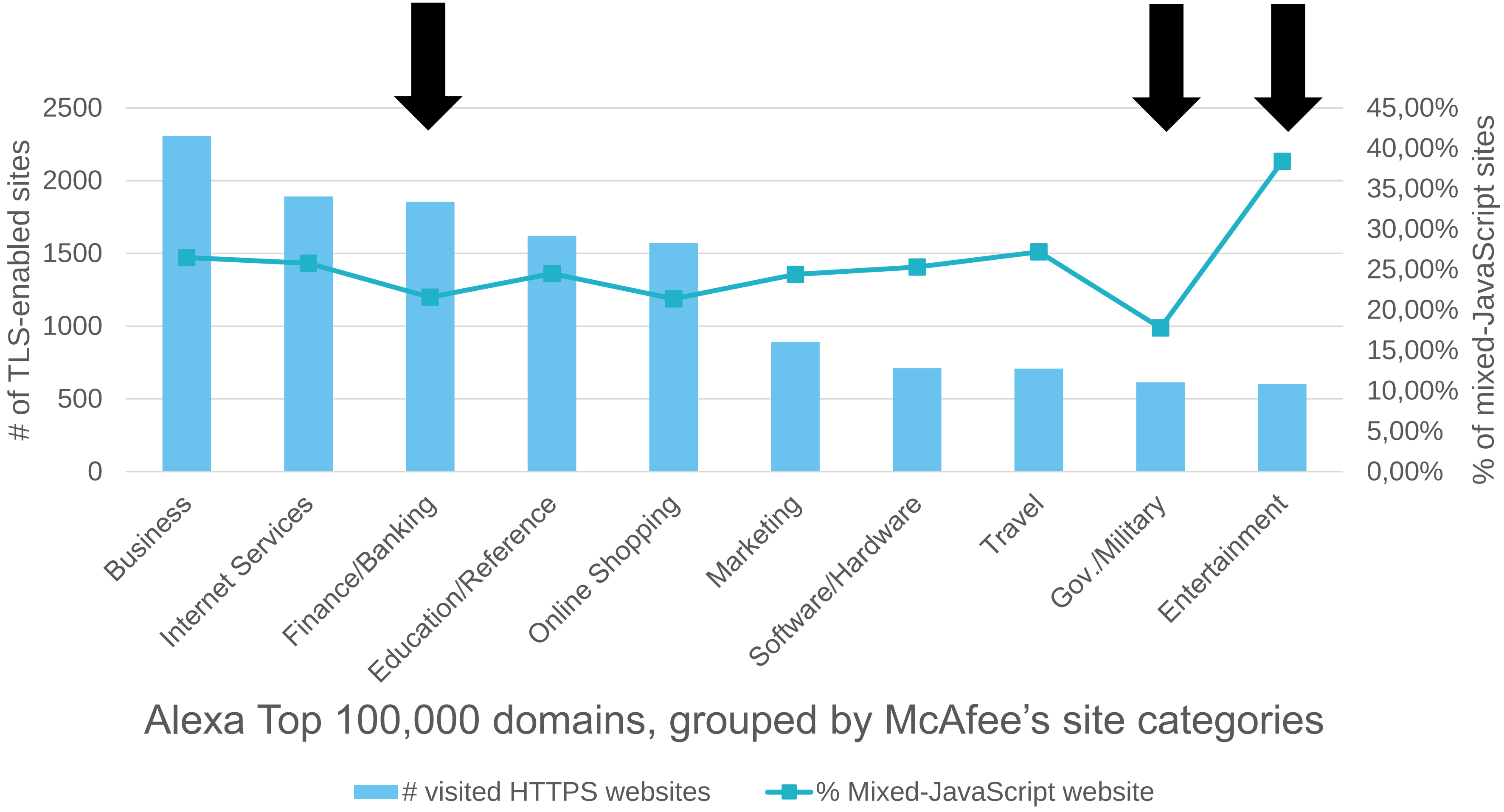


DistriNet iMinds KU LEUVEN

Source: *Ping Chen et. al.* A Dangerous Mix: Large-scale analysis of mixed-content websites. *ISC 2013*

Distribution of mixed-JavaScript sites

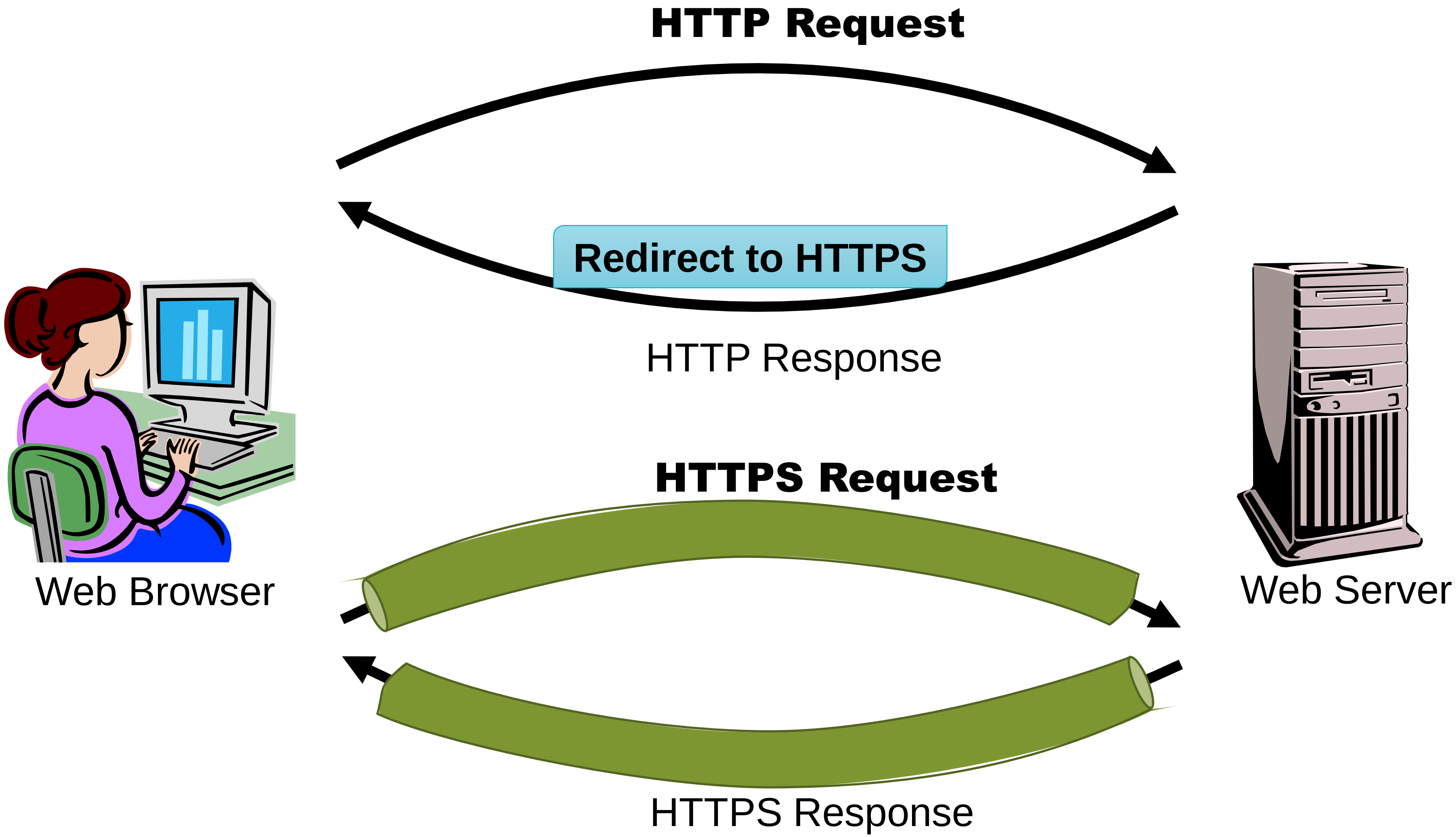
across Top 10 site categories (McAfee’s web database)



DistriNet iMinds KU LEUVEN

Source: *Ping Chen et. al.* A Dangerous Mix: Large-scale analysis of mixed-content websites. *ISC 2013*

HTTP to HTTPS bootstrapping



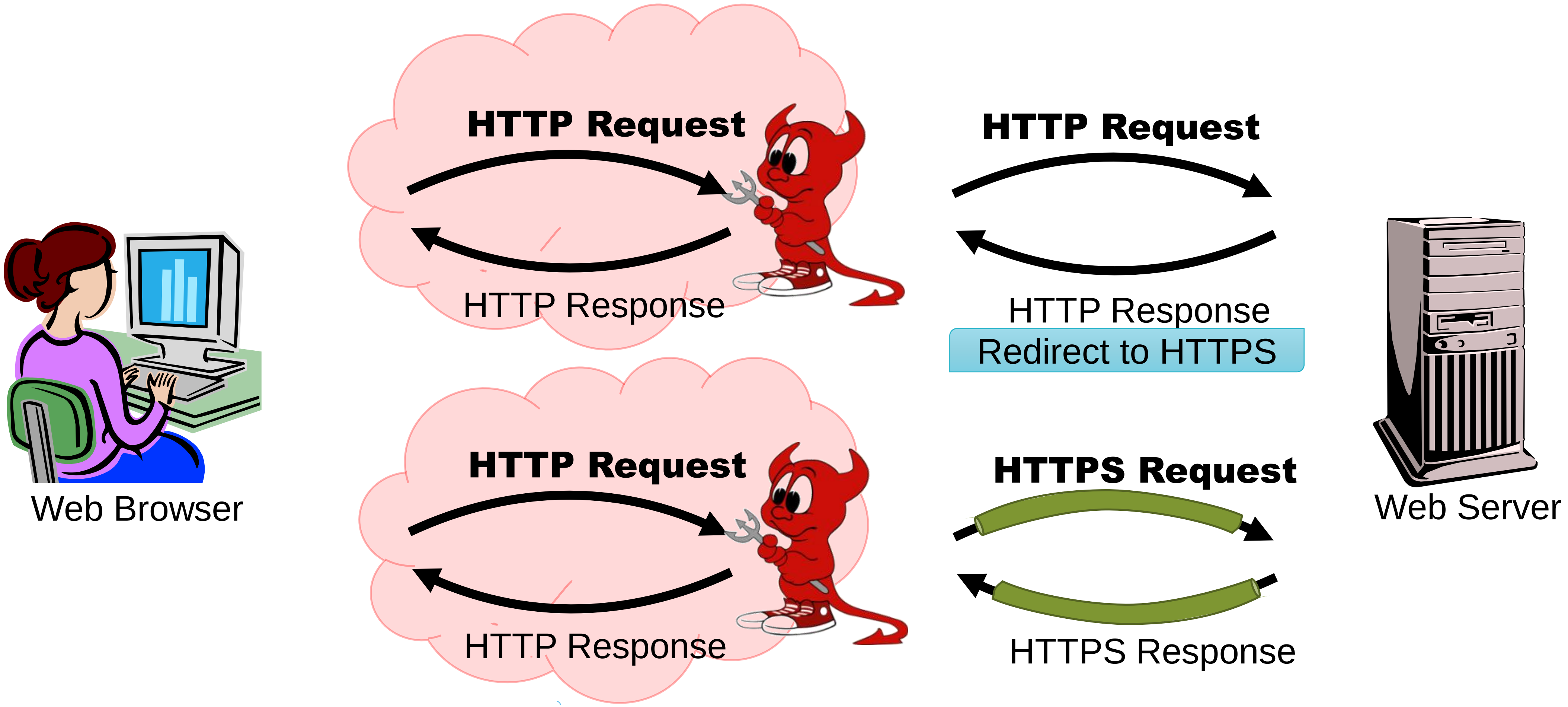
HTTP to HTTPS bootstrapping

- HTTP 301/302 response
  - Location header redirects browser to the resource over HTTPS
  - Location: https://mysite.com/

- Meta refresh
  - Meta-tag in HEAD of HTML page
  - <meta http-equiv="refresh" content="0;URL='https://mysite.com/'">

- Via JavaScript
  - document.location = “https://mysite.com”

Network attacks: SSL Stripping



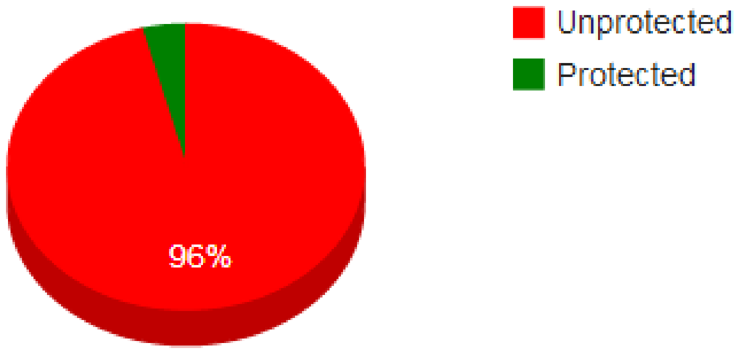
Moxie Marlinspike, BlackHat DC 2009

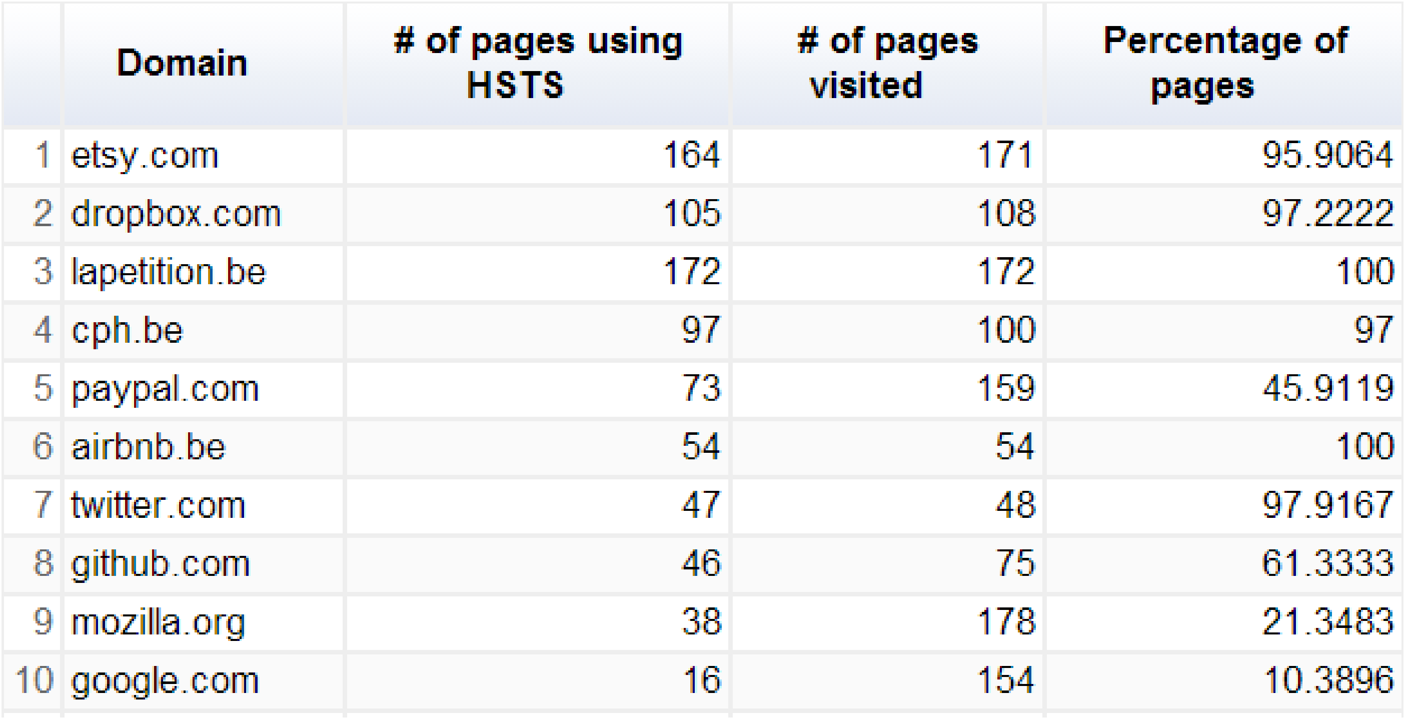
Strict Transport Security (HSTS)

- Issued by the HTTP response header
  - Strict-Transport-Security: max-age=60000
- If set, the browser is instructed to visit this domain only via HTTPS
  - No HTTP traffic to this domain will leave the browser
- Optionally, also protect all subdomains
  - Strict-Transport-Security: max-age=60000; includeSubDomains

HSTS: state-of-practice

- Browser compatibility
  - Chrome 4+, Firefox 4+, Opera 12+, Safari 7+
- Usage statistics





	Domain	# of pages using HSTS	# of pages visited	Percentage of pages
1	etsy.com	164	171	95.9064
2	dropbox.com	105	108	97.2222
3	lapetition.be	172	172	100
4	cph.be	97	100	97
5	paypal.com	73	159	45.9119
6	airbnb.be	54	54	100
7	twitter.com	47	48	97.9167
8	github.com	46	75	61.3333
9	mozilla.org	38	178	21.3483
10	google.com	16	154	10.3896

Own experiment on top 2500 websites, visited from Belgium (Alexa)

But can I trust the CAs ?

- Comodo (March 2011)
  - 9 fraudelent SSL certificates
- Diginotar (July 2011)
  - Wildcard certificates for Google, Yahoo!, Mozilla, WordPress, …
- Breaches at StartSSL (June 2011) and GlobalSign (Sept 2012) reported unsuccessful
- …

Public Key Pinning

- Issued as HTTP response header

  -
    ```
    Public-Key-Pins: max-age=500;
    pin-sha1="4n972HfV354KP560yw4uqe/baXc=";
    pin-sha1="IvGeLsbqzPxdI0b0wuj2xVTdXgc="
    ```
- Freezes the certificate by pushing a fingerprint of (parts of) the certificate chain to the browser
- Currently an IETF Internet-Draft
- Supported in Chrome 18+

Recap: Securing browser-server communication

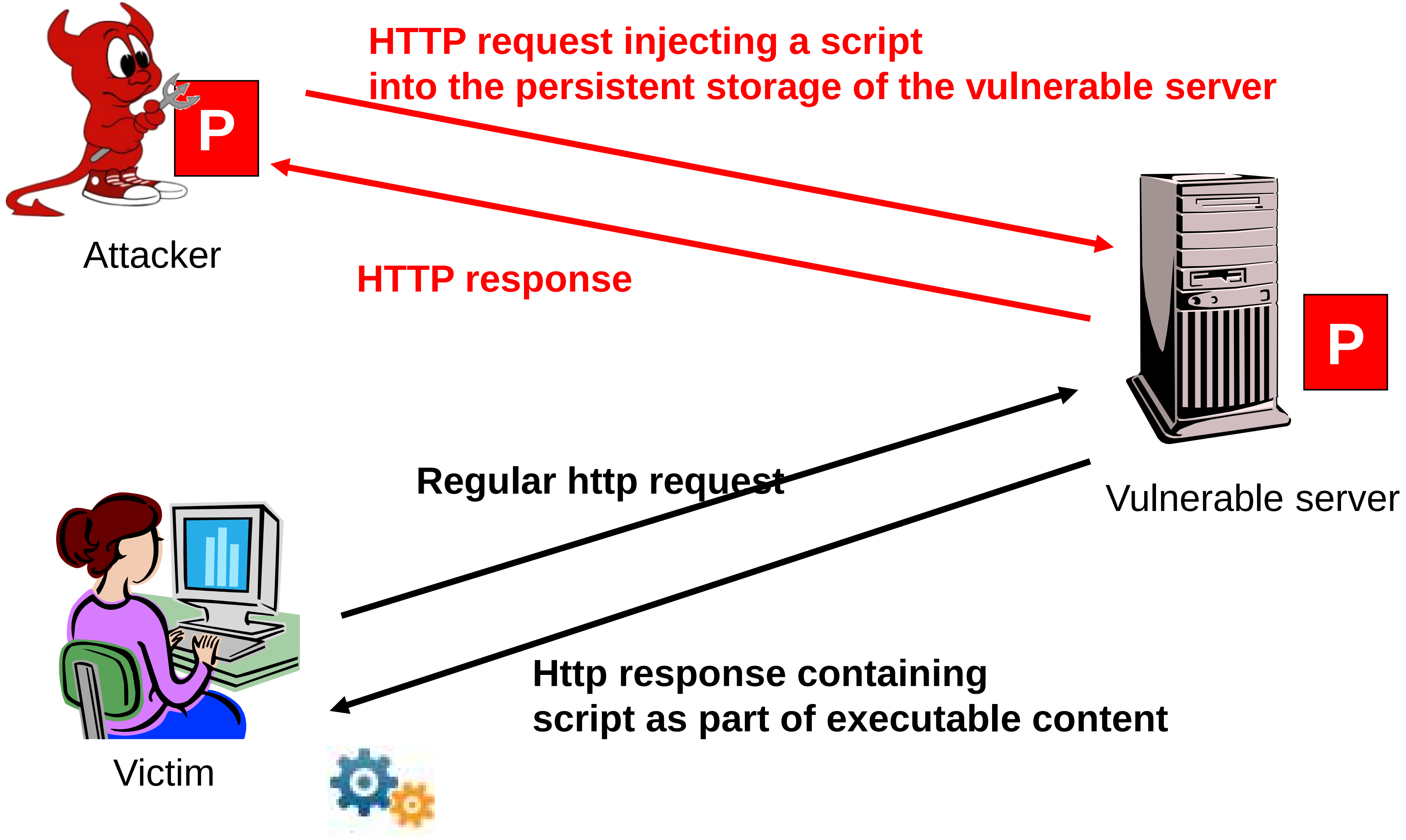
- Use of TLS
- Secure flag for cookies
  - to protect cookies against leaking over HTTP
- HSTS header
  - to force TLS for all future connections
- Public Key Pinning
  - to protect against fraudulent certificates

Mitigating script injection attacks

Overview

- Attack:
  - Cross-Site Scripting (XSS)
- Countermeasures:
  - HttpOnly flag for session cookies
  - X-XSS-Protection header
  - Content Security Policy (CSP)

Example: Stored or persistent XSS



P

Attacker

HTTP request injecting a script
into the persistent storage of the vulnerable server

HTTP response

P

Vulnerable server

Regular http request

Victim

Http response containing
script as part of executable content

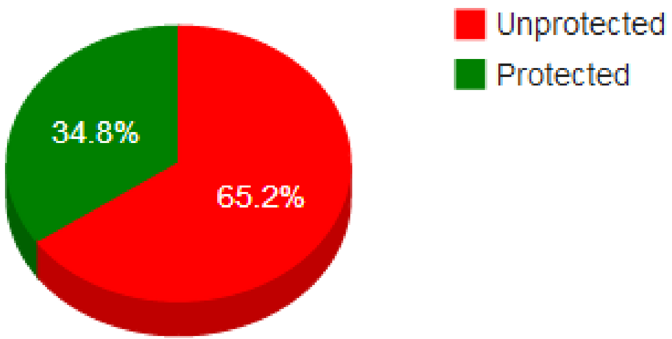
HttpOnly flag for cookies

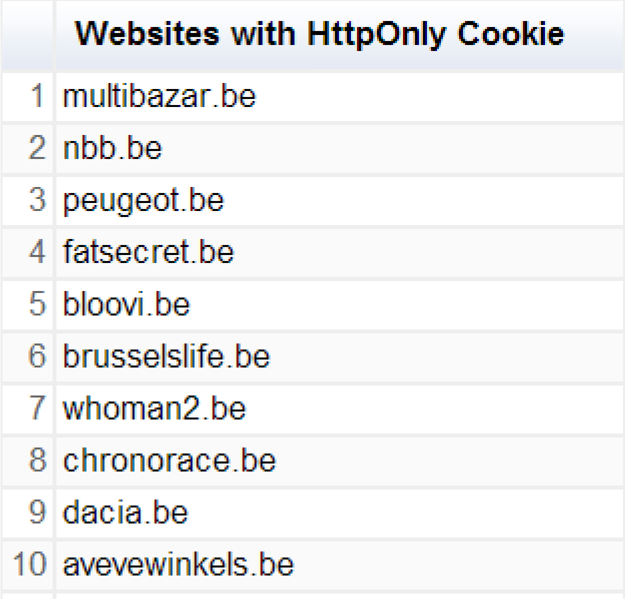
- Issued at cookie creation (HTTP response)
  - Set-Cookie: PREF=766awg-VZ; Domain=yourdomain.com; Secure; HttpOnly
- If set, the cookie is not accessible via DOM
  - JavaScript can not read or write this cookie
- Mitigates XSS impact on session cookies
  - Protects against hijacking and fixation
- Should be enabled by default for your session cookies!

HttpOnly: state-of-practice

- Browser compatibility
  - Support in all browsers
  - Only recently on Android

- Usage statistics





	Websites with HttpOnly Cookie
1	multibazar.be
2	nbb.be
3	peugeot.be
4	fatsecret.be
5	bloovi.be
6	brusselslife.be
7	whoman2.be
8	chronorace.be
9	dacia.be
10	avevewinkels.be

Own experiment on top 2500 websites, visited from Belgium (Alexa)

X-XSS-Protection

- Best-effort protection in the browser against reflected XSS
  - Can be controlled via the X-XSS-Protection header in the HTTP response
  - On by default

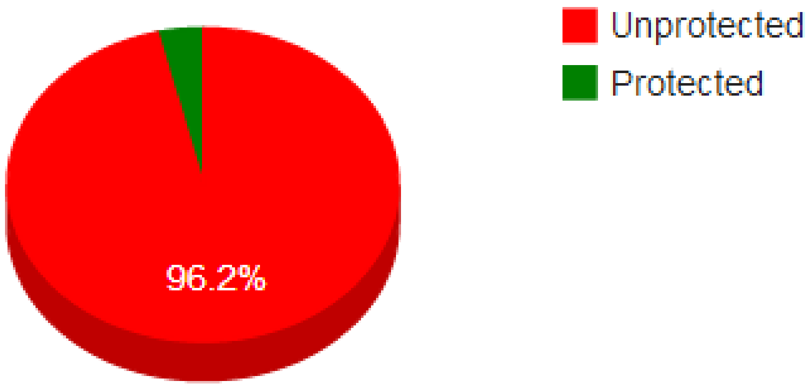
- Completeness of protection
  - Protects only against reflected XSS
  - Multiple bypasses have been reported

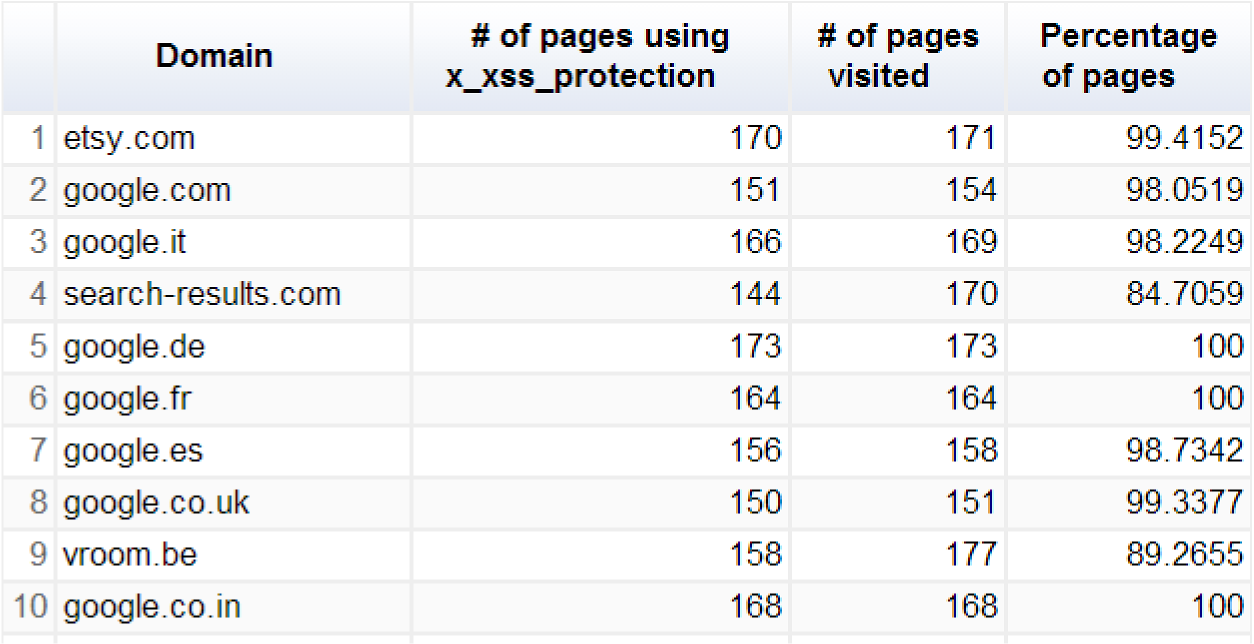
X-XSS-Protection: modes of operation

- Default protection
  - X-XSS-Protection: 1
- Optional opt-out
  - X-XSS-Protection: 0
- Blocking mode
  - X-XSS-Protection: 1; mode=block
  - Prevents the page from rendering

X-XSS-Protection: state-of-practice

- Browser compatibility:
  - Internet Explorer 8+, Chrome and Safari
- Usage statistics





	Domain	# of pages using x_xss_protection	# of pages visited	Percentage of pages
1	etsy.com	170	171	99.4152
2	google.com	151	154	98.0519
3	google.it	166	169	98.2249
4	search-results.com	144	170	84.7059
5	google.de	173	173	100
6	google.fr	164	164	100
7	google.es	156	158	98.7342
8	google.co.uk	150	151	99.3377
9	vroom.be	158	177	89.2655
10	google.co.in	168	168	100

Own experiment on top 2500 websites, visited from Belgium (Alexa)

Content Security Policy (CSP)

- Issued as HTTP response header
  - Content-Security-Policy: script-src 'self'; object-src 'none'
- Specifies which resources are allowed to be loaded as part of your page
- Extremely promising as an additional layer of defense against script injection

CSP set of directives

- There are a whole set of directives
  - Here we discuss CSP v1.0
- default-src
  - Takes a sourcelist as value
  - Default for all resources, unless overridden by specific directives
  - Only allowed resources are loaded

CSP source lists

- Space delimited list of sources
  - ‘self’
  - ‘none’
  - origin(s)
- Examples
  - https://mydomain.com
  - https://mydomain.com:443
  - http://134.58.40.10
  - https://*.mydomain.com
  - https:
  - *://mydomain.com

CSP set of directives (2)

- script-src
  - From which sources, scripts are allowed to be included
- object-src
  - Flash and other plugins
- style-src
  - stylesheets
- img-src
  - images
- media-src
  - sources of video and audio

CSP set of directives (3)

- frame-src
  - list of origins allowed to be embedded as frames
- font-src
  - web fonts
- connect-src
  - To which origins can you connect (e.g. XHR, websockets)
- sandbox
  - Optional
  - Trigger sandboxing attribute of included iframes

CSP requires sites to “behave”

- Inline scripts and CSS is not allowed
  - All scripts need to be externalized in dedicated JS files
  - All style directives need to be externalized in dedicated style files
  - Clean code separation
- The use of *eval* is not allowed
  - To prevent unsafe string (e.g. user input) to be executed

Example: inline scripts

page.html

```
<script>
  function runMyScript() {
    alert('My alert');
  }
</script>

<a href="#" onClick="runMyScript();">
This link shows an alert!</a>
```

Example: externalized scripts

page.html

```
<script src="myscript.js"></script>
<a href="#" id="myLink">This link shows an alert!</a>
```

myscript.js

```
function runMyScript() {
  alert('My alert');
}
document.addEventListener('DOMContentReady',
function () {
  document.getElementById('myLink')
          .addEventListener('click', runMyScript);
});
```

Insecure relaxations, but be careful!

- To temporary allow inline scripts
  - Content-Security-Policy: script-src 'self' 'unsafe-inline'
- To temporary allow eval
  - Content-Security-Policy: script-src 'self' 'unsafe-inline' 'unsafe-eval'
- To temporary allow inline style directives
  - Content-Security-Policy: style-src 'self' 'unsafe-inline'

Be careful!

CSP reporting feature

- CSP reports violations back to the server owner
  - server owner gets insides in actual attacks
    - i.e. violations against the supplied policy
  - allows to further fine-tune the CSP policy
    - e.g. if the policy is too restrictive

- report-uri directive
  - report-uri /my-csp-reporting-handler
  - URI to which the violation report will be posted

Example violation report

```
Content-Security-Policy: script-src 'self' https://apis.google.com;
report-uri http://example.org/my_amazing_csp_report_parser
```

CSP violation report

```
{
  "csp-report": {
    "document-uri": "http://example.org/page.html",
    "referrer": "http://evil.example.com/",
    "blocked-uri": "http://evil.example.com/evil.js",
    "violated-directive": "script-src 'self' https://apis.google.com",
    "original-policy": "script-src 'self' https://apis.google.com; report-uri http://example.org/my_amazing_csp_report_parser"
  }
}
```

Based on “HTML5Rocks: An introduction to Content Security Policy” (Mike West)

CSP Reporting: one step further

- Apart from reporting violations via the report-uri directive

- CSP can also run in report only mode
  - Content-Security-Policy-Report-Only: default-src: 'none'; script-src 'self'; report-uri /my-csp-reporting-handler
  - Violation are reported
  - Policies are not enforced

Some CSP examples

- Examples:
  - Mybank.net lockdown
  - SSL only
  - Social media integration
  - Facebook snapshot

Based on “HTML5Rocks: An introduction to Content Security Policy” (Mike West)

Example: mybank.net lockdown

- Scripts, images, stylesheets
  - from a CDN at https://cdn.mybank.net
- XHR requests
  - Interaction with the mybank APIs at https://api.mybank.com
- Iframes
  - From the website itself
- No flash, java, ….

```
Content-Security-Policy: default-src 'none';
script-src https://cdn.mybank.net;
style-src https://cdn.mybank.net;
img-src https://cdn.mybank.net;
connect-src https://api.mybank.com;
frame-src 'self'
```

Based on “HTML5Rocks: An introduction to Content Security Policy” (Mike West)

Example: SSL only

- Can we ensure to only include HTTPS content in our website?

```
Content-Security-Policy: default-src https: ;
script-src https: 'unsafe-inline';
style-src https: 'unsafe-inline'
```

- Obviously, this should only be the first step, not the final one!

Based on “HTML5Rocks: An introduction to Content Security Policy” (Mike West)

Example: social media integration

- Google +1 button
  - Script from https://apis.google.com
  - Iframe from https://plusone.google.com

- Facebook
  - Iframe from https://facebook.com

- Twitter tweet button
  - Script from https://platform.twitter.com
  - Iframe from https://platform.twitter.com

```
Content-Security-Policy: script-src https://apis.google.com
https://platform.twitter.com;
frame-src https://plusone.google.com https://facebook.com
https://platform.twitter.com
```

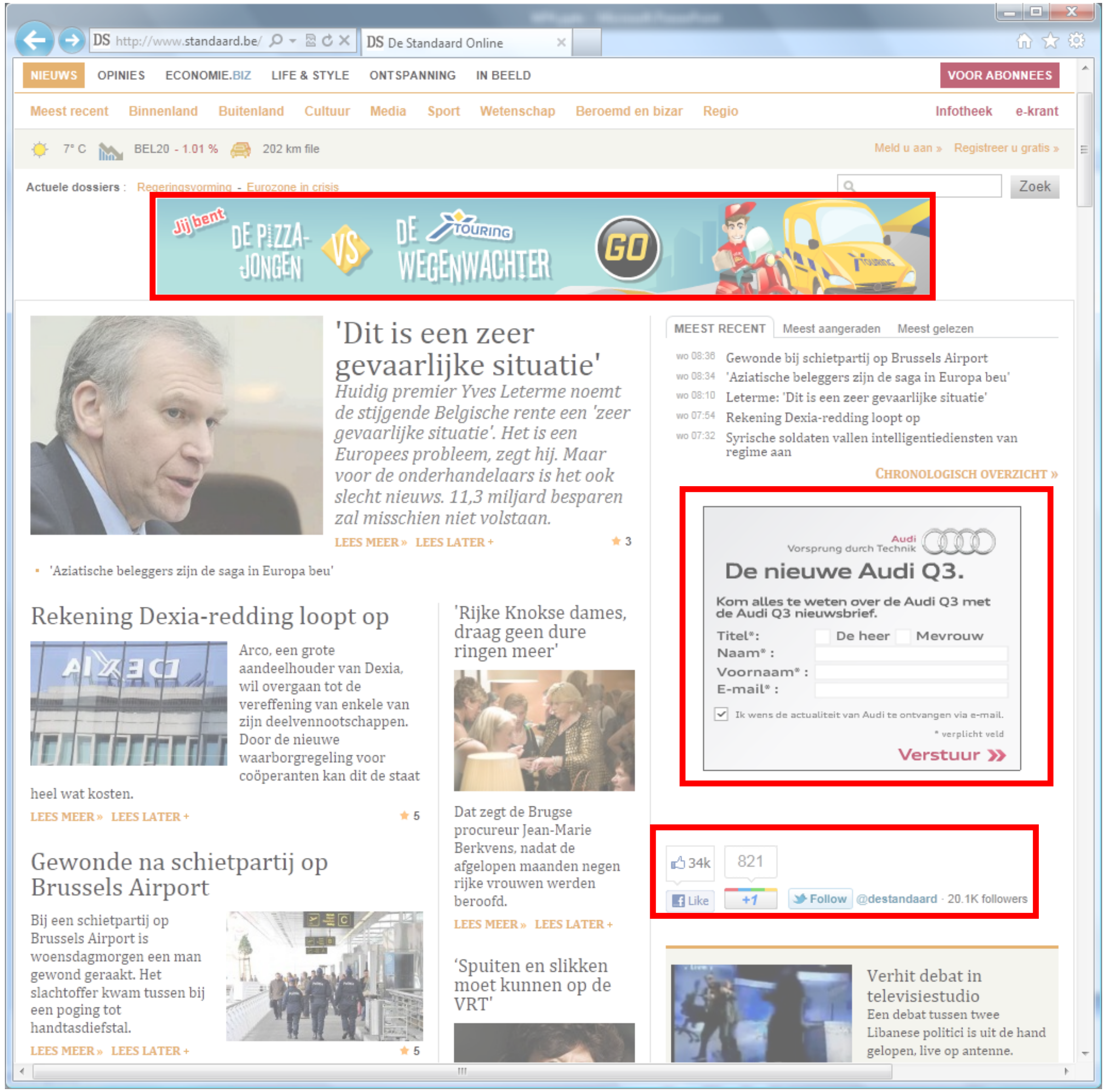
Based on “HTML5Rocks: An introduction to Content Security Policy” (Mike West)

Example: Facebook snapshot

```
X-WebKit-CSP: default-src *;
script-src https://*.facebook.com http://*.facebook.com
https://*.fbcdn.net http://*.fbcdn.net *.facebook.net *.google-
analytics.com *.virtualearth.net *.google.com *.spotilocal.com:*
chrome-extension://lifbcibllhkdhoafpjfnlhfpfgnpldfl 'unsafe-inline'
'unsafe-eval' https://*.akamaihd.net http://*.akamaihd.net;style-
src * 'unsafe-inline';
connect-src https://*.facebook.com http://*.facebook.com
https://*.fbcdn.net http://*.fbcdn.net *.facebook.net
*.spotilocal.com:* https://*.akamaihd.net ws://*.facebook.com:*
http://*.akamaihd.net;
```

Third-party JavaScript is everywhere

- Advertisements
  - Adhese ad network
- Social web
  - Facebook Connect
  - Google+
  - Twitter
  - Feedsburner
- Tracking
  - Scorecardresearch
- Web Analytics
  - Yahoo! Web Analytics
  - Google Analytics
- …



'Dit is een zeer gevaarlijke situatie'

Huidig premier Yves Leterme noemt de stijgende Belgische rente een 'zeer gevaarlijke situatie'. Het is een Europees probleem, zegt hij. Maar voor de onderhandelaars is het ook slecht nieuws. 11,3 miljard besparen zal misschien niet volstaan.

LEES MEER » LEES LATER + ★ 3

- 'Aziatische beleggers zijn de saga in Europa beu'

Rekening Dexia-redding loopt op

Arco, een grote aandeelhouder van Dexia, wil overgaan tot de vereffening van enkele van zijn deelvennootschappen. Door de nieuwe waarborgregeling voor coöperanten kan dit de staat heel wat kosten.

LEES MEER » LEES LATER + ★ 5

Gewonde na schietpartij op Brussels Airport

Bij een schietpartij op Brussels Airport is woensdagmorgen een man gewond geraakt. Het slachtoffer kwam tussen bij een poging tot handtasdiefstal.

LEES MEER » LEES LATER + ★ 5

'Rijke Knokse dames, draag geen dure ringen meer'

Dat zegt de Brugse procureur Jean-Marie Berkvens, nadat de afgelopen maanden negen rijke vrouwen werden beroofd.

LEES MEER » LEES LATER +

'Spuiten en slikken moet kunnen op de VRT'

MEEST RECENT

wo 08:36 Gewonde bij schietpartij op Brussels Airport
wo 08:34 'Aziatische beleggers zijn de saga in Europa beu'
wo 08:10 Leterme: 'Dit is een zeer gevaarlijke situatie'
wo 07:54 Rekening Dexia-redding loopt op
wo 07:32 Syrische soldaten vallen intelligentiediensten van regime aan

Audi
Vorsprung durch Technik

De nieuwe Audi Q3.

Kom alles te weten over de Audi Q3 met de Audi Q3 nieuwsbrief.

Titel*: □ De heer □ Mevrouw
Naam* :
Voornaam* :
E-mail* :
☑ Ik wens de actualiteit van Audi te ontvangen via e-mail.
* verplicht veld
Verstuur »

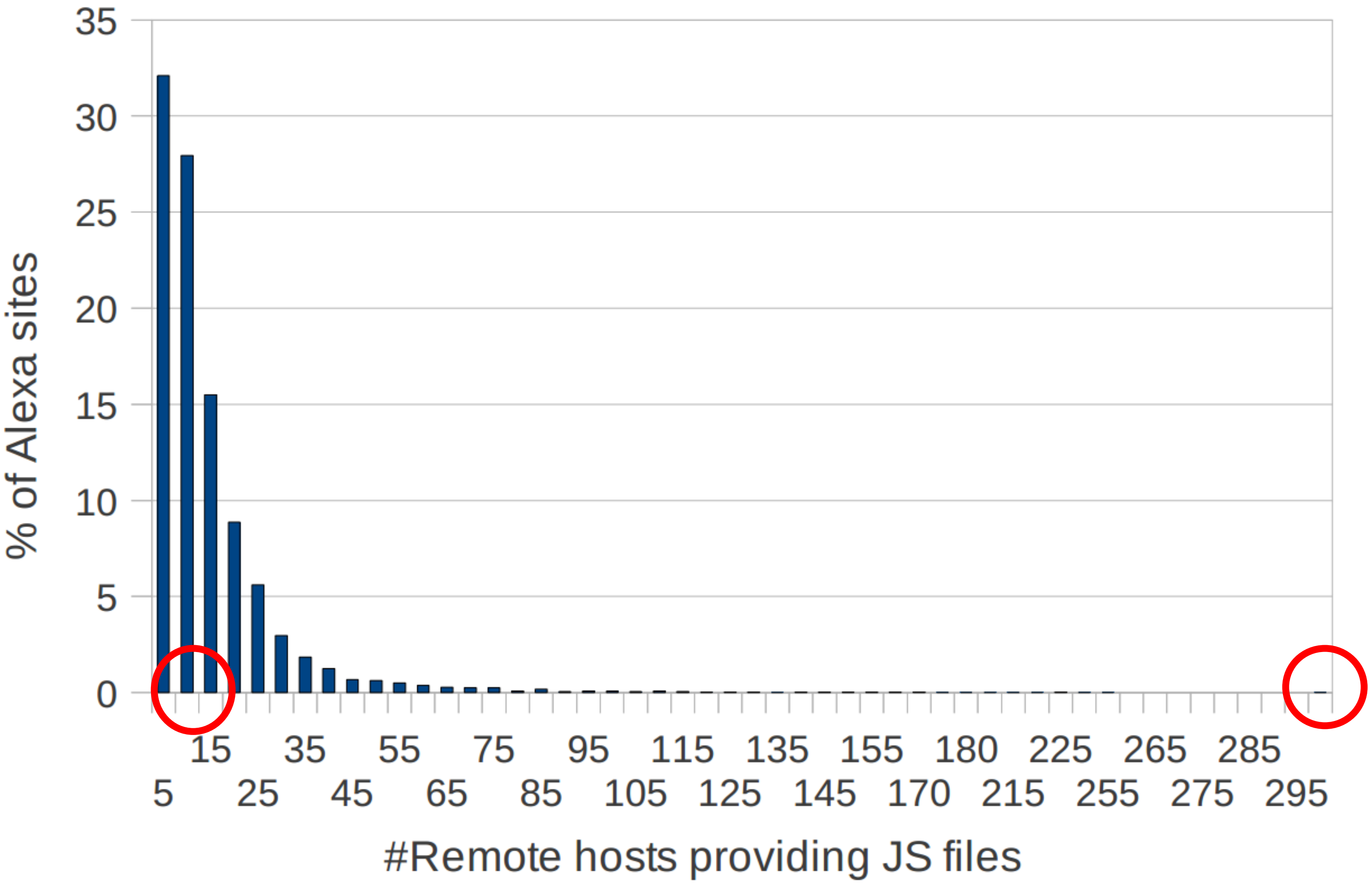
Verhit debat in televisiestudio

Een debat tussen twee Libanese politici is uit de hand gelopen, live op antenne.

Spongecell
Jhula
kiko
Trumba
eskobo
mayomi
Pageflakes
vimeo
skobee
shadows
groovee
YouTube
Zimbra
LookSmart FURL | Your Personal Web File
smugmug
newsgator
Blogniscient
TinFinger
shutterfly
mefeedia
PodDater
Feedster
favoor
Planzo.com
ZAZZLE
Tailrank
TagWorld
nuvvo
dogear
yakalike
grouper
ODDPOST
QOOP
iNods
Lulu
rbloc.com
BR
blish
flagr
FireAnt
simplyhired
veoh
theadcloud
pixagogo
gather
Agatra
browsr
oyogi
cafepress
Renkoo
standpoint
meebo
EXTRA TASTY!
last.fm
Jotspot
Frappr!
jeteye
dabble db
YEDDA
Writeboard
SHOUTWIRE
iKarma
AirSet
tech.memeorandum
CalendarHub
Suprglu
PIECING YOUR WEB TOGETHER
pando
zigtag
Findory
backfence
clipmarks
wayfaring
gOFFICE
AllPeers
orb
Rallypoint
Zoozio
blogbeat
Ziggs
zooto
vSocial
Boltfolio
wink
riya
Audible Wordcast
Opinity
Because your reputation matters!
reddit
measure map
gumshoo
bluepulse
imvu BETA
STREAMLOAD
Ta-da Lists
FeedSky
jellyBam.INC
FeedTier
phanfare
Fruitcast
PubSub
nativetext
CONGOO BETA
PODZINGER
RSS MAD
dPolls
flickr
Ning
Ookles
Strongspace
zoominfo
CASTPOST
WIKIPEDIA
yubnub
AC

Number of remote script providers per site

- 88.45% includes at least 1 remote JavaScript library
- 2 out of 3 sites relies on 5 or more script providers
- 1 site includes up to 295 remote script providers



Source: *Nick Nikiforakis et. al. You are what you include: Large-scale evaluation of remote JavaScript inclusions. CCS 2012*

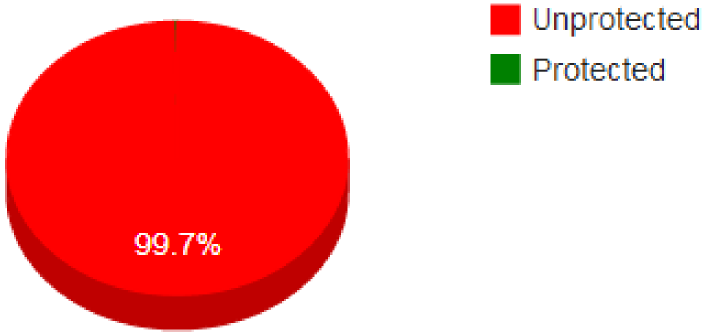
Most popular JavaScript libraries and APIs

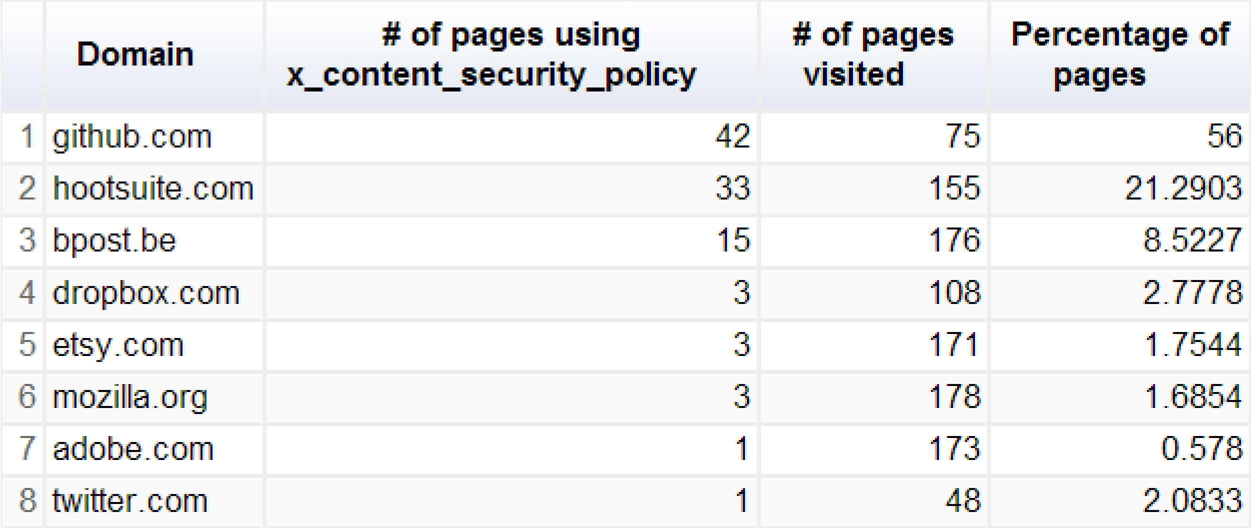
Offered service	JavaScript file	% Alexa Top 10K
Web analytics	www.google-analytics.com/ga.js	68,37%
Dynamic Ads	pagead2.googlesyndication.com/pagead/show_ads.js	23,87%
Web analytics	www.google-analytics.com/urchin.js	17,32%
Social Networking	connect.facebook.net/en_us/all.js	16,82%
Social Networking	platform.twitter.com/widgets.js	13,87%
Social Networking & Web analytics	s7.addthis.com/js/250/addthis_widget.js	12,68%
Web analytics & Tracking	edge.quantserve.com/quant.js	11,98%
Market Research	b.scorecardresearch.com/beacon.js	10,45%
Google Helper Functions	www.google.com/jsapi	10,14%
Web analytics	ssl.google-analytics.com/ga.js	10,12%

Source: Nick Nikiforakis et. al. You are what you include:
Large-scale evaluation of remote JavaScript inclusions. CCS 2012

CSP: state-of-practice

- Browser compatibility:
  - Firefox 4, Chrome 14+, Safari 5+, Opera 15+, Internet Explorer 10+
  - Older header names: X-WebKit-CSP, X-Content-Security-Policy
- Usage statistics





	Domain	# of pages using x_content_security_policy	# of pages visited	Percentage of pages
1	github.com	42	75	56
2	hootsuite.com	33	155	21.2903
3	bpost.be	15	176	8.5227
4	dropbox.com	3	108	2.7778
5	etsy.com	3	171	1.7544
6	mozilla.org	3	178	1.6854
7	adobe.com	1	173	0.578
8	twitter.com	1	48	2.0833

Own experiment on top 2500 websites, visited from Belgium (Alexa)

Recap: Mitigating script injection attacks

- HttpOnly flag for session cookies
  - To protect cookies against hijacking and fixation from JavaScript

- X-XSS-Protection header
  - Coarse-grained control over built-in browser protection against reflected XSS

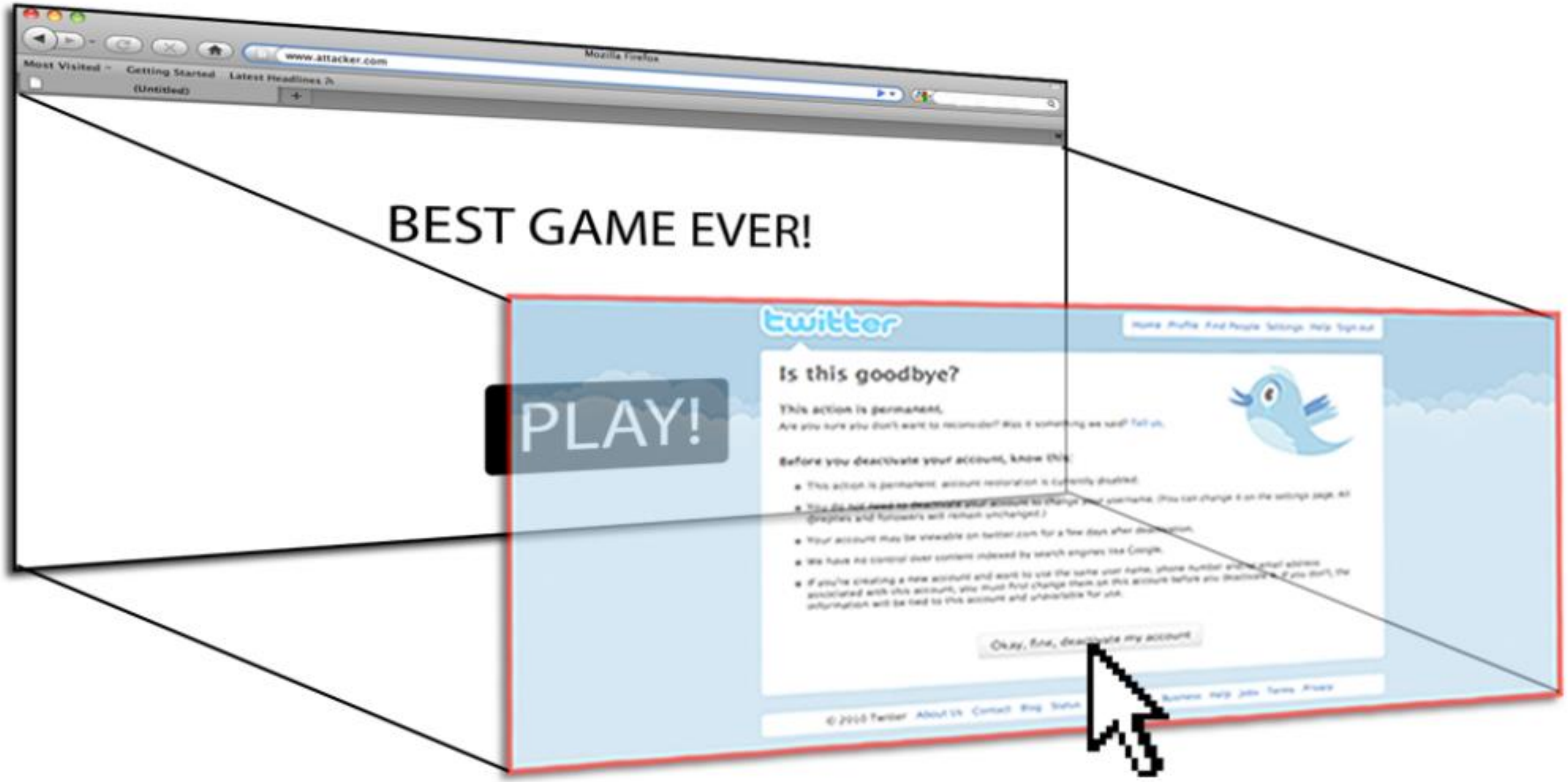
- Content Security Policy (CSP)
  - Domain-level control over resources to be included
  - Most promising infrastructural technique against XSS
  - Interesting reporting-only mode

Framing content securely

Overview

- Attacks:
  - Click-jacking
  - Same domain XSS
- Countermeasures:
  - X-Frame-Options header
  - HTML5 sandbox attribute for iframes

Click-jacking


Mozilla Firefox
www.attacker.com
Most Visited
Getting Started
Latest Headlines
(Untitled)
BEST GAME EVER!
PLAY!
twitter
Home Profile Find People Settings Help Sign out
Okay, fine, deactivate my account
© 2010 Twitter About Us Contact Blog Status
Business Help Jobs Terms Privacy

Is this goodbye?

This action is permanent.
Are you sure you don't want to reconsider? Was it something we said? Tell us.

Before you deactivate your account, know this:

- This action is permanent; account restoration is currently disabled.
- You do not need to deactivate your account to change your username. (You can change it on the settings page. All @replies and followers will remain unchanged.)
- Your account may be viewable on twitter.com for a few days after deactivation.
- We have no control over content indexed by search engines like Google.
- If you're creating a new account and want to use the same user name, phone number and/or email address associated with this account, you must first change them on this account before you deactivate. If you don't, the information will be tied to this account and unavailable for use.

Source: “Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites” (W2SP 2010)

Unsafe countermeasures

- A lot of unsafe ways exist to protect against clickjacking

  -
    ```
    if (top.location != location)
    top.location = self.location;
    ```

  -
    ```
    if (parent.location != self.location)
    parent.location = self.location;
    ```
- Can easily be defeated by
  - Script disabling/sandboxing techniques
  - Frame navigation policies
  - XSS filters in browsers

Source: “Busting Frame Busting: a Study of Clickjacking Vulnerabilities on Popular Sites” (W2SP 2010)

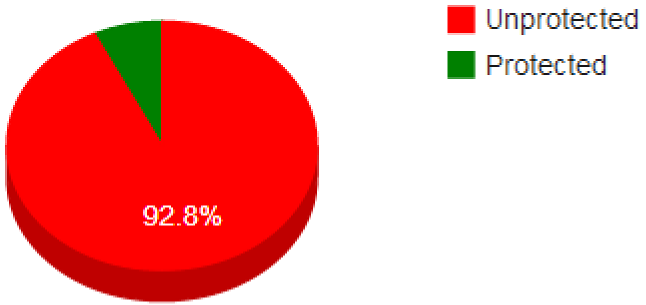
X-Frame-Options

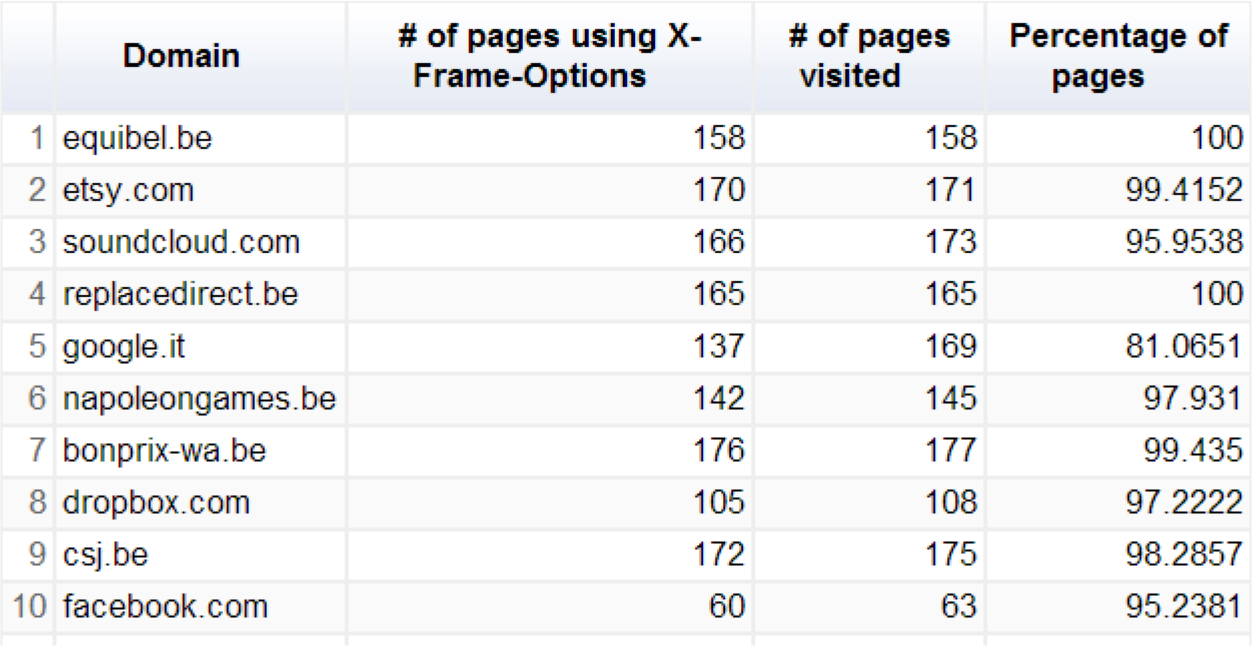
- Issued by the HTTP response header
  - X-Frame-Options: SAMEORIGIN
  - Indicates if and by who the page might be framed

- 3 options:
  - DENY
  - SAMEORIGIN
  - ALLOW-FROM uri

X-Frame-Options

- Browser compatibility:
  - Firefox, Internet Explorer, Opera
  - *Safari, Chrome*
- Usage statistics





	Domain	# of pages using X-Frame-Options	# of pages visited	Percentage of pages
1	equibel.be	158	158	100
2	etsy.com	170	171	99.4152
3	soundcloud.com	166	173	95.9538
4	replacedirect.be	165	165	100
5	google.it	137	169	81.0651
6	napoleongames.be	142	145	97.931
7	bonprix-wa.be	176	177	99.435
8	dropbox.com	105	108	97.2222
9	csj.be	172	175	98.2857
10	facebook.com	60	63	95.2381

Own experiment on top 2500 websites, visited from Belgium (Alexa)

Limitations of framing content in same origin

- Iframe integration provides a good isolation mechanism
  - Each origin runs in its own security context, thanks to the Same-Origin Policy
  - Isolation only holds if outer and inner frame belong to a different origin
- Hard to isolate untrusted content within the same origin

HTML5 sandbox attribute

- Expressed as attribute of the iframe tag
  - <iframe src= "/untrusted-path/index.html" sandbox></iframe>
  - <iframe src="/untrusted-path/index.html" sandbox= "allow-scripts"></iframe>

- Level of Protection
  - Coarse-grained sandboxing
  - ‘SOP but within the same domain’

Default sandbox behavior

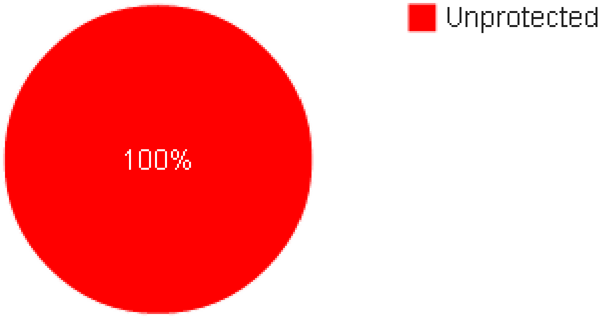
- Plugins are disabled
- Frame runs in a unique origin
- Scripts can not execute
- Form submission is not allowed
- Top-level context can not be navigated
- Popups are blocked
- No access to raw mouse movements data

Sandbox relaxation directives

- Relaxations:
  - allow-forms
  - allow-popups
  - allow-pointer-lock
  - allow-same-origin
  - allow-scripts
  - allow-top-navigation
- Careful!
  - Combining allow-scripts & allow-same-origin voids the sandbox isolation
- Plugins can not be re-enabled

HTML5 sandbox

- Browser compatibility
  - Internet Explorer, Chrome, Safari, Firefox, Opera
- Usage statistics



Own experiment on top 100 websites, visited from Belgium (Alexa)

Recap: Framing content securely

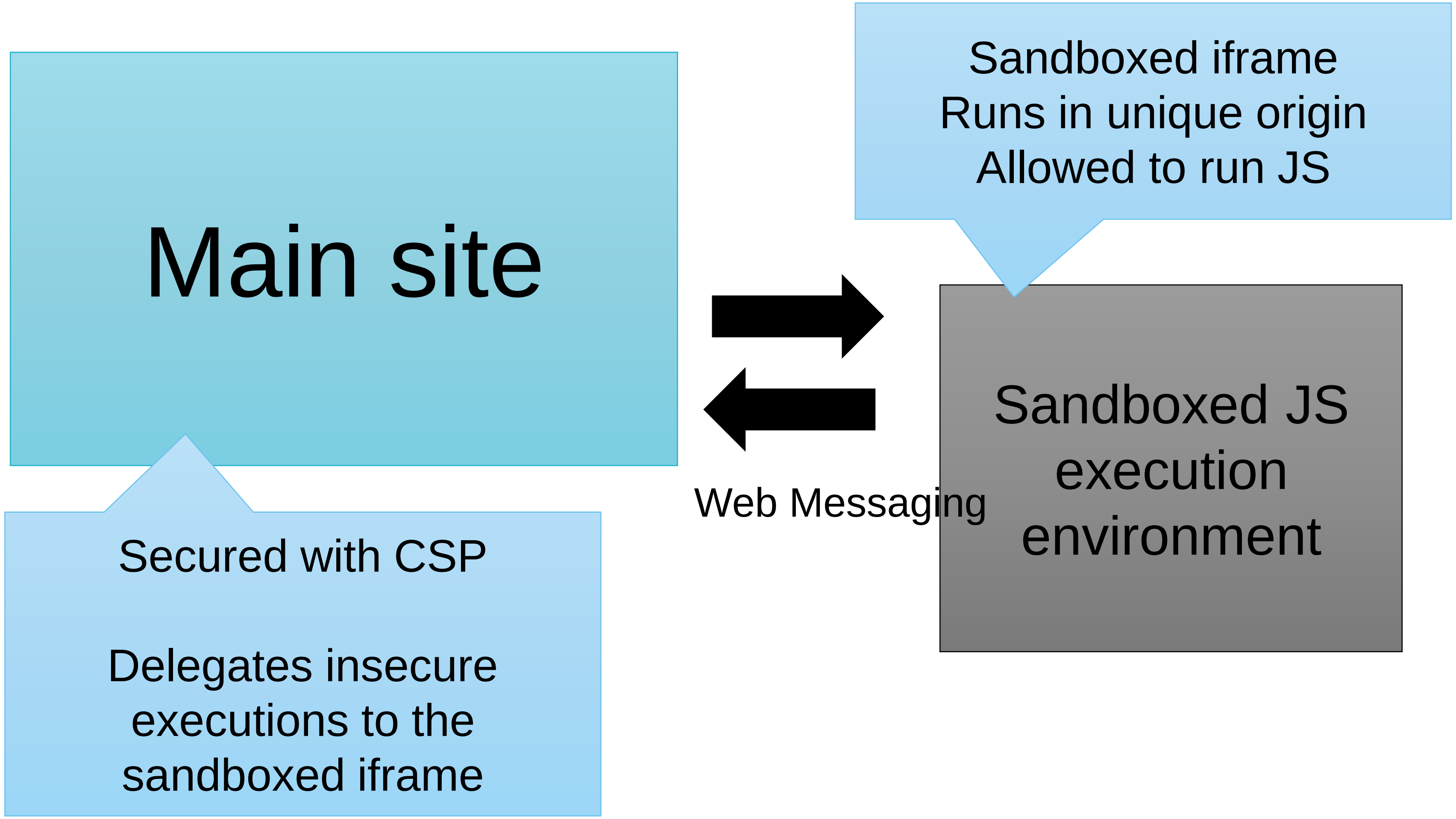
- X-Frame-Options header
  - Robust defense against click-jacking
  - Any state-changing page should be protected
- HTML5 sandbox attribute for iframes
  - Coarse-grained sandboxing of resources and JavaScript
  - Interesting enabler for security architectures

Example security architecture: Combining CSP & Sandbox

CSP & HTML5 sandbox as security enabler

- Combination of CSP and HTML5 sandbox
  - Enabling technologies for drafting a web application security architecture
  - Allows to define whether or not certain functions/scripts are allowed to run in the origin of the site
- Presented by Mike West at Devoxx 2012
  - Used for document rendering in ChromeOS, …

Example of sandboxing unsafe javascript



Main site

Secured with CSP

Delegates insecure executions to the sandboxed iframe

Web Messaging

Sandboxed JS execution environment

Sandboxed iframe
Runs in unique origin
Allowed to run JS

Main page (index.html)

```
Content-Security-Policy: script-src 'self'
```

```
<html><head>
    <script src="main.js"></script>
</head>
<body>
    <a href="#" id="sandboxFrame"/>Click here</a>
    <iframe id="sandboxFrame" sandbox="allow-scripts"
src="sandbox.html">
    </iframe>
    <div ="#content"></div>
</body></html>
```

“Securing the Client-Side: Building safe web applications with HTML5” (Mike West, Devoxx 2012)

Sandboxed frame (sandbox.html)

```
<html><head>
  <script>
        window.EventListener('message', function(event) {
         var command = event.data.command;
              var context = event.data.context;
              var result = callUnsafeFunction(command, context);
              event.source.postMessage({
                     html: result}, event.origin);
              });
  </script>
</head></html>
```

“Securing the Client-Side: Building safe web applications with HTML5” (Mike West, Devoxx 2012)

Main script (main.js)

```
document.querySelector('#click').addEventListener('click',
 function(){
  var iframe = document.querySelector('#sandboxFrame');
       var message = {
        command = 'render';
        context = {thing: 'world‘}};
       iframe.contentWindow.postMessage(message, '*');
});

window.addEventListener('message', function(event){
 //Would be dangerous without the CSP policy!
 var content = document.querySelector('#content');
 content.innerHTML = event.data.html;
});
```

“Securing the Client-Side: Building safe web applications with HTML5” (Mike West, Devoxx 2012)

And what’s next?

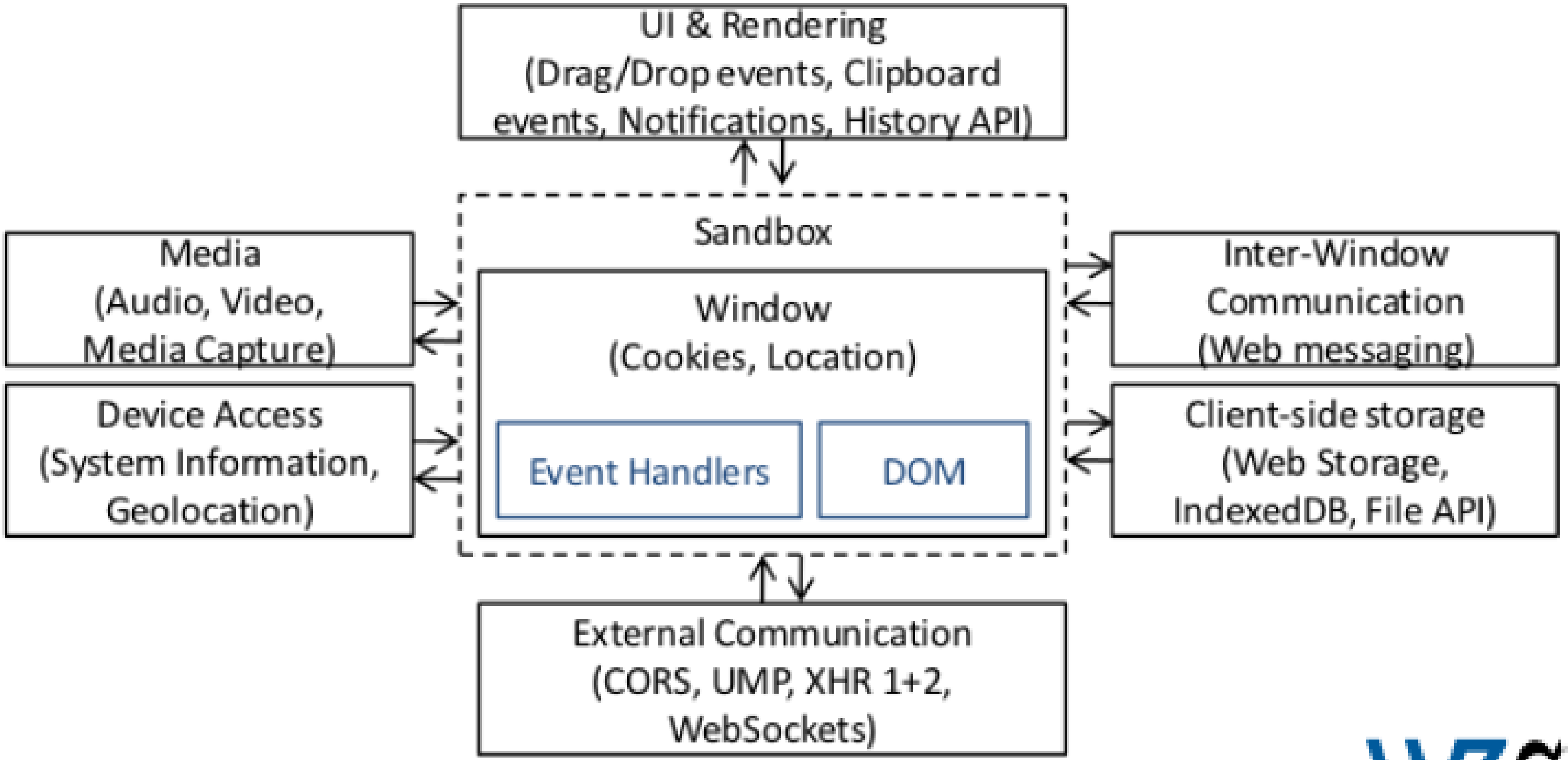
- Seamless integrating unsafe input with the sandbox attribute
  - <iframe **sandbox seamless srcdoc**="<p>Some paragraph</p>"> </iframe>
- seamless attribute
  - Renders visually as part of your site
  - Only for same-origin content
- srcdoc attribute
  - Content as a attribute value instead of a remote page

Enabling cross-domain interactions

And there is a lot more …

- Problem:
  - Sometimes the Same-Origin Policy is too restrictive
- Enabling technologies:
  - Cross Origin Resource Sharing (CORS)
  - Crossdomain.xml
  - Web Messaging (aka postMessage)
  - …

HTML5: security analysis


UI & Rendering
(Drag/Drop events, Clipboard events, Notifications, History API)
Sandbox
Media
(Audio, Video, Media Capture)
Device Access
(System Information, Geolocation)
Window
(Cookies, Location)
Event Handlers
DOM
Inter-Window Communication
(Web messaging)
Client-side storage
(Web Storage, IndexedDB, File API)
External Communication
(CORS, UMP, XHR 1+2, WebSockets)

Analysis of the specifications

A Security Analysis of
Next Generation Web Standards

July | 2011

- A Security Analysis of Next Generation Web Standards
  - Commissioned by European Network and Information Security Agency (ENISA)
  - Performed by iMinds-DistriNet, KU Leuven

- Full report available at ENISA
  - http://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/web-security/a-security-analysis-of-next-generation-web-standards

Analysis results

	Well-defined / Secure	Isolation Properties	Consistency	User Involvement
HTML5	8	3	2	2
Web Messaging		1	2	
XMLHttpRequest 1 + 2	1			
CORS	2	1		
UMP				
Web Storage	3	1	1	
Geolocation API	5	1	1	1
Media Capture API			3	
System Information API	3	1	1	2
Widgets - Digital Signatures				2
Widgets - Access Req Policy	3			1
Total	**25**	**8**	**10**	**8**

Wrap-up

Conclusion

- Whole new range of security features
  - Browser-side enforcement, under control of the server
- NOT a replacement of secure coding guidelines, but an interesting additional line of defense for
  - Legacy applications
  - Newly deployed applications
- And most probably, there is many more to come in the next few years…

References

- Ph. De Ryck, M. Decat, L. Desmet, F. Piessens, W. Joosen. Security of web mashups: a survey (NordSec 2010)
- P. Chen, N. Nikiforakis, L. Desmet and Ch. Huygens. A Dangerous Mix: Large-scale analysis of mixed-content websites (ISC 2013)
- N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, Ch. Kruegel, F. Piessens, G. Vigna, You are what you include: Large-scale evaluation of remote JavaScript inclusions (CCS 2012)
- Ph. De Ryck et al., Web-platform security guide: Security assessment of the Web ecosystem (STREWS Deliverable D1.1)
- G.Rydstedt, E. Bursztein, D. Boneh, and C. Jackson. Busting frame busting: a study of clickjacking vulnerabilities at popular sites (W2SP 2010)
- Mike West. An introduction to Content Security Policy (HTML5 Rocks tutorials)
- Mike West. Confound Malicious Middlemen with HTTPS and HTTP Strict Transport Security (HTML5 Rocks tutorials)
- Mike West. Play safely in sandboxed iframes (HTML5 Rocks tutorials)
- Ivan Ristic. Internet SSL Survey 2010 (Black Hat USA 2010)
- Moxie Marlinspike. New Tricks for Defeating SSL in Practice (BlackHat DC 2009)
- Mike West. Securing the Client-Side: Building safe web applications with HTML5 (Devoxx 2012)
- B. Sterne, A. Barth. Content Security Policy 1.0 (W3C Candidate Recommendation)
- D. Ross, T. Gondrom. HTTP Header Frame Options (IETF Internet Draft)
- J. Hodges, C. Jackson, A. Barth. HTTP Strict Transport Security (HSTS) (IETF RFC 6797)
- C. Evans, C. Palmer, R. Sleevi. Public Key Pinning Extension for HTTP (IETF Internet Draft)