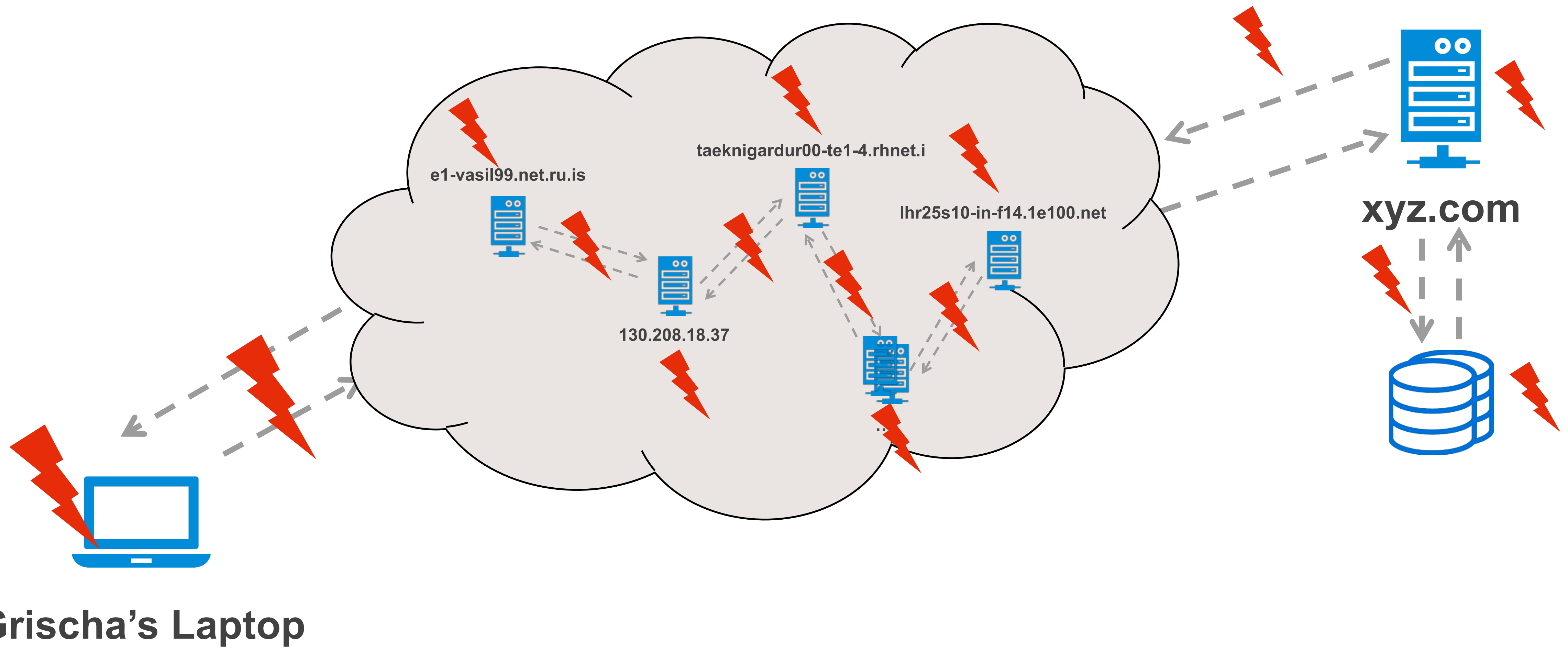
T-213-VEFF: Web Programming I

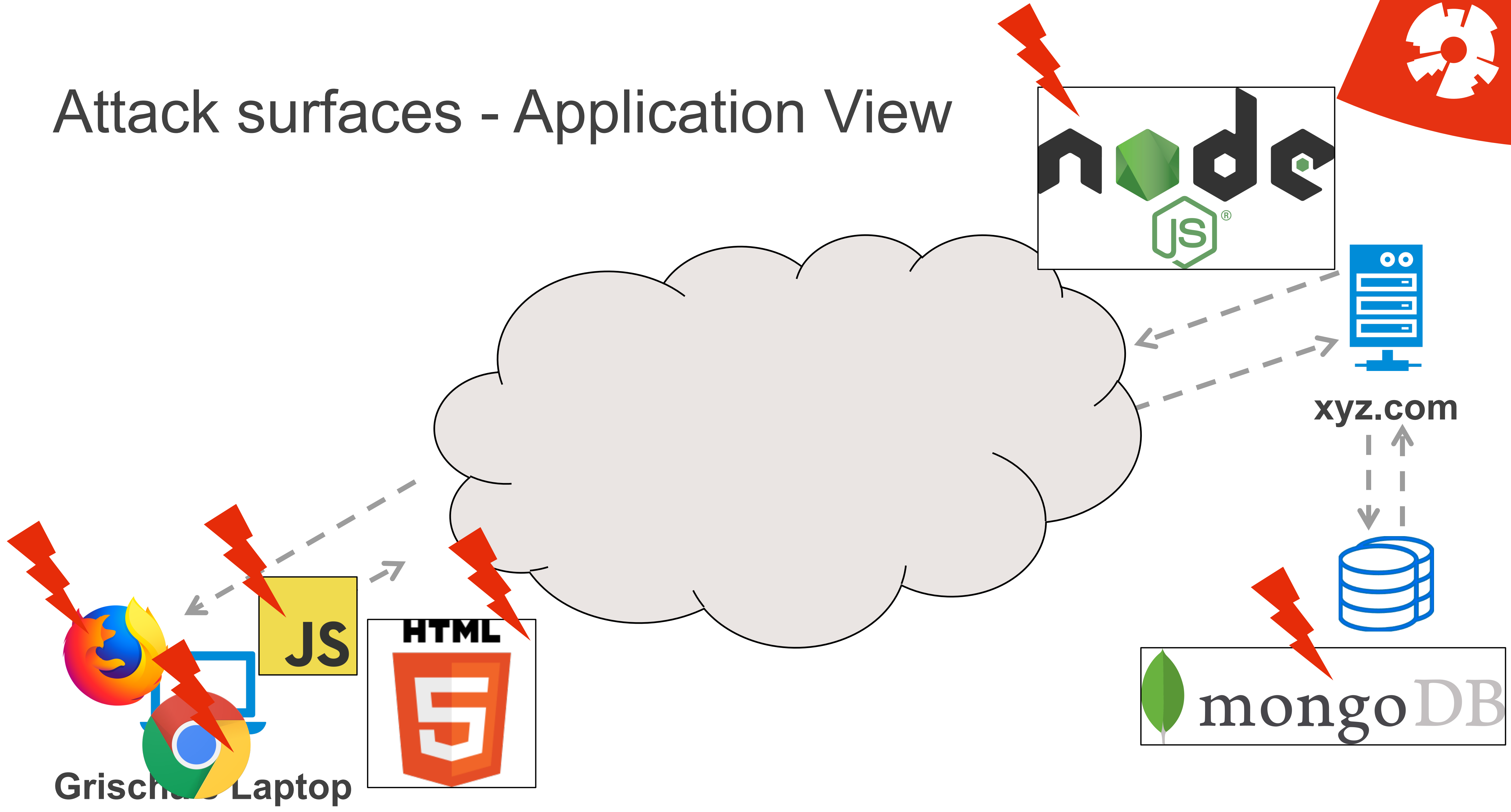
L18: Web Security I

Grischa Liebel

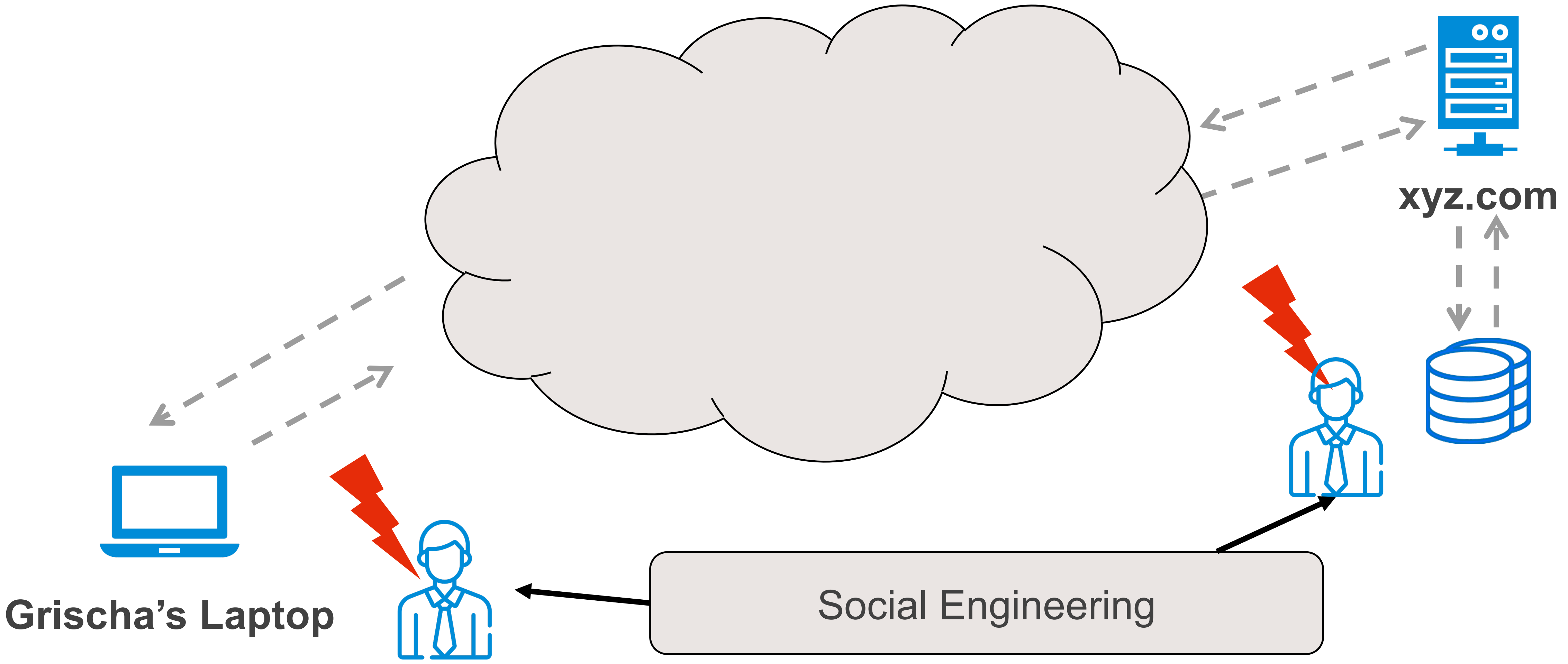
Attack surfaces - Network View



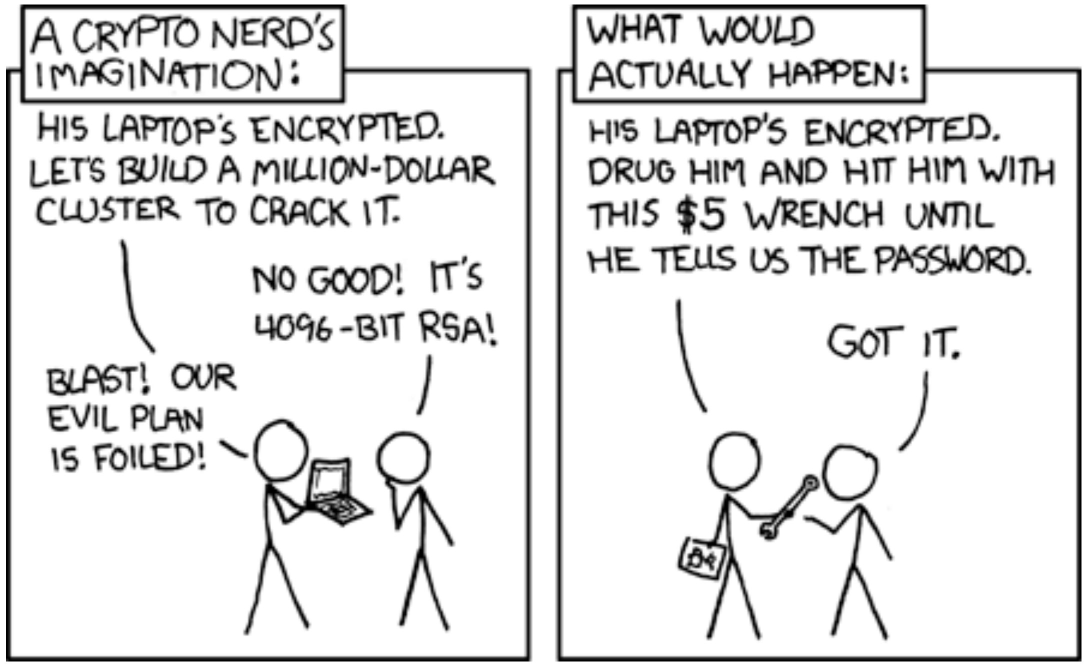
Attack surfaces - Application View

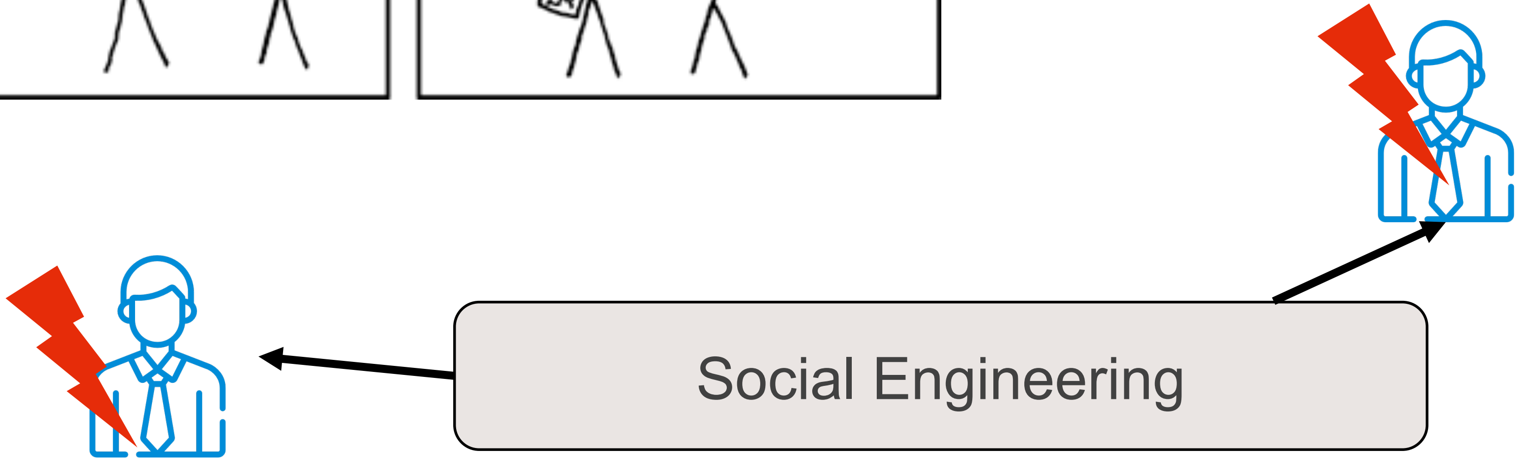


Attack surfaces - Do not forget the users!



Attack surfaces - Do not forget the users!


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.



Attack Surfaces

- There are many possible attack surfaces in web applications
  - We focus here on a few (specific to web applications)
  - Later courses cover other aspects of security (e.g., network security)

- So far in this course:
  - Unencrypted communication (HTTP)
  - All APIs completely open (all the endpoints can be used by all users)

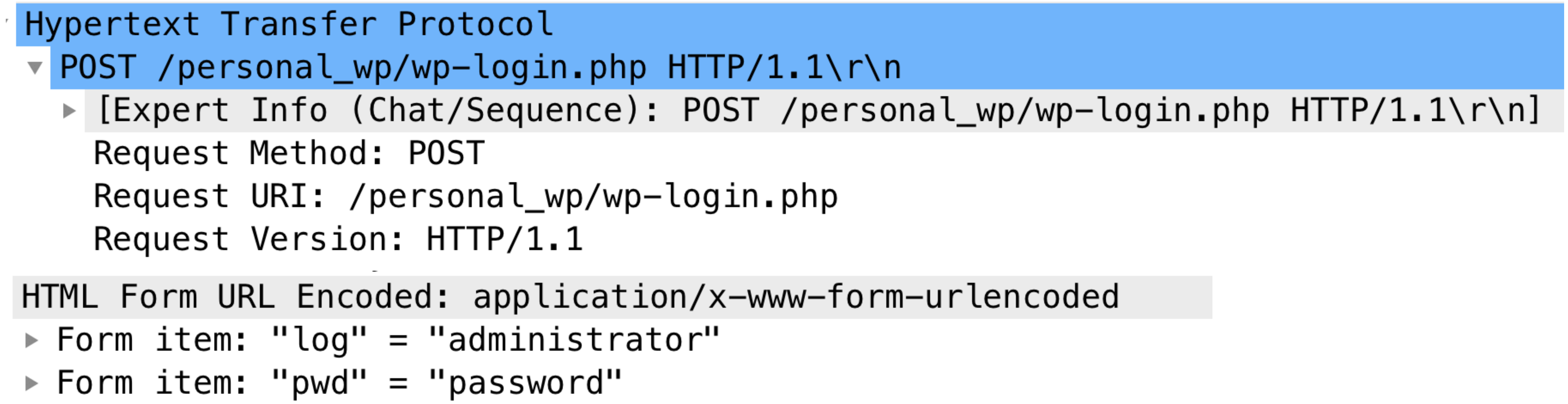
Some Security Principles

- Do not rely on "Security by Obscurity"
  - "Nobody knows the URLs of our API, so it's safe"
  - "Nobody knows how I implement my security, so it's safe"
  - "Nobody knows that I hide my money under the pillow, so it's safe"

- Rely instead on popular (open source) approaches to security
  - Linus's Law: "Given enough eyeballs, all bugs are shallow"

Use HTTPS

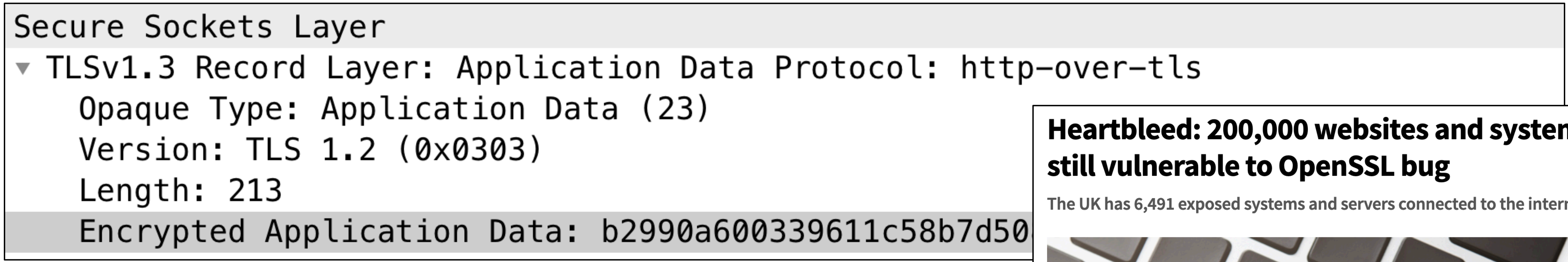
- HTTP transmits all requests/responses in clear text
  - Including all headers
  - Including passwords, tokens, session ids, ...
  - It's trivial to read that information within the same network



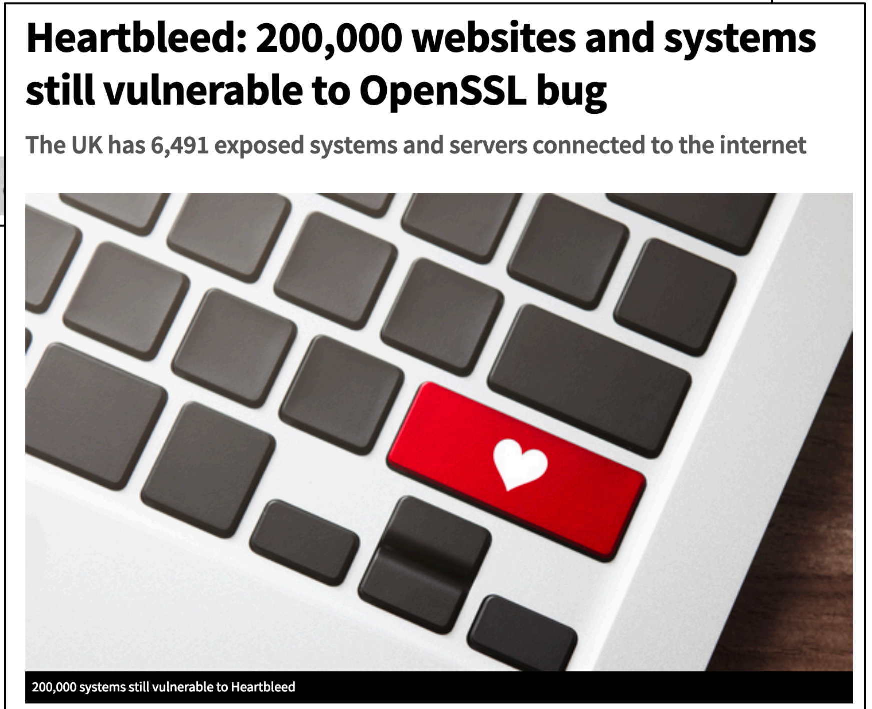
```
Hypertext Transfer Protocol
▼ POST /personal_wp/wp-login.php HTTP/1.1\r\n
  ▶ [Expert Info (Chat/Sequence): POST /personal_wp/wp-login.php HTTP/1.1\r\n]
    Request Method: POST
    Request URI: /personal_wp/wp-login.php
    Request Version: HTTP/1.1

HTML Form URL Encoded: application/x-www-form-urlencoded
▶ Form item: "log" = "administrator"
▶ Form item: "pwd" = "password"
```

Use HTTPS

- HTTPS uses secure (encrypted) communication
  - Much harder to access the requests/responses



```
Secure Sockets Layer
▼ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 213
    Encrypted Application Data: b2990a600339611c58b7d50
```



Heartbleed: 200,000 websites and systems still vulnerable to OpenSSL bug

The UK has 6,491 exposed systems and servers connected to the internet

200,000 systems still vulnerable to Heartbleed

  - But: Do not assume that it's impossible!

Use Authentication/Authorisation

- Make sure your web application/API is only accessible by the 'right' people
  - Registered users
  - Users with the 'right' credentials

- Users have to be authenticated or authorised

Summary

- There are numerous **attack surfaces** in web applications, on different layers
  - E.g.: Attacks on the **network**, the **applications** (through bugs), the user (**social engineering**)
- HTTP communication is **entirely unencrypted**, HTTPS should be used instead
  - But also HTTPS should not be assumed to be 100% safe
- Different **authentication/authorisation** mechanisms exist
  - All with their own trade-offs
  - Principle: Do not implement authentication all by yourself
  - Use external, well-tested libraries and auth delegation (OAuth 2.0)