Fundamentos de Segurança Informática (FSI)

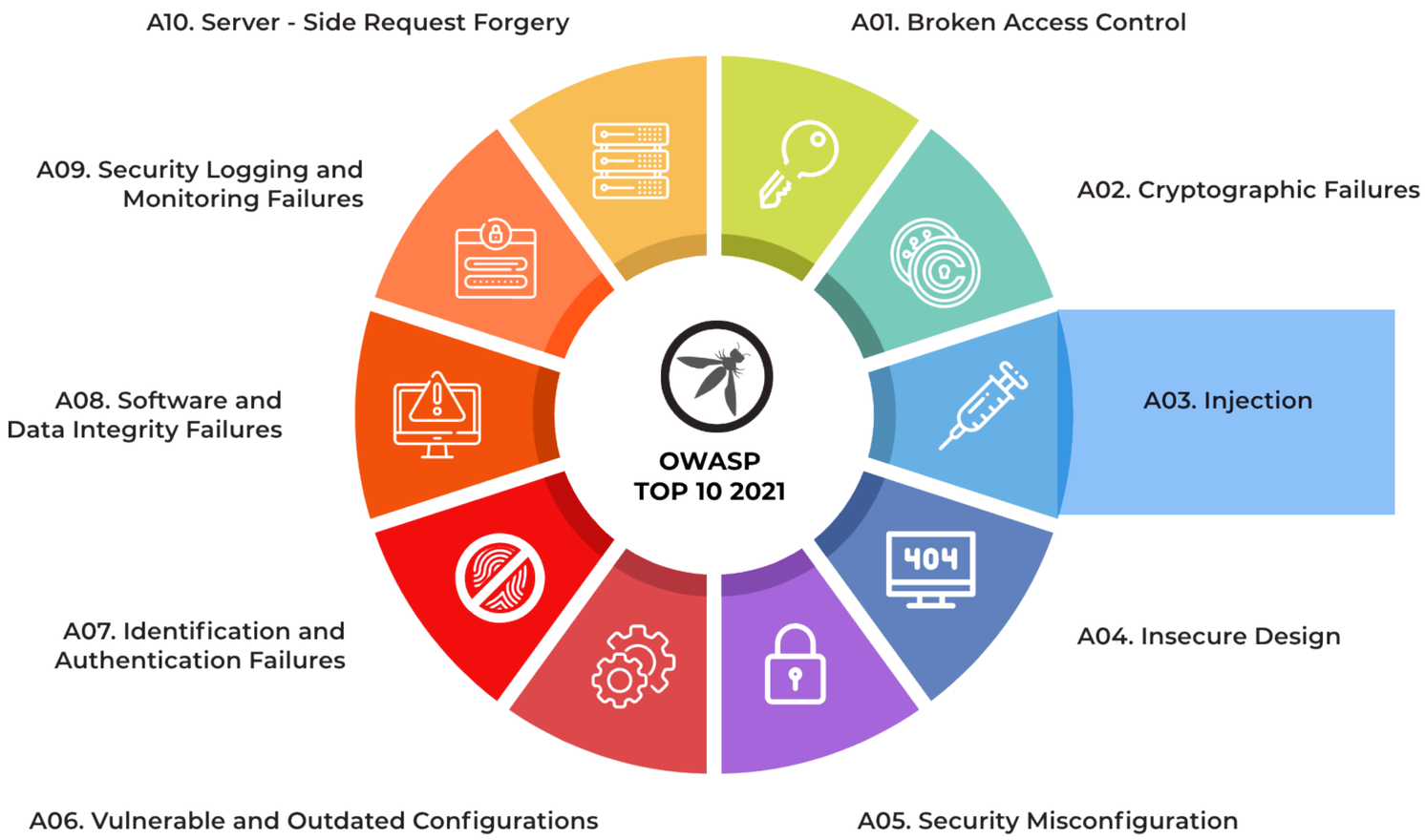
2024/2025 - LEIC

Web Security (Part 3)

Hugo Pacheco
hpacheco@fc.up.pt

OWASP Top 10

- We will look at other attacks beyond injection



A01:2021
Broken Access Control

https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Insecure Direct Object Reference (IDOR)

Insecure Direct Object Reference

- Among the simplest vulnerabilities, also called Web Parameter Tampering: inexistent access control 🤦

- Citibank (2011) lost 360k credit cards

  - **Attack:** `https://citi.com/myacct/9725126314/summary`

Citi Credit Card Hack Bigger Than Originally Disclosed

Citigroup has been forced to reveal that a recent hack of its network exposed the financial data of more than 360,000 customers, a much higher number than the bank originally disclosed.

https://www.wired.com/2011/06/citibank-hacked/

- Failure in the Portuguese Sinave platform from the DGS (2022)

Vulnerabilidade no website da DGS expõe dados pessoais dos portugueses como nomes, moradas e número de telefone

A vulnerabilidade foi detetada em março e corrigida poucos dias depois, não se sabendo se houve aproveitamento por parte de agentes maliciosos.

Em março foi detetada uma vulnerabilidade na plataforma do Sinave (Sistema Nacional de Vigilância Epidemiológica) no website da Direção-Geral da Saúde, que permitia aceder a dados pessoais dos portugueses, sem a necessidade de efetuar qualquer tipo de autenticação.
Segundo foi avançado pelo Público, o **problema dava acesso a informações tais como o número de identificação fiscal, a morada, número de telefone, a data de nascimento e o nome dos registados**.

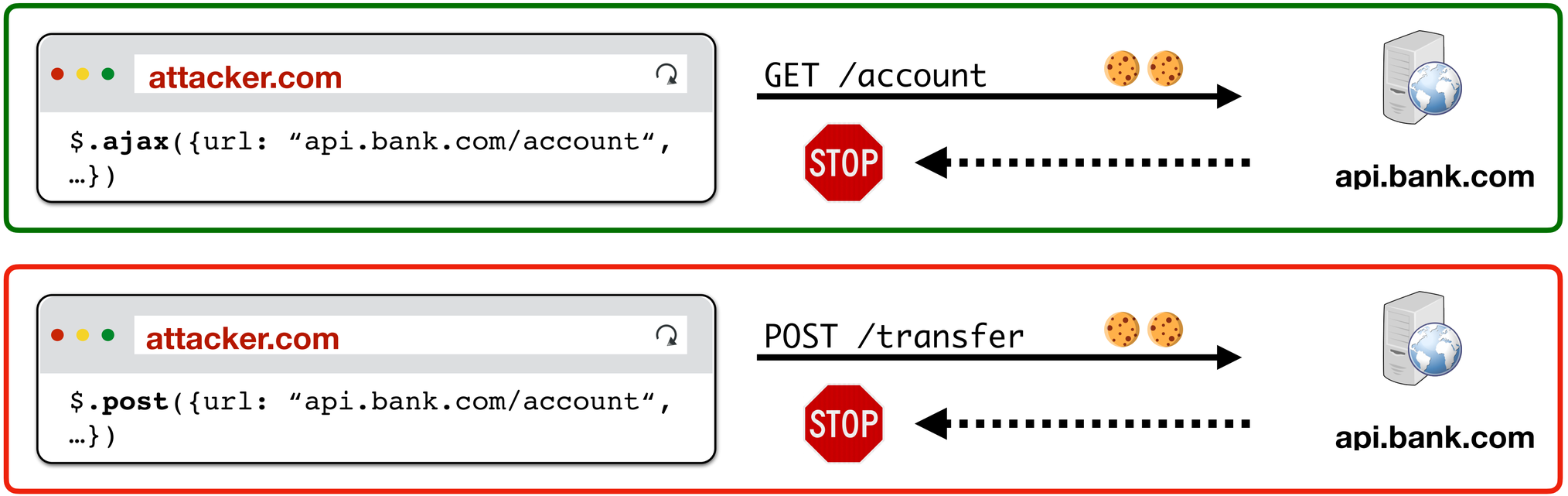
A falha foi encontrada por um programador português, quando estava a tentar ligar a base de dados do website aos serviços digitais de um cliente. O programador **percebeu que estavam disponíveis abertamente extensas bases de dados privados**.

O problema foi corrigido poucos dias depois do Centro Nacional de Cibersegurança ter sido alertada. No entanto, **não se sabe se a falha chegou a ser explorada por agentes maliciosos ou durante quanto tempo esteve exposto**. Segundo o jornal, o erro foi encontrado por acaso, bastando colocar mais alguns caracteres no próprio endereço do website. Este tipo de dados pode ser vendido no mercado negro a cibercriminosos, assim como empresas de publicidade e outros.

https://tek.sapo.pt/noticias/internet/artigos/vulnerabilidade-no-website-da-dgs-expoe-dados-pessoais-dos-portugueses-como-nomes-moradas-e-numero-de-telefone

Cross-Site Request Forgery (CSRF)

Using cookies for authentication

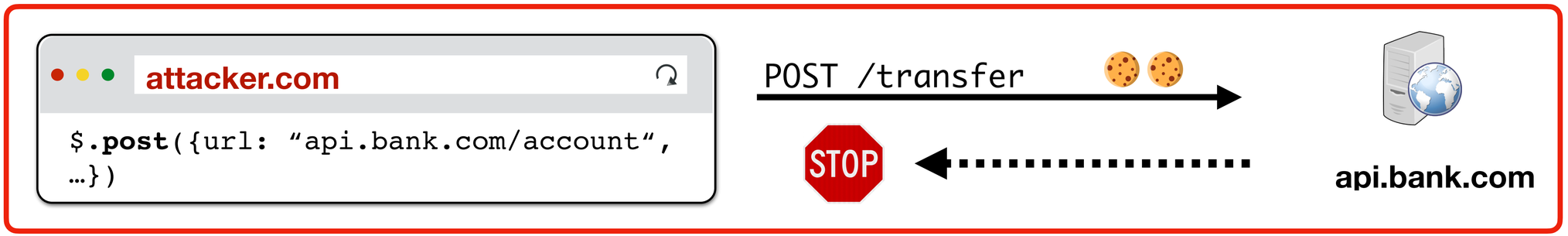


- Who has access to the cookie (legitimate client) can make requests
- SOP? The cookie is sent… Server does not know the origin of the request!
- **Cookie-based authentication is not sufficient when requests have side-effects!**

CSRF

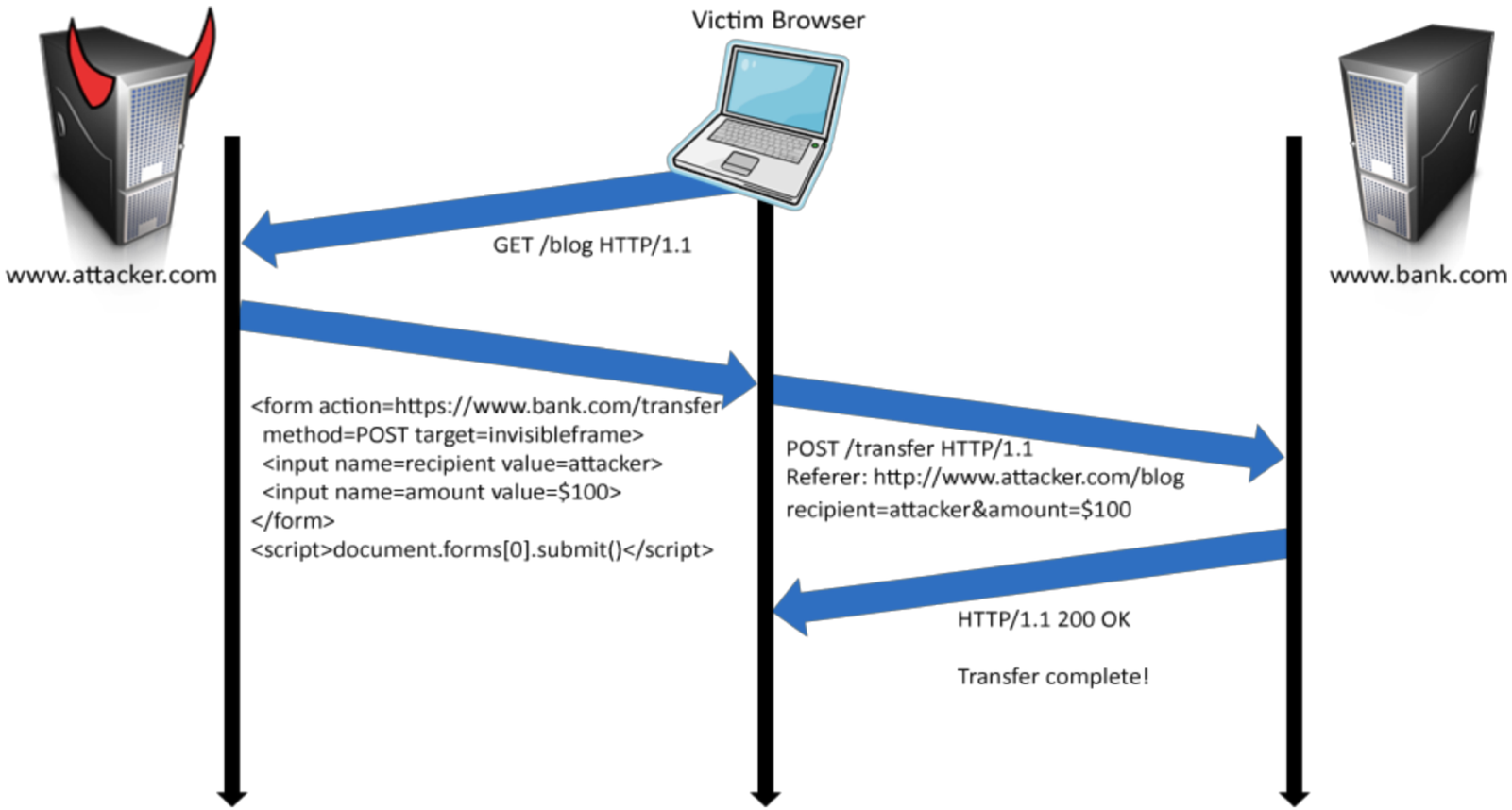
- Cross-site request forgery (CSRF):

1. Attacker tricks a legitimate user into clicking a link (e.g, social engineering)
2. Legitimate user makes unintended / unrealised request to legitimate website
3. Legitimate website executes operations **in the name of the legitimate user**



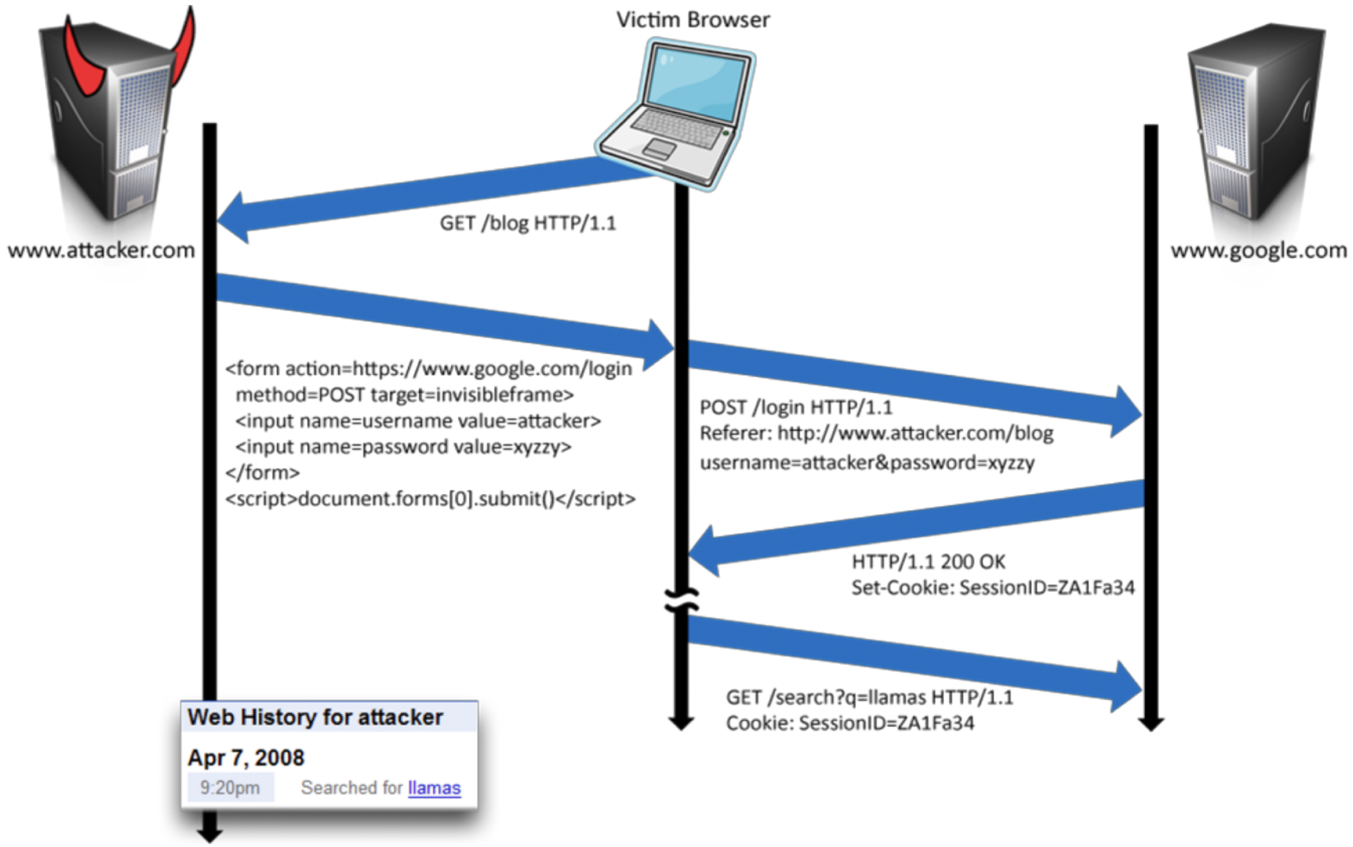
CSRF

- Attacker can dissimulate request which causes a side-effect **in the server**:
  - SOP ⇒ attacker cannot see the response on another origin 👍
  - But the side-effect is executed 😱



CSRF

- Another example of CSRF:
  - Malicious site creates a session in its name in a target site (e.g., Google)
  - The user's search history becomes tracked in the attacker's account! 😱



XS-Leaks

- Related to CSRF (but client-side, at the level of the browser):
  - Attacker can make requests in the name of the user …
  - … cannot see the response 👍
  - … but can use side-channels 😱
    - E.g., monitoring error events, measuring response time, counting frames
    - Attack works even for requests without server side-effects 😱
  - Repeating requests may extract lots in information, e.g., transaction history

https://xsleaks.dev/

CSRF

Default Key Algorithm In Thomson And BT Home Hub Routers

Mon, 14 Apr 2008 08:00:33 GMT
by pagvac

Yes, we're back with more embedded devices vulnerability research! And yes, we're also back with more security attacks against the BT Home Hub (most popular DSL router in the UK)!

As you know, we encourage folks in the community to team up with us in different projects as we've had very successful experiences doing so. This time it was Kevin Devine's turn. Kevin, who is an independent senior security researcher, did an awesome job at reverse engineering the **default WEP/WPA key algorithm** used by some Thomson Speedtouch routers including the BT Home Hub. Kevin noticed that all the public vulnerability research conducted in the past for the BT Home Hub had been released by GNUCITIZEN, so he decided to share his findings and work with us in this fascinating project.

- The problem is not limited to cookies!
  - Malicious site makes request to home router
  - If it exists, tries to login with default credentials ⇒ remember
  - If successful, changes settings (e.g., DNS, firmware, etc)
    - Phishing  , e.g., fake bank web page
    - Stealing traffic  , e.g., towards attacker's ads

https://decoded.avast.io/threatintel/router-exploit-kits-an-overview-of-routercsrf-attacks-and-dns-hijacking-in-brazil/

CSRF

- Many applications:
  - Employ web technology to simplify cross-platform deployment
  - Run locally in server mode!

https://medium.com/@keegan.ryan/patched-zoom-exploit-altering-camera-settings-via-remote-sql-injection-4fdf3de8a0d

Preventing CSRF

- **Allow the server to know that a request comes from a trusted page**
  - **CSRF tokens**: secret token (dynamic) in the HTML form that the attacker cannot infer when simulating a POST request

```
<form action="https://bank.com/transfer" method="post">
  <input  type="hidden" name="csrf_token" value="434ec7e838ec3167ef5">
  <input  type="text" name="to">
  <input  type="text" name="amount">
  <button type="submit">Transfer!</button>
</form>
```

  - **SameSite=Strict**: prevent use of cookies in cross-site requests (e.g., if we follow a link that comes from an email, we won't be authenticated)
  - **SameSite=Lax**: allows following cross-site links (such as the example above), but does not permit cross-site requests when loading images or frames

Preventing CSRF

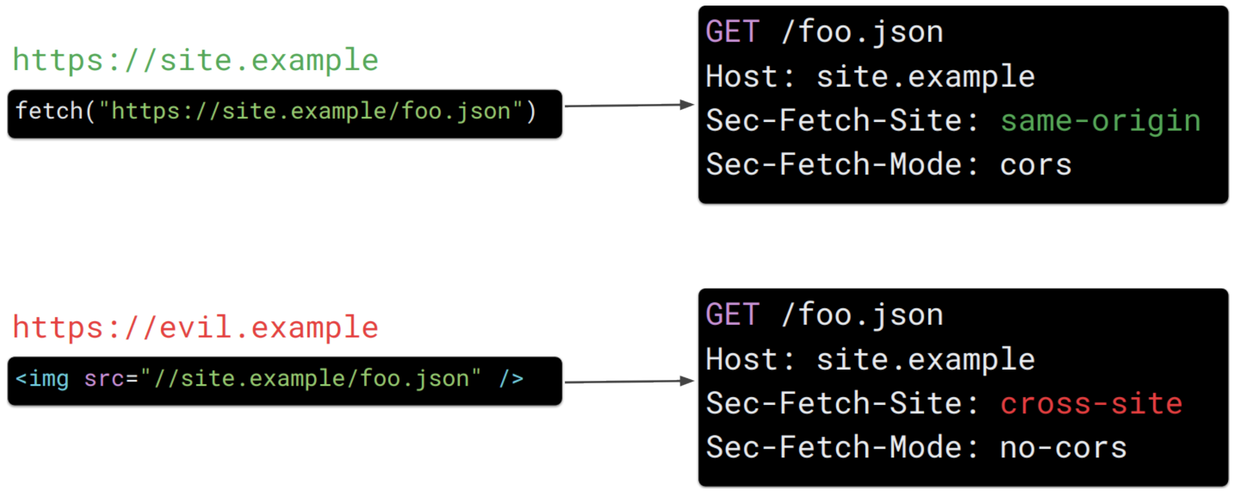
- **Allow the server to make decisions based on the request's origin**
  - **Referrer validation**
    - When following links, explicit server validation of `Referrer` and `Origin` headers ⇒ tracking via referrer… 😮
      See also header Referrer-Policy

Referrer		Origin	
https://bank.com	→	https://bank.com	✓
https://attacker.com	→	https://bank.com	X
	→	https://bank.com	??

  - Who can control these headers? user, site, …

Preventing CSRF

- **Allow the server to make decisions based on the request's origin**
  - New Fetch Metadata headers (**enforced by the browser**)
    - `Sec-Fetch-Site`: Which site originated the request? `same-origin`, `same-site`, `cross-site`, `none`
    - `Sec-Fetch-Mode`: Which kind of request?
    - `Sec-Fetch-User`: Was the request explicitly made by the user (e.g. clicking on a button)?



https://webappsec.dev/assets/pub/Google_IO-Securing_Web_Apps_with_Modern_Platform_Features.pdf

Preventing CSRF

- Remember: we are always trusting the client's browser (may be old…)
- **Defense in depth:**
  - Try forcing CORS pre-flight using custom-headers
  - The `Cross-Origin-Opener-Policy` header guarantees that the a site which makes a cross-origin request in a new page (e.g., popup) loses reference to that request (e.g., in JavaScript)

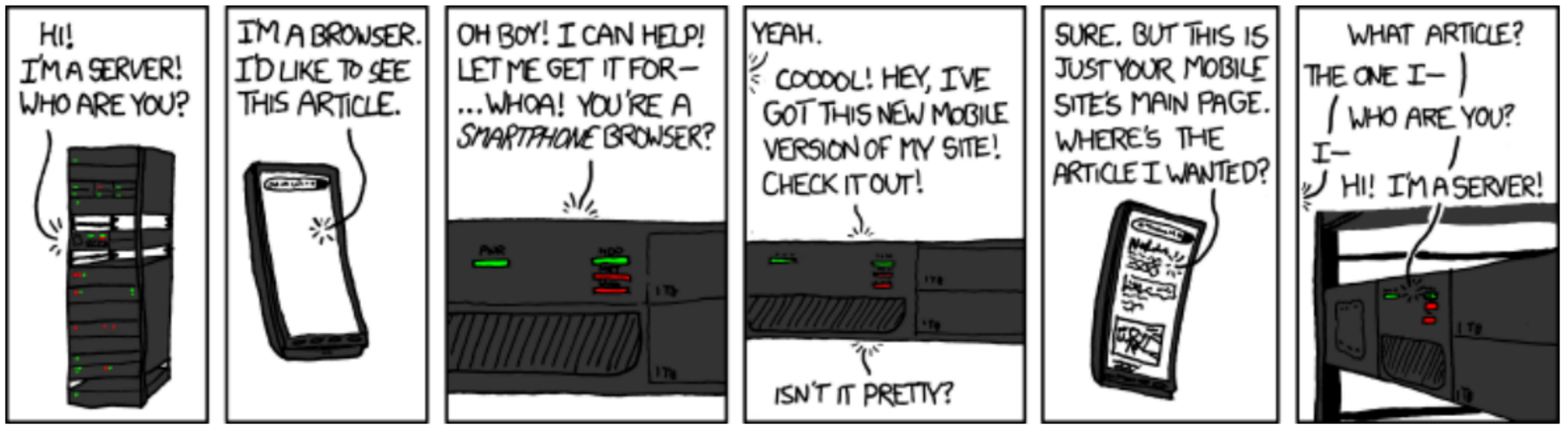
    ⇒ preventing XS-Leaks

A07:2021
Identification and Authentication Failures

https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

Session Management Flaws

- Remember:
  - HTTP is inherently stateless
  - Cookies are used to simulate state (it is just a simulation though, any party may at any time abandon it…)



https://xkcd.com/869/

Session Management Flaws

- **Server-side** session management:
  - Server stores and manages all state information
  - Creates a random token to serve as a reference for the client cookie
  - **Recommended as the most secure solution**
  - Lots of state management and operational overhead:
    - Requires high entropy, fresh session IDs, securely stored, invalidated after logout or idle ⇒ attacks when any fails 

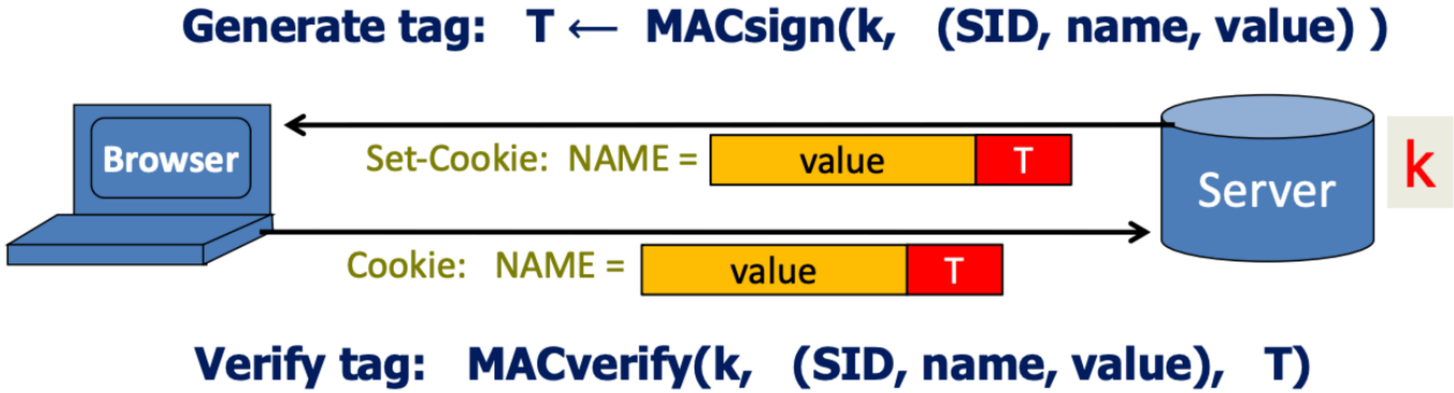
Session Management Flaws

- **Client-side** session management:
  - Client stores and manages all state information in its cookie
  - Much more flexible to deploy for developers
  - **A much more dangerous solution**
  - Client can edit the cookie or modify the `Cookie` header! ⇒ cookie poisoning 
  - Similar problems with local storage and hidden fields:

```
localStorage.setItem("price", "150");
<INPUT TYPE=“hidden” NAME=price VALUE=“150”>
```

Session Management Flaws

- **Client-side** session management:
  - Server can encrypt cookies to guarantee confidentiality?
    - You really should not include any sensitive information in a cookie…
    - … for whatever you do, use secure cookies only sent by HTTPS
  - Server can use MACs to guarantee integrity of server-generated cookies against client disruption (more later)
    - Requires server-side secret key k



A04:2021
Insecure Design

https://owasp.org/Top10/A04_2021-Insecure_Design/

Insecure Design

- General, much overlap with other categories ⇒ **new in OWASP Top 10 2001 ⇒ goal is to enforce the principle of Secure by Design**
- Encompasses **business logic errors** ⇒ **typically harder to detect**
  - Requires understanding the context of the application!
  - Cannot be fixed by a “perfect implementation” or by general countermeasures such as input sanitisation
  - Often outside the scope of automated tools

Insecure Design

- Web applications are split across **servers** and **clients**
- **Server** cannot trust the **client**
  - Typical vulnerability: validating input only in the **client** frontend
    - **Client** can send invalid requests, e.g., buy a negative amount of a product
  - Typical vulnerability: **server** generates error messages with sensitive information (e.g., versions)
- **Client** cannot trust the **server**
  - Typical vulnerability: **server** may be corrupted (e.g., due to stored XSS)
  - Typical vulnerability: **server** insecurely stores credentials

Business Logic Flaws

- E.g., online store: `$Total = $Price x $Quantity`
- What if I order a negative number of books?
  - Do I get rejected? Do you send me the `$Total` and presumably wait for me to ship you books?
- What if I order 0.1 TVs?
  - I pay 10% what I should have paid? Does the shipping department just send me 10% of a TV?

How to Shop for Free Online -- Security Analysis of Cashier-as-a-Service Based Web Stores
https://doi.org/10.1109/SP.2011.26

Business Logic Flaws

- E.g., press release site that announces earnings for publicly traded companies
- Only to be made available after a scheduled time
- Earnings are staged on the site ahead of time, but are not publicly linked until release…
- Maybe no one will notice?

Ukrainian Hacker Admits Role In Largest Known Computer Hacking And Securities Fraud Scheme

Monday, May 16, 2016

For Immediate Release

U.S. Attorney's Office, District of New Jersey

First Hacker Convicted in Conspiracy to Steal 150,000 Press Releases from Three Major Newswire Companies for Use in Illicit Trades

NEWARK, N.J. – A Ukrainian hacker today admitted his role in an international scheme to hack into three business newswires, steal yet-to-be published press releases containing non-public financial information, and use the information to make trades that allegedly generated approximately $30 million in illegal profits, U.S. Attorney Paul J. Fishman announced.

https://vulnsite/press_release/08/29/2007/00014.html
https://vulnsite/press_release/08/29/2007/00015.html

https://www.justice.gov/usao-nj/pr/ukrainian-hacker-admits-role-largest-known-computer-hacking-and-securities-fraud-scheme

A05:2021
Security Misconfiguration

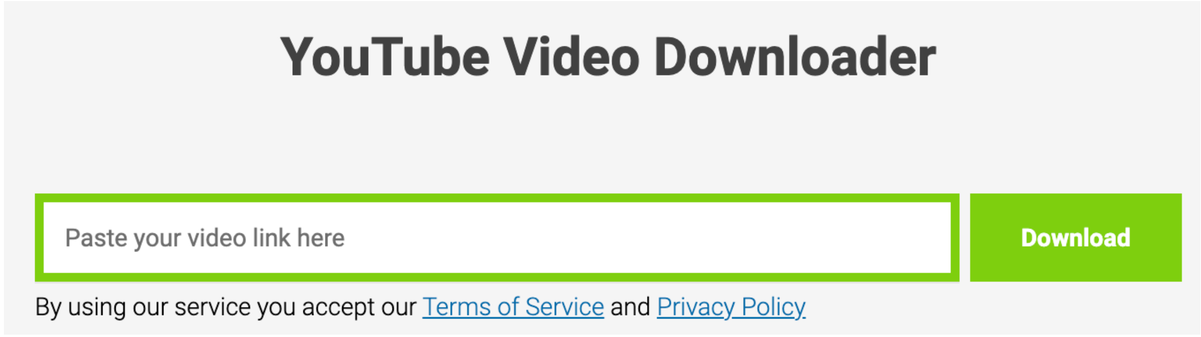
https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

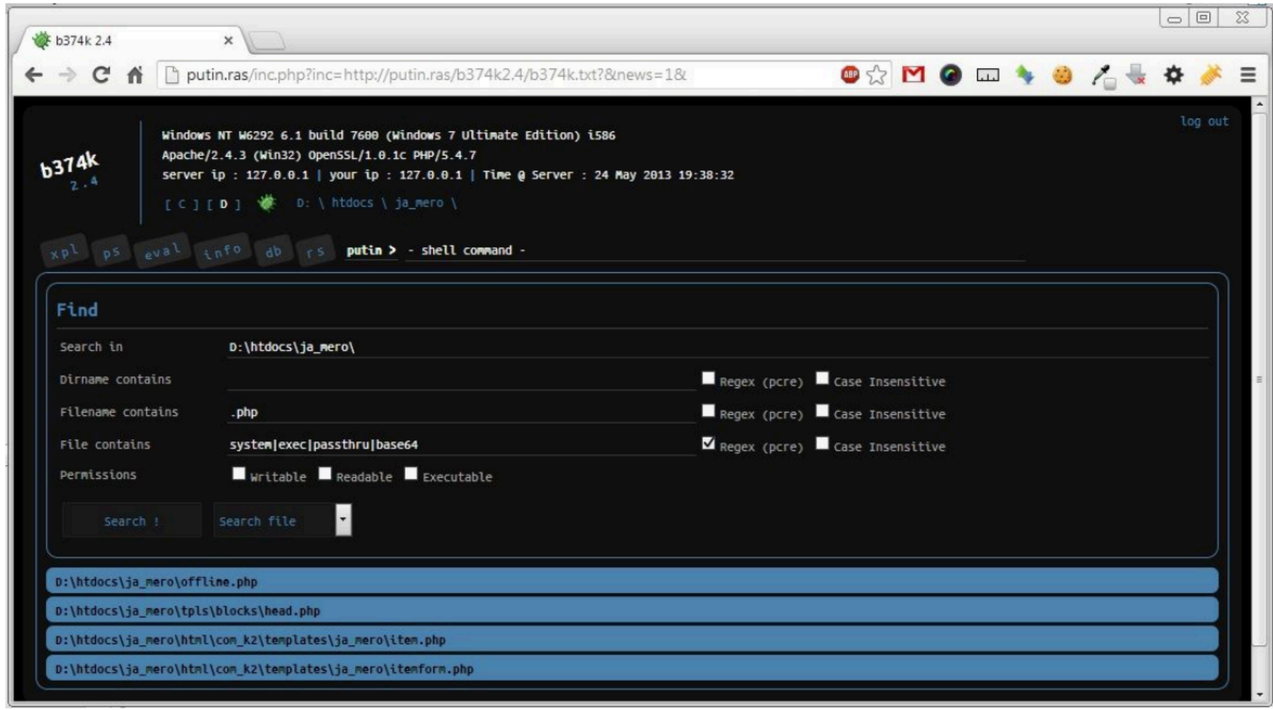
Undesired Disclosure

- **Web applications often (and inadvertently) expose sensitive data:**
  - Forgotten CGI-vulnerable files
  - “Private” API endpoints not meant for the public
  - Backup of the site or backend DB
  - Untreated error messages (revealing system or database configuration, etc)
  - Public version control repositories in production (see https://en.internetwache.org/dont-publicly-expose-git-or-how-we-downloaded-your-websites-sourcecode-an-analysis-of-alexas-1m-28-07-2015/)
- There are various tools for automated discovery of this data!
- **Countermeasures:** ensure proper access control and exception handling; disable directory listing

File Uploads

- **Web applications are under threat if they:**
  - Fetch files on behalf of clients
  - Allow clients to upload files
- **All input is potentially malicious:**
  - Attacker may perform arbitrary injection
  - Attacker may get a malicious web application running on the server
  - Attacker may get full access to the system, e.g., webshells



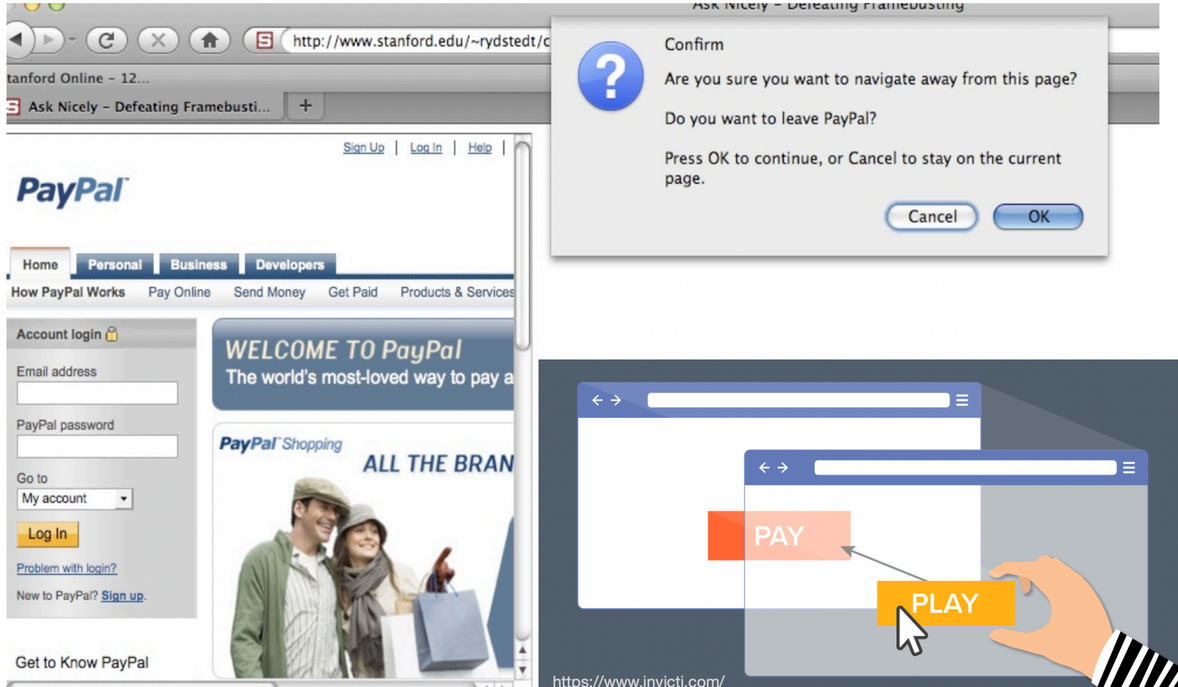


File Uploads

- **Countermeasures** for uploaded files**:**
  - Give them hard-to-guess names
  - Do not allow direct access to them via the web page
  - If media files (image, video, etc), push them to an external CDN
  - Scan them with an antivirus
  - Take particular care with archival / compression formats:
    - Zip bombs (often used against the antivirus itself!) 
    - Embedded absolute / relative paths 

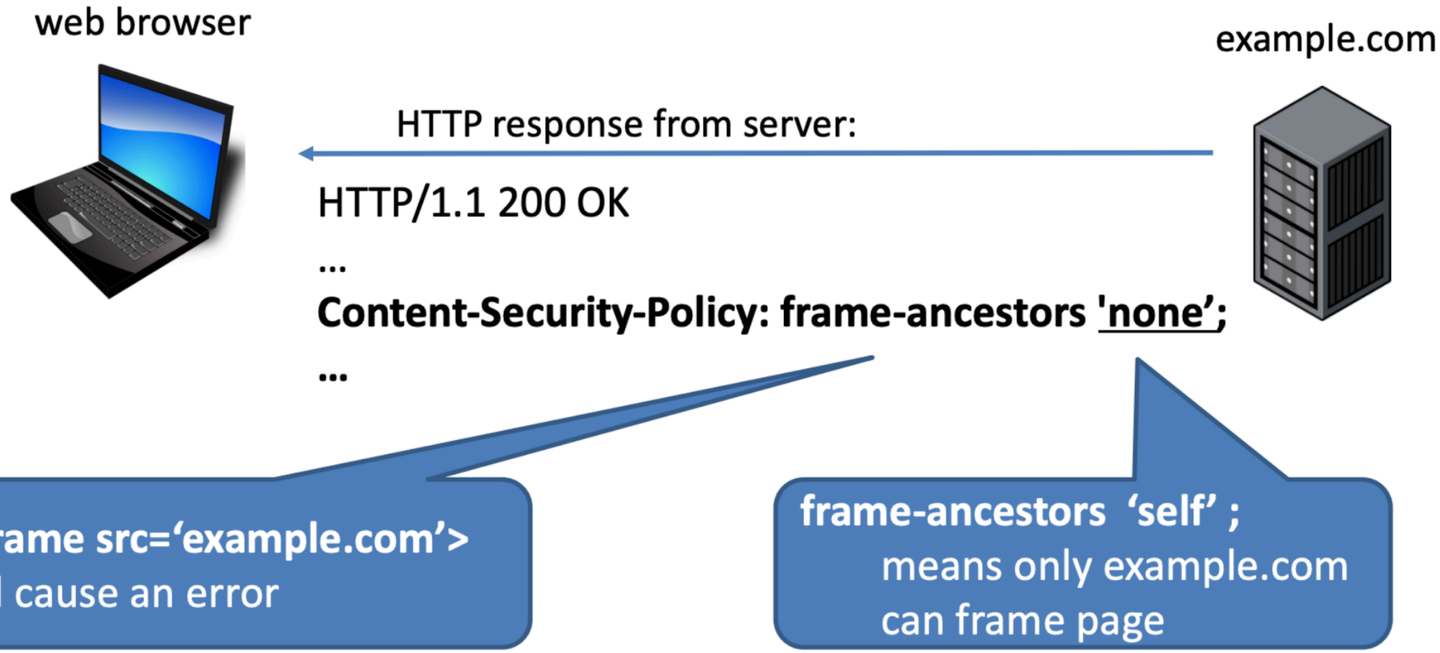
Clickjacking

- Web page components can overlap and be transparent
- **Clickjacking**: user 🤓 visits malicious site ⇒ attacker 😈 makes him see one frame but interaction is with another frame
- **Attack 1:** User see malicious frame but is interacting with an underlying victim frame
  - E.g., malicious web site displays a game to compel user to click on parts of the screen
- **Attack 2:** User sees victim frame but is interacting with malicious frame
  - E.g., victim frame is positioned so that a specific UI element is right below a button on the malicious frame



Clickjacking: protections

- HTML5 Sandboxes
  - Sandboxed frame is treated as being from a unique origin
- `X-Frame-Options: deny, sameorigin`
  - Page cannot be rendered on a frame in another origin
- CSP: frame-ancestors ‘none’
  - site cannot be “framed” by another site



Phishing

- One of the most common **social engineering attacks**:

  1. Attacker sends a fraudulent message that tricks user into visiting a mockup of a trusted web site 😈

  2. User visits malicious site and reveals sensitive data (e.g., login, credit card) 🤓

- Almost all phishing attacks take place over the web: difficult to know if you're in the right place as a user 😨

Facebook and Google Were Victims of 100 Million-Dollar Phishing Scam

Stu Sjouwerman

Tweet Share

We have been reporting on this massive Cyberheist for a while now, but Fortune Magazine decided to unleash their investigative reporters and find out exactly who those two mysterious high-tech companies were that got snookered for a whopping 100 million dollars.

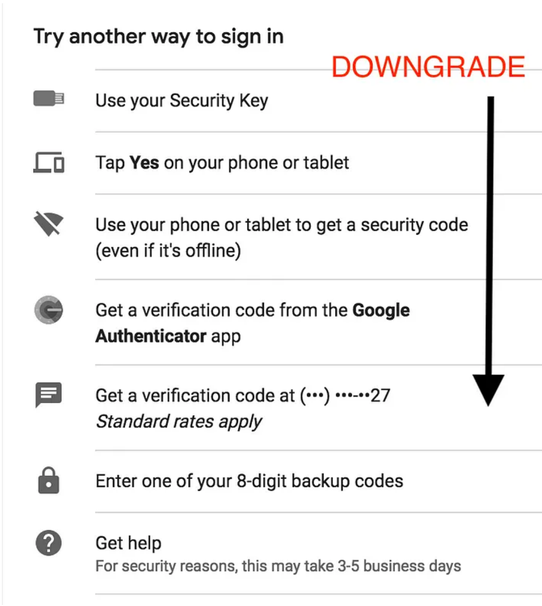
It is excellent ammo to send to C-level executives to illustrate the urgent need to train employees so they can recognize red flags related to spear phishing.

In 2013, a 40-something Lithuanian named Evaldas Rimasauskas allegedly hatched an elaborate scheme to defraud U.S. tech companies. According to the Justice Department, he forged email addresses, invoices, and corporate stamps in order to impersonate a large Asian-based manufacturer with whom the tech firms regularly did business. The point was to trick companies into paying for computer supplies.

https://blog.knowbe4.com/facebook-and-google-were-victims-of-100-million-dollar-phishing-scam

Phishing: protections

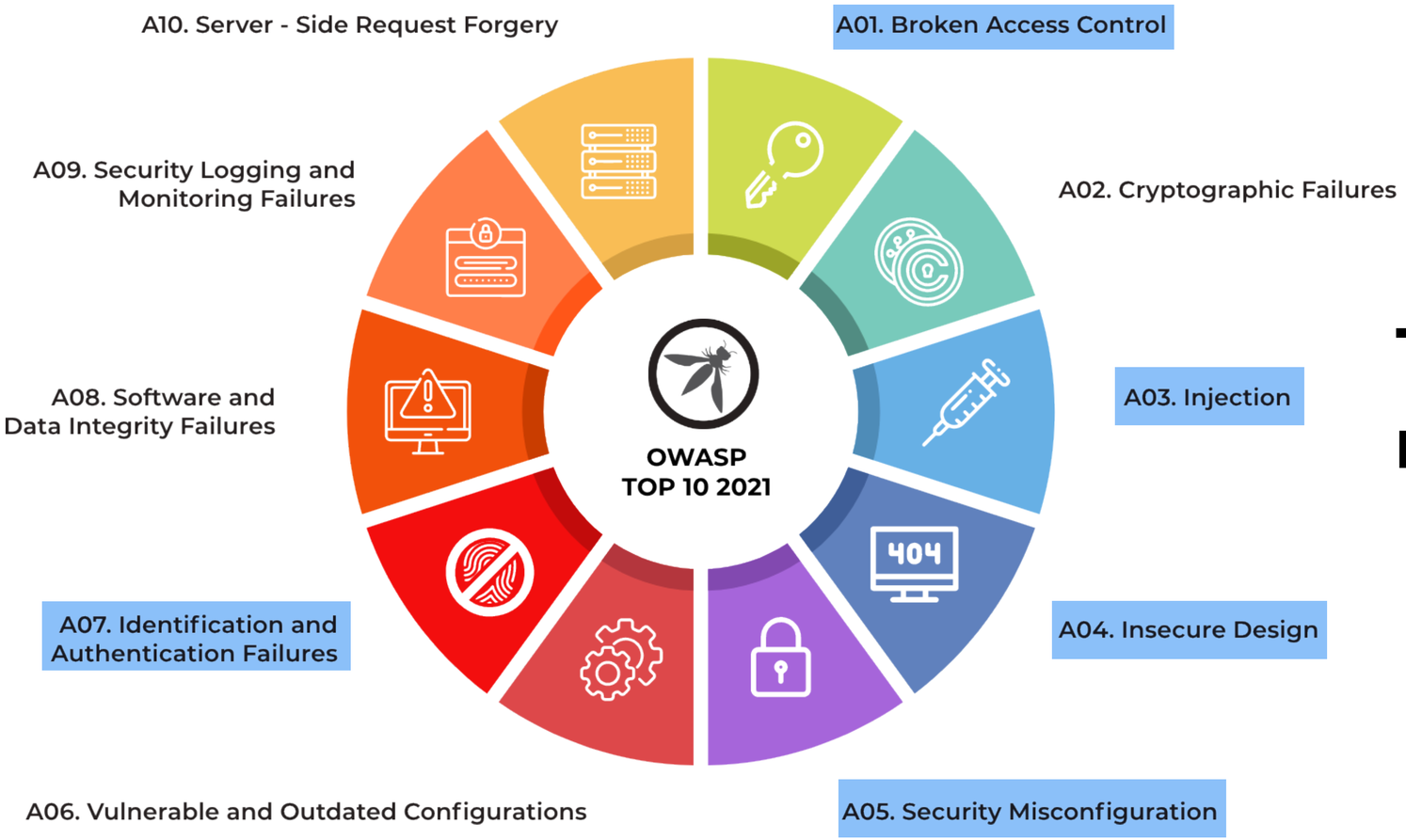
- Main protection is a heightened good sense 
- Two-Factor Authentication (2FA) **does little good**
  - Mostly protects against stolen credentials
  - User also sends his tokens to the attacker
- Universal 2nd Factor (U2F) **solves it** …
  - Small portable device stores user's secret keys
  - Attacker cannot play the middle man
- … if attackers cannot downgrade the authentication method
  - What happens if the user loses his hardware token?



https://eprint.iacr.org/2020/1298

OWASP Top 10

- We have seen examples of…



There is still much more…

Next class ⇒ bportela

Acknowledgements

- This lecture's slides have been inspired by the following lectures:
  - CSE127: Web Security II
  - CS155: Web Attacks + Web Defenses
  - CS343: Web Security