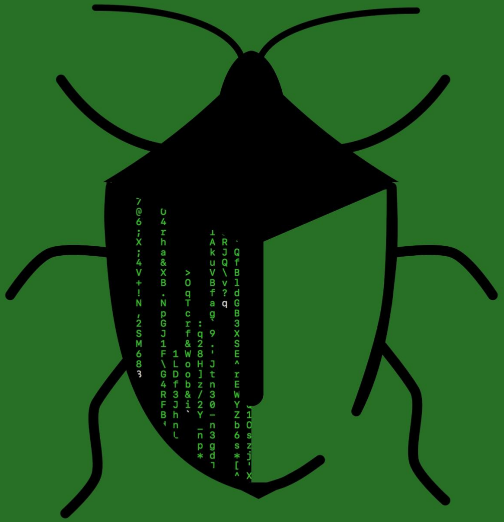
Sig●Mal

Meeting 9

What’s poppin’ ?

```
/
\ o
  | \
  | |
 / \
/   \
```



Web Security

- Directory Traversal
- Session Hijacking / Session Spoofing (Cookies)
- Client-Side Authentication
- Plain-text Passwords
- XSS (Cross-Site Scripting)
- SQL Injection

Client-Side → “What you see”. On the website

Server-Side → “You don’t really see”. In the server

Directory Traversal

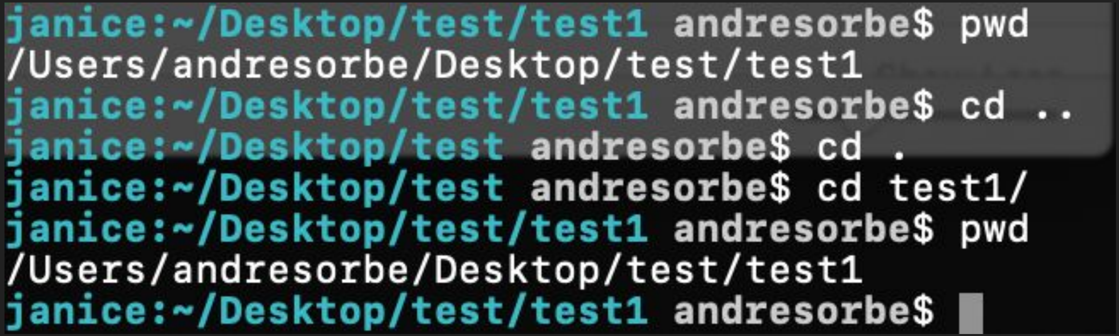
What is it?

Directory traversal (also known as file path traversal) is a web security vulnerability that allows an attacker to read arbitrary files on the server that is running an application.

How does it work?

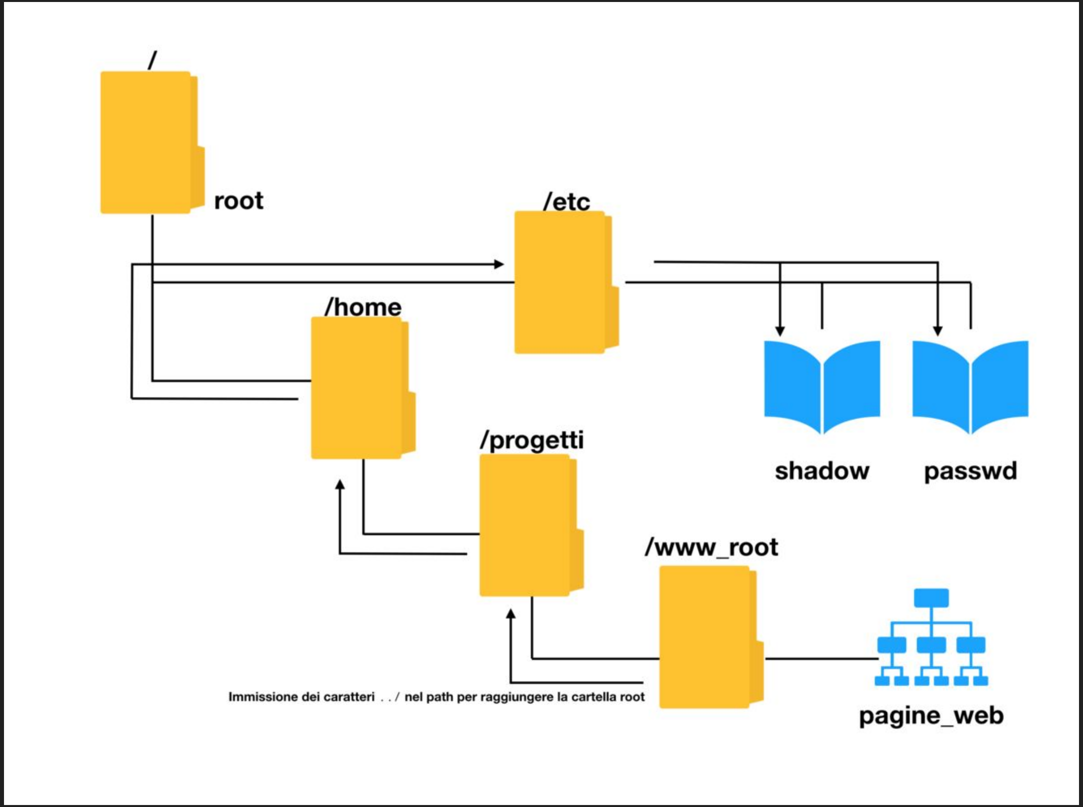
. → Current Directory

.. → Previous Directory



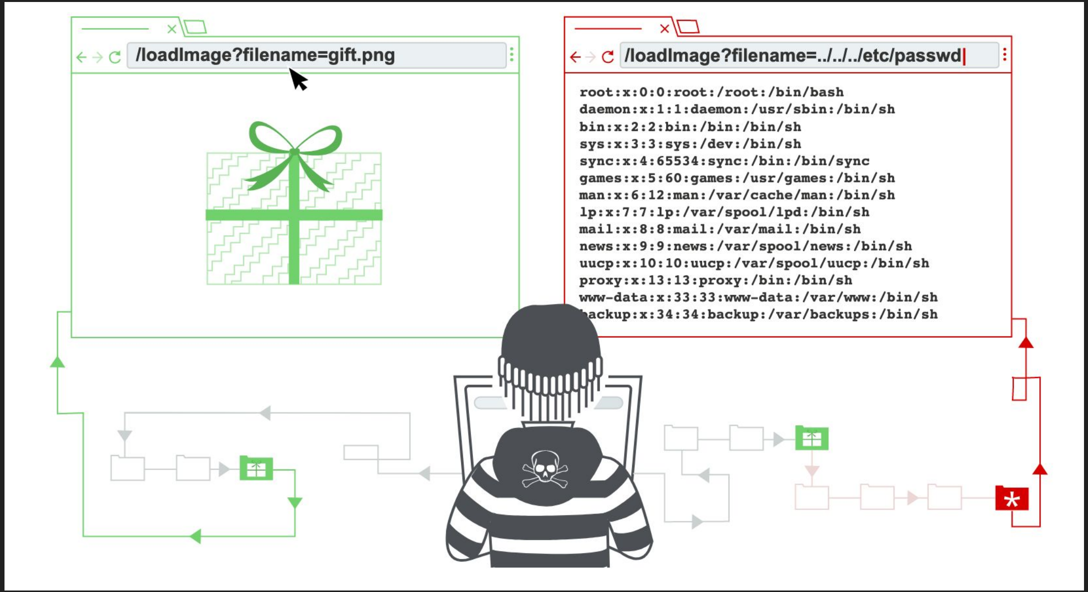
```
janice:~/Desktop/test/test1 andresorbe$ pwd
/Users/andresorbe/Desktop/test/test1
janice:~/Desktop/test/test1 andresorbe$ cd ..
janice:~/Desktop/test andresorbe$ cd .
janice:~/Desktop/test andresorbe$ cd test1/
janice:~/Desktop/test/test1 andresorbe$ pwd
/Users/andresorbe/Desktop/test/test1
janice:~/Desktop/test/test1 andresorbe$
```

Directory Traversal


/
root
/etc
/home
/progetti
/www_root
shadow
passwd
pagine_web
Immissione dei caratteri . . / nel path per raggiungere la cartella root

Directory Traversal

How would one exploit this? (“For Learning Purposes”)

→ 
/loadImage?filename=gift.png
/loadImage?filename=../../../etc/passwd

```
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
```

+ Alternatively sometimes you may be able to do it solely via url

Terms for
Session Hijacking / Session Spoofing

Cookies → Used to store information. Stored on the client-side

Sessions → Used to store information. Stored both on client- and server-side

Hijacking / Spoofing → pretending that we are someone else.
(Simply) If we were on the same network, I may edit TCP packets to have your IP address.

+ Both Hijacking / Spoofing are used interchangeably.
+ Session Hijacking / Cookie Hijacking are also used interchangeably. But it is the default to just say session hijacking.

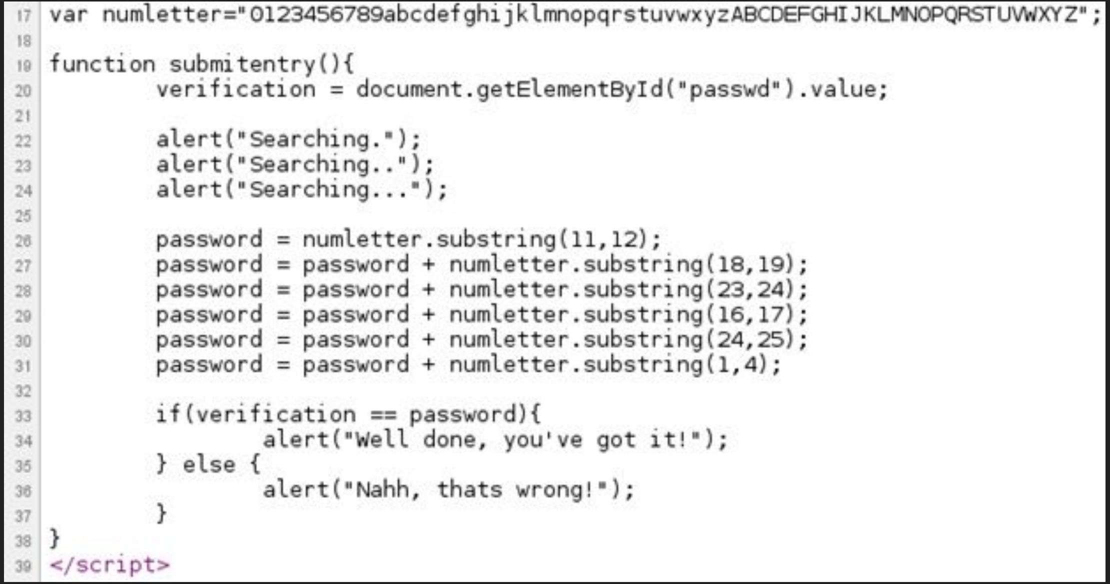
Session Hijacking / Session Spoofing

How would one exploit this?
→ Using packet sniffers (i.e. wireshark)
→ Run XSS attack and steal
→ Just Edit the cookies

How to prevent this?
→ Encrypt packets
→ Force to use HTTPS, Regenerate Session ID frequently, Cryptographically secure random number of strings as the session key

Client-Side Authentication

When Authentication checks are ran completely on the website/client-side. Inherently, it is extremely insecure because you can just read the password.



```
var numletter="0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

function submitentry(){
        verification = document.getElementById("passwd").value;

        alert("Searching.");
        alert("Searching..");
        alert("Searching...");

        password = numletter.substring(11,12);
        password = password + numletter.substring(18,19);
        password = password + numletter.substring(23,24);
        password = password + numletter.substring(16,17);
        password = password + numletter.substring(24,25);
        password = password + numletter.substring(1,4);

        if(verification == password){
                alert("Well done, you've got it!");
        } else {
                alert("Nahh, thats wrong!");
        }
}
</script>
```

It's harder to read, but it's not impossible to figure out.

← BTW This is (Security through Obscurity)

Plain-text Passwords

What is easier to read? (Specifically for the attacker)

```
password

5d41402abc4b2a76b9719d911017c592    (hash of password in md5)
```

(Not an example of “Security through Obscurity” because no one should be able to get access to password or credentials)

→ Hash of string

XSS (Cross-Site Scripting)

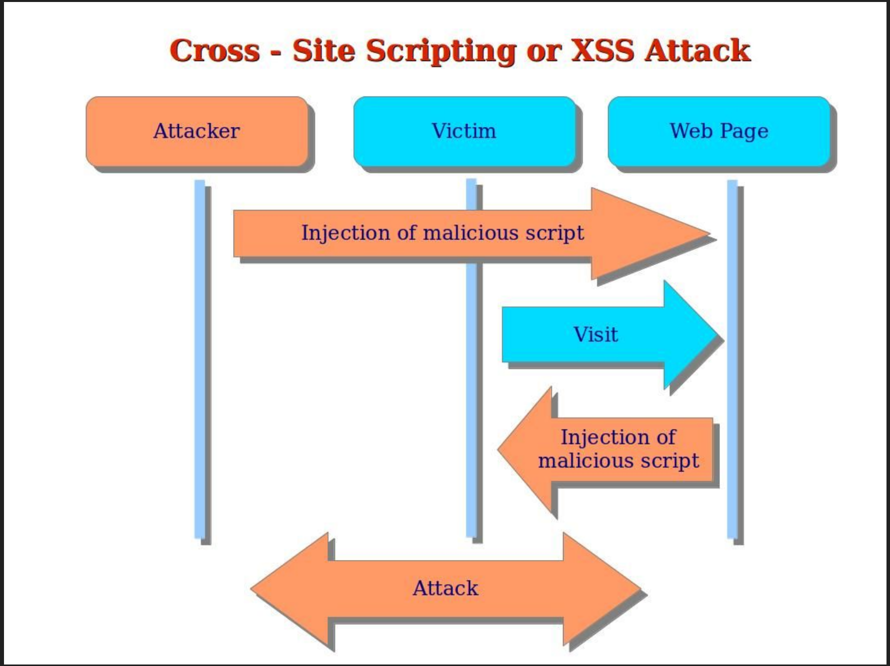
Type of security vulnerability that allows the attacker to inject client-side scripts into web pages viewed by other users

How to exploit?

→ (Simple but goodie)

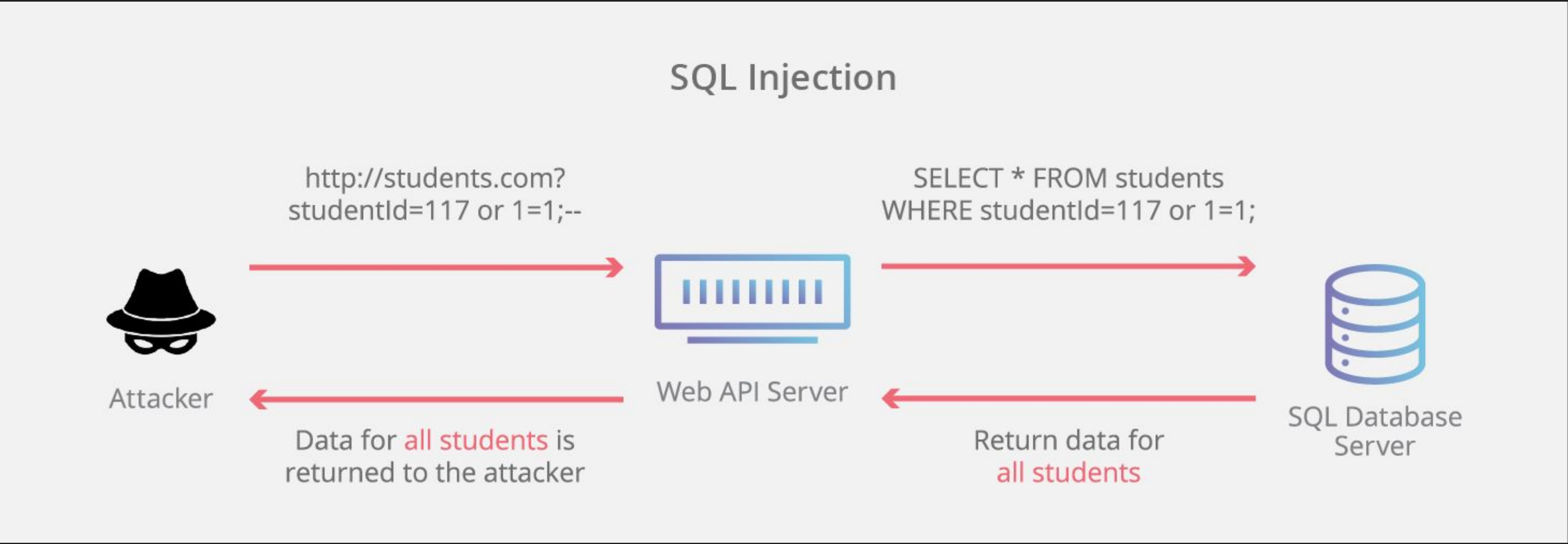
```
<script>alert(“Sup”);</script>
```

→ There are far more advanced ones


Cross - Site Scripting or XSS Attack
Attacker
Victim
Web Page
Injection of malicious script
Visit
Injection of malicious script
Attack

SQL Injection

An attack where malicious SQL statements are inserted into an entry field for execution.



SQL Injection

How to prevent?

→ “Sanitize your inputs”

~ Ensure that you were looking for that specific data. It doesn’t get anything ‘funky’

How to exploit?

→ Matter of exploiting logic. (Look in slide before)

~ Normally looks something like: { ‘ or 1=1-- }

```sql
SELECT *
  FROM users
  WHERE email = '<email>'
  AND pass  = '<pass>'
```

Example:

Normal Code	Exploited
`SELECT *`	`SELECT *`
`  FROM users`	`  FROM users`
`  WHERE email = 'user@email.com'`	`  WHERE email = 'user@email.com'`
`  AND pass  = 'password'`	`  AND pass  = '' or 1=1--'`

Sources

- Exploring Client Side Web Exploits
- Client-Side Authentication