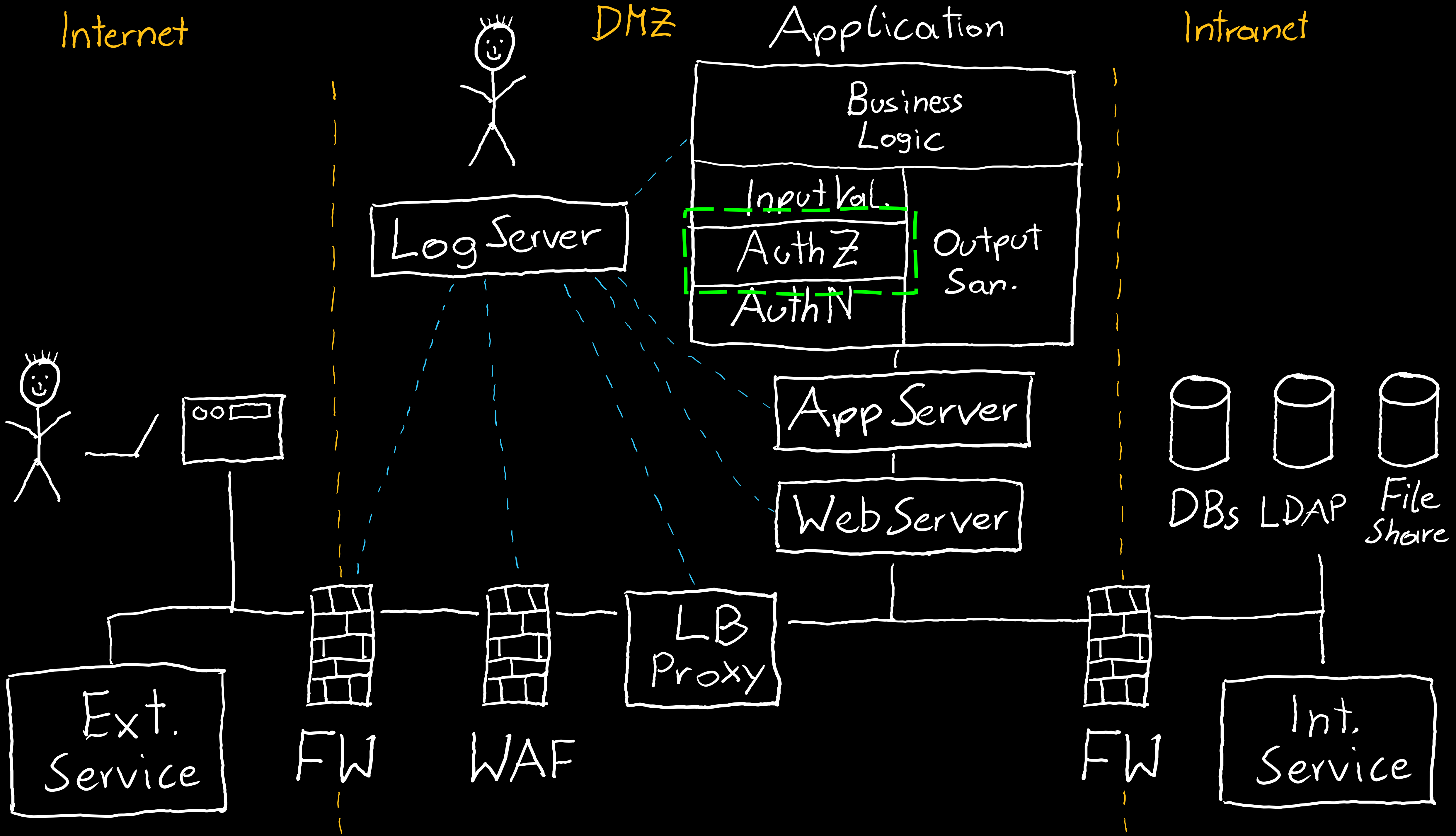
Authorization

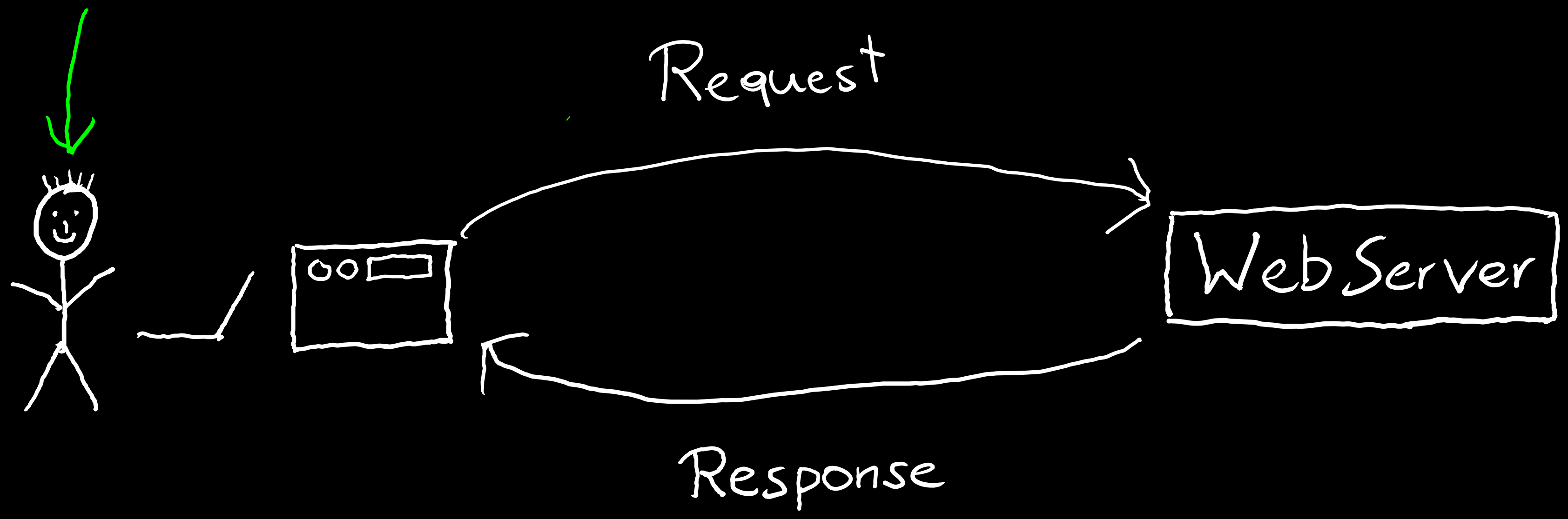
Forceful Browsing, Path Traversal, TOCTOU

Rough Overview

1. Introduction
2. Basic Principles and Resources
3. Architecture & Basic Web Procedure
4. Authentication and Session Management
5. >> Authorization <<
6. Server and Backend Attacks
7. Remaining Client Attacks
8. General Topics
9. Conclusions


Internet
DMZ
Application
Intranet
Business Logic
Input Val.
AuthZ
AuthN
Output San.
Log Server
App Server
Web Server
DBs
LDAP
File Share
Ext. Service
FW
WAF
LB Proxy
FW
Int. Service

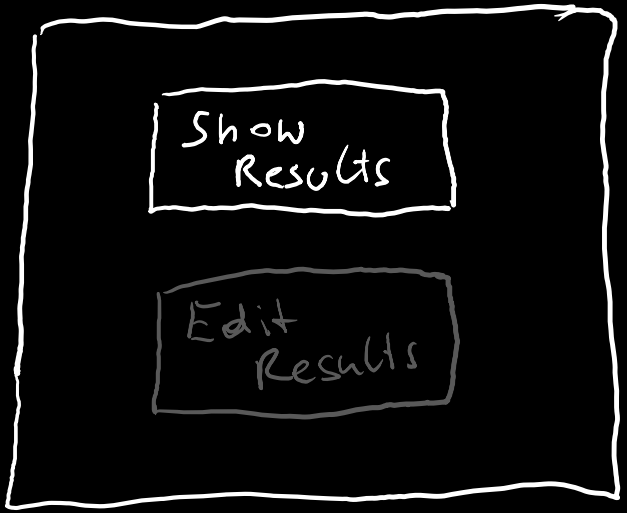
ok, we now really know who this guy is…


Request
Web Server
Response

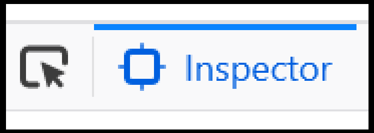
… and we also know how session management works

but what is he actually allowed to do?

what is the first thing ~~you~~ an attacker would do?

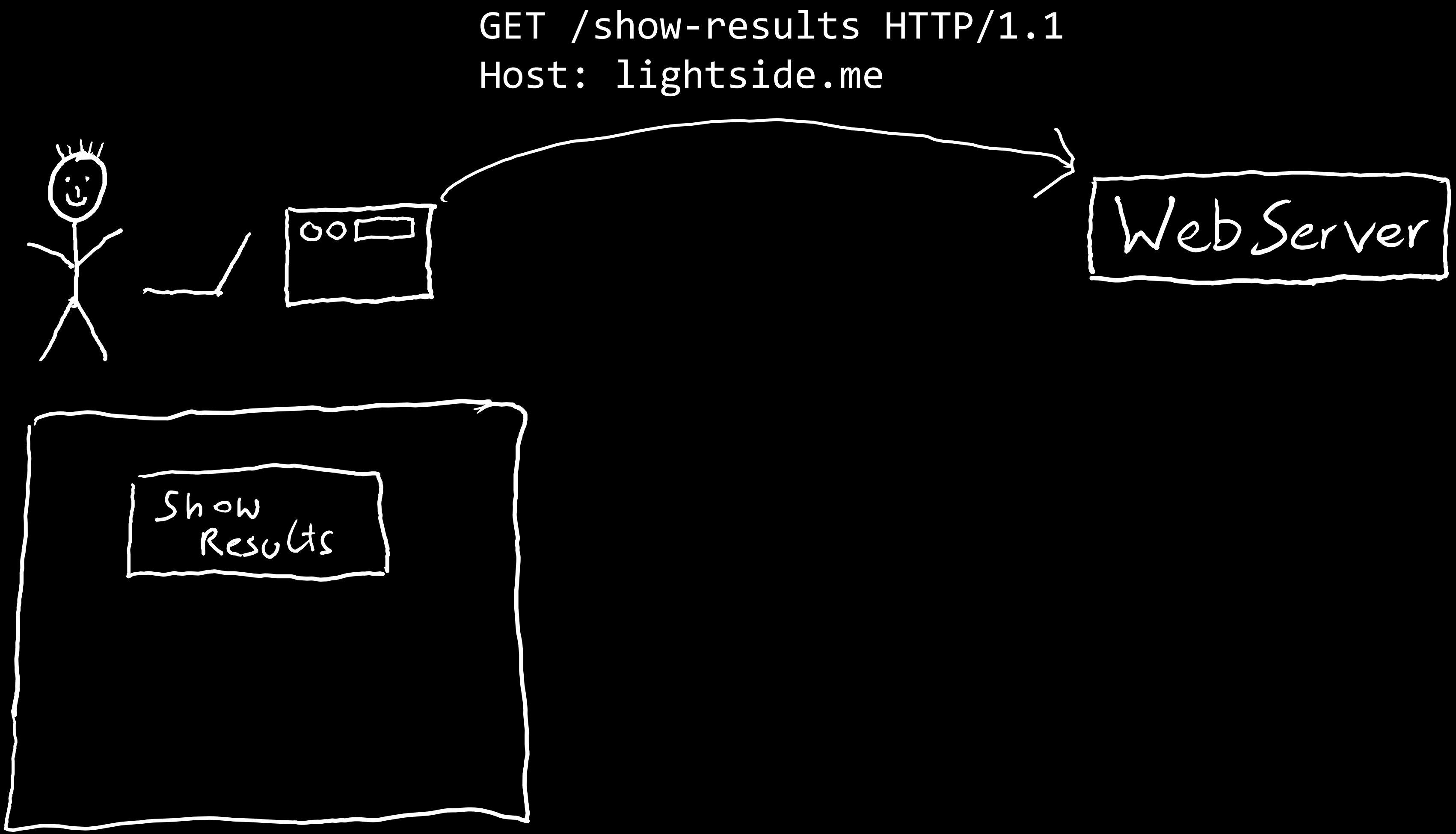

Show Results
Edit Results

```
<button disabled>Edit Results</button>
```



F12

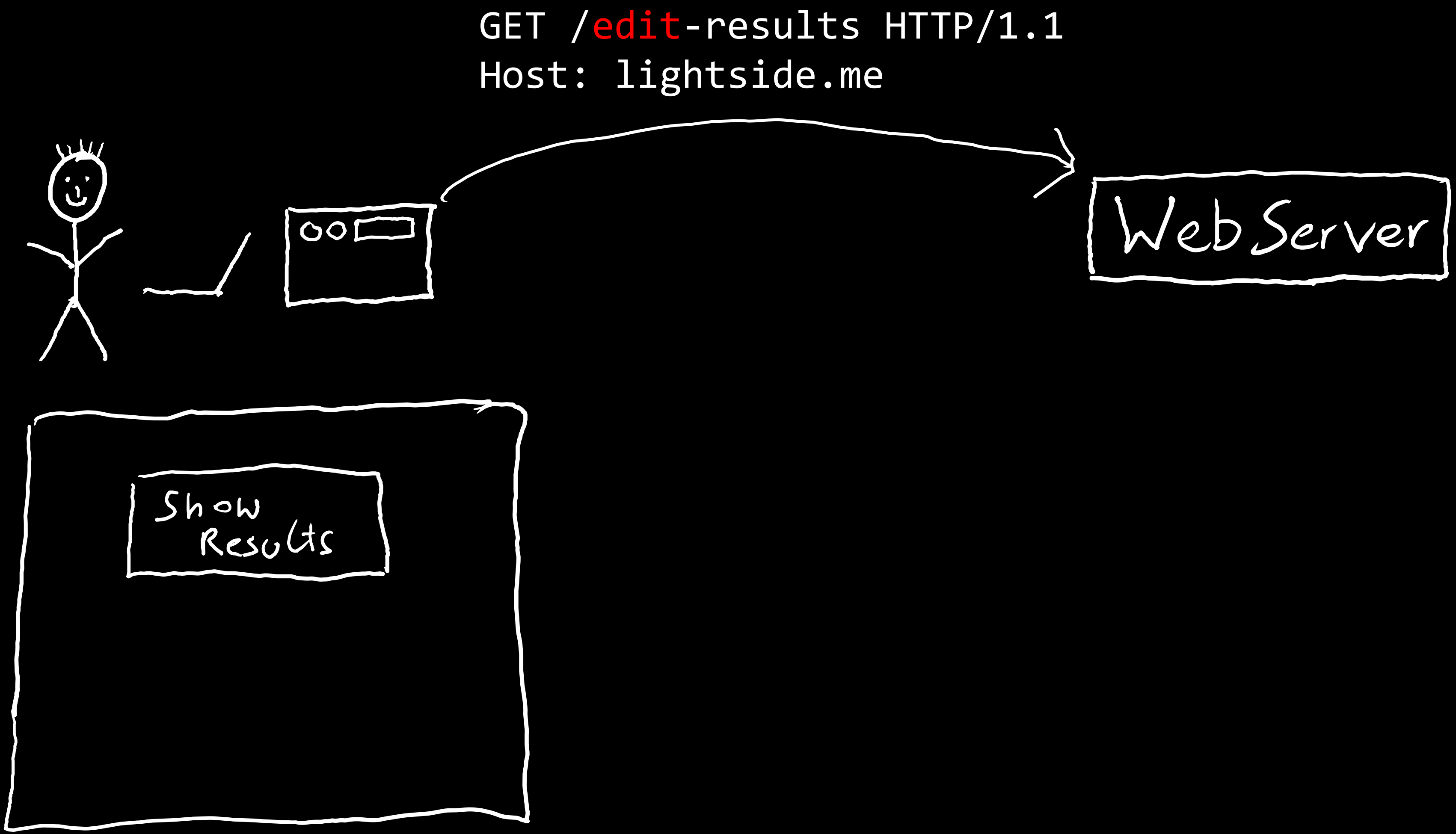
what would an attacker do now?



```
GET /show-results HTTP/1.1
Host: lightside.me
```

Web Server
Show Results

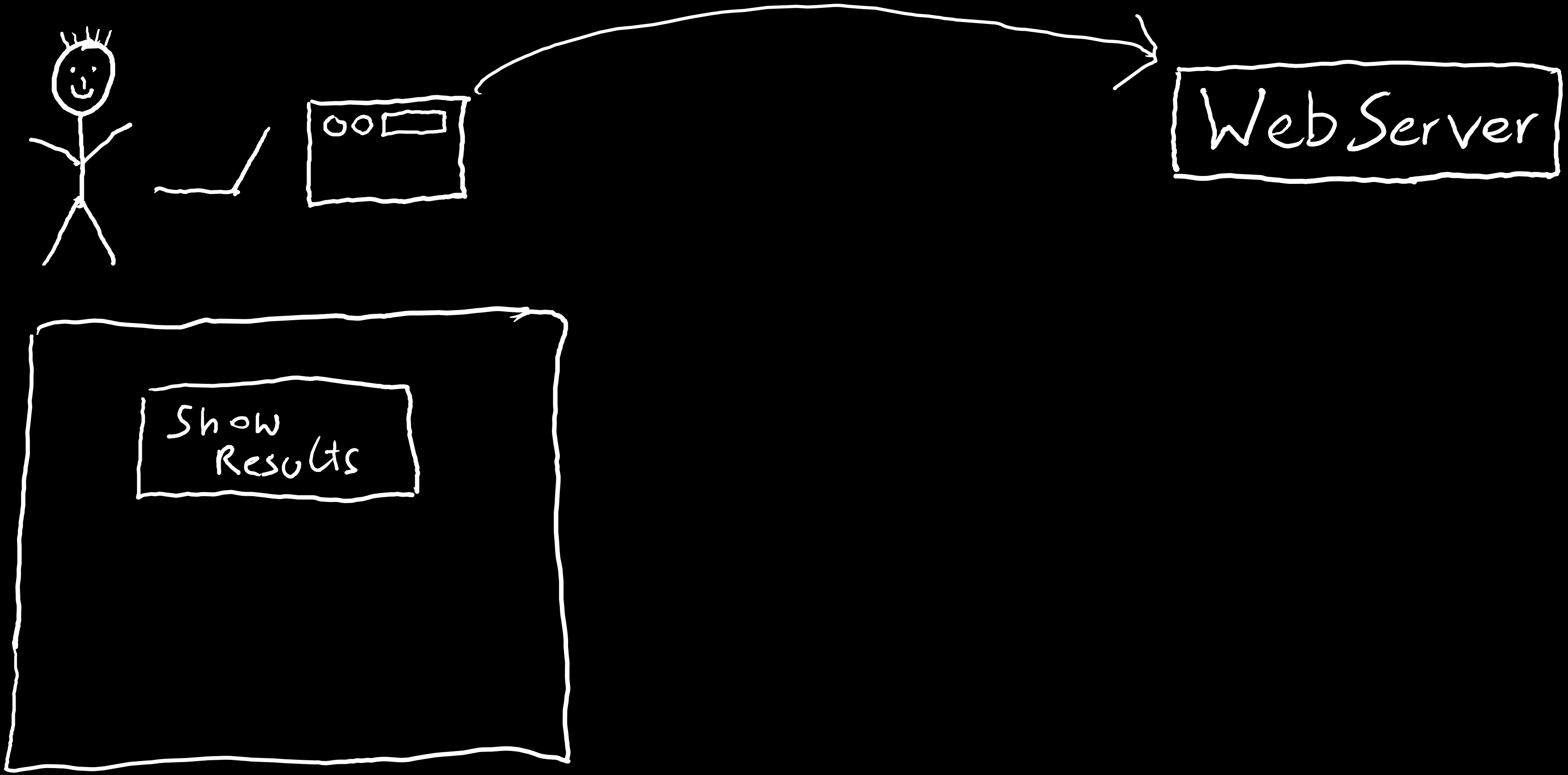
what would an attacker do now?



```
GET /edit-results HTTP/1.1
Host: lightside.me
```

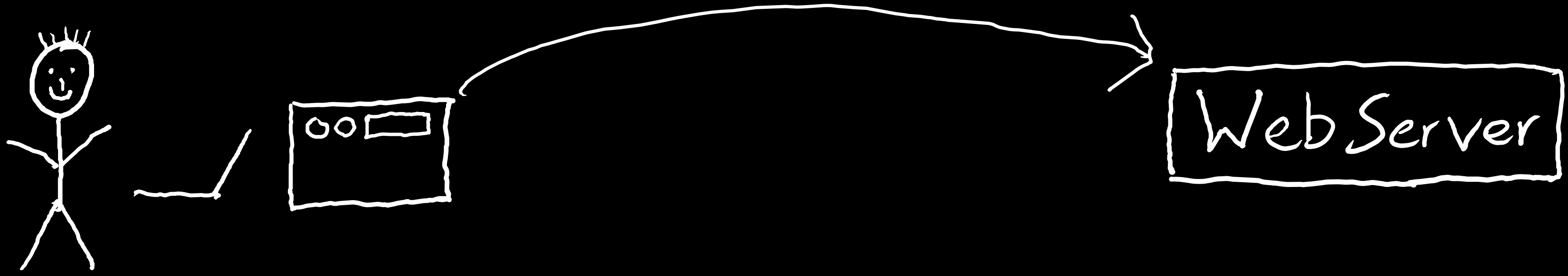
Web Server
Show Results

what would an attacker do now?



```
POST /edit-results HTTP/1.1
Host: lightside.me
```

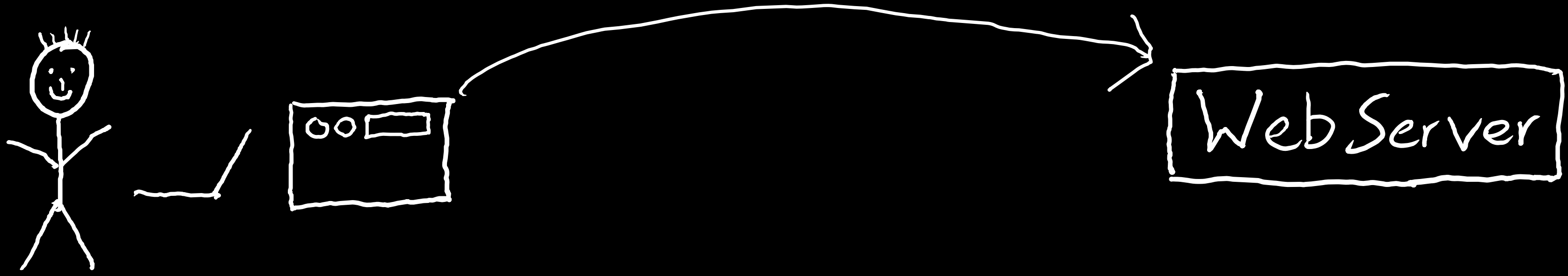
Web Server
Show Results



```
GET /invoice?id=1337 HTTP/1.1
Host: lightside.me
```

what would an attacker do?

```
GET /invoice?id=1338 HTTP/1.1
Host: lightside.me
```



Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	
OWASP Top 10	
(Primary) Violated Principle	

client-side authorization checks yay or nay?

- not relevant for security
  - security needs server side authorization checks

- as an addition good for
  - usability – faster feedback
  - alerting
    - if server-side checks are mirrored at the frontend, every negative authorization decision at the backend is a strong indicator for an attack

Forceful Browsing

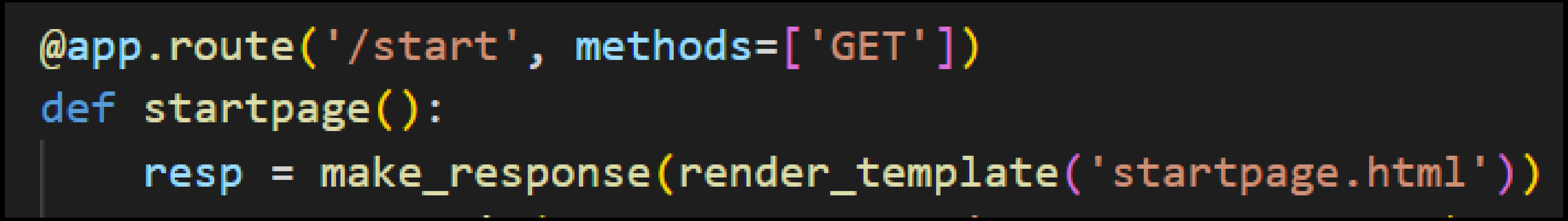
Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend
OWASP Top 10	
(Primary) Violated Principle	

Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user’s authorization for function/resource on server-side
OWASP Top 10	
(Primary) Violated Principle	

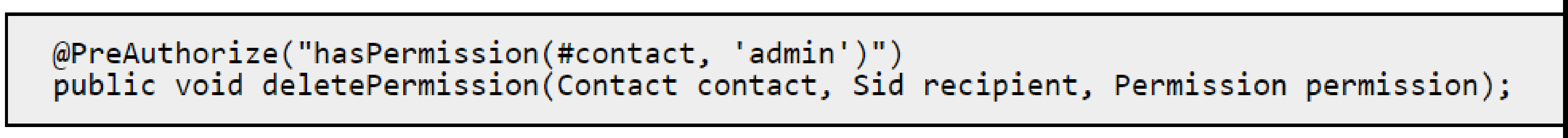
Access Control Levels

- define allowed http method for every url/function



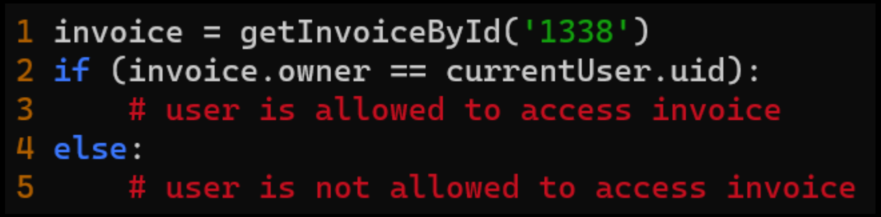
```
@app.route('/start', methods=['GET'])
def startpage():
    resp = make_response(render_template('startpage.html'))
```

- check if user is allowed to
  - call url/function
  - interact with the object/resource



```
@PreAuthorize("hasPermission(#contact, 'admin')")
public void deletePermission(Contact contact, Sid recipient, Permission permission);
```

https://docs.spring.io/spring-security/site/docs/3.0.x/reference/el-access.html



```
invoice = getInvoiceById('1338')
if (invoice.owner == currentUser.uid):
    # user is allowed to access invoice
else:
    # user is not allowed to access invoice
```

Centralization for better auditability

try to centralize authorization decisions

- own class / library in your own code
- dedicated authorization service
  - e.g. https://www.openpolicyagent.org/

Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user’s authorization for function/resource on server-side
OWASP Top 10	
(Primary) Violated Principle	

Forceful Browsing

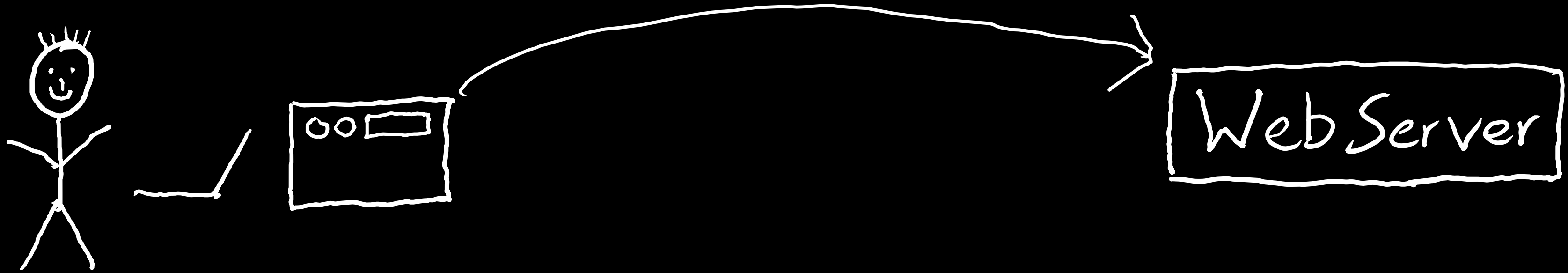
Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user’s authorization for function/resource on server-side use unguessable IDs (e.g. UUIDs) as 2nd line of defense
OWASP Top 10	
(Primary) Violated Principle	

Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user’s authorization for function/resource on server-side use unguessable IDs (e.g. UUIDs) as 2nd line of defense
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	

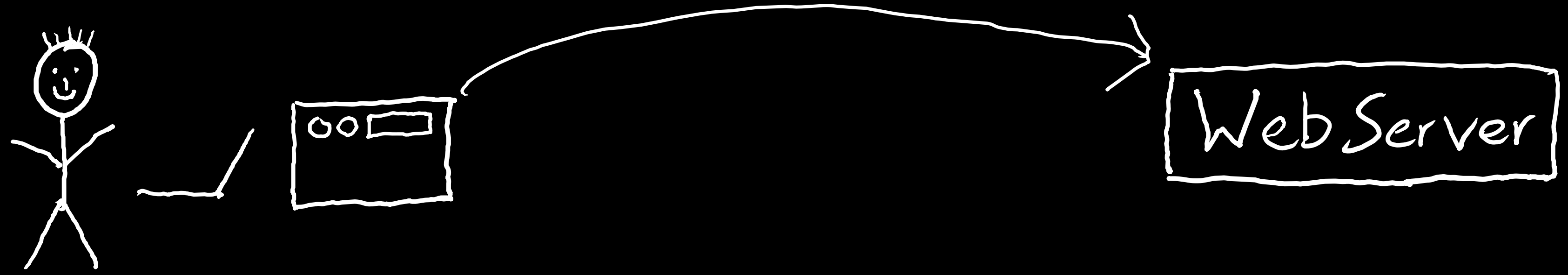
Forceful Browsing

Goal	Directly call a URL / function which you‘re not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user’s authorization for function/resource on server-side use unguessable IDs (e.g. UUIDs) as 2nd line of defense
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	„Earn or give, but never assume, trust.“ „Authorize after you authenticate“



```
GET /invoice?f=inv_202010_luke.pdf HTTP/1.1
Host: lightside.me
```

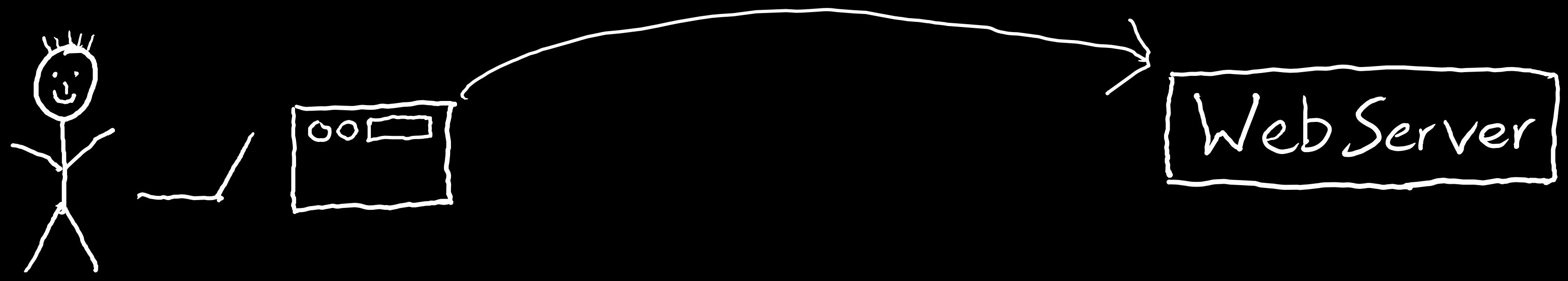
what would an attacker do?

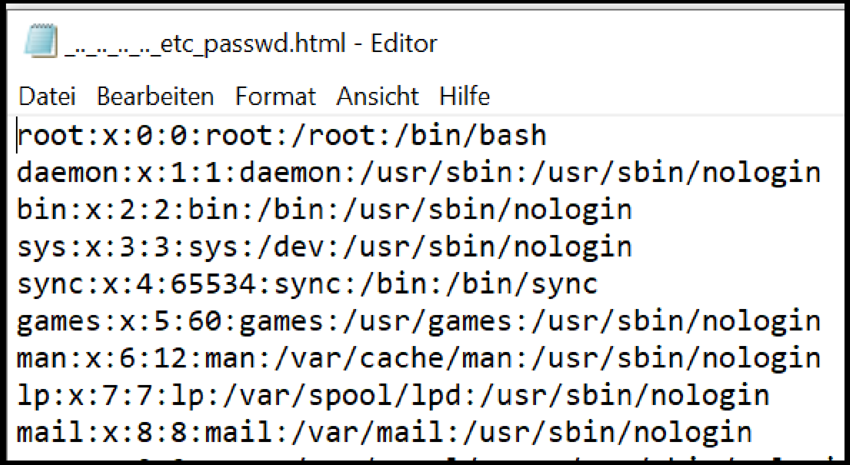


```
GET /invoice?f=inv_202010_lea.pdf HTTP/1.1
Host: lightside.me
```

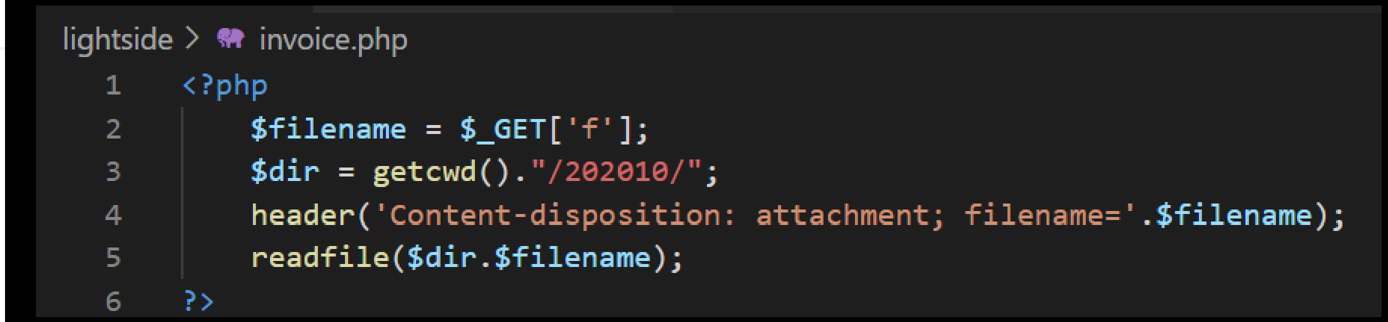
of course... can you think of something else?

```
GET /invoice?f=../../../../../etc/passwd HTTP/1.1
Host: lightside.me
```




_.._.._.._.._etc_passwd.html - Editor
Datei Bearbeiten Format Ansicht Hilfe

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```


lightside > invoice.php

```php
<?php
    $filename = $_GET['f'];
    $dir = getcwd()."/202010/";
    header('Content-disposition: attachment; filename='.$filename);
    readfile($dir.$filename);
?>
```

Path Traversal

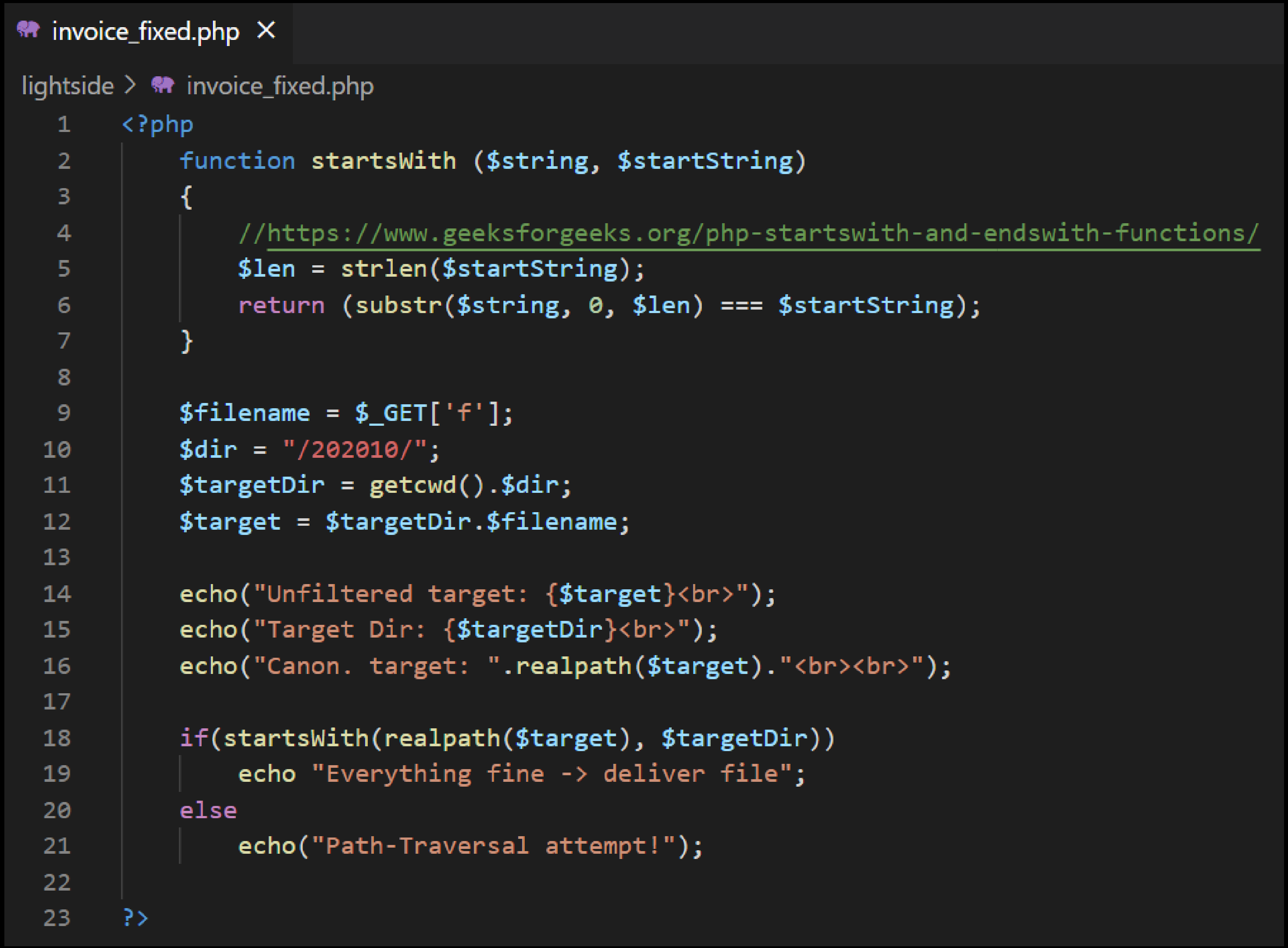
Goal	Directly access files (outside the webroot) which you‘re not supposed to by the application
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Path Traversal

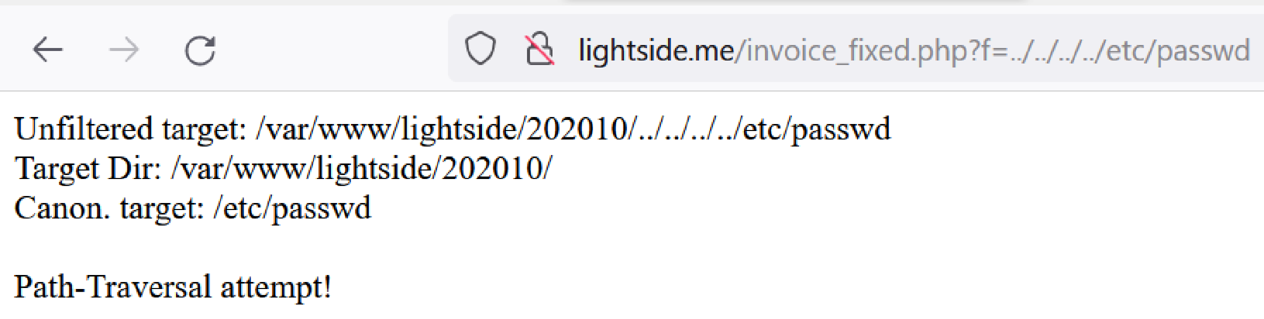
Goal	Directly access files (outside the webroot) which you‘re not supposed to by the application
How	whenever an application handles path’s incorrectly, try to iterate through it
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Path Traversal

Goal	Directly access files (outside the webroot) which you‘re not supposed to by the application
How	whenever an application handles path’s incorrectly, try to iterate through it
Solution	Don’t use user input in filesystem APIs If you have to validate it (explicit allowlist) check the canonicalized path Run webserver under a low privileged user
OWASP Top 10	
(Primary) Violated Principle	

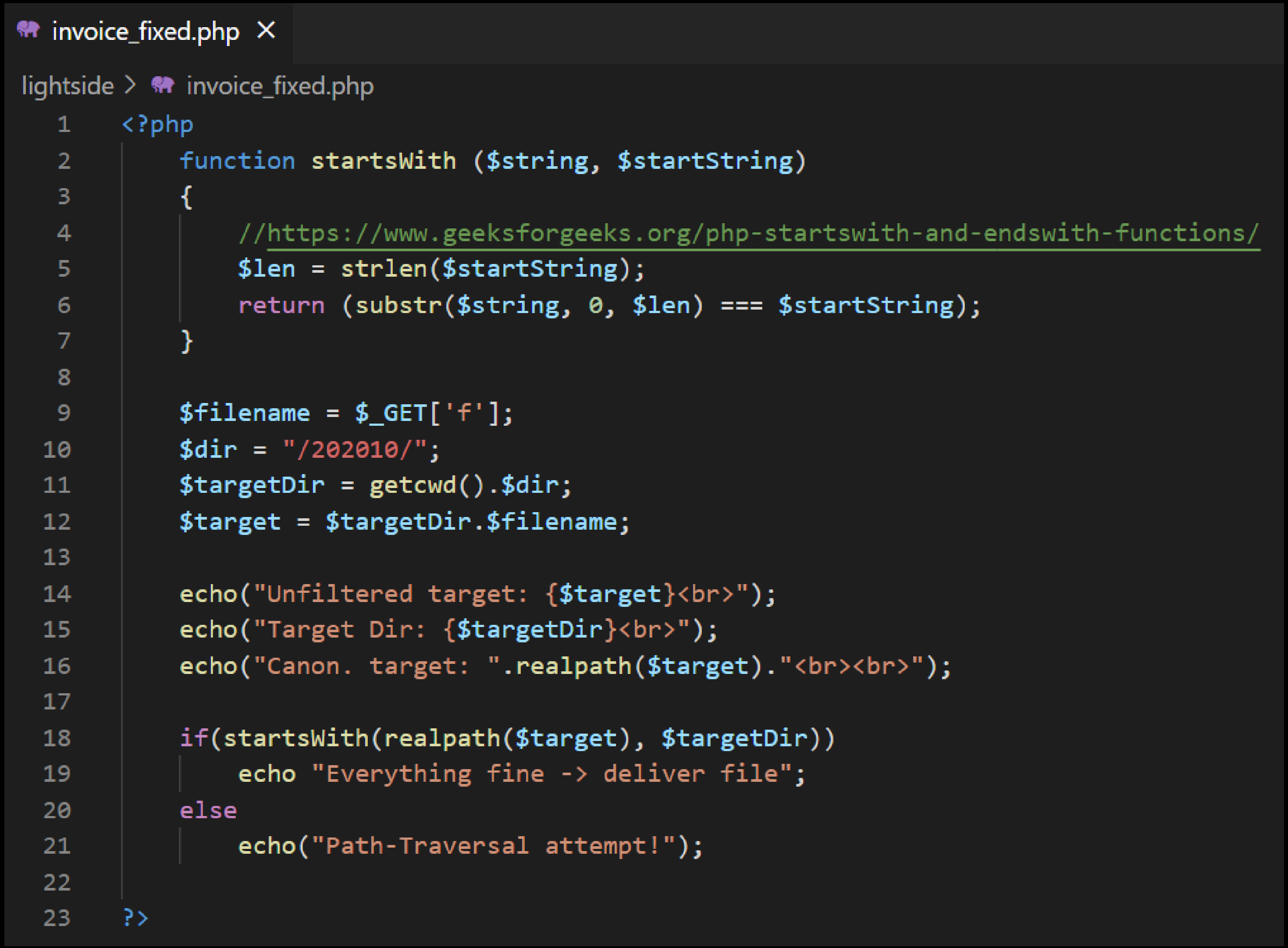


```php
<?php
    function startsWith ($string, $startString)
    {
        //https://www.geeksforgeeks.org/php-startswith-and-endswith-functions/
        $len = strlen($startString);
        return (substr($string, 0, $len) === $startString);
    }

    $filename = $_GET['f'];
    $dir = "/202010/";
    $targetDir = getcwd().$dir;
    $target = $targetDir.$filename;

    echo("Unfiltered target: {$target}<br>");
    echo("Target Dir: {$targetDir}<br>");
    echo("Canon. target: ".realpath($target)."<br><br>");

    if(startsWith(realpath($target), $targetDir))
        echo "Everything fine -> deliver file";
    else
        echo("Path-Traversal attempt!");

?>
```

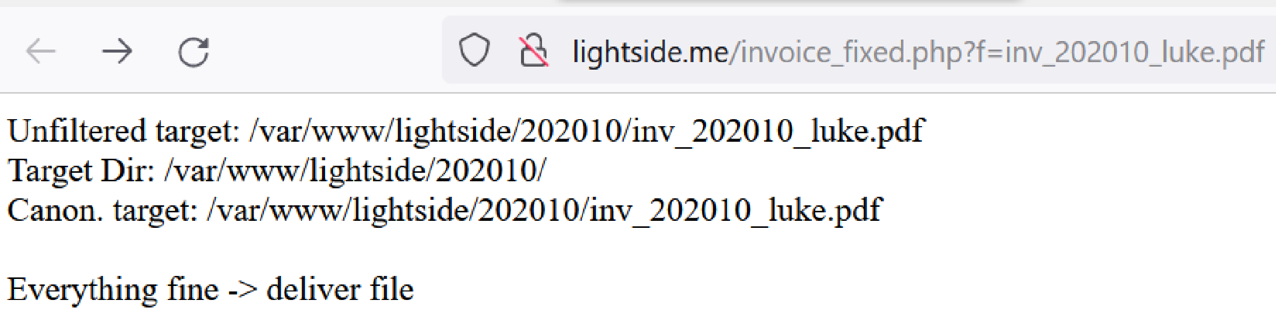


lightside.me/invoice_fixed.php?f=../../../../etc/passwd

```
Unfiltered target: /var/www/lightside/202010/../../../../etc/passwd
Target Dir: /var/www/lightside/202010/
Canon. target: /etc/passwd

Path-Traversal attempt!
```


invoice_fixed.php
lightside > invoice_fixed.php

```php
<?php
    function startsWith ($string, $startString)
    {
        //https://www.geeksforgeeks.org/php-startswith-and-endswith-functions/
        $len = strlen($startString);
        return (substr($string, 0, $len) === $startString);
    }

    $filename = $_GET['f'];
    $dir = "/202010/";
    $targetDir = getcwd().$dir;
    $target = $targetDir.$filename;

    echo("Unfiltered target: {$target}<br>");
    echo("Target Dir: {$targetDir}<br>");
    echo("Canon. target: ".realpath($target)."<br><br>");

    if(startsWith(realpath($target), $targetDir))
        echo "Everything fine -> deliver file";
    else
        echo("Path-Traversal attempt!");

?>
```


lightside.me/invoice_fixed.php?f=inv_202010_luke.pdf

```
Unfiltered target: /var/www/lightside/202010/inv_202010_luke.pdf
Target Dir: /var/www/lightside/202010/
Canon. target: /var/www/lightside/202010/inv_202010_luke.pdf

Everything fine -> deliver file
```

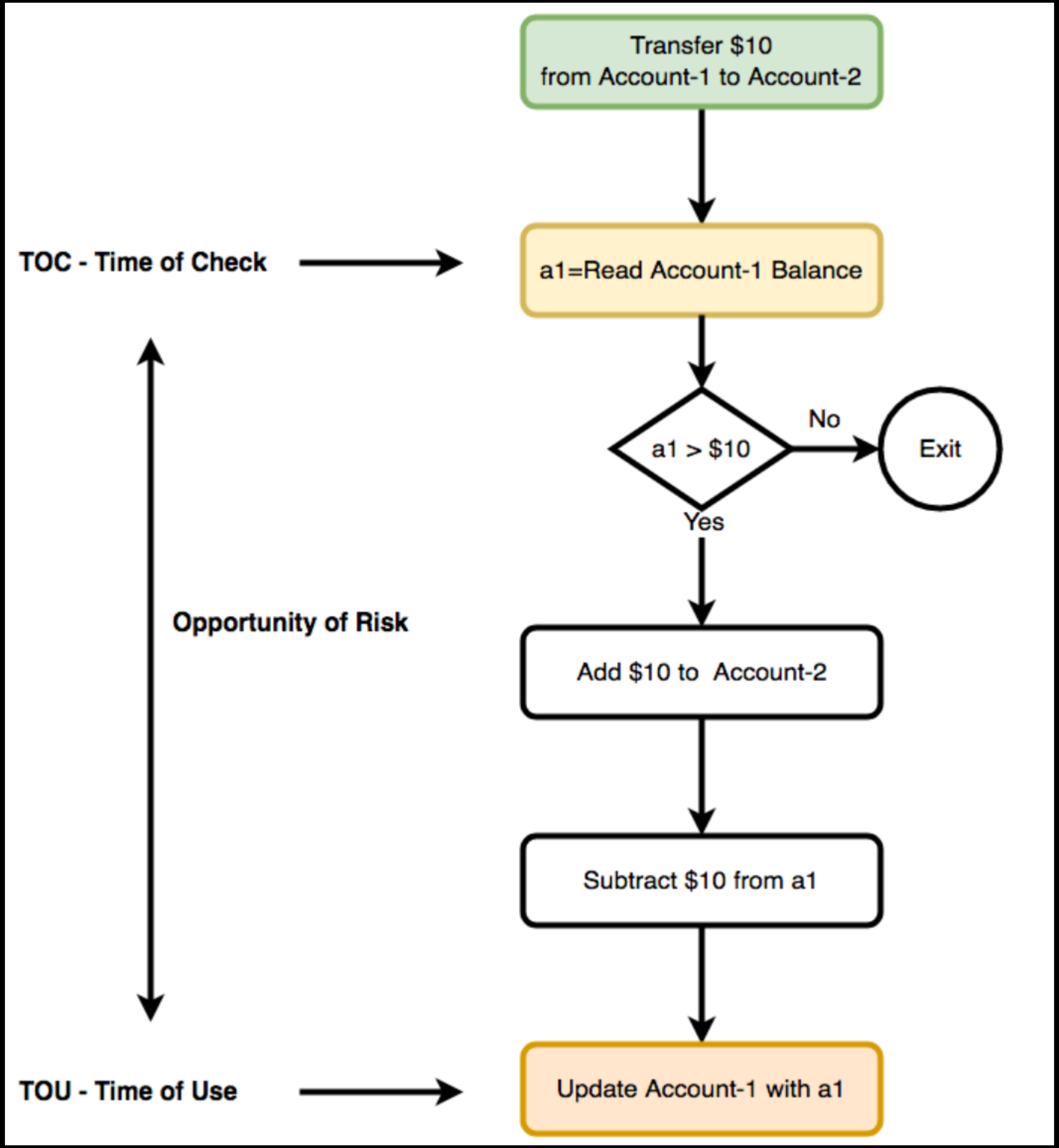
Path Traversal

Goal	Directly access files (outside the webroot) which you‘re not supposed to by the application
How	whenever an application handles path’s incorrectly, try to iterate through it
Solution	Don’t use user input in filesystem APIs If you have to validate it (explicit allowlist) check the canonicalized path Run webserver under a low privileged user
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	

Path Traversal

Goal	Directly access files (outside the webroot) which you‘re not supposed to by the application
How	whenever an application handles path’s incorrectly, try to iterate through it
Solution	Don’t use user input in filesystem APIs If you have to validate it (explicit allowlist) check the canonicalized path Run webserver under a low privileged user
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	“Define an approach that ensures all data are explicitly validated.”

so one last thing about authorization…


Transfer $10
from Account-1 to Account-2
TOC - Time of Check
a1=Read Account-1 Balance
a1 > $10
No
Exit
Yes
Opportunity of Risk
Add $10 to Account-2
Subtract $10 from a1
TOU - Time of Use
Update Account-1 with a1

https://medium.com/@schogini/toctou-time-of-check-and-time-of-use-a-demonstration-and-mitigation-609c999042cb

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	Take advantage of a time-window between a security check and the actual action this check should protect
Solution	
OWASP Top 10	
(Primary) Violated Principle	

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	Take advantage of a time-window between a security check and the actual action this check should protect
Solution	Apply security checks as close to the actual action as possible
OWASP Top 10	
(Primary) Violated Principle	

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	Take advantage of a time-window between a security check and the actual action this check should protect
Solution	Apply security checks as close to the actual action as possible Use locking mechanisms, e.g. locking / semaphores database transactions etc.
OWASP Top 10	
(Primary) Violated Principle	

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	Take advantage of a time-window between a security check and the actual action this check should protect
Solution	Apply security checks as close to the actual action as possible Use locking mechanisms, e.g. locking / semaphores database transactions etc.
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	

TOCTOU (Race Condition)

Goal	Execute an operation you are actually not allowed to
How	Take advantage of a time-window between a security check and the actual action this check should protect
Solution	Apply security checks as close to the actual action as possible Use locking mechanisms, e.g. locking / semaphores database transactions etc.
OWASP Top 10	A01:2021-Broken Access Control
(Primary) Violated Principle	“Authorize after you authenticate”

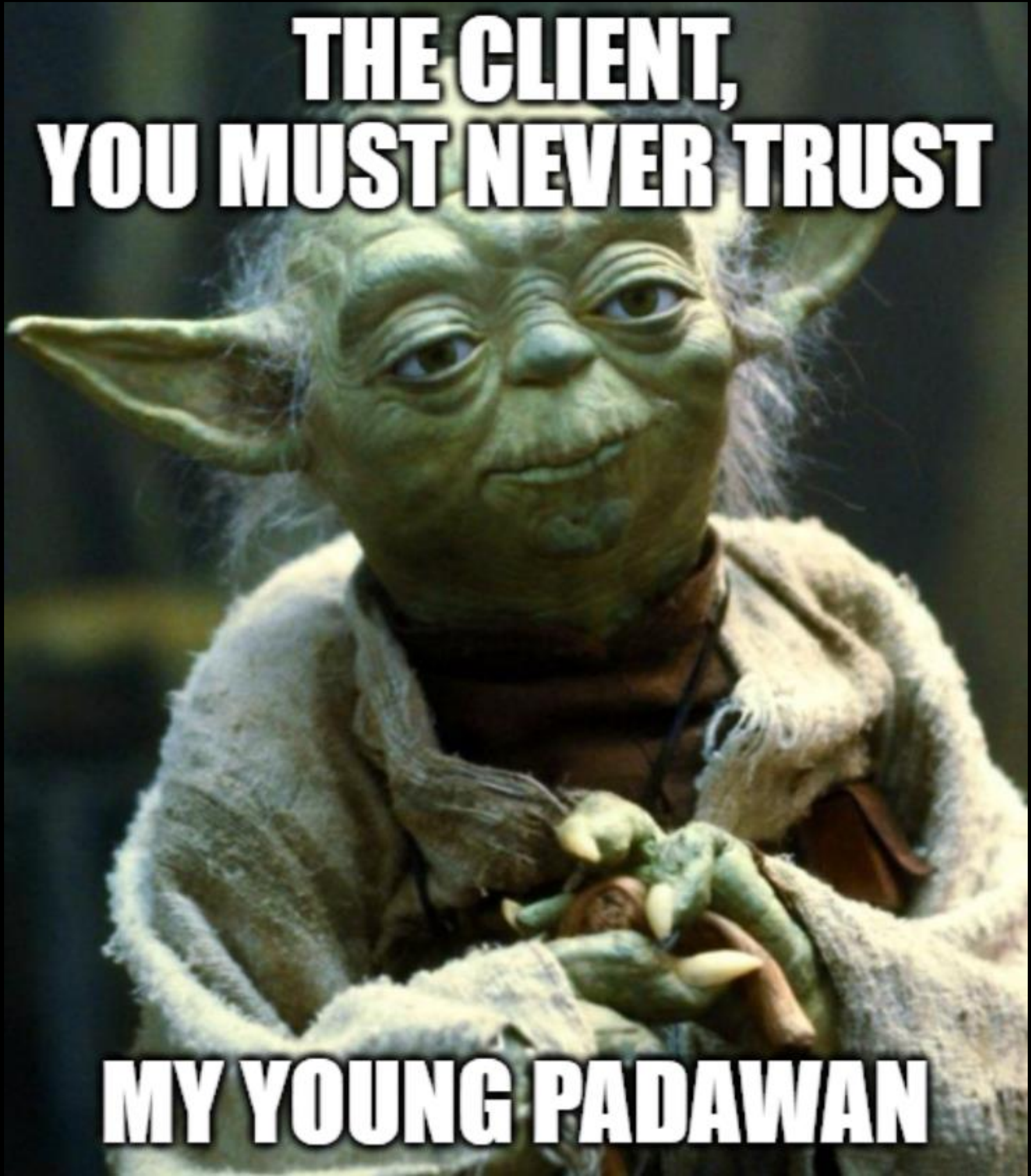
Authorization pitfalls

- Worst: no authorization
- Second: client-side authorization
  - actually, it‘s the same as „no authorization“
- Third: Inconsistent authorization, e.g.:
  - check GET, but not POST
  - check via direct web access but not via SOAP service

Golden Authorization Rules

- Always check everything for authorization
  - every request
  - every function call
  - every resource access
  - every everything

- NEVER ever trust the client!
  - always check authorization on server side


THE CLIENT,
YOU MUST NEVER TRUST
MY YOUNG PADAWAN