

# Authorization

Forceful Browsing, IDOR, Path Traversal, TOCTOU

# Rough Overview

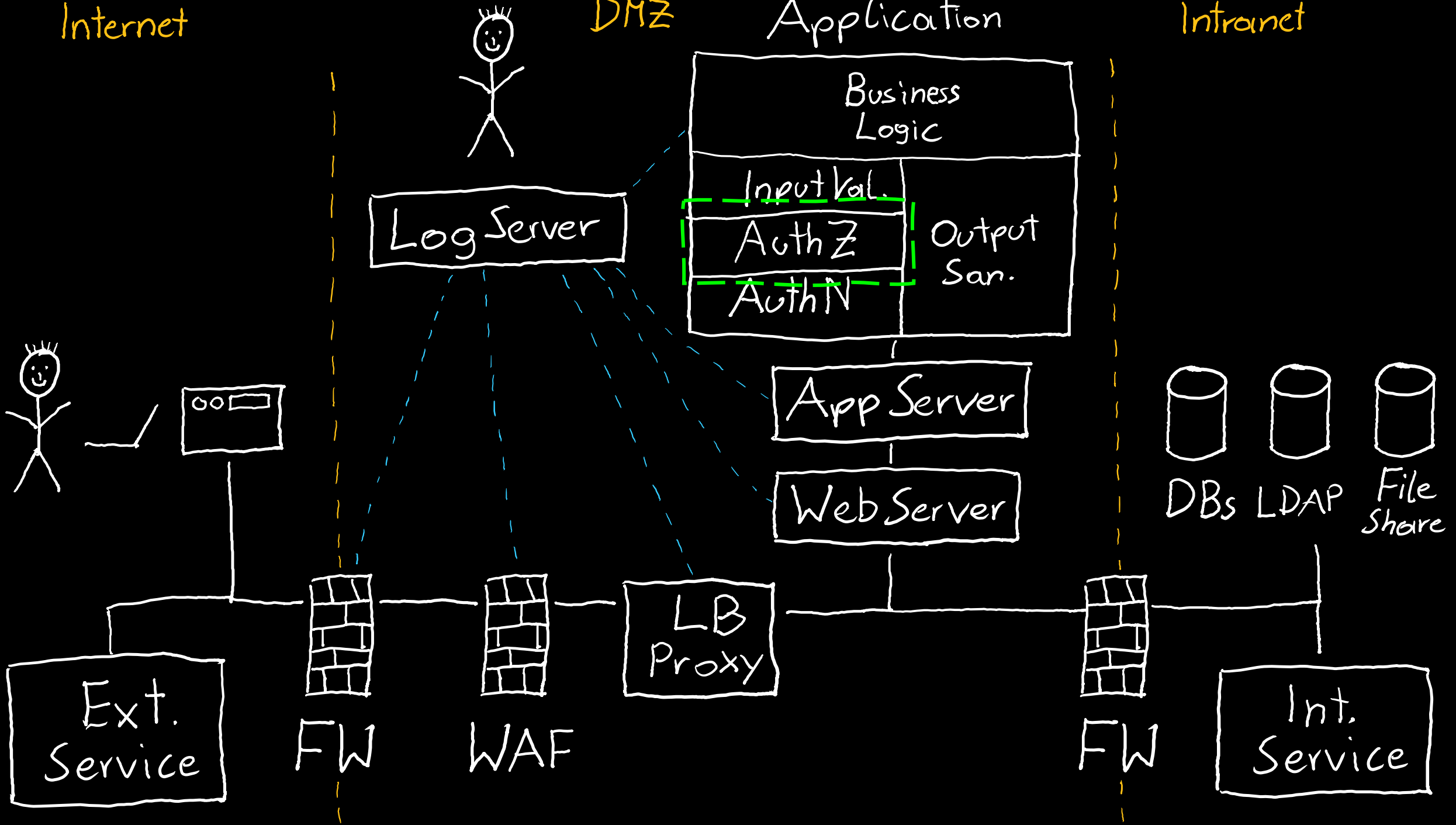
1. Introduction
2. Basic Principles and Resources
3. Architecture & Basic Web Procedure
4. Authentication and Session Management
5. >> Authorization <<
6. Server and Backend Attacks
7. Remaining Client Attacks
8. General Topics
9. Conclusions

Internet

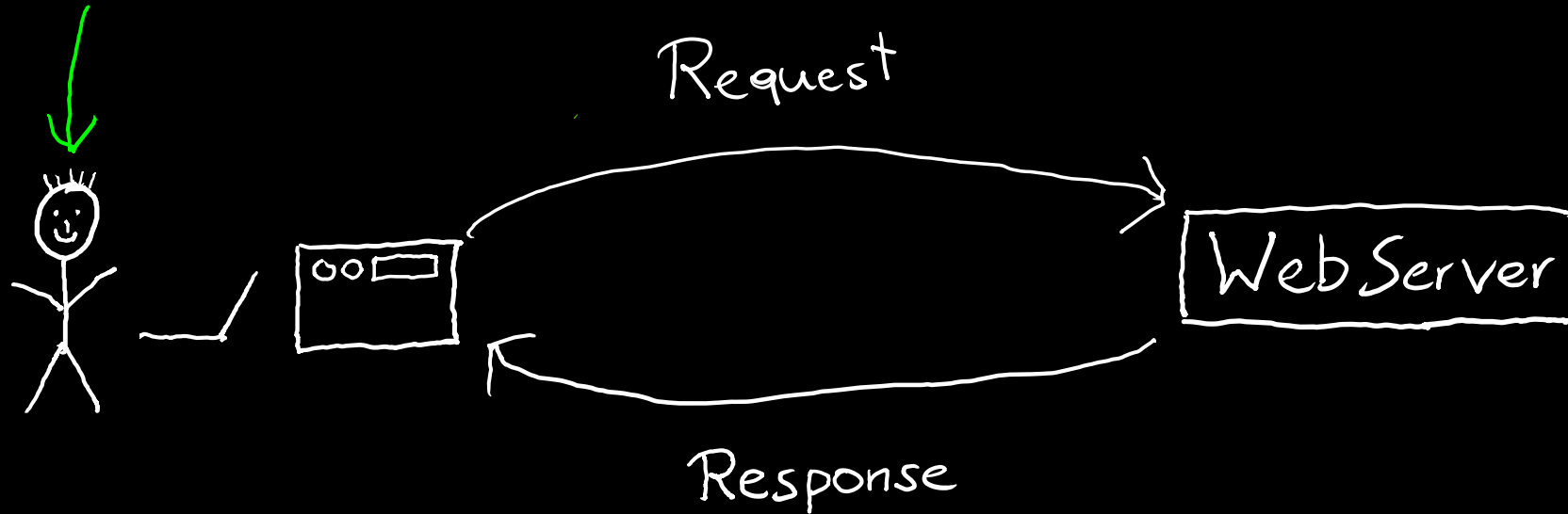
DMZ

Application

Intranet



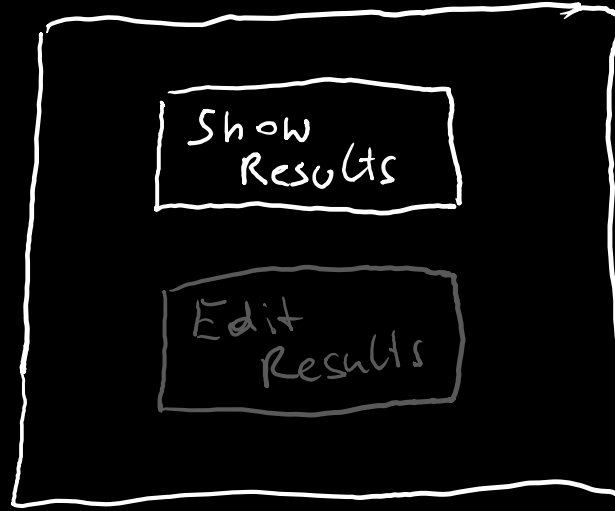
ok, we now really know who this guy is...



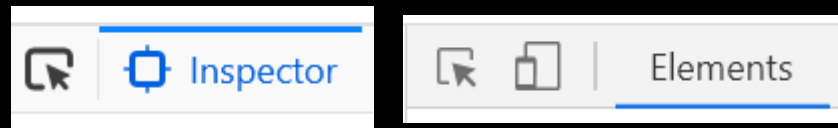
... and we also know how session management works

but what is he actually allowed to do?

what is the first thing you an attacker would do?



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



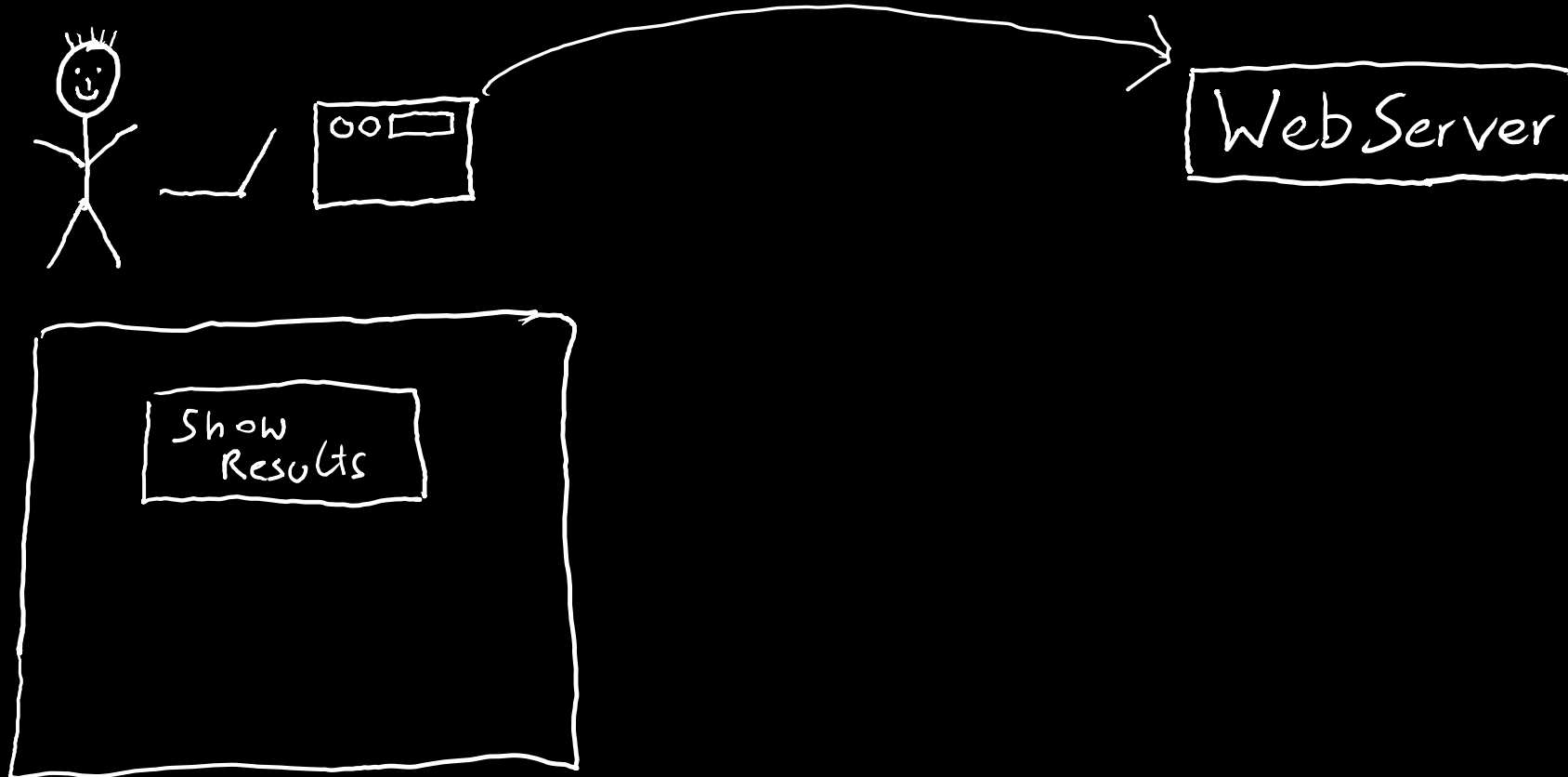
F12

what would an attacker do now?



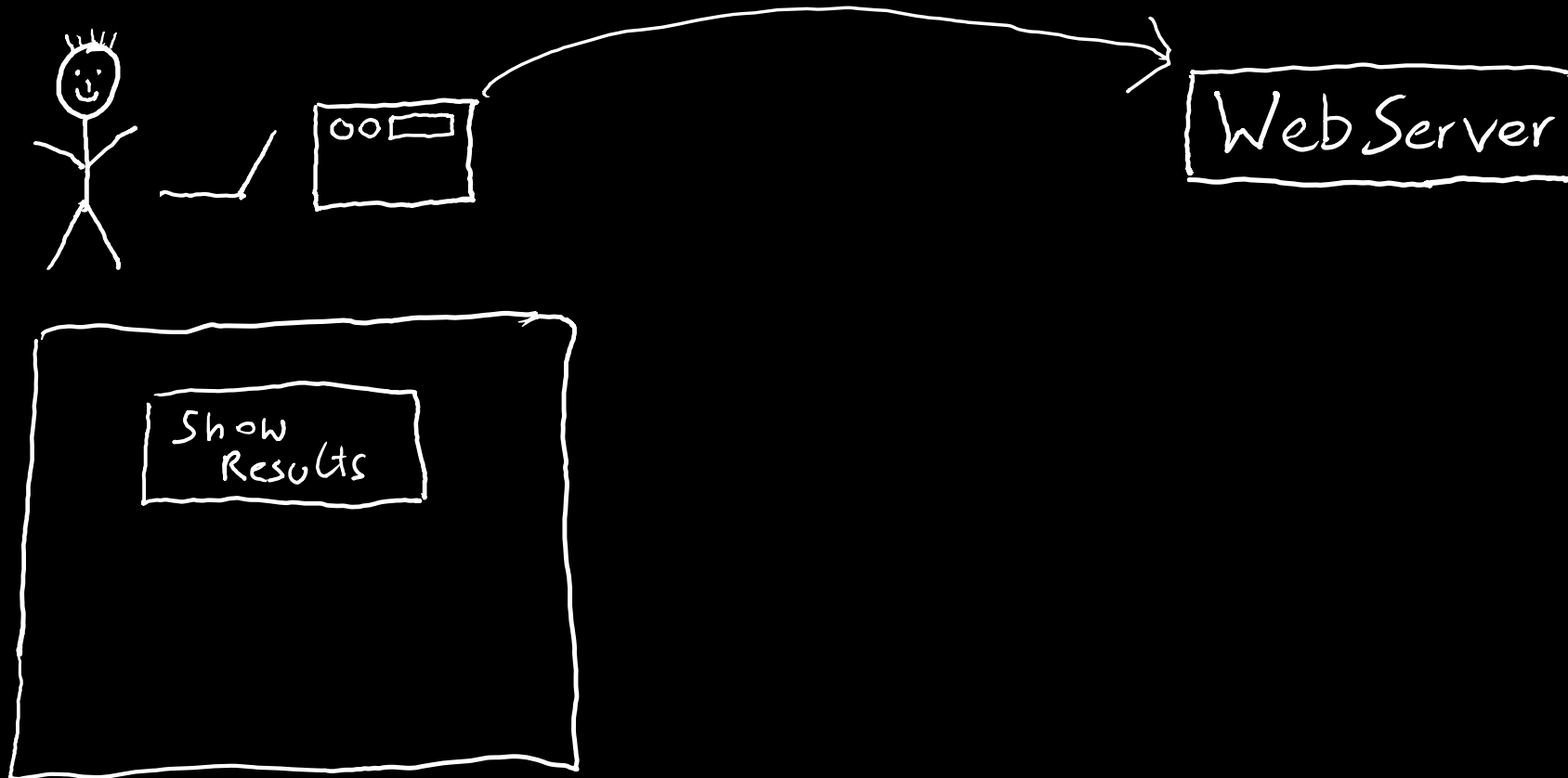
what would an attacker do now?

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



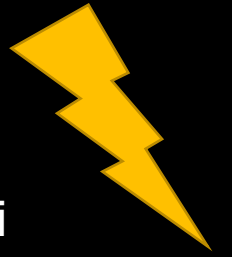
what would an attacker do now?

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





# Forceful Browsing



Goal

Directly call a URL / function which you're not supposed to by the ui

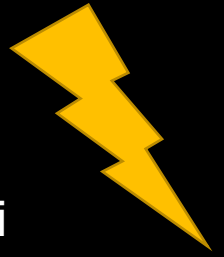
How

Solution

OWASP Top 10

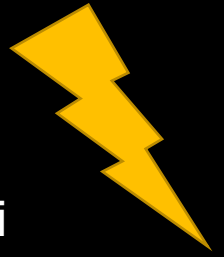
(Primary)  
Violated Principle

# Forceful Browsing



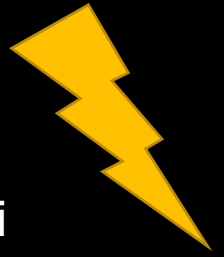
Goal	Directly call a URL / function which you're not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Forceful Browsing



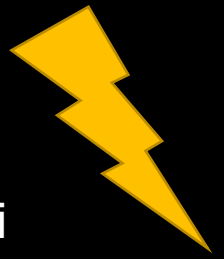
Goal	Directly call a URL / function which you're not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges  hacker instinct / guessing ;-)
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Forceful Browsing



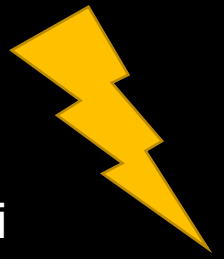
Goal	Directly call a URL / function which you're not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to call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

# Forceful Browsing



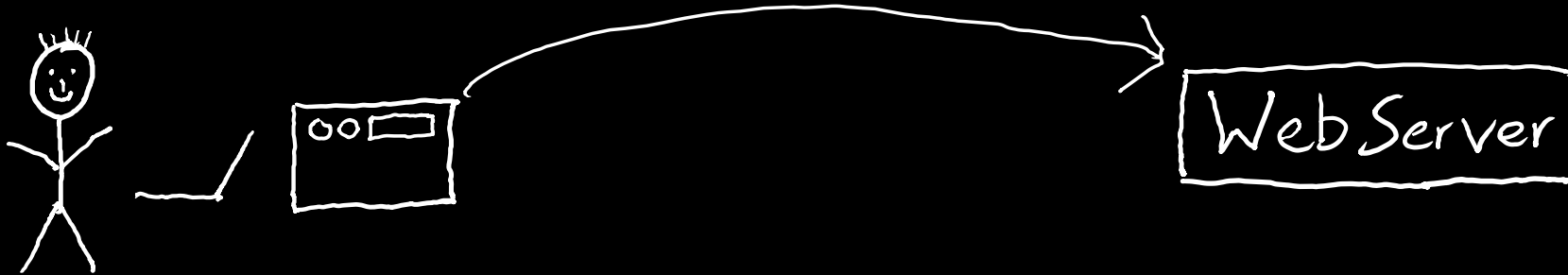
Goal	Directly call a URL / function which you're not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to call function on server-side
OWASP Top 10	A5:2017-Broken Access Control
(Primary) Violated Principle	

# Forceful Browsing



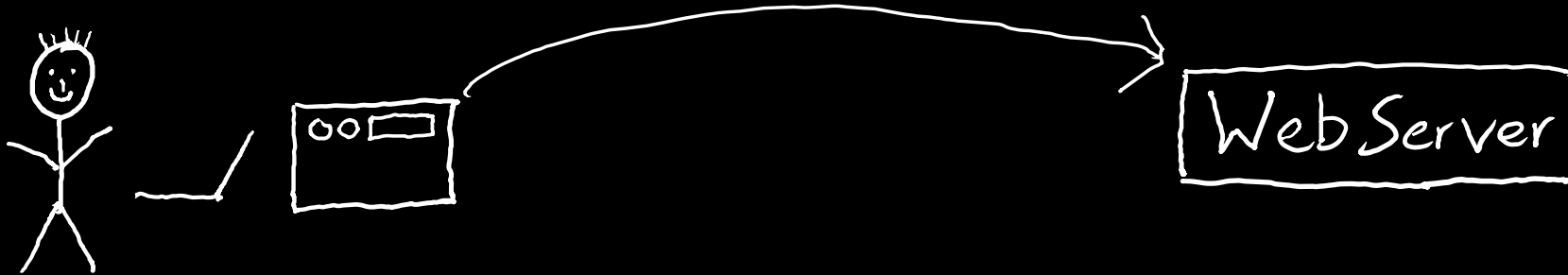
Goal	Directly call a URL / function which you're not supposed to by the ui
How	information gathering by client source code logs error messages robots.txt other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to call function on server-side
OWASP Top 10	A5:2017-Broken Access Control
(Primary) Violated Principle	„Earn or give, but never assume, trust.“ „Authorize after you authenticate“

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



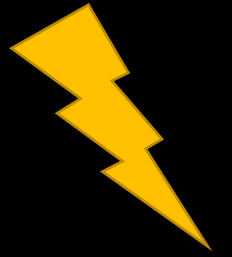
what would an attacker do?

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





# Insec. Direct Object References



Goal

Directly access a resource which you're not supposed to by the ui

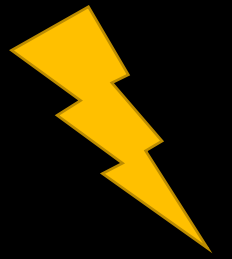
How

Solution

OWASP Top 10

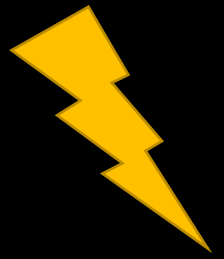
(Primary)  
Violated Principle

# Insec. Direct Object References



Goal	Directly access a resource which you're not supposed to by the ui
How	information gathering by http parameters client source code logs error messages other users with higher privileges  hacker instinct / guessing ;-)
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Insec. Direct Object References



Goal	Directly access a resource which you're not supposed to by the ui
How	information gathering by http parameters client source code logs error messages other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to access resources on server-side
OWASP Top 10	
(Primary) Violated Principle	

# Direct vs. Indirect Object Reference

ID	Object	Owner
1336	inv_202010_luke.pdf	Luke
1337	inv_202010_lea.pdf	Lea
1338	inv_202011_luke.pdf	Luke
1339	inv_202011_lea.pdf	Lea
1340	inv_202011_vader.pdf	Vader

Session	RID	ID	Object
Luke	1	1336	inv_202010_luke.pdf
	2	1338	inv_202011_luke.pdf
Session	RID	ID	Object
Lea	1	1337	inv_202010_lea.pdf
	2	1339	inv_202011_lea.pdf
Session	RID	ID	Object
Vader	1	1340	inv_202011_vader.pdf

# Using random values (e.g. UUID)

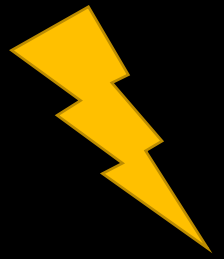
ID	Object	Owner
14b7bdea-e58d-46ec-bfd7-e175022f32ce	inv_202010_luke.pdf	Luke
b113479b-46ad-42e6-a3f3-0d5f80847d38	inv_202010_lea.pdf	Lea
11dfa49e-f585-4e93-8a76-2a1843b7d118	inv_202011_luke.pdf	Luke
7255e1bd-417a-425b-ba88-98c3fe2405b2	inv_202011_lea.pdf	Lea
6a08f5e6-4fa8-44e6-92bb-db0d7b38a393	inv_202011_vader.pdf	Vader

still needs a mapping table...

# Salted Hashes

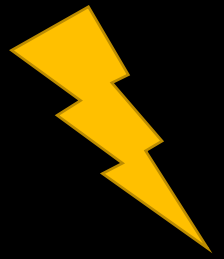
ID	Salt	Object	Owner
6cc433f2803080c9a633b2d110a745e26 4baca6c9e4455ea4d02f9ae41efe146	MYSALT	inv_202010_luke.pdf	Luke
bd4fbb2af573895ab6f239b2ba9f60f651 098a1f6c357a7ca7d018f1c46432f6	MYSALT	inv_202010_lea.pdf	Lea
b0a85076b44450cab45b507c8c0ddb0b3 6e46402204ca593bcc9a8db8119edb6	MYSALT	inv_202011_luke.pdf	Luke
ff33b6dfddcd14ed700d19dcca736b3b6 015f2badab5b9e740988e1ab70cc1b	MYSALT	inv_202011_lea.pdf	Lea
1bf50583d983a9981a86761daae7c1b6c 136bb932f9f921bd6cf20c01e6271b1	MYSALT	inv_202011_vader.pdf	Vader

# Insec. Direct Object References



Goal	Directly access a resource which you're not supposed to by the ui
How	information gathering by http parameters client source code logs error messages other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to access resources on server-side use indirect object references, unguessable IDs (e.g. UUIDs) or salted hashes
OWASP Top 10	
(Primary) Violated Principle	

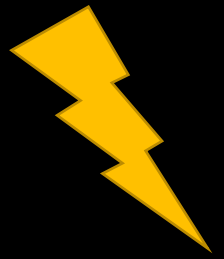
# Insec. Direct Object References



Goal	Directly access a resource which you're not supposed to by the ui
How	information gathering by http parameters client source code logs error messages other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to access resources on server-side use indirect object references, unguessable IDs (e.g. UUIDs) or salted hashes
OWASP Top 10	A5:2017-Broken Access Control
(Primary) Violated Principle	

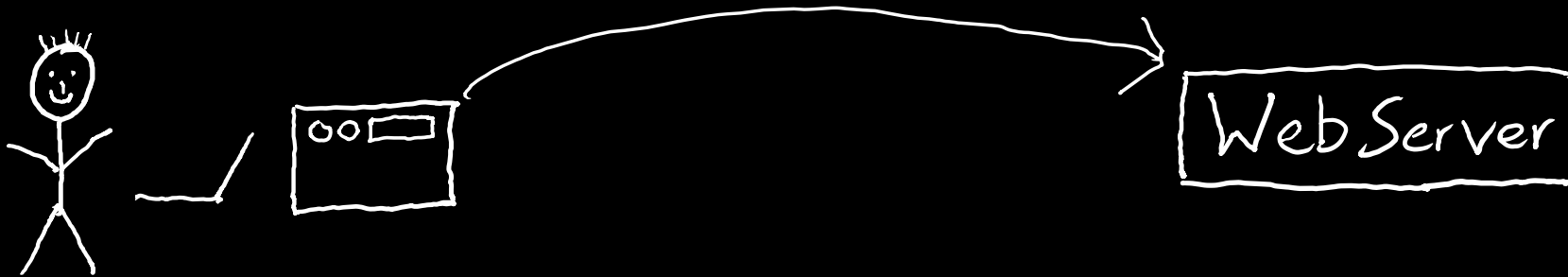


# Insec. Direct Object References



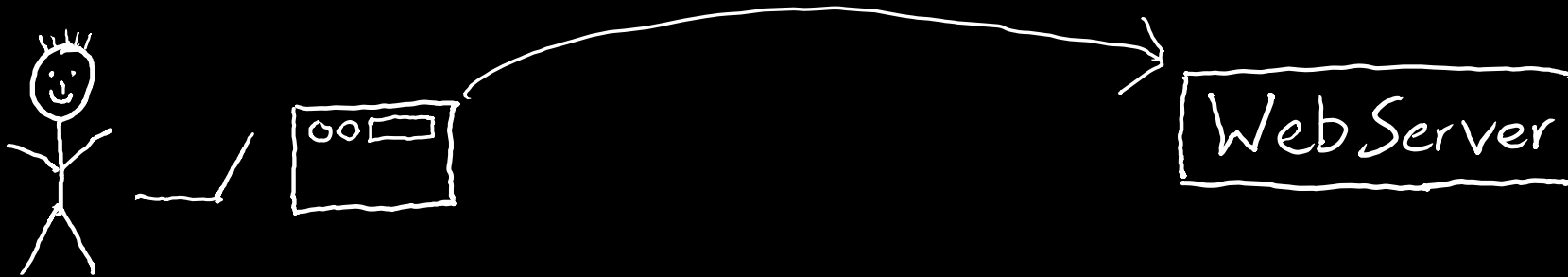
Goal	Directly access a resource which you're not supposed to by the ui
How	information gathering by http parameters client source code logs error messages other users with higher privileges  hacker instinct / guessing ;-)
Solution	NEVER rely on (client-side) frontend check user's authorization to access resources on server-side use indirect object references, unguessable IDs (e.g. UUIDs) or salted hashes
OWASP Top 10	A5:2017-Broken Access Control
(Primary) Violated Principle	„Authorize after you authenticate“

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



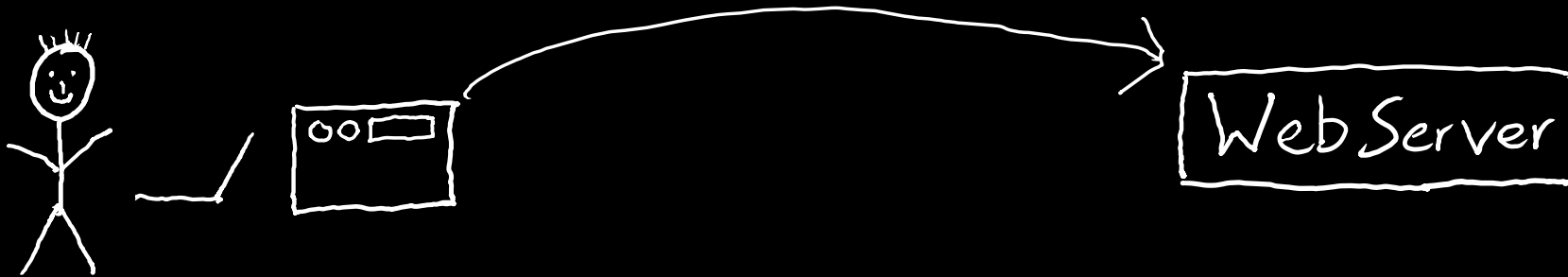
what would an attacker do?

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



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

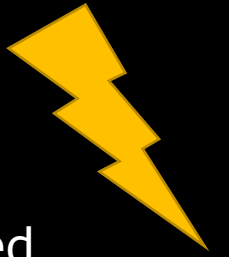
GET /invoice?f=../../../../../../../../etc/passwd HTTP/1.1  
Host: lightside.me



```
__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
```

```
invoice.php
1 <?php
2     $filename = $_GET['f'];
3     $dir = "./202010/";
4     header('Content-disposition: attachment; filename='.$filename);
5     readfile($dir.$filename);
6 >
```

# Path Traversal



Goal

Directly access files (outside the webroot) which you're not supposed to by the application

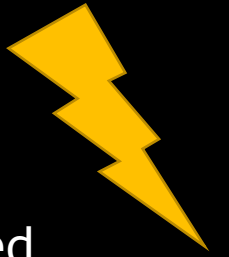
How

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

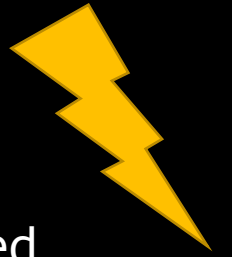
whenever an application handles path's incorrectly, try to iterate through it

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
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;
$target = ".$dir.$filename;

echo("Unfiltered target: {$target}<br><br>");
echo("Canon. target: ".realpath($target)."<br>");
echo("Target Dir: {$targetDir}<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: ./202010/../../../../../etc/passwd

Canon. target: /etc/passwd

Target Dir: /var/www/lightside/202010/

Path-Traversal attempt!



```
<?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;
$target = ".$dir.$filename;

echo("Unfiltered target: {$target}<br><br>");
echo("Canon. target: ".realpath($target)."<br>");
echo("Target Dir: {$targetDir}<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: ./202010/inv\_202010\_luke.pdf

Canon. target: /var/www/lightside/202010/inv\_202010\_luke.pdf

Target Dir: /var/www/lightside/202010/

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	A5:2017-Broken Access Control
--------------	-------------------------------

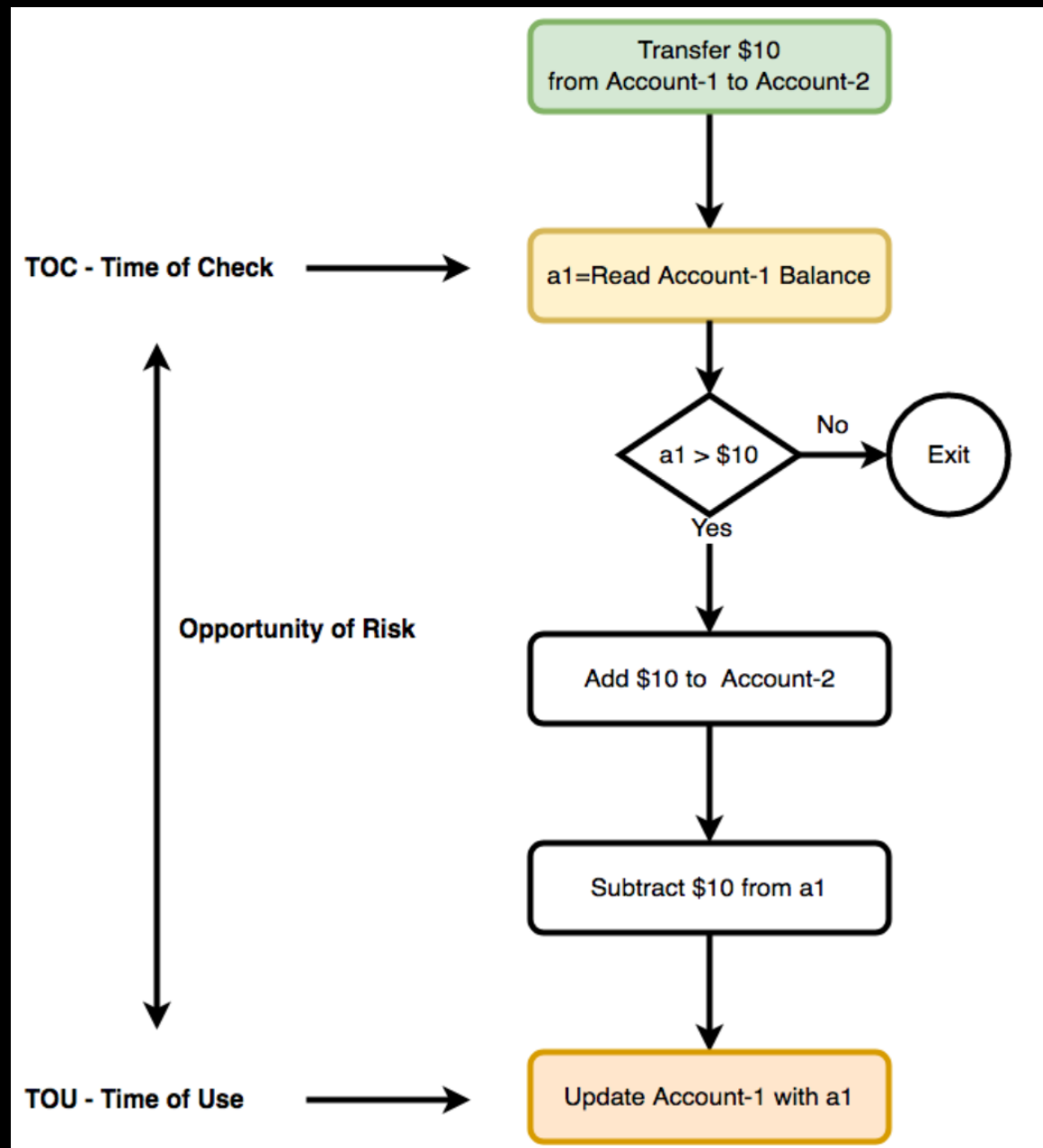
(Primary) Violated Principle	
---------------------------------	--

# Path Traversal

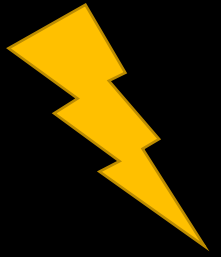


Goal	Directly access files (outside the webroot) which you're not supposed to by the application
How	whenever an application handles path's incorrectly, try to iterate through it
Solution	Don't use user input in filesystem APIs If you have to validate it (explicit allowlist) check the canonicalized path Run webserver under a low privileged user
OWASP Top 10	A5:2017-Broken Access Control
(Primary) Violated Principle	"Define an approach that ensures all data are explicitly validated."

so one last thing about authorization...



# TOCTOU (Race Condition)



Goal

Execute an operation you are actually not allowed to

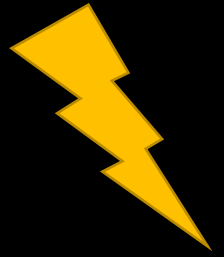
How

Solution

OWASP Top 10

(Primary)  
Violated Principle

# TOCTOU (Race Condition)



Goal

Execute an operation you are actually not allowed to

How

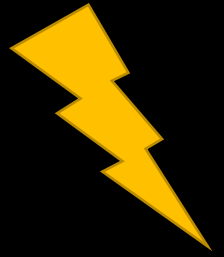
Take advantage of a time-window between a security check and the actual action this check should protect

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

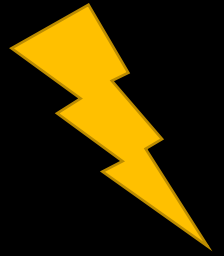
Apply security checks as close to the actual action as possible

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

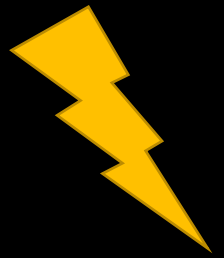
Solution

Apply security checks as close to the actual action as possible  
Use locking mechanisms, e.g.  
locking / semaphores  
database transactions  
etc.

OWASP Top 10

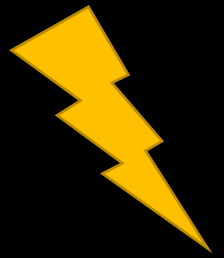
(Primary)  
Violated Principle

# 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	A5:2017-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	A5:2017-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**