Missing Function Level Access Control

CYBER SECURITY & PRIVACY FOUNDATION

Cyber Security & Privacy Foundation(CSPF)

Introduction

The vulnerability occurs when a web application enforces "**Access Control**" in Presentation layer and fails to do in Business Logic.

This vulnerability enables an unauthorized users to access a page or a privileged function that they shouldn't have ability to access.

Hiding links to a web page from the UI doesn't prevent an unauthorized user from accessing it.

Failure to Restrict URL Access

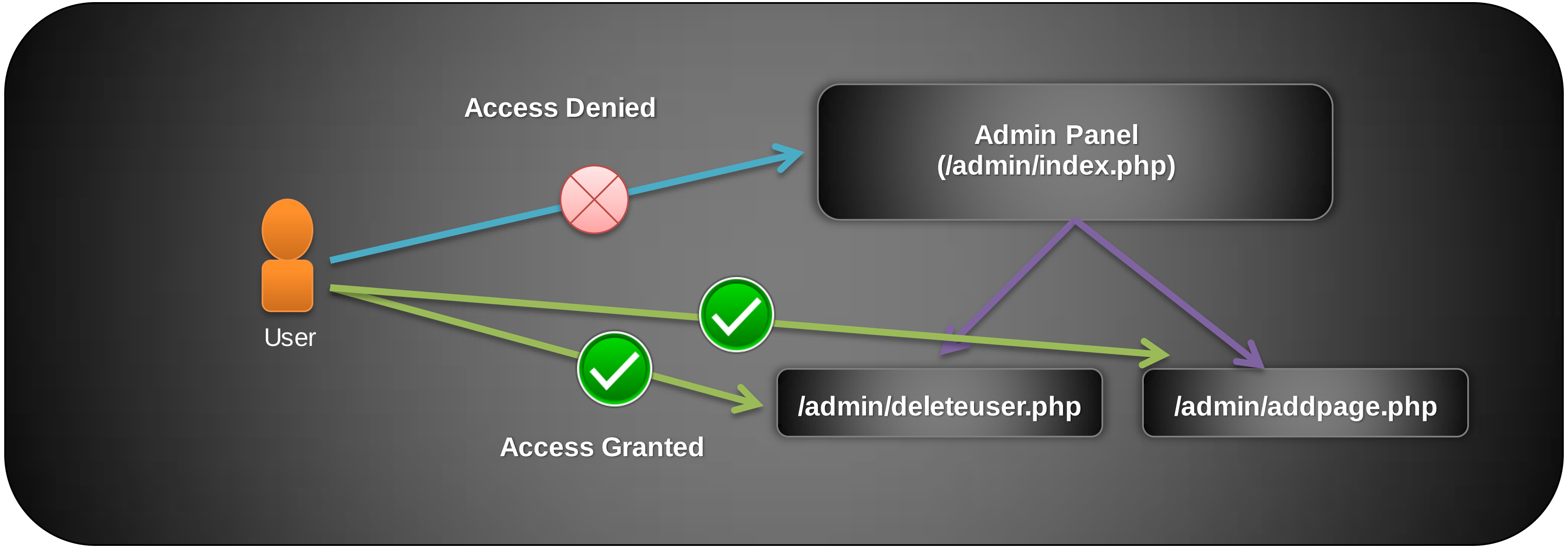
Hidden URLs:

Developer assumes that they can prevent a normal users from accessing a private functionality by creating a hidden pages – A pages that doesn't have links pointing to them & can't be crawled by Search Engines.

They failed to check whether the person is allowed to access the page or not.

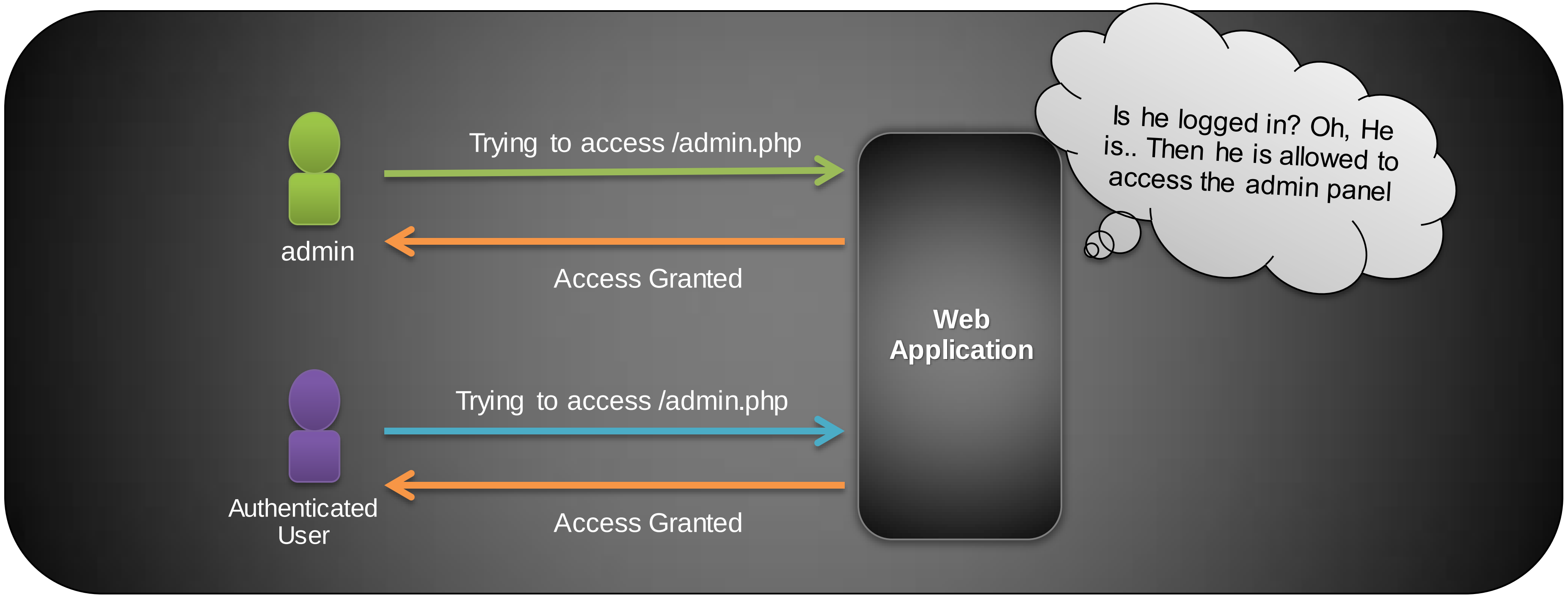
“Access Control Checks” only in Starting Page:

Developers check whether a user is allowed to access the admin panel page or not. They assumes that only admin is going to access other pages referenced in the admin panel and fails to perform “access control checks” on those pages.



Checking Only whether a user is logged in:

In some cases, developers only check whether user has a valid session ID before allowing them to access a protected page. If so, an authenticated user can access administrative functions.


Trying to access /admin.php
admin
Access Granted
Web Application
Is he logged in? Oh, He is.. Then he is allowed to access the admin panel
Trying to access /admin.php
Authenticated User
Access Granted

Forced Browsing

Forced Browsing, Also known as “**Predictable Resource Location**”, is an attack technique used to enumerate and access the resources that are not referenced by the application and intended to be hidden.

Common Files and Directories

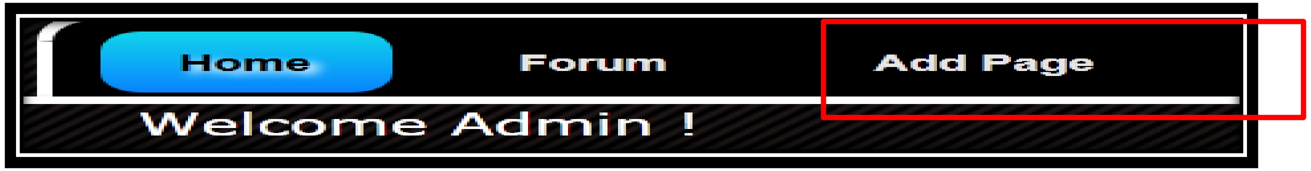
/admin/	admin.php	/administrator/
/logs/	/admin_area/	/password/
/logs/	/config/	/backup/

Robots.txt

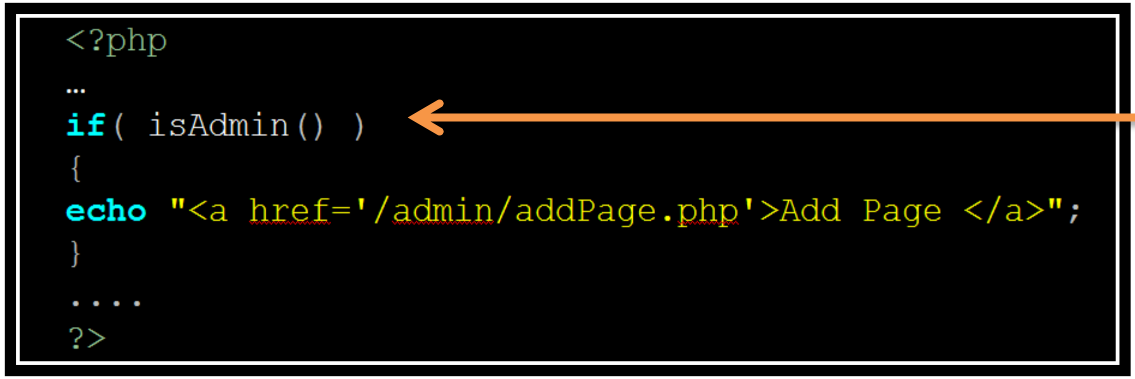
- A file that tells web crawlers exactly what part of a website are allowed to access.
- Some developers/admins assume that they can prevent a private resources being accessed by simply adding “Disallow” in robots.txt.
- This will prevent Search Engines from indexing the Private URLs. But attackers find it useful. If the web application fails to perform function level access control on those resources, attackers can easily access those resources.

Example

Let’s consider a Web Application that shows “Add Page” link only if the admin user is logged in.



PHP code of the **Index.php**:

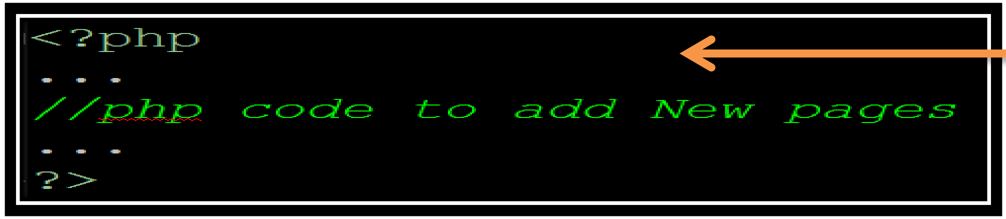


```
<?php
…
if( isAdmin() )
{
echo "<a href='/admin/addPage.php'>Add Page </a>";
}
....
?>
```

Shows Reference to a private page only if the admin is user logged in.

(Here, isAdmin is a user-defined function that checks whether the user is an admin)

PHP code of the ‘**addpage.php**’ :



```
<?php
...
//php code to add New pages
...
?>
```

Missing **‘ isAdmin() ’** Check.

(assuming: *isAdmin is a user-defined function that checks whether the user is an admin*)

- Here, the developer assumes that URL of this page is not known to anyone except admin (*as the link is only being shown to admin*).
- Fails to check whether this page is being **Accessed** by admin user or others.
- An unauthenticated user can directly visit the “**addpage.php**” and add new web pages.