IDOR
AND
HTTP SECURITY HEADERS

What is IDOR?

- When a web server processes a user's request, it determines the resource accessed using parameters stated within the HTTP request. The direct object reference is information that is used to locate and access a particular resource. While the server is retrieving a resource, attackers can manipulate these parameters and access internal implementation object details in the event of a lack of adequate server-side validation. This attack is known as an Insecure Direct Object Reference (IDOR) vulnerability.

IDOR examples

There are many examples of access control vulnerabilities where user-controlled parameter values are used to access resources or functions directly.

1. IDOR vulnerability with direct reference to database objects

Consider a website that uses the following URL to access the customer account page, by retrieving information from the back-end database:

```
https://insecure-website.com/customer_account?customer_number=132355
```

Here, the customer number is used directly as a record index in queries that are performed on the back-end database. If no other controls are in place, an attacker can simply modify the customer_number value, bypassing access controls to view the records of other customers. This is an example of an IDOR vulnerability leading to horizontal privilege escalation.

An attacker might be able to perform horizontal and vertical privilege escalation by altering the user to one with additional privileges while bypassing access controls. Other possibilities include exploiting password leakage or modifying parameters once the attacker has landed in the user's accounts page, for example.

2. IDOR vulnerability with direct reference to static files

IDOR vulnerabilities often arise when sensitive resources are located in static files on the server-side filesystem. For example, a website might save chat message transcripts to disk using an incrementing filename, and allow users to retrieve these by visiting a URL like the following:

```
https://insecure-website.com/static/12144.txt
```

In this situation, an attacker can simply modify the filename to retrieve a transcript created by another user and potentially obtain user credentials and other sensitive data.

Impacts of IDOR Vulnerability

- Exposure of Confidential Information: When the attacker will have control over your account via this vulnerability, it is obvious that an attacker will be able to come across your personal information.

- Authentication Bypass: As the attacker can have access to millions of account with this vulnerability, it will be a type of Authentication bypass mechanism.
  Alteration of Data: An attacker may have privileges to access your data and alter it. By this, an attacker may have

permission to make changes to your data, which may lead to manipulation of records.

- Account Takeover: While an attacker may have multiple access to user accounts just by changing the “UID” values, this will lead to account takeover vulnerability. When one vulnerability leads to another vulnerability (like in this case), It is known as Chaining of BUGS.

What are HTTP Security Headers?

When we visit any website in the browser, the browser sends some request headers to the server and the server responds with HTTP response headers. These headers are used by the client and server to share information as a part of the HTTP protocol. Browsers have defined behaviour of the web page according to these headers during communication with the server. These headers are mainly a combination of key-value pairs separated by a colon: There are many HTTP headers, but here I'm covering some very useful web security headers, which will improve your website security.

Types of security headers

You can use these headers to outline communication and improve web security. Let’s have a look at five security headers that will give your site some much-needed protection.

1. HTTP Strict Transport Security (HSTS)

Let's say you have a website named example.com and you installed an SSL/TLS certificate and migrated from HTTP to HTTPS. This is good, right? That was rhetorical. It definitely is. But this isn't where the work stops. What if your website is still available over HTTP? It would be utterly pointless, right? Many website admins migrate to HTTPS and then forget about it without realizing this. This is where HSTS enters the picture.

If a site is equipped with HTTPS, the server forces the browser to communicate over secure HTTPS. This way, the possibility of an HTTP connection is eliminated entirely.

Syntax:

```
Strict-Transport-Security: max-age=<expire-time>


Strict-Transport-Security: max-age=<expire-time>;
includeSubDomains


Strict-Transport-Security: max-age=<expire-time>;
preload
```

2. Content Security Policy (CSP)

The HTTP Content Security Policy response header gives website admins a sense of control by giving them the authority to restrict the resources a user is allowed to load within site. In other words, you can whitelist your site's content sources.

Content Security Policy protects against Cross Site Scripting and other code injection attacks. Although it doesn't eliminate their

possibility entirely, it can sure minimize the damage. Compatibility isn't a problem as most of the major browsers support CSP.

Syntax:

```
Content-Security-Policy: <policy-directive>;
<policy-directive>
```

3. Cross Site Scripting Protection (X-XSS)

As the name suggests, X-XSS header protects against Cross-Site Scripting attacks. XSS Filter is enabled in Chrome, IE, and Safari by default. This filter doesn't let the page load when it detects a cross-site scripting attack.

Syntax:

```
X-XSS-Protection: 0

X-XSS-Protection: 1

X-XSS-Protection: 1; mode=block

X-XSS-Protection: 1; report=<reporting-uri>
```

4. X-Frame-Options

In the Orkut era, a spoofing technique called 'Clickjacking' was pretty popular. It still is. In this technique, an attacker fools a user into clicking something that isn't there. For example, a user

might think that he's on the official Orkut website, but something else is running in the background. A user may reveal his/her confidential information in the process.

X-Frame-Options help guard against these kinds of attacks. This is done by disabling the iframes present on the site. In other words, it doesn't let others embed your content.

Syntax:

```
X-Frame-Options: DENY

X-Frame-Options: SAMEORIGIN

X-Frame-Options: ALLOW-FROM https://example.com/
```

5. X-Content-Type-Options

The X-Content-Type header offers a countermeasure against MIME sniffing. It instructs the browser to follow the MIME types indicated in the header. Used as a feature to discover an asset's file format, MIME sniffing can also be used to execute cross-site scripting attacks.

Syntax:

```
X-Content-Type-Options: nosniff
```

References

- https://crashtest-security.com/insecure-direct-object-reference-idor/
- https://portswigger.net/web-security/access-control/idor
- https://www.loginradius.com/blog/engineering/http-security-headers
- https://www.geeksforgeeks.org/insecure-direct-object-reference-idor-vulnerability/
- https://www.thesslstore.com/blog/http-security-headers/