Session Fixation

Session Fixation occurs when an attacker can fixate a (valid) session identifier. As you can imagine, the attacker will then have to trick the victim into logging into the application using the aforementioned session identifier. If the victim does so, the attacker can proceed to a Session Hijacking attack (since the session identifier is already known).

Such bugs usually occur when session identifiers (such as cookies) are being accepted from *URL Query Strings* or *Post Data* (more on that in a bit).

Session Fixation attacks are usually mounted in three stages:

Stage 1: Attacker manages to obtain a valid session identifier

Authenticating to an application is not always a requirement to get a valid session identifier, and a large number of applications assign valid session identifiers to anyone who browses them. This also means that an attacker can be assigned a valid session identifier without having to authenticate.

Note: An attacker can also obtain a valid session identifier by creating an account on the targeted application (if this is a possibility).

Stage 2: Attacker manages to fixate a valid session identifier

The above is expected behavior, but it can turn into a session fixation vulnerability if:

- `The assigned session identifier pre-login remains the same post-login and`
- `Session identifiers (such as cookies) are being accepted from URL Query Strings or Post Data and propagated to the application`

If, for example, a session-related parameter is included in the URL (and not on the cookie header) and any specified value eventually becomes a session identifier, then the attacker can fixate a session.

Stage 3: Attacker tricks the victim into establishing a session using the abovementioned session identifier

All the attacker has to do is craft a URL and lure the victim into visiting it. If the victim does so, the web application will then assign this session identifier to the victim.

The attacker can then proceed to a session hijacking attack since the session identifier is already known.

Session Fixation Example

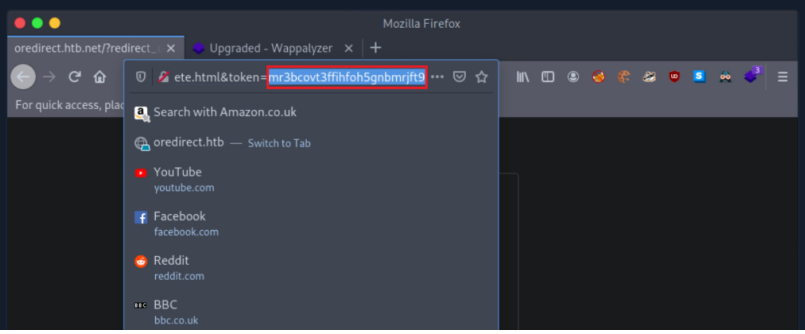
Proceed to the end of this section and click on `Click here to spawn the target system!` or the `Reset Target` icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target application and follow along. Don't forget to configure the specified vhost (`oredirect.htb.net`) to access the application.

Part 1: Session fixation identification

Navigate to `oredirect.htb.net`. You will come across a URL of the below format:

```
http://oredirect.htb.net/?redirect_uri=/complete.html&token=<RANDOM TOKEN VALUE>
```

Using Web Developer Tools (Shift+Ctrl+I in the case of Firefox), notice that the application uses a session cookie named `PHPSESSID` and that the cookie's value is the same as the `token` parameter's value on the URL.



? Go to Questions

Table of Contents

Introduction to Sessions

Session Attacks

Session Hijacking
Session Fixation
Obtaining Session Identifiers without User Interaction
Cross-Site Scripting (XSS)
Cross-Site Request Forgery
Cross-Site Request Forgery (GET-based)
Cross-Site Request Forgery (POST-based)
XSS & CSRF Chaining
Exploiting Weak CSRF Tokens
Additional CSRF Protection Bypasses
Open Redirect
Remediation Advice

Skills Assessment

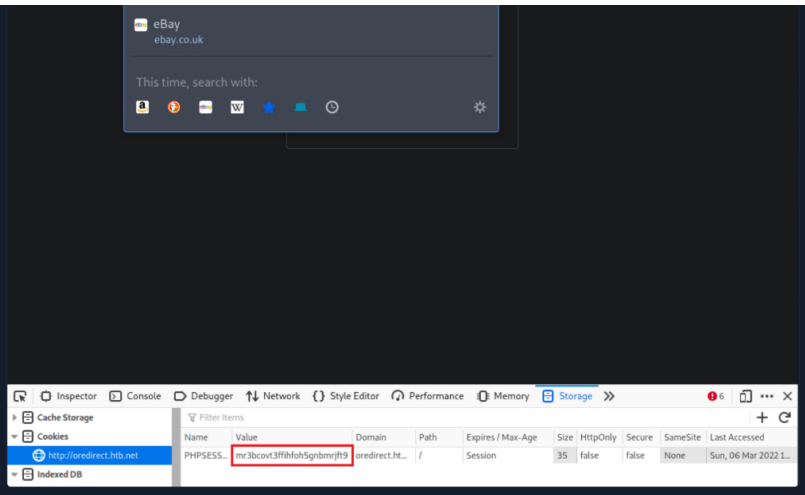
Session Security - Skills Assessment

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left



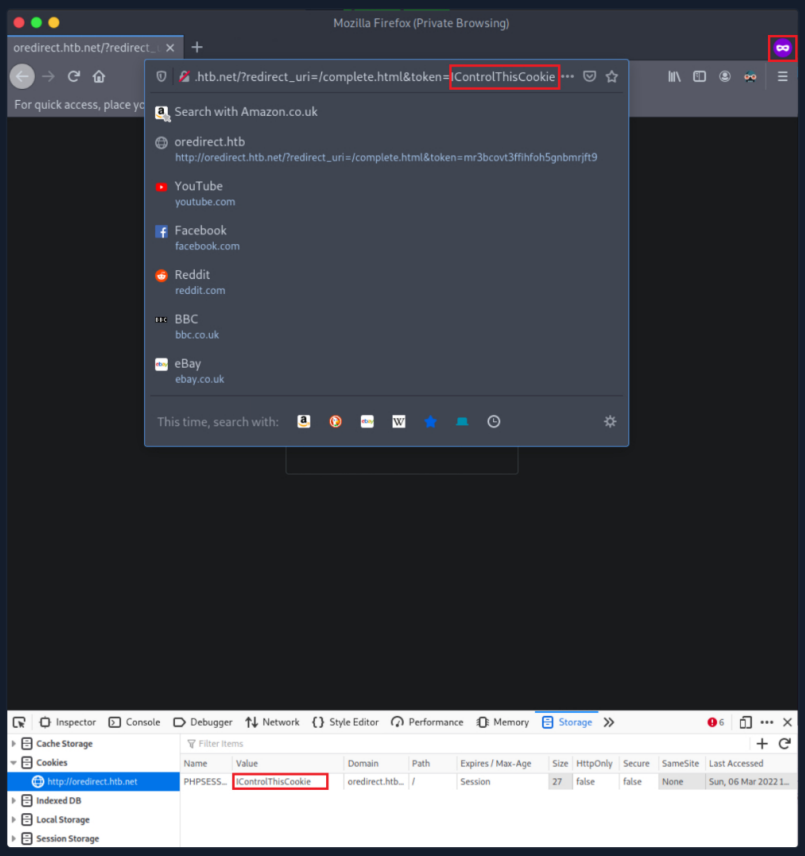
If any value or a valid session identifier specified in the `token` parameter on the URL is propagated to the `PHPSESSID` cookie's value, we are probably dealing with a session fixation vulnerability.

Let us see if that is the case, as follows.

Part 2: Session fixation exploitation attempt

Open a `New Private Window` and navigate to `http://oredirect.htb.net/?redirect_uri=/complete.html&token=IControlThisCookie`

Using Web Developer Tools (Shift+Ctrl+I in the case of Firefox), notice that the `PHPSESSID` cookie's value is `IControlThisCookie`



We are dealing with a Session Fixation vulnerability. An attacker could send a URL similar to the above to a victim. If the victim logs into the application, the attacker could easily hijack their session since the session identifier is already known (the attacker fixated it).

Note: Another way of identifying this is via blindly putting the session identifier name and value in the URL and then refreshing.

For example, suppose we are looking into `http://insecure.exampleapp.com/login` for session fixation bugs, and the session identifier being used is a cookie named `PHPSESSID`. To test for session fixation, we could try the following `http://insecure.exampleapp.com/login?PHPSESSID=AttackerSpecifiedCookieValue` and see if the specified cookie value is propagated to the application (as we did in this section's lab exercise).

Below is the vulnerable code of this section's lab exercise.

Code: php

```php
<?php
    if (!isset($_GET["token"])) {
        session_start();
        header("Location: /?redirect_uri=/complete.html&token=" . session_id());
    } else {
        setcookie("PHPSESSID", $_GET["token"]);
    }
?>
```

Let us break the above piece of code down.

Code: php

```php
if (!isset($_GET["token"])) {
     session_start();
```

The above piece of code can be translated as follows: If the *token* parameter hasn't been defined, start a session (generate and provide a valid session identifier).

Code: php

```php
header("Location: /?redirect_uri=/complete.html&token=" . session_id());
```

The above piece of code can be translated as follows: Redirect the user to `/?redirect_uri=/complete.html&token=` and then call the *session_id()* function to append *session_id* onto the token value.

Code: php

```php
 } else {
        setcookie("PHPSESSID", $_GET["token"]);
    }
```

The above piece of code can be translated as follows: If the *token* parameter is already set (else statement), set *PHPSESSID* to the value of the *token* parameter. Any URL in the following format `http://oredirect.htb.net/?redirect_uri=/complete.html&token=AttackerSpecifiedCookieValue` will update *PHPSESSID*'s value with the *token* parameter's value.

By now, we have covered session hijacking and session fixation. Moving forward, let us see some ways through which a bug bounty hunter or penetration tester can obtain valid session identifiers that can be then used to hijack a user's session.

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Download VPN Connection File

Target(s): Click here to spawn the target system!

vHosts needed for these questions:

- `oredirect.htb.net`

+1 If the HttpOnly flag was set, would the application still be vulnerable to session fixation? Answer Format: Yes or No

Yes

Submit

Previous Next

Mark Complete & Next