Testing for Session Fixation

ID
WSTG-SESS-03

Summary

Session fixation is enabled by the insecure practice of preserving the same value of the session cookies before and after authentication. This typically happens when session cookies are used to store state information even before login, e.g., to add items to a shopping cart.

In the generic exploit of session fixation vulnerabilities, an attacker can obtain a set of session cookies from the target website without authenticating and force them into the victim's browser, using different techniques; if the victim later authenticates at the target, she will be identified by the session cookies chosen by the attacker, who will then become able to impersonate the victim.

The issue can be fixed by refreshing the session cookies after the authentication, otherwise the attack can be prevented by ensuring the cookie integrity, i.e. using full HSTS adoption[1] or `__Host-` and `__Secure-` prefixes in the cookie name.

How to Test

Black-Box Testing

Intuition

In this section will be given a general idea of the testing strategy that will be shown in the next section.

The first step is to make a request to the site to be tested (*e.g.* `www.example.com`). If the tester requests the following:

`GET www.example.com`

They will obtain the following answer:

```
HTTP/1.1 200 OK
Date: Wed, 14 Aug 2008 08:45:11 GMT
Server: IBM_HTTP_Server
Set-Cookie: JSESSIONID=0000d8eyYq3L0z2fgq10m4v-rt4:-1; Path=/; secure
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=Cp1254
Content-Language: en-US
```

The application sets a new session identifier JSESSIONID=0000d8eyYq3L0z2fgq10m4v-rt4:-1 for the client.

Next, if the tester successfully authenticates to the application with the following POST (`https://www.example.com/authentication.php`):

```
POST /authentication.php HTTP/1.1
Host: www.example.com
[...]
Referer: http://www.example.com
Cookie: JSESSIONID=0000d8eyYq3L0z2fgq10m4v-rt4:-1
Content-Type: application/x-www-form-urlencoded
Content-length: 57

Name=Meucci&wpPassword=secret!&wpLoginattempt=Log+in
```

The tester observes the following response from the server:

```
HTTP/1.1 200 OK
Date: Thu, 14 Aug 2008 14:52:58 GMT
Server: Apache/2.2.2 (Fedora)
X-Powered-By: PHP/5.1.6
Content-language: en
Cache-Control: private, must-revalidate, max-age=0
X-Content-Encoding: gzip
Content-length: 4090
Connection: close
Content-Type: text/html; charset=UTF-8
...
HTML data
...
```

As no new cookie has been issued upon a successful authentication the tester knows that it is possible to perform session hijacking.

> The tester can send a valid session identifier to a user (possibly using a social engineering trick), wait for them to authenticate, and subsequently verify that privileges have been assigned to this cookie.

Strategy

We assume to have a testing account on the website for the victim, that we call Alice. We simulate a scenario where a network attacker (i.e., an attacker who has access to the same network as the victim) forces in Alice's browser all the cookies which are not freshly issued after login and do not have integrity against her. After Alice's login, the attacker presents the forced cookies to access Alice's account: if they are enough to act on Alice's behalf, session fixation is possible.

Specifically, the testing strategy proceeds as follows:

1. Login to `www.target.com` as Alice and reach the page under test;

2. Find the cookies which satisfy both the following cookie compromission conditions:
    - lack of full HSTS adoption;
    - lack of `Host-` and `Secure-` prefixes in the cookie name;
3. Clear all the other cookies from the browser;
4. Perform an operation on Alice's account under test;
5. Check: has the operation been performed? If yes, report as insecure;
6. Clear the cookies from the browser;
7. Login to `www.target.com` as the attacker and reach the page under test;
8. Restore in the browser the cookies previously kept at step 2;
9. Perform again the operation under test;
10. Clear the cookies from the browser and login to `www.target.com` as Alice;
11. Check: has the operation been performed? If yes, report as insecure.

We recommend using two different machines and/or browsers for Alice and the attacker. This allows to decrease the number of false positives if the web application does fingerprinting to verify an access enabled from a given cookie.

1: We refer to full HSTS adoption when a host activates HSTS for itself and all its sub-domains, and to partial HSTS adoption when a host activates HSTS just for itself.

Gray-Box Testing

Talk with developers and understand if they have implemented a session token renew after a user successful authentication.

> The application should always first invalidate the existing session ID before authenticating a user, and if the authentication is successful, provide another session ID.

Tools

- OWASP ZAP

References

- Calzavara, S., Rabitti, A., Ragazzo, A., Bugliesi, M.: Testing for Integrity Flaws in Web Sessions.

Whitepapers

- Session Fixation
- ACROS Security
- Chris Shiflett