PSI-CAD-Project Report SoSe 2019

Alexander Böhner (1937944)
mailto:alexander.boehner@stud.uni-bamberg.de

Melanie Vogel (1738258)
mailto:melanie-margot.vogel@stud.uni-bamberg.de

September 30, 2019

Contents

List of Figures

1 Instructor Guides

1.1 Instructors Guide: Session Hijack

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token. Because http communication uses many different TCP connections, the web server needs a method to recognize every user's connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like in the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition. The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server[1].

1.1.1 Related Work

For this topic OWASP provides a short introduction online[2]. It is a good source for initial research in this area.
The type of session hijacking in this task is based on a predictable session ID, which can be brute forced. For example [End01] shows how such attacks may work.
The brute forcing is possible because the session ids here are poorly constructed. Following the guidelines of [Fu+01] helps in understanding and preventing such mistakes.
In other common session hijack variants, XSS plays an important role. Then chapter **??** may be interesting as well. For solving this scenario an insight to sessions and cookies[3] might be helpful.

1.1.2 Task: Session Hijack

You should solve one task in this scenario. The goal is that you get the session ID of the user Donald and with his ID get access to his profile page that holds sensitive information such as the username and password for the webshop.

When you solve the task successfully, you can set Donald's cookie as yours and view his profile. Enter the stolen sessionId as the secret.

[1] https://www.owasp.org/index.php/Session_hijacking_attack
[2] https://www.owasp.org/index.php/Session_hijacking_attack
[3] https://www.ietf.org/rfc/rfc2109.txt

1.1.2.1 Special Notes

- It is required to write a script in any programming language to produce a number of sessionIds, since this is kind of a brute force attack.
- The sessionId consists of two values that must be used as a combination to hack the session of Donald.

1.1.2.2 Guide

The task can be guided as follows including additional hints to achieve the goal and understand the basics of the attack:

1. **What is hidden in the server configs?.**

   In general, existing vulnerable mechanisms of the server configuration should be exploited. At first you should observe how your own session ID looks like and how the requests of you and Donald to the server look like. Every server has a logging page for the requests send from the client. In this case it can be accessed by everyone via its filename. Find out what webserver is used and what the file name could be. The log file contains the own sessionId and another one which is from Donald. Compare the log file with the sessionId in the web developer tools in the browser. The clue is that the sessionId consists of two parts named *id* and *session* where *id* is basically the cookie and the *session* is a generated string. The attacker should notice that the logged requests consist only of the first part, the *id*, whereas a successful hack of the profile requires both *id* and *session* value.

   Learning objective: Understand mechanisms to monitor server communication traffic and understand how sessionIds are structured.

2. **Hands-on: Collect sessionIds!**

   View how sessionIds change with each request. They are newly generated by each request. The attacker should now know that new *session* cookies were generated with each request to `/session?id=...` at the backendserver (i.e. expresserver at port 3000). Requesting those sessionIds for an already existing *id* cookie, the server responds with a specific error message. After querying a few of those sessionIds, the attacker should at some point find out, that these sessionId's are restricted to specific values. The goal hereby is to collect as many *session*'s as possible in order to get information about the space of possible session ids.

   Learning objective: Learn how sessions are generated.

3. **Whats the right combination?**

   In order for the attacker to come up with own session cookies, he or she should notice that the *session* value is an text encoded as a hexadecimal ascii value. After decoding the values, it should be obvious that the sessionId's are made

up of a bavarian adjective and corresponding noun. Several combinations of adjectives and nouns must be combined and sent via an request. With the help of these values the attacker can construct own sessionId's by combining all the retrieved values and perform a brute force attack with the first part of the cookie from the access.log and the unknown second part via the request url: `http://192.168.178.32:3000/profile?id=<id>&session=<session>`.

Learning objective: Learn to perform a brute force attack.

Bibliography

[End01] David Endler. “Brute-Force Exploitation of Web Application Session IDs”. In: (Dec. 2001).

[Fu+01] Kevin Fu et al. “The Dos and Don’ts of Client Authentication on the Web.” In: *USENIX Security Symposium*. 2001, pp. 251–268.