Fancy Challenge Title

Network Security HS16 - Group 6
Lukas Bischofberger, Silvan Egli, Jonas Passerini, Lukas Widmer

November 10, 2016

Abstract

The basis of our challenge is a chat service. There are normal users and administrators. To verify all the users Jason Web Tokens (JWT) are used. The administrators can make changes to the appearance. The challenge consists of two tasks. The ultimate goal is to extract the private key of the server.
The first tasks is to exploit of JWT. The problem of JWT is that they need an algorithm for the verification step however an attacker can bypass this and so be seen as administrator. This is done in the way that the attacker forges his own token.
The second task is to use the administrator rights. The changes of the appearance is done over the bash so a bash injection is possible. This vulnerability can then be used to extract the private key of the server.

1 Challenge Description

1.1 Type of Challenge

The first task requires some offline work as one has to forge a token to get administrator rights. The rest of the challenge is an online as our chat service is a web application and the extraction of the private key will only be possible when the application is online.

1.2 Category

The challenge belongs to the category web security.

1.3 Mission

The objective of the first task is to become an administrator by using the JWT vulnerability. The vulnerability is that it is possible to sign the tokens oneself and send them with the 'none' algorithm.
The objective of the second task is to exploit the bash injection vulnerability. There one has then to use this to extract the private key of the server by writing it to the html-file.

1.4 Learning Goal

The first task should increase the awareness of the problem when the user can choose the algorithm for an exchange by himself. This is especially a problem if it is also possible to not use any algorithm. This has to be handled, if it is not done by the library itself. The second task should make aware of the problem when one allows users to make inputs to the bash. If it is really needed that users can make inputs to the bash, one should check the string for unintended commands or only allow certain strings to be executed.

2 Implementation

2.1 Requirements

We have to make a web interface, where users first have to get a JWT to determine if they are standard users or administrators. In the web interface itself the users can chat. The administrators can also apply changes to the appearance over special commands, which are written in the normal textbox like with Slack. These commands are then bash commands, which can change the html-file where the css-code is integrated.

2.2 Deployment

We separated the work in several parts. On one hand there has to be Django, which uses JWT. Then also the background of the server.

3 Solution

3.1 Hints

We will give some hints like that we use JWT. This should help to find out the problem with the 'none' algorithm. Then in the chat history some special

commands will be written, so that a user notices that an administrator can do more then just chat.

3.2 Step-by-Step Instructions

The first step is to forge a token. Then he has to send this token so that he becomes an administrator. Then he uses his new rights to write a bash command to write the private key to html file.

4 Mitigation

To tackle the first task one has to prevent that the 'none' algorithm can be used. However this can still lead to some problems so the best thing is to only accept one algorithm predefined by the server.
The problem of bash injection in the second task can be solved by either not using the text field for the changes one would like to make but buttons for example. An other way would be to check the strings for malicious substrings or even only allow certain strings and if a string does not match the predefined ones it is not accepted.

5 Conclusion

We made a challenge consisting of two general problems. On one hand not perfectly implemented access measures, which can lead to a permission increase by malicious users. On the other hand bash injections, which are the consequence of not properly checking inputs. Both are common problems, which could mostly be tackled in a simple way.