JWT and Bash Injection

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

November 14, 2016

Abstract

The basis of our challenge is a chat service. There are normal users and administrators. To authenticate the users JSON Web Tokens (JWT) are used. The administrators can make changes to the appearance of the website. The challenge consists of two consecutive tasks, where the ultimate goal is to extract a private key on the server. The first tasks is to take advantage of a possible misconfiguration of JWT. We take advantage of the fact that certain implementations of JWT libraries allow the client to choose the signing algorithm.The second task is to exploit the system by use of a bash injection which is only possible with use of the administrator rights.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

There is web application (the chat service) hosted by some major company. Your task is it to break into this application as a weak entry point and steal

the private key of the web application which you know is also used by other web applications hosted by this company. The objective can be achieved by using two vulnerabilities in the web application, first the JWT signing algorithm and second a bash injection.

A simple chat service is provided by a company in the form of a web application. There is only one chat room, and all authenticated users might read and send messages to this room. There is an additional administrative role, which allows users belonging to this role to invoke a selection of admin commands, e.g. to change the background color of the chat window.

The challenge is expose the secret key of the web application by exploiting two vulnerabilities in the web application. The first vulnerability can be used to gain administrative user rights and the second allows an administrator to craft a malicious admin command to reveal the secret key.

1.4 Learning Goal

The first task should teach the student about JWT and token based authentication in general. But also increase the awareness of the problem when the client can choose the signing algorithm. The second task should make students aware of how user input needs to be sanitized when used as input to SQL or bash.

2 Implementation

2.1 Requirements

We need a webserver which runs Python and standard bash. Thereon we use the web framework dDango and either an old or patched JWT library, such as REFERENCE. To run the admin commands, the command is first authenticated and parsed using Django and then executed as a bash command in a subprocess. The commands directly write into CSS files to modify the color options.

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

The first problem can be solved by disallowing the client to choose the 'none' algorithm as signing algorithm for his JWT token.
The problem of bash injection in the second task can be solved by checking 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.