978-1-6654-3709-7

# JSON Web Token Penetration Testing on Cookie Storage with CSRF Techniques

Irfan Darmawan
*Department of Information System*
*Telkom University*
Bandung, Indonesia
irfandarmawan@telkomuniversity.ac.id

Aditya Pratama Abdul Karim
*Departement of Informatics*
*Siliwangi University*
Tasikmalaya, Indonesia
aditya1995.jr@gmail.com

Alam Rahmatulloh
*Departement of Informatics*
*Siliwangi University*
Tasikmalaya, Indonesia
alam@unsil.ac.id

Rohmat Gunawan
*Departement of Informatics*
*Siliwangi University*
Tasikmalaya, Indonesia
rohmatgunawan@unsil.ac.id

Dita Pramesti
*Department of Information System*
*Telkom University*
Bandung, Indonesia
ditapramesti@telkomuniversity.ac.id

2021 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS) | 978-1-6654-3709-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICADEIS52521.2021.9701965

***Abstract*—An authentication process is an act of proving the identity of a user when entering a system. Token-based authentication is a type of authentication that is stateless. This means that when the authentication process is carried out, there is absolutely no information about the user because the use of tokens in every request is made from the client to the server. Java Script Object Notation (JSON) Web Token is an authentication technique that provides an open and secure way to represent claims between two parties, cryptographically signed, which is designed not to be forged. However, this needs to be proven safe and not vulnerable. The purpose of this study is to conduct penetration testing of the security of JSON Web Token (JWT) storage on cookie storage using CSRF techniques. Scenarios for performing the CSRF technique were prepared in the experiment. The system architecture and tools to be used are prepared before the experiment is carried out. The experimental results in this study show that the part of the cookie attribute that embeds the flag "set-httponly: false", can be accessed by javascript on the client-side (read and write). The CSRF technique that was tried in the research has succeeded in utilizing JWT tokens stored in cookies to send faked requests. Eventually, the victim's account was used, and the resource was taken over.**

***Keywords*—CSRF, JWT, Penetration Testing**

## I. Introduction

The World Wide Web has changed people's lives very drastically. Large individuals and organizations use the Web every day. Web applications such as personal websites, discussion forums, e-commerce applications are an important thing which is spread all over the world [1]. Most of the infrastructure, such as banks, stock market, health, education, transportation, communications, defense, all use web applications [2]. The increasing dependence on web applications, the variety of services provided, and the ever-increasing amount of data sparked attackers' interest in these systems [3]. Data stored in a computer system must be proven safe and not vulnerable. One way to do this is called penetration testing [4]. In penetration testing, the tester simulates the activity of a malicious attacker who tries to exploit a target system's vulnerability [5].

When using web-based applications such as e-commerce or e-learning, users usually need to have an account first. Account information is very important and confidential, usually used when accessing a service. Account authentication needs to be done to verify real or fake users [6]. Common authentication techniques used in web applications include session-based authentication and token-based authentication [6]. Session-based authentication was invented earlier and is an outdated method that almost every site uses. Meanwhile, the token-based authentication method is stateless, having absolutely no information about the user because the use of tokens in every request is made from the client to the server.

Token-based authentication, allowing the user to verify their identity, and in return, receive a unique access token [7], [8]. During the lifetime of the token, users can access the website or application where the token is issued. This method is more concise than having to re-enter credentials every time you return to the same web page, any application or resource that is protected by the same token. The user maintains access as long as the token remains valid. After the user logs out or exits the application, the token becomes invalid. Token-based authentication differs from traditional password-based or server-based authentication techniques. The token offers a second layer of security, and administrators have detailed control over every action and transaction [9]. Token-based authentication shows better performance than session-based methods [6], because in token-based authentication, no session is created every time a user logs in but only the time between login and logout. New trends in token-based authentication include using JSON Web Tokens (JWT). JWT is a JSON-based credential that provides an open and secure way to represent claims between two parties [10], cryptographically signed design not to be counterfeited [11].

Several studies related to JWT have been conducted before, including Token-based authentication using JSON Web Token in RESTful Web applications [12], Performance comparison of signed algorithms on JSON Web Token [13]. However, in this study, the location of the token has not been discussed, and the JWT token has not been tested for security to determine its characteristics in facing various kinds of threats.

Several studies related to penetration testing have been carried out before, including Exploiting web application vulnerabilities with Cross Site Scripting (XSS) and Cross Site Request Forgery (CSRF) techniques [2] [3], Security testing methodology for XSS vulnerability detection in web services [14], presentation of statistical results and security consolidation of various web applications against Cross Site

Request Forgery (CSRF) attacks [15]. JWT storage methods commonly used in web-based applications are HTML5 Web Storage (Session storage, Local storage) and Cookie Storage. Token storage on Cookies has vulnerability to Cross Site Request Forgery (CRSF) attacks[16]. The purpose of this study is to perform penetration testing of the security of JWT token storage in cookie storage using CSRF techniques.

## II. Related Work

Research [2], [3], trying to do penetration testing on web applications with the XSS and CRSF techniques. Support tools and scenarios are prepared to conduct penetration testing experiments. XSS and CSRF attacks are implemented by manipulating the connection between the user and the server, further tricking the user and server into running unauthorized scripts. His research has successfully used scripts to exploit security holes with XSS and CRSF techniques in web applications.

On research [14], attempts to attack XSS techniques to exploit a vulnerability in a web service. The approach taken uses two Security Testing techniques, namely Penetration Testing and Fault Injection, to mimic XSS attacks on Web Services in combination with WS-Security (WSS) and Security Tokens so as to identify senders and ensure legitimate access control to SOAP messages. Interchangeable. The soapUI and WSInject vulnerability scanners are a new error injection tool that introduces errors or errors in Web Services to analyze behavior in environments that are not robustly used in an experiment. The results show that the use of WSInject, compared to soapUI, improves vulnerability detection allowing it to emulate XSS attacks and generate new types.

Other research [15], tries to present statistical and consolidated results received in security studies of various web applications against cross-site request forgery attacks. The results of the consolidation of information about attacks and protection measures currently used by web application developers are presented, as well as showing the different types of distribution: distribution of identified vulnerabilities according to developer type, distribution of security measures used in web applications, distribution of identified vulnerabilities according to language programmed, studied data about the number of security measures used in web applications. The results show that in many cases web application developers do not pay attention to protection against cross-site request forgery attacks.

Penetration testing with the CSRF technique on the JSON Web Token is the main focus of this research. JWT tokens are stored on cookies. Script prepared to run on an experiment to be performed. The system architecture is designed, and several tools are prepared to support the experiment. In the final stage, an analysis of the results of the experiment was carried out and concluded.

## III. Experimental Design

There are four stages in this study, namely: Determining the system requirements, Installation and Configuration Tools, Penetration Testing with CSRF Technique, Analysis of experimental results.

### 1. *Determining System requirements*

At this stage, they technically defined the flow of experiments, in general, to be carried out. The stages of penetration testing using the CSRF technique are generally shown in Fig. 1.

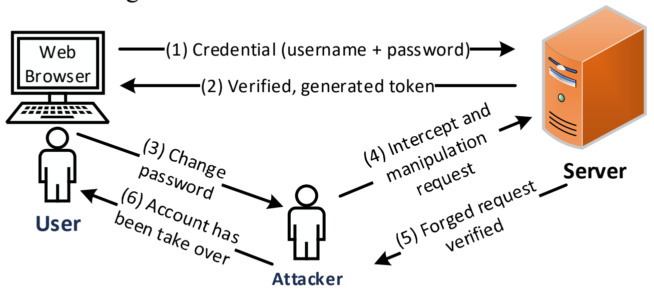

Web Browser
(1) Credential (username + password)
(2) Verified, generated token
(3) Change password
(4) Intercept and manipulation request
Server
User
(6) Account has been take over
(5) Forged request verified
Attacker

Fig. 1. The flow of penetration testing with the CSRF

Fig. 1. displays an overview of 6 sequences of activities carried out in the penetration process with CSRF techniques. There are three main entities involved, namely: User, Server, Attacker. The activity begins with inputting the username and password by the user via a web browser. The web browser will make a request to the server. In the second stage, the server verifies and generates a token, then sends a response to the client. In the third stage, the user tries to change the password. In the fourth stage, requests sent from the client to the server are intercepted and manipulated by the attacker. Manipulated requests are then sent to the server by the attacker.

In the fifth stage, the server verifies the request because the data received is valid, even though it has actually been faked by the attacker. In the sixth stage, the attacker manages to take over the user account. In order for each stage of penetration testing activities with the CSRF technique, as shown in Fig 1, several tools are required that must be provided. In general, the tools that must be prepared are shown in table 1.

Tabel 1. System Requirement

| No | Item | Description | Version |
|---|---|---|---|
| **1.** | Kali Linux | Operating System | 2020.3 |
| **2.** | Ubuntu server | Operating System | 20.04.1 |
| **3.** | Burp Suite | Testing Tool | 2020.9.1 |
| **4.** | Hashcat | Password recovery | 6.1.1 |
| **5.** | OWASP Juice Shop | Vulnerable Web Application | 12.1.1 |
| **6.** | OWASP Web Goat | Vulnerable Web Application | 8.1.0 |
| **7.** | FoxyProxy Standard | Web browser extension proxy | 7.4.3 |
| **8.** | Docker Container | Operating System for container | 19.03.13 |
| **9.** | Virtual Box | Virtual Machine | 6.1.12 |

### 2. *Installation and Configuration Tools*

At this stage, supporting tools are installed to run web applications such as docker containers and support applications for penetration and exploitation such as Burpsuite and hashcat. As well as configuring a proxy on a web browser and burpsuite.

*3. Penetration Testing with CSRF Technique*

At this stage, the web application is tested using the CSRF technique.

*4. Analysis of experimental results*

At this stage, analysis, and drawing conclusions are carried out on the results of the penetration testing experiments that have been carried out.

## IV. Result and Analysis

The system architecture used in the experiment in this study runs on a virtual machine. In general, the system architecture developed is shown in Fig. 2.

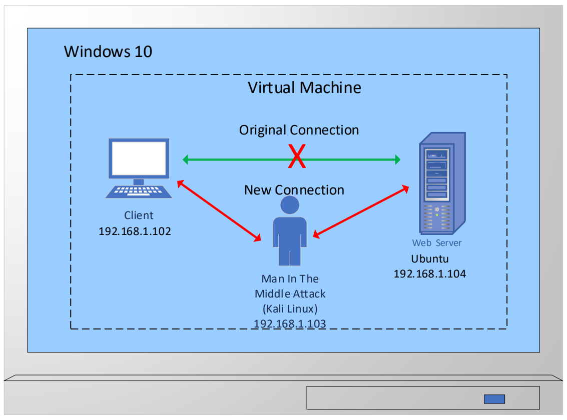


Fig. 2. System Architecture

*1. Installing Docker Container*

At this stage, the docker container installation is carried out from the packages that are already available in the Linux repository. Docker functions as a server library. Docker installation is done with the command **`sudo apt-get install docker.io`**.

*2. Installing OWASP Juice Shop*

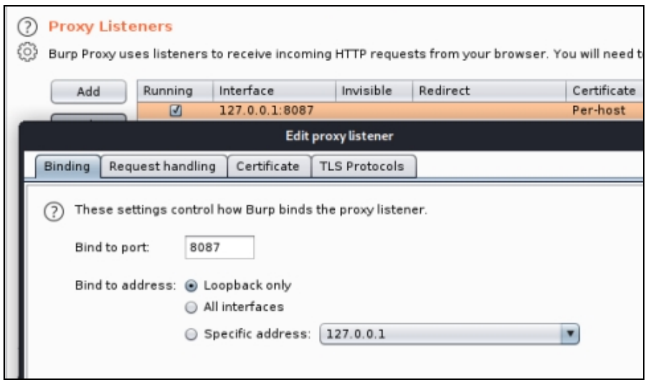
This stage is the download process from github bkimminich / juice-shop; with the command **`sudo docker pull bkiminich/juice-shop`**.

*3. Installing Hascat*

In order for the hashcat installation to be carried out, it is necessary to update the repository on Linux by running the command **`sudo apt-get install hashcat`**.

*4. Configuration Tool*

This is done with a proxy configuration on Burp, and this is useful for receiving HTTP requests from a web browser. Fig. 3 shows a proxy listener with the default IP along with port 8087 has been added.


Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need t

| Running | Interface | Invisible | Redirect | Certificate |
|---|---|---|---|---|
| ✓ | 127.0.0.1:8087 | | | Per-host |

Edit proxy listener

Binding | Request handling | Certificate | TLS Protocols

These settings control how Burp binds the proxy listener.

Bind to port: 8087

Bind to address: [✓ Loopback only] All interfaces Specific address: 127.0.0.1

Fig. 3. Configure Burp Proxy

*5. Cross Site Request Forgery (CSRF) Penetration Testing*

All user login information that has been loaded in the JWT token is stored on browser cookies. However, it can be seen in Fig. 4, the part of the cookie attribute that becomes a weakness is to embed the flag "set-HTTPponly: false", which means that cookies can be accessed by javascript on the client-side (read and write).

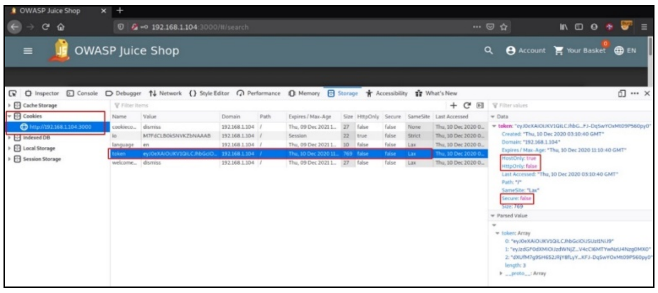


Fig. 4. JWT Information on Cookies

At this stage, the intercepted GET request parameter is displayed when the client updates the password. Fig. 5. shows the request sent to the server, including a valid API token, so the server responds with a success status.

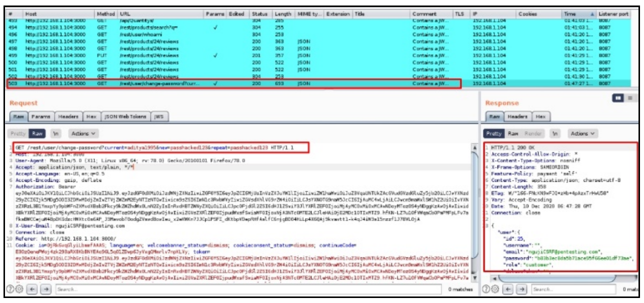
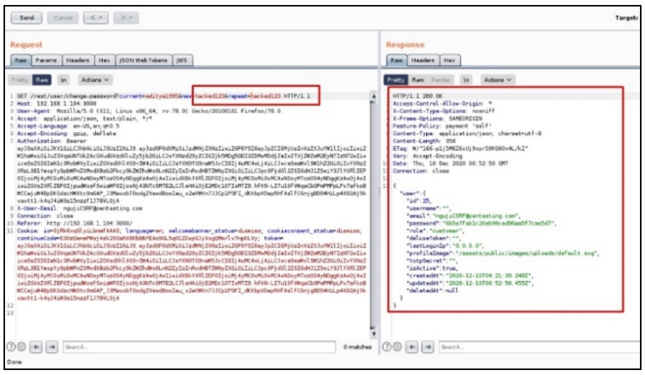


Fig. 5. Intercept GET request

The real user unknowingly sent an HTTP request that the attacker forged; the server responds with "200 OK" status that means the server correctly validates the fake request that has been sent, which can be seen in Fig. 6. The server cannot identify the forgery because the request was made by the user, which is authenticated and sends all necessary data.



```
GET /rest/user/change-password?current=adityal995&new=hacked123&repeat=hacked123 HTTP/1.1
Host: 192.168.1.104:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.[illegible]
X-User-Email: ngujiCSRF@pentesting.com
Connection: close
Referer: http://192.168.1.104:3000/
Cookie: io=[illegible]; language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
continueCode=[illegible]; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.[illegible]
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 358
ETag: W/"166-plj5M6Z6xUj9xpr59K060v4L/kI"
Vary: Accept-Encoding
Date: Thu, 10 Dec 2020 06:52:50 GMT
Connection: close

{
  "user":{
    "id":25,
    "username":"",
    "email":"ngujiCSRF@pentesting.com",
    "password":"6b5e7fab1c26eb98ced96aa5f7cae5d7",
    "role":"customer",
    "deluxeToken":"",
    "lastLoginIp":"0.0.0.0",
    "profileImage":"/assets/public/images/uploads/default.svg",
    "totpSecret":"",
    "isActive":true,
    "createdAt":"2020-12-10T04:21:39.248Z",
    "updatedAt":"2020-12-10T06:52:50.455Z",
    "deletedAt":null
  }
}
```

Fig. 6. Request Manipulation

The CSRF attack has been successfully carried out, and the attacker can log in with the manipulated password as shown in Fig. 7.

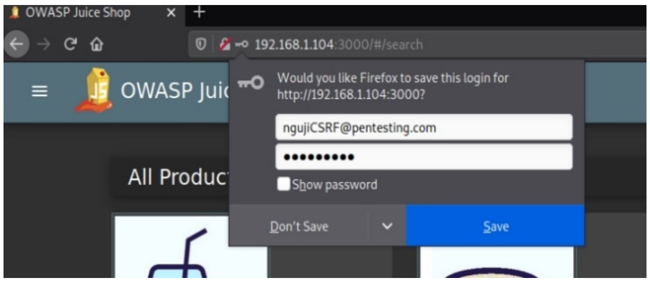

OWASP Juice Shop
192.168.1.104:3000/#/search
OWASP Juic
All Produc
Would you like Firefox to save this login for http://192.168.1.104:3000?
ngujiCSRF@pentesting.com
•••••••••
Show password
Don't Save
Save

Fig.7. Login with a new password

### 6. *Analysis of test results*

Penetration testing using the CSRF technique has been successfully carried out by forging requests from real users who cannot be identified by the server so that the server validates and responds correctly because it considers the fake request to come from a legitimate, trusted, and authenticated user. The attacker managed to take over access to the victim's account.

## V. CONCLUSION

JWT penetration testing has been successfully carried out with the CSRF technique so that the victim's account can be controlled and resources can be taken over. JWT token has become a good security standard for user authentication and authorization, but there is a vulnerability if it is stored in cookies. Web application developers have to do more coding. One of them is by implementing JWT on the HttpOnly cookie, which is a special type of cookie that is sent in an HTTP request to the server and can never be accessed from javascript running in the browser.

## REFERENCES

[1] R. Kannan and G. Umasankar, "Secure External Login Based on Authorization Code Flow using JWT," pp. 961–965, 2018.

[2] S. Rawat, T. Bhatia, and E. Chopra, "Web Application Vulnerability Exploitation using Penetration Testing scripts," *Int. J. Sci. Res. Eng. Trends*, vol. 6, no. 1, pp. 311–317, 2020.

[3] T. Farah, M. Shojol, M. Hassan, and D. Alam, "Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF," *2016 6th Int. Conf. Digit. Inf. Commun. Technol. Its Appl. DICTAP 2016*, pp. 74–78, 2016, doi: 10.1109/DICTAP.2016.7544004.

[4] F. Holik, J. Horalek, O. Marik, S. Neradova, and S. Zitta, "Effective penetration testing with Metasploit framework and methodologies," *CINTI 2014 - 15th IEEE Int. Symp. Comput. Intell. Informatics, Proc.*, pp. 237–242, 2014, doi: 10.1109/CINTI.2014.7028682.

[5] S. Shah and B. M. Mehtre, "An overview of vulnerability assessment and penetration testing techniques," *J. Comput. Virol. Hacking Tech.*, vol. 11, no. 1, pp. 27–49, 2015, doi: 10.1007/s11416-014-0231-x.

[6] Y. Balaj, "Token-Based vs Session-Based Authentication : A survey," no. September, pp. 1–6, 2017.

[7] A. Bhawiyuga, M. Data, and A. Warda, "Architectural design of token based authentication of MQTT protocol in constrained IoT device," *Proceeding 2017 11th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2017*, vol. 2018-Janua, pp. 1–4, 2018, doi: 10.1109/TSSA.2017.8272933.

[8] O. Ethelbert, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A JSON token-based authentication and access management schema for cloud SaaS applications," *Proc. - 2017 IEEE 5th Int. Conf. Futur. Internet Things Cloud, FiCloud 2017*, vol. 2017-Janua, pp. 47–53, 2017, doi: 10.1109/FiCloud.2017.29.

[9] G. Kbar, "Challenge Token-based Authentication – CTA," no. c, pp. 294–300, 2011.

[10] L. V. Jánoky, J. Levendovszky, and P. Ekler, "An analysis on the revoking mechanisms for JSON Web Tokens," *Int. J. Distrib. Sens. Networks*, vol. 14, no. 9, 2018, doi: 10.1177/1550147718801535.

[11] S. Calzavara, A. Rabitti, and M. Bugliesi, "Dr Cookie and Mr Token - Web session implementations and how to live with them," *CEUR Workshop Proc.*, vol. 2058, 2018.

[12] M. Haekal and Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," *2016 Int. Conf. Informatics Comput. ICIC 2016*, no. Icic, pp. 175–179, 2017, doi: 10.1109/IAC.2016.7905711.

[13] A. Rahmatulloh, R. Gunawan, and F. M. S. Nursuwars, " Performance comparison of signed algorithms on JSON Web Token ," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 550, p. 012023, 2019, doi: 10.1088/1757-899x/550/1/012023.

[14] M. I. P. Salas and E. Martins, "Security testing methodology for vulnerabilities detection of XSS in web services and WS-security," *Electron. Notes Theor. Comput. Sci.*, vol. 302, pp. 133–154, 2014, doi: 10.1016/j.entcs.2014.01.024.

[15] A. V. Barabanov, A. S. Markov, and V. L. Tsirlov, "Information Security Controls against Cross-Site Request Forgery Attacks on Software Applications of Automated Systems," *J. Phys. Conf. Ser.*, vol. 1015, no. 4, 2018, doi: 10.1088/1742-6596/1015/4/042034.

[16] C. K. Ho, "Descriptive Research for JWT Implementation as Session Data," no. March, pp. 1–9, 2018.