MOBILE BANKING – A CASE STUDY OF PRE-AUTHORIZATION AND POST-AUTHORIZATION ON SESSION HIJACKING

Abstract

With the extensive growth of smartphone communication modernization into the corporate world, in the current Mobile banking transaction is for the most part biggest and viable in monetary financial locale. In the mobile banking region concern most key variable is Authentication is a security cycle to characterize entree phases or client/client honors related to framework resources holding PC programs, records, applications, data and services. In this chapter we discuss Cookie based Session hijacking problem in man in the middle attack technique in mobile banking technology. Based on this problem we are tested to detection of the session hijacking problem in various operating systems and analyze each attack detection time. Then proposed two types of techniques on authorization, i.e. Pre-Authorization request and Post-Authorization request. This technique are helpful prevention of session hijacking in mobile banking technology.

Keywords: Authentication, Attacker, Session Hijacking, mobile financial, banking.

Authors

B Vishnuvardhan
Department of Computer Science
Kakatiya University
Warangal, Telangana, India.

B Manjula
Department of Computer Science
Kakatiya University
Warangal, Telangana, India.

I. INTRODUCTION

Mobile banking is currently prevalent and effective in the financial banking sector due to the widespread adoption of smartphone communication technology in the business world. The current mobile financial comprise of mobile accounting, mobile brokerage and mobile financial information services. Mobile banking accessible their clients every minute of every day. The inundation of cell phone use and the openness of the multitude of additional exceptional portable handsets and organization bandwidth have made cell phones an interesting to the opportunities to esteem added services.

The mobile banking security system design has been ordered into dual sorts in view of districts. First mobile security in the middle of between the cell phone and mobile operator (network organization), and the subsequent one is banking security in the middle of between the mobile operator (network organization) and the financial banking system [1]. The mobile banking security principally relies upon three elements; Verification (Authentication), Approval (Authorization) and transmission of Information.

1. Authentication interaction can be checked who is permitted to get to the information; his approved or not.
2. Authorization interaction can be checked methodology by which a web server concludes whether the user has approval to use an asset or admittance a report or record. In most cases, authorization is pooled with confirmation so that the web server can determine which client is requesting access.
3. The process of transmitting data should be secured to prevent data hacking by an attacker or hacker. An encrypted connection is required during this procedure. [1, 2].

Authorization: The Authorization process of defining admittance phases or client/user honors for system resources, which include PC programs, features of application, files, and information is known as authorization. This is one of the procedures for tolerating or declining admittance to an organization asset which allows the client to right to use to different resources in view of the client's uniqueness.

Hijacking wins when an outsider programmer or interloper assumed command over a session. At the point when a center man programmer (MIMA) enhances an appeal to the client, a correspondence begins and the client acquires the web session. Novel arrangement statistics and web session cookies are encompassed in the safeguard mechanism. When the client logs in to a service, for example, a session begins. Banking web application completes when one logout. The attacker's connection is then misinterpreted by the server as the genuine session of the original client. Hijacking of session is the charming benefit of a client session to get illegal admittance to its data. Malicious hackers delight in session IDs. With the session ID, the client can offer unaccredited admittance to a web server application and mirror a legitimate client. Session capturing, generally called man-in-the-middle attack that will provide a programmer full permission to a web-based account information. [1].

II. RELATED WORK

Sun's Net Dynamics application server platform was found to have a vulnerability in 2001 that gave users who authenticated with Net Dynamics a session id and a random unique

identifier. After the user logs in, this session identifier and id remain active for up to 15 seconds, and a consequent client can utilize those authorization to take over the logged-in financial account.

Garima Bajwa[3] suggest “to implement improvement instead of conventional authentication methods utilizing alphanumeric username and passwords, PIN numbers, or any combination thereof have many weaknesses, with Pass-Pic is to implement a picture based authentication system that is both more secure and easier for the user to both input and remember”.

Session cookies are used to accumulation the various activities of the users on a site, for a model, clicking buttons, sign-in a site. Here is the manner in which session cookies work in three phases [4]. Most importantly, the client contacts the web server interestingly; in the radiance of this requesting for the content of page, the server creates a session-ID (identifier), which will be significant for the cookie. Second, the server sends the cookie to the client, as an element of the headers of the site page; the cookie is then taken by the client's program [1, 5]. Third, the site content of page is gradually recuperated from the web financial server.

Hijacking of Session done MITB [6] to perform by sending Trojans in fatality's framework. While the user are endorsed in to claim records, there are redirected to malicious locales. A unique meeting laying hold of is a famous MITM (man-in-the-center) assault in the space of web application security, and it's potentially the extreme-regarded assaults for the aggressors considering the chance of the attack.

Hijacking of session, the intruder imitates the target's character and gains access to the victim's assets in the same way [7]. The implications can be overwhelming because they may lead to the abandonment of definitive info. In this manner, session capturing has reliably been a point of convergence researchers who consider methodology to prevent and further develop session hijacking [8].

III. PROBLEM STATEMENT

Most of available Web site procedure algorithms based on simply expectable variables, for example, IP address to create the IDs of session, affecting them to be predictable. If encryption is not used like SSL, session IDs are transferred in the clear and are vulnerable to eavesdropping.

An attacker can hijack a legitimate client's session by using secured brute force or figured session IDs to take control of the session while it is still in the cycle. In the standard of purposes, after tolerably seizing a session, the attacker gets all out admittance to the whole of the client's information and is supported to execute undertakings as a choice of the client whose session was caught. IDs of the session can in like manner be stolen using content infusions, for instance, cross site scripting. The client plays out a malignant substance that readdresses the confidential client's data to the attacker [1].

IV. PROPOSED METHODOLOGY

Hijacking of the Session is a methodology used to take command for one more user's session and attained unapproved admittance to information or resources. A token of the session is for the most part ready of a line of changeable size and it might be used modified procedures, as in the URL, a cookie of HTTP request header as, in various bits of the HTTP request header, or such a long way in the HTTP request body. Through enumeration and analysis on the client's website was suggested first step. Thusly, we fostered a per-authorization and post-authorization for the distinguishing and forestalling the session hijacking to protect single assets from the unapproved client [1].

1. **Pre-Authorization Request:** We fosters the programs to make Web financial application, this application is runs on the web financial server.

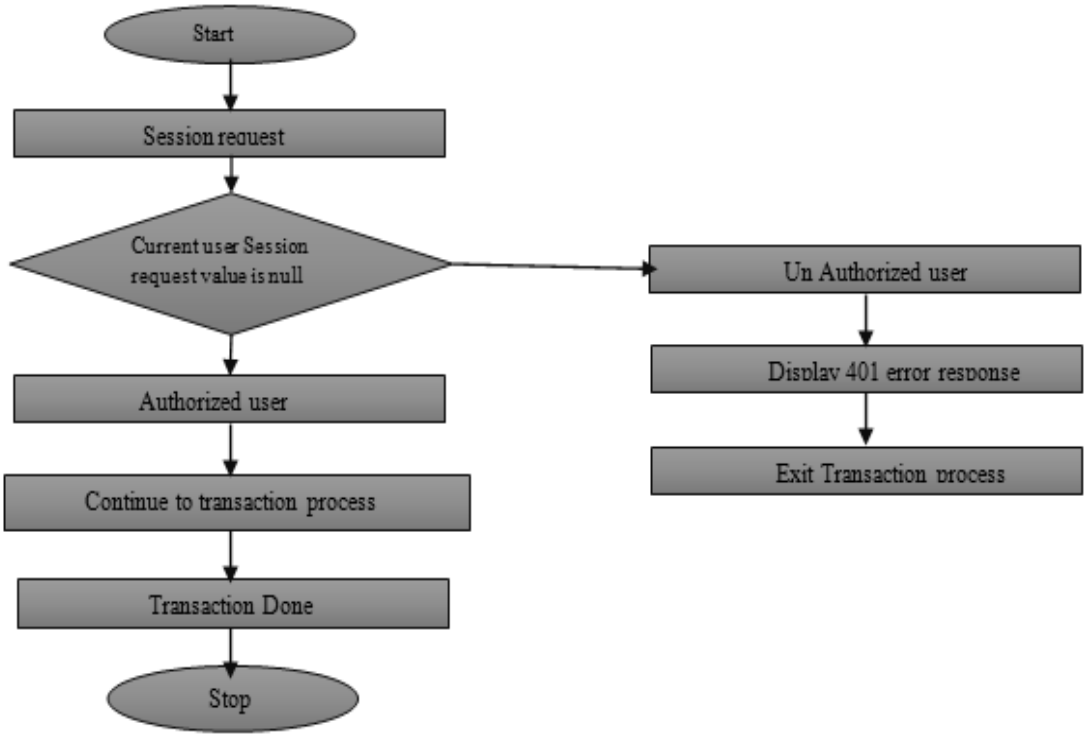

Figure 1: Pre-Authorization Request

Based on the above figure.1, we designed a structure to keep customers safe from hackers when it comes to mobile banking. Individuals uninformed about programmers inclined to numerous issue in regards financial transactions. We have tried to categorize it with the program to protect their account and money. Our plan start that when a customer opens the mobile banking, the bank web server caution and responses the user. When the user enter the client id and secret password, the server makes the session-id. The session-id would be used during the transaction process. A third-party hacker might be able to steal the session-id, making the transaction straightforward for the hackers. Therefore, in order to eliminate or resolve this issue, we have programmed the program so that the server verifies whether the current session value is null or not each time the customer logs in using their user

id and password. On the off chance that the client's ongoing session esteem is null, the web server communicates something specific "it is unauthorized user" and show 401 error message response and consequently leaves the financial transaction cycle [1].

In the event that the client current session esteem isn't null, then, at that point, it proceeds with the financial transaction cycle and financial transaction would be finished. This software package is enthusiastically suggest in light of the fact that it can protect our cash and keep us from turning into the casualties of outsider hackers.

2. **Post-Authorization Request:** We planned a program to eliminate the issue made by programmers to defend our cash and secure our account. As per our design, while a client directs a request for the process of financial transaction, the bank web server authorization whether the request is with respect to the session request or Web API request. The state of the session's behavior is set if the web session shows as a Web API request. In Web API request the conduct would be read only state. In this specific request the client will actually want to read only, couldn't compose or handle anything. When the state is just reading the web session exits process of financial transaction consequently [1].

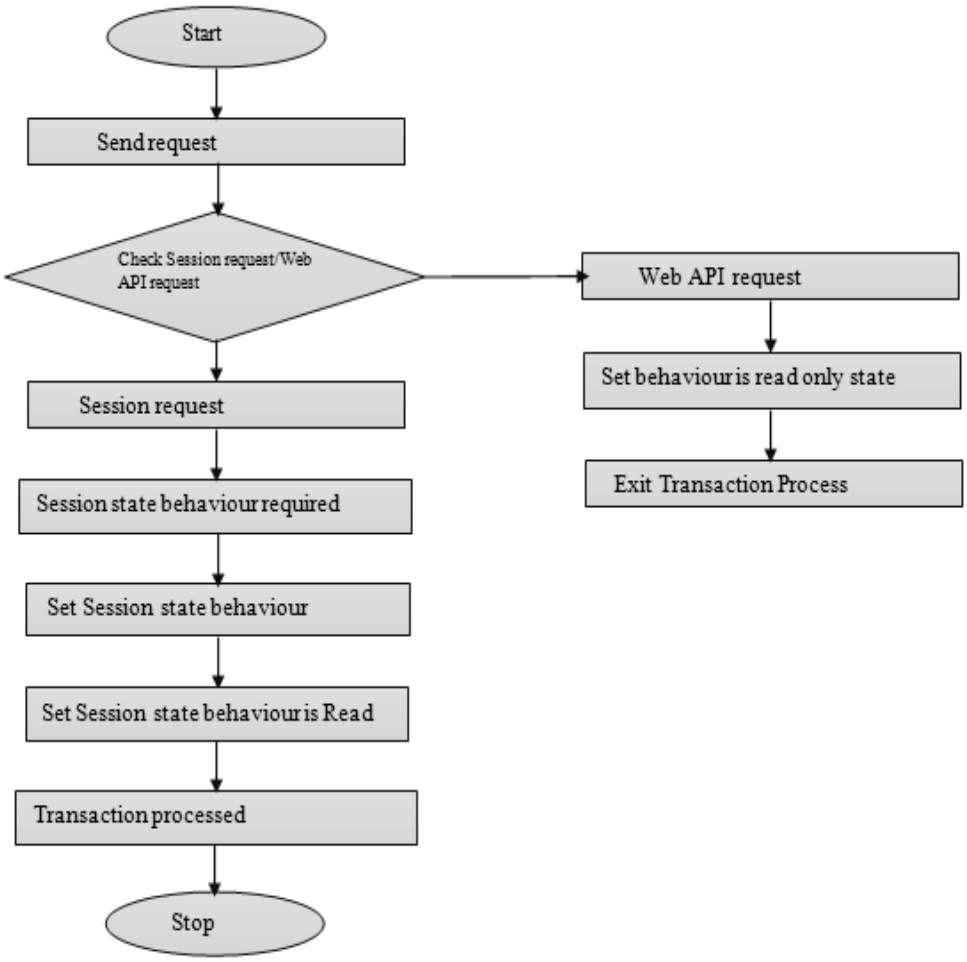

Figure 2: Post-Authorization Request

On the off chance that the appeal is shown as the session request, the web session would express the way of state behavior required. The read and write states of the session's state behavior set are then set. Because the user has been authenticated, he or she will be able to use both the read and write permissions in this session and move on to the transaction process. After that, the procedure comes to an end. The below figure. 2 plan assists clients with forestalling individuals by turning into the survivors of the man in the middle aggressors.

V. RESULTS AND ANALYSIS

The design of the system for the process was unique to the network strategy. In order to test the recommended structure, a client-server application has been developed by tools are:

- Visual Basics Studio
- Wireshark
- Apache Tomcat Server
- Kali Linux
- HTML & JavaScript for Webpage development

In the Network approach, to create a raw socket application using visual basics and the application developed in .Net. The application of raw socket will provide this method with the necessary benefit of generating a custom packet. This custom package have an effective IP address but a false transmission address such as "FF:FF:FF:FF:FF:FF:00", If a host accepts this false diffusion package, then the user will be notified that there is a host, including the interface of network card in unrestrained mode.

1. Session Hijack Detectability Rate in A Linux Based Ubuntu and Windows 10 Pro Machine

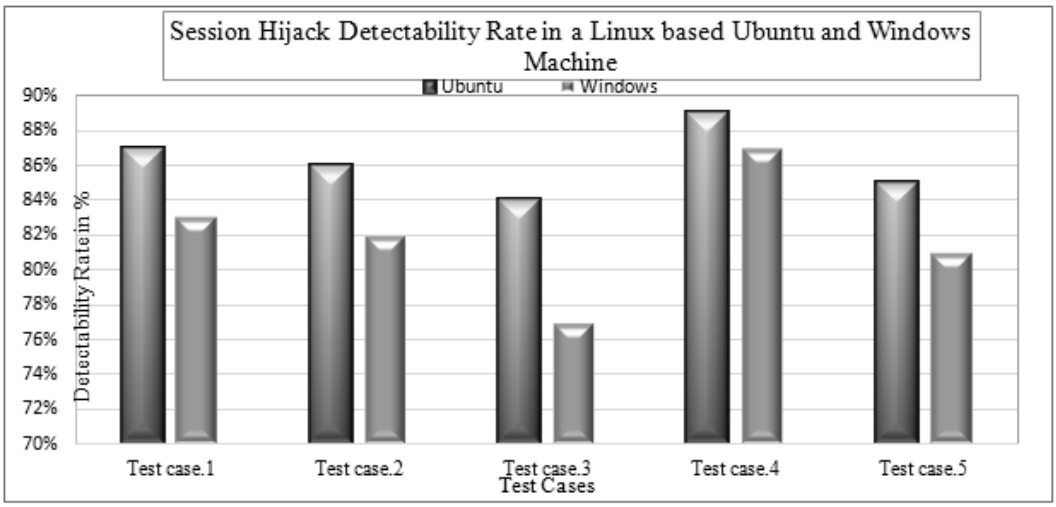

Figure 3: Session Hijack Detectability Rate in Ubuntu and Windows Machine

To show the detectability rate in session hijacking, in figure. 3, which represents the ability of each system in detecting the session hijacking rate. This graph representation helps the users understand the effective work of each system. In this scenario we take two operating systems, one is Linux based Ubuntu and other is windows to check the detectability rate of session hijacking firstly, when the client login, a session id is created and within the fraction of seconds the session id saved in a cookie. The man in the middle attacker tries to steal the session id through the cookie which helps to do the fraud transaction easily. So we took five test cases to check the hijack rate based on different parameters in each test.

In the first test case, the detectability rate by Ubuntu is 87%, whereas windows is 83%. There are around four percentage of difference. In more change when compared to the first case. But according to the third test case difference is massive when it is compared to all test cases. The difference between two operating systems is 7%. In out of five test cases the fourth case detectability rate is too high in both systems. Finally, in the fifth test case, the rate of Ubuntu is higher than windows. This is presented as a ratio unit. It also gives us a virtuous sense of the platform that gives us the litheness to set up tailored mechanisms. Although the basic TCP stack procedure is the same for Linux based on Ubuntu and Windows, the kernel is web programmed differently; there are safety disadvantages which prevent us to develop an effective detection tool. It is very clear that in each and every test case of Ubuntu is highly effective then windows in detecting the session hijacking, and it capture very easily them the other system.

2. **Each Attack Detection Time in Session Hijacking:** With the help of a figure. 4, analyse the time of each attack detection time in milliseconds. There are various attacks in the session hijacking process. But we have taken nine different attacks to show the time detection that took place in milliseconds. In the graph representation shown, in all the nine attacks, third attack took more time i.e., 4.5ms, whereas the sixth attack took only 2ms in detecting the attack. Remaining seven attacks there are some fluctuations maintain the time during the attack scenario. The first attack took 3 milliseconds, the second took 2.4 milliseconds, and the fourth took 3.9 milliseconds. The same happened with the other attacks also.

We have discussed nine attacks in session hijacking. Each of attack detection time is very low means attacks are done very fast. By this we can say that the man in the middle attack takes very less time to hijack the sessions. We can estimate the accuracy of the hijacker, the man in middle attacker takes only milliseconds in the session hijacking, and it is very risky to banking users. The time devoted to each attack was repeated for nine different diversion attempts, the graph clearly displays the duration of each attack in milliseconds. In conclusion, we can say the speed of the hijacker in hijacking the sessions and user banks accounts which is very easy and steal accounts information. This demonstrates that the detectability rate is significantly higher and that it takes less time to detect each attack.

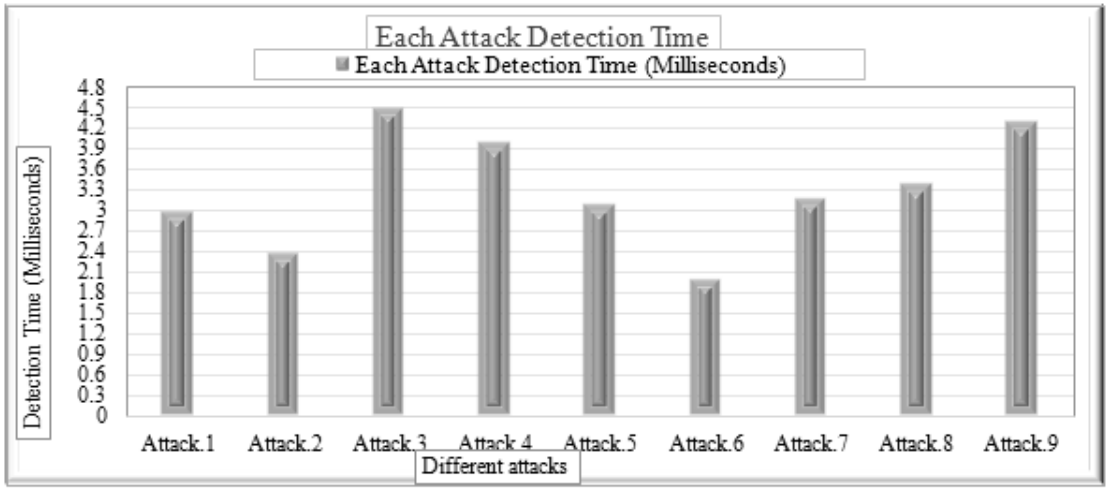

Figure 4: Each Attack Detection Time

3. Prevention of Session Hijacking

```
INFO: Session is authorised, continuing with normal operation
April 1, 2021 5:46:21 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 1982 ms (/SessionHijack/login.jsp)
April 1, 2021 5:46:30 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:31 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 185 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:46:32 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:32 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 6 ms (/SessionHijack/login.jsp)
April 1, 2021 5:46:33 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:33 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 1 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:46:35 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:35 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 1 ms (/SessionHijack/login.jsp)
April 1, 2021 5:46:41 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:41 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 5 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:46:43 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:43 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 7 ms (/SessionHijack/login.jsp)
April 1, 2021 5:46:44 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:44 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 3 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:46:45 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:46:45 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 1 ms (/SessionHijack/login.jsp)
April 1, 2021 5:46:58 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:47:00 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 4 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:47:04 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:47:04 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 2 ms (/SessionHijack/sessionaction.jsp)
April 1, 2021 5:47:06 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is unauthorised, terminate with no operation
April 1, 2021 5:47:12 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:47:12 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 9 ms (/SessionHijack/login.jsp)
April 1, 2021 5:47:15 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Session is authorised, continuing with nomal operation
April 1, 2021 5:47:15 PM com.tomacat.session.valve.sessionUnHijackValve invoke
INFO: Request took 3 ms (/SessionHijack/sessionaction.jsp)
```

Figure 5: Description of Secure Sessions of Users

We have developed two innovative algorithms when the consumer logs in the account using credentials the session id is created. Many of hijacking takes place the man in the middle attack hijacks users account sensitive information through session ids. For this we have introduced pre-authorization algorithm and post-authorization algorithm. In pre-authorization algorithm, it works based on null value detection. When the user try to sign in through the user id and password, then the server authorisations if the session request value is null or other. If it is null value, then it shows unauthorized user and terminate the user immediately. It means some attackers to hijack the session and steal session ids and login through session id. If the original user have username and passwords. In this method when the user try to login if the session request value is not null, then the process further continues and the financial transaction process takes place. In post authorization algorithm, works on checking whether request is WebAPI or SessionAPI. It is design when the client enters the session of the transaction, the certifications given by the client is correct yet enter through WebAPI, then it upholds WebAPI which permits the clients just read, it cannot do any transaction interaction process. If the user reach to server through SessionAPI, then the server gives full mode authorizations like read and write, which means to do transaction process.

In the above figure.5 shows algorithm implementation process. It clearly shows the server log file that demonstrations the info about logged hijacking efforts and the consistent time engaged to record this information. It depicts when the client enter client name and secret key the server checks whether the server credentials are matched then give data is session is authorized, proceed to typical activity, it implies clients enter to server and shows login date and time as well as client request taking time is 1982 ms. Then second request is when the user already enter the server checks user API, it is SessionAPI so give permission to do transactions and this request takes time is 185ms. Likewise, the user can log in many times or many times to do the transactions process is shown with complete details like request time, etc. In this way, multiple users enter to server and to do their transaction scenario is displayed. At the time of 5:47:06 pm one of the user to enter the server but the server checks user credentials are mismatch so then server can terminate the user and displayed an unauthorized user.

VI. CONCLUSION

In this chapter, we discussed security factors and analysis. We designed Pre-Authorization and Post-Authorization algorithms on the investigational basis. The results of the algorithms are explained and represented through graphical representation. In session hijacking, experimental result analysis discussed and proved null value based algorithm or method. This technique are helpful prevention of session hijacking in mobile banking technology.

REFERENCES

[1] B. Vishnuvardhan, B. Manjula, “Pre-Authorization And Post-Authorization Techniques For Detecting And Preventing The Session Hijacking”, International Journal of Future Generation Communication and Networking Vol. 14, No. 1, (2021), pp. 359–371.
[2] Vishnuvardhan B., Manjula B., Lakshman Naik R. (2020) A Study of Digital Banking: Security Issues and Challenges. Proceedings of the Third International Conference on Computational Intelligence and

Informatics. Advances in Intelligent Systems and Computing, vol 1090. Springer, Singapore. https://doi.org/10.1007/978-981-15-1480-7_14.

[3] Garima Bajwa, Ram Dantu and Ryan Aldridge, (2015), "Pass-Pic: A Mobile User Authentication", 978-1-4799- 9889-0/15/$31.00 ©2015 IEEE.

[4] Xiaofeng Zheng, Jian Jiang, Jinjin Liang, Haixin Duan, Shuo Chen, Tao Wan, andNicholas Weaver. 2015. Cookies Lack Integrity: Real-World Implications. In24thUSENIX Security Symposium (USENIX Security 15). USENIX Association, Wash-ington, D.C.

[5] Rodica Tirtea. 2011. Bittersweet cookies. Some security and privacy considerations— ENISA. https://www.enisa.europa.eu/publications/copy_of_cookies. (Accessedon 02/10/2019).

[6] T. Analysis, "Making Sense of Man- in-the-browser Attacks: Threat Analysis and Mitigation for Financial Institutions:2010,"[Online].Available:http://viewer.media.bitpipe.com.

[7] D. R. Sahu and D. S. Tomar, "Strategy to Handle End User Session in Web Environment," Proceedings of National Conference on Computing Concepts in Current Trends, NC4T'11 11th & 12th Aug. 2011, Chennai, India.

[8] Marjani Peterson, "Investigation of cookie vulnerabilities: poster", WiSec '19: Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile NetworksMay 2019 Pages 330–331. https://doi.org/10.1145/3317549.3326316.

