iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** VIII **Month of publication:** August 2020

DOI: https://doi.org/10.22214/ijraset.2020.31099

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VIII Aug 2020- Available at www.ijraset.com

Enhancing Session Security on Browser with Disposable Credentials using OTC

Niki Modi[1], R. R. Sedamkar[2]
[1]*M.E. Second Year in Computer Engineering,* [2]*Professor (Department of Computer Engineering), Thakur College of Engineering and Technology Mumbai, India*

Abstract: ***Many web applications are vulnerable to session hijacking attacks due to the insecure use of cookies for session management. The most recommended defense against this threat is to completely replace HTTP with HTTPS. However, this approach presents several challenges (e.g., performance and compatibility concerns) and therefore, has not been widely adopted. In this paper, "One-Time Cookies" (OTC), an HTTP session authentication protocol for improving session hijacking features, easy to deploy and resistant to session hijacking. OTC's security relies on the use of disposable credentials based on a modified browsers name. Experiments demonstrate the ability to maintain session integrity with a throughput improvement over HTTPS and a performance approximately similar to a cookie-based approach, Here I have Created web configuration page based on that it will fetch IP address, After that based on each session OTC will be generated, In doing so, I demonstrate that one-time cookies can significantly improve the security of web sessions with minimal changes to current infrastructure and browser page.***
Keywords: session hijacking, https, one time cookie (otc), security, disposal credentials, IP Address, Web Configigruation.

I. INTRODUCTION

HTTP cookies are the de facto mechanism for session authentication in web applications. However, their inherent security weaknesses allow attacks against the integrity of web sessions. HTTPS is often recommended to protect cookies, but deploying full HTTPS support can be challenging due to performance and financial concerns, especially for highly distributed applications. Moreover, cookies can be exposed in a variety of ways even when HTTPS is enabled. In this paper, we propose One-Time Cookies (OTC), a more robust alternative for session authentication. Each time you surf the Internet, your machine communicates with thousands of routers and servers in the world. Internet can be used for various purposes like social networking sites, online transactions, online shopping, etc. So there is constant exchange of information over the Internet means it is open to threats and vulnerabilities. As a result, it has led to increase in cyber- crime. Hackers are getting better and better at penetrating systems nowadays. There are various types of attacks a hacker or an attacker would perform on internet. While simple and scalable, this design makes the creation of applications requiring the association of multiple transactions to a single user (e.g., banking) somewhat difficult natively. HTTP cookies, which generally contain one or a small number of short identifier strings allowing a server to associate seemingly unrelated requests, rapidly became the dominant mechanism for web session management. Unfortunately, the use of cookies introduces a number of security risks, especially when they are employed as session authentication tokens. As an example, many websites rely on strong security mechanisms such as HTTPS (i.e., HTTP over TLS/SSL) to initially authenticate a user. During this secure session, the server generates cookies that the user can later employ as lightweight authentication tokens.

II. LITERATURE REVIEW

The use of cookies as session authentication tokens has raised a lot of security issues. Several surveys [2,3] have demonstrated multiple problems with web authentication mechanisms, including susceptibility to session hijacking attacks. As a , security researchers have proposed changes to improve the robustness of authentication cookies. Park et al. [4] and Fu et al. [2] suggested cookie mechanisms that provide better confidentiality and integrity guarantees by using well-known cryptographic techniques. In addition, these authors have proposed the use of cookie expiration time to reduce the impact of session hijacking attacks. However, many applications use long expiration time to avoid affecting a user's experience, reducing the effectiveness of this approach. Juels et al. [5] proposed the use of cache cookies, different forms of persistent state in the browser (e.g., browser history and temporary internet files) as an alternative to cookies for storing user and session identifiers. While resistant to pharming attacks, cache cookies still need HTTPS protection to prevent active attacks. Moreover, HTTPS only protect cookies on the network. An adversary can also steal cookies from a user's computer through many different attacks (e.g., cross-site scripting attacks [6], cross-site tracing attacks [7], and domain-related attacks [8]). Always-on HTTPS is the most recommended defense against session hijacking.

To secure communication in an Internet session, cryptographic techniques, such as one-way hash chain (OHC) technique that relies on one-time passwords proposed by Lamport [9], have been utilized. In particular, the OHC technique has been employed in many applications with the aim of mitigating the potential of session hijacking. For example, the authors in [10] proposed using OTC where disposable credentials called OTC replace authentication credentials. To protect a user's session, [9] implemented a framework that ties a session to a current browser by fingerprinting and monitoring an underlying browser, its capabilities, and detecting browser changes at the server side. The OTC scheme generates a set of tokens that are only used once and discarded once used. In [11], a hybrid scheme was proposed that utilizes one-way hashing and sparse caching techniques, but practically it is not implementable; their research focuses only on hashing..

None of the previously described mechanisms have been widely deployed. While many of them prevent session hijacking, they fail to address the requirements of highly distributed web applications, particularly requests' statelessness. Consequently, most web applications have chosen always-on 5 HTTPS as the main defense against session hijacking attacks. However, always-on HTTPS may be difficult to deploy, particularly in large web applications not originally designed for such requirement. Always-on HTTPS not only affects the performance (e.g., added cryptographic overhead and web caching mechanisms do not work with HTTPS) but also impacts existing functionality (e.g., virtual hosting, applications [12], and network content filtering [13]). Therefore, to effectively prevent session hijacking attacks, a more robust, efficient and practical alternative to is needed.

III. OBJECTIVE

In this paper, I have built a prevention technique for session hijacking. In this technique, we bind the network layer and application layer together through reverse proxy server. This reverse proxy server will generate session credentials such as session ID, IP, technique. This mechanism detects the change in browser due to which an adversary cannot get the illegal access. Since users are bind with machine and browser and with new disposable cookie for each request in the session. Session hijacking can potentially take place on several levels of the OSI model (possibly all), as well as outside of the network.

1) *Physical:* Tap someone's physical connection, and send all packets to the MiTM.
2) *Data Link:* ARP poison someone's Ethernet connection, and send all packets to the MiTM
3) *Network:* Manipulate the packet routing, and send all packets to the MiTM.
4) *Transport/Session:* A secure protocol such as SSL/TLS will protect against compromise of the data, but if an attacker has also broken TLS/SSL, then a break at this level would break the protection from compromises at lower levels.
5) *Presentation:* I can't think of anything at this level, and it doesn't map well onto TCP/IP and protocols, but that doesn't mean it's not possible.
6) *Application:* You might debate about this, but I'd argue that CSRF, Code injection, and XSS are all at the Application level.
7) *Outside:* Any compromise of the machine itself that can grab a session key and transmit it to an attack, be it physical, OS or some other application would be outside of the OSI model.

IV. SCOPE OF SESSION HIJACKING

All the mentioned factors play a crucial role in the success of Session Hijacking:

1) Weak session ID generation algorithm: Most websites are using linear algorithms based on easily predictable values such as time or IP address for generation of session ID.
2) Indefinite session expiration time: The session ID's that have an indefinite expiration time provides an attacker ample time to guess a legitimate session ID.
3) Clear text transmission: The session ID is often sniffed across a network easily if the SSL is not being employed while the cookie is transmitted to and from the browser.
4) Small Session ID: Although cryptographically a robust algorithm is used, a legitimate session ID may be determined easily if the length of the string is small.
5) Insecure Handling: An attacker will retrieve the stored session ID information by misleading the user into visiting a malicious website. Later the attacker can exploit the information before that session expires.
6) No account lockout for invalid session Ids: If account lockout function is not implemented on the website, the attacker can try a number of attempts with varying session Ids until the actual session ID is determined.

V. SESSION HIJACKING PROCESS

(MITM is Man in-the middle Attack)

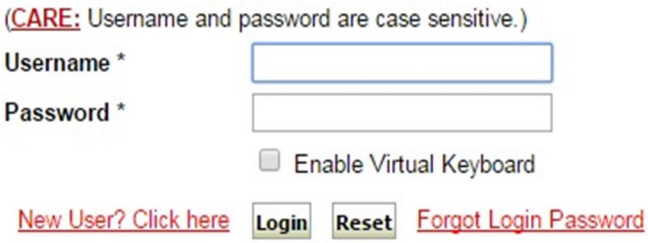
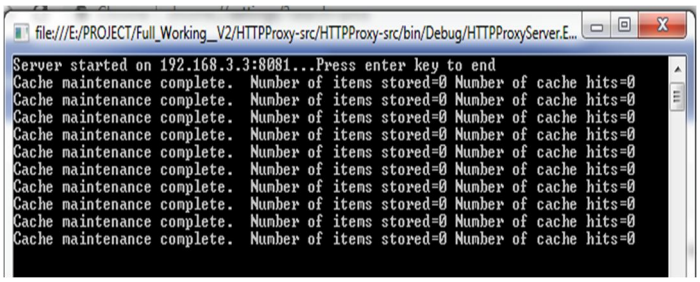


Fig. [1]: Session Hijacking Attack

1) After login, all requests are sends to the web application using a cookie for authentication as shown in Fig [1].

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ListeningIPInterface" value="192.168.3.3"/>
    <add key="ListeningPort" value="8081"/>
    <add key="CertificateFile" value="cert.cer"/>
    <add key="ConnString" value="Data Source=DELL-PC\SQLEXPRESS;Initial Catalog=Shopping_db_V2;Integrated
Security=true;"/>
  </appSettings>
</configuration>
```



Fig[2]: HTTP Request Received.

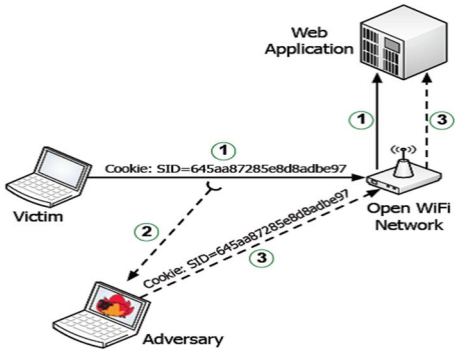


Fig.[3]: Session Hijacking Attack Process in Web Application.

2) Because this request is sent over unsecured protocol, an adversary can eaves-drop the request and capture the encrypted cooking as shown in Fig[3].

3) Finally, the adversary can use this cookie to send arbitrary requests to the web application, hence hijacking the victim's session.

VI. TYPES OF ATTACK

There are two types of attack happen during communication.

A. Online Attack
B. Offline Attack

Online attack refers to communication with an entity under attack, that must be online (and participating) during the attack.
An attack using a coalition of adversaries communicating online, with little or no communication with the entity under attack (if any), is an offline attack requiring online communication.

There are two ways of online attack.

1) First, they are limited by the speed of the network. Each username/password combination has to be sent over the network to the authentication server and then the server responds accordingly. This time it takes for this back and forth transmission depends widely on the speed of the application server and the speed of the network, but a typical password attack can only get around 3 – 5 login attempts per second.
2) The second way online password attacks are limited is that they are extremely noisy. When we are attempting 5 logins every second for an average password dictionary (around 10,000 passwords), this is likely going to be flagged by almost any type of logging and alerting mechanism. Additionally, most applications are protected with account lockouts. When a password is guessed incorrectly a certain number of times in a row, it may lock out the targeted account, block the attacker's IP address, or both.

An offline attack require work from the attacker only (or mostly), with no (or little) communication with the system (e.g. server) under attack (holding the key). An offline password attack will take this hash offline and try to find the clear-text value that computes to that hash. To do this, an attacker will use a computer (or a beefed up computer) to take passwords, compute the hash, and compare them very quickly. This will be performed over and over again until a match is found. For Example: parallel hash collision search is an offline brute-force attack.
The attack against HMAC-MD5 that asks for the MAC of random messages ending with the same block until a collision is found (requiring about 264264 queries), then modifies the last block of the two colliding messages to (likely) get a new collision allowing a MAC forgery, is an online brute-force attack, since there is massive work involving communication with an entity capable of computing MACs (holding the key).
An adversary with limited access can post a script on a webpage (e.g. via cross site scripting XSS) and wait for the genuine user to access the infected website. When the user opens the page, the malicious script executes automatically and gains access to the decrypted credentials. Such a script often tries to recover the session ID and discretely communicates it back to the adversary. A variation script from within the browser.

VII. DESIGN SYSTEM

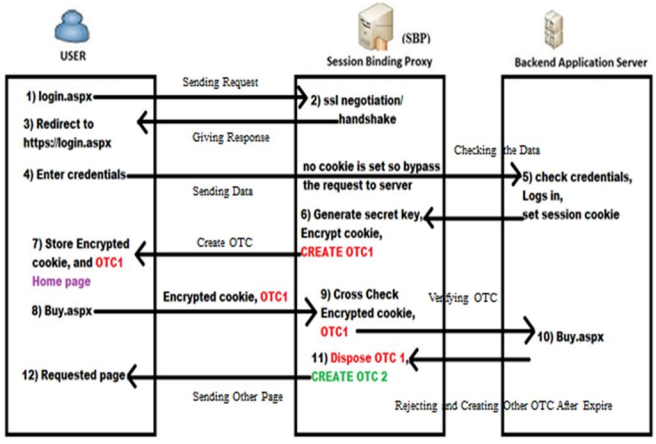


Fig [4]: Preventing Session Hijacking Attack using Proxy Server and OTC

A. To handle the sessions, the reverse proxy needs to be extended with functionality to read the requests and responses and manage the SSL/TLS session and application session. The proxy stores the SSL/TLS session and application session combination in its own memory. The public key of the client should be enough to authenticate the client.

B. You encrypt the application data sent to the client with this key and the incoming requests are encrypted and can only be decrypted with the public key of the client. If you intercept the "set-cookie" header sent by the application server, you can also read the application session status.

C. When a request comes in, the cookie 8 header must be read and checked against the key value pair that is stored in the proxy. If the public key, session id pair of the request does not match one in the local database of pairs, the session is invalid. To invalidate the session on the application server, the invalid request can be sent to the server without the cookie header.

D. The server will then return the login page. In practice, the client's public key cannot be requested from the SSL suite that implements and handles the SSL connection as shown in Fig[3]. The suite does provide an SSL session id value. This value is a unique identifier of an SSL session, but it does not identify a client.

VIII. ARCHITECURE AND METHODOLOGY

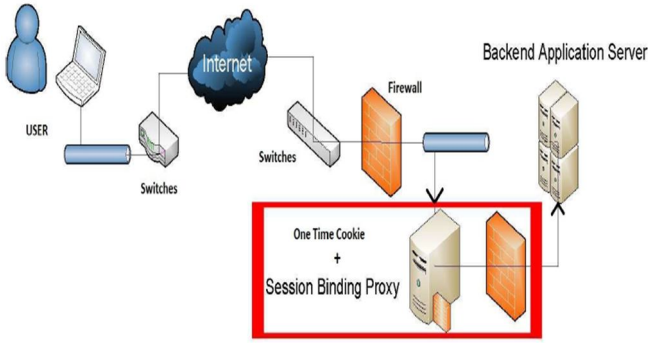


Fig [5]: Architecture of Session Hijacking Attack using Proxy Server

Our propose a method that combines SSL/TLS session-aware authentication with a reverse proxy. It is much like the method Rolf Oppliger et al. proposed. Instead of implementing it inside the application, we want to implement this inside a simple reverse proxy. This proxy relays the requests to the backend server only if the client that originally got the application session id is sending the request. To authenticate a client over HTTPS, you register the SSL session and application session information.

When a request with the same application session id is used with a different SSL session, you know that the session is stolen. By removing the session cookie from the request, the application session is invalidated. The proxy makes sure the HTTPS session and application session combination does not need to be kept inside the application (server). The idea is to use a server side reverse proxy that handles the HTTP(S) requests as they come in and sends them to the back end application server as shown in Fig[5].The application server should only be accessible internally and not from the Internet.

The extensively used HTTP works in a request–response fashion. First, a client sends a request to a server. Next, the server processes the request sent by the client and sends back a response to the client. After this, the connection between the client and server is dropped and forgotten since HTTP is stateless, i.e., the server cannot differentiate between different connections of different users. An HTTP server treats each request independently of any previous requests. However, many web applications built on top of HTTP need to be stateful.

IX. FLOW OF IMPLEMENTATION

1) First I created one web configuration page in that I have created IP address.

2) It will match to server page after that server will start as shown in Fig [2].

3) After Server will accept the request and it will fetch the data from given URL.

4) If IP address match then web page will open that request from any URL from Browser

5) In Session binding, I have created one query based on OTC,It will accept the request and OTC will be generated as shown in Fig.[10]

X. RESULT AND DISCSSION

A. First it will match the IP Address if its proper then only it will connect to the server.
B. After that Port number should be different i.e.8081 same port number cannot acknowledged the packet
C. Open SSL to implement https connection.
D. Created application instances for reverse binding proxy.
E. Incorporate OTC in reverse proxy instances to handle each request.
F. Provide flexibility to use Https connection.
G. Encrypt/ Decrypt SID by using AES function (Rijndael)
H. Used SHA256 for creating HMAC of password required as KEY for the AES function.

In This paper we are presenting Experimental Evaluation of our implementations. Our goal is to characterize and compare the performance overheads added by OTC and current session authentication alternatives (e.g. cookies and cookies with HTTPS)

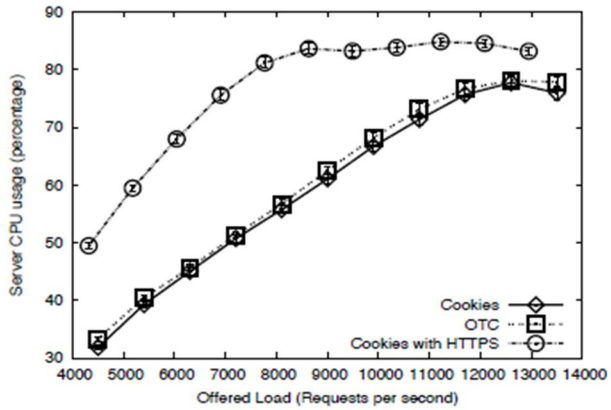
Fig [6]: Flow diagram of a web session authenticated with OTC. Messages 1 to 4 show the OTC setup phase and messages 5 to 8 show the OTC authentication phase. Each HTTP request and response includes an OTC header with protocol information.

1) C→ S : url
2) S→ C : [X-OTC:0, n, login-url]
3) C→ S : uid, pwd, [X-OTC-CRED:n,Hn(r), s,nonce]
4) S→ C : [X-OTC:1, uid,n − 1, nonce, HMAC(s, uid||n − 1||nonce)]
5) C→ S : [X-OTC-VAL:uid,Hi(r), i, nonce,HMAC(s, uid||url||Hi(r)||)]
6) S→ C : [X-OTC:2, i − 1, nonce, HMAC(s, i − 1||nonce)]

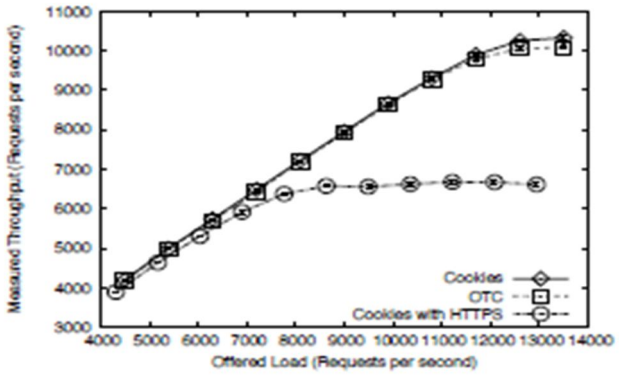
a) C, S : browser, Web application
b) uid, pwd : username and password
c) r, n, i : Hash chain secret seed, length and current
d) sequence number
e) s : shared session secret
f) url : URL of the resource requested by the client
g) login-url : URL of the application's login service
h) Hi(x) : i-th hash value of x, H(H(...H(x)...))
i) HMAC(k, x) : HMAC with key k on x
j) X-OTC* : HTTP headers used for exchanging
k) OTC protocol information

One-Time Cookie protocol: Formal definition of the OTC protocol. OTC assumes that the setup phase (steps 1 to 4) is executed over an encrypted connection (HTTPS)

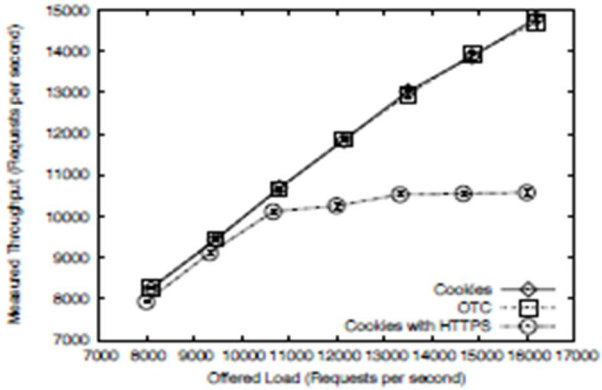
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue VIII Aug 2020- Available at www.ijraset.com



Fig[7]: Web server CPU utilization for cookies, OTC and cookies with HTTPS configurations. As expected, the use of HTTPS requires more CPU time for the same load than cookies or OTC cally significant. The impact of HTTPS was more noticeable under higher test loads.



Fig[8]: Request throughput supported by the web server for cookies, OTC and cookies with HTTPS configurations. While OTC and cookies have approximately the same performance, the use of HTTPS considerably reduces the throughput the web server can support.



Fig[9]: Request throughput supported by the web server in the presence of a reverse proxy for 3 configurations: cookies, OTC and cookies with HTTPS. While OTC and cookies benefit from the use of the reverse proxy, the performance of HTTPS quickly degrades.

PageUrl	OTC	Date
http://192.168.3.3/Laptop.aspx	2bba799a1f9abb5226d1ca436c1058297c889872b14c9e16...	2020-07-22 17:09:02.150
http://192.168.3.3/Desktop.aspx	3260cf29f357bd8e5271c7851416970b72dc9248b66df3a42...	2020-07-22 17:09:08.090
http://192.168.3.3/Login.aspx	609c1fae23ccbd7990e2821fd63b510e0c2cb19ba96f39877...	2020-07-22 17:10:18.937
http://192.168.3.3/Laptop.aspx	b031db977f136dbdead4565a99952776b5289a1266bdc12e...	2020-07-22 17:16:33.237
http://192.168.3.3/Login.aspx	1766c8510ae05c774b08674b05398ef2776a5b8bb0c6a4fc2...	2020-07-22 17:16:38.880
http://192.168.3.3/Login.aspx	27adb82cecff7e2513670ea4a7ab97123f8cc295f2429675b5...	2020-07-22 17:17:20.207

Fig[10]:Actual Outcome From Session Binding Query Fetching Data,Creating OTC Based on Time Stamp & URL Path From Current Browser.

XI. CONCLUSION

The main purpose of our paper is to Application and N/W session is binded in reverse proxy by using IP address & SID. Attack is protected if OTC and encrypted Session credential are sniffed. Since OTC can't be reused and session credential is binded hence IP address gets changed if adversary try to hijack the session. The experimental evaluation of our implementation. Our goal is to characterize and perform overheads added by OTC and current session authentication alternatives (e.g., cookies and cookies with HTTPS).

XII. FUTURE SCOPE

With advancement in IT technology, Implementation can be easily extended with existing and future fingerprinting methods, e.g., text font rendering or JavaScript engine fingerprinting. This work can also be implemented in the framework could be extended by supporting CSS selector and CSS filter fingerprinting in the future. I am planning to implement HTML5 fingerprinting as an asynchronous checker in the near future.

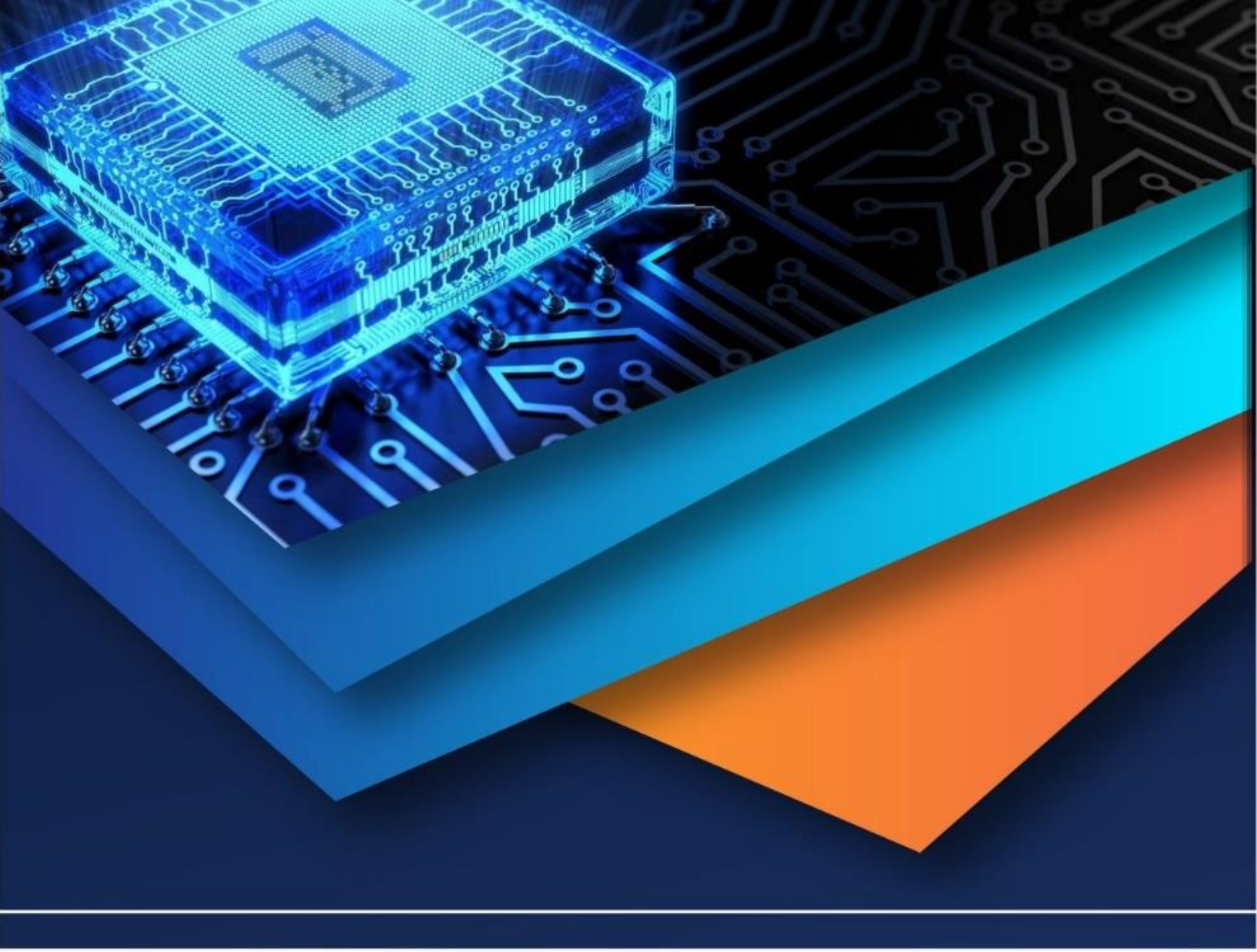
XIII. ACKNOWLEDGMENT

We hereby take the privileged to present our paper on Enhancing Session Security on Browser With Disposable Credentials Using OTC.We are very grateful to our paper guide Dean Dr.R.R.Sedamkar for contributing and valuable time in the paper from their busy and hectic schedule. Thank you for being after us like a true mentor and great academic parents.
We are very thankful to Dean Dr. R.R .Sedamkar whose guidance and support was an immense motivation for us to carry on with our paper. Also suggestions have greatly contributed for the betterment of our paper.

REFERENCES

[1] C. Aggarwal, Y. Zhao and P. Yu, "On the Use of Side Information for Mining Text Data," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 6, pp.1415-1429 2014
[2] S.Saranya , R.Munieswari, "A Survey on Improving the Clustering Performance in Text Mining for Efficient Information Retrieval", International Journal of Engineering Trends and Technology (IJETT), Volume-8 Number-5 , 2014.
[3] Rekha Baghel and Dr.Renu Dhir, "A Frequent Concepts Based Document Clustering Algorithm," International Journal of Computer Applications (IJCA), Volume 4 – No.5, July 2010.
[4] Rosemary Tripura and P.Selvaraj, "Efficient Text Mining Using Side Information of Documents," International Journal of Engineering Development and Research (IJEDR), Volume 3, Issue 1, 2015
[5] Noor Kamal Kaur, Usvir Kaur and Dr.Dheerendra Singh "K-Medoid Clustering Algorithm," International Journal of Computer Application and Technology (IJCAT), Volume 1, April 2014.
[6] Divya P and G.S.Nanda Kumar, "Effective Feature Selection for Mining Text Data with Side-Information," International Journal for Trends in Engineering and Technology, Volume 4, April 2015.
[7] Xingheng Wang, Jun Cao, Yao Liu and Shi Gao, "Text Clustering Based on the Improved TFIDF by the Iterative Algorithm," IEEE Symposium on Electrical & Electronics Engineering (EEESYM), 2012.
[8] Y.R.Gurav, "A Review on Side Information Entangling For Effective Clustering Of Text Documents in Data Mining," International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5 (6), 2014
[9] Firdous Sadaf M.Ismail, Prof. Amol G. Muley, "A Review on Clustering Techniques using Side-Information for Mining," International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), Volume: 3 Issue: 2, February 2015.
[10] Parul Agarwal, M. Afshar Alam and Ranjit Biswas, "Analysing the agglomerative hierarchical Clustering Algorithm for Categorical Attributes," International Journal of Innovation, Management and Technology, Vol. 1, No. 2, June 2010.

[11] Yogita Rani and Dr. Harish Rohil, "A Study of Hierarchical Clustering Algorithm," International Journal of Information and Computation Technology, Vol. 3, 2013, pp. 1225-1232.
[12] Aruna Bhat, "K-Medoids Clustering using partitioning around medoids for performing Face Recognition," International Journal of Soft Computing, Mathematics and Control (IJSCMC), Vol. 3, No. 3, August 2014.
[13] Raghuvira Pratap A , K Suvarna Vani , "An Efficient Density based Improved K- Medoids Clustering algorithm," International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 2, No. 6, 2011.
[14] Nikita P.Katariya1, M. S. Chaudhari, "Text Preprocessing For Text Mining Using Side Information," International Journal of Computer Science and Mobile Applications (IJCSMA), Vol.3, January- 2015, pp. 01-05.
[15] Raymond T.Ng, Jiawei Han, "CLARANS: A Method for Clustering Objects for Spatial Data Mining," IEEE Transactions on Knowledge and Data Engineering, Vol.15, October 2002.
[16] Tian Zhang, Raghu Ramakrishnan, Miron Livny, "BIRCH: A New Data Clustering Algorithm and Its Applications," IEEE Transactions on Data Mining and Knowledge Discovery, June 1997, Volume 1, pp 141–182.
[17] M. Shahriar Hossain, Praveen Kumar Reddy Ojili, Layne T. Watson, Naren Ramakrishnan, "Scatter/Gather Clustering: Flexibly Incorporating User Feedback to Steer Clustering Results," IEEE Transactions on Knowledge and Data Mining, 2012.
[18] Ho Chung Wu And Robert Wing Pong Luk "Interpreting TF-IDF term weights as making relevance decisions," ACM Transactions on Information Systems (TOIS), Vol.26, June 2008.
[19] K.Sruthi, B.Venkateshwar Reddy, "Document Clustering on Various Similarity Measures," International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), Vol. 3, August 2013.
[20] M.Deepa, P. Revathy, "Validation of Document Clustering based on Purity and Entropy measures," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol. 1, May 2012.
[21] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu, "On Using Partial Supervision for Text Categorization," IEEE Transactions On Knowledge And Data Engineering, Vol. 16, February 2004.
[22] Naveena.M, Karthik.R, Balaji.M3, "Side Information Gathering for Mining Text Data," International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Vol. 3, February 2015.
[23] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey, "Scatter/Gather: a cluster-based approach to browsing large document collections," in Proc. of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, 1992.
[24] Florian Beil , Martin Ester , Xiaowei Xu "Frequent term-based text clustering," in Proc. of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002, pp 436-442.
[25] R. Angelova and S. Siersdorfer, "A neighborhood-based approach for clustering oflinked document collections," in Proc. CIKM Conf., New York, NY, USA, 2006, pp.778–779.
[26] Wei Cheng , Xiang Zhang , Feng Pan , Wei Wang, "Hierarchical co-clustering based on entropy splitting," in Proc. of the 21st ACM international conference on Information and knowledge management, 2012 .
[27] Omar H. Karam, Ahmed M. Hamad, and Sherin M. Moussa, "Analysis of documents clustering using sampled agglomerative technique," in Proc. of the 12th International Conference on Computer Theory and Applications (ICCTA'2002), Alexandria, August. 2002, pages 193-197.
[28] R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in Proc. VLDB Conf., San Francisco, CA, USA, 1994, pp. 144–155.
[29] ABADI, M., AND BLANCHET, B. Analyzing Security Protocols with Secrecy Types and Logic Programs. In Proceedings of the ACM Symposium on Principles of Programming Languages (POPL)(2002).
[30] JACKSON, C., AND BARTH, A. Forcehttps: protecting high-security web sites from network attacks. In Proceedings of the ACM Conference on World Wide Web (WWW) (2008).
[31] BLANCHET, B., AND CHAUDHURI, A. Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage. In Proceedings of the IEEE Symposium on Security and Privacy (OAKLAND) (2008).
[32] METZ, C. Google turns on SSL encryption for search. http://www.theregister.co.uk/2010/05/21/google search ssl encryption/,2010
[33] MOSBERGER, D., AND JIN, T. httperf—a tool for measuring web server performance. ACM SIGMETRICS Performance Evaluation Review 26, 3 (Dec. 1998), 31–37.
[34] SYAMSUDDIN, I., DILLON, T., CHANG, E., AND HAN, S. A Survey of RFID Authentication Protocols Based on Hash-Chain Method. In Proceedings of the International Conference on Convergence and Hybrid Information Technology (2008).
[35] WANG, X. Finding collisions in the full SHA-1. In Proceedings of Crypto (2005), Springer.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY