**Cookies, Privacy and Cyber Security**

**Krishna Teja Ayinala Sainath Reddy Methukupally Sathvika Komalla**

**CSCI-8410 Term Project**

**Introduction**

**Cookie:**

Cookies are text files, which maintains the user information to handle and serve the web requests. They are maintained by the web browsers in the local systems of the user. Cookies help in maintaining session data, personalization of web pages and tracking the previous activity on particular website.

**History of Cookie:**

Netscape Communication Corporation has developed the concept of cookie around 1994. Until then websites do not have any information to keep track of it’s users. So, every user will be considered as a first time user. The concept of cookies helps to maintain the data and remember the user’s id, preferences, state information and directly display them.

**Components of a cookie:**

The structure of a cookie is usually a name, value pair. It also holds additional attributes to enforce security. The additional parameters are not passed along with the requests to the server unlike name value pair but remain with the browser and handle security.

* <”name”, “value”, “=”>
* Secure: Allows cookie transfers only on encrypted networks. (no-value)
* Domain: Restricts the accessibility to specified domains and its sub-domains.
* Path: It specifies the path for which cookies are valid.
* HTTPOnly: Limits the accessibility to client-side scripts. (no-value)
* Expires: sets the validity of the cookie by specifying date and time.
* Max-age: sets the validity of the cookie by specifying the seconds.
* SameSite: restricts access of cookies by cross-sites.

**Types of cookies:**

* **Session cookies**
  + These cookies are temporary and are expired after a session or closing the browser.
* **Persistent cookies**
  + These cookies are stored across multiple sessions on the hard-disk until a expiry date.
* **First Party Cookies**
  + These cookies are from same domains of the current browsing website.
* **Third Party cookies**
  + These cookies are from cross domains other than the current browsing domain which are mainly for advertisements referring domain.

**Problems and security issues**

Though cookies help to keep track of states and preferences, they do have some risk factors.

Cookies maintain personal data, submitted by the user in a web form and associated action events. Their data is Personally Identifiable Information (PII) like credit-card details, passport numbers, names, etc. This kind of personal data can be stolen, modified or misused. Yahoo mail attack([Yahoo Mail Attack](https://thehackernews.com/2017/03/yahoo-cookie-forging-hack.html)) is a recent attack in 2017 where about 32 Million accounts were hacked by cookie forging. The attackers were able to access the accounts without even logging in.

Cookies can also help to monitor and predict the user activities on the web. They can be used to perform the network analysis. Many domains populate advertisements from other websites when they are loaded. These advertisements use the cookie data, to know how many times the user has clicked on their advertisement and what kind of users are interested in their product.

Usually, to maintain security some users delete their cookies frequently which are locally stored on their computers by the browsers. There are a kind of cookies called Flash cookies, which are stored in different locations locally by the browser’s plugin Adobe flash player. These Flash cookies are also known as Local Shared Objects (LSO). They are not abided by the browser privacy settings making the user data vulnerable to tracking. These flash cookies are also used to recreate the deleted cookies related to the browsers using LSO. These recreated cookies are known as Zombie cookies.

There are other kind of cookies called Super cookies which are injected at the network level in Unique Identifier Headers (UIDH). They are stored at Internet Service Provider (ISP) servers. They track the browsing information of users on the network. Since they are not present on the local systems, they are harder to remove than Flash cookies which are stored at some location on the user system. [Verizon Issue](https://www.accessnow.org/verizon-fined-1-35-million-use-supercookies/)

All these threats are at network or application level which results in Cookie theft, Cookie Poisoning, Cross-site Scripting and Cross-site Request forgery.

**Cookie Theft**

Nowadays, the technology has advanced so far that many of the hackers are stealing the personal data and gain control over the private accounts through internet as a medium. This can be done by various methods, one of that is Cookie theft. Many reports have been filed on this issue. In Cookie theft, the hackers will be accessing the sensitive data from the user, when user uses the unsecured LAN/Wifi connections. A large portion of the outgoing traffic in public wireless networks remains unencrypted, thus, exposing a significant amount of users to cookie hijacking attacks. The Stolen HTTP cookies can allow the hacker to access the users browsing history information. This Cookie data is seen by the hacker over a network traffic on request to web server, when the data is sent over unsecured network.

For Example, consider a user, who login to his personal account over a unsecured wifi network and one bad click of the user may result in personal data theft. With the stolen data, the hacker will gain control over the account and perform various malicious operations like Data stealing in banking sectors, e-commerce sites.

**Methods to avoid Cookie Thefts:**

* **Frequent Deletion of Cookies:** Using the stolen cookie, the adversary can start issuing Google searches for various terms of interest (**Browser History**) and can perform malicious activities depending on the user browser history over a session period.
  + To overcome this type of attack, it is preferred to use the secure network connection for data transfer and if you are using someone’s computer then its better to clear the cookies in browsing history.
* **Using Secure VPN connection:** Also, there would be another type of data leak from the 3rd party user, if we use the extensions for opening a website on web browser such as using number of extensions for a site to load i.e. ([https://\*/\*/\*](about:blank)) and fetch the data from the server, then there will be chance of cookie attack as we use unsecured network.
  + To overcome this type of cases, we need to use the **HSTS** and **secure VPN** connection on the network.
* **Using SSL/TLS connection**: For secure connection and the communication between the web-browser and the server over a network, many websites use the SSL/TLS connection as it provides privacy and the data integrity.
* **Three tier verification Technique:** This technique is used to prevent the data leakage due to session sidejacking. This technique assumes that the server uses a secure HTTPS connection for login purposes to avoid transmission of password in the clear text. It uses a feature of Hyper Text Markup Language Version 5 (HTML5) called “local storage” to overcome the vulnerabilities of cookies and it foils any attempt to sidejack a session.
* **Use of encryption methodologies:** Even though there are certain security mechanisms for protecting cookies on network, there may arise some problems while cookies are stored in browser. So, using encryption methodologies on cookie values would increase safety.

**Cookie Poisoning**

Basically, Cookie poisoning is a process of attack by an unauthorized person to modify the content in the cookie, which will allow him to gain access to sensitive information stored in cookie or server of the web browser. Typically, a cookie may store information such as session id, user id, data, values, expiration, path and domain. If the attacker tries to change any of the above the above parameters, it would result in gaining the access to web browser. For example, we might consider attacking the online retailer’s cookie before it is sent to the server from web browser during the checkout process and changing the price values so that the server is tricked to charge user less money.

As cookie poisoning is an easy process, many web applications are designed in such a way that only a unique session identifier is stored in cookie and all the key parameters are stored in server. Even though the websites ensure the safety of the content within cookies by storing session identifier and the rest of the data in server, there may be cause of Session Fixation Attack. Basically, Session Fixation Attack is a system which allows to identify one person’s session identifier to other. These attacks would be avoided by regenerating the session identifier for each request. This process reduces the chance of data manipulation and unauthorized access.

There are many approaches implemented to avoid cookie poisoning:

* **Web Application Firewall:** A good Web Application Firewall secures the data from cookie poisoning as it checks the cookie ‘set’ commands send by the server with all the HTTP requests to verify the information stored in the received cookie. Cookie will be accepted only when there is no kind of third-party manipulation of data by checking that the information is accurate.
* **Symmetric or asymmetric encryption:** To protect cookies against cookie poisoning, websites also use symmetric or asymmetric encryption methodologies to encrypt the cookies stored in computer. In symmetric password technical security plan, when user sends request for cookie encryption then the server shares the same symmetric key. Server will use the same key to encrypt the cookie and will send the encrypted cookie to user where browser technology generates a time stamp linked to cookie encryption. Message authentication code is calculated at the same time by MAC technology. Browser will send the request, time stamp, message authentication code and encrypted cookie to the server. Server checks the data and will send the cookie to the browser if there is no problem. In asymmetric password technical security plan, at first, user will encrypt the data with the private key and will send to server, then server encrypts the cookie with a secret key where the secret key will be encrypted by the user public key. Encrypted secret key, encrypted cookie is sent to the user and then the user can decrypt the cookie. This approach has major advantages as the cookie will be encrypted where it reduces the data leakage but there may be a drawback of key attacks.
* **One-Time pad Cookie Encryption:** In order to limit key attacks,One-Time pad Cookie Encryption mechanism is used for protecting sensitive information. Each time a cookie is generated or the older one is updated, a new random key would be generated and used to encrypt the cookie values. Then the new encryption key is stored in the database instead of the old one. Searching for new encryption key each time when server receives cookie would be a drawback. But however, this mechanism is mostly used to secure highly sensitive information.

**Cross-Site Scripting (XSS)**

It is one of the major vulnerabilities in the computer security. In the XSS, the attacker accesses the personal information of the web user from the browser’s session cookies. The attacker injects some kind of script into the web application, which when accessed by the application user and gets vulnerable to the script. The script gets executed and access the session cookie information from the user’s web browser. As a result, the attacker can impersonate the user in different applications. The major XSS attacks using session cookies is Session Hijacking.

**Session Hijacking:** We pass our details to the web server after logging into the application. Using Set-Cookie the server obtains the information of us. For each communication request with the server, the cookie is used by the Javascript which if compromised, an attacker can impersonate us.

* **Same Origin Policy (SOS):** This specification restricts loading scripts from different domains. Only the scripts which match the protocol, port and host are loaded. The attacker can perform XSS from the same origin.
* **Cross Origin Resource Sharing (CORS):** In the current scenario, Cross Origin Resource Sharing is considered as a very common criteria in order to display advertisements related to their search criteria. In this scenario preflight checks are conducted by the browsers. The web applications are not involved in this precheck. Attackers can use the server to manipulate the browser preflight checks by XSS.
* **Content Security Policy (CSP):** CSP is a W3C standard which supports to identify the source of malicious scripts. It maintains a white-list of acceptable source of scripts. CSP is independent and is not a mandatory standard for web applications.
* **HTTP Strict Transport Security (HSTS):** It is a standard which restricts communication of data only under secure connection HTTPS.
* **HttpOnly:** It is an optional flag set during Set-Cookie initialization. When HttpOnly flag is enabled, the cookie is not accessible by the unauthorized scripts from the client. The only limitation is the browser has to support the HttpOnly flag.

**Cross-Site Request Forgery (XSRF)**

The CSRF is an attack on the web application which leads to an undesirable action. This can only happen when a user has been already authenticated. Any actions on the web page related to social engineering techniques like clicking on an email link, can lead to undesired action, in the form of a state change request to the server. The attacker exploits url when user clicks the email or image links on the web page. The CSRF attacks does not steal our data but can lead to serious actions.

**Session riding:** The related action on clicking the email links or the images leads to sending the user credentials, session cookie with required details. The server cannot differentiate if it’s the authorized request or not. The attacker sets a script to exploit the URL like transferring the money from the user account.

* **Token based Mitigation:** We have three kinds of token-based mitigation. Synchronized token pattern, Encryption based token pattern and HMAC based token pattern.
* **Synchronized Token Pattern:** Randomly generated tokens are used for each request on the website, which can identify the valid and invalid requests. The tokens should be generated by a secure cryptographic random number generator and inserted in HTTP request header. If the server fails to authenticate the generated token, the request is not served.
* **Encryption based Token Pattern:** The server generates an unique token and stores the token as a hidden field or parameter on the request header. The token is generated based on user id, timestamp and nonce. Each time the user makes a request, the key is decrypted at the server using the same key used for its creation. If the decryption fails, the request is not served.
* **HMAC based Token Pattern:** Hash based Message Authentication Code is almost similar to the encryption-based token pattern with an additional field operation. It uses hash-based encryption and indicates the purpose of the operation. Each request is validated using the token by regenerating the token with same parameters. If the token can be produced again, it serves the request else aborted.
* **Header Checks:** It verifies the source and target origins and checks the source header against the target header. If the source origin is not present, referrer header is verified against the target origin header.
* **One-time passcode generator:**  A confirmation passcode to authenticate the request each time you place along with the request details.

**Summary**

Cookies are files that store user information to handle web requests. Cookies contain personal data that can be stolen and misused. An attacker intercepts cookie and attempts to steal and modify the information so that he can gain access to the website. This may result in cookie theft, Cookie Poisoning, Cross-site Scripting and Cross-Site Request forgery. Though, cookies have many advantages but accepting a wrong cookie may elicit malicious attacks on user’s privacy. As a result, it is beneficial for users to be aware of the security issues caused and measures taken to avoid them.

* **One-Time Cookies (OTC) protocol:** Cookies are the shared secret between browser and web application, they are standard and can be easily disclosed. So additional protection mechanisms are required to ensure authentication while travelling on network and also while stored in browser. One-time Cookies creates a unique token per request. Each token is bound to a particular request by using a session secret, so, a token cannot be reused for different requests. Each token is encrypted with a long-term key which is shared between web application servers.
* **Use Of cryptographic algorithms in cookie values:** This proposal would apply cryptography on attribute values, which would generate hash of these values, with the intention of raising safety level. Cryptographic algorithms are proven efficient in terms of confidentiality, integrity and collision making interception, modification and duplication difficult.
* **Avoiding Replay Attacks by adding SSL into HMAC:** Replay attacks are nothing but stealing a cookie that issued by serve to client. Here, Attacker first collect large number of data being encrypted by session key and obtains session key, then, initiates SSL connection with server using this session key and replaces original. Server incorrectly authenticates as attacker spoofs client. This could be avoided by adding SSL session key into the keyed-hash message authentication code of a cookie, i.e., to use HMAC (user name | expiration name | data | session key) as the keyed-hash message authentication code of each cookie. Therefore, a cookie becomes session specific. Even if an attacker steals a cookie, he cannot successfully replay it since the session key is known only to a legitimate client and the server that creates the cookie.
* **Avoiding Volume Attacks using Encrypted HMAC:** These attacks are caused due to computing HMAC and steal the cookie. This is avoided by using encrypted HMAC.
* **Substitution of HTTP protocol by HTTPS:** Substitution of HTTP protocol by HTTPS possesses another safety layer known as TLS (Transport Layer Security) would greatly increase cookies safety.
* **Use of flag secure with cookies:** Secure flag of the cookies is to determine if the cookie should be transmitted by the secure channel
* **Using HTTP Only:** HTTP Only flag is to decide if client-side scripts can access the cookie, this helps in avoiding certain types of attacks such as XSS attacks

**Conclusion**

In this literature review, we elaborated the usage of cookies, problems caused by cookies and security methodologies implemented to avoid them. Based on different examples provided, we know that how easy it is to capture a cookie and gain access to website. In this paper, we have done a literature review on problems caused due to Cookie theft, Cookie Poisoning, Cross- Site Scripting (XSS) and Cross-site Request Forgery (XSRF). We found that the solutions to these attacks are using cryptographic algorithms and also using the secure network connection while transmitting cookies. Usage of flags such as HttpOnly and Secure would mainly increase safety on cookies while travelling on network and stored cookies in local systems. These methods mainly reduce the attacks on cookies, but as cookies has wide uses and maintains personal data, there may be a cause of more security concerns. So, there is a need for more new methodologies to mitigate risks and limit the effectiveness of an attacker.

**References**

* B. Li, S. Lv, Y. Zhang and M. Tian, "The application research of Cookies in network security," *PROCEEDINGS OF 2013 International Conference on Sensor Network Security Technology and Privacy Communication System*, Nangang, 2013, pp. 152-155.

doi: 10.1109/SNS-PCS.2013.6553855

* G. Pujolle, A. Serhrouchni and I. Ayadi, "Secure session management with cookies," *2009 7th International Conference on Information, Communications and Signal Processing (ICICS)*, Macau, 2009, pp. 1-6.

doi: 10.1109/ICICS.2009.5397550

* I. Yusof and A. K. Pathan, "Mitigating Cross-Site Scripting Attacks with a Content Security Policy," in Computer, vol. 49, no. 3, pp. 56-63, Mar. 2016.

doi: 10.1109/MC.2016.76

* K. Nirmal, B. Janet and R. Kumar, "It's More Than Stealing Cookies - Exploitability of XSS," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 490-493.

doi: 10.1109/ICCONS.2018.8663230

* S. Sivakorn, I. Polakis and A. D. Keromytis, "The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information," *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, 2016, pp. 724-742.

doi: 10.1109/SP.2016.49

* A. M. Osman, A. Dafa-Allah and A. A. M. Elhag, "Proposed security model for web based applications and services," 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, 2017, pp. 1-6.

doi: 10.1109/ICCCCEE.2017.7866696

* J. S. Park and R. Sandhu, "Secure cookies on the Web," in IEEE Internet Computing, vol. 4, no. 4, pp. 36-44, July-Aug. 2000.

doi: 10.1109/4236.865085

* Allaboutcookies.org. (2019). All About Computer Cookies - Session Cookies, Persistent Cookies,How to Enable/Disable/Manage Cookies. [online] Available at: http://www.allaboutcookies.org/ [Accessed 11 Apr. 2019].
* Yogesh Mundada, Nick Feamster, and Balachander Krishnamurthy. 2016. Half-Baked Cookies: Hardening Cookie-Based Authentication for the Modern Web. In Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security (ASIA CCS '16). ACM, New York, NY, USA, 675-685. DOI: <https://doi-org.leo.lib.unomaha.edu/10.1145/2897845.2897889>
* Italo Dacosta, Saurabh Chakradeo, Mustaque Ahamad, and Patrick Traynor. 2012. One-time cookies: Preventing session hijacking attacks with stateless authentication tokens. ACM Trans. Internet Technol. 12, 1, Article 1 (July 2012), 24 pages. DOI=<http://dx.doi.org.leo.lib.unomaha.edu/10.1145/2220352.2220353>
* Anderson A. L. Queiroz and Ruy J. G. B. de Queiroz. 2010. Breach of internet privacy through the use of cookies. In Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '10), Fillia Makedon, Ilias Maglogiannis, and Sarantos Kapidakis (Eds.). ACM, New York, NY, USA, , Article 71 , 5 pages. DOI=<http://dx.doi.org.leo.lib.unomaha.edu/10.1145/1839294.1839378>
* Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. 2016. An Empirical Study of Web Cookies. In Proceedings of the 25th International Conference on World Wide Web (WWW '16). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 891-901. DOI: <https://doi-org.leo.lib.unomaha.edu/10.1145/2872427.2882991>
* Flash cookies (LSOs) – Security risks and privacy issues. April 12, 2019. <<https://www.pcbuyerbeware.co.uk/blog/flash-cookies-lsos-security-risks-privacy-issues/>>.
* Cross-Site\_Request\_Forgery\_Prevention\_Cheat\_Sheet. April 12, 2019. <https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site\_Request\_Forgery\_Prevention\_Cheat\_Sheet.md>.
* B. Li, S. Lv, Y. Zhang and M. Tian, "The application research of Cookies in network security," *PROCEEDINGS OF 2013 International Conference on Sensor Network Security Technology and Privacy Communication System*, Nangang, 2013, pp. 152-155.

doi: 10.1109/SNS-PCS.2013.6553855

* K. LaCroix, Y. L. Loo and Y. B. Choi, "Cookies and Sessions: A Study of What They Are, How They Work and How They Can Be Stolen," *2017 International Conference on Software Security and Assurance (ICSSA)*, Altoona, PA, 2017, pp. 20-24.

doi: 10.1109/ICSSA.2017.9

* P. Yadav and C. D. Parekh, "A report on CSRF security challenges & prevention techniques," *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, 2017, pp. 1-4.

doi: 10.1109/ICIIECS.2017.8275852