Taking a Look into the Cookie Jar: A Comprehensive Study towards the Security of Web Cookies

Sean Chen*
Texas A&M University
College Station, Texas, USA
seanchen25@tamu.edu

Jaelyn McCracken*
Towson University
Towson, Maryland, USA
jaemcc21@gmail.com

Kevin Lu
St. Mark's School of Texas
Dallas, Texas, USA
lukbrinker@gmail.com

Tao Wang
UNC Charlotte
Charlotte, North Carolina, USA
twang27@charlotte.edu

Tao Hou
Texas State University
San Marcos, Texas, USA
taohou@txstate.edu

ABSTRACT

Cookies play a vital role in our Internet browsing experience, enabling various functions on websites. However, their significance also extends to potential vulnerabilities, especially in the face of cyber-attacks like cross-site scripting. As we interact with websites daily, it remains uncertain how well these platforms protect user data from such attacks or how efficiently they address vulnerabilities. To address these concerns, our research endeavors to conduct a comprehensive measurement study across the entire Internet landscape. Our goal is to shed light on the potential security risks and the extent to which protective measures are deployed on websites. We developed a customized toolkit to scrape a multitude of websites on the Internet, this objective can be assessed by analyzing the security flags of cookies. Through this study on web cookies, we obtained a better understanding of the current state of web security and identified potential areas of improvement.

CCS CONCEPTS

• **Security and privacy** → **Web application security**.

KEYWORDS

Web Cookies, Web Security, User Privacy, Online Browsing, Cookie Hijacking, Session Spoofing

ACM Reference Format:
Sean Chen, Jaelyn McCracken, Kevin Lu, Tao Wang, and Tao Hou. 2023. Taking a Look into the Cookie Jar: A Comprehensive Study towards the Security of Web Cookies. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3565287.3617625

*Co-First Authors

1 INTRODUCTION

Cookies play a crucial role in authenticating users and enabling various functionalities on websites. Specifically, a cookie is a small, unique piece of data that contains an alphanumeric string [2]. As users browse the web, websites store these cookies on their computers. Cookies serve to identify and authenticate users on the Internet, making it a crucial part of the browsing experience. When a user visits a website for the first time, the server sends a cookie to the user's web browser. This cookie is sent through the "Set-Cookie" header and is then stored on the user's computer's hard disk. The website uses this identification to track the user's session from start to finish as well as any future sessions assuming the cookie remains stored and unexpired.

Cookies typically serve three primary purposes: session management, personalization, and tracking. Session management involves retaining important data such as logins, shopping carts, game scores, or any other information that needs to be remembered by the server during the user's session or the next time they visit the web page. Personalization is another important feature, where websites remember user preferences and settings, such as themes, to tailor the browsing experience according to individual choices. Tracking involves analyzing and recording the user's behavior as they browse [19]. While tracking can benefit websites as they can gain insights on user's browsing history and preference, it also raises privacy concerns. Some websites use tracking to gather browsing activity data and deliver targeted information, such as advertisements for goods or services.

Nevertheless, cookies also present a vulnerability to cyber attacks, particularly cross-site scripting (XSS) attacks, which poses a serious risk to user privacy [7]. XSS attacks are general web attacks where malicious JavaScript code is sent via an input form by an attacker. It is then mistaken for legitimate code and is executed by the server. A well known type of XSS attack is Cross Site Request Forgery (CSRF) where the attacker can initiate unwanted requests on the user's client side to the server [21]. It can perform malicious actions such as transferring funds to their account on behalf of the user who is now authenticated and is a victim from clicking on the vulnerable website. This attack is highly prevalent due to browsers automatically attaching any type of cookie to every request by default. This makes it relatively easy for attackers to exploit vulnerabilities and compromise user data.

There are many other ways of attacking cookies that attackers perform to exploit users and their information. Man-in-the-Middle is another common and popular attack. This is when requests are intercepted by an attacker in between the user agent and the server and may possibly be altered. Another form of attack is Cookie Poisoning, where an attacker steals and alters a user's cookies. Within this category falls Session Hijacking, where an attacker gains control of a user's browsing session by stealing or predicting a valid cookie, thereby obtaining unauthorized access to the server. The attacker waits for the user to log in, steals their cookie, and subsequently hijacks their session, enabling them to make arbitrary requests on behalf of the user [4]. One variant of Session Hijacking is Session Fixation, which exploits cookies accepted from URLs or post request data. Here, the attacker sets up a fake session before the actual user logs in. Once they obtain the session cookie, they gain access to the user's account [16]. Moreover, Session Spoofing is another method used in attacks. The objective is similar to Session Hijacking, but instead of taking over the user's session and forcing them offline, the attacker impersonates another user to gain access to their information [17]. Along with these come with cookie injection, cookie theft, and cookie tampering. Cookie poisoning encompasses a variety of techniques that attackers can employ to manipulate or misuse cookies.

To address these concerns and enhance overall web security, security header flags such as SameSite, HttpOnly, and Secure were created. These measures of protection via HTTP headers aim to deter attackers from exploiting cookies and safeguard user privacy. In this research, we focus on gaining valuable insights into the overall usage of these security flags on the Internet. To do so, we developed a customized toolkit to scrape the top, middle, and bottom 1,000 websites out of the 1,000,000 most visited websites for their cookie settings. With this tool, we were able to conduct a thorough and comprehensive analysis of how these security flags are utilized throughout the Internet. By analyzing these cookies, we obtained a better understanding of the current state of web security and identified potential areas of improvement. Our aim is to contribute to the overall enhancement of online security and protect users' data and privacy from potential threats.

The remainder of this paper is structured as follows: In Section 2, we introduce the preliminaries related to cookie security. Section 3 presents the methodology for conducting an Internet-wide measurement study to understand cookie security settings. In Sections 4 and 5, we present the results of the measurement study and engage in a discussion to analyze the measurement observations. We outline our plans for future work in Section 6, and finally, we conclude this paper in Section 7.

2 PRELIMINARIES

In this section, we briefly introduce the preliminaries related to cookie security.

2.1 3rd Party Cookies

Third-party cookies operate similarly to first-party cookies, but they possess distinct characteristics that set them apart. Unlike first-party cookies, which are set by the server hosting the webpage, third-party cookies are established by a server associated with a different domain. This means that any website with access to the code of the third-party server can access these cookies as well. As a result, third-party cookies allow entities to continue serving targeted information even after the user's session ends, as the session data that remains on the user's computer.

Trackers use third-party cookies as a technique to remember users across multiple domains without their consent. This is achieved through Javascript that appears on numerous websites to display advertisements. Whenever the web page containing the Javascript is visited, it triggers a request to the tracker's server. If the request contains a cookie, the tracker associates the triggered request with the user's profile. Subsequently, the user's browser links the cookie with the tracker's website and includes it in all future requests to the same server. This grants the tracker the capability to follow users across every website utilizing the script that initiates requests to their server, enabling extensive tracking across the web [6, 10].

The violation of privacy using third-party cookies has become a significant concern, prompting government intervention through regulations like the General Data Protection Regulation (GDPR) in the European Union. However, despite the introduction of these privacy regulations, several studies indicate that users are not fully utilizing the consent management options provided by GDPR [15]. To combat these tracking methods employed by trackers through cookies, it is imperative that web developers and experts implement effective strategies that prioritize user privacy and security of cookies.

2.2 HTTP Headers

As mentioned earlier, when a user visits a website for the first time, the server sends a cookie to the user's web browser using the Hypertext Transfer Protocol (HTTP) header [2]. This protocol facilitates communication between the browser and the server, enabling data exchange through requests and responses. Specifically, the server can include a 'Set-Cookie' header in its response to the user's request, which contains information about the cookie in the form of attributes. Taking a look into this header gives information about the cookie through instances called attributes. These attributes provide essential details about the cookie, such as its name, value, domain, and expiration time [8]. While these attributes are crucial for managing cookies, they may not reveal much about the cookie's security level. Another attribute called 'Max-Age' displays the number of seconds until the cookie expires, and this can be useful to the security of the cookie. The larger the number of seconds, the older the cookie is, and the longer it exists. This can become a vulnerability by giving the attacker more time to access the content of the cookie. However, to address security concerns related to cookies, additional attributes like SameSite, Secure, and HttpOnly are implemented.

2.3 Security Flags

The SameSite attribute is a major security flag that will be explored because it controls whether the cookie should be sent in cross-site requests, helping prevent certain attacks depicted earlier.This flag is determined by the website that sets the cookie, and plays a crucial role in determining how first and third-party cookies are handled. Typically, the SameSite attribute can take one of three values: strict, lax, or none. When the SameSite attribute is set to strict, it ensures that the cookie is never included in any cross-site requests. This means that only 1st party cookies can be sent and accessed, but not when the incoming link is from an external site. When the value is set to lax, the cookie can only be included in cross-site GET requests that are top-level [6]. This means that only 1st party cookies are enabled to be accessed and sent, while still maintaining a level of security from external sites. Lastly, when the value is set to none, it allows the cookie data to be shared with third parties and external sites. By properly configuring the SameSite attribute, website owners can enhance the security of their cookies and mitigate the risks associated with cross-site requests and potential attacks.

The HttpOnly flag serves as a critical security measure by restricting the accessibility of a cookie from client-side scripts like JavaScript. By doing so, it effectively prevents attackers from gaining access to sensitive data through Cross-Site Scripting (XSS) attacks that specifically target cookies. Furthermore, the Secure flag plays a vital role in enhancing the security of cookies during transmission. When this flag is set, the cookie is only sent to the server when a secure HTTPS request is made. The transmission of the cookie occurs through the secure Security Sockets Layer (SSL) channel. While the cookie contents themselves are not encrypted, the use of the SSL channel ensures that the data is transmitted securely, making it more resistant to potential Man-in-the-Middle attacks [3, 5, 12, 22]. It is important to note that cookies may not have all these security flags enabled, as some cookies might have multiple security measures in place while lacking others. While these security flags are powerful defenses, it is imperative to emphasize that they do not provide absolute protection against all attacks. Instead, they serve as essential security layers that make cookies more resilient to threats.

3 METHODOLOGY

Now that we know the purpose and effectiveness of the security flags in cookies, we want to see how many websites actually implement these attributes. For this, we'll need to take a look at the HTTP headers of a number of websites for the "Set-Cookie". All websites on the Internet deal with HTTP headers which allow for the server and client to pass additional information along with HTTP requests. For this purpose, we'll develop a web scraper that will send a GET request to a number of websites with the most traffic from a .txt file.

3.1 Web Crawling and Scraping

A web crawler, also known as a web spider, is a program that indexes web pages as it visits as many websites as possible. When given a starter URL, the crawler will queue all links on that webpage and visit each one, queuing each link on those webpages and so on. All webpages visited by the web crawler are indexed and organized so that those same web pages can be revisited. Search engines such as Google make use of web crawlers to rank web pages so that they appear in search results. Web scraping refers to the extraction of data of a web page that is scraped. Unlike crawlers which are mainly concerned with collecting webpages and their URL links, scrapers extract specific data off the webpage and store it for later usage or analysis.

3.2 Python Libraries

To automate the process of acquiring websites' Set-Cookie header settings, we need to send a GET request to access the HTTP headers of the website. Well-designed websites are able to detect bots such as web crawlers and scrapers so we'll need to send a legitimate User-Agent with the request to appear as an authentic user. To develop the scraper, we took a look at a variety of python libraries to use for sending the GET requests. For this project, we settled on using Requests.

Requests is an HTTP library for python. It allows for sending custom headers into requests to appear as a real user and not a python script. Through running a simple python script with requests or by inspecting the HTML code of any website, the User-Agent header of the current machine can be viewed and copied.

3.3 The Scraper

The completed scraper is a python script which utilizes the requests library. The scraper reads from a .txt file containing the domain names of selected websites to scrape. The websites are picked from a larger folder containing .txt files listing out the top 1,000,000 most visited websites as ranked by Alexa before it was shut down.

The script goes through the .txt file and initiates a GET request to a URL for each domain name in the file to retrieve the data from the "Set-Cookie" header. A new text file is then created for each successful URL connection with the website name being printed to the file followed by the attributes of the "Set-Cookie" header line-by-line. The text files are created starting at index 0 and end at 999 for a total of 1,000 valid website visits. The separate .txt files make it easier for counting and separating data. The process is repeated so that we have data for the top, middle and bottom 1,000 websites out of the 1,000,000.

An additional cautionary detail to keep in mind while developing the scraper is the possibility of being detected as a Distributed Denial-of-Service (DDoS) attack [11, 13, 18]. A denial-of-service attack works by flooding a server with a high number of superfluous requests, draining the server of resources to serve authentic, human users. Because an automated script can send a large number of requests in a small amount of time, a sleep function from the time library will be used to place a delay of 2 seconds in between requests.

3.4 Parsing and Merging Files

Once we have our data, we can begin analyzing the trends in the Set-Cookie header settings. After the scraping process is complete, the resulting data consists of three folders for each of the categories of websites (top, middle, and bottom) with each one containing 1,000 .txt files.

A Python script was then used to merge all 1,000 .txt files into a single file for each website category folder. From there, additional scripts were run to parse the file to look for a selected attribute and print the domain names of the websites utilizing that attribute to an additional text file. From there, the number of websites can then be recorded. Since websites can contain multiple cookies with different attributes, the script was written so that a website would be recorded in the output .txt file if it managed to contain at least one cookie with at least one instance of the targeted attribute. This was done for the following attributes: Secure, HttpOnly, SameSite, SameSite=None, SameSite=Lax, SameSite=Strict. The script was modified later to parse the txt. files for all Set-Cookie instances so that we could record the total number and average usage of attributes among the cookies.

4 MEASUREMENT RESULTS

Out of the websites, the top 1000 implemented more security flags in their cookies than the middle and the bottom. In addition, it is possible for some websites to have more than one cookie, and sometimes their Samesite settings differ as well. Although there was a lower usage of SameSite amongst the top websites, there was a greater number of cookies with SameSite flags per website. Some examples of websites that had their cookies set to strict were Reddit, BleacherReport, and Change.org. Examples of lax include Google and UsaToday. The none setting had numerous sites such as LinkedIn, YouTube, and ScienceDirect. Sites such as Netflix and Pornhub didn't even have a Samesite flag set. Regarding the 'Secure' flag, the top 1000 websites also had the most instances with 627 total cookies. However, when it came to the 'HttpOnly' flag, the middle 1000 had the most occurrences accounting for 582 cookies.

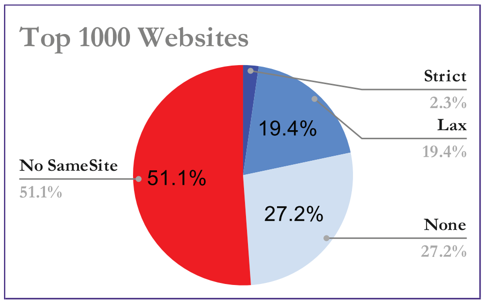

Figure 1: Samesite instances within the top 1000 websites.

In Figure 1, we see the results for the SameSite attributes observed among the top 1,000 websites. As predicted, the top sites show the most usage in SameSite attributes out of all three categories. However, the SameSite attribute only has a usage rate of a little less than 50% and out of the websites that do mention the SameSite attribute, more than half of them have it set to "None".

In Figure 2, we see the results for the SameSite attributes observed among the middle 1,000 websites. Here, we see some noticeable changes from the top 1,000 websites. There are significantly less websites using the SameSite attribute. The percentage of website with SameSite equal to Strict and Lax are approximately the same. However, because more websites do not have the SameSite attribute, that means they do not bear the Secure attribute, either.

In Figure 3, we see the results for the SameSite attributes observed among the bottom 1,000 websites. The percentage of websites without the SameSite attribute increases only marginally and the distribution of SameSite settings among the websites with the attribute remain largely unchanged. If we were to scrape below the top 1,000,000 websites, we can assume that the presence of websites

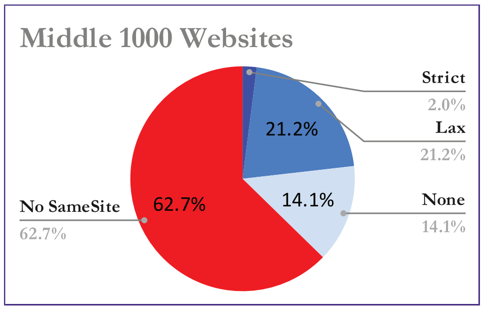

Figure 2: Samesite instances within the middle 1000 websites.

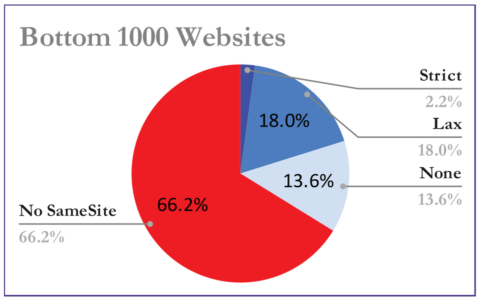

Figure 3: Samesite instances within the bottom 1000 websites.

without the SameSite attribute will continue to increase. The reason for there being a smaller difference in websites without the SameSite attribute between the bottom and middle websites and the middle and top websites could be due to the nature of the Internet to have exponential growth in traffic as we approach the top visited sites.

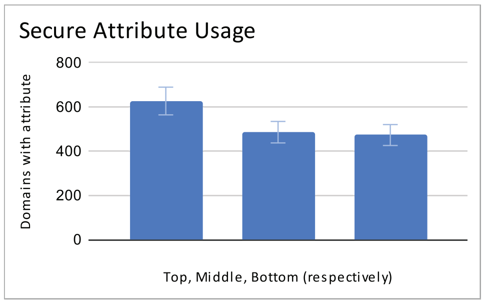

Figure 4: The number of cookies with a 'Secure' flag from top 100, middle 100, to bottom 1000 respectively.

In Figure 4, we see the distribution of all the Secure attribute instances across the three website categories. As predicted, the graph is skewed right with the top websites showing the most usage at a little over 600 attributes. This follows the trend of the amount of Internet traffic being skewed as we approach the most visited sites. The middle and bottom websites show significant decrease with a little over 450 instances of the SameSite attribute each.

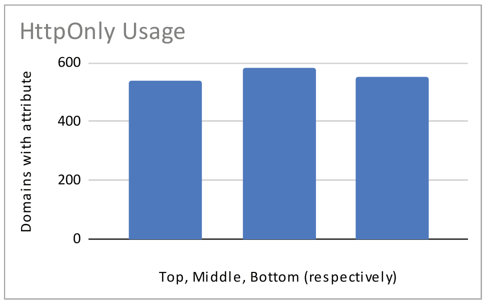

Figure 5: The number of cookies with a 'HttpOnly" flag from top 1000, middle 1000, to bottom 1000 respectively.

In Figure 5, we see the distribution of all the HttpOnly attribute instances across the three website categories. There was not a clear distinct trend in the graph. The graph shows an approximately normal distribution with the middle websites showing the most usage at a little under 600 instances of the attribute. The top and bottom trail behind at a bit over 500. However, the top shows the least amount of attributes which was surprising.

Table 1: Summary of all results found in number of cookies.

All Headers	Top 1000	Middle 1000	Bottom 1000
HttpOnly	540	582	553
Secure	627	486	473
No SameSite	535	649	676
SameSite	465	351	324
None	284	146	139
Lax	203	219	184
Strict	24	21	22

In Table 1, the number of instances for each of the attributes HttpOnly, Secure, and SameSite with all its settings are shown for each of the three categories of websites. Here, we can see that the top sites may have the lowest number of cookies with the HttpOnly attribute, it is made up for with the highest number of Secure attributes. This follows as the top sites also have the most number of instances of the SameSite attribute set to "None". For all attributes except for SameSite set to Lax or Strict, the middle and bottom are extremely similar. This could be explained in that past the top most visited websites, all other websites are more or less the same in terms of security. We should also keep in mind that given the surface web hosts over 1 billion websites, the top 1 million are still relatively popular.

5 DISCUSSION

The findings revealed a noteworthy trend among the top 1000 websites, indicating a substantial increase in the usage of SameSite attributes, which enhance security compared to the middle and bottom 1000s. In addition, no significant difference was observed between the middle and bottom 1000 websites in their use of SameSite attributes, which could be the result of their levels of popularity and traffic. Because the top had a significant increase in the usage of SameSite attributes compared to the middle and bottom, this could due to the fact that the top websites have more reputable advertisers. In contrast, middle and bottom sites may need to rely more on advertising from third parties and external sites to increase traffic to generate revenue. Allowing more third-party presence on their sites despite the risks is a gamble that least popular websites may be willing to take since they do not have as many strong sponsorship and reputable advertisers as the top do. The middle and bottom websites did not differ dramatically in their cookie attributes most likely due to not being as popular as the top websites, showing similarities in security.

The discovery that more than half of the websites in the top 1000 lacked the 'SameSite' security flag raises an important observation. Even though this is evident, we also extrapolated that they also had the most 'Secure' security flags amongst the top and bottom. It is a possibility that many of these websites use 'Secure' to make up for not having the 'SameSite' flag. To enhance security and prevent potential exploits, it is highly advisable for every website on the Internet to implement additional security flags on their cookies. Specifically, the top 1000 websites should prioritize setting more of their cookies as 'SameSite', and for those already using it, consider setting them to 'Strict'. By implementing 'SameSite' attributes on cookies, websites can reduce the risk of CSRF attacks and other vulnerabilities related to cross-site interactions.

In addition to the default Set-Cookie attributes, the scraper managed to record a variety of other cookies with unique string names. For example, GPS is a cookie used by YouTube to store geographic information pertaining to embedded videos on websites. This is most likely used as a tool to aid in YouTube's algorithms where video recommendations may consider users' location as a factor in likelihood of recommendation. Geographic data along with the users' may show fruitful trends in the area's interests at that current point in time. "1P_JAR" is a cookie that relates to Google services. It is used for tracking user's browsing history, preferences, and other activities to better serve personalized advertisements based on the users' Internet profile. Most websites will create their own cookies to better track and profile users so that they may utilize that information to better design and improve their website and services. The exact purpose and functionality of these cookies are left to their developers most likely out of proprietary discretion.

Another noteworthy detail about the Set-Cookie header is that although the Max-Age and Expiration are not directly relevant to cookie security, they are still related and can serve a great purpose to maintaining cookie confidentiality. These attributes simply entail how long the cookie lasts before it expires or becomes invalid [2]. The longer a cookie lives, the more time an attacker has to guess or obtain the cookie via brute forcing attacks. Shorter-lived cookies provide better security in that they are more temporary but may also be an inconvenience as the user's browsing session for website may not be preserved as long. Web developers are tasked with finding the balance between setting the expiration time of cookies so that their average users are satisfied and reducing vulnerabilities in long-lasting cookies.

6 FUTURE WORK

While cookies are essential for various website functionalities, they also raise concerns about user privacy. The investigation into cookie security is an ongoing endeavor, presenting numerous avenues for further exploration. Given the ever-evolving nature of the Internet, it becomes imperative to continually gather more information to ensure the research remains current and relevant.

One important next step is categorizing the websites used in the study. This categorization would enable a deeper analysis of cookies from different sources, leading to insightful conclusions about how various types of websites employ specific cookie attributes. For instance, we might observe differences in security attributes between news outlets, blogs, and streaming platforms across the Internet. Understanding these patterns would provide valuable insights into the reasoning behind setting certain cookie attributes for different types of websites. Furthermore, expanding the sample size by including more websites, particularly those with lower traffic, is essential. With over one billion websites on the Internet, our current data represents just a fraction of the available information. By scraping a more extensive range of websites, we can obtain a better understanding on the continuous trend of the security settings of web cookies, and conduct meaningful comparisons with our existing results in this current work.

Another direction we are keen to explore in this research involves identifying vulnerabilities and potential exploits. This approach would entail evaluating the effectiveness of browsers in safeguarding against cookie-based attacks. To carry out this investigation, we intend to harness the capabilities of Selenium [20] for browser automation, enabling comprehensive testing across multiple browser platforms. Additionally, we plan to employ machine learning techniques [1, 9, 14] to scrutinize the target browsers in-depth. For a thorough analysis of the data transmitted, a proxy will be essential to intercept requests and potentially uncover underlying threats. These findings will provide insights into the security levels of different browsers and highlight areas that require enhancement.

7 CONCLUSION

The significance of cookies becomes evident while browsing the Internet, as they are imperative for handling user data. When security concerns related to cookies arise, they could be addressed using security flags in the cookie attributes. To assess how well websites deploy these security flags, we conducted an Internet wide measurement study in this research. It is important to recognize that while the cookie attribute is a valuable security measure, it is not the sole guarantee of comprehensive protection. In addition, users also play a pivotal role in safeguarding their own data. When encountering websites that prompt messages about third-party cookies, we suggest the users to peruse and ensure only essential cookies are accepted. Security should be an ongoing, diligent effort, with websites continuously updating their security measures to adapt to emerging threats. To promote a safer browsing experience for users, all websites, regardless of their popularity, should prioritize security by adopting best practices. These proactive measures will fortify their defenses against potential vulnerabilities and bolster user trust in their platforms.

ACKNOWLEDGEMENTS

This research was supported in part by NSF REU site grant CNS-2149950.

REFERENCES

[1] Hamidah Alanazi, Shengping Bi, Tao Wang, and Tao Hou. Exquisite feature selection for machine learning powered probing attack detection. In *IEEE International Conference on Communications (IEEE ICC'23)*. IEEE, 2023.

[2] A. Barth. Http state management mechanism. RFC 6265, RFC Editor, 4 2011.

[3] Shengping Bi, Tao Hou, Tao Wang, Yao Liu, Zhuo Lu, and Qingqi Pei. Dywcp: Dynamic and lightweight data-channel coupling towards confidentiality in iot security. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (ACM WiSec'22)*, pages 222–232, 2022.

[4] Dan BonehJo. Web security: Session management.

[5] Franco Callegati, Walter Cerroni, and Marco Ramilli. Man-in-the-middle attack to the https protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.

[6] Gertjan Franken, Tom Van Goethem, and Wouter Joosen. Who left open the cookie jar? a comprehensive evaluation of Third-Party cookie policies. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 151–168, Baltimore, MD, August 2018. USENIX Association.

[7] Shashank Gupta and Brij Bhooshan Gupta. Cross-site scripting (xss) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of System Assurance Engineering and Management*, 8:512–530, 2017.

[8] Hamish Willee. Set-cookie, 2023. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie",addendum, Last accessed on 2023-08-10.

[9] Tao Hou, Shengping Bi, Tao Wang, Zhuo Lu, Yao Liu, Satyajayant Misra, and Yalin Sagduyu. Muster: Subverting user selection in mu-mimo networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 140–149. IEEE, 2022.

[10] Tao Hou, Shengping Bi, Mingkui Wei, Tao Wang, Zhuo Lu, and Yao Liu. When third-party javascript meets cache: Explosively amplifying security risks on the internet. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 290–298. IEEE, 2022.

[11] Tao Hou, Zhe Qu, Tao Wang, Zhuo Lu, and Yao Liu. Proto: Proactive topology obfuscation against adversarial network topology inference. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1598–1607. IEEE, 2020.

[12] Tao Hou, Tao Wang, Zhuo Lu, and Yao Liu. Smart spying via deep learning: inferring your activities from encrypted wireless traffic. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

[13] Tao Hou, Tao Wang, Zhuo Lu, and Yao Liu. Combating adversarial network topology inference by proactive topology obfuscation. *IEEE/ACM Transactions on Networking*, 29(6):2779–2792, 2021.

[14] Tao Hou, Tao Wang, Zhuo Lu, Yao Liu, and Yalin Sagduyu. Iotgan: Gan powered camouflage against machine learning based iot device identification. In *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 280–287. IEEE, 2021.

[15] Xuehui Hu, Nishanth Sastry, and Mainack Mondal. Cccc: Corralling cookies into categories with cookiemonster. In *Proceedings of the 13th ACM Web Science Conference 2021*, WebSci '21, page 234–242, New York, NY, USA, 2021. Association for Computing Machinery.

[16] Martin Johns, Bastian Braun, Michael Schrank, and Joachim Posegga. Reliable protection against session fixation attacks. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1531–1537, 2011.

[17] Kenneth LaCroix, Yin L Loo, and Young B Choi. Cookies and sessions: a study of what they are, how they work and how they can be stolen. In *2017 International Conference on Software Security and Assurance (ICSSA)*, pages 20–24. IEEE, 2017.

[18] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[19] Mozilla Developer Network. Using http cookies. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

[20] Selenium. Selenium automates browsers.primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. https://www.selenium.dev.

[21] Ethan Shernan, Henry Carter, Dave Tian, Patrick Traynor, and Kevin Butler. More guidelines than rules: Csrf vulnerabilities from noncompliant oauth 2.0 implementations. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings 12*, pages 239–260. Springer, 2015.

[22] Tao Wang, Yao Liu, Tao Hou, Qingqi Pei, and Song Fang. Signal entanglement based pinpoint waveforming for location-restricted service access control. *IEEE Transactions on Dependable and Secure Computing*, 15(5):853–867, 2016.