An Empirical Study of Cookie Theft Vulnerability for Modern Websites

Zhifeng Xiao, Jeffrey Knapp, Robert E. Steiner III
Department of Computer Science and Software Engineering
School of Engineering
Penn State Erie, The Behrend College
Erie, PA, 16563
zux2@psu.edu, {knapp.jeffrey91, robertsteiner10}@gmail.com

Abstract

This paper explores the cookie theft vulnerability for 50 websites, including 46 Alexa 500 sites and 4 online banking sites. We demonstrate that 72% of the tested sites are vulnerable to the cookie theft attack, including some most notable ones such as Google, Facebook, EBay and Amazon. In an attempt to evaluate the cookie security of a website, a cookie from a test user will be stolen from one computer, transferred to another computer, injected into a webpage, and eventually used to take control of the victim's account. We had successful attacks on 36 tested websites. This means the security community has not well responded to the event of cookie theft, although it might just be an easy fix.

1 Introduction

Web cookies are small pieces of information being passed back and forth between web browsers and servers to either maintain a web session, waive re-authentication, or remember user preference. In other words, it has been an effective and widely adopted approach to keep web applications stateful. However, cookie security has been a long-standing concern for over a decade. The notorious problem lies in the fact that a stolen session cookie could be abused to hijack a user's account, as long as the cookie is not expired, and that the web application is lack of server side protection. This handy trick has inspired some serious hacking techniques, and cookies have been one of the main targets. For example, hackers can use cross-site scripting (XSS) to collect session cookies and send them to a remote server for further exploitation. Surprisingly, although XSS vulnerability has been patched by most modern web applications, the cookie stealing problem has not been well responded by the security community. The paper is focused on the abuse of stolen cookies. We conduct an experiment on a list of popular websites from Alexa.com to assess the cookie vulnerability for each tested website. It turns out that with over a decade of security development and enhancement, a majority of tested websites are still vulnerable to the cookie stealing attack. In particular, we find out that some websites have taken effective security measures like two-factor authentication to completely block this attack, some have used less effective approaches like email reminder in case of abnormal user login events, and some do not have any protection at all. It is hard to imagine that most modern websites still show a lack of countermeasures to this basic security issue.

The contribution of this paper is twofold. First, we conduct a systematic evaluation of the cookie-stealing vulnerability on a list of 50 popular websites, including 46 Alexa 500 sites and 4 online banking sites; to the best of our knowledge, this is the first time that the cookie stealing attack is tested in a large scale on real world websites. The experiment results indicate a potentially dangerous phenomenon, which calls for closer attention from the security community. Second, we summarize a suite of defensive techniques and justify the effectiveness of them.

The rest of this paper is organized as follows. We start with as section of background review that covers that basics of web cookies as well as cookie stealing techniques. We then turn our attention to the experiment design. We provide results and insights in section 4. A discussion of feasible solutions is given in section 5. We then conclude the paper in section 6.

2 Background

2.1 Web Cookies

An HTTP cookie is a piece of information stored on your computer by a website you visit. This information can be account identifiers, tax information about a person, or even still basic shopping preferences from a website. A cookie may contain any information that a website wishes to collect and is authorized by a user to collect. This means, when a valid website receives permission to create and access cookies on a particular user's machine, the website generates the information to be stored in the cookie. The cookie generation algorithm is out of the scope of this paper, but it is usually an encoded and unique string that is difficult to predict/guess on the client side. The server embeds the cookie(s) in a "*Set-Cookie*" field added in the HTTP response message. Moreover, additional attributes like expiration time can also be set. Cookies are then carried along a HTTP connection with each interaction to a webpage to maintain the state of the client. To enhance the browsing experience, a web server may issue multiple cookies to remember different kinds of information from its client. There are three typical uses of cookies: session management, personalization, and tracking [1].

- Session management is implemented by a session cookie, which is a unique identifier between the particular client and the web server. A session cookie could be set for authentication purpose, as it is only issued once a user successfully completes the first-time authentication. Later on, the user comes back and is authenticated with the correct session cookie presented. If the cookie information is invalid or inaccurate, then typically a new login page will be displayed. Also, a session cookie can be used one time or multiple times. A one-time session cookie will only live within the live session, which becomes invalid once the session terminates. If an expiration time is set longer, a user can stay signed in even the browser is restarted.
- Personalization allows clients to store their browsing preference into cookies which persist until the cookie expires. For example, a user can specifier his/her preferred color scheme, font size, page layout, etc., which can be remembered by the browser.
- Tracking cookies have gained more popularity recently due to the booming application of Big Data and machine learning. A tracking cookie records a user's browsing history and habits, and is sent to the web server to provide custom service for the user. For example, an advertiser may extract the items a user browsed during online shopping from tracking cookies and provide recommendations.

2.2 Cookie Theft

Cookies pose both security and privacy concerns. From the privacy perspective, cookies may contain a user's browsing history, preference, and other personal information that should not be revealed to any third party. From the security perspective, a hacker who owns a user's session cookie could bypass web authentication and gain access to the user's web account as long as 1) the cookie is not expired, and 2) the web server is lack of protection for cookie-related attacks. The security of cookies depends on the cookie generation algorithm, which should create cryptographically secure and unique identifiers. In other words, a cookie must not be predicted by attackers for session hijacking. However, even if the cookie generation algorithm is good enough to block cookie prediction, hackers can still obtain a valid cookie by stealing it. A variety of cookie stealing techniques have been developed, as summarized below.

- **Network eavesdropping** is a way of man-in-the-middle attack. The idea is that the attacker intercepts the traffic between the client browser and web server, and then extracts the cookie from the HTTP header, if it is not encrypted. For example, it would be an ideal scenario to conduct this attack in an unencrypted WiFi channel as it is quite easy to capture traffic from end hosts in the same network. A simple solution would be incorporate a secure layer on top of HTTP to create an encrypted tunnel. VPN and HTTPS are the good options.
- **DNS poisoning** refers to DNS entries being fabricated by hackers so that HTTP requests are directed to servers controlled by hackers. For instance, a DNS entry originally maps domain "www.example.com" to IP 192.168.1.101, which is the correct IP of the server. Once the server is compromised, a hacker can change the IP, pointing the domain to a server under his/her control. As a result, all HTTP requests containing cookies can be collected by the malicious server. DNS

poisoning is usually caused because Internet Service Providers do not properly secure their DNS servers.

- **Cross-site scripting (XSS)** is another popular technique to steal cookies, although it is capable of causing much more damage. XSS enables hackers to embed a piece of script code into the HTTP response message, which will be rendered and executed by all browsers visiting the compromised web page. To steal the cookie, a hacker only needs to access the cookies through "*document.cookie*" in the script, and send the cookies to a remote server owned by the hacker through another HTTP request. This technique can be easily migrated by HTTPOnly cookies, which is a special kind of cookie that stops client side languages to read cookie values. However, [4] points out that HTTPOnly cookies are not widely adopted by modern websites due to various reasons.
- **Malware** can undoubtedly steal cookies, which are stored in the file system by browsers. An infected machine could leak cookies due to the malicious behavior of malware covertly running in the background.

2.3 Related Work

There are numerous prior studies on the matter ranging from analysis of how cookie stealing attacks can be launched to how they can be prevented. The authors also explore into further detail about how different attack can occur, since this is not limited to an XSS attack [5]. Another form of attack can come from phishing emails where a malicious program can be installed and cookies stolen or used to intercept HTTP packets through a browser session where cookies travel through. Another document goes into great detail about dynamic cookie rewriting [3]. As the user interacts with a website, the cookie is constantly being modified. This is to say that the cookie will be changing with each interaction the user has thus ensuring that if a cookie is stolen it will most likely be expired. Throughout this experiment this arose as a possible solution to the vulnerabilities with cookies and is explained in further detail in the next section.

HTTP-only cookies are introduced to prevent cookie-stealing attacks through XSS. With this property turned on, cookies cannot be retrieved through JavaScript. However, despite its simplicity, HTTP-only cookies are not widely adopted even after 14 years of its invention. Zhou and Evans [4] have conducted a survey to test the support of HTTP-only cookies for 50 popular websites from Alexa.com, and showed that over half of the tested websites did not use HTTP-only as of year 2010, including some famous ones like Amazon.com and Twitter.com. They also pointed out several reasons. One is that except cookie-stealing, HTTP-only cookies are unable to defend any other ways of XSS attacks, and developers tend to adopt more universal schemes to stop XSS. Our assessment results, however, shows that a vast majority of websites have adopted HTTP-only cookies as of 2016.

This paper differs from the previous studies in that it attempts to assess the defense of most popular websites against this particular cookie abuse attack, which, to the best of our knowledge, has not been done before.

3 Experiment Design

We conduct an empirical study to test the cookie vulnerability against 50 of the Alexa 500 websites [2]. The approach is to see if a stolen cookie can be transferred to another computer, and used to make a valid request to a webpage and eventually hijack a web session to gain access to another user's account without authentication. The previous section lists four typical approaches for cookie theft. However, only network eavesdropping and XSS can be defeated at the end of the web server, while DNS poisoning and malware still make cookie stealing possible. That being said, a website is unable to completely block cookie theft. Therefore, the focus of this study is not to test whether a cookie can be stolen or not, but to observe how a website responds in case of cookie theft.

For the purpose of this study, we have developed a piece of proof-of-concept malware that is assumed to be installed and run on the target machine for cookie stealing. In particular, the malware searches for cookie information in the computer's file system, delivers the found cookies to another computer, which will attempt to hijack a user's account. Table 1 shows the cookie file locations for different browsers on the Windows 10 file system. It turns out there is NO read-protection for these cookies files, which can be

easily stolen and transferred to a remote server. Three browsers, Chrome, Firefox, and Opera, choose to store cookies in a SQLite file, which is essentially a mini database that can be accessed by the browser.

Table 1: cookie files stored in the file system by different browsers on Windows 10

Browser	**Path to the cookie file**
Chrome	%LocalAppData%\Google\Chrome\User Data\Default\cookies
Firefox	%AppData%\Mozilla\Firefox\Profiles*random_string*.default\cookies.sqlite
Internet Explorer	%appdata%\Microsoft\Windows\cookies
Microsoft Edge	%LocalAppData%\MicrosoftEdge\cookies
Opera	%AppData%\Opera Software\Opera Stable\cookies

Two Windows machines with different IPs are used for this experiment. Machine one refers to the victim where cookies will be stolen from, and machine two refers to the attacker where the stolen cookies are reused to impersonate legitimate users. To test a website's response to cookie theft, a "cookie_guard" account is created for each tested website. There is significant manual work for this study as a login is required for each of the websites. In addition, the option "keep me signed in" will always be checked. Once logged in, session cookies will be stored in the browser and in the file system. We use Firefox as the testing browser for both the victim and the attacker. Before testing each website, the Firefox cookie files on both machines are deleted to ensure the freshness of newly received cookies. After the manual login, we close Firefox, and the malware will transport the cookie file to the attacker, which will directly plug it in the same location where Firefox stores its cookies. We then access the website again, and report the findings.

We have selected 50 websites, in which 46 sites are from the Alexa 500 websites. We have excluded the ones without login function and adult sites. In addition, we eliminate the ones that share an authentication system like YouTube and Google. The chosen websites are divided into five categories:

- **Portals (9)**: Google.com, Baidu.com, Yahoo.com, Qq.com, Live.com, Apple.com, Cntv.cn, Outbrain.com, and Usps.com.
- **Social media (10)**: Facebook.com, Twitter.com, Webo.com, Linkedin.com, vk.com, Instagram.com, Pinterest.com, Imgur.com, Pixnet.net, and Quora.com.
- **E-commerce (9)**: Amazaon.com, Taobao.com, Ebay.com, Paypal.com, en.Jd.com, Walmart.com, Booking.com, Bestbuy.com, and Newegg.com.
- **Content publishing (18)**: Dailymotion.com, Youku.com, Indeed.com, Slideshare.net, Yelp.com, Zillow.com, Tripadvisor.com, Douban.com, Wikipedia.com, Imdb.com, Graigslist.org, Github.com, Stackoverflow.com, Reddit.com, Wordpress.com, Spotify.com, Gawker.com, and Pandora.com, Dropbox.com.
- **Banking (4)**: chaseonline.chase.com, online.americanexpress.com, pnc.com, online.citibank.com.

4 Evaluation

4.1 Experiment Results

The evaluation results are reported by website categories in the following tables. For each tested website, we count the number of cookies received after a successful login, and the number of cookies with the HTTP-only attribute enabled. The testing result "**Success**" indicates that the stolen cookie can be used to hijack a session on another machine, i.e., the tested website is vulnerable to cookie theft. A "**Failure**" result means that stolen cookies cannot be used to gain access to the victim's account from another machine. A "**Partial Success**" means that the stolen cookies can help the attack access certain functions of the victim's account, but not all of them.

Table 2: Portals

Website	**# cookies**	**# HTTP-only cookies**	**Result**

Google.com	22	11	Success
Baidu.com	21	12	Success
Yahoo.com	17	5	Success
Qq.com	22	1	Failure
Live.com	57	9	Success
Apple.com	20	2	Partial Success
Cntv.cn	15	1	Success
Outbrain.com	47	1	Partial Success
Usps.com	20	2	Failure

Table 3: Social Network

Website	**# cookies**	**# HTTP-only cookies**	**Result**
Facebook.com	9	6	Success
Twitter.com	11	4	Success
Weibo.com	17	2	Success
Linkedin.com	24	5	Success
vk.com	16	5	Success
Instagram.com	7	1	Success
Pinterest.com	9	2	Success
Imgur.com	16	4	Success
Pixnet.net	16	0	Success
Quora.com	9	4	Success

Table 4: E-commerce

Website	**# cookies**	**# HTTP-only cookies**	**Result**
Amazon.com	11	0	Success
Taobao.com	42	2	Failure
Ebay.com	30	5	Success
Paypal.com	28	9	Failure
en.Jd.com	29	1	Success
Walmart.com	49	2	Failure
Booking.com	41	7	Success
Bestbuy.com	37	6	Failure
Newegg.com	67	5	Success

Table 5: Content Publishing

Website	**# Cookies**	**# HTTP-only cookies**	**Result**
Dailymotion.com	29	0	Success
Youku.com	24	0	Success
Indeed.com	14	2	Success

Slideshare.net	18	2	Success
Yelp.com	23	5	Success
Zillow.com	46	4	Success
Tripadvisor.com	22	3	Partial Success
Wikipedia.com	66	66	Success
Imdb.com	10	1	Success
Graigslist.org	2	0	Success
Github.com	6	3	Success
Stackoverflow.com	31	20	Success
Reddit.com	24	6	Success
Wordpress.com	18	6	Success
Spotify.com	26	6	Success
Gawker.com	26	3	Success
Pandora.com	13	0	Failure
Dropbox	62	3	Success

Table 6: Banking

Website	# cookies	# HTTP-only cookies	Result
chaseonline.chase.com	11	0	Failure
online.americanexpress.com	23	5	Failure
pnc.com	13	0	Failure
online.citibank.com	51	2	Failure

4.2 Result Analysis

Some general findings include:

- The results in the tables show that 36 of the tested 50 websites (72%) are vulnerable to the attack. Among the vulnerable sites we can find the notable Google, Facebook, Twitter, Amazon, and EBay, etc. Facebook can detect login from a different device and notify the user via email, but the attacker has no problem to login.
- All surveyed sites employ cookies to deliver a stateful service. Notably, when accessing a website, cookies not only come from the host being accessed, but also from other collaborative domains and sites. For example, when we sign in at google.com, the browser also receives cookies from accounts.youtube.com, google.com.hk, etc. In average, a site issues 25 cookies, but not all of them are related to session management.
- Only 7 out of 50 sites do not use HTTP-only cookies. This is not entirely aligned with the result in [4]. However, given five years passing by since the year of publication of [4], the community should have increased awareness of the usage of HTTP-only cookies in defense of XSS-based cookie theft. Given that a majority of sites have realized the importance of cookie protection, it is surprising that most of them did not take effective measures to deal with the consequence of cookie theft.
- Interestingly, for all websites that failed the attack, only Pandora can keep a user signed in. The rest sites employ one-time session cookies to prevent stolen cookies from being abused. We think that these one-time session cookies are reasonable to use for particular services like banking, but for most sites simply disabling the option of “keep me signed in” may not be a user-friendly

option, because it should be a user's decision to make. For the sites we surveyed, only Pandora keeps this option and can still block the attack.

We also provide a category-based analysis as follows:

- **Portals**: From the portals we surveyed, big companies such as Google, Baidu, and Yahoo suffer the cookie theft vulnerability. The Chinese Internet company QQ fails the attack on the portal site, however, its email service mail.qq.com is vulnerable. Apple and Outbrain have partially enabled one-time session cookies. This way, users need to re-input their login credentials to access sensitive functions like tracking orders. However, Apple does not protect usernames with one-time session cookies. Attackers with the stolen cookies can view the usernames from a different machine.
- **Social Network**: Surprisingly, all social networking sites we tested are vulnerable. Attackers can access a user's account with the stolen cookies and cause further damage.
- **E-Commerce**: For e-commerce sites, only three out of nine sites employ protected cookies. The vulnerable sites include Amazon, EBay, Newegg, etc. The Chinese flagship e-commerce sites Taobao and JD are able to defend against this attack, while the international version of JD is vulnerable. All three sites with failed testing result employ one-time session cookies, and they do not have the option of "keep me signed in".
- **Content Publishing**: For the 18 content publishing sites, only one failed the attack. Another site, Tripadvisor, uses one-time session cookies to protect certain functions such as the payment subsystem, while attackers can still view the user profile and check messages in the user's inbox.
- **Banking:** As expected, all tested banking sites well defended the attack. None of four sites have the "keep me signed in" option. That being said, the cookies will become invalid once the session is disconnected.

It should be noted that during this experiment another issue was identified. This would be the use of website logins that have known cookie vulnerabilities. In an effort to streamline the logging in process of many of these websites, these websites utilize the API's for many other services; most notable Google, Facebook, and Twitter. This means that should a person use one of these other services to log in, and that service have security flaw, then the target website is also vulnerable. For example, Sound Cloud allows a user to sign up and log in with a Facebook or Google account in addition to the traditional email / username option. This means that should Google or Facebook not provide some sort of security to their cookies, these vulnerable cookies can be used to log into other websites. As a consequence, a comprised Google account through cookie theft can be used to further compromise accounts of any other sites with account integration with Google.

5 Defensive Techniques

With the proof that stolen cookies can be maliciously used for account hijacking, the paper now turns to defense against the cookie vulnerability. Before discussing the concrete security measures, we propose a set of criteria to evaluate the solution quality.

- **Efficiency**: the scheme should be efficient, meaning that the scheme should not involve time-consuming operations like database lookups or cryptographic functions.
- **Easiness of implementation**: the solution does not need to change the current Internet cookie specification.
- **Easiness of deployment**: the solution does not require major change in either client side or server side.

A principle to block the attack would be binding session cookies with a specific browser on a specific computer, and ensure that the cookie value, the browser version, and the computer IP will not be leaked, faked, and abused. It is the website's responsibility to add more security policies, while leaving the client side unchanged.

5.1 IP Address

One method of making a cookie more secure would be to include another piece of information to identify the user. Each computer has an IP address can be another element to the cookie. Facebook is able to detect a login action from a different device, meaning IP information has been used for security purpose. However, it is still not enough because a site needs to bind an IP with a set of cookies issued to that IP to stop cookie abuse. When the user's browser sends a login request to the server, the server then obtains the IP address of the computer. Then, every subsequent request to the server for a webpage would include that information and the server can check against the validity of the IP address adding another factor of security. This information would be hard to duplicate as the attacker cannot simply use a fake IP while still keeping a valid connection. Additionally, any hacker would have difficulty with just using the cookie, because the IP address would be hashed inside the cookie. This means that when the server receives a valid request, it would validate the cookie login information and then compare the IP address with the IP address in the hashed cookie.

A benefit to this method would be that if a hacker obtained a cookie and the IP's didn't match, the hacker would be prompted to log in. However a drawback to this method would be that it would not be difficult to spoof and IP address considering that hacker obtained the cookie from the user. This more or less provides a nuisance to the hacker and is simply more of deterrent.

5.2 Expiration Times

Another security feature that could be added would be including an expiration time, while an expiration time alone is unable to block the attack. However, would help ensure that a cookie only last a certain length of time and would expire. Fortunately, nearly all websites we tested have enabled this feature, which would at least narrow the window of time that any unauthorized user could utilize a stolen cookie. Depending on the types of application, the expiration time can be adjusted.

Overall the benefit to this method is it narrows the window in which a hacker has to exploit a stolen cookie. However the drawback to this would the cookie still has some validity and if a hacker scales back the number of targeted users this time may always be short enough for exploitation. This method just reduces the scale of potential attacks.

Another drawback to this method would come in the form of an inconvenience for the user. Since a time based cookie would cause the website session to expire and thus force the user to log in again. There would be a tradeoff. The shorter the time for the cookie, the more secure it would be. However, the shorter the time is until the cookie expires would require a greater number of logins in a session for a user. This would ultimately become frustrating for the user.

5.3 Changing Cookie Key

A security feature could be one time use keys placed into the cookies. What this would do is allow a cookie to be used only once and then a new cookie would be placed in the cookie file with each and every request. An example would be as follows.

A user logs into their bank website with their credentials. A cookie with a special key is generated and sent to the user as well an expiration time. The user selects a different webpage within the website and is directed to that page. The cookie will be sent back to the server where it will compare the contents of the server. Once the server verifies that this is the correct user, it updates itself with a new special key and sends the new cookie to back to the user. Each and every time the user makes a page request this happens updating the cookie on the user's end. Should the cookie be left for too long this expiration time will cause the cookie to expire and be rendered useless. Paper [3] discusses in depth a similar approach; the notion of using a dynamically changing key.

There are two reasons why this is believed to be a secure method. First, the expiration of the cookie will occur before the hacker has a chance to use the stolen cookie or the user will have requested a different webpage thus causing the server to update the local cookie only and not the stolen one. This means that when a hacker attempts to log in the special key of the cookie will not line up and the hacker cannot gain access to the user's account.

The same benefits apply to the expiration of time method but improves upon it by shrinking the window even further. A major drawback to this method means that increased bandwidth and communication with the server which. Now the server would not only just be responsible for returning just the html code each page request, but authentication, cookie fabrication, and page generation for each navigation.

6 Conclusion

Overall, stolen cookies pose an enormous problem to the web security community. They have existed for some time now and are well documented across numerous websites. There appear to be a few plausible solutions as well as a few theoretical solutions. The implementation mostly falls all on the host's desire to implement such security features or to enhance their current server capabilities both through software and hardware. This security problems falls to the servers and their ability to protect their users. The likelihood that things will change are slim until a large security breached. A security breach will most likely prompt the companies to enhance security through many forms, hopefully including cookie security.

A user can protect themselves by ensuring that they run malware and virus scanners thus keeping their computer clean. Additionally, making sure their websites are trusted websites before visiting. A user can clear their cookie history to ensure they cookies are not stolen and always monitor their accounts to ensure that fraudulent activities do not occur. There is still the potential for rigorous and in depth exploration into this topic and this paper could serve as a foundation for future research.

References

[1] Wikipedia, HTTP cookie, [online]. Available: *https://en.wikipedia.org/wiki/HTTP_cookie*
[2] Alexa.com, 'Alexa Top 500 Global Sites'. [Online]. Available: http://www.alexa.com/topsites
[3] Putthacharoen, R., & Bunyatnoparat, P. (2011, February). Protecting cookies from cross site script attacks using dynamic cookies rewriting technique. In Advanced Communication Technology (ICACT), 2011 13th International Conference on (pp. 1090-1094). IEEE.
[4] Zhou, Y., & Evans, D. (2010). Why aren't HTTP-only cookies more widely deployed. Proceedings of 4th Web, 2.
[5] Takahashi, H., Yasunaga, K., Mambo, M., Kim, K., & Youm, H. Y. (2013, July). Preventing Abuse of Cookies Stolen by XSS. In Information Security (Asia JCIS), 2013 Eighth Asia Joint Conference on (pp. 85-89). IEEE.