South Hills School of Business and Technology

Web Login Security:

Session Management and Its Vulnerabilities

Ian Paulhamus, Rishi Desai

Server-Side Web Development – IT 241

Nicholas Page

November 4th, 2024

In today’s digital landscape, people’s personal data flows freely across global networks. Due to this, the security of user authentication has become more important than ever. The thievery of consumer data is not only a threat to corporations, but a threat to everyone who is a victim of such attacks. The backbone behind user logins is a concept known as sessions. Sessions allow the end user to communicate with a server using their credentials, verified by a session ID. Every web session has the threat of vulnerabilities, such as session hijacking and session fixation, which allow attackers to silently control user sessions to steal sensitive information or impersonate users. As these threats continue to evolve, understanding how PHP manages sessions, as well as how to secure them has never been more crucial. Developers of web applications have an obligation to strengthen their applications against session-based attacks to protect the privacy of users.

In the modern day, sessions are one of the crucial pieces of web development, as they fix a major oversight that webpage communications have, which is maintaining a state. HTTP requests are the most common method that web pages use to communicate between the client and server. While the protocol works well for delivering information to the client, a major flaw that it has is that it cannot maintain a state on its own. This means that the server will not remember any past actions the user makes and will send the same data no matter what. To alleviate this problem, sessions save user logins and other information in a browser cookie. The server authenticates this cookie through an identifier known as the session ID. A session ID is a unique identifier for each time a user visits a website, the client sends this ID back to the server each time the user makes a request. The use of sessions and their session IDs allows webpages to personalize the experience for the client while also allowing sites to manage and record information for each specific session.

To utilize sessions in their web pages, developers need a way for the site to interact with sessions dynamically. PHP is one of the most widely used server-side languages and is especially prevalent in the handling of user sessions. While the language may be on a slow decline, most sites still utilize PHP for session management and user logins. PHP offers built-in tools for managing sessions, with a collection of methods that allow developers to easily create and interface with them. PHP session data is stored in a super-global value named ‘$\_SESSION,’ which can be read by any function within the program. With these tools from PHP in hand, developers can easily implement session management into their web pages.

A computer code on a black background

Description automatically generatedMost PHP session implementations follow a similar set of steps while working with the medium. The first of these steps is checking if the user has an existing session ID; if not, the program generates a new one. Next, the program creates a cookie that stores the session information on the client’s device. Lastly, the data from this session cookie is saved to the ‘$\_SESSION’ super-global, which includes the session ID.

Example of The Construction of a Session:

These tools PHP provides are incredibly useful; however, default configurations within implementations of sessions may leave applications vulnerable to attacks. If an attacker is able to steal a session ID, it could lead to sensitive information being taken from a login. By understanding the works of session vulnerabilities and effective security measures, developers can significantly reduce the risks by session-based attacks.

Because sessions provide the opportunity for hackers to login to user accounts, they are a prime target for vulnerability exploitations. The most common methods that attackers will use to attempt to get user information from sessions is through session hijacking or session fixation. These two methods have multiple similarities, as they are both attacks that take advantage of session IDs to hack user accounts; however, they have significant differences in the way that they access the correct session ID. When Session hijacking occurs, an attacker gains access to a user’s active session by obtaining their session ID and using it on their own device to hack into accounts and make changes without the user knowing. This type of attack often targets users on unencrypted networks such as public Wi-Fi. Attackers use techniques that involve capturing unencrypted network traffic, such as a Man-In-The-Middle (MITM) attack. By contrast, session fixation gives the victim a bad session ID that the attacker knows, which the user unknowingly authenticates for them within the hacker’s session. The attacker will generate a session ID through the website before starting the attack, then they use a malicious URL to obtain login information from the user that will authenticate the cookie they generated. Both session hijacking and session fixation are serious vulnerability concerns, which could have dangerous consequences if attackers are able to successfully perform them.

When it comes to session related threats such as session hijacking and session fixation, it is imperative that developers take steps within their implementations of session management to protect against these threats. One of these methods includes session regeneration. Session regeneration is a form of security improvement that helps limit access to a session by regenerating a new login ID after logging in. Implementing this into an application will help significantly with session security, voiding old session IDs after they grant access to the user. This is especially powerful against session fixation because it gives the attacker an invalid ID. Another method that developers can use to help prevent session attacks in their programs is enforcement of session expiration. Having frequent checks in code for session expiration and ensuring that sessions are only active for a certain amount of time will reduce the chances that an attacker will be able to steal a valid ID. One last example of session protection a developer could implement into their sites is the use of HTTPs. HTTPs is a much more secure protocol than HTTP because it encrypts data during transit, making it far less likely that a MITM attack will yield any results for the attacker.

A computer screen with white text

Description automatically generated A Session Security “.inc” File:

Using these methods and more, a developer can help ensure that their sites take the proper steps to prevent disastrous attacks against their users.

Preventative measures can be utilized within programs to reduce the risk of session related attacks; however, there are still a multitude of these attacks that happen regularly in the real world. One of the largest examples of this in the recent year is an account hijacking attack that happened to Google accounts in late 2023. Hackers found an exploit in the code Google used for session cookies, allowing them to obtain access to user’s accounts, bypassing the measures that Google had in place to prevent these attacks. The exploit came from a vulnerability in a part of Google’s authentication called “MultiLogin,” which allowed bypassing of the session security they had in place. This exploit was enabled by a form of malware on the client-side device, which hackers would then use to hijack sessions and regenerate their own cookies using the hacked user’s credentials. This attack is a glaring example that even the biggest companies holding sensitive user information can be a victim to these attacks, highlighting the importance of security in session management.

While clients use the modern-day internet, clients are constantly passing sensitive information back and forth with domains. If this data is routed insecurely, it is a threat to everyone. Developers must make their best efforts to prevent attacks that could happen within any program that manages user data. This is especially important when we think about session management, as the user session is the key to unlocking accounts and getting unlimited access to the information a user has provided to the given service. Session security is an ever-evolving process that requires all developers using languages such as PHP to script sites to be constantly aware of new threats that may face their programs at any given time. While this may be an arduous process, it is one that is necessary to ensure that personal data stays safe from session breaches such as session hijacking, session fixation, and more to come.

Resources:

* <https://www.descope.com/learn/post/session-fixation>
* <https://www.descope.com/blog/post/refresh-token-rotation>
* <https://www.imperva.com/learn/application-security/session-hijacking/>
* <https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html>
* <https://www.pixelstech.net/article/1356723388-How-does-PHP-session-work>
* <https://github.com/yakeing/php_session/blob/main/src/session.php>
* <https://latesthackingnews.com/2024/01/01/multiple-malware-exploit-google-cookie-flaw-for-session-hijacking/>