Assignment 1 – Broken Access Control & Cryptographic Failures

WEBD2075

By Mohammad Al Jokhadar

Question #1:

(~ 2 pages, min 1, max 3, 12pt double spacing with proper citation in IEEE style[1])

Broken Access Control

Broken Access Control is the Open Worldwide Application Security Project’s leading vulnerability. Why would that be a tremendous worry on our minds? Let us first start by defining what Broken Access Control is, then we would ponder an example of the different forms that OWASP’s leading vulnerability manifests into our digital existence.

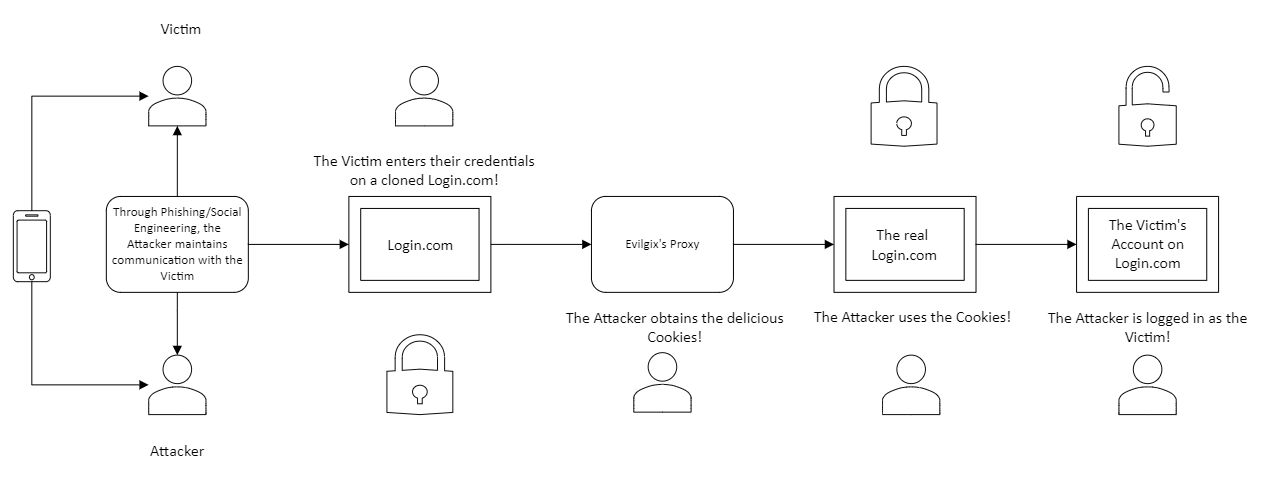
Access Control is a concept involving what we can do, and what we can not do. This applies in the digital realm, as well as our physical world. Every location that exists in our cities has access control established in place. It governs who can visit the location, what rights and privileges do they have, and what they can not do. In the digital realm, is it what a user can and can not do within the application, operating system and the device itself. The Administrator is the entity in power of permissions, and they reign supreme. They have the power to grant permissions, provide access to certain users or roles to certain files and folders, and define the limitations of each user. When the rules are set in place, users can not go beyond the limitations of access control.

Although, there are ways of bypassing access control limitations altogether, and this can cause catastrophic results. A misconfiguration could lead to unauthorized individuals obtaining the powers of the supreme Administrator, and the results could be the exposure of sensitive information to an un-intended audience, altering the information or the sheer destruction of data. The vulnerabilities under this category are numerous, such as using a session token belonging to another user with the right privileges by an unauthorized user ( such as the attacker) , utilizing the lack of access controls on API calls, modifying the URL of a web page to bypass access control, and so on [1]. To demonstrate the severity of this category of vulnerabilities, we shall discuss Elevation of Privilege by utilizing a stolen Session Cookie.

We shall use a tool named Evilginx 3.0 [2], which is a monster-in-the-middle framework. How it works is via creating a near-exact duplicate of current login pages, which use multi-factor authentication. Evilginx would intercept the taget’s login session, since the cloned website is hosted by the attacker, and then provides the session token for our use. This is how we would have administrator privileges over the target’s account. After the cloned page is loaded, Evilginx could generate a shortened link refered to as a *Lure* for the cloned page. The attacker could entice or social-engineer the target into clicking on the link ( for example, it could be via a QR code pointing to the attacker’s cloned page). In ideal circumstances, the attacker would host the cloned page on an external domain and would have generated the Certificate for the cloned page to avoid an error message on the target’s browser, indicating that the site may not be secure.

When the target arrives at the cloned page, it would appear as an almost exact copy of the genuine login page, with the exception of the link itself. This is done to confuse the target in judging the link as genuine ( such as choosing *mircosoft.com* instead of *microsoft.com*). When the target logins into the cloned page, Evilginx presents the attacker with the Session Cookie. The later prize is used by the attacker by editing the current browser Session Cookie, and inserting the stolen Cookie gained by Evilginx.

The stage in the attack, when the attacker uses the stolen Cookie, is the demonstration of Broken Access Control, due to the elevation of privileges via utilizing a Session Identifier.



# Cryptographic Failures

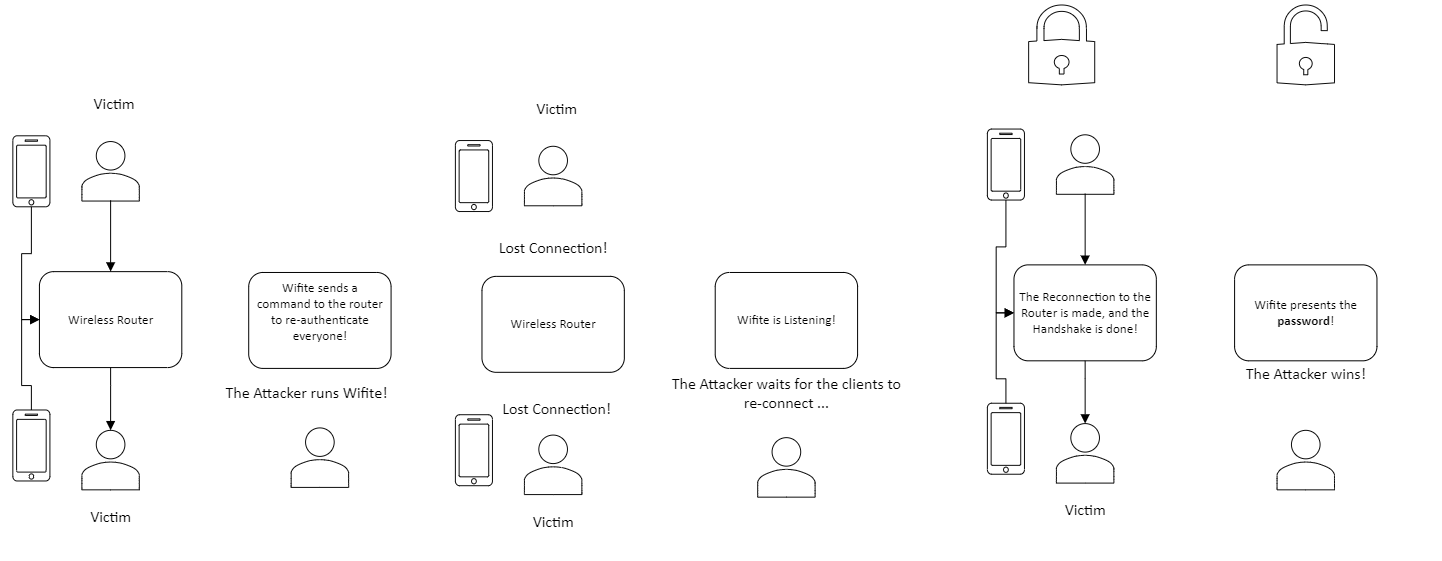
We do the things we do in Cybersecurity to maintain the confidentiality, integrity and availability of data and non-data assets. Since the consequences of a security breach would range in severity , one of the most feared repercussions is damage to the organization’s reputation. This is the scenario no one would want unto themselves, as it is a heavy weight to bear. Thus, encryption is key to maintaining the desired level of confidentiality and integrity of the data assets.

To know how to protect an asset, we should know the threats that adversaries are utilizing in their arsenal. Cryptographic Failures is one of the top threats listed by the Open Worldwide Application Security Project. Its prevalence lead it to the second spot, which is indicative of the vulnerability’s utilization. Today, we will discuss a scenario that involves a recently-deafeated cryptographic algorithm used in Wireless Area Networks, and it is the WPA2 encryption protocol.

Cryptographic failures can take many forms, but the common patterns are the usage of outdated or depreceted cryptographic algorithms, utilization of weak/outdated Hash Functions, insecure transmission of data, lack of encryption usage, improper use of the elements of the cryptographic formula to encrypt/decrypt information, and so on. The vulnerability’s root stems from weak cryptographic elements with human error added in to the mix [3].

In our Wireless Network example, the WPA2 encryption, like many of its predecessors, was made with confidentiality in its core. The driving force behind encryption is the race between cryptographers and malicious actors towards the solution of the cryptographic riddle. The challenge is accepted on both sides of the fence. Every encryption algorithm’s complexity or ingenuity acts as its lifetime period, as it gets replaced by a newer rendition of the algorithm to maintain the confidentiality of our data. The WPA2 encryption algorithm has been defeated, and this ushered in its descendant WPA3. The Wifite tool is utilized in the following scenario, which highlights the gravity of this vulnerability:

Wifite requires the presence of different libraries, such the Aircrack-ng and Airreplay-ng for the purposes of capturing the information that is exchanged when the client joins a wireless network, and for sending a message to the router to kick everyone out of the wireless network. This strategy begins with the message to the router to re-authenticate every client, which works to force every client to provide the wireless network’s credentials. Following this event, we would find Wifite working its magic of capturing the outcome of the conversation between the client and the router. The tool is extremely quick in capturing and cracking the password provided by the client. The attacker at this point would have plenty of opportunities to do further malicious actions, such as acting as a Monster-in-The-Middle for information interception. Thus, it is recommended to upgrade to a router that features the WPA3 encryption, or use the transitional WPA2/3 Encryption Protocol [4].



# Sources

[1]

“Broken Access Control | OWASP Foundation.” Accessed: Sep. 27, 2024. [Online]. Available: <https://owasp.org/www-community/Broken_Access_Control>

[2]

“Evilginx 2 - Next Generation of Phishing 2FA Tokens,” BREAKDEV. Accessed: Sep. 27, 2024. [Online]. Available: <https://breakdev.org/evilginx-2-next-generation-of-phishing-2fa-tokens/>[3]

“A02 Cryptographic Failures - OWASP Top 10:2021.” Accessed: Sep. 27, 2024. [Online]. Available: <https://owasp.org/Top10/A02_2021-Cryptographic_Failures/>

[4]

derv, *derv82/wifite2*. (Sep. 27, 2024). Python. Accessed: Sep. 27, 2024. [Online]. Available: <https://github.com/derv82/wifite2>