Credential Stuffing Prevention Cheat Sheet

Introduction

This cheatsheet covers defences against two common types of authentication-related attacks: credential stuffing and password spraying. Although these are separate, distinct attacks, in many cases the defences that would be implemented to protect against them are the same, and they would also be effective at protecting against brute-force attacks. A summary of these different attacks is listed below:

Attack Type	**Description**
Brute Force	Testing multiple passwords from dictionary or other source against a single account.
Credential Stuffing	Testing username/password pairs obtained from the breach of another site.
Password Spraying	Testing a single weak password against a large number of different accounts.

Multi-Factor Authentication

Multi-factor authentication (MFA) is by far the best defense against the majority of password-related attacks, including credential stuffing and password spraying, with analysis by Microsoft suggesting that it would have stopped 99.9% of account compromises. As such, it should be implemented wherever possible; however, depending on the audience of the application, it may not be practical or feasible to enforce the use of MFA.

In order to balance security and usability, multi-factor authentication can be combined with other techniques to require for 2nd factor only in specific circumstances where there is reason to suspect that the login attempt may not be legitimate, such as a login from:

- A new browser/device or IP address.
- An unusual country or location.
- Specific countries that are considered untrusted.
- An IP address that appears on known block lists.
- An IP address that has tried to login to multiple accounts.

- A login attempt that appears to be scripted rather than manual.

Additionally, for enterprise applications, known trusted IP ranges could be added to an allow list so that MFA is not required when users connect from these ranges.

Alternative Defenses

Where it is not possible to implement MFA, there are many alternative defenses that can be used to protect against credential stuffing and password spraying. In isolation none of these are as effective as MFA, however if multiple defenses are implemented in a layered approach, they can provide a reasonable degree of protection. In many cases, these mechanisms will also protect against brute-force or password spraying attacks.

Where an application has multiple user roles, it may be appropriate to implement different defenses for different roles. For example, it may not be feasible to enforce MFA for all users, but it should be possible to require that all administrators use it.

Secondary Passwords, PINs and Security Questions

As well as requiring a user to enter their password when authenticating, they can also be prompted to provide additional security information such as:

- A PIN
- Specific characters from a secondary passwords or memorable word
- Answers to security questions

It must be emphasised that this **does not** constitute multi-factor authentication (as both factors are the same - something you know). However, it can still provide a useful layer of protection against both credential stuffing and password spraying where proper MFA can't be implemented.

CAPTCHA

Requiring a user to solve a CAPTCHA for each login attempt can help to prevent automated login attempts, which would significantly slow down a credential stuffing or password spraying attack. However, CAPTCHAs are not perfect, and in many cases tools exist that can be used to break them with a reasonably high success rate.

To improve usability, it may be desirable to only require the user solve a CAPTCHA when the login request is considered suspicious, using the same criteria discussed above.

IP Block-listing

Less sophisticated attacks will often use a relatively small number of IP addresses, which can be block-listed after a number of failed login attempts. These failures should be tracked separately to the per-user failures, which are intended to protect against brute-force attacks. The block list should be temporary, in order to reduce the likelihood of permanently blocking legitimate users.

Additionally, there are publicly available block lists of known bad IP addresses which are collected by websites such as AbuseIPDB based on abuse reports from users.

Consider storing the last IP address which successfully logged in to each account, and if this IP address is added to a block list, then taking appropriate action such as locking the account and notifying the user, as it likely that their account has been compromised.

Device Fingerprinting

Aside from the IP address, there are a number of different factors that can be used to attempt to fingerprint a device. Some of these can be obtained passively by the server from the HTTP headers (particularly the "User-Agent" header), including:

- Operating system
- Browser
- Language

Using JavaScript it is possible to access far more information, such as:

- Screen resolution
- Installed fonts
- Installed browser plugins

Using these various attributes, it is possible to create a fingerprint of the device. This fingerprint can then be matched against any browser attempting to login to the account, and if it doesn't match then the user can be prompted for additional authentication. Many users will have multiple devices or browsers that they use, so it is not practical to block attempts that do not match the existing fingerprints.

The fingerprintjs2 JavaScript library can be used to carry out client-side fingerprinting.

It should be noted that as all this information is provided by the client, it can potentially be spoofed by an attacker. In some cases spoofing these attributes is trivial (such as the "User-Agent") header, but in other cases it may be more difficult to modify these attributes.

Require Unpredictable Usernames

Credential stuffing attacks rely on not just the re-use of passwords between multiple sites, but also the re-use of usernames. A significant number of websites use the email address as the username, and as most users will have a single email address they use for all their accounts, this makes the combination of an email address and password very effective for credential stuffing attacks.

Requiring users to create their own username when registering on the website makes it harder for an attacker to obtain valid username and password pairs for credential stuffing, as many of the available credential lists only include email addresses. Providing the user with a generated username can provide a higher degree of protection (as users are likely to choose the same username on most websites), but is user friendly. Additionally, care needs to be taken to ensure that the generated username is not predictable (such as being based on the user's full name, or sequential numeric IDs), as this could make enumerating valid usernames for a password spraying attack easier.

Defense in Depth

The following mechanisms are not sufficient to prevent credential stuffing or password spraying attacks; however they can be used to make the attacks more time consuming or technically difficult to implement. This can be useful to defend against opportunistic attackers, who use off-the-shelf tools and are likely to be discouraged by any technical barriers, but will not be sufficient against a more targeted attack.

Multi-Step Login Processes

The majority of off-the-shelf tools are designed for a single step login process, where the credentials are POSTed to the server, and the response indicates whether or not the login attempt was successful. By adding additional steps to this process, such as requiring the username and password to be entered sequentially, or requiring that the user first obtains a random CSRF Token before they can login, this makes the attack slightly more difficult to perform, and doubles the number of requests that the attacker must make.

Require JavaScript and Block Headless Browsers

Most tools used for these types of attacks will make direct POST requests to the server and read the responses, but will not download or execute JavaScript that was contained in them. By requiring the attacker to evaluate JavaScript in the response (for example to generate a valid token that must be submitted with the request), this forces the attacker to either use a real browser with an automation framework like Selenium or Headless Chrome, or to implement JavaScript parsing with another tool such as PhantomJS. Additionally, there are a number of techniques that can be used to identify Headless Chrome or PhantomJS.

Please note that blocking visitors who have JavaScript disabled will reduce the accessibility of the website, especially to visitors who use screen readers. In certain jurisdictions this may be in breach of equalities legislation.

Identifying Leaked Passwords

When a user sets a new password on the application, as well as checking it against a list of known weak passwords, it can also be checked against passwords that have previously been breached. The most well known public service for this is Pwned Passwords. You can host a copy of the application yourself, or use the API.

In order to protect the value of the source password being searched for, Pwned Passwords implements a k-Anonymity model that allows a password to be searched for by partial hash. This allows the first 5 characters of a SHA-1 password hash to be passed to the API.

Notify users about unusual security events

When suspicious or unusual activity is detected, it may be appropriate to notify or warn the user. However, care should be taken that the user does not get overwhelmed with a large number of notifications that are not important to them, or they will just start to ignore or delete them.

For example, it would generally not be appropriate to notify a user that there had been an attempt to login to their account with an incorrect password. However, if there had been a login with the correct password, but which had then failed the subsequent MFA check, the user should be notified so that they can change their password.

Details related to current or recent logins should also be made visible to the user. For example, when they login to the application, the date, time and location of their previous login attempt could be displayed to them. Additionally, if the application supports concurrent sessions, the user should be able to view a list of all active sessions, and to terminate any other sessions that are not legitimate.

References

- OWASP Credential Stuffing Article
- OWASP Automated Threats to Web Applications
- Project: OAT-008 Credential Stuffing, which is one of 20 defined threats in the OWASP Automated Threat Handbook this project produced.