ZROC102

zumaroc

Module 3: Testing Common Web Application Vulnerabilities (2)

BRUTE FORCING & RATE LIMITING

❖ What is a Bruteforce Attack?

A brute force attack is a method to determine an unknown value by using an automated process to try many possible values.

The most common type of a brute force attack in web applications is an attack against log-in credentials. Since users need to remember passwords, they often select easy to memorize words or phrases as passwords, whereby making a brute force attack using a dictionary useful.

❖ How does rate-limiting work?

Rate limiting runs within an application, rather than running on the web server itself. Typically, rate limiting is based on tracking the IP addresses that requests are coming from and tracking how much time elapses between each request. The IP address is the main way an application identifies who or what is making the request.

A rate limiting solution measures the amount of time between each request from each IP address and measures the number of requests within a specified timeframe. If there are too many requests from a single IP within the given timeframe, the rate limiting solution will not fulfill the IP address's requests for a certain amount of time.

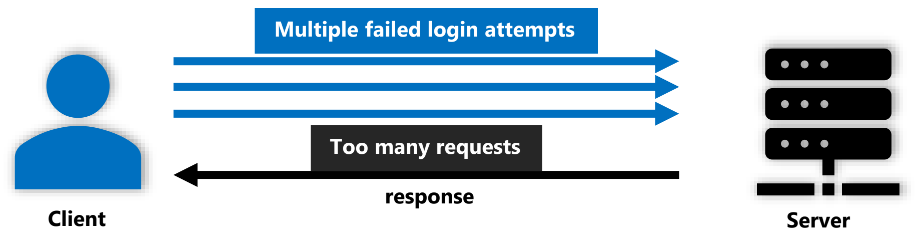
Essentially, a rate-limited application will say, "**Hey, slow down**," to unique users that are making requests at a rapid rate. This is comparable to a police officer who pulls over a driver for exceeding the road's speed limit, or to a parent who tells their child not to eat so much candy in such a short span of time.

BRUTE FORCING & RATE LIMITING (CONT.)

❖ How does rate-limiting work with user logins?

Users may find themselves locked out of an account if they unsuccessfully attempt to log in too many times in a short amount of time. This occurs when a website has login rate limiting in place.

This precaution exists, not to frustrate users who have forgotten their passwords, but to block brute force attacks in which a bot tries thousands of different passwords in order to guess the correct one and break into the account. If a bot can only make 3 or 4 login attempts an hour, then such an attack is statistically unlikely to be successful.



BRUTE FORCING MITIGATION

❖ Solutions

- ✓ **Account Lockout** - To lock out accounts after a defined number of incorrect password attempts e.g. After four(4) wrong attempts. Account lockouts can last a specific duration, such as one hour, or the accounts could remain locked until manually unlocked by an administrator.
- ✓ **Captchas** - They work by presenting some test that is easy for humans to pass but difficult for computers to pass; therefore, they can conclude with some certainty whether there is a human on the other end.
- ✓ **Two factor authentication (2FA)** - 2FA is an extra layer of security used to make sure that people trying to gain access to an online account are who they say they are.

  First, a user will enter their username and a password. Then, instead of immediately gaining access, they will be required to provide another piece of information. This second factor could come from one of the following categories:

  - ❑ **Something you know:** This could be a personal identification number (PIN), a password, answers to "**secret questions**" or a specific keystroke pattern
  - ❑ **Something you have:** Typically, a user would have something in their possession, like a credit card, a smartphone, or a small hardware token
  - ❑ **Something you are:** This category is a little more advanced, and might include biometric pattern of a fingerprint, an iris scan, or a voice print

- ✓ **Fortify passwords** - By using **Strong password** that must have (Uppercase letters, Lowercase letters, numbers and special characters).

PARAMETER TAMPERING

❖ What is Parameter Tampering?

The Web Parameter Tampering attack is based on the manipulation of parameters exchanged between client and server in order to modify application data, such as user credentials and permissions, price and quantity of products, etc. Usually, this information is stored in cookies, hidden form fields, or URL Query Strings, and is used to increase application functionality and control.

Example:

When a web application uses hidden fields to store status information, a malicious user can tamper with the values stored on their browser and change the referred information. For example, an e-commerce shopping site uses hidden fields to refer to its items, as follows:

```
<input type="hidden" id="1008" name="cost" value="70.00">
```

In this example, an attacker can modify the "**value**" information of a specific item, thus lowering its cost.

PARAMETER TAMPERING MITIGATION

❖ **Solutions**

- ✓ Use a whitelist format for the application’s inputs.
- ✓ Use web application firewalls for utmost protection.
- ✓ Encrypt the session cookies to prevent tampering.
- ✓ If the cookie originated from the client-side, such as a referrer it should not be used to make any security decisions.
- ✓ Avoid including parameters into the query string.

ACCOUNT TAKEOVER

❖ What is Account Takeover?

Account takeover is a form of identity theft and fraud, where a malicious third party successfully gains access to a user's account credentials

❖ Account Takeover Techniques

- ❑ **Hacking:** There are multiple hacking techniques used by ATO attackers – the most popular type is a brute force attack, where the cybercriminal develops automated scripts that churn through password combinations, hoping to generate a successful login key.
- ❑ **Phishing & Spear Phishing:** Cybercriminals will use email correspondence to trick users into to revealing their personal information. While phishing emails can be automated and easier to spot, spear phishing emails are highly targeted and much more deceptive.
- ❑ **Social Engineering:** Account takeover perpetrators will spend time researching across open databases and social media, looking for pertinent information like name, location, phone number, or names of family members – anything that will assist in guessing a password.
- ❑ **Botnets:** Hackers will deploy bots to hack into customers' accounts – bots can plug in commonly-used passwords and usernames to perform high-volume, rapid attacks and take over the maximum number of accounts, all while staying hidden from immediate view. Because bots deploy from multiple locations, it's harder to identify malicious IP addresses logging in.
- ❑ **Credential Stuffing:** Credentials stolen from or leaked from various businesses (or purchased from the dark web) are tested against multiple websites, in the hopes of catching a victim who hasn't realized their login information is compromised.

EXPLOITING FILE UPLOADS

❖ What is Local File Inclusion (LFI)?

LFI is a vulnerability which an attacker can exploit to include/read files.

❖ How LFI does it occur?

LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement.

❖ The Negative Impact:

- ❑ **Denial of Service (DOS):** is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic or sending it information that triggers a crash.
- ❑ **Remote Code Execution:** is a cyber-attack whereby an attacker can remotely execute commands on a victims target machine. RCEs usually occur due to malicious malware downloaded by the host and can happen regardless of the geographic location of the device.
- ❑ **Sensitive Information/Data Exposure:** When an application, company, or other entity inadvertently exposes personal data.

PATH TRAVERSAL (EXAMPLE)

Example:

In these examples it’s possible to insert a malicious string as the variable parameter to access files located outside the web publish directory.

```
http://some_site.com.br/get-files?file=../../../../some dir/some file
http://some_site.com.br/../../../../some dir/some file
```

FILE UPLOAD MITIGATION

❖ **Solutions**

- ✓ The file types allowed to be uploaded should be restricted to only those that are necessary for business functionality.
- ✓ Never accept a filename and its extension directly without having an allow list filter.
- ✓ It is recommended to use an algorithm to determine the filenames. For instance, a filename can be a MD5 hash of the name of file plus the date of the day.
- ✓ All the control characters and Unicode ones should be removed from the filenames and their extensions without any exception. Also, the special characters such as "**;**", "**:**", "**>**", "**<**", "**/**" ,"\", additional "**.**", "*****", "**%**", "**$**", and so on should be discarded as well.
- ✓ Log users' activities. However, the logging mechanism should be secured against log forgery and code injection itself.
- ✓ Try to use **POST** method instead of **PUT (or GET!)**
- ✓ Ensure that files with double extensions (e.g., "file.php.txt") cannot be executed especially in Apache.
- ✓ Ensure that uploaded files cannot be accessed by unauthorized users.

SOURCES

Bruteforce
http://projects.webappsec.org/w/page/13246915/Brute%20Force

Rate Limiting
https://www.cloudflare.com/learning/bots/what-is-rate-limiting/

Parameter Tampering
https://owasp.org/www-community/attacks/Web_Parameter_Tampering

Account Takeover
https://www.barracuda.com/glossary/account-takeover

Path Traversal
https://owasp.org/www-community/attacks/Path_Traversal