Nome	RA
Caio Von Atzingen Pfeilsticker	25.01488-7
Felipe Abrantes Pereira Lima	25.00211-4

Capacitação 3

Prática 1

Exercício 1 - BruteForce com Hydra

Primeira coisa que fiz ao entrar no site foi verificar os elementos da página e da URL. Não achei nada de interessante, então fui procurar aprender mais sobre BruteForce já que era o que foi proposto no título.

Pesquisei sobre algumas ferramentas de BruteForce, e dentre elas achei o hydra, assim, vi que ele possui diversos métodos, http-get, ssh, e outros. Contudo, um me chamou atenção, o http-post-form. Pois eu conseguiria obter todas as informações necessárias.

Então, com essa ferramenta do Hydra que ao especificar o usuário (o qual eu já tinha), especificar uma wordlist para a senha, o IP da página (que estava na URL), a porta (também estava) e por fim user=^USER^&password=^PASS^:Invalid Password!", que serve para especificar o local que será testado o user na página, a senha e o elemento que irá mudar ao inserir a senha correta, eu entendi que seria a melhor forma de fazer.

Baixei uma wordlist que achei no GitHub, e dei "cd" na pasta em que ela estava. Depois, especifiquei no comando do próximo slide.

2 – Ataque com vários usuários e senhas

O comando a seguir mostra como executar um ataque com várias combinações de usuários e senhas em um aplicativo web:

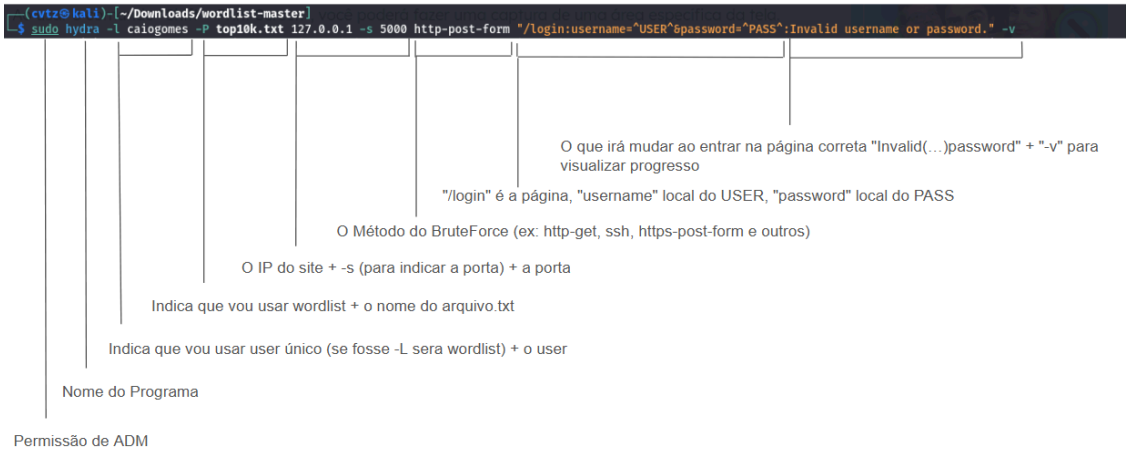
```
shell
hydra -L <caminho para o arquivo de usuários> -P <caminho para o arquivo de senhas> <endereço IP do alvo> <protocolo>
```

Por exemplo, para testar senhas para uma lista de usuários em um site com o endereço IP "192.168.1.100" usando o protocolo HTTP, você pode usar o seguinte comando:

```
"shell
hydra -L /usr/share/wordlists/usernames.txt -P /usr/share/wordlists/rockyou.txt 192.168.1.100 http-post-form "/login.php:user=^USER^&password=^PASS^:Invalid Password!"
```

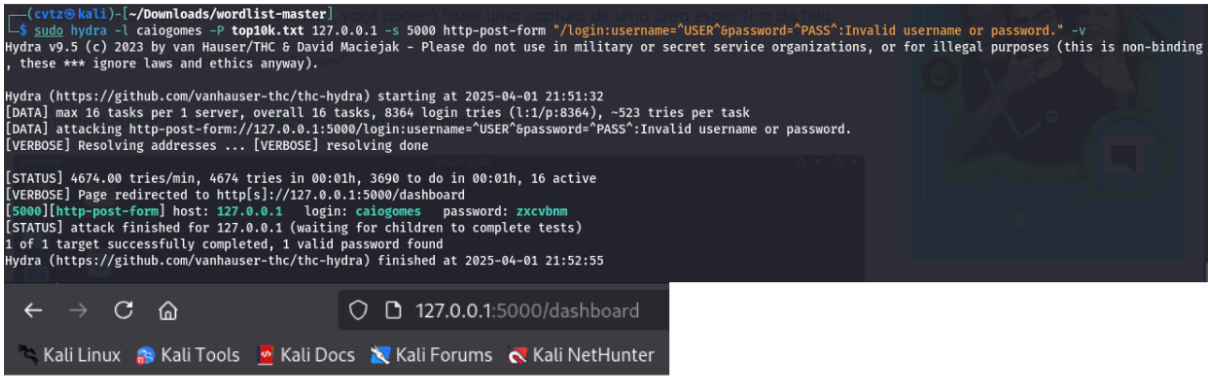
Nesse exemplo, a Hydra usará o arquivo "usernames.txt" como lista de usuários e o dicionário de senhas "rockyou.txt" para tentar fazer login no site. O protocolo HTTP é especificado, e a opção "-post-form" é usada para enviar os dados de login para o servidor.

Exercício 1 - BruteForce com Hydra



Exercício 1 - BruteForce com Hydra

Por fim, achei a senha "zxcvbnm", coloquei a mesma no site e funcionou.



Parabéns! Você concluiu o exercício 1 😄

Já tinha feito esse exercício usando o hydra na capacitação 0, mas basicamente, fiz um bruteforce usando o user já apresentado, na senha a wordlist top10k.txt, e a porta 5000. Aí inseri o tipo, que nesse caso é por validação a partir de mudança do html (http-post-form) e mostrei os locais que mudam quando os dados estiverem corretos, nesse caso a mensagem de usuário ou senha inválida.

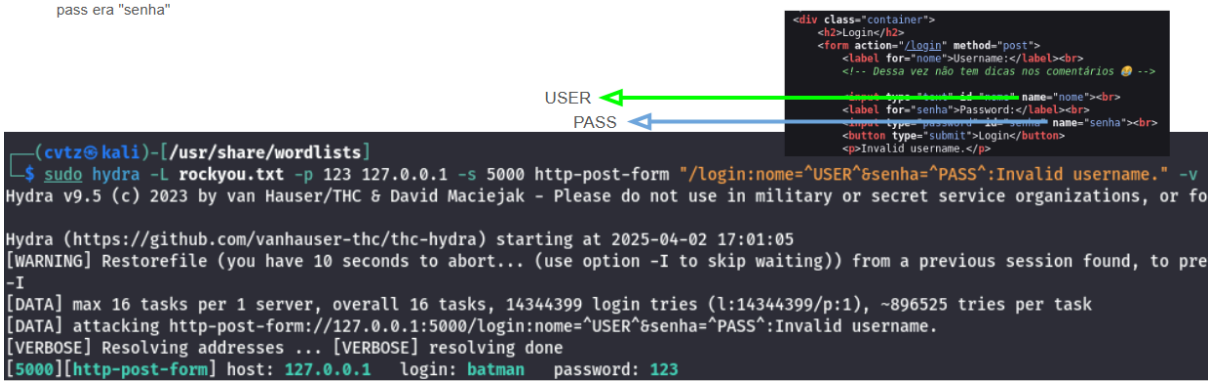
Ex 3 -

Exercício 3 - BruteForce com Hydra; Achando User antes da Password

Nesse exercício, notei que ao tentar logar sem nada, aparecia a mensagem de erro "Invalid Username.", significando que ocorre uma validação do Usuário, e posteriormente da senha, com uma mensagem diferente.

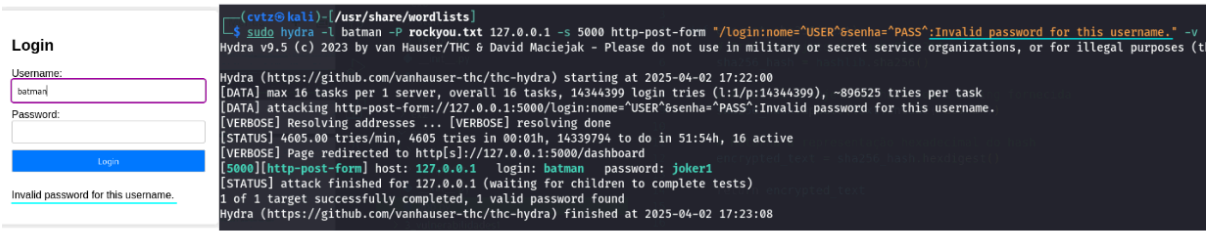
Assim, utilizando o Hydra, eu especifiquei nos requisitos (a escrita em laranja) que para o processo se dar como concluído, era necessário que a mensagem de erro "Invalid Username" não existisse após as tentativas. Então, coloquei no campo da senha, uma password propositalmente incorreta (nessa caso: 123) para testar o usuário antes.

Além de tudo, também troquei nos requisitos o campo que introduz o teste, no anterior era username=^USER^ e password=^PASS^ e nesse o user era "nome" e o pass era "senha"

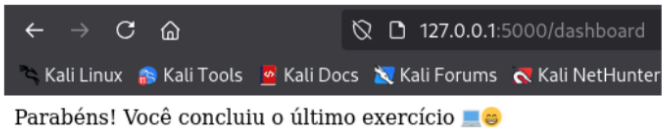


Exercício 3 - BruteForce com Hydra; Achando User antes da Password

Com o usuário em mãos, usei o Hydra para achar a senha. Igual do Ex2, porém trocando os parâmetros, igual como foi para o username, mas com a nova mensagem de erro: "Invalid password for this username." E também trocando para "-l batman" e "-P rockyou.txt" para testar as senhas da wordlist rockyou no user batman.



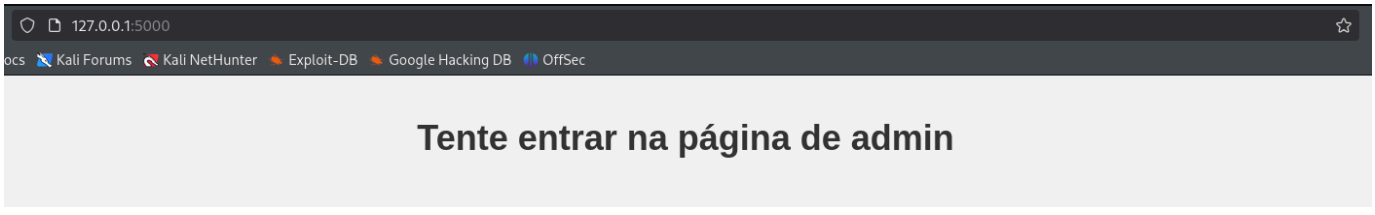
Por fim, achei a senha "joker1", traduzindo que ao colocar esse cadastro a mensagem "Invalid password for this username." desaparece. Dando assim a resposta do exercício.



Antes aparecia um erro para validação de usuário e outro para senha, ou seja, ao acertar o usuário o erro muda. Portanto fiz a validação com uma senha errada, 123, e usei wordlist para o user. Deu certo, assim com o user, fiz outro bruteforce para achar a senha com a wordlist rockyou.txt.

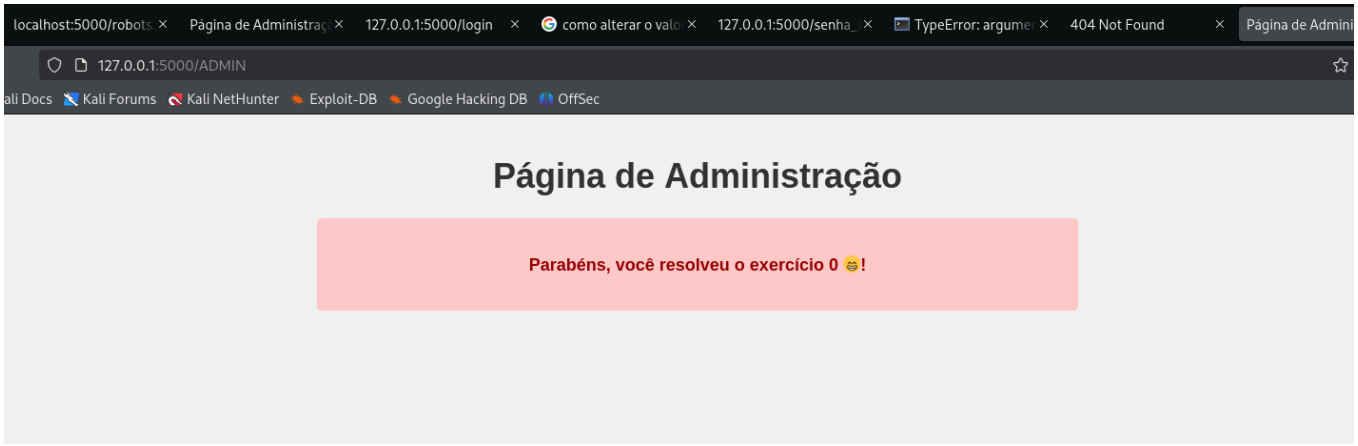
Prática 2

Ex0 -
Ao analisar o código, percebe-se que o def admin, no escopo do @app.route('/admin'), nega o acesso se o path for escrito todo em minúsculo(admin), antes de que o processo do "if path.lower() == 'admin'" seja realizado, o qual verifica independente da variação de maiusculas, chamando assim a def go_to _admin, que concluí o exercício. Portanto, deduz-se que para resolver, é necessário escrever na url variações com uppercase: Admin, aDmin, ADMIN e etc…



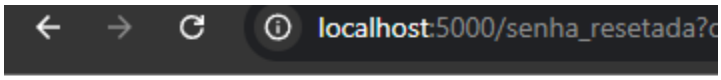

```python
@app.route('/<path:path>', strict_slashes=True)
def any_other_route(path):
    print(request.base_url)
    path = '/'.join(request.base_url.split('/')[3:]).lower() # ex: in http://google.com/images, path = images

    if path.lower() == 'admin':
        return go_to_admin()
    else:
        return 'Página não encontrada'
```



Ex1 -

Ao abrir o site do exercício, entende-se uma instrução bem clara, e ao preencher o campo de email com o "hacker@security.com", somos direcionados para uma página de parabéns.



Parabéns, você completou o exercício 1

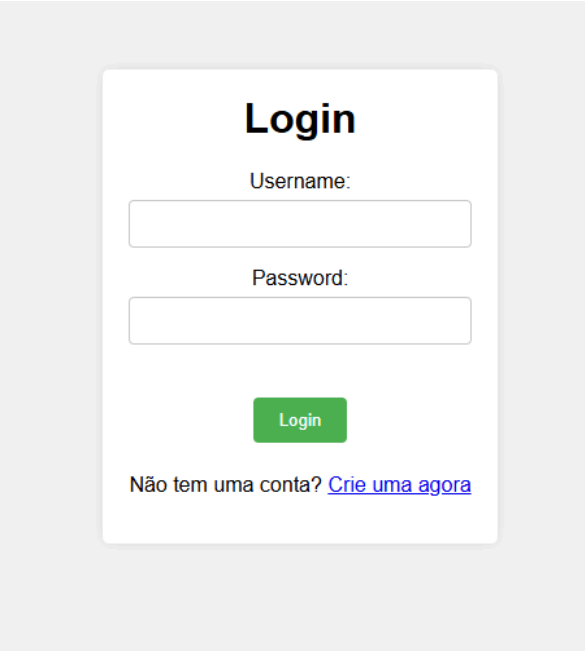
Ao analisar o código-fonte, percebe-se que realmente a única verificação para esse direcionamento e conclusão era o do campo de email.

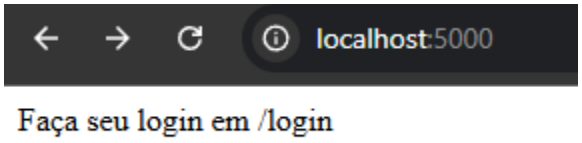
```python
@app.route('/reset_senha', methods=['POST'])
def reset_senha():
    data = request.json
    email = data.get('email')
    username = data.get('username')

    if email == "hacker@security.com":
        return "Parabéns, você completou o exercício 1 "
    return f'email: {email} & username: {username}'
```

Prática 3

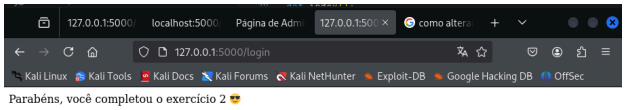
Abrindo a tela inicial e seguindo a instrução, coloca-se o “/login” na URL do site para ir até a página de login:





Faça seu login em /login

Após analisar as interfaces por um bom tempo, deduziu-se que o melhor serie olhar no código do ”app.py”. Lá o código foi analisado, def por def, e compreendeu-se a relação entre os hashs e o cookie “session”. Assim, modificando essa variável via inspecionar, com o hash_user "YWRtaW4=" encontrado no código(que é “admin” codificado em base64), viabiliza-se a conexão com a página de sucesso:



Parabéns, você completou o exercício 2 😎