当攻击者向动态表单引入恶意脚本以便获取私人会话信息时，就会发生跨站点脚本攻击(XSS)。在本文中，Anand K. Sharma 揭示了容易受 XSS 攻击的领域，解释了用户如何进行自我保护，以及站点管理员如何才能保护站点免受这类恶意入侵。

大多数现有的浏览器都能解释和执行来自 Web 服务器的嵌入到 Web 页面下载部分的脚本(用 JavaScript、JScript、VBScript 等脚本语言创建)。当攻击者向用户提交的动态表单引入恶意脚本时，就会产生跨站点脚本 (XSS) 攻击。XSS 攻击会导致不良后果。例如攻击者将能够获取会话信息，窃取 ID、密码、信用卡信息、家庭住址和电话、社会保险/税 ID 等用户的隐私资料。如果目标 Web 站点不检查这类恶意代码，就可能造成恶意使用用户信息。

为了让脚本不被识别为恶意脚本，攻击者会用不同的编码方法对其进行编码，比如使用 HEX。通过这种方法，Web 站点就会像显示站点中的有效内容一样在页面上显示恶意内容。如果 Web 应用程序没有验证输入，攻击者只需诱导用户选择恶意链接，然后 Web 应用程序就会从用户那里收集机密信息。这就让攻击者就能够获取用户会话信息并窃取用户凭证，重定向到另一个 Web 站点上的网页，然后插入恶意代码来破坏 cookie、公开 SSL 连接、访问受限或私人站点，甚至触发一系列这样的攻击。

要阻止这类攻击的第一个应对办法是认清哪些领域容易受到 XSS 入侵。

发现易受 XSS 攻击的漏洞

下面的例子可以帮助您识别易受 XSS 攻击的领域。

网银系统的弊端

假设一个攻击者试图窃取虚拟的网银站点(www.simplebank.com)上的一个用户信息。要想登录这个站点，攻击者首先要输入一个测试用户 ID ("test") 和密码 ("test")。当错误信息页面出现，告知用户 ID 和密码*组合* 错误时，这个攻击者就找到了一个向 Web 页面中插入恶意代码的理想时机。这是如何实现的呢？

- 他首先将下面的信息输入 ID 文本框：`<script>alert('Test')</script>`。
- 然后提交表单并看到这个 JavaScript 警告信息："TO BE DONE."，现在他知道这个站点容易受 XSS 类型的攻击。
- 接下来攻击者会将类似于 清单 1 的脚本引入到 URL 中，这个 URL 将把所提交的用户信息重定向到 maliciousssite.com。

  该代码通常会将任何登录到此 Web 站点的用户的 ID 及密码信息传递到攻击者的 Web 站点。

- 窃取用户 ID 和密码的脚本完成后，攻击者就通过这个链接向网银站点的用户发送具有诱人优惠信息的电子邮件和帖子。
- 受这些诱人的优惠信息的鼓动，用户有可能会单击链接并登录到网银站点。于是浏览器会执行攻击者引入的恶意代码，数据将被传递到黑客的网站。这样，黑客就轻而易举地利用了受害者的凭证登录到网银站点。

这种情况常常发生在：

1. Web 服务器没有采取充分的步骤，确保生成适当编码的页面。
2. 没有正确验证输入。

在线论坛和消息公告栏

Web 上最常受攻击的渠道是搜索框和在线论坛。攻击者会在脚本标记间插入恶意代码，而 Web 页会加以接受和解释，根据所使用的页面，采用的可能是 FORM，也可能是 APPLET 标记。所插入的恶意代码可以通过窃取会话信息或 cookie 进行各种破坏。

由于 Web 设计人员很难同时具备应对攻击的多种语言和技术知识，所以这种漏洞很普遍。很多语言（CGI、JavaScript、ASP、Perl，甚至 HTML 标记）都很容易被这些攻击利用。

附加到消息和电子邮件的链接

攻击者可以将有关网银站点的电子邮件发给用户。假设电子邮件包含一个 URL 中嵌入了恶意代码的链接。用户可能会被提示单击此链接并登录到该站点，攻击者会借此获得用户的登录信息。

对于链接内含有恶意代码的动态生成页面，情况也是一样的。我们以作为页面的一部分的 URL（参见 清单 2）为例。

如果攻击者通过应用程序显示一组 HTML，这有可能就会带来麻烦。IMG 和 IFRAME 标记都允许在 HTML 显示时加载新的 URL。

搜索引擎

响应搜索关键字的搜索引擎也容易遭受这类攻击。因为恶意代码可作为搜索关键字的一部分。因此，用户提交搜索时，就会执行恶意的搜索关键字。这种危险包括访问站点不愿意被访问或隐私的区域。

例如，清单 3 所示的代码片断在目标计算机上执行代码。而攻击者正是以这种方式插入 HTML。

错误消息

某些 Web 站点不仅响应输入，还响应由业务验证生成的错误消息。如果输入字符串具有一些恶意设计的代码，脚本也会被执行。这可能会危及保密信息的安全，或创建可能被误认为是来自用户的请求。

设置帐号

当用户在设置电子邮件帐号期间提交表单（或在提交包含数据的 Web 表单期间），应用程序在接受输入信息后可能会显示相同的信息。输入的内容可能包含浏览器能够执行的恶意信息。这可能会导致会话重要信息的泄漏，也可能会暴露 Web 服务器的隐私通路。

蠕虫

HTML 内的 IMG 和 IFRAME 标记允许在 HTML 显示期间加载新的 URL。某些蠕虫使用由 IFRAME 标记提供的 load-on-view 特性来毒害运行浏览器的系统。

这些都是易受 XSS 攻击的漏洞。接下来，我将简单介绍 XSS 侵入的主要后果。

回页首

cookie 受窃：XSS 攻击的后果

当应用程序发出的 cookie 遭到恶意劫持时，就会发生 cookie 窃取。通过向 URL 适当地插入脚本代码以便调用使用 cookie，这也是容易受攻击的网站部分，攻击者就能获取这些 cookie 并能破坏其内容以及模仿业务功能和执行假交易。

以 清单 4 中所示的代码为例，如果在其上单击，就会将此 cookie 发送给 www.destination.com/cgi-bin/cookie.cgi 并加以显示。如果您看到页面正在显示一个

cookie，那么就可能发生用户帐号的会话攻击。该脚本也可以用 HEX
编写，减少被发现的机会。总之，此脚本将用户的 cookie
发送给攻击者的网站，在该网站攻击者可以获得进行破坏和攻击所需的全部信息。

回页首

用户保护措施

用户可以通过 5 个有效手段来减少 XSS 类型的攻击：

1. 不必要时禁用脚本功能。
2. 不要轻信电子邮件或消息栏内到其他站点的链接。它们可能包含具有破坏性的恶意代码。
3. 除非特别信任，否则不要单击站点上那些到涉及个人和商业信息的安全敏感页面的链接。
4. 通过地址直接访问涉及敏感信息的站点，而不是通过第三方站点。
5. 获得各种攻击及其发生的站点和公告栏的列表，访问时要格外小心。

回页首

Web 开发人员的最佳实践

Web
站点的开发人员和维护人员又应该如何做呢？本节重点介绍了可以减少此类攻击发生的各种方法，并在文章的末尾给出了针对 Web 开发人员的检查列表。
首先，验证输入。下面的脚本接受一个参数并在显示时反映相同的参数。

清单 5. 接受和响应参数的易受攻击的脚本

```
#!/usr/bin/perl
        use CGI;

        my $var1 = CGI->new();
        my $parameter = $cgi->param('text');

        print $var1->header();
        print "parameter";
```

这段代码很容易受 XSS 攻击，因为没有进行任何的输入验证。解决的方法是验证输入和 HTML，并在显示相同参数之前转义该数据。对数据进行 *HTML 转义* 意味着非字母和数字的字符在内部以不同方式表示；比如，用小于 (<) 和大于 (>) 符号表示字符串 (< 和 >)。
以下面的方式添加输入验证：

```
$parameter = ~ s/[^A-Za-z0-9 ]*/ /g;
```

该验证只允许字母、数字和空格符，并会过滤剩余内容。通过以下方式进行加强验证：

```
HTML::Entities::encode($parameter)
```

这段代码实际上是将 HTML 符号编码为 HTML 实体引用。像小于符号 < 这样的字符被编码为 "<" 以协助过滤出这类攻击。不过，这并不能彻底解决问题 —— 攻击还会以其他方式出现。
还可以通过在输出前插入以下代码来向脚本添加输入验证。该代码消除了除字母、数字和空格之外的任何其他输入内容。

```
HTML::Entities::encode($text);

        $text =~ s/[^A-Za-z0-9 ]*/ /g;
```

该脚本容易受跨站点脚本攻击，因为它盲目输出所提交的表单数据。要消除这种漏洞，可以执行输入验证，或者确保用户所提交的数据总是在显示之前已被 HTML 转义。
根据这个原理，`TaintRequest` 会自动对数据进行 HTML 转义。它总是在显示或打印数据之前进行内容验证和 HTML 转义。(Perl 具备内置的像 `Taint` 模式这样的特性，可以用于安全性检查)。这种方法就确保了所有流入程序的外部数据都不会直接用于处理文件和目录或直接用于执行过程。`Apache::TaintRequest` 是一种功能强大的检查，可阻止这类攻击并大大减少了应用程序遭受的 XSS 攻击。
要激活此特性，向 httpd.conf 文件插入以下内容。

```
PerlTaintCheck on
```

通过以下方法确保脚本使用 `Apache::TaintRequest`：

```
use Apache::TaintRequest;
my $var1 = Apache::TaintRequest->new (Apache->request);
my $parameter = $var1->param('parameter');
$r->content_type("text/html");
$r->send_http_header;
$parameter =~ s/[^A-Za-z0-9 ]//;
$r->print($parameter);
```

这段代码接受来自用户的受感染数据，检查要打印的字符只有字母和数字字符以及空格符，以此来提供安全保护。
下面是站点管理员和开发人员可用来阻止 XSS 攻击的检查列表：

- 保证 Web 站点的页面只有在验证了用户输入没有恶意代码之后才返回用户输入。
- 在验证的同时过滤输入中的 Meta 字符。(这是消除 XSS 攻击的一个重要检查点。虽然它不能消除全部的 XSS 问题，但它能警告 Web 维护人员在站点安全性方面存在的不足)。
- 不要完全相信使用了 HTTPS(Secure Sockets Layer)的 Web 站点就不会受到 XSS 攻击;HTTPS 只能确保安全连接，处理用户输入的数据是应用程序内部的事情。如果应用程序有 XSS 漏洞，攻击者就可能会发送能被应用程序执行的恶意脚本，导致 XSS 侵入。
- 在搜索引擎和论坛内显示用户输入之前，将所有非字母和数字字符转变成 HTML 字符。
- 在应用程序正式付诸使用之前，在设计阶段大量使用测试工具，消除这类 XSS 漏洞。(强调这点的最佳实践指南具有 Extreme Programming 提到的思想)。
- 开发一些具有私有和公共密钥的标准或标志脚本，这些脚本要确实进行检查，以保证所引入的脚本真正经过了身份验证。(要大规模地实现，Internet 规则就必须进行标准化，以得出一套由 W3C 等主要参与者参与的公认方法)。

回页首

结束语

跨站点脚本攻击带来了巨大的风险。这类攻击导致很多的问题，从单个用户的身份窃取到涉及数百万消费者和经营者的重要金融和安全问题。诸如*输入验证* 和 *HTML 转义* 这样的补救措施只是个开始，但必须将它们应用到接受数据的应用程序点。哪怕只有一个表单数据字段被忽略，应用程序就跟从未进行过任何安全检查一样。

本文并未涵盖应对 XSS 攻击的全部解决方案 —— 只讨论了用户和 Web 开发人员能够采取的一些手段。阻止这类攻击的另一个必不可少的组成部分应该是全球和行业级别的 —— 对国际标准的更多研究和协调。