**XSS攻击检测方法综述**

**摘要**

跨站点脚本攻击（简称XSS）是自2000年代初以来Web应用程序一直存在的一个问题，它是一种客户端的代码注入攻击，攻击者将恶意负载注入到易受攻击的Web应用程序中。攻击者通常在不了解用户的情况下成功地在无辜用户的浏览器中执行恶意代码。通过XSS攻击，攻击者可以执行恶意活动，如cookie窃取、会话劫持、重定向到其他恶意站点、下载不需要的软件和传播恶意软件。XSS攻击的主要类别是：非持久性和持久性XSS攻击。本次调查的重点是研究文献中关于XSS攻击的全面检测方法。本研究所探讨的侦测方法是根据其部署地点进行分类，并根据其使用的分析机制进行进一步分类。除了讨论每种方法的优缺点之外，本次调查还列出了支持检测XSS攻击的工具列表。我们还详细讨论了成功发起XSS攻击必须满足的三个先决条件。本次调查的主要目标之一是确定问题列表和开放性研究挑战。任何希望了解、评估、建立或设计对抗XSS攻击的检测机制的人都可以将此调查作为基础阅读手册。

**1.导论**

网络是我们个人日常生活以及整个社会活动的重要组成部分。随着技术的进步，即使是最复杂的应用程序也越来越多地通过网络交付。然而，随着所提供服务的激增，出现了一些关键问题。网络有多安全？当我们访问网络上的资源时，我们有多安全？这些问题的答案有一个单一的重点——网络上所有交互级别的安全性。随着云计算技术的巨大进步，数据安全是维护用户隐私、保密性、完整性和权威性的主要方面之一。诸如选择性开放安全性和延展性等问题与云计算安全性密切相关。此外，由于重复数据删除等问题，当用户享有访问存储在此类平台中的数据的不同权限时，维护安全变得非常困难。为了找到解决这些问题的方法，作者在中提出了一种混合云架构。为了寻找云计算中与安全相关的问题与研究，作者在中讨论并强调了解决这些问题的对策。与云计算安全相关的一些重要研究文章将在[6][7][8][9]中讨论。

安全性在所有Web应用程序中都扮演着重要的角色。与各自的Web服务器耦合的应用程序为用户提供广泛的有用服务。然而，存在一个特定的亚群体不断用Web应用程序和服务器中潜在漏洞的高级用户是许多用户个人交流和信息的中心。通常，第一步发起破坏性攻击的目的是发现应用程序中的安全缺陷，然后利用这些缺陷获取敏感信息。

在过去的几十年中，一种特殊类型的应用程序层攻击（称为跨站点脚本攻击（XSS））已经变得可怕。传统上，这些攻击是用来窃取个人信息的，导致可能的受害者冒充。然而，最近随着技术的发展，这些攻击正与社会工程技术一起使用，以创建和发起其他惩罚性攻击。跨站点脚本攻击最好描述为客户端的应用程序层代码注入攻击，攻击者将恶意脚本注入易受攻击的Web应用程序，为在无辜用户的浏览器中成功执行恶意代码[10][11]。一个Web应用程序被称为如果没有正确的输入验证，并且没有正确的检查处理，则存在XSS漏洞。日常工作。早在20世纪90年代就发现了XSS漏洞，但这些漏洞仅为专家所知。在21世纪初，这种攻击的严重性在过去几年中有了很大的提高。XSS攻击可以现在可以用来操作Web内容、传播恶意软件、发起ddos攻击和劫持用户会话。

除了将个人敏感信息传输到攻击者的服务器之外在XSS攻击中，恶意内容（恶意脚本）被引入到易受攻击的Web应用程序的可信上下文中。在执行Web应用程序时，受害者会收到恶意内容，它伪装成合法应用程序代码的一部分。由于受害者的浏览器无法区分恶意脚本和合法内容。图1显示了成功启动XSS攻击所涉及的步骤。攻击者不会直接针对受害者，而是使用易受攻击的Web应用程序中的缺陷作为工具来交付受害者浏览器的恶意代码。XSS攻击中有三个参与者，如图2所示。根据恶意代码注入Web应用程序的方式，XSS攻击被分类。如图3所示。实际上，恶意脚本的方式交付给受害者浏览器的方式和执行方式在这三个类别中有所不同，以及由此而来的分类。反射式XSS是最常见的XSS攻击，尽管可能更多存储的XSS攻击很危险。反射和存储的XSS攻击不同于基于DOM的XSS攻击，因为后一种类型是由于浏览器脚本解释器的缺陷而产生的。相反，反射和

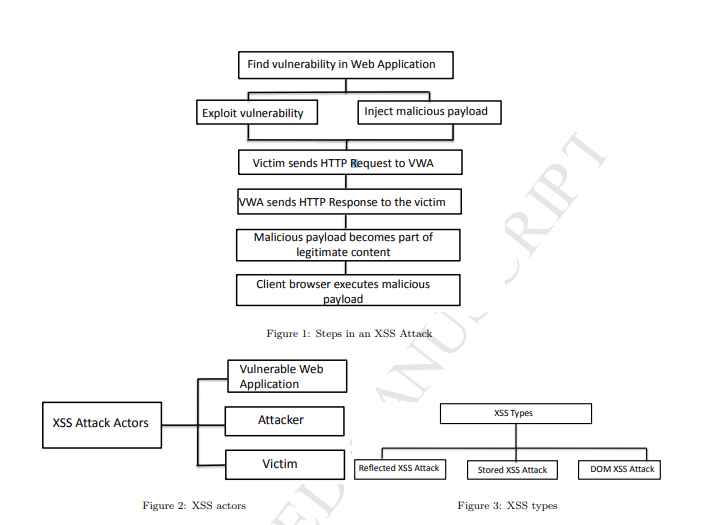
存储的XSS攻击是Web应用程序中存在漏洞的结果。基于DOM的XSS攻击是

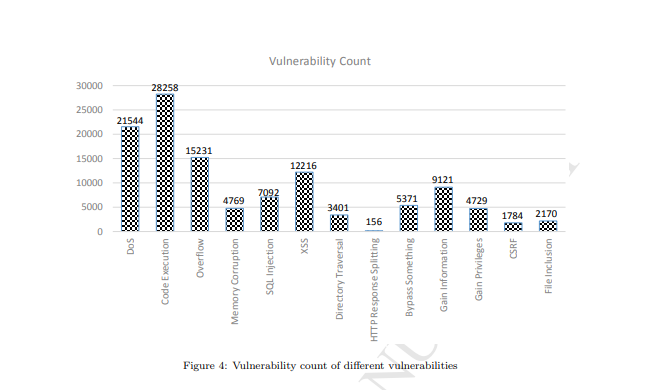
相对不常见。

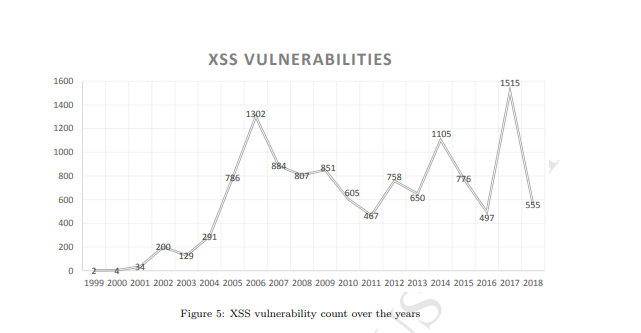
随着现代互联网时代向社会提供越来越多的服务，敏感信息的安全和维护风险越来越高。潜伏着许多弱点

Web应用程序，通常没有开发人员的知识。恶意犯罪者利用这些Web应用程序中存在漏洞，对应用程序的最终用户构成严重威胁。因此，易受攻击的应用程序的用户成为攻击者的牺牲品。图4描绘了1999年至2018年4月9日不同类型漏洞的漏洞计数。可以看到，代码执行漏洞的计数最高。跨站点脚本具有指定时间内的总漏洞计数为12216，约为12%。报告的漏洞总数。图5显示了XSS漏洞的数量在这些年中是如何增加的。图6，由White Hat Security 2报告给出不同

行业。在识别来自不同行业的应用程序中的漏洞时，使用动态应用程序安全测试3或DAST技术。在脆弱性方面，工业遭受的损失最大，其次是运输业。







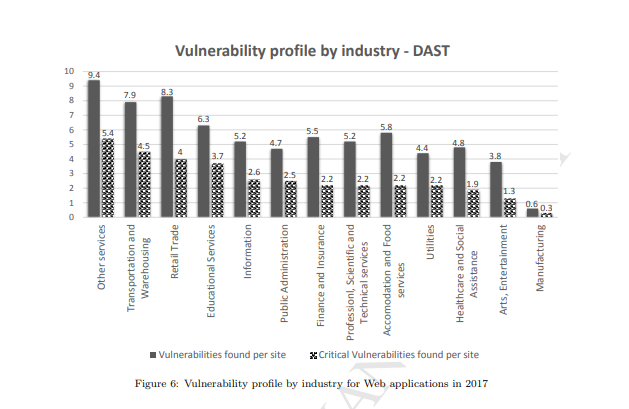


图7描述了截至2017年的Web应用程序漏洞类别的可能性。（12）该报告总共使用了15000个Web应用程序和65600多个移动应用程序。考虑到。图8更深入地了解了漏洞暴露的窗口不同行业。曝光窗口的技术定义是计数在给定的时间框架。从图中可以看出，五个行业，即公用事业、住宿，60%左右的时间里，零售业、教育业和制造业始终处于弱势地位。这个不同类别的曝光窗口如下。

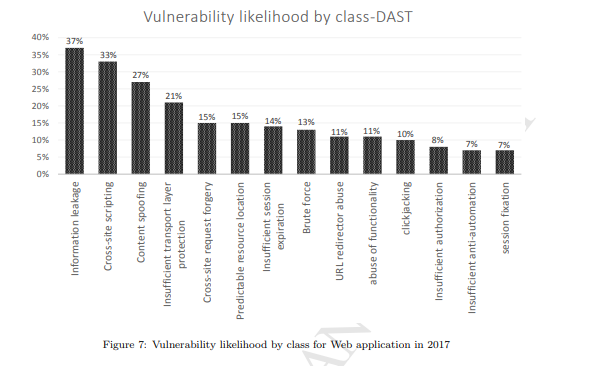
1.始终易受攻击：如果应用程序一年365天都面临漏洞，它将下降在这个类别下。

2.经常易受攻击：如果应用程序每年有271-364天易受攻击，则属于此类别。

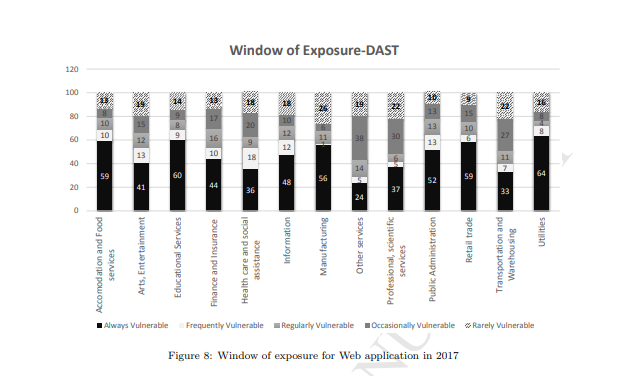
3.经常易受攻击：如果应用程序每年有151-270天易受攻击，则属于此类别。

4.偶尔易受攻击：如果应用程序每年31-150天都会受到攻击，它属于这一类。

5.很少易受攻击：如果应用程序每年的易受攻击时间少于30天，则属于此类别。



值得注意的是，在2017年，卡巴斯基实验室共确认了199个百万个唯一恶意URL和1500万个恶意对象（脚本、可执行文件等）。



1.1**与先前调查的比较**

XSS攻击造成巨大的安全威胁。在过去逃避这种攻击。XSS攻击检测与预防的系统文献综述方法见[13]。作者详细讨论了XSS攻击检测自开始以来所做的大量研究，以及针对此类攻击问题提出的解决方案/技术。他们重点关注XSS攻击检测和预防领域，以及XSS漏洞检测领域。在[14]中，作者除了介绍检测XSS漏洞的解决方案外，还讨论了XSS攻击对Web应用程序正常运行造成的几个关键安全风险。他们也报告一些与成功检测XSS攻击相关的研究空白一致的开放性问题。李等。[15]提供一篇关于服务器端Web应用程序安全性的调查文章。作者

特别讨论Web应用程序攻击者利用三个非常常见的漏洞发起攻击。这些漏洞包括：i）输入验证漏洞；i i）会话管理漏洞以及iii）应用程序逻辑漏洞。

在[16]中，Prokhorenko等人进行广泛的调查，报告各种保护方法可用于保护Web应用程序免受多种威胁。类似的调查是在[17]作者讨论了保护Web应用程序不受注入和逻辑漏洞影响的方法。

本文对各种XSS攻击检测方法进行了深入的研究。多年来提出的。检测方法分为三类，即：客户端检测方法、服务器端检测方法和客户端服务器检测方法。这些类别中讨论的方法再次被细分。根据他们使用的分析机制，成功地诊断了攻击。我们也强调每种方法的优缺点。这项调查还列出了一系列可用于防御XSS攻击的工具。与问题有关的问题和挑战同时指出了XSS攻击的存在和研究的不足。表1提供了一个比较在我们的调查和文献中提供的调查中。

•与[13]一样，我们也将防御方法分为不同的分析机制。此外，我们还将分析机制分组到部署站点下国防系统。此外，我们还详细讨论了发起成功的XSS攻击。我们还提供了一系列问题和挑战。

•和[14]一样，我们也将不同的防御方法分类到部署地点。然后进一步分组进入分析机制。此外，我们还讨论了对付XSS攻击的几种工具。执行XSS所需的前提条件攻击以及一些研究问题和挑战由我们提供。

•如[18]所述，我们还报告了一些现有技术中存在的弱点，解决XSS问题。与[18]不同，我们将防御方法分为部署站点和进一步将它们分为分析机制。

•如[19]所述，我们详细讨论了防御的检测和预防机制。XSS攻击。但是，我们首先将这些机制分类到它们的部署站点，然后然后根据分析机制对其进行进一步的分类。我们还提供成功执行XSS攻击所需的各种技术指标以及一套问题和研究挑战。

•在[20]中，作者根据软件安全方法，不像我们。作者对分析机制。另一方面，我们的调查根据检测机制的部署位置。此外，我们还广泛研究分析每个类别下的机制。

**1.2.动机**

跨站点脚本攻击或XSS攻击，自其开始被指定为一种攻击Web应用程序中存在的最大漏洞之一。一旦成功，受害者将受到攻击者的控制，攻击者可以模仿受害者或窃取其敏感信息。信息。情况恶化，因为这种攻击很难被发现，但很容易被发现实施。这就是为什么XSS攻击是一个活问题的主要原因，尽管很多多年来，人们提出了检测方法。XSS持续存在的问题多年来的攻击促使我们作为研究人员准备了一份国防目录。提出的解决方案、相关问题和挑战，甚至与解决问题。我们也借此机会提出了一种新的检测分类法。XSS攻击。XSS检测机制根据其部署进行分类。站点，因为每个检测机制都部署在客户端或服务器端，或者两个客户端服务器。此外，检测机制在方式上也各不相同。他们分析XSS攻击行为。因此，检测机制根据其使用的分析机制（即静态、动态或混合）被进一步细分。我们的主要目标是为研究界提供一本关于XSS攻击，因此可以提出新的解决方案来结束XSS问题攻击。

**1.3贡献**

我们进行这项调查的目的是对防御机制进行编目和比较。抵御XSS攻击。防御机制可根据以下方式分类：他们分析应用程序的代码。我们的调查主要集中在专注于i）漏洞检测i i）攻击的会议记录和期刊检测，以及iii）攻击预防。这项调查可作为一项基础性的阅读希望了解、评估、建立或设计检测机制以应对

XSS攻击。

本文所考虑的调查包含了文献中最著名和引用最多的方法。本次调查的主要贡献如下：

1. 对各种XSS攻击类型及其启动机制的系统调查。此外，我们还说明了发生XSS攻击的不同场景。
2. 详细讨论了XSS检测方法的优缺点。每种方法都在技术、性能、用于评估的数据集（如果使用）和考虑的参数方面进行了详细讨论，用于检测。

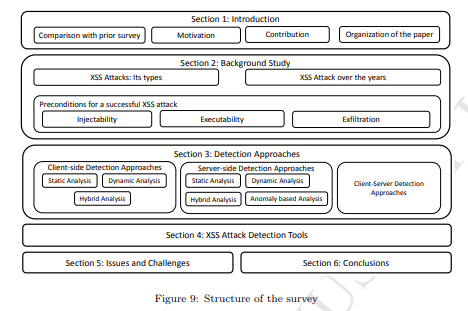
（c）我们根据国防部署地点对探测方法进行系统分类。部署既可以在客户端，也可以在服务器端，也可以同时在客户端和服务器端。

（d）我们还讨论了成功发起XSS攻击所需的各种先决条件。

（e）最后，我们从实施的角度讨论了一系列研究问题和挑战。

**1.4. 论文的组织**

论文分为以下几节。第2节详细讨论了XSS攻击、类型、攻击场景、一些现实世界中的XSS攻击以及成功启动XSS攻击所需的前提条件。第三节介绍了检测方法，分为客户端检测方法、服务器端检测方法和基于客户端-服务器端的检测方法。第4节讨论了可用于XSS攻击检测的工具。在第5节中，我们列举了与XSS攻击相关的各种问题和挑战。最后，我们将在第6节中得出结论。图9给出了所述勘测的结构。



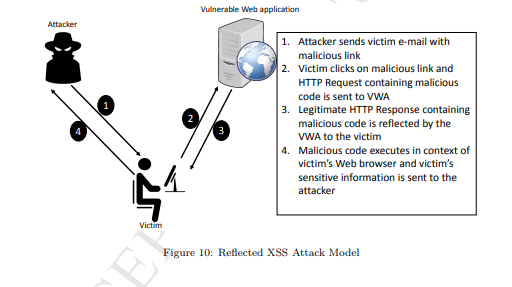
**2.背景**

**2.1. XSS 攻击:关于其类型**

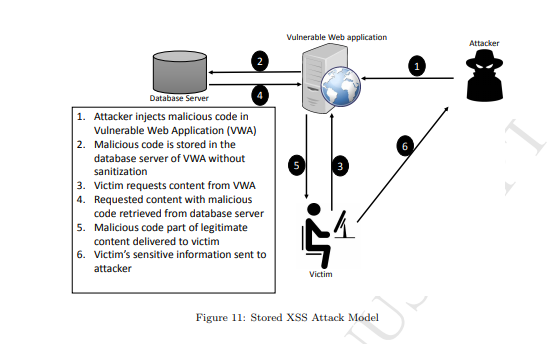
跨站点脚本攻击是应用层的代码注入攻击，它也被缩写为作为XSS攻击。首先，攻击者需要在Web应用程序中找到漏洞并利用它通过注入恶意代码，从而针对应用程序的最终用户。换句话说，恶意内容被注入到受信任的上下文中。根据用户（受害者）的合法请求，从受影响的应用程序使用恶意注入的代码提供给他/她。上的Web浏览器受害者在不知情的情况下将恶意代码作为Web应用程序的合法部分执行。作为一个结果任何事情都可能发生，包括他/她的所有敏感信息最终都会出现在攻击者的服务器上。必须注意的是，只有当Web应用程序成功执行XSS攻击时，根本不验证其输入，或者不正确验证其输入。特别是由Web应用程序使用原始无效输入。

XSS攻击可以利用ActiveX、HTML、Flash或JavaScript。从这批货中，最多的被广泛利用的是JavaScript。据统计，全球93.6%的网站使用率JavaScript 5恶意的javascript代码段利用相同的源策略。网站上的任何内容如果源站点被授予访问权限，则可能具有访问系统资源的权限。客户端的浏览器成为攻击者恶意意图的牺牲品，因为它无法区分从同一网站传递的合法和恶意内容。XSS攻击可以在不同的方式取决于类型。下面将简要讨论不同的类型。

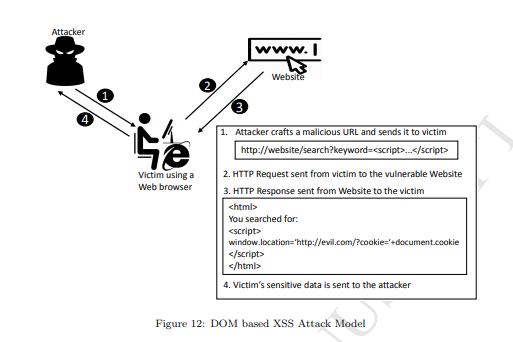
(a)反射XSS攻击：也称为非持久性XSS攻击或I型XSS攻击。这个攻击者巧妙地设计了一个包含脚本的恶意链接，并引诱受害者点击它。受害者不可避免地对服务器的请求也包含恶意字符串。反应从服务器合并恶意代码片段（即，反射回）和受害者的浏览器冷漠地执行它。因此，如果用户输入直接是输出的一部分，就存在XSS漏洞。由应用程序生成，不进行任何清理。反射的XSS攻击模型如图10所示。



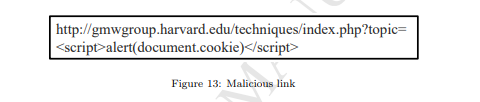
（b）存储XSS攻击：通常称为持久攻击或II型XSS攻击。攻击者制造永久存储在易受攻击服务器上的恶意代码段。恶意代码可能通过消息论坛或博客帖子注入。然后它存储在应用程序的服务器上。最终，当受害者导航到被破坏的站点时，他/她会受到恶意攻击代码段作为网页的一部分。最后，受害者的浏览器最终执行了恶意代码。因此，缺乏对用户输入和消毒程序的正确验证会导致服务器应用程序中的XSS漏洞。事实上，访问此网站的所有用户都是现在冒着在浏览器中不知情地执行恶意代码的风险。适当的程序因此，在将数据存储到数据库之前，应该在适当的位置。存储的XSS攻击模型如图11所示。

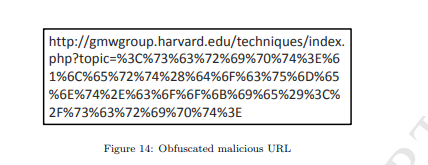


（c）基于DOM的XSS攻击：也称为0型XSS攻击。这类攻击与上面的攻击有大的不同，主要是由于存在一些客户端浏览器的脚本解释器中存在漏洞，而其他两种类型的攻击是由于到服务器端的漏洞。修改了客户端浏览器上网页的DOM结构。这就是攻击者成功执行恶意脚本的原因。因此，值得注意的是，在响应中找不到攻击者精心设计的恶意负载，就像在其他两种类型的攻击。它可以通过仔细检查DOM或当网页已加载，即在运行时加载。基于DOM的XSS攻击模型如图12所示。



为了更好地理解XSS攻击，我们演示了一个场景。攻击者向受害者，包含注入恶意脚本的链接。通过社会工程，受害者被引诱点击此链接。将生成来自客户端的HTTP请求，用于承载易受攻击的Web应用程序。HTTP请求中嵌入了恶意脚本。万维网应用程序的服务器为客户机生成一个HTTP响应。HTTP响应包括从服务器反映恶意脚本。恶意脚本现在是来自的合法内容的一部分服务器。受害者的浏览器无疑会执行它。最终，攻击者掌握了很多受害者的重要信息。攻击者现在可以很容易地模仿受害者并访问代表他的服务器。图13显示了攻击者发送给受害者的恶意链接示例。从URL检索到参数主题后，将其放置在网页中的指定位置。因此，<script>元素成为HTML代码的一部分。然后由网络处理以与HTML文档中任何其他元素相同的方式浏览。受害者不知道他/她正在执行一个脚本，这是可能的，因为存在XSS漏洞在Web应用程序中。有几种可能的方法可以让攻击者以不可读或不够可疑的方式创建URL。为了使URL不可读，攻击者可以使用URL模糊技术，这些技术很容易作为工具提供。URL的模糊版本如图14所示。第二种方法可能是缩短URL，使其看起来毫无疑问，如图图15所示。

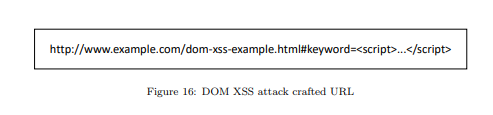






模糊导致人的可读性低。此类技术也可用于合法目的。但是，混淆恶意脚本的意图和混淆合法内容的意图是完全不同。例如，一些成熟的软件开发人员可能不希望其他人理解它们的代码，因此使用各种模糊技术。相反，恶意作者的恶意脚本使用模糊来规避入侵检测系统，特别是基于签名和静态分析以及基于系统的入侵检测系统。

在三种类型的XSS攻击中，基于DOM的XSS攻击需要特别关注。这主要是因为它的性质。如前所述，与反射和存储的XSS不同攻击，基于DOM的XSS由于客户端的漏洞而出现。当执行基于DOM的XSS攻击，攻击者操纵DOM中的对象，不适当地处理HTML第6页的属性。这种攻击对于检测至关重要，因为HTML源代码和攻击响应精确到这意味着，攻击者有效负载不是响应的一部分，而是HTML页面的DOM的一部分。像这样的物体document.url，document.location，document.referer，window.location，location.href可以是被攻击者利用。很重要的一点是，基于DOM的XSS不能服务器端检测到有效防御。如果使用“”字符，则为真，因为当遇到字符时，浏览器会将其标识为片段，而不会转发它。进一步。其中一个例子如图16所示。因此，大多数时候注入的恶意代码根本无法到达服务器。因此，防御或消毒服务器端的例程在检测基于DOM的XSS时不起任何作用。代码审查对于漏洞检测，必须明确消毒程序或预防技术在客户端执行以检测此类攻击。



**2.2多年来的XSS攻击**

可能围绕XSS漏洞潜伏的最大危险是它可能在应用程序的用户之间传播，直到其中一个漏洞修补或整个系统感染。这就是它被称为XSS蠕虫的原因[21]。XSS蠕虫有两种类型。服务器端XSS蠕虫和客户端XSS蠕虫。这个很少发生的服务器端XSS蠕虫将恶意负载存储在Web应用程序中本身。但是客户端XSS蠕虫将有效负载存储在外部文件或参数值中。客户端XSS蠕虫可以通过简单地查看受感染的网页来激活。这个服务器端XSS蠕虫需要用户执行更多操作，例如安装假恶意程序。XSS蠕虫的最大武器是它在客户端工作，因此它利用用户的信任圈8XSS蠕虫的传播速度比基于操作系统的传统蠕虫。类似于自我传播恶意软件的功能，XSS蠕虫还有两项重要任务[22]。

1. 传播：此任务很重要，因为一旦启动蠕虫，它就必须传播把自己带到新的未受影响的地区，以接触和破坏大量的观众。Samy worm是第一个自我传播的javascript蠕虫。
2. 执行有效负载：传播后，蠕虫不会闲置，不会导致任何损坏。蠕虫可以自己执行一些任务，比如窃取用户的cookie，删除用户计算机中的文件，或从用户帐户发布任意消息在OSNs。

javascript蠕虫的最大优点是它不是用编译语言编写的，它与底层处理器体系结构或操作系统有几个依赖关系。这使得用javascript编写的XSS蠕虫可以在任何机器上运行，而不考虑不同的操作系统和芯片组合。这是浏览器的javascript解释器出现的时候发挥作用。javascript解释器将蠕虫代码解释为本地指令。这个使javascript成为真正的跨平台解释编程语言。这就是为什么XSS蠕虫控制应该是自动的原因。17中的数字描述了这一点事实。蠕虫的行为因平台和传播矢量而异。不同类型的蠕虫可以以不同的方式传播，无论是通过附件还是通过恶意链接或利用邮件服务中的安全漏洞，就像Yamanner XSS蠕虫。它联系了电子邮件地址以@yahoo.com结尾的用户。或@yahoogroups.com。下面将讨论一些现实世界中的XSS。

1. 萨米蠕虫：它是一种自我复制的XSS蠕虫，感染的时间不到一天。超过100万myspace用户。“但最重要的是，萨米是我的英雄”的信息是显示在受害者的档案上。它把被感染的用户添加到创造者Samy Kamkar的MySpace帐户。最终，它导致了社交网站的关闭。

Source: <https://betanews.com/2005/10/13/cross-site-scripting-worm-hits-myspace/>

1. Yamanner蠕虫：它首先向雅虎用户发送邮件！邮件。当用户打开时邮件，它传播到用户的联系人。事实上，它利用了雅虎！邮件，允许在用户浏览器中执行HTML文档中的脚本，当它实际上应该被阻塞的时候。受感染邮件的主题出现了

Source: <https://www.theregister.co.uk/2006/06/12/javscript_worm_targets_yahoo/>

1. 作为“新图形站点”。发件人的电子邮件地址似乎是一个带有名称“av3@yahoo.com”。用户被重定向到网站www.av3.net/index.html。受影响的电子邮件地址最终存储在远程服务器上。它是不同的从蠕虫的传播过程开始于服务器端而不是客户端。

Source:<https://xavsec.blogspot.in/2005/12/new-myspace-xss-worcirculating.html>

1. QuickTime XSS蠕虫：该蠕虫利用的浏览器插件中的漏洞QuickTime，一个由苹果公司开发的多媒体框架。嵌入到视频中，点击视频链接后，蠕虫病毒感染了MySpace用户。上的不可靠用户访问受感染的配置文件也会受到感染。被感染的个人资料的唯一方法与未受影响的配置文件的区别在于存在一个空的QuickTime MySpace标题部分中的视频或修改的链接。

Source:https://www.bleepingcomputer.com/forums/t/74242/myspace-xss-quicktime-worm/

1. orkut-xss蠕虫：orkut蠕虫是经典XSS攻击的一个例子。它会利用Orkut的URL解析机制中的安全缺陷。允许此安全缺陷在orkut环境中加载和执行任意代码的攻击者。在首先，攻击者创建了一个假帐户，并精心编制了一个Ajax代码，可以找到用户的朋友和嵌入到他们的剪贴簿恶意代码。收到后当用户访问剪贴簿阅读报废蠕虫会自己执行。在内部，代码只是一个javascript名为virus.js并从远程站点9访问的文件用户感染后，该用户自动成为一个伪造社区的一部分，攻击者使用该社区追踪他的目标。多达400000名奥克特用户被感染。

Source: <https://nakedsecurity.sophos.com/2007/12/19/large-scale-orkut-virus-outbreaknot-cool/>

1. hi5-xss蠕虫：缺乏正确的输入验证导致存在xss漏洞在应用程序中。我们可以很容易地在评论或在邮件中。在应用程序也可以使用双编码技术来绕过XSS过滤器。

Source: <https://packetstormsecurity.com/files/101038/hi5-xssxsrf.txt>

1. Justin.tv蠕虫：蠕虫的影响持续了24小时，大约2525个JTV用户的账户被泄露。个人交流用户在Flickr上公开。

Source:https://www.zdnet.com/article/xss-worm-at-justin-tv-infects-2525-profiles/

1. Renren虫：人人虫利用了中国社交网络人人虫的XSS缺陷

网站。它会伪装成粉色弗洛伊德的闪光音乐录影带，然后传播开来。

通过人人的原料药感染。它的制作方式是参数设置为Always，表示嵌入HTML页面的SWF文件，可以访问页面的DOM，而不管它在何处承载。因此，SWF文件仅用作加载和执行evil.js脚本中的javascript代码的工具，该脚本宿主为在外部域中。后来，这种蠕虫被命名为W32/Pinkren-A。值得注意的是，传播蠕虫的技术与Orkut-XSS蠕虫。

Source：<https://blogs.securiteam.com/index.php/archives/166>

1. 这是一个偷饼干的脚本。感染导致用户收到废品上面写着“bom sabado”，意思是葡萄牙语的“星期六快乐”。也，收到此脚本的用户是假冒社区的一部分。

Source: <https://upalc.com/bom-sabado.php>

1. java蠕虫: 蠕虫实际上是一个特洛伊木马恶意软件，特洛伊木马.osx.boonana.a。它使用Facebook信息作为传播媒介，伪装成视频链接主题为“这是你在这段视频中吗？”。单击恶意链接后，它会将用户定向到外部网站，在该网站上提示用户授予执行权限。一个叫JPhotoAlbum.class的Java小程序。一旦授予运行小程序的权限，它已将多个文件下载到计算机系统，包括自动启动安装程序。安装程序有恶意代码以这样的方式操作系统文件所有的安全密码都被绕过，允许外部人员访问用户的系统。此外，特洛伊木马每次启动系统时都会自动启动，而在后台执行会联系命令和控制服务器提交有关主机系统的信息。当主动执行时，特洛伊木马攻击用户并通过电子邮件和社交媒体帐户进行复制。

Source:https://www.securemac.com/osx/boonana-trojan-horse-trojan-osx-boonana-a

1. 彩虹XSS蠕虫：这个蠕虫特别利用了JavaScript中的onMouseOver事件。被感染用户的tweet会包含一些奇怪的信息，其中包含巨大的字母，或者打招呼。接着是一条条黑线。恶意代码只会在上执行将鼠标移到这些文本上，然后重新推送自己。被感染者的追随者帐户自动接收到恶意字符串，从而感染其他非可疑的用户。访问时，受感染用户的配置文件将受害者重定向到外部网站。每10秒钟就有多达1000名用户受到感染。漏洞是首先被一个名为彩虹twtr的Twitter用户发现。用户使用该漏洞显示彩虹的颜色和名称。

Source: <https://www.theguardian.com/technology/blog/2010/sep/21/twitter-hack-explainedxss-javascript>

1. Facebook XSS蠕虫：蠕虫很好地利用了Facebook的移动API版本。未正确筛选javascript编写的代码。它授予任何网站包含恶意iframe的权限。中的代码恶意iframe将浏览器重定向到包含javascript的准备好的URL。论执行时，它成功地自动将消息发布到其他人的墙上。它需要注意的是，蠕虫只需要很少的用户交互，没有使用专门的技巧。在访问受感染的网站时，攻击者的选择将在用户墙中共享。只有noscript xss筛选器成功识别蠕虫。

Source:https://www.symantec.com/connect/blogs/new-xss-facebook-worm-allows-automaticwall-posts

1. xanga-xss蠕虫：这个特定于浏览器的蠕虫利用xmlhttp接口进行传播。xanga用户在访问已感染的人时感染了病毒。其执行结果在Xanga的网络日志上创建了带有令人讨厌的消息的受感染邮件用户。哑巴蠕虫不断地自我定位好几次，传递着同样的信息。到已感染的站点

Source: <https://blogs.securiteam.com/index.php/archives/166>

像Yamanner这样的蠕虫有些是特定于操作系统的，而有些则不是。它们中的许多扩散通过社交网站或网络邮件服务，还有一些通过游戏应用。一些蠕虫被设计成实验的一部分，以观察有多少用户会受到影响。最多或查找应用程序中的漏洞或漏洞。意图不是控制任何账户。但这显然为发动更具破坏性的攻击铺平了道路。这里一个重要的检测机制是观众[23]自动执行检测XSS蠕虫。它也适用于控制蠕虫。表2给出了这些年来真实世界的XSS蠕虫的细节。从传播的角度来看XSS蠕虫使用了xmlhttpRequest方法。此方法用于发送异步在后台请求，因此许多恶意活动未被注意到。这是该方法优于表单提交方法。表单提交方法表示由于正在单击提交表单或链接而可能导致的HTTP请求。

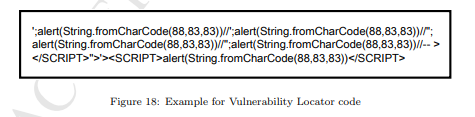
**2.3成功XSS攻击的前提条件**

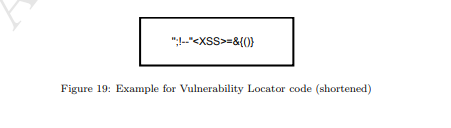
攻击者要成功发起XSS攻击，首先必须满足三个前提条件。启动XSS攻击前的三个先决条件是：可注入性、可执行性以及渗出。如果攻击者绕过这些前提条件，他/她将能够启动一个XSS攻击。以下章节将详细讨论前提条件。

**2.3.1Injectability**

使用攻击向量注入Web应用程序。攻击向量是使用的方法由攻击者成功地执行攻击，在我们的情况下是XSS攻击。已知的利用漏洞交付攻击者的恶意负载并获得对敏感信息。随着应用程序的交互性增加，攻击向量的数目也显著增加。

首先，为了检查网页中是否存在XSS漏洞，需要编写代码可以使用图18。如果该漏洞存在，则用户将看到弹出的警报单词“xss”。如图所示的fromcharcode函数转换Unicode字符串xss的字符（88、83、83）。图19只是一个简短的代码版本图（a）中的代码，具有相同的用途。在注入代码之后，用户需要查看页面源并搜索<xss与&lt；xss。这将允许用户识别存在XSS漏洞。表3中提到的攻击向量10是攻击者可以用来在Web文档的各个位置插入恶意javascript的一些攻击向量。这个第一列，标记名描述用于构造攻击向量的HTML标记。这个第二列描述了用于相应标记的特征或属性插入恶意代码。第三列提供了有关矢量。最后，第四列用一个例子描述了攻击向量。这很重要注意，这些攻击向量只对应于反射和存储的XSS攻击。





1.img tag：img tag可用于通过使用javascript指令。浏览器IE7.0（Internet Explorer 7.0版）没有允许在图像上下文中使用javascript指令。但是，指令可以在其他上下文中使用。在序号1下的表格中，原则文中给出了XSS的有效实现方法。这些可以用于其他标签也。这里有几点可能有用。

•大多数XSS过滤器不检查是否存在重音符模糊。

•如果过滤器不允许任何类型的引用，那么攻击者可以eval（）。来自charcode的字符，并根据他/她自定义攻击向量需要。

•在Gecko渲染引擎模式下，firefox或netscape 8.1+不允许使用img标记内的javascript指令。因此，编码过程可能是被攻击者采用。

•javascript指令可以按文本分解，如中的示例所示。

以便混淆XSS过滤器。可以通过嵌入一个选项卡来破坏该指令，换行符、回车符或三者的任何一个编码版本。因此，XSS过滤器不应考虑这些中断。

•在javascript指令之前，可以使用空格或其他元字符。这个当XSS过滤器使用模式匹配进行检测时，琐事很重要。

•如果使用开放角括号而不是闭合角，则Firefox浏览器可以正常工作。在关闭标签时加括号。但这种行为在Netscape中是不允许的。

•dynsrc和lowsrc属性可用于构建XSS矢量。

2.脚本标记：脚本标记也可用作攻击向量来执行XSS攻击。以各种方式。以下几点可能会有所帮助。

•如示例所示，可以使用外部的开方括号。尤其是，如果一些XSS检测机制依赖于通过匹配检测恶意代码开角括号和闭角括号的数目相等，一定会失败。攻击向量。示例中使用的双斜线将结尾置于无关位置批注下的括号。

•实际上可以跳过gecko中script标记的“>.<script>”部分Firefox和Netscape 8.1浏览器的呈现引擎模式。浏览器通过附加脚本结束标记自动修复此问题。这样的概念可以用于创建恶意脚本，如示例所示。

•脚本标记中协议解析下的示例适用于IEIE中的浏览器和Netscape。如果示例中附加了结束脚本标记，它也适用于Opera浏览器。扩展名“.j”没有给出任何错误因为它是在浏览器已知的脚本标记上下文中使用的。十七接受的手稿接受的手稿

•可以使用扩展名.jpg重命名恶意javascript.js文件。这种攻击向量将避开依赖于识别.js的检测延伸。

3. 标题标签：结束标题标签可用于创建恶意代码，如serial下所示。桌子上的第三个。

4. iframe标签：iframe标签可以用来创建大量恶意脚本。使用iframes的实际目的是将一个网页加载到另一个网页中。为了使其恶意工作，攻击者以这样一种方式构建iframe：嵌入的页面只是一个像素正方形11。当此恶意代码被混淆时，此恶意活动将不被注意，并且看起来是非恶意的。iframe标记可以与事件处理程序一起使用，以创建破坏，如序列号11下的示例所示。

5. 对象标记：对象标记可用于嵌入恶意数据，恶意数据可以是包含XSS负载的外部网站。

6. 嵌入标记：嵌入标记可用于链接外部网站中承载的Flash文件。属性“allowscriptaccess”如果设置为“always”，则允许从任何地方，不管是哪个领域。

除了本节讨论的攻击向量外，恶意XSS攻击向量还可以通过使用其他技术（如HTML引号封装、URL字符串规避）来制作或者使用ssi（服务器端包括）12。除了这里讨论的攻击向量外，还有一些可以使用事件处理程序创建更多。事件处理程序可以合并到不同的在浏览器13中发生特定事件时执行指定操作的标记。事件处理程序分为不同的类别，如窗口事件属性、基于窗体的事件，键盘事件、鼠标事件、拖动事件、媒体事件、剪贴板事件和其他事件。事件处理程序的完整列表可以在14中找到。一些重要的分类见表4。

**2.3.2. 可执行性**

沙盒iframes:。如前所述，iframe标记或内联帧可以用作攻击发起XSS攻击的载体。攻击者通过注入恶意代码进入其内容。HTML5向名为沙箱。属性“sandbox”有助于控制恶意iframes的行为。这个属性为iframe内容15构建一组限制。换句话说，恶意iframe中的内容置于受控环境中。沙盒属性是支持Chrome（4.0及以上）、IE（10.0及以上）、Firefox（17.0及以上）、Navigator（5.0及以上）和Opera（15.0及以上）Web浏览器。沙盒属性执行

以下：

1. 表单提交被阻止

2. 脚本执行被阻止

3. API被禁用

4. 将iframe内容视为来自唯一来源

5. 拒绝通过标签使用插件，如<embed><object><applet>

6. iframe内容无法导航其顶级浏览内容

7. 自动触发的功能被阻止。示例：自动播放视频沙盒属性可以采用的值16如表5所述。如果属性值保留为空，则默认情况下会设置所有限制。这里还需要注意的一点是，如果攻击者在sandbox iframe之外启动XSS攻击向量，则沙盒概念将永远不会得到充分利用。

**2.3.3. 渗出**

为了完全保护Web应用程序免受XSS攻击，防御团队需要把几层安全措施落实到位。有几个层次的主要想法是即使一层未能检测到XSS攻击，其他安全层也会成功在识别它的时候。为了驯服飞速发展的网络，Mozilla的安全研究人员对内容限制提出了一个非同寻常的法。内容安全策略（CSP）是用于强制内容限制并充当安全层之一的方案。对于Web应用程序[24]。CSP对Web应用程序开发人员和和现在一样，管理员可以控制他们的网站之间的交互。以及该网站上的相应内容。

内容安全策略的概念受到了Web应用程序开发人员和支持的Web浏览器的一些限制。Web应用程序开发人员应该定义应用程序在正常情况下的行为方式。在其他换句话说，应用程序的预期行为应该由开发人员提出。

有了Web浏览器的额外支持，可以对应用程序的行为施加适当的限制。如果观察到任何偏离预期行为的情况，应予以阻止。毫无疑问。这些限制性功能有助于限制第三方不可信的数据。更具体地说，CSP提出了一个应用程序开发人员可以掌握两个非常重要的东西的方案。

1.可以加载到其应用程序站点中的内容类型。

2.如果所有的内容都是从哪里加载的。

这两点还可以确保在攻击者即将携带的某个时间点在XSS攻击中，他/她将无法将任何第三方内容导入Web应用程序以及他/她将无法从任何不受信任的第三方发出任何请求派对狂欢。为了进一步减少发生XSS攻击的可能性，脚本只允许从白名单来源执行。CSP已在中成功实现火狐浏览器及其实现细节可从[Ref No 3]获得。从纸上看。CSP也部署在Mozilla 17的网站上。以下是解释如何激活CSP和CSP内容的几个要点。下面我们讨论了CSP提出的基本限制，如[24]所述。

1.内容安全策略的激活：内容安全策略在HTTP响应的x-content-security-policy HTTP头。然后激活策略通过客户的浏览器。

2.CSP的内容：在HTTP头或以文本格式在文件中声明。受保护的文档和包含策略的文件应该有相同的起源。

（a）基本限制：为了防御XSS攻击，在激活CSP时浏览器会禁用一些功能。选项指令可用于重新启用这些禁用的功能。

i.规则1：停止执行内联脚本：Web应用程序代码和

不受信任的用户提供的不受信任的内容必须分开。这种分离是由CSP强制执行的。以前，浏览器无法理解两者之间的差异导致了XSS攻击的成功执行。由于这种分离，攻击者不受信任和注入的内容将

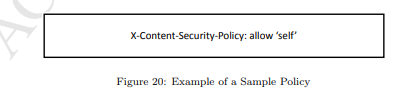
无法在浏览器中执行。表6显示了禁用的功能

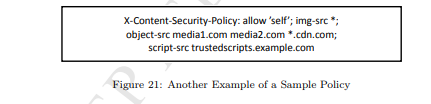
按照这个规则。第一列显示禁用的功能，第二列显示列显示禁用功能的功能如何仍可以由他允许的功能。

ii. 规则2:不应从字符串生成代码：eval（）等函数在处理特定于用户的不受信任数据时扮演非常重要的角色。使用eval（），字符串可以转换为代码。这种邪恶的手法函数作为从eval（）函数生成的代码对攻击者很有用很难在JavaScript程序中进行跟踪。所以，CSP完全阻止eval（）的使用。函数setTimeout（）、setInterval（）和函数构造函数都被CSP阻止，因为它们的行为与表达式（）。

（b）政策语言：Mozilla的研究团队在设计CSP概念时有两个目标。首先，限制可以加载到应用。第二，从内容加载的位置。第一个目标是通过执行上述基本限制。第二个目标实现了通过实现限制浏览器行为的指令。这个不同的指令如下表7所示。

图20给出了一个示例策略的示例。策略强制资源请求无论是图像、脚本还是任何其他资源，都应该与受保护的资源具有相同的来源资源。否则，请求将被忽略。图21给出了强制图像可以来自任何地方的策略，媒体内容必须来自一组受信任的媒体提供程序。脚本只能来自主机已知的服务器已清理的javascript。有关CSP[24]的详细信息，请参阅。





**参考文献**

[1] B. Gupta, D. P. Agrawal, S. Yamaguchi, Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security, IGI Global, 2016.

[2] Z. Huang, S. Liu, X. Mao, K. Chen, J. Li, Insight of the Protection for Data Security Under Selective Opening Attacks, Information Sciences 412 (2017) 223–241.

[3] S. Quinlan, S. Dorward, Venti: A New Approach to Archival Storage, in: In Proceedings of the USENIX Conference on File And Storage Technologies, Vol. 2, 2002, pp. 89–101.

[4] J. Li, Y. K. Li, X. Chen, P. P. Lee, W. Lou, A Hybrid Cloud Approach for Secure Authorized Deduplication, IEEE Transactions on Parallel and Distributed Systems 26 (5) (2015) 1206–1216. [5] M. Jouini, L. B. A. Rabai, A Security Framework for Secure Cloud Computing Environments, International Journal of Cloud Applications and Computing (IJCAC) 6 (3) (2016) 32–44. [6] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, K. Chen, Multi-key Privacy-preserving Deep Learning in Cloud Computing, Future Generation Computer Systems 74 (2017) 76–85.

[6] P. Li, J. Li, Z. Huang, T. Li, C.-Z. Gao, S.-M. Yiu, K. Chen, Multi-key Privacy-preserving Deep Learning in Cloud Computing, Future Generation Computer Systems 74 (2017) 76–85.

[7] J. Aikat, A. Akella, J. S. Chase, A. Juels, M. Reiter, T. Ristenpart, V. Sekar, M. Swift, Rethinking Security in the era of Cloud Computing, IEEE Security & Privacy 15 (2017) 60–69.

[8] J. Li, J. Li, X. Chen, C. Jia, W. Lou, Identity-based Encryption with Outsourced Revocation in Cloud Computing, Ieee Transactions on Computers 64 (2) (2015) 425–437.

[9] M. Ibtihal, N. Hassan, E. O. Driss, Homomorphic Encryption as a Service for Outsourced Images in Mobile Cloud Computing Environment, International Journal of Cloud Applications and Computing (IJCAC) 7 (2) (2017) 27–40.

[10] J. P. Singh, Analysis of SQL Injection Detection Techniques, Theoretical and Applied Informatics (TAAI) 28 (1-2) (2016) 37–55.