Computer Science and Application 计算机科学与应用, 2025, 15(7), 49-55
Published Online July 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.157179

Hans汉斯

基于Session欺骗攻击的Web应用程序防护

刘　柳，温且兵

江西工业工程职业技术学院信息工程学院，江西　萍乡

收稿日期：2025年6月14日；录用日期：2025年7月12日；发布日期：2025年7月18日

摘　要

本文对Web应用程序开发中的Session欺骗攻击进行了阐述，详细讲解了防范Session欺骗攻击的三种传统方法，并给出了防范代码，分析了三种传统防范方法的不足，新设计了一种通过Referer信息验证来加强对Session欺骗的防范，并给出了详细的实现代码和调用方法，最后提出了综合防护策略，将四种防范方法集成到一个模块中，通过调用该模块，方便实现了Session欺骗攻击的防范，增加了Web应用程序的安全性。

关键词

PHP，Session欺骗攻击，Session欺骗防护，Web应用程序安全

Protection of Web Application Programming Based on Session Spoofing Attacks

Liu Liu, Qiebing Wen

College of Information Engineering, Jiangxi Vocational College of Industry & Engineering, Pingxiang Jiangxi

Received: Jun. 14th, 2025; accepted: Jul. 12th, 2025; published: Jul. 18th, 2025

Abstract

This paper provides an in-depth examination of Session spoofing attacks in Web application development. It elucidates the three conventional methods of preventing Session spoofing attacks and presents prevention codes. The shortcomings of the three traditional prevention methods are analyzed. A new method is designed to strengthen the prevention of Session spoofing through Referer information verification, and detailed implementation codes and calling methods are provided. Finally, a comprehensive protection strategy is proposed; the four prevention methods are integrated into a module, which can be conveniently called to implement the prevention of Session spoofing attacks and thus increase

文章引用: 刘柳, 温且兵. 基于 Session 欺骗攻击的 Web 应用程序防护[J]. 计算机科学与应用, 2025, 15(7): 49-55.
DOI: 10.12677/csa.2025.157179

the security of Web applications.

Keywords

PHP, Session Spoofing Attack, Session Spoofing Protection, Web Application Security

Copyright © 2025 by author(s) and Hans Publishers Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/
Open Access

1. 引言

随着 Web 应用程序的不断发展和普及，对其安全性的关注也日益增加。Web 应用程序的后台管理模块通常采用 Session 来保存用户的登录状态，并通过 Session 变量来判断用户的操作权限。然而，这种便捷的身份验证机制也带来了潜在的安全风险。在现代 Web 应用程序和网站中，经常会出现许多开发者往往未加关注的漏洞[1]。如果黑客成功获取了登录用户的 Session_id 值，他们就可以通过伪造该 Session_id 来冒充合法用户，从而获得与该用户相同的权限，进而访问后台管理程序，执行管理操作。这对于整个应用系统来说，无疑是一个致命的安全隐患，可能导致数据泄露、系统被篡改甚至完全控制。

有学者回顾了 2016 年至 2023 年期间通过适当的研究筛选程序挑选出的约 30 项研究，描述了中间人攻击和会话劫持研究的总体趋势[2]。多年来，HTML 和 CSS 的功能得到了增强，以创建响应速度更快的网页，但所有这些网页创建更多的是信息共享，并没有真正处理用户输入问题[3]。本文将深入探讨 PHP 应用程序开发中 Session 欺骗攻击的原理与防范。通过与传统的 Session 欺骗攻击防范方法对比，分析其局限性，重点阐述增加 Referer 验证实现 Session 欺骗防范。本文旨在更有效地抵御 Session 欺骗攻击，显著提升 Web 应用程序的安全性。

2. PHP Session 简介

PHP 是一种被广泛使用的开源服务器端脚本语言和运行在 Web 服务器端的 HTML 内嵌式脚本语言，提供了简单、实用的动态主页解决方案，能够实现边开发边应用及跨平台运行，语法设计简单。同时，作为开源软件，PHP 的源代码完全公开，用户可免费获取安装包，在 Windows 或 UNIX/Linux 平台上快速搭建运行环境，具有良好的跨平台性和广泛的应用基础。但是 PHP 语言的脆弱性，使有效发现 Web 漏洞成为网站安全的重中之重[4]。在 PHP 中，Session 用于在用户访问网站期间记录其状态信息。从进入网站开始到会话结束的整个过程，用户可在 Session 中注册变量，这些变量存储在服务器端，安全性高、生命周期内可在多个页面中共享使用。

当启动 Session 时，服务器会生成一个唯一的 Session_id，并通过 Cookie 方式发送到客户端浏览器。用户再次访问时，浏览器会将 Session_id 返回，服务器由此识别并调用对应的会话数据，实现用户状态的持续追踪。

通常情况下，在网站上的一个 PHP 页面中的变量是不能在另一个 PHP 页面中共享数据的。然而，有了 Session 后，Session 中注册的变量可以作为用户的个人全局变量使用，使得用户存储在 Session 中的变量可以在其他 PHP 页面中被存取和修改。利用这一特性，Session 被广泛应用于用户身份认证、程序状态记录、PHP 页面之间参数传递。

Session 是另一种维护用户状态的技术手段，与 Cookie 的不同之处在于，其信息储存在服务器端，用

户无直接查看权限[5]。Session 机制极大地提高了 Web 应用程序的交互性和用户体验，但其安全性也一直是 Web 开发中需要重点关注的问题。

3. Session 欺骗的实现

Session 欺骗是指攻击者利用非法手段获得用户当前正在使用的 Session_id，之后利用这个 Session_id 来访问站点，从而获得合法用户的身份，进而对 Web 应用程序进行恶意操作。

当用户访问 Web 站点并启用 Session 会话时，服务器会为该用户生成一个唯一的 Session_id。这个 Session_id 通常会存储在用户浏览器端的 Cookie 中。如果黑客通过某种手段(例如 XSS 跨站脚本攻击、网络监听、中间人攻击等)成功获取了存储在用户浏览器端 Cookie 中的 Session_id 值，他们就可以利用这个 Session_id 来冒充正常用户。

Session 欺骗攻击对 Web 应用程序来说是一个致命的安全隐患。它不仅会导致用户数据泄露、系统完整性被破坏，还可能对整个服务器的基础设施造成严重威胁，因此，针对 Session 欺骗的防范措施显得尤为重要。

4. Session 欺骗的防范措施

在 Session 欺骗的防范措施中，传统的防范方法主要有三种：给 Session 设置生存时间防范、检测 User-Agent 一致性防范，以及重置 Session_id 值防范。从理论上来讲，这些防范方法只是增加了黑客攻击的难度，并不能完全防范 Session 欺骗攻击。下面将详细介绍这几种防范措施及其对应的程序实现代码，并分析它们的优缺点。

4.1. 给 Session 设置生存时间进行防范

Session 的有效生存时间是控制其安全性的一个重要参数。在 PHP 中，Session 的默认有效生存时间通常是 1440 秒(即 24 分钟)。通过设置 Session 的准确生存时间，可以有效防止 Session_id 被窃取后被长期非法使用。当 Session_id 的生存时间到期后，即使黑客获取了该 ID，也无法再继续使用它进行欺骗。

需要注意的是，直接修改 php.ini 文件会影响到服务器上所有的 Web 应用。如果不想影响其他 Web 应用，可以通过程序控制的方法，仅对当前应用进行限定。以下是相应的程序代码实现：

```
<?php
function check_time()   //检查生存时间的函数{
    $expires = 20;   //超时时间变量，此处设置为 20 秒超时，可自行调整设置
    if (isset($_SESSION['last_time'])) {
        $tm = time() - $_SESSION['last_time'];   //计算离上次访问的时间间隔
        if ($tm >= $expires) {
            Session_unset();      //如果访问超时，则删除 Session 变量。
            Session_destroy();   // 销毁 Session，彻底清除会话数据
            exit('登录超时错误。');
        }
    }
    $_SESSION['last_time'] = time();   //将当前时间存在 Session 变量中
}
?>
```

在需要防护的后台程序中调用该函数 check_time()，即可起到判断时间是否超时，在两次访问的时间间隔超过 20 秒，程序就会清除 Session 变量，起到后台防护的作用，此法不能完全防范黑客攻击，超时时间设定得当，可以增加黑客入侵的难度，如果黑客取得了用户的 Session_id 值，只要在设定的时间间隔内欺骗，仍然能实现入侵。此法的弊端是当超时时间设定过长，黑客渗透的时间更充裕，安全性差些；若超时时间设定过短，安全性高些，但会影响用户后台操作的连贯性，用户在后台程序中停留的时间超时，就会退出后台登录状态，用户体验感会变差。

4.2. 检测 User-Agent 一致性进行防范

第一种给 Session 设置生存时间进行防范方法并不能完全抵御 Session 欺骗攻击，为了进一步增加黑客攻击的难度，可以采用检测 User-Agent 一致性进行防护。User-Agent 是 HTTP 请求头中的一个字段，它包含了发起请求的客户端(通常是浏览器)的类型、版本、操作系统等信息。不同用户浏览器的请求头部信息数据通常是不同的，因此，在防范 Session 欺骗中，可以通过判断用户的 User-Agent 头部信息是否一致来增强防护。

以下是相应的程序代码实现：

```
<?php
function check_uagent()    //检查 User_Agent 的一致性{
    if (isset($_SESSION['uagent'])) {
        // 比较当前 User-Agent 的 MD5 哈希值与 Session 中保存的哈希值
        if ($_SESSION['uagent'] != md5($_SERVER['HTTP_USER_AGENT'])) {
            Session_unset();      // 头部信息不一致，删除 Session 变量。
            Session_destroy();   // 销毁 Session，彻底清除会话数据
            exit('客户端信息异常。');
        }
    } else {
        // 若没有保存头部信息在 Session 变量中，则将信息保存(首次访问)
        $_SESSION['uagent'] = md5($_SERVER['HTTP_USER_AGENT']);
    }
}
?>
```

在需要防护的后台程序中调用该函数 check_uagent()，即可判断用户前后访问时用户头部信息是否一致，黑客 Session 欺骗攻击时，若访问服务器的头部信息和正常用户访问时的头部信息不一致，会出现“客户端信息异常”的提示，阻止黑客的访问请求。但是请求头部的信息可以伪造，若黑客获取 Session_id 的同时，再获取用户的 User-Agent 数据，就可伪造 User-Agent 信息绕过该函数的防范，实现欺骗攻击。

4.3. 重置 Session_id 值防范

重置 Session_id (Session Regeneration)是一种旨在降低 Session_id 被盗用后欺骗成功概率的防范措施。其核心思想是在用户执行某些敏感操作(如登录成功、权限变更)或定期地重新生成一个新的 Session_id，并使旧的 Session_id 失效。这样，即使黑客获取了旧的 Session_id，也无法再使用它进行有效的欺骗。

在 PHP 中，重置 Session_id 的语句非常简单：

```
Session_regenerate_id(true);
```

这种防范措施并不完全可靠，只要用户没有刷新 Session_id，黑客就可以通过获得 Cookie 中保存的重置 Session_id 进行伪造，实现欺骗攻击，但是加上重置语句可以增加 Session 欺骗攻击的难度。

5. 改进方案

5.1. 增加 Referer 验证

从前面三种防范方法可以看出，黑客可以通过伪造 Session_id 和 User-Agent 信息实现欺骗攻击，为了增加黑客欺骗攻击的难度，本文提出了采用 Referer 验证来加强 Session 欺骗的防范，增加了程序的安全性，防范代码如下：

```
<?php
function check_referer()    //检查来源访问页面是否一致
{
if(isset($_SESSION['referer']))
{
$temp=explode("/", $_SERVER['HTTP_REFERER']); //获得上次请求 url
$refer="/" . end($temp);    //从上次请求的 URL 中取出网页文件名
//判断上次访问的网页文件名和 Session 中存的网页文件名是否一致
    if ($_SESSION['referer']  ！ =$refer ) )
    {
Session_unset();     //来源页面不一致，删除 Session 变量。
        exit('客户端信息来源错误。');
    }
}
//将当前页面 URL 保存在 Session 变量中，以备下次访问时检查请求来源是否一致。
$_SESSION['referer'] = $_SERVER['REQUEST_URI'];
}
?>
```

上面的代码，通过判断用户上次访问页面是否一致，来确定用户的操作是否为正常操作，黑客不知道用户最后访问的 PHP 页面，无法伪造相关信息，提高攻击难度。

5.2. 综合防护策略

然而 Referer 验证仍存在局限，如字段缺失(如书签访问、HTTPS 跳转、浏览器设置等)或被篡改(如攻击者伪造、代理修改等)，仍可能被绕过。因此，应综合 Session 有效期控制、User-Agent 校验、CSRF Token 等多重防范措施，并合理设置 Referer 策略，对敏感页面严格验证，对公共页面适当放宽。同时，记录异常日志，提升整体安全性。为增强 Web 应用安全性，用户可根据需要将防范方法集成到一个函数中，在需要防范的页面中调用该函数即可。整合后的代码如下：

```
function Session_protect()     //防范 Session 欺骗攻击
{
    check_time();   //设置生存时间进行防范
    check_uagent();//检查 user_agent 的一致性进行防范
    Session_regenerate_id(true); //重置 Session_id 值防范
```

```
    check_referer();   //检查用户请求来源页面是否一致进行防范
}
```

在需要防范 Session 欺骗的页面中加入如下代码，调用 Session_protect 函数，即可实现综合防护。

```
Session_start();     //启用 Session 会话
Session_protect();   //防范 Session 欺骗攻击
```

将所有防范措施集合到一个函数中，可以提高代码的复用性和管理效率。然而，在实际应用中，需要根据具体情况调整这些防范措施的启用时机和严格程度。例如：

1) Session_regenerate_id(true)：不一定适合在每次页面加载时都调用，因为它可能会在某些情况下影响用户体验或与某些框架的会话管理机制冲突。更推荐在用户登录成功后或权限发生变化时调用。

2) check_referer()：对于某些应用场景可能过于严格，导致合法用户被误判。对于公共页面，Referer 验证可能没有必要；对于后台管理页面，则可以适当启用。

因此，虽然提供了整合代码，但开发者在实际部署时应根据应用程序的特点、安全需求和用户体验进行灵活配置和调整。

5.3. 实验数据与对比分析

为了验证 Referer 验证在防护 Session 欺骗中的效果，我们设计了基于 Apache 和 PHP 的实验环境，搭建了一个包含登录与后台管理页面的 Web 系统，并引入 Burp Suite、Postman 和自定义脚本进行攻击模拟。

实验包括四种防护策略：

1) 无防护：不部署任何防护作为对照组；

2) 传统组合：包括 Session 有效期限制、User-Agent 检测、Session_id 重置；

3) 仅 Referer 信息验证；

4) 综合防护：结合上述所有措施。

攻击方式包括 Session_id 窃取、伪造、User-Agent 与 Referer 欺骗，以及模拟合法用户行为，并记录攻击成功率及用户体验影响，见表 1。

Table 1. Comparison of protection strategies
表 1. 防护策略对比

防护策略	Session 欺骗攻击成功率(%)	用户体验影响 (1~5，5 为最高影响)	备注
无防护措施	100%	1	攻击者可轻易冒充合法用户
传统防护组合	20%	2	攻击者仍可通过快速欺骗或伪造 User-Agent 绕过，但难度增加
Referer 验证	60%	4	对伪造 Referer 的攻击有效，但对缺失 Referer 的正常用户影响较大
综合防护策略	5%	3	显著降低攻击成功率，但仍需平衡用户体验与安全性

实验结果表明，在无任何防护措施的情况下，Session 欺骗攻击的成功率高达 100%；采用传统防护(如 Session 有效期限制、User-Agent 检测和 Session_id 重置)可将成功率降至 20%，但由于 User-Agent 易被伪造，仍存在风险；单独使用 Referer 验证时，攻击成功率为 60%，虽能防御部分伪造攻击，但对正常

用户的访问造成较大影响；综合多种防护措施后，攻击成功率降至 5%，安全性显著提升，虽然对用户体验有一定影响。可见，Referer 验证具备一定效果，但必须与其他手段配合使用，构建多层次的防护体系，才能在保障 Web 应用安全的同时，兼顾用户体验。

6. 结语

本文深入探讨了 Web 应用程序开发中 Session 欺骗攻击的原理与防范。我们详细阐述了三种传统的防范方法：设置 Session 生存时间、检测 User-Agent 一致性，以及重置 Session_id，并分析了它们的局限性。在此基础上，本文提出并详细分析了增加 Referer 信息验证来加强对 Session 欺骗的防范，并给出了详细的实现代码和调用方法。通过实验数据对比，我们发现单一的 Referer 验证虽有其独到之处，但在实际应用中存在 Referer 字段缺失和被篡改的局限性。然而，当将 Referer 验证与其他传统防范方法(如时间限定、User-Agent 检测、Session_id 重置)集成到一个防护模块中时，能够显著提升 Web 应用程序抵御 Session 欺骗攻击的能力，有效增加了应用程序的安全性。本文提出的综合防范策略为 Web 应用程序的安全性提供了更强大的保障。在编写 Web 应用程序时，开发者应根据应用程序的特点、安全需求和用户体验，灵活配置和调整这些防范措施的启用时机和严格程度。虽然本文以 PHP 编程为例，但其防范思路和多层防御理念同样适用于其他动态网页编程语言(例如 ASP、JSP 等)，为解决 Web 应用中的安全问题提供了普适性的指导。

基金项目

2024 年度江西省教育厅科学技术研究项目“基于多源数据的网络安全可视化系统的研究与应用”(项目编号：GJJ2406808)；2022 年度江西省教育厅科学技术研究项目“对地方高校网络应用的完善与开发”(项目编号：GJJ2202113)。

参考文献

[1] Cherckesova, L., Revyakina, E., Roshchina, E. and Porksheyan, V. (2024) The Development of Countermeasures against Session Hijacking. *E3S Web of Conferences*, **531**, Article No. 03019. https://doi.org/10.1051/e3sconf/202453103019

[2] Muzammil, M.B., Bilal, M., Ajmal, S., Shongwe, S.C. and Ghadi, Y.Y. (2024) Unveiling Vulnerabilities of Web Attacks Considering Man in the Middle Attack and Session Hijacking. *IEEE Access*, **12**, 6365-6375. https://doi.org/10.1109/access.2024.3350444

[3] Xuan, N.Y., Juremi J. and Saad, N.H.M. (2021) Securing E-Commerce against SQL Injection, Cross Site Scripting and Broken Authentication. *Journal of Applied Technology and Innovation*, **5**, 73-77. https://api.semanticscholar.org/CorpusID:245667954

[4] 李家梁, 彭建山. 函数语义识别的 PHP 漏洞检测[J]. 信息工程大学学报, 2025, 26(1): 90-96.

[5] 徐杰, 陈宇琪, 兰浩良, 等. 基于 PHP 会话劫持的网络安全教学研究[J]. 电脑知识与技术, 2024, 20(10): 100-103, 107.