·安全技术· 文章编号：1000-3428(2018)11-0158-07 文献标志码：A 中图分类号：TP393

基于随机化参数名的跨站请求伪造防御方法

王应军，傅建明，姜百合

（武汉大学 计算机学院，武汉 430072）

摘 要：传统基于客户端的防御方法存在用户体验与兼容性差的问题，容易产生误报和漏报现象，不能有效地防御跨站请求伪造（CSRF）攻击。为此，提出一种对请求参数名随机化的防御方法。通过对网站的统一资源定位器地址中的参数名称，如 Form 表单中的参数名进行可逆加密，确保在一次会话交互过程中的所有请求的参数名都被随机化，防止攻击者获取参数名信息实施 CSRF 攻击。基于该方法设计并实现了开源 PHP 库，并部署在开源的 PHP 程序上。实验结果证明，与基于 Token 方法相比，该方法能够更有效地防御 CSRF 攻击。

关键词：Web 安全；跨站请求伪造攻击；基于多样性安全；参数名；随机化

中文引用格式：王应军，傅建明，姜百合. 基于随机化参数名的跨站请求伪造防御方法[J]. 计算机工程，2018，44(11)：158-164.

英文引用格式：WANG Yingjun，FU Jianming，JIANG Baihe. Cross-site request forgery defense method based on randomization parameter name[J]. Computer Engineering，2018，44(11)：158-164.

Cross-site Request Forgery Defense Method Based on Randomization Parameter Name

WANG Yingjun，FU Jianming，JIANG Baihe

（School of Computer，Wuhan University，Wuhan 430072，China）

【Abstract】Traditional client-based defense methods have poor user experience，poor compatibility，prone to false positives and false negatives，and cannot effectively prevent Cross-site Request Forgery（CSRF） attacks. To this end，a defense method for randomizing request parameter names is proposed. By reversibly encrypting the parameter names in the Uniform Resource Locator（URL） address of the website，such as the parameter names in the Form，it is ensured that the parameter names of all the requests in a session interaction are randomized，preventing the attacker from obtaining the parameter name information and implementing a CSRF attack. Based on this method，an open source PHP library is designed and implemented，which is deployed in open source PHP programs. Experimental results show that compared with Token-based methods，this method can effectively defend against CSRF attacks.

【Key words】Web security；Cross-site Request Forgery（CSRF） attack；diversity-based security；parameter name；randomization

DOI：10.19678/j.issn.1000-3428.0048368

0 概述

跨站请求伪造（Cross-site Request Forgery，CSRF）攻击就是攻击者通过一些技术手段欺骗用户的浏览器去访问一个自己曾经认证过的网站并执行一些操作（如发邮件、发消息、加好友、甚至是财产操作）。由于浏览器曾经认证过，因此被访问的网站会认为是真正用户的操作而去执行，实际上是执行攻击者构造的操作。

随着互联网、云计算、移动计算的高速发展，网站提供的功能越来越丰富，用户能够进行的操作也越来越多，从而也存在更多潜在的 CSRF 攻击。根据 OWASP[1] 发布的 Web 应用十大安全隐患报告，CSRF 排名前八位。2007 年，谷歌 Gmail 遭受了严重的 CSRF 攻击[2]，攻击者通过在恶意网站上伪造对受害者 Gmail 中邮件的转发请求链接，使得成千上万用户的邮件遭到泄露。新浪某社区因存在 CSRF 漏洞，诱骗用户点击可造成蠕虫[3]。Facebook 和 Dropbox 在联合使用过程中，因为 auth 使用不当，导致利用 CSRF 攻击可以上传任意文件到 Facebook[4]。据不完全统计，截止 2016 年 5 月，在乌云漏洞平台[5] 上提交的 CSRF 漏洞数累计有 330 多

基金项目：国家自然科学基金（61373168）。

作者简介：王应军（1993—），男，硕士研究生，主研方向为网络安全、Web 安全；傅建明（通信作者），教授、博士；姜百合，博士研究生。

收稿日期：2017-08-16 **修回日期：**2017-10-28 **E-mail：**yjwang_cs@whu.edu.cn

个,说明 CSRF 漏洞广泛存在。

目前主流防御 CSRF 的思路包括验证请求来源与防止请求重放。验证请求来源如验证用户提交请求的 referer,但 referer 是由客户端发送可以被伪造从而产生漏报。防止请求重放的常见方法包括在 Form 表单中增加 Token 或者在执行敏感操作时需要用户输入验证码,但要求用户输入验证码时会影响用户体验。基于 Token 的防御方法也能够利用 postMessage、JSONP 的方法,获取 Token 进而执行 CSRF 攻击[6]。以上的防御方法存在着用户体验差、兼容性差、防御方法易被绕过、Token 泄漏导致 CSRF 攻击等多种问题,不能在实际应用中有效地防御 CSRF 攻击。

为此,本文提出一种基于 URL 地址请求参数名随机化的 CSRF 防御方案。通过对请求参数名进行随机化,使其不可预测从而使攻击者无法进行重放。服务端接受浏览器请求后对请求参数名进行验证来判断请求是否合法。随机化的密钥在每次建立会话时产生,攻击者无法预知。被随机化的参数也是在服务器端随机化之后发送到浏览器,攻击者无法获知随机化的请求参数名。所以,基于参数名的随机化的 CSRF 防御方案本身不会泄露随机化参数的信息,同时随机化的操作是在服务器端完成,在保障用户体验的同时防御 CSRF 攻击。

1　CSRF 防御相关研究

CSRF 防御方法根据 CSRF 检测位置的不同可以分为服务端防御、客户端防御。客户端防御侧重于跨域请求的审查;服务端防御侧重于防止请求重放。

服务端常用的防御方法就是采用基于验证码的防御方法、基于 referer 的验证防御方法与 Token 的防御方法。

基于验证码的防御方案是在关键位置要求用户输入验证码。但是由于输入验证码的用户体验较差,目前这种方案没有广泛使用。基于 referer 的防御思路是服务器端验证请求的 referer。但是由于用户的隐私设置或者是网络传输的问题,服务器最终不一定能得到 referer,因此基于 referer 的防御方法存在漏报、误报的问题,也没有被广泛采用。

基于 Token 的防御方法[7]的思路是在需要进行敏感、重要操作时,在对应的 Form 表单的地方添加 Token,Token 一般是一个随机数。当用户通过表单提交请求时,这个 Token 也会一并提交上去。在正常访问时,客户端浏览器能够正确得到这个 Token。但是在 CSRF 攻击中,攻击者无法事先知道这个 Token,从而能够防御 CSRF 攻击。目前这种防御方法,如 Django、Ruby on Rails、ThinkPHP 被广泛地采用。但是这种防御方法也存在缺陷:1)如果对 Cookie 处理不当,则导致 CSRF 的防御方法可以被绕过[8];2)Web 应用与第三方进行跨域交互时可能会泄露 Token,如利用 JSONP、postMessage 可以获取到 Token[9]。

客户端的防御方法一般都是在浏览器端识别出 CSRF 攻击,阻断 CSRF 的请求或者剥离所有的认证信息。鉴于服务器的防护方法需要修改服务器端的代码,很多的学者都是研究基于客户端的防御方法。CsFire[10]是一款基于 Firefox 的插件,CSFire 对跨域的 2 个网站的相关性进行判断以此界定是否为 CSRF 攻击。CsFire 如果发现跨域的 2 个网站之前存在交互,则认为此请求合法,否则认为是 CSRF 攻击。但是此工具在面对 Flicker、Yahoo 等第三方登录的情形时会存在很大的误报。BEAP[11]则是通过判断当前用户发起的请求是否是基于用户意图以此来防御 CSRF 攻击的一款浏览器插件。BEAP 通过监控用户网页行为以此判断当前点击是否是基于用户的意图点击。例如如果用户在地址栏输入 URL 地址,则认为是基于用户的本意访问;如果用户点击邮件中的链接,则认为是 CSRF 攻击,从而阻止该请求。BEAP 这种方式存在很大的误报,目前很多网站注册都需要用户点击邮件中的注册链接进行确认,BEAP 对这类行为都会产生误报。文献[12]提出的同源信用应用协议以 Web 站点的信用度为指标,计算访问请求地址的信用来决定该地址是否可达,防止从跨域名网站引入恶意资源。但是该方法还是通过插件的方式加以实现的,导致协议还是可以被攻击者破解从而绕过。

从研究者们的研究成果以及实际应用可以看出,基于客户端的防御存在兼容性、用户体验差、漏报、误报以及易被绕过等不同程度的问题。基于服务器端的防御才能够有效地防御 CSRF 攻击,但是基于 referer 的防御方法由于 referer 容易被伪造从而绕过检测。验证码虽然能够有效地防御 CSRF 攻击,但是用户体验差。所以,目前最常用的防御方法是基于 Token 的防御方法,但是 Token 也存在着 Token 泄露的问题。

本文提出基于随机化的参数名跨站请求伪造的防御方法,将攻击者可伪造的请求参数名作为随机化对象,使其具有随机性和不可预测性,以此防御 CSRF 攻击。随机化技术是一种新型的保护系统免受攻击的通用方法,可以有效地打破同构和稳定性的环境,以异构和随机的环境来应对未知的攻击。

目前也有很多的学者研究如何将随机化的技术应用在 Web 安全防御中。文献[13-14]均提出通过指令集随机化的方法防御 SQL 注入攻击,其主要思想就是对 SQL 语句中的关键字后添加随机数,之后对组装的 SQL 语句进行验证。但是由于实现难度较高,最后采用的是通过代理服务器的方式实现

的。由于代理服务器需要对 HTML 解析和分析，因此这种方式的兼容性、开销都存在问题。文献[15]提出基于指令随机化的 XSS 的检测方法，通过 HTML/JavaScript 关键字会在末尾追加随机生成的字符串，攻击者无法预测，由于这种方式实现复杂同样采用的是代理的方式，因此也会存在兼容性差、开销大的问题。文献[16]提出网页标记随机化对抗 XSS 攻击，鉴于通过代理的方式实现随机化的效果较差，该方法主要是在 PHP 平台上实现。在服务器通过修改 Smarty 的代码来对 HTML 的标签进行标记，在客户端通过代理的方式对标记的标签进行验证，虽然具有很好的效果，但是仅适用于使用了 Smarty 的 PHP 代码。

鉴于目前随机化的方法在 Web 安全方面开始逐步应用，但是针对 CSRF 的防御却并没有使用随机化的方法，同时之前研究者们提出的针对 CSRF 的防御方法均存在不同程度的问题，本文提出了基于参数名的随机化防御方法，不同于通过代理的方式进行随机化和去随机化，本文主要是基于源代码的分析，提供一种通用的 CSRF 防御思路并且在 PHP 开源程序上加以验证。

2 CSRF 随机化防御方法

本节首先分析 CSRF 攻击的原理，然后提出一种基于请求参数名随机化的 CSRF 防御方法，最后说明基于随机化防御方法的 PHP 代码的实现。

2.1 威胁模型

CSRF 利用网站对用户网页浏览器的信任，欺骗用户的浏览器去执行非本意的操作。CSRF 攻击的原理如图 1 所示。

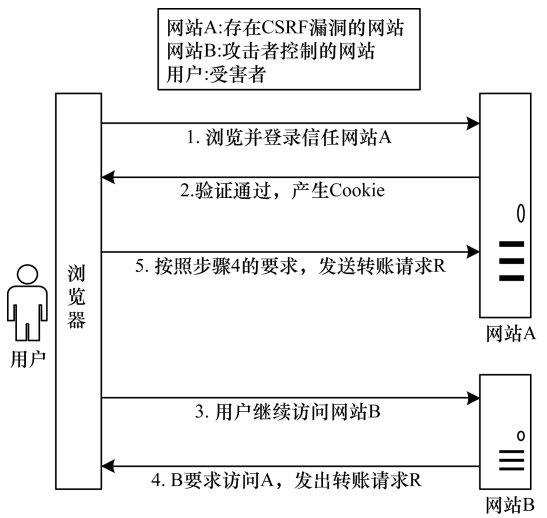

图 1 CSRF 攻击原理

当网站 A 收到了用户浏览器发送的转账请求 R 时，由于浏览器携带了用户的 Cookie，网站 A 认为请求 R 是由用户发送的就会按照用户的权限处理步骤 5的请求，这样网站 B 就达到了模拟用户操作的目的，成功地实施 CSRF 攻击。CSRF 能够成功实施的一个重要的原因是在于所有的用户在登录网站 A 时看到的网页结构与参数名都是一样的，攻击者很容易构造出一个与普通用户一样的操作的请求。

目前主要的 HTTP 请求方法是 GET 和 POST。从数据表现形式上来看，GET 请求方法的请求数据显式地嵌入在 URL 中；而 POST 请求方法的请求数据隐式地嵌入在 HTML 表单中，提交表单时用户无法看到提交的内容。但是无论是通过什么方式提交数据，提交的字段名对于所有的用户都是不变的。对于用户来说，假设转账请求的字段名为 fromaccount、toaccount、money。攻击者看到转账请求的字段名同样是 fromaccount、toaccount、money。这种静态的可预测的参数名，使得所有的用户发送请求时的参数名都一样，因而攻击者能够伪造出“合法”的恶意请求实施 CSRF 攻击。这种 CSRF 攻击方法本质上就是一种重放攻击。

2.2 请求参数名随机化

根据 2.1 节中对 CSRF 攻击原理的分析发现，阻止请求重放就能够有效地防御 CSRF 攻击。通过 Token 的方式来防御 CSRF 的防御方法利用的就是阻止攻击者重放的思想。本文提出的基于请求参数名随机化的防御方法，将传统的静态可预测的请求参数名作为随机化为不可预测的字符串以防御 CSRF 攻击，本质上也是一种阻止攻击者请求重放的方法，但是比 Token 方法能够更有效地防御 CSRF 攻击。

在随机化算法的选择方面，考虑到需要对参数进行加密和解密，还要保证在用户访问的时候参数名称不同，综合考虑采用的是对称性加密算法，将加密密钥保存在 SESSION 中。为了保证用户在每次与网站建立会话时的加密密钥都是不同的，密钥的生成及用户与服务建立会话的时间相关，同时采用摘要算法生成密钥。这样就保证了同一用户在不同时刻以及不同用户在同一时刻访问网站所得到的网站的参数名都是不同的，攻击者无法猜测到用户的随机化的参数名，也无法获知密钥，保证了 CSRF 攻击难以实施。整个随机化防御原理如图 2 所示。

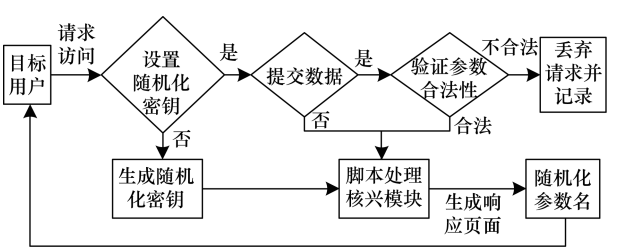

图 2 随机化防御原理

当用户向服务器发送请求之后,服务端如果判断用户不是第一次访问,则直接返回已经随机化后的页面。如果用户是第一次访问,则创建随机化密钥,同时对访问页面中的参数进行随机化处理,返回随机化之后的页面给用户。用户在页面中填写完毕发送至服务器之后,服务器对随机化的参数进行还原。如果服务器能够还原成为原来的参数,则正常响应用户的请求;否则,服务器判定为 CSRF 攻击,拒绝响应。

2.3　随机化方案实现

为了方便开发者部署本文的方案,本文实现一个 PHP 的第三方库 CSRFP[17],用于实现生成随机化密钥、参数名随机化以及对已经随机化的参数名还原。通过对国内外常见的 PHP 开源程序(wordpress、joomla、drupal、discuz、phpcms、dedecms、74cms)的分析,发现这些程序都会引入公共文件,如 config. php、common. php、functions. php 等。这些文件一般都是集合了程序常使用的功能,如格式化字符串、页面输出,同时也用于对 GET、POST 参数过滤,以防止 XSS、SQL 注入攻击。在这些公共文件中引入 CSFRP 库就可以在整个程序中对参数进行随机化和对随机化的参数名还原。开发者可以自行选择需要进行随机化防御的位置。开发者通过引入 CSRFP 库,不需要修改原来程序的逻辑和代码就能够很容易实现对网站的加固。

考虑到目前 CSRF 攻击主要是 POST 型的攻击。所以,在实验过程中只对表单中的参数名进行随机化。

对参数名进行随机化的示例代码如下:

```
< input type = " text"  id = " username"
name = " < ? php echo CSRFP: : encrypt( username) ? > " >
```

通过在公共文件类似 common. php 中引入 CSRFP 库,就能够完成对随机化的参数名进行还原,代码如下:

```
foreach ( $ _POST as $ random_name = > $ value) {
 $ name = CSRFP: : decrypt( $ random_name) ;
 $ _POST[ $ name] = $ _POST[ $ random_name] ; }
```

通过解密还原的代码可以看出,主要就是对 $ _POST中的已经随机化的 random_name 进行解密还原为 name。如果随机化的 random_name 是被攻击者篡改或者是由攻击者创建的,那么在进行解密为 name 时无法正确还原,之后的处理流程就会失败,从而丢弃这次请求,保护网站不会受到 CSRF 攻击。CSRF 随机化防御流程如图 3 所示。

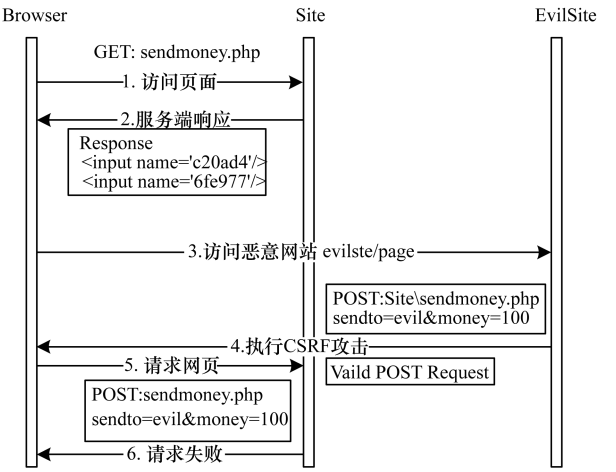

图 3　随机化防御流程

对比 2.1 节 CSRF 的攻击原理,当使用了随机化参数名的 CSRF 防御方法之后,参数名称完全是随机的字符串(步骤 2)。当攻击者发起 CSRF 攻击时,攻击者无法知道新的参数名而是按照之前的参数名发起攻击(步骤 4)。当服务器端收到这些参数名进行校验时,检验失败,从而判定为不是正常请求,拒绝响应(步骤 5 和步骤 6)。通过这种方法可以有效地防御 CSRF 攻击。

3　实验与结果分析

本节对基于随机化参数名 CSRF 防御方法的有效性和性能进行测试以及安全性分析,同时与目前主流的基于 Token 的 CSRF 防御方法进行对比分析。实验分析方法是通过在本地搭建 Apache、MySQL、PHP 的网站运行环境,应用随机化的防御方案加固存在 CSRF 漏洞的 PHP 开源程序,检测有效性;通过记录应用随机化的防御方案之后的访问时间来分析性能。

3.1　有效性测试

在有效性方面,实验选取了国内外存在 CSRF 漏洞的开源 PHP 程序进行部署和实验分析。根据 Czeskis[18] 所进行的实验方法,选取国外的 PHP 开源程序是 Selectapix、UseBB、PHPDug、PoMMo。国内的 PHP 开源程序选择的是 PHPCMS、EmpireCMS、FineCMS 和蝉知 CMS。所选取的 PHP 开源程序都存在 CSRF 漏洞。同时还与目前主流的 CSRF 防御方法——基于 Token 的防御方法进行了对比实验。实验结果如表 1 所示,其中,√为可以被防御,—为无法防御。

表 1 随机化防御实验结果

应用	版本	漏洞原因	Token 防御	随机化防御
selectapix	1.4.1	关键位置没验证	√	√
useBB	1.011	关键位置没验证	√	√
PHPDug	2.0.0	关键位置没验证	√	√
PoMMo	PR16.1	关键位置没验证	√	√
PHPCMS	V9.5.8	Token 绕过	—	√
EmpireCMS	6.6	关键位置没验证	√	√
FineCMS	V2.1.5	XSS 漏洞导致验证绕过	—	—
蝉知 CMS	5.3	关键位置没验证	√	√

从实验结果可以发现,采用基于随机化的防御方法比基于 Token 的防御方法更有效。例如 PHPCMS 的 CSRF 攻击使用 Token 无法防御,但是使用基于随机化的方法可以被防御。PHPCMS 的 CSRF 漏洞出现添加友情链接的地方,默认情况下会在添加链接后面加上 Token,第三方链接就能够得到 Token,造成了 Token 的泄露。当管理员审核此链接时,就会执行 CSRF 攻击。

如果采用随机化的防御方法,不会在任何地方添加 Token,那么就不存在泄露的问题,所以,能够防御 CSRF 攻击。而 FineCMS 的 CSRF 攻击,2 种防御方法均无法防御,原因在于 FineCMS 是通过 XSS 的方法得到了 Token,从而执行 CSRF 攻击。无论是采用 Token 还是随机化的防御方法,都可以通过 XSS 的方法得到相关参数名和参数值,所以无法防御。

综合实验结果可以看出,基于参数名的随机化方案比基于 Token 的防御方法能够更加有效地防御 CSRF 攻击。

3.2 性能测试

在随机化的方法中,需要生成随机化密钥并对参数进行加密。所以,性能测试包括不同的对称性加密算法与不同摘要算法的性能差异、基于 Token 的防御方法的性能差异、随机化的 input 数量对性能的影响。

实验方法是用 PHP 编写一个常见的用户注册页面,此页面中包含了一个注册的 Form 表单,通过增加 Form 表单中的 input 数量来测试 input 数量对防御方法的性能的影响。性能测试的方法采用的是文献[19]的分析方法,性能测试的数据是基于页面所加载的时间,页面加载的时间是以 Chrome Dev Tools Load 的时间作为访问时间,连续访问 10 次获取平均值作为最终的访问时间。图 4 是性能测试的实验结果。

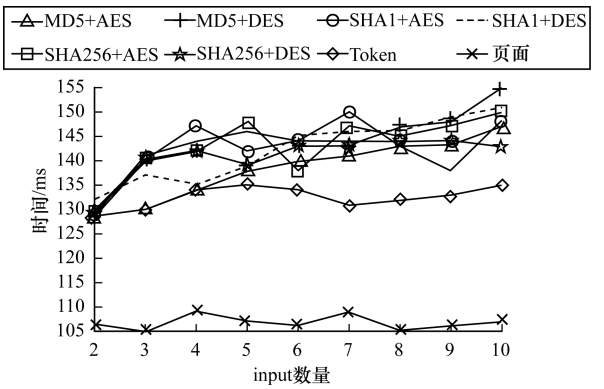

图 4 随机化防御性能测试结果

图 4 横坐标标识的是 Form 表单中的 input 数量,纵坐标标识页面加载的时间,每一条曲线表示不同的摘要算法和对称性加密算法组合一起的随机化算法。曲线 Token 表示的是在 Form 表单中增加一个隐藏的 Token,页面曲线表示只有 Form 表单(以下均称为页面曲线)。

从图 4 可以看出,不同的摘要算法和对称性算法组合在一起的随机化算法之前没有很大的差别,但是随着 Form 表单中的 input 数量变多,算法的耗时增加。基于 Token 的防御方法的耗时保持不变。虽然如此,但是基于随机化参数名防御方法的耗时与基于 Token 的防御方法耗时相差不大,最大差距为 15 ms,与页面曲线的差距为 25 ms,最大差距为 40 ms,不影响用户的体验。在空间消耗方面,基于 Token 的防御方法和基于随机化参数名的方法的空间消耗是一样的,对于每一个用户,只需要保存 32 位的字符串即可。对于 Token 来说,保存的是 Form 表单中的 Token 域的值,对于基于随机化参数名的方法来说,保存的是随机化的密钥,两者的长度都是 32 位。所以,2 种方法在空间损耗上面没有差异性。

在性能损耗方面,可以发现当只有 2 个 input 标签时,基于随机化算法的耗时与基于 Token 的防御方法耗时相当。所以,当 Form 中的 input 数量较多时,开发者可以有选择性地只对其中的几个 input 标签进行随机化,在保证随机化方法的有效性的同时还可以确保性能开销较小。

3.3 安全性分析

安全性分析主要包括 2 个方面,即防御方法本身的安全性以及程序使用了防御方法之后的安全性。

对于防御方法本身的安全性分析的前提在于,攻击者只能发起 CSRF 攻击而无法通过其他的攻击方法使随机化的防御方法失效,比如通过 SQL 注入、文件读取的方式获得网站的源代码从而绕过 CSRF 的验证。如果攻击者需要绕过防御方法就必

须要知道随机化密钥和随机化的加密算法。密钥的生成在每一次用户与网站建立会话时都不一样,与当前服务器端的时间相关,所以攻击者无法猜解出密钥的,同时密钥是保存在 SESSION 中的,攻击者无法通过任何的攻击手段得到 SESSION 中的值,从而保证了随机化密钥的安全性。至于随机化加密算法的安全性,由于加密算法是保存在服务器端,除非攻击者能够得到网站的源代码,在一般情况下攻击者是无法得到源代码的,因此加密算法也是比较安全的。

本文对比使用了防御方案前后的请求流程,随机化防御分析如图 5 所示。

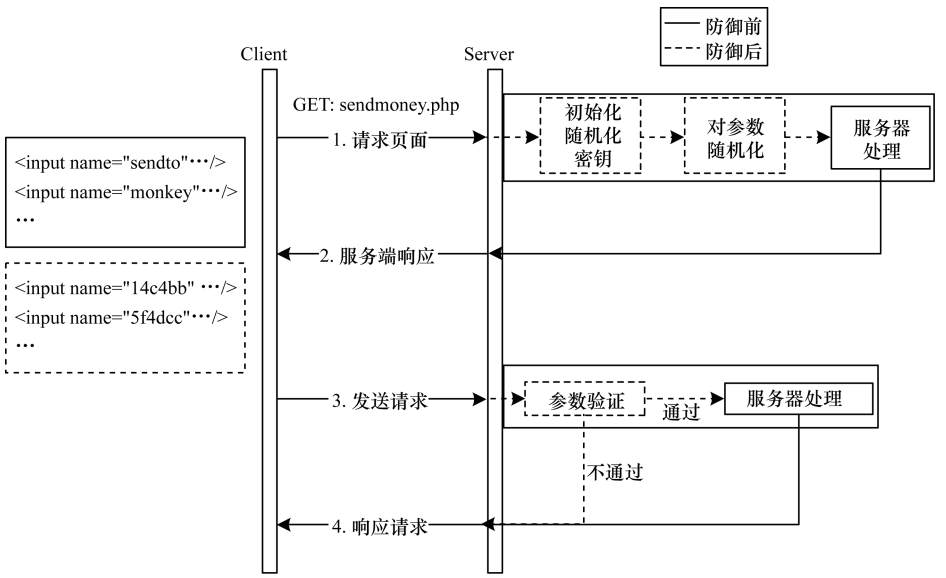

图 5　随机化防御分析对比

在图 5 中,虚线标识的是使用随机化防御之后在整个流程中发生变化的地方,包括页面的随机化和后台的验证操作。其中在客户端的改变是页面中的参数的变化,由原来的具有含义的参数变为了无意义的字符串。从前台页面中分析,对攻击者来说,页面中的参数名发生改变,并没有增加新的输入和输出以及功能方面的问题,因此虽然参数名发生了变化,但是并没有增加新的攻击点。后台服务器端程序的改变在于随机化密钥的生成和保存、随机化参数名以及参数验证。密钥的生成是由算法完成,密钥的保存是保存在程序中的 SESSION,SESSION 是保存在服务器端的键值对,密钥的生成和存储对程序的整个功能不会有影响。其次是对 Form 表单中的参数进行随机化,通过对参数名进行随机化的加密操作,不会加入新的输入输出和新功能,只会改变页面中的参数名,因此不会引入新的问题。最后是对参数名的验证,通过图 5 的流程发现,对参数名的检验是在服务器处理之前,这种处理方式在目前的 Web 开发中十分常见,类似 Java 中的 Filter、PHP 中的 include common. php。如果验证通过,再由后面后台服务器的程序进行处理。所以,对参数名的检验并不会影响整个程序的功能,只是多增加了对参数名的检验和过滤。

因此通过综合分析,当程序应用了随机化的防御方案之后,除了在 SESSION 中保存随机化的密钥,并没有在程序中引用新的变量、新的输入和输出,随机化的参数检验不会对程序的整个流程造成影响,这种防御部署方式也是目前 Web 开发中常用的防御部署方式。

4　结束语

针对目前流行的 CSRF 攻击,本文提出一种参数名随机化 CSRF 防御方法。该随机化方法部署在 Web 服务端,开发者应用本文开发的 PHP 开源库,自行选择需要随机化的参数名,利用请求和响应中随机化参数名的一致性识别伪造请求,从而检测 CSRF 攻击。与已有基于客户端的防御方法相比,随机化方法的漏报和误报较低。与已有服务端的 Token 策略相比,不存在因为 JSONP 和 postMessage 的方法使得 Token 泄露而导致 CSRF 攻击的情况。实验结果表明,CSRF 随机化防御方案比基于 Token 的方法能够更加有效地防御 CSRF 攻击,同时不影响用户体验。随着其不断应用,该方案会得到不断的完善。

为了能够更加方便地应用基于参数名随机化 CSRF 的防御方法,下一步在 Web 服务器如 apache、nginx 上以插件的形式进行参数名随机化,以在不修改程序代码的情况下完成参数名随机化的防护工作,同时解决性能开销的问题。

参考文献

[1] Category:OWASP top ten project-OWASP[EB/OL].[2017-07-24]. https://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013.

[2] Vulnerabilitynote VU#571584-Google Gmail cross-site request forgery vulnerability[EB/OL].[2017-06-24]. http://www.kb.cert.org/vuls/id/571584.

[3] 分享一个jsonp劫持造成的新浪某社区CSRF蠕虫|离别歌[EB/OL].[2017-07-24]. https://www.leavesongs.com/HTML/sina-jsonp-hijacking-csrf-worm.html.

[4] CSRF in Facebook/Dropbox[EB/OL].[2017-07-24]. http://blog.intothesymmetry.com/2017/04/csrf-in-facebookdropbox-mallory-added.html.

[5] 乌云搜索|搜索结果[EB/OL].[2017-07-24]. http://cb.drops.wiki/search.php?kind=bugs&keywords=csrf.

[6] 孙 丹.浅析CSRF攻击方式及防御技术研究[J].科技广场,2016(7):78-83.

[7] CALZAVARA S, TOLOMEI G, CASINI A, et al. A supervised learning approach to protect client authentication on the Web[J]. ACM Transactions on the Web, 2015, 9(3):15.

[8] CVE-2016-7401-Django CSRF防御绕过漏洞分析[EB/OL].[2017-08-12]. http://bobao.360.cn/learning/detail/3065.html.

[9] Hacking Slack using postMessage and WebSocket-reconnct to steal your precious token[EB/OL].[2017-08-12]. https://labs.detectify.com/2017/02/28/hacking-slack-using-postmessage-and-websocket-reconnect-to-steal-your-precious-token/.

[10] DE RYCK P, DESMET L, HEYMAN T, et al. CsFire: transparent client-side mitigation of malicious cross-domain requests[C]//Proceedings of International Symposium on Engineering Secure Software and Systems. Berlin, Germany: Springer, 2010: 18-34.

[11] MAO Z, LI N, MOLLOY I. Defeating cross-site request forgery attacks with browser-enforced authenticity protection[C]//Proceedings of International Conference on Financial Cryptography and Data Security. Washington D.C., USA: IEEE Press, 2009: 238-255.

[12] 蔡 亮,刘世贤.同源信用引用协议研究[J].计算机工程,2010,36(12):184-186.

[13] BOYD S W, KEROMYTIS A D. SQLrand: preventing SQL injection attacks[C]//Proceedings of International Conference on Applied Cryptography and Network Security. Berlin, Germany: Springer, 2004: 292-302.

[14] 李 原,蒋华伟.基于指令集随机化的SQL注入防御技术研究[J].计算机与数字工程,2009,37(1):96-99.

[15] 黄 俊,程绍银,蒋 凡.基于指令集随机化的XSS检测和防御系统[J].电子技术,2014(4):8-11.

[16] VAN GUNDY M, CHEN H. Noncespaces: using randomization to enforce information flow tracking and thwart cross-site scripting attacks[C]//Proceedings of NDSS'09. Washington D.C., USA: IEEE Press, 2009: 125-136.

[17] CSRF-protection: randomization defensive against CSRF attack[EB/OL].[2017-05-21]. https://github.com/spoock1024/CSRF-Protection.

[18] CZESKIS A, MOSHCHUK A, KOHNO T, et al. Lightweight server support for browser-based CSRF protection[C]//Proceedings of the 22nd International Conference on World Wide Web. New York, USA: ACM Press, 2013: 273-284.

[19] 梁 晟,李明树,梁金能,等.Web应用程序运行响应时间的实验研究[J].计算机研究与发展,2003,40(7):1076-1080.

编辑 索书志

(上接第157页)

参考文献

[1] 韩 敏,周应华,蔡应繁,等.HSPA终端小区选择与重选的设计与实现[J].软件,2011,32(4):19-22.

[2] 郑 立,沈 政,董志远.基于接入概率的LTE小区重选优化算法分析[J].电子技术应用,2012,38(9):103-106.

[3] 郑龙龙.LTE-A异构网络中小区选择与重选算法研究[D].西安:西安邮电大学,2015.

[4] 尚青为.面向移动通信安全的伪基站识别研究[D].北京:北京邮电大学,2015.

[5] PAPPU R. Physical one-way functions[D]. Cambridge, USA: Massachusetts Institute of Technology, 2001.

[6] 阴亚芳,李 锋.LTE小区选择和重选的分析与研究[J].邮电设计技术,2013(8):15-18.

[7] 熊健翔,莫宏波,果 敢,等.LTE小区重选准则分析与研究[J].电信网技术,2009(10):23-26.

[8] BOLOTNYY L, ROBINS G. Physically unclonable function-based security and privacy in RFID systems[C]//Proceedings of the 5th Annual IEEE International Conference on Pervasive Computing and Communications. Washington D.C., USA: IEEE Press, 2007: 211-220.

[9] 寇红召,张紫楠,马 骏.基于物理不可克隆函数的RFID双向认证[J].计算机工程,2013,39(6):142-145.

[10] BURROWS M, ABADI M, NEEDHAM R M. A logic of authentication[J]. Proceedings of the Royal Society of London: A Mathematical, Physical and Engineering Sciences, 1989, 426(1871): 233-271.

[11] 刘博雅,刘年义,杨亚涛,等.基于椭圆曲线密码的无线射频识别双向认证协议[J].计算机工程,2017,43(1):196-200.

[12] 李子臣,刘博雅,王培东,等.基于SM2与零知识的射频识别双向认证协议[J].计算机工程,2017,43(6):97-100,104.

[13] SAUI E. A graphical environment for the facilitation of logic based security protocol analysis[J]. South African Computer Journal, 2000(26): 196-200.

[14] 张亚力,郭亚军,崔建群,等.一种新的超轻量级RFID认证协议[J].计算机科学,2017,44(1):183-187.

[15] 马成林,肖美华,邓春艳,等.基于改进GNY逻辑的Kerberos*协议安全性分析[J].计算机与数字工程,2014,42(10):1758-1762,1882.

编辑 金胡考