

本科生毕业论文（设计）



题 目  基于代码混淆的Web站点动态防御系统

学 院  软件学院

专 业 软件工程

学生姓名 邵欣欣

学 号  **2015141463138** 年级 **2015**

指导教师 王海舟

教务处制表

二Ο一九年四月二十日

基于代码混淆的Web站点动态防御系统

软件工程

**学生** 邵欣欣 **指导老师** 王海舟

网络攻击从简单的脚本运行，已经发展到自动化工具操作。借助自动化工具，网络攻击不再是高级黑客组织专属，开始向“低成本、高效率”的趋势发展；漏洞被快速曝光，快速利用。传统基于特征和规则的入侵检测系统、Web应用防火墙（WAF）等已经无法有效地检测和拦截这些攻击。

由此针对自动化漏洞扫描、撞库扫号、恶意爬虫、暴力破解、薅羊毛等自动化攻击技术，开发“基于代码混淆的Web站点动态防御系统”。

“基于代码混淆的Web站点动态防御系统”，是在Nginx作为反向代理服务器的环境下，首先对请求做自动化攻击的检测与过滤；其次对发送给用户的响应做代码保护处理。在系统实现上，首先基于Nginx平台开发的一个可编译进Nginx来使用的第三方模块，然后利用一次性动态令牌来过滤自动化请求；最后使用代码混淆技术来对发送到用户浏览器的JavaScript代码做混淆保护。

该系统转变以往的被动防御思路，展开主动网站防御，改变攻防态势。有效降低自动化攻击技术的安全威胁，防护网站的安全。

**关键字**：反向代理；网站防御；动态令牌；主动防御

Website Dynamic Defense System based on Code Obfuscation

Software Engineering

**Graduate:** Shao Xinxin **Supervisor:**Wang HaiZhou

The security of the website has not received the attention it deserves. Many companies' online business systems are attacked by robots with simulating human behavior. The consequences are preemption of resources, fraudulent use of identity, fraudulent transactions, and information disclosure. The development speed of automated attack technologies is fast.These attak include automated vulnerability scanning, database scanning, malicious crawling and brute force cracking. The lack of security is always a problem for enterprise. Most of the current security protection products on the market are based on rules, and the overall product focus is on matching characteristics. Traditional feature-based and rule-based intrusion detection systems, Web Application Firewall (WAF), etc. have been unable to effectively detect and intercept these attacks, and the protection of websites urgently needs a new way to deal with them. In response to the above problems, this paper designed and developed a Web defense module used in the Nginx reverse proxy environment to help website developers solve the above problems.

The main contents of this paper are as follows:

(1) Analyze the advantages and disadvantages of the current solutions to the automated attack, and propose improvements based on the actual security defense status of each website.

(2) Analyze the goals that the system needs to achieve, and implement ideas. Carry out detailed requirements analysis; design the system architecture, propose the architecture mode that should be used; divide the system into two modules: data acquisition module and monitoring system display module, select the development language and development framework adopted by each module; design The data structure of the system completes the data dictionary design of the database.

(3) Design and implement the following modules: filter the client's request through dynamic ID matching technology; support the dynamic ID matching mechanism on the browser through the browser plug-in; use the code obfuscation technology to the JavaScript script involved in the project and The JavaScript script at the target site is confusing.

(4) Test with an actual website. Improve the system from performance and fault tolerance to ensure that the function of the protected site is not affected.

**Keywords**: reverse proxy; website protectrion;development of Nginx module; Dynamic ID

目录

[第一章 绪论 - 8 -](#_Toc7254162)

[1.1 研究背景及意义 - 8 -](#_Toc7254163)

[1.2 研究现状 - 9 -](#_Toc7254164)

[1.3 主要工作 - 11 -](#_Toc7254165)

[1.4 论文组织 - 12 -](#_Toc7254166)

[1.5 小结 - 12 -](#_Toc7254167)

[第二章 预备知识 - 13 -](#_Toc7254168)

[2.1开发技术 - 13 -](#_Toc7254169)

[2.2小结 - 15 -](#_Toc7254170)

[第三章 系统分析 - 16 -](#_Toc7254171)

[3.1 需求分析 - 16 -](#_Toc7254172)

[3.2 系统设计 - 19 -](#_Toc7254173)

[3.3 小结 - 24 -](#_Toc7254174)

[第四章 系统实现 - 25 -](#_Toc7254175)

[4.1系统环境 - 25 -](#_Toc7254176)

[4.2模块描述 - 25 -](#_Toc7254177)

[4.3运行效果 - 29 -](#_Toc7254178)

[4.4小结 - 30 -](#_Toc7254179)

[第五章 系统测试 - 31 -](#_Toc7254180)

[5.1 测试环境构建 - 31 -](#_Toc7254181)

[5.2 测试内容 - 32 -](#_Toc7254182)

[5.3 测试结论 - 39 -](#_Toc7254183)

[5.4 小结 - 39 -](#_Toc7254184)

[第六章 总结和体会 - 40 -](#_Toc7254185)

[6.1 工作总结 - 40 -](#_Toc7254186)

[6.2 心得体会 - 41 -](#_Toc7254187)

[参考文献 - 42 -](#_Toc7254188)

[附录1 任务申请书 - 45 -](#_Toc7254189)

[附录2 可行性研究报告书 - 49 -](#_Toc7254190)

[附录3 项目开发计划书 - 56 -](#_Toc7254191)

[附录4 数据要求说明书 - 62 -](#_Toc7254192)

[附录5 需求说明书 - 65 -](#_Toc7254193)

[附录6 用户指导手册 - 76 -](#_Toc7254194)

[附录7 概要设计说明书 - 87 -](#_Toc7254195)

[附录8 数据库设计说明 - 93 -](#_Toc7254196)

[附录9 组装测试计划 - 97 -](#_Toc7254197)

[附录10 详细设计说明书 - 104 -](#_Toc7254198)

[附录11 模块开发说明书 - 111 -](#_Toc7254199)

[附录12 单元测试报告 - 115 -](#_Toc7254200)

[翻译 - 124 -](#_Toc7254201)

第一章 绪论

1.1 研究背景及意义

传统的网站设计往往侧重于系统的功能和服务，而忽略了网站自身的安全问题。目前，网站的使用面临很多安全威胁。例如政府服务类网站，需要向用户提供查询服务，但是一旦遭遇爬虫，很容易被攻陷，或者消耗资源，或者被发现网站漏洞;电信运营商的网上营业厅，银行、甚至很多企业的网上在线业务系统，经常会被机器人通过模拟人的行为发动攻击。这些非“人”行为，极易得手的最根本原因是，面对看似完全正常的业务逻辑操作，现有安全手段却乏力于如何做出甄别，而这些威胁或者攻击一旦成功，即可抢占各类资源，进而可冒用身份，进行欺诈交易，甚至获得大量的客户信息、企业信息，造成大规模信息泄露。比如：针对一些商城的兑换服务，进行篡改积分，一些航空公司每每放出资源，几秒钟内就会被黄牛党抢光，抢占低价票资源。

当前市场上安全防护产品大多都是基于规则，产品总体偏重于匹配特征。显然，这样不停地找漏洞、打补丁令企业置身于千变万化攻击的被动中。特别是随着互联网+在国内的大力发展，像自动化漏洞扫描、撞库扫号、恶意爬虫、暴力破解、薅羊毛等自动化攻击技术的发展速度丝毫不亚于企业新业务的发展，安全手段的乏力始终是Web站点防护的一个难题。传统基于特征和规则的入侵检测系统、Web应用防火墙（WAF）等已经无法有效地检测和拦截这些攻击，网站的防护急需一种新型的方式来应对。

同时，我们知道，Javascript脚本是一种可读性较高的解释性语言，通常直接嵌入到网页文件中，一般用户都可以使用浏览器获得源码，对攻击者来说可轻易地获取源码进行逆向分析和恶意利用，这对承载Web应用重要核心业务流程信息的Javascript安全性带来巨大威胁。[1]

鉴于此，本项目拟通过人机识别技术和动态ID匹配技术对客户端的请求进行人机识别，以及通过随机化代码混淆技术对目标站点页面关键元素进行封装处理后再发送给用户，实现对网站的动态防御，有效降低自动化攻击技术的安全威胁，进一步防护网站的安全。

Nginx 是优秀的代理服务器，表现出众。根据Netcraft在2019年3月发布的调研报告显示，2018浏览器市场各占比如下：[2]

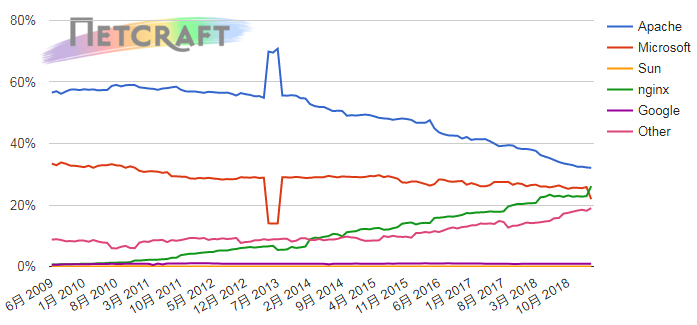


图 1- 1 Web服务器开发人员：域的市场份额

中国用户群体数量巨大，在特定工作或休息时间段的并发请求压力巨大，对Web服务器的设计要比欧美更加困难。快速增长的互联网

1.2 研究现状

美国国家技术委员会在2011年提出“移动目标防御”(MTD)的概念，也有学者将MTD技术称为“动态防御技术”、“动态弹性安全防御技术”或者“动态赋能网络防御技术”。

动态防御不同于以往的网络安全研究思路，它旨在部署和运行不确定、随机动态的网络和系统，让攻击者难以发现目标。动态防御还可以主动欺骗攻击者，扰乱攻击者的视线，将具引入死胡同，至少可以设置一个伪目标/诱饵，诱骗攻击者对具实施攻击，从而触发攻击告警。动态防御改变了网络防御被动的态势，改变了攻防双方的“游戏规则”，真止实现“主动”防御。[3]

我国也对动态防御技术进行了重点研究，邬江兴院士提出了网络空问拟态防御CMD思想。CMD理论在可靠性领域非相似余度架构基础上导入异构冗余动态重构机制，造成在功能不变条件下，目标对象内部的非相似余度构造元素始终在作数量或类型、时问或空问维度上的策略性变化或变换，用不确定防御原理来对抗网络空问的确定或不确定威肋。[3]

在邬江兴院士文章中提到，拟态安全防御是指在主动和被动触发条件下动态地、伪随机地选择执行各种硬件变体以及相应的软件变体，使得内外部攻击者观察到的硬件执行环境和软件工作状况非常不确定，无法或很难构建起基于漏洞(bug)或后门的攻击链.以达成降低系统安全风险的目的。[4]

目前的国内研究以高效能计算为目标，从体系结构创新入手，对高性能计算在多个典型领域的应用、结构和效能关系进行了深入分析。揭示出“刚性不变的体系结构支持差异巨大的应用是使计算效能低下的根本原因”，引入了“应用决定结构，结构决定效能”的理念，提出了基于多维重构函数化结构与动态多变体运行机制的拟态计算体系—拟态计算(mimic computing，MC)。拟态计算固有的随机性、动态性和不确定性阻断了目前攻击技术所依赖的攻击链完整性。拟态计算的信息系统具备内在的主动防御能力,因此，基于拟态计算的信息系统具备内在的主动防御能力，称为拟态安全防御(mimic security defense, MSD)[4]

MSD主要针对网络空间攻击成本和防御成本的严重不对称性，以及我国信息领域核心技术与产业基础严重滞后国家安全需求的严峻性而提出，它是一种改变游戏规则的变革性技术.力图扭转目前网络空间“易攻难守”的战略格局。[5]

美国阿贡国家实验室2016年3月22日报道，更多的动态防御技术被美国国家专利局授权。2015年美国新型动态防御公司吸引到投资界的关注，动态防御概念的3家公司获得超过5000万美元的融资。具中，CrowdStrike(动态防御概念)获得3000万美元天使轮融资，Morphisec获得800万美元融资，ShapeSecurity获得2600万美元融资。北京卫达科一技有限公司将动态防御技术应用到内网防御、边界防御、工控防御、DDos防御、Web应用防御、云安全等应用场景，研发6个系列的产品。[3]

截止目前，现有的网络安全产品大多采用传统的防火墙技术和入侵检测技术来实现对Web网站进行防护，在面对自动化攻击时十分脆弱。瑞数信息技术（上海）有限公司提供了类似的产品。

（1）动态算法生成：每次检查代码逻辑与形态均不同，且有效时间随整体访问量动态调整，攻击者每次必须在极短的时间内完成逆向，大幅提升了攻击难度，迫使攻击者放弃攻击。

（2）真实环境检查：利用不同浏览器中脚本引擎对特殊属性和语法的不同处理，验证真实的浏览器形态，大幅提升攻击者伪造终端运行环境的难度。

（3）攻击行为模式分析：基于人机互动理论，分析终端操作行为模式，有效识别非人为的操作行为，有效防止低频模拟操作攻击



图 1- 2瑞数信息公司动态安全

我们在此项目中将开发动态的防御，使防御手段变动，增强防御的机制，增加攻击者破解的难度，使用动态的id让攻击者无法用规律的自动化计算快速攻击、获取页面信息。

1.3 主要工作

基于代码混淆的Web站点动态防御系统，主要工作：

1. Nginx HTTP 模块的开发。Nginx整体采用高度模块化的架构设计。本系统需要开发一个根据Nginx框架定义的HTTP 模块。这样才能把程序嵌入到Nginx中，使得最终编译出的二进制程序Nginx 包含本系统的代码。
2. 动态令牌实现。没有令牌或者令牌错误的请求，下发令牌给予再次访问的机会；令牌正确的请求，给予正确响应，并准备好下一次的令牌。
3. JavaScript代码混淆。发送到用户浏览器的JavaScript代码，需要经过代码混淆进行保护处理。这些JavaScript代码包括受保护网站网页中使用的JavaScript代码和本系统中动态令牌在浏览器端的JavaScript程序代码。

。

1.4 论文组织

这篇论文内容结构如下：

第一章，本章讲述基于代码混淆的Web站点动态防御系统的应用背景和开发必要性，然后通过分析研究现状，对现有的防御进行详细分析调研，提出这些系统目前存在的一些问题，明确了开发本系统的重要意义。

第二章，本章描述开发J基于代码混淆的Web站点动态防御系统所需要的预备技术和知识，主要有Nginx模块开发，Chrome extension开发。

第三章。本章着重分析基于代码混淆的Web站点动态防御系统的功能和需求，并为整个系统进行概要设计和说明。着重分析了系统的功能性需求和数据库设计。

第四章，本章主要讲述了系统的实现方法，详细介绍动态ID模块和代码混淆模块的设计和实现，并且展示了基于代码混淆的Web站点动态防御系统的部分效果图例。

第五章，本章主要介绍了系统测试的过程，用户真实的操作手段对数据进行采集，对系统进行详细的测试，最后分析系统开发的结果。

第六章，本章总结整个项目，描述自己在基于代码混淆的Web站点动态防御系统的开发和设计过程中的理解和感悟。

1.5 小结

本章主要描述了基于代码混淆的Web站点动态防御系统系统的开发背景和开发意义，在分析研究了现有产品的研究现状和不足之处后，对本系统应该进行的改进进行了分析，进而提出了本系统应做的主要工作即功能需求，最后介绍了整篇论文的实现思路和组织结构。

第二章 预备知识

基于代码混淆的Web站点动态防御系统主要可以分成两个模块，即动态ID验证模块和JavaScript混淆模块。动态ID验证模块又由浏览器端插件和服务器端配合实现；使用混淆技术的代码保护模块使用JavaScript代码混淆技术。

2.1 Nginx模块开发

Nginx是一个高性能的HTTP和反向代理Web服务器。同时也提供了IMAP/POP3/SMTP服务。[7]

Nginx整体采用高度模块化的架构设计。Nginx 服务器被分解为多个模块，每个模块就是一个功能模块，只负责自身的功能，模块之间严格遵循“高内聚，低耦合”的原则。这下模块可以简单分类为按照核心模块，标准HTP模块，可选HTTP模块，邮件服务模块和第三方模块。其中，核心模块是 Nginx 服务器正常运行必不可少的模块，提供错误日志记录、配置文件解析、事件驱动机制、进程管理等核心功能；标准 HTTP 模块提供 HTTP 协议解析相关的功能，如：端口配置、网页编码设置、HTTP 响应头设置等；可选 HTTP 模块主要用于扩展标准的 HTTP 功能，让 Nginx 能处理一些特殊的服务，如：Flash 多媒体传输、解析 GeoIP 请求、SSL 支持等。

本系统开发了一个第三方模块，在该模块的开发中使用了Nginx基本的subrequest，proxy，slab等模块。过程中运用Nginx基本数据结构知识，模块结构知识，事件处理方式，共享内存管理和模块内引用库等技术进行模块开发。

2.2 Gumbo解析库

Gumbo提供一个轻量级的 HTML 解析库给开发人员，它没有外部依赖，大多数语言都可调用它。它遵循 HTML5 规范，符合HTML 解析算法标准，它通过了所有的html5lib-0.95测试，并且已经在 Google 索引的 25 亿个网页上进行过测试。并且Gumbo功能强大，可处理一些不规范的 HTML 标签。Google 开放了Gumbo的源代码。

本系统在反向代理转发目标网站页面内容时，使用Gumbo进行HTML解析，寻找ID的注入点。

2.3 Duktape引擎

Duktape是一个轻量级的嵌入式 JavaScript 引擎，专注可移植性和减少内存占用，有最小的平台依赖。遵循 Ecmascript E5/E5.1，对ECMAScript兼容性较好。Duktape专注于可移植性和低占用率。可以被轻松地集成进一个 C/C++ 项目中，在使用时，只需添加duktape.c 和 duktape.h文件到项目中就可以使用。Duktape有着丰富的文档资料，社区良好，帮助开发人员使用。它构建一个栈式环境来进行操作，开发需要着重关注计算堆栈入的次数，出的次数；Duktape的作者开发能力强，对开发和维护积极，并陆续在添加一些实用的特性。

在本系统开发中，针对Nginx模块编译的特殊性，将Duktape生成静态库，使用语法集成到本模块中。

本系统在实现混淆JavaScript脚本的开发中，使用JavaScript程序代码来实现混淆算法，混淆受保护站点的JavaScript脚本文件。因此，使用Duktape来执行实现混淆算法的JavaScript代码。

2.4 Chrome Extension插件

Chrome Extension扩展浏览器的功能，例如：捕捉特定网页的内容，捕捉HTTP报文，捕捉用户浏览动作，改变浏览器地址栏/起始页/书签/Tab等界面元素的行为，与别的站点通信，修改网页内容等。

相比起Firefox：

Chrome占有率更高，根据StatCounter公布的2019年3月份全球浏览器市场份额数据，Chrome浏览器以 62.63% 的市场占有率位于第一；[5]

应用场景更广泛，Firefox插件只能运行在Firefox上，而Chrome除了Chrome浏览器之外，还可以运行在所有Webkit内核的国产浏览器，比如360极速浏览器、360安全浏览器、搜狗浏览器、QQ浏览器等等；除此之外，WebExtension API作为新附加组件的开发标准，并且可以大部分兼容chrome的crx扩展。

Chrome Extension开发涉及到HTML，JavaScript和CSS技术点。浏览器扩展拥有一定高度的权限，在实现“捕获页面发出的HTTP请求请在URL上缀加ID”这一目标过程中，使用浏览器扩展是开发变得清晰明朗，避开了包括解析HTML页面超链接，文件引用和JavaScript函数覆盖等一系列复杂的实现。

在本系统中，Nginx模块在转发目标站点的响应到浏览器的过程中向响应页面注入ID，然后浏览器扩展从页面捕捉到ID。实现了ID验证机制的前后端配合。

2.5 AugularJS框架

AngularJS 是一款为了克服HTML在构建应用上的不足而设计的优秀的前端JS框架，是以一个JavaScript 文件形式发布的JavaScript 编写的库，可通过 script 标签添加到网页中。AngularJS有着诸多特性，最为核心的是：MVC、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。它通过指令扩展了 HTML，且通过表达式绑定数据到 HTML。

在本系统的浏览器插件“protect”的开发中，使用了ng-app 指令定义一个 AngularJS 应用程序。ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

2.6 JavaScript

JavaScript是因特网上最流行的脚本语言。[]一种直译式脚本语言，一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。[6]

2.7 小结

本章主要描述了开发基于代码混淆的Web站点动态防御系统所需要的预备技术和知识。主要介绍使用到的Nginx项目知识，包括模块结构，如何引用库到Nginx项目中，以及引用的Gumbo库和Duktape库，并简单描述了基于代码混淆的Web站点动态防御系统如何将这些技术引用其中。

第三章 系统分析

3.1 需求分析

基于代码混淆的web站点动态防御系统的需求从前后端两个方面来分析

Chrome Extension “protect”能够接收并存储用户设置的要作用的网页的一类网址的一个通配-pattern；抓取浏览器中网址符合该pattern的网页页面中id属性为” hxID”的“input”元素的“value”属性获得ID值;截获浏览器发往网址符合pattern的HTTP请求，在添加ID后再前往该网址。

Nginx第三方模块ngx\_http\_ipslab\_module能够管理（IP，ID）对，包括给新访问新增（IP，ID），根据IP查找（IP，ID），更新（IP，ID），删除（IP，ID）；能够转发浏览器请求；转发目标服务器响应；能够向所转发的响应内容中注入ID相关信息；能够混淆JavaScript代码。

3.1.1 数据分析

基于代码混淆的Web站点动态防御系统可以从Slab共享内存中的数据和内部交互，Chrome Extension 中的数据和内部交互以及二者之间的交互这三方面来进行分析。

1. Nginx模块ngx\_http\_ipslab\_module中的数据及内部交互

使用红黑树存储（IP，ID）对；节点数据结构以IP为键值，包括在红黑树结构上的增加，删除，查找，修改操作。

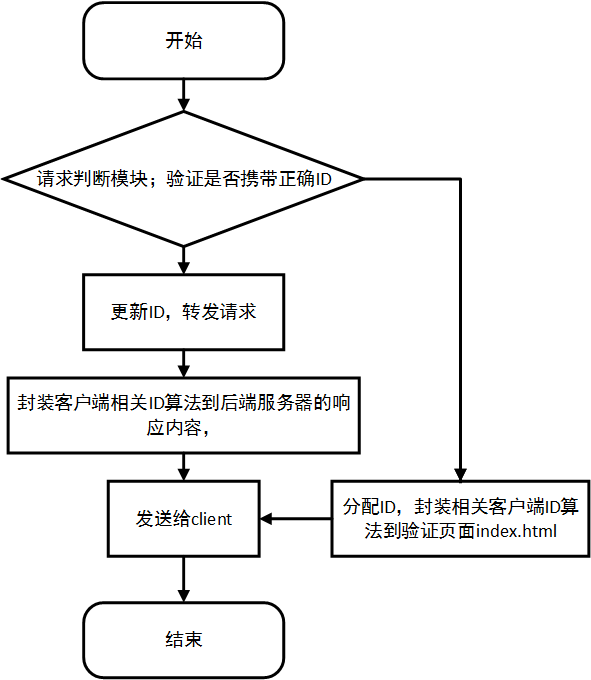


图3-1 动态ID模块服务器端对收到的请求进行验证和对应响应

1. Chrome Extension 中的数据及内部交互

扩展程序储存用户数据，即用户输入的要作用的网址模式。在加载到相应网站的网页页面后，抓取携带在网页页面中的ID并临时存储，在用户操作使浏览器发起符合网址模式的请求时，依据ID生成新的newID，携带在请求中发出。

1. 前后端数据交互

对于针对某一个IP产生的ID，我们将它下发给浏览器，同时存储到服务器的数据库slab,为方便区分,分别称它们为ID1和ID1’。

当浏览器根据ID1生成ID2，再次发起请求时，携带ID2。

Nginx反向代理接收到请求，后根据IP从slab中找到ID1’,并随即生成ID2‘。

比较ID2与ID2‘来确定是否是合法请求。

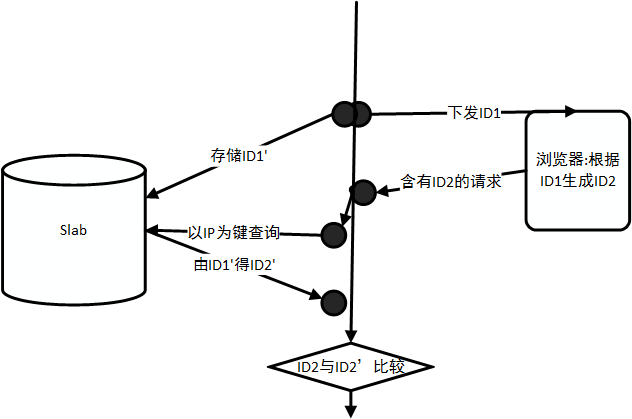


图 3- 2 客户端和服务器就ID的交互

3.1.2 功能分析

基于代码混淆的Web站点动态防御系统使用的用户是为自己的网站部署Nginx反向代理来增加安全，提高性能的网站技术人员。在选取Nginx提供的服务时，用户可以添加部署本模块来过滤掉一些自动化攻击，在一定程度上增加网站防御系数。为了实现过滤掉一些自动化攻击的功能，将基于代码混淆的Web站点动态防御系统主要分为两个部分：ID验证模块和混淆保护模块，其中ID验证模块又包括浏览器扩展和Nginx后端验证两部分相互配合来实现。

3.1.2.1 动态ID匹配进行人机识别

在slab共享内存中ID的更新，存储，修改，查看

浏览器与代理服务器之间ID交流的逻辑功能

3.1.2.2 混淆功能

在返回目标服务器的响应内容前，对发给用户的文件进行混淆操作。混淆使用随机代码混淆技术对受保护站点页面关键元素和客户端ID生成算法进行随混淆处理，增加攻击者对网页的逆向分析的成本。

3.1.3 性能分析

基于代码混淆的Web站点动态防御系统，结合实际使用情况需要满足以下非功能性要求。首先要保证可用性和性能。

1. 一定的透明性

作为以反向代理为基础的系统，要保证用户使用受保护网站提供的服务时通畅的。保证其完整性，本系统开发的模块必须具有一定的透明性，对用户正常使用服务不造成影响。

1. 性能

保证一定的性能，不能在反向代理处做无故的滞留影响响应速度。具体度量不超过原目标站点响应时间的两倍。

1. 容错性

基于代码混淆的Web站点动态防御系统，对一些开发中无法预料的请求可能会在使用中出现并影响到用户正常使用的情况，需提供一种容错的补充方案，一定保证受保护站点业务服务可以提供给用户。

1. 可用性

对于供浏览器用户使用的Chrome Extension“protect”需要简洁清楚的界面，需要给出操作成功的提示信息和错误操作的出错详细信息，对用户有指导作用，提供给用户的步骤要简单单调，提高可用性。

对于供网站部署人员，本系统提供的Nginx第三方模块ngx\_http\_ipslab\_modul是通过指令配置的方法来使用。为网站部署人员提供详细的指令介绍，配合在日志中输出详细的操作信息来让网站部署人员更深入了解本系统。

3.2 系统设计

基于代码混淆的Web站点动态防御系统。如下图是反向代理服务器的作用结构：



图 3- 3 反向代理器代理服务

动态ID模块将集成到Nginx中。

3.2.1 架构设计

系统架构如下：

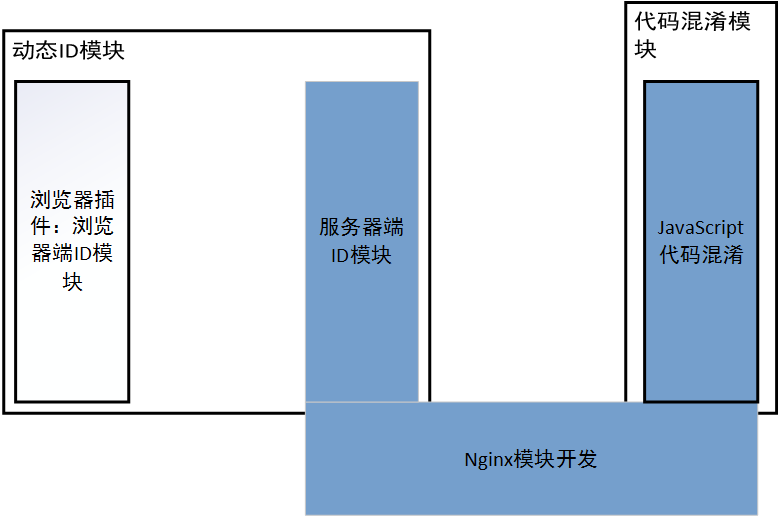


图 3- 4 系统架构图

1. 动态ID模块

动态ID模块整体设计方案

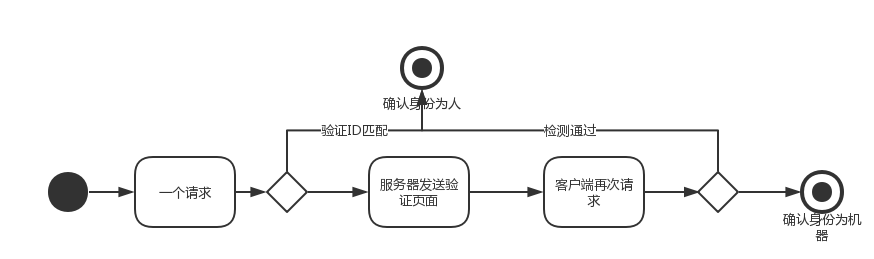
****

图 3- 5 动态ID模块

* 1. 客户端算法：在发起请求时，使用之前服务器下发的ID生成新的newID,并附加newID到请求中。
  2. 服务器端算法流程：

1. 存储下发给IP的ID即（IP，ID）记录。
2. 接收请求，进入自动化请求过滤环节。通过进入④，未通过进入③。
3. 对于ID不匹配的IP请求更新（IP，ID）记录，对于没有ID记录的IP请求添加（IP，ID）记录到存储库中，不转发用户请求到第三方服务器，同时响应用户一个特殊构造的HTML页面，该HTML页面保留了用户未通过的请求信息以及ID相关内容，以便用户再次发起携带ID的请求。

向第三方服务器发起请求，得到响应后，注入“检测自动化攻击”模块客户端需要的代码并对网页代码做保护后，转发给用户，更新（IP，ID）记录。

2. 系统流程-各模块交互：

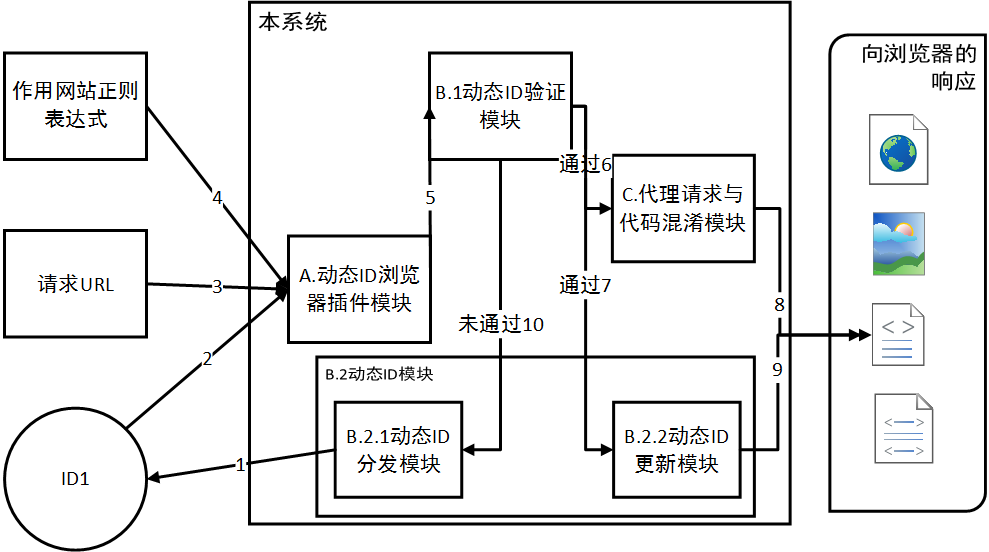


图 3- 6 解决方案流程图

3.2.2 数据设计

ngx\_http\_ipslab\_module模块内的结构体设计如下：

表3-1 ngx\_http\_ipslab\_module模块配置结构体ngx\_http\_ipslab\_conf\_t

|  |  |  |
| --- | --- | --- |
| **变量名称** | **变量类型** | **含义** |
| shmsize\_int | ngx\_int\_t | 共享内存的大小，以pagesize=4096为单位大小 |
| shpool | ngx\_slab\_pool\_t | 共享内存结构体 |
| sh | ngx\_http\_ipslab\_shm\_t | 控制红黑树和链表 |

表3-2 存储(IP,ID)对的节点ngx\_http\_ipslab\_node\_t结构体

|  |  |  |
| --- | --- | --- |
| **变量名称** | **变量类型** | **含义** |
| rbtree\_node\_data | u\_char | 实际存储信息开始的位置 |
| slab\_ID | u\_char[33] | 下发给浏览器端的ID |
| last\_slab\_ID | u\_char[33] | 上一次下发给浏览器端的ID |
| queue | ngx\_queue\_t | 一条一定长度保持访问记录的链表，用来淘汰过期节点 |

表3-3 ngx\_http\_ipslab\_shm\_t结构体

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| rbtree | ngx\_rbtree\_t | 红黑树用于快速检索 |
| sentinel | ngx\_rbtree\_node\_t | 使用红黑树必须定义的哨兵节点 |
| queue | ngx\_queue\_t | 淘汰链表 |

表3-4 ngx\_http\_ipslab\_ctx\_t表

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| out\_ctx | ngx\_chain\_t \* | 传递子请求返回结果 |
| hxID | ngx\_str\_t | 父子请求间传递ID |

表3-5枚举类型Append\_flag定义

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| pure\_chrctr | 0 | 标识添加“hxID=xxxx” |
| ampersand\_chrctr | 1 | 标识添加“&hxID=xxxx” |
| question\_chrctr | 2 | 标识添加“?hxID=xxxx” |

**3.2.3 UI设计**

基于代码混淆的Web站点动态防御系统中，Nginx端模块的使用是在Nginx的启动配置文件nginx.conf文件中配置使用。浏览器端的浏览器扩展页面设计简介，有必要提示信息，有操作成功提示，以下是浏览器扩展的界面设计：

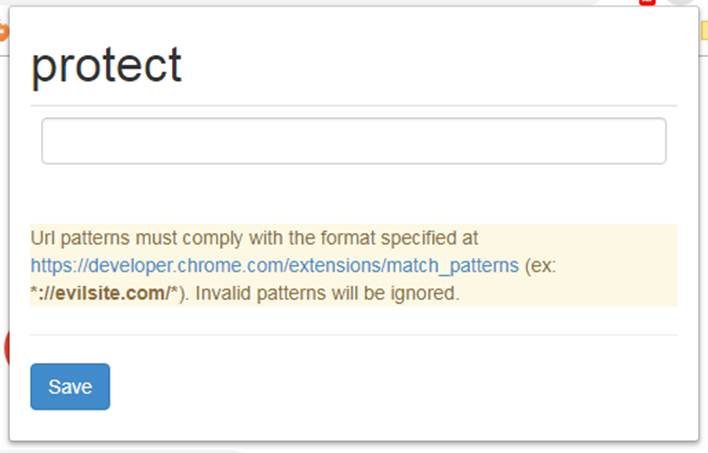


图 3- 7 浏览器扩展界面

需要介入Nginx的部分遵循Nginx使用指导。在配置文件中配置使用。

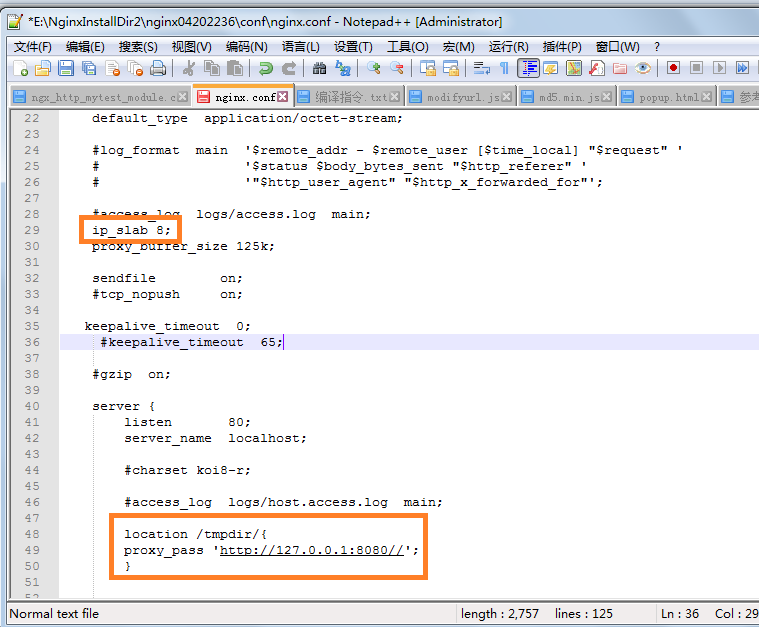


图 3- 8 Nginx 配置文件配置

3.3 小结

本章节着重分析了基于代码混淆的Web站点动态防御系统的功能性需求以及性能需求，从三个组件描述了系统的设计实现，详细描述了ngx\_http\_ipslab\_module模块所使用到的数据结构。

第四章 系统实现

4.1系统环境

基于代码混淆的Web站点动态防御系统

本项目将ID验证模块服务器端支持与混淆模块在ngx\_http\_ipslab\_module中用C实现。

将ID验证模块浏览器端支持使用Chrome Extension实现，使用HTML和JavaScript语言。

本项目开发环境：

在Windows操作系统下，安装Cygwin进行Nginx编译、部署。并将项目导入Eclipse-cdt IDE进行开发。

4.2模块描述

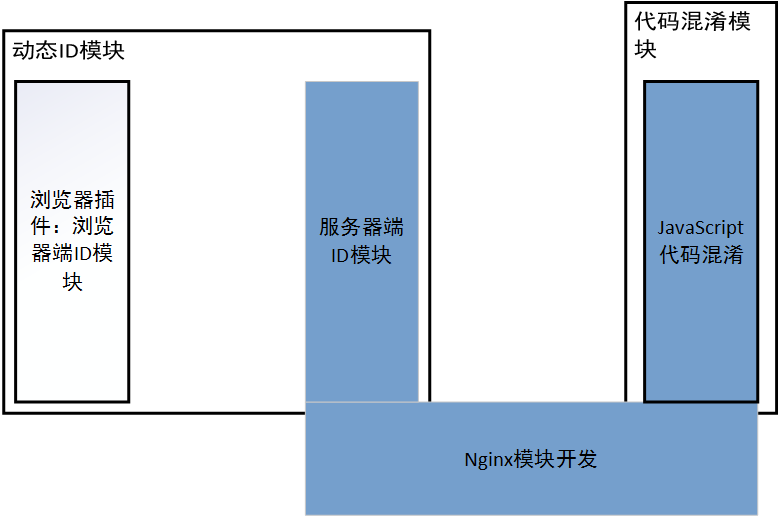


图 4- 1系统模块结构图

4.2.1 ID验证模块浏览器端支持

采用Chrome Extension，从网页页面获取ID生成newID，截获从网页发出的HTTP请求进行添加newID操作，然后再使用重定向技术真正发出请求。

Popup模块接收用户设置的网址模式：pattern并将其通过变量传递给Bcackground模块。

Content-Script模块保持从页面中抓取到的ID：hxID并将其通过message传递给background模块。

Background模块将收到的pattern和hxID 均存储到Chrome.storage中。

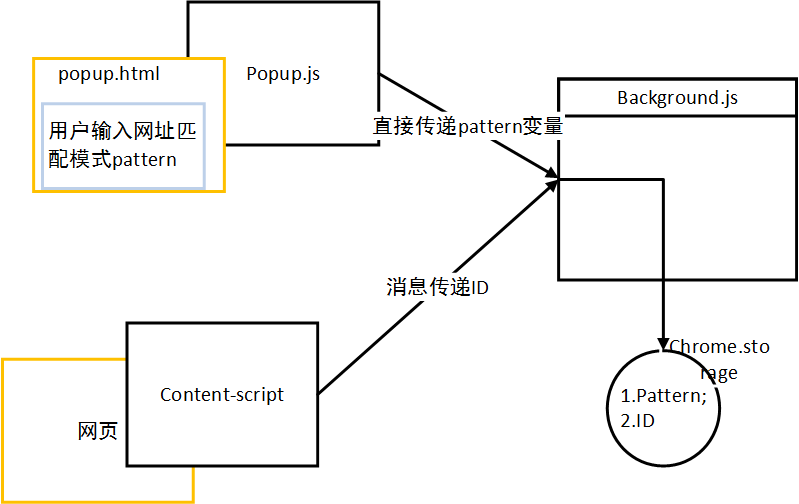


图 4- 2浏览器扩展内部消息交互图

Background.js的内部:

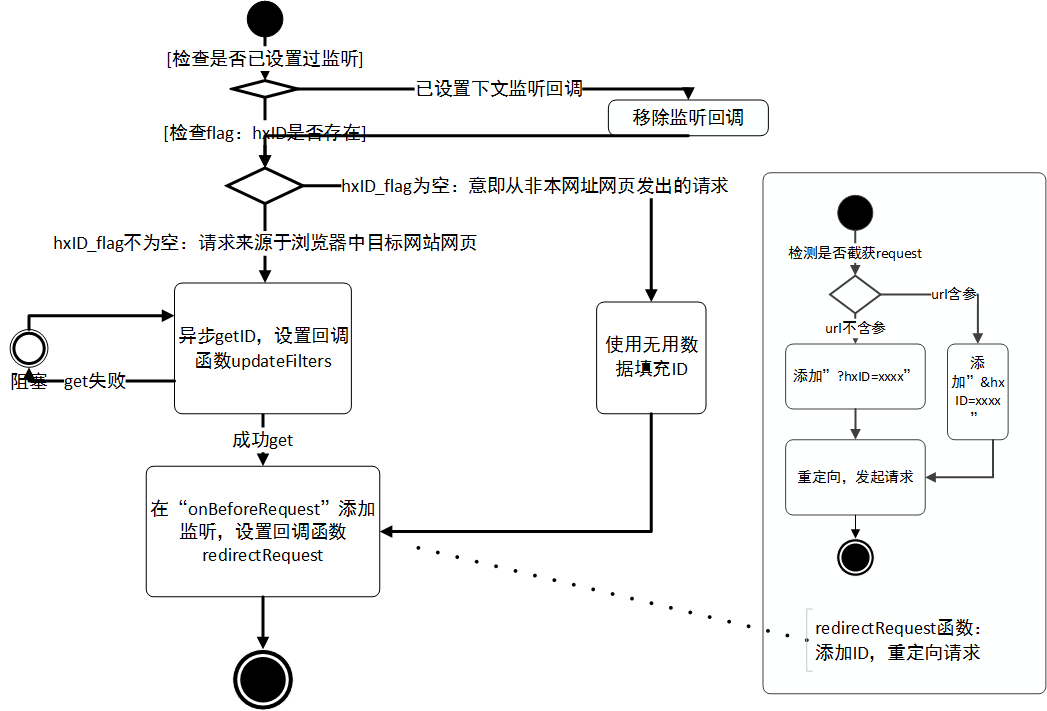


图 4- 3 浏览器扩展的后台逻辑

#### 4.2.1.1 动态ID模块服务器部分

这部分的代码实现位置在Nginx第三方模块ngx\_http\_ipslab\_module

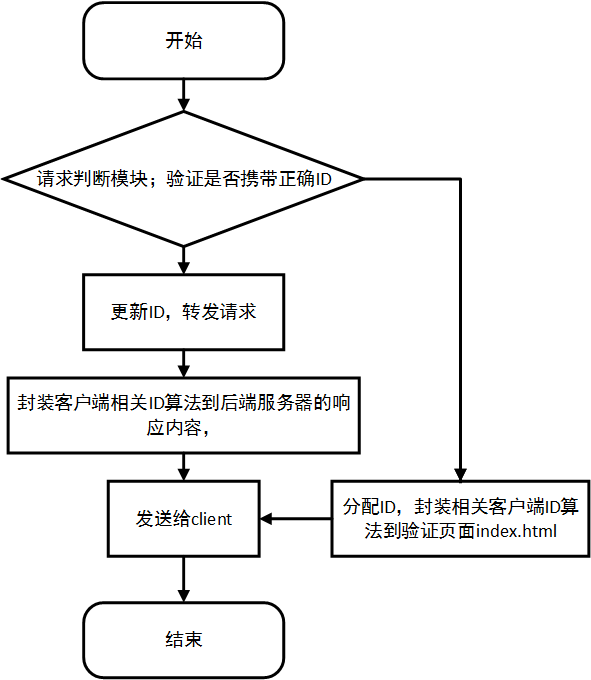


图 4- 4 动态ID模块服务器端处理流程

#### 4.2.2.4代码混淆

这部分的代码实现位置在Nginx第三方模块ngx\_http\_ipslab\_module。使用简单混淆技术进行混淆。

4.3运行效果



图 4- 5 等待用户设置作用网站

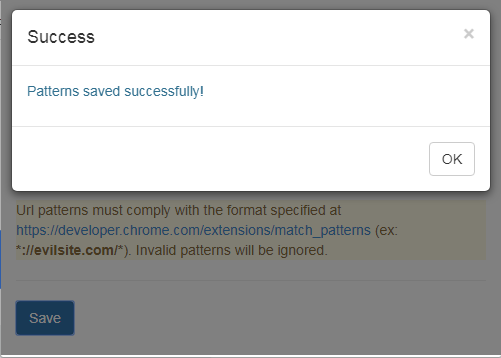


图 4- 6 用户设置成功

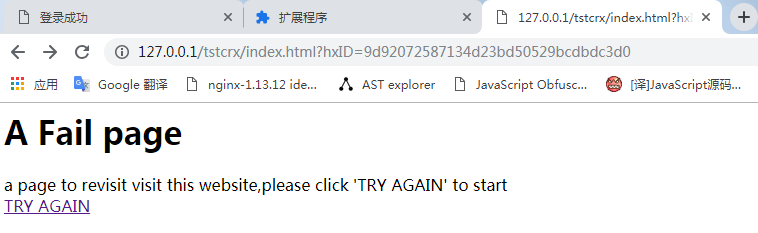


图 4- 7 不合法请求收到的验证页面

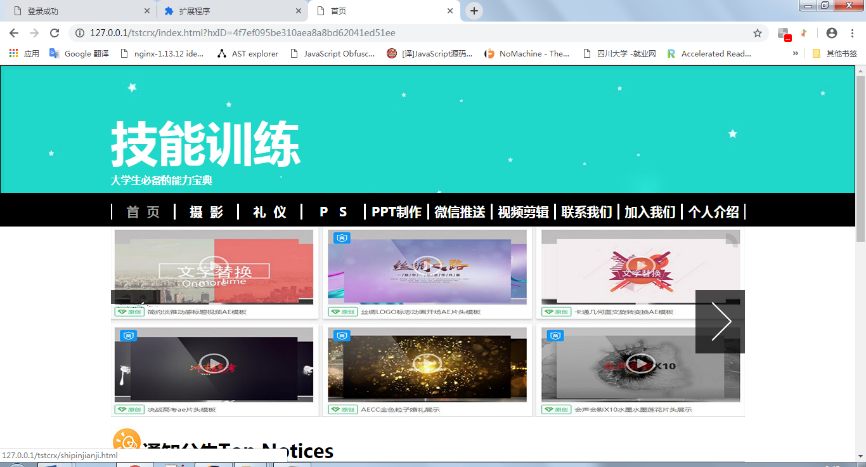


图 4- 8 合法请求得到响应



图 4- 9 从浏览器地址栏看到添加ID成功

4.4小结

第三章描述了基于代码混淆的Web站点动态防御系统的概要的设计，本章对这些设计部分详细揭示了其中的细节，详细的描绘了系统的模块构成，并对每一个模块的版块组成和流程进行了阐述，描绘了其中具体的实现思路和技术，给出了一些板块的运行效果页面。

第五章 系统测试

系统测试可以保证一个软件系统稳定、可靠的运行，对基于代码混淆的Web站点动态防御系统进行系统测试，保证系统满足用户的需求并且有着较高的质量，应该保证系统的bug率在允许的情况内。

5.1 测试环境构建

基于代码混淆的Web站点动态防御系统的测试：

在Windows 7 PC上安装并启动Nginx Windows版本服务器作为目标服务器，配置监听8080端口，部署的网站是“大学生技能训练网站”。这时，目标网站“大学生技能训练网站”，就可以通过”127.0.0.1:8080”访问到。

在Windows 7 PC上安装Cygwin 64 虚拟Linux包环境，通过该环境编译Nginx 1.12.2，编译时将“基于代码混淆的Web站点动态防御系统”开发的ngx\_http\_ipslab\_module模块编译进去。在nginx.conf文件中根据“用户手册进行配置”。

测试时Nginx-1.12.1的nginx.conf需要的配置有：

使用ngx\_http\_ipslab\_module模块自定义指令配置slab共享内存 8\*pagesize(4096)；

配置proxy\_buffer\_size 125K；

配置Keepalive\_time 0；

配置监听80端口；

配置符合”/tmpdir/”的location块代理到目标服务器8080端口。

具体在nginx.conf文件的对应的指令分别如下：

ip\_slab 8;

proxy\_buffer\_size 125k;

keepalive\_timeout 0;

listen 80;

location /tmpdir/{proxy\_pass 'http://127.0.0.1:8080//';}

指令在文件中的具体配置详细参见“用户指导手册”

5.2 测试内容

基于代码混淆的Web站点动态防御系统。采取白盒和黑盒相结合的测试方法。

下面是数据采集模块测试记录：

表5-1 测试01过程记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect”作用网站的设置：成功 | 正确的反向代理服务器网址的正则表达式 | 用户输入正确的反向代理服务器网址的正则表达式 | 设置成功的提示信息 | 设置成功的提示信息 |

表5-2 测试02过程记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect”作用网站的设置：失败 | 错误的反向代理服务器网址的正则表达式 | 用户输入错误的反向代理服务器网址的正则表达式 | 错误输入的提示信息 | 错误输入的提示信息 |

表5-3 测试02过程记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” 捕捉符合网址正则表达式的请求 | 输入”http://127.0.0.1/a/a.html” | 用户在地址栏手动输入请求网址发起请求 | 浏览器扩展“protect”拦截请求，在控制台输出拦截网址 | 在控制台输出拦截网址 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” 在符合网址正则表达式的请求后添加ID | 输入”http://127.0.0.1/a/a.html” | 用户输入请求网址 | 浏览器扩展“protect”拦截请求，在控制台输出添加ID后新的URL | 在控制台输出添加ID后新的URL |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” 重定向 | 输入”http://127.0.0.1/a/a.html” | 用户输入请求网址 | 浏览器扩展“protect”后台重定向请求 | 重定向请求 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” 的ContentScript从响应页面获得ID | 浏览器当前显示从方向代理服务器收到的任何页面 | 收到方向代理服务器的响应页面 | 从页面获得hxID，在浏览器扩展”protect”的ContentScript控制台输出ID查看 | 控制台输出ID查看 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” ContentScript通过Message向background传递ID | 浏览器当前显示从方向代理服务器收到的任何页面 | 从反向代理服务器得到响应页面， | 在浏览器扩展”protect”的背景页控制台查看ID | 背景页控制台输出ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” background中ID通过localStorage存取 | 浏览器当前显示从方向代理服务器收到的任何页面 | 从反向代理服务器得到响应页面，背景页存取ID。 | 输出ID值 | 输出ID值 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID模块中浏览器扩展“protect”测试 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 浏览器扩展”protect” 中ID的MD5加密 | 浏览器当前显示从方向代理服务器收到的任何页面 | 从反向代理服务器得到响应页面 | 输出ID的MD5值 | 输出ID的MD5值 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 给初次访问的IP分配ID | 请求体的IP | 在启动Nginx反向代理服务器后，使用未曾访问过的IP访问反向代理服务器：在地址栏输入网址 | 输出ID的即将要下发给浏览器的，也就是存储在slab中的ID | 输出ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| 给携带正确ID的请求准备下一次使用的ID | 请求体的IP，ID | 在启动Nginx反向代理服务器后，从反向代理的响应页面点击继续访问 | 输出即将要下发给浏览器的，也就是存储在slab中的ID | 输出ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| Slab中(IP,ID)节点的过时淘汰 | 请求体的IP，ID(ID可为NULL) | 多个IP多次访问 | 输出slab中(IP,ID)节点的链表状态，应该根据访问情况发生变化 | 输出slab中(IP,ID)节点的链表状态，根据访问情况发生变化 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| Nginx反向代理服务器向转发的响应HTML页面注入ID | 目标网站的响应包 | 当响应文件类型是”text/html”时，解析HTML，注入ID | 注入了ID的响应页面 | 注入了ID的响应页面 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| Nginx反向代理服务器向转发的响应js文件进行混淆 | 目标网站的响应包 | 当响应文件类型是”application/JavaScript”时，对JavaScript脚本进行混淆操作 | 混淆过的JavaScript | 混淆过的JavaScript |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试编号** | 01 | | | |
| **测试模块** | 动态ID功能模块中反向代理服务器ID管理 | | | |
| **测试环境** | 计算机操作系统：Windows7，浏览器：Chrome | | | |
| **限制条件** | 无 | | | |
| **测试功能** | **输入内容** | **测试过程** | **输出期望** | **实际输出内容** |
| Nginx反向代理服务器向转发的响应js文件进行混淆 | 目标网站的响应包 | 当响应文件类型是”application/JavaScript”时，对JavaScript脚本进行混淆操作 | 混淆过的JavaScript | 混淆过的JavaScript |

5.3 测试结论

经过详细的系统功能的测试，可以判定基于代码混淆的Web站点动态防御系统可以实现预先规定的功能，能够满足用户的需求，并且bug率在可以预见和接受的范围内，系统功能测试通过。

系统在非功能性方面同样满足需求。可用性方面，系统UI设计友好规范，符合软件界面设计原则。

5.4 小结

本章节对于基于代码混淆的Web站点动态防御系统的测试环境、测试方法以及测试结果进行了详细描述，测试结合功能的黑盒测试，具体模块内代码的白盒测试。保证了系统的可用性。经过多次的测试与反馈，提高系统的可用性，增加系统代理网站是的通用性。对于系统的两大模块：动态ID和代码混淆，也就是三小模块：动态ID模块的浏览器插件，动态ID模块的服务器部分和代码混淆模块均通过测试，此外，系统还满足了易用性等非功能性的需求。

第六章 总结和体会

6.1 工作总结

工作目的方面，针对网站的防御，网络安全从事人员在不断和恶意攻击者做着攻防较量，提供安全服务的公司为此做着巨大贡献。目前大多为付费服务，寻找解决方法。本项目为用户广泛的Nginx服务器开发安全模块，将极大便利网站开发人员，针对自动化攻击，在特征匹配的旧方法难以解决时，将反向代理ID验证的解决方案在Nginx上进行开发。

开发技术方面，开发目标是系统的两大模块：动态ID和代码混淆，也就是三个小模块：动态ID模块的浏览器插件，动态ID模块的服务器部分和代码混淆模块。

首先是Nginx的学习，从了解Nginx是什么，Nginx的配置到Nginx模块开发。再进一步学习Nginx已有的模块源码，深入学习，构建自己的实现方案。

其次是开发环境，构建Nginx在Windows操作系统下的开发，Nginx模块开发利于调试的配置到Eclipse如何导入Nginx项目，构建好一个好的开发，测试环境。

再次是应对业务需求，引进了两个开源项目，集成到本项目中。结合业务实际要求以及在非功能性上的要求，选择了Gumbo作为HTML解析器，Duktape作为JavaScript引擎。在此基础上，也考虑到了将来的扩展性，所选择工具Gumbo对于HTML和Duktape对于JavaScript均是强大的。

再次是浏览器扩展的开发。首先，在有限的时间内，选择一个最合适的浏览进行扩展开发。本系统基于一些实际情况选择了Chrome浏览器。其次，了解浏览器扩展的开发只是，从学习官方文档，学习分析已有扩展源码，网上学习他人开发经验，开发满足本系统的浏览器插件。在本系统的浏览器扩展开发中使用AngularJS框架，应用它的MVC框架，数据前后台通讯良好，对用户友好。

过程管理方面，基于代码混淆的Web站点动态防御系统的开发过程，尽量参照软件工程系统方法，尽量严格规范开发流程，尽量及时进行文档记录，遵循需求分析，系统设计，系统实现，系统测试的流程。在进行代码的迭代规范。实际过程中的困难有，项目经验少，代码能力欠缺，源码分析能力不足等等，存在开发时间延长，文档记录不详尽等问题。

开发成果方面，基于代码混淆的Web站点动态防御系统实现了预先设计好的动态ID验证和代码混淆，对于用户使用流畅，对于网站部署人员指导详细，并且系统稳定，功能测试通过，增加容错性，可以满足开发人员的基本需求。

6.2 心得体会

基于代码混淆的Web站点动态防御系统，由本人开发，体验了从对一门领域完全不了解到开发出可用软件，体验到其中的困难，对开发人员的高要求，要有广泛的基础了解，要能选择好需要深入研究的技术点。整个过程中，自己找需要的工具，用手中所拥有的工具找出解决方法。在面对难点，既要有宽泛的了解，有时找出能利用的最简洁的工具，也要有深入研究的劲头，避免将精力花费在无休止的寻找上而浪费成本。这本身就是一个充满探索的过程，其中不可避免地要走许多弯路。

要有迎难而上的勇气，当前行很慢时，要有蜗牛的耐心和毅力，不能放弃。不断进取，才能解决困难。

参考文献

1. 林水明.不透明谓词在JavaScript代码保护的应用研究[D]. 广东工业大学, 2016
2. Netcraft.January 2019 Web Server Survey[EB/OL].(2019-01-24)[2019-04-25].https://news.netcraft.com/archives/2019/01/24/january-2019-web-server-survey.html
3. 马卫局. 网络空间安全进入动态防御时代[J]. 现代军事,2017,07:108-109
4. 邬江兴. 拟态计算与拟态安全防御的原意和愿景[J]. 电信科学, 2014,07:2
5. statcounter GlobalStats.Browser Market Share Worldwide:Mar 2018 - Mar 2019[EB/OL].(2019-04-01)[2019-04-25].http://gs.statcounter.com/
6. JetBrains.The State of Developer Ecosystem in 2018[R/OL].(2019-01-01)[2019-04-25].https://www.jetbrains.com/research/devecosystem-2018/
7. 陶辉.深入理解Nginx-模块开发与架构解析第二版[M].北京：机械工业出版社，2016:6-7

附录1 任务申请书

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **论文（设计）题目** | **基于代码混淆技术的Web站点动态防御系统** | | | | | | | |
| **学 院** | **软件学院** | | **专 业** | | **软件工程** | **年 级** | **2015** | |
| **题目来源**  **（选择其一）** | **教师科研课题** | **纵向课题（　）** | | | **题目类型** | **理论研究（ ）**  **应用研究（ √ ）**  **技术开发（ ）** | | **注：请直接在所属项目括号内打“√”** |
| **横向课题（√）** | | |
| **企事业单位研发项目相关课题 （ ）** | | | | **是否在实验、实习、工程实践和社会调查等社会实践中完成** | **是（ √ ） 否（ ）** | |
| **教师自拟课题（　）** | | | |
| **学生自拟课题（　）** | | | |
| **论文（设计）选题目的、工作任务：**  选题目的：  随着自动化漏洞扫描、撞库扫号、恶意爬虫、暴力破解、薅羊毛等自动化攻击技术的快速发展，传统基于特征和规则的入侵检测系统、Web应用防火墙（WAF）已经无法有效地检测和拦截这些攻击。这些安全问题可能会导致数据库泄露，网页被篡改等，影响正常业务。  本项目拟通过随机化代码混淆技术和动态ID验证技术，对目标站点页面关键元素进行混淆加密，对客户请求进行人机验证，实现对网站的动态防御，有效降低自动化攻击技术的安全威胁。网站运营商家运用此技术可以防止恶意爬虫、薅羊毛等自动化攻击，不仅可以减少因访问量降低和销量、广告收益降低的恶性循环导致的经济损失，还保障了用户信息安全。  毕业设计目标：  对于目前不足的自动化攻击的防御，提取了网站防御中的“机器人”攻击问题。脱取这部分功能开发成nginx模块。该模块具有高可用，轻量的特性，增加了应用的灵活性。将很大程度上提高Web站点的安全防护水平。同时，对本项目在客户端的代码和网页代码做了一定保护。项目介绍如下：  1.利用Nginx平台的反向代理服务功能, 在此节点，运用人机识别和动态ID匹配算法进行人机识别，拦截恶意爬虫等自动化攻击，动态防护原网站的安全；  2.利用随机代码混淆技术对1中人机识别客户端代码以及目标站点响应请求的返回内容进行混淆封装处理，使攻击者即便拿到网站的页面信息，也难以阅读网页中的JS代码。  上述解决方案要求的具体功能点如下：  1. 浏览器判断：代理节点处通过判断请求方是否使用正确、真实的浏览器，来判断用户是“人”还是“机器”。  2. 动态ID验证：一般情况，nginx实施以下的保护模式：在 nginx处登记并派发一次性的令牌给访问者作为下一次验证通过的凭据；在每一次与用户交互过程中将某些信息附属在用户请求的内容上发给client，来维护这个保护模式。另外，如果用户发来的请求不符合或者不通过验证代码，给用户发送特殊的验证页面来进入nginx实施网站保护的模式。  3. 代码保护：该保护模式的一部分代码暴露在用户端。因此，需要代码保护技术加大网页内容分析的难度。代码混淆是有效的代码保护技术。  毕业设计工作内容：   1. 文献资料阅读； 2. 现有系统、软件、算法等分析； 3. 编写开题报告和开题报告答辩； 4. Ninx模块开发，JavaScript和HTML学习； 5. 按照软件工程方法进行系统开发； 6. 选取并实现有效的代码混淆，ID验证网页内容保护方法； 7. 开发文档的整理； 8. 毕业设计论文的编写、修改； 9. 编写答辩提纲（PPT）； 10. 毕业论文答辩；   要求完成的毕业设计成果，需要提交的文档、软件：   1. 打印版和电子版的毕业论文和文献翻译； 2. 打印版和电子版的毕业设计套表；    1. 《四川大学本科毕业论文（设计）任务书》    2. 《四川大学本科毕业论文（设计）开题报告》 3. 电子版的软件软件开发文档；    1. 需求规格说明书    2. 概要设计文档    3. 详细设计文档    4. 测试文档    5. 用户文档 4. 电子版的软件源代码； 5. 电子版的可安装、可运行的软件 | | | | | | | | |
| **目前资料收集情况（含指定参考资料）：**  1. <https://www.ibm.com/developerworks/cn/web/wa-lo-jshttp/index.html>  2. 深入理解Nginx模块开发与架构解析 第2版  3. 《基于WEB反向代理的动态混淆方法》  4. 《代码混淆模型研究》 杨宇波 | | | | | | | | |
| **论文（设计）完成计划（含时间进度）：**  （1）2018年11月12日-2018年12月初：完成反向代理并ID验证服务器端模块；  （2）2018年12月初-2019年1月底：完成反向代理并ID验证服务器端模块；  （3）2019年1月底-2019年3月初：完成代码保护；  （4）2019年3月初-2019年4月初：完善，优化细节；  （5）2019年4月初-2019年4月末：完善毕业论文，准备论文答辩。  （6）2019年5月：毕业论文答辩。 | | | | | | | | |
| **接受任务日2018年 10 月 19 日** | | | | **预计完成日期： 2019 年 4 月 27 日** | | | | |
| **学生接受任务（签名）：邵欣欣** | | | |
| **指 导 教 师 （签名）：** | | | | **学院（系）负责人审定（签名）：** | | | | |

附录2 可行性研究报告书

基于代码混淆的Web站点动态防御系统

可行性研究报告

拟 制 人\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二日]

### 1. 引言

1.1 编写目的

本报告旨在环境、数据和功能等方面系统分析基于代码混淆的Web站点动态防御系统设计与实现的可行性，明确了项目开发目标、规划和处理流程，指出了目前该项目存在的缺陷与不足，并根据实际情况进行分析，预期针对的读者为基于代码混淆的Web站点动态防御系统设计与实现的开发、设计、测试、文档编写和评审人员。

1.2 背景

a. 基于代码混淆的Web站点动态防御系统

b. 本项目的任务提出者、开发者:邵欣欣；需要部署网站的技术人员、运维人员；

c.当访问量大量到一定阈值，需要开发外部数据库来存储。

1.3 定义

Chrome Extension：Chrome浏览器的扩展，扩展浏览器的功能，例如：捕捉特定网页的内容，捕捉HTTP报文，捕捉用户浏览动作，改变浏览器地址栏/起始页/书签/Tab等界面元素的行为，与别的站点通信，修改网页内容等。

HTML：Hyper Text Markup Language的缩写，超文本标记语言。HTML 不是一种编程语言，而是制作网页必需的标记语言。“超文本”指页面可以包含图片，链接，音乐和程序等非文字元素。

JavaScript：JavaScript是因特网上最流行的脚本语言，一种直译式脚本语言。

Gumbo：一个C语言编写的开源的HTML解析器。

Duktape：一个C语言编写的可以嵌入C/C++工程的JavaScript引擎。

反向代理服务器：反向代理服务器是服务器的一种，它连接一组或多组后端服务器。用户只知道反向代理服务器的IP地址，对于用户请求，由反向代理服务器向后端服务器进行请求，在发送给用户。它与前向代理的明显区别是，前向代理作为客户端的代理，可能会将资源发送给一个或多个客户端。

AngularJS：是一个JavaScript框架，使用MVC 架构在HTML与JavaScript见提供良好用户体验。

slab共享内存：Nginx的设计架构是一个多进程的程序，收到的请求分布在不同的进程上，当请求的完成需要各进程相互配合时，进程间需要通信，但我们知道，进程间通信开发往往会很困难，为此，Nginx实现了一套共享内存管理机制-slab。

红黑树：一种数据结构，自平衡二叉查找树。

subrequest:Nginx模块中处理子请求的模块。

DOM：Document Object Model，文档对象模型。简单来说，就是一个HTML文件对应结构的数据结构。DOM是表示和处理一个HTML或XML文档的常用方法。是以面向对象的方式描述文档的模型。每一个标签都是树上的一个节点。

AST：Abstract Syntax Tree，抽象语法树，简称语法树，是源代码语法结构的抽象表示。它以树的形式表示编程语言的语法结构，分析源代码更直观，更形象，更贴近人的感性认知。

代码混淆：本系统中涉及到的代码混淆均不涉及到加密解密，对于人来说，通过改变形式让脚本代码变得晦涩难懂，但对于机器来说，并没有实质不同。

MD5：MD5 Message-Digest Algorithm，MD5消息摘要算法。MD5将文本信息，通过其不可逆的字符串变换算法，产生了这个唯一的MD5信息。

1.4 参考资料

[1] 马蔚,郑霖,何久. 动态变换技术防御 Ｓtｒｕｔｓ２ Ｓ2 －０３２ 零日攻击研究[J].信息安全研究,2016,08:748

### 2. 可行性研究的前提

2.1 要求

本系统要实现动态ID机制和代码混淆处理，要求系统对用户友好，本系统目标于安全方向，但从业务方面考虑，在安全之前提，务必保证对被代理网站的业务没有影响，对用户的正常使用没有造成困扰，保证良好的用户体验才是基础。

2.2 目标

一个可以添加到反向代理服务器上的模块，目标是过滤掉一些自动化请求。需要注明的是，这里的反向代理服务器具体指Nginx。

2.3 条件、假定和限制

条件：

必须配置Nginx对上游响应的缓存完全容纳可能出现的最大的文本文件。请估计好网站涉及到的HTML文件或JavaScript文件的最大大小。

保证请求的及时结束，本系统中要求keepalive为off。

2.4 进行可行性研究的方法

调研目前市场上的网站防御系统，使用原理，目前网站防御普及程度，防范程度。以及，目前自动化攻击常用形式。

从目前专利论文情况了解前沿技术。确定我们的技术路线。

宽泛了解技术难以程度，结合自身能力进行评估，进一步加强可行性研究。

2.5 评价尺度

系统的评价主要从下面几个方面来评估

用户使用难易程度，这是评估系统易用性的重要手段；

能通畅代理并处理的网站的等级，根据网站使用技术的复杂程度，本系统能够解决的网站等级是评估系统成熟度的尺度。

拦截自动化攻击的成功率是本系统追求的重要尺度。

### 3. 对现有系统的分析

3.1 处理流程和数据流程

现有系统的网站防御解决方法，通过在网关部署监控设备，监控流量，根据以往经验，常规特征，根据规则 来发现异常，在进行预警，提醒管理员可能出现的问题。这种解决方案，仍需人工的大量介入，决策依然掌握在网站监控人员的手里，需要网站监控人员倾注心力，密切关注。

另外应对更具体，针对性更强，目的极其明确，持续性强的，与黑客斗智斗勇的攻击，目前在市场前沿已经有动态防御，蜜罐等技术来着重解决这些问题。

因此，借鉴更先进的动态主动防御的思想，应用于低端，常见的自动化攻击将是一大提高。

3.2 工作负荷

流量监控

规则匹配

预警系统

3.3 费用开支

外包的安全业务费用。

现有的网站防御，需要向提供安全服务的公司进行安全服务的购买。

3.4 人员

网站运维人员

3.5 设备

防火墙，流量监控设备

3.6 局限性

对于服务比较有规模的网站，购买安全服务公司服务是比较划算的。对于小规模的网站，如果购买安全服务，将会大比例提高成本，对本身业务造成影响。

### 4. 所建议的系统

4.1 对所建议系统的说明

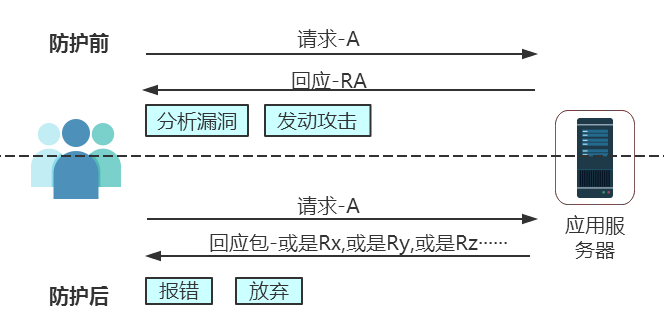


图1- 1 网络攻击的前提

一般网站直接提供服务，所有资源文件都对用户开放。基于目前网页开发架构和开发语言，网页应用最主要的一个特点是：客户端对服务器进行的请求和访问，返回页面在客户端网页代码中可以看到大量除屏幕展现的信息之外的参数名和参数信息．而获得这些信息正是网页攻击的最重要的一步．

4.2 处理流程和数据流程。



图1- 2 Nginx反向代理

4.3 改进之处

不需要实施监督

将不必要的请求带来的的压力加诸反向代理服务器上。

4.4 影响

对现有设备的影响是需要添加反向代理功能。

4.4.1.对设备的影响

使用Nginx服务器进行反向代理

4.4.2.对软件的影响

两者之间没有交集

4.4.3.对用户单位机构的影响

通过了解Nginx，并使用本模块，减轻开发人员在安全方面的额外设计，并且，作为开源免费，使用方便，作为其中一个模块，灵活易用。

4.4.4.对系统运行过程的影响

本系统理想目标对网站与用户之间的业务没有任何影响。

4.4.5.对开发的影响

本系统理想目标对开发没有任何影响。

4.4.6.对地点和设施的影响

需要部署额外的反向代理服务器。

4.4.7.对经费开支的影响

经费支出用于反向代理服务器相关软硬件后期维护。

4.5 技术条件方面的可能性

Nginx模块开发技术，对HTML的改造技术，对JavaScript的改造技术。

### 5. 可选择的其他系统方案

5.1 可选择的系统方案1

另一种预想的解决方案是完全通过在Nginx反向代理处提前拿到HTML，JavaScript等文件，通过解析源代码，再通过AOP技术，即通过JavaScript的函数覆盖等技术，完完全全在服务器处改造代码吗，让代码可以做到自己“捕捉所有HTTP请求”来实现动态ID在浏览器端的配合部分。

如果可以实现，那么对使用浏览器访问网站的用户真正做到了透明，将是极好的。

但在技术实现上遇到问题。需要对JavaScript的精通，在目前的开发周期内，还未能实现。

### 6. 投资及效益分析

6.1 支出

部署反向代理的支出

6.1.1 基本建设投资

按指导手册安装即可

6.1.2 其他一次性支出 ：无

6.1.3 非一次性支出

关注本系统优化升级。

6.2 收益

向安全公司购买安全服务的费用。硬件设备磨损费用

6.2.1 一次性收益]

专职网站监控人员的成本费

6.2.2 非一次性收益

不可估量

6.2.3 不可定量的收益

对网站工作人员的减负

对目标网站的负载减轻

6.3 收益/投资比

接近80%

6.4 投资回收周期

一周

6.5 敏感性分析

本项目目标实现基本动态ID机制和代码混淆，目前还未测试较复杂JavaScript技术的使用效果。

### 7. 社会因素方面的可能性

7.1.Nginx开源，大量第三方模块是其珍贵资源，本项目同属于其开源项目的一个第三方模块开发。

7.2.浏览器用户涉及到的操作简单，通过指导文件可以方便使用。对网站开发人员的要求是了解Nginx，并按照本系统用户手册操作即可。

### 8. 结论

本项目的开发从技术，市场方面均具有可行性。本项目的开发对网站防御有积极作用。

附录3 项目开发计划书

基于代码混淆的Web站点动态防御系统

项目开发计划的编写

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二十六日]

### 1. 引言

1.1 编写目的

本文档对应项目是“基于代码混淆的Web站点动态防御系统”。本文档预期读者是项目经历，项目开发人员以及有意了解项目进度的人员。

1.2 背景

1. 基于代码混淆的Web站点动态防御系统；
2. 任务提出者：邵欣欣；开发者：邵欣欣；用户：网站安全运维人员，访问网站的用户；实现该软件的计算机网络：处于反向代理节点的网络；
3. 本系统从安装位置上来看可以分成两部分，一部分安装在是访问网站的浏览器上；一部分安装在反向代理服务器Nginx中。

1.3 定义

Chrome Extension：Chrome浏览器扩展

Cygwin: Cygwin是一个在windows平台上运行的类UNIX模拟环境，是cygnus solutions公司开发的自由软件（该公司开发的著名工具还有eCos，不过现已被Redhat收购）。它对于学习UNIX/Linux操作环境，或者从UNIX到Windows的应用程序移植，或者进行某些特殊的开发工作，尤其是使用GNU工具集在Windows上进行嵌入式系统开发，非常有用。随着嵌入式系统开发在国内日渐流行，越来越多的开发者对Cygwin产生了兴趣。[1]

Eclipse-cdt: (C/C++开发环境)CDT项目提供了基于Eclipse平台的功能齐全的C和C ++集成开发环境。 功能包括：支持项目创建和各种工具链的托管构建，标准构建，源导航，各种源知识工具，如类型层次结构，调用图，包含浏览器，宏定义浏览器，具有语法突出显示的代码编辑器，折叠和超链接 导航，源代码重构和代码生成，可视化调试工具，包括内存，寄存器和反汇编查看器。

1.4 参考资料

[1]Eclipse.Eclipse CDT :C/C++ Development Tooling [EB/OL] . [2019-04-26] . http://www.eclipse.org/cdt/

### 2. 项目概述

2.1 工作内容

背景调查：了解目前自动化攻击的现状，包括攻击手段，以及在网络攻击中的占比和目前应对自动化攻击的手段。

市场调查：鉴于我们的系统开发依赖于特定的已存在软件： Nginx和Chrome，了解目前Nginx和Chrome在各自领域的市场占比。保证我们开发的系统的应用前景的限制是比较小的。

需求分析；

系统设计；

系统实现；

系统测试。

2.2 主要参加人员

指导老师：王海舟；开发人员：邵欣欣。

2.3 产品

2.3.1 程序

[列出需移交给用户的程序的名称、所用的编程语言及存储程序的媒体形式，并通过引用有关文件。逐项说明其功能和能力。]

产品：基于代码混淆的Web站点动态防御系统

“hx\_protect”:一个Chrome浏览器扩展；使用HTML，CSS，JavaScript编写。

”ngx\_http\_ipslab\_module”一个可以安装到Nginx上的模块。使用C编写。

2.3.2.文件

《用户手册概要》：该用户手册主要包含对于浏览器用户，对于网站运维人员的系统使用环境，安装指导，配置指导，以及会出现的限制，可实现的功能等。

2.3.3.服务

代码混淆；

自动化请求拦截。

2.3.4.非移交的产品

《可行性与计划阶段--可行性研究报告》

《可行性与计划阶段--项目开发计划》

《需求分析阶段--需求说明书》

《概要设计阶段--概要设计说明书》

《概要设计阶段--数据库设计说明书》

《详细设计阶段--详细设计说明书》

《单元测试阶段--单元测试报告》

2.4 验收标准

|  |  |
| --- | --- |
| **产品服务** | **验收标准** |
| 《任务申请》 | 清晰而有条理的描述概括出整个项目的功能 |
| 《可行性与计划阶段--可行性研究报告》 | 全面而清晰地说明该软件开发项目的实现在技术上、经济上和社会因素上的可行性，评述为了合理地达到开发目标可供选择的各种可能实施方案，说明并论证所选定实施方案的理由。 |
| 《可行性与计划阶段--项目开发计划》 | 详细而全面的为软件项目实施方案制订出具体计划，应该包括各部分工作的负责人员、开发的进度、开发经费的预算、所需的硬件及软件资源等。 |
| 《需求分析阶段--数据要求说明书》 | 清晰完整的向整个软件开发时期提供关于被处理数据的描述和数据采集要求的技术信息，包括数据的逻辑描述、数据的采集等。 |
| 《需求分析阶段--需求说明书》 | 详细而清晰的对所开发软件的功能、性能、用户界面及运行环境等作出详细的说明。它是在用户与开发人员双方对软件需求取得共同理解并达成协议的条件下编写的，也是实施开发工作的基础。该说明书应给出数据逻辑和数据采集的各项要求，为生成和维护系统数据文件做好准备。 |
| 《概要设计阶段--概要设计说明书》 | 概要设计说明书是概要设计阶段的工作总结。主要包括功能分配、模块划分、程序总体结构、输入输出以及接口设计、运行设计、数据结构设计和出错处理等，为详细设计作好准备。 |
| 《概要设计阶段--组装测试计划》 | 是否提供了一个对该软件的测试计划，包括对每项测试活动的内容、进度安排、设计考虑、测试数据的整体性方法及评价准则 |
| 《概要设计阶段--数据库设计说明书》 | 是否从数据库外部设计、ER图、详细表设计、数据字典设计和运行设计等部分清晰地产阐述了数据库设计的详细过程已经如何同其他系统的交互。 |
| 《详细设计阶段--详细设计说明书》 | 清晰地描述每一模块是怎样实现的，包括实现算法、逻辑流程等。 |
| 《实现阶段--模块开发说明》 | 扼要说明系统中所有模块的功能,主要是输入、要求的处理、输出。可以从系统设计说明书中摘录。同时列出在需求说明书中对这些功能的说明的章、条、款。 |
| 《单元测试阶段--单元测试报告》 | 单元测试人员根据单元测试计划对已完成的系统单元进行测试，确保已完成的系统单元符合相应部分系统详细设计说明书所规定的要求。如果单元测试发现系统单元与其相应的详细设计说明书不符，则此系统单元必须修改以最终符合说明书的规定。且满足：  1.测试用例不通过数的比例<3%;  2.不存在错误等级为1的错误;  3.不存在错误等级为2的错误;  4.错误等级为3的错误数量≤20;  5.所有提交的错误都已得到更正。 |

2.5完成项目的最迟期限

2019年4月26日

2.6 本计划的批准者和批准日期

2017年12月

### 3. 实施计划

3.1 工作任务的分解与人员分工

本项目开发中需完成的各项工作，从需求分析、设计、实现、测试直到维护，包括文件的编制、审批、打印、分发工作，用户培训工作，软件安装工作等，均由邵欣欣完成。

3.2 接口人员

无

3.3 进度

2018.11.12 完成开题报告

2018.11.13-2018.11.30 资料搜集和阅读

2018.12.1-2018.12.30 需求分析，确定论文大纲

2018.1.1-2019.3.10 设计及代码编写

2019.3.11-2019.3.20 测试及总结

2019.3.21-2019.4.5 完成论文初稿，系统后期维护

2019.4.6-2019.4.20 修改论文

2019.4.20 论文定稿，准备答辩资料

3.4 预算

无

3.5 关键问题

技术难点：模块在实现拦截自动化请求的同时要对被反向代理的网站提供的正常业务的支持达到100%，可以有在操作上的冗余，但必须保证网站业务的完整性。

风险：对网站的业务出现损害。

### 4.支持条件

开发支持：Windows+Cygwin+Eclipse-cdt;在Windows操作系统安装Cygwin软件，在Cygwin中下载gcc,perl等必需包后，下载nginx-1.12.1源码，在Cygwin中解压，编译。在完成“从源代码到可运行软件”的nginx这一安装过程后，已经生成nginx服务器。将整个项目作为“Makefile工程”导入Eclipse-cdt进行开发。

4.1 计算机系统支持

计算机：PC

编译程序:Cygwin管理的gcc

操作系统：Windows

测试支持能力:Chrome 浏览器

其他软件：Chrome浏览器，Notepad++

4.2 需由用户承担的工作

在使用阶段，用户需要提供希望拦截的请求网址的网站的正则表达式。

4.3 需由外单位提供的条件

无

### 5.专题计划要点

拦截自动化请求：使用有效手段进行拦截，改善拦截效果。

代码保护：代码保护效果有效

附录4 数据要求说明书

基于代码混淆的Web站点动态防御系统

数据要求说明书

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二十六日]

### 1．引言

1.1编写目的

本数据要求说明书呈现项目中所涉及到的重要数据，预期读者包括本系统的使用人员和本系统项目的开发人员。

本系统的使用人员可以根据本文档掌握使用本系统过程中的数据要求，并大致了解数据背后的实现来更好地理解使用的数据。项目开发人员以本文档数据为标准进行开发。

1.2背景

1. a.任务：基于代码混淆的Web站点动态防御系统

b.提出者：邵欣欣；开发者：邵欣欣；用户：网站部署、管理人员；运行该项软件对象：网站的代理服务器即Nginx和浏览器。

1.3定义

无

1.4参考资料

无

### 2．数据的逻辑描述

对数据进行逻辑描述时可把数据分为动态数据和静态数据。

2.1静态数据

列出所有作为控制或参考用的静态数据元素。

ID长度

2.2动态输入数据

Nginx反向代理需要的配置数据：

Slab的大小

代理的网站

upstream可以缓冲的包的大小

Chrome Extension 捕捉并重定向需要的数据：

pattern 拦截的网址的通配表示，类似” \*://127.0.0.1/\*”；

hxID 用来动态生成新ID的旧ID。

2.3动态输出数据

Nginx模块输出的数据：

ID 即将注入响应发送给浏览器端的ID

Chrome Extension 输出的数据：

hxID 生成的新ID

2.4内部生成数据

列出向用户或开发单位中的维护调试人员提供的内部生成数据。

IP 获取的用户请求的IP，作为用户身份验证的key

Client\_ID 从Nginx接收到的Request中提取的，浏览器发送过来的ID

Slab\_ID slab中存储的ID，也是上一次响应中发送给浏览器端的ID

Last\_slab\_ID slab中存储的上一次注入到响应页面发送给浏览器端的ID

Server\_ID 从slab中取出的ID

2.5数据约定

ID 长度限制32位，使用ngx\_str\_t结构体时，用32大小；使用char\* 或u\_char\* 分配时使用33大小。

### 3．数据的采集

3.1要求和范围

承担者：网站安全运维人员

可用共享内存大小

可用转发上游相应的缓存大小

涉及网站网址的正则表达式

3.2输入的承担者

动态ID模块数据来源于两类用户：浏览器用户和网站部署人员。

代码混淆模块数据来源于开发者

3.3预期处理

动态ID模块中

来源于用户的输入数据要进行有效性验证。

ID数据维持字符串形态。IP数据使用int型状态。

代码混淆模块中输入数据要进行预编译。

3.4影响

Linux下C开发的相关数据结构。

附录5 需求说明书

基于代码混淆的Web站点动态防御系统

需求说明书

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年四月十日]

### 1．引言

1.1编写的目的

本项目将开发一个基于代码混淆的Web站点动态防御系统。

本“需求规格说明书”主要用于：明确项目背景和任务、分析项目功能需求和非功能需求、并选择合适的分析模型对项目需求进行更详细、更有逻辑的说明等，为团队成员在后期项目开发过程中提供指导性文档，以达到尽可能在更短时间内花费更小代价下完成项目的目的。其次，本文档还有利于为非本团队的开发者提供参考资料，有利于提高项目的关注度、拓展项目成果的推广。

本文档供项目经理、设计人员、开发人员、文档编写人员参考。

1.2背景

1. 待开发的系统的名称：基于代码混淆的Web站点动态防御系统
2. 本项目的任务提出者、开发者、用户；

任务提出者：指导老师王海舟

开发者：邵欣欣

用户：Web网站持有者、管理者

1. 该系统同其他系统或其他机构的基本的相互来往关系。

本项目的所有操作都基于反向代理服务器，且所使用的开发平台是Nginx。通过Nginx分别同实际服务器或客户端联系。

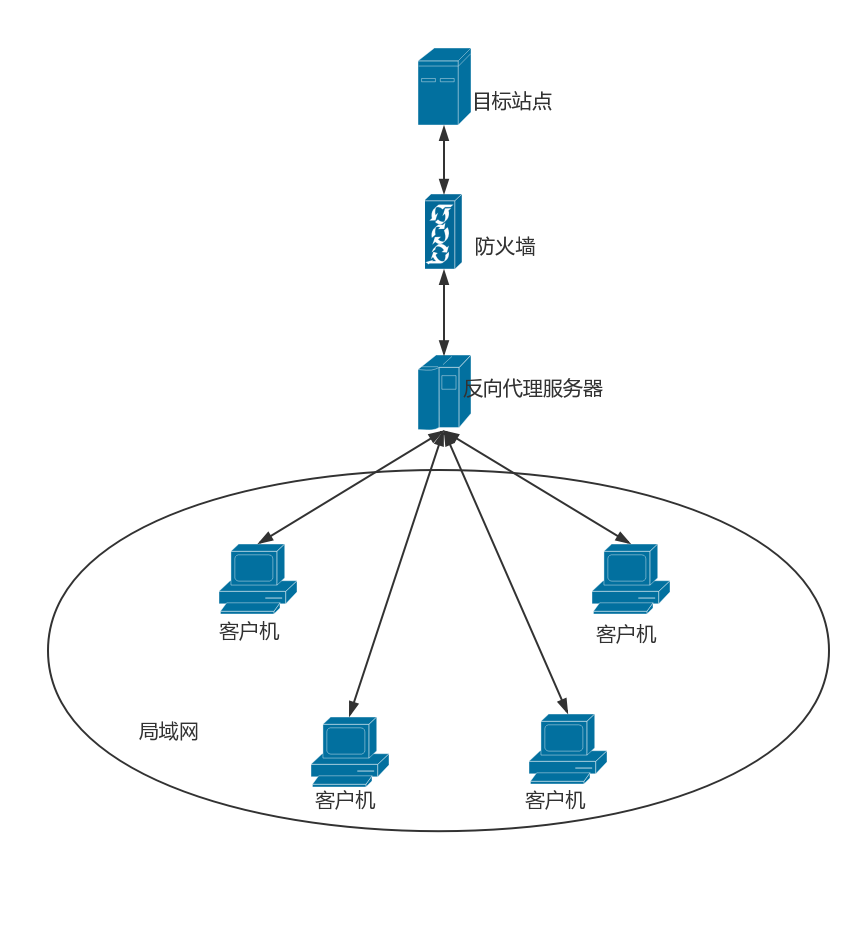


图5- 1反向代理服务器

1.3定义

反向代理服务器：反向代理（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器；并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器，即反向代理服务器。

动态ID：本项目所要实现算法的重要标识元素。

代码混淆：代码混淆(Obfuscated code)是将计算机程序的代码，转换成一种功能上等价，但是难于阅读和理解的形式的行为。混淆后的代码很难被反编译，即使反编译成功也很难得出程序的真正语义。

1.4参考资料

### 2．任务概述

2.1目标

2.1.1 项目的背景

最初，新技术的使用通常以提供服务为着眼点，而忽略安全性，没有考虑用户的不可信。目前，网站的使用面临很多安全威胁。例如政府服务类网站，需要向用户提供查询服务，但是一旦遭遇爬虫，很容易被攻陷，或者消耗资源，或者被发现网站漏洞;电信运营商的网上营业厅，银行、甚至很多企业的网上在线业务系统，经常会被机器人通过模拟人的行为发动攻击。这些非“人”行为，极易得手的最根本原因是，面对看似完全正常的业务逻辑操作，现有安全手段却乏力于如何做出甄别，而这些威胁或者攻击一旦成功，即可抢占各类资源，进而可冒用身份，进行欺诈交易，甚至获得大量的客户信息、企业信息，造成大规模信息泄露。比如：针对一些商城的兑换服务，进行篡改积分，一些航空公司每每放出资源，几秒钟内就会被黄牛党抢光，抢占低价票资源。

当前市场上安全防护产品大多都是基于规则，产品总体偏重于匹配特征。显然，这样不停地找漏洞、打补丁令企业置身于千变万化攻击的被动中。特别是随着互联网+在国内的大力发展，像自动化漏洞扫描、撞库扫号、恶意爬虫、暴力破解、薅羊毛等自动化攻击技术的发展速度丝毫不亚于企业新业务的发展，安全手段的乏力始终是企业IT的一个难题。传统基于特征和规则的入侵检测系统、Web应用防火墙（WAF）等已经无法有效地检测和拦截这些攻击，网站的防护急需一种新型的方式来应对。

同时，我们知道，Javascript脚本是一种可读性较高的解释性语言，通常直接嵌入到网页文件中，一般用户都可以使用浏览器获得源码，对攻击者来说可轻易地获取源码进行逆向分析和恶意利用，这对承载WebWeb应用重要核心业务流程信息的Javascript安全性带来巨大威胁。[1]

鉴于此，本项目拟通过动态ID匹配技术对客户端的请求进行人机识别，以及通过随机化代码混淆技术对目标站点页面关键元素进行封装处理后再发送给用户，实现对网站的动态防御，有效降低自动化攻击技术的安全威胁，进一步防护网站的安全。

2.1.2 项目意义

1.利用Nginx平台搭建反向代理服务器，提供一道攻击缓冲区

反向代理服务器会强制将外部网络对要代理的服务器的访问经过它，这样反向代理服务器负责接收客户端的请求，然后到目标服务器上获取内容，把内容返回给用户，此时反向代理服务器对外就表现为一个服务器，从而使目标服务器对外“隐藏”起来，客户端无法直接访问。我们利用这样的功能，在反向代理服务器上加载我们的防护系统，拦截攻击。Nginx的并发能力在同类型的网页服务器中表现较好，因此，其多用来搭建反向代理服务器。同时，这样高并发高性能的特性使得我们的系统对用户体验感的影响最小化。

中国大陆使用Nginx网站有：百度、新浪、网易、腾讯、淘宝等。

2.利用动态ID匹配算法进行人机识别，防御自动化攻击

反向代理服务器的缓冲机制通过降低向WEB服务器的请求数从而降低了WEB服务器的负载。防御很有限，对目前新兴的许多自动化技术达不到防御的效果。因此，本项目提出了使用动态ID匹配算法来判断请求网站的操作是否是“人”的操作。

当用户请求访问时，进行动态ID匹配流程，若匹配成功，则反向代理服务器向目标服务器发出请求并接收网站文件，随后将随机化混淆的网站文件和扩展到客户端进行动态ID验证的相关文件进行封装后发送给用户；若匹配失败，则反向代理服务器将对请求发出者以自动化工具对待。

3.通过随机化代码混淆技术混淆关键代码使攻击者即便拿到源代码也无法破解动态防御算法和网站关键信息

代码混淆(Obfuscated code)是将计算机程序的代码，转换成一种功能上等价，但是难于阅读和理解的形式的行为。混淆后的代码很难被反编译，即使反编译成功也很难得出程序的真正语义。

随机化的混淆使用多种方法，减少可重复性，增加源码的分析难度。

反向代理服务器向目标服务器发出请求并接收网站文件，随后将随机化混淆后的网站文件发送给用户。混淆后的文件不需要还原即可在客户端运行，这样可以让攻击者很难得出网页文件的关键信息。同样，在浏览器上辅助运行的动态ID匹配算法也经过随机化混淆，使攻击者即便拿到源代码也无法破解动态防御算法。

对于用户端，每次拿到的东西不尽相同 ，每次请求的URL串有变化，用户端可窥探的服务器相关呈现动态性，从而实现对网站的动态防御。

2.2用户的特点

本系统的最终用户是WEB站点的持有者、管理者。

操作人员、维护人员皆有一定水平的计算机基础技能，这些人员也多出自计算机相关专业，技术性水平较高。

本系统的一经投入使用，将持续运行，直到下一次系统更新或系统重启等。

2.3假定和约束

1.假定

本系统假定Web网站持有者与管理者想要防护自身网站安全。

本系统假定用户对要访问的网站不介意多一个下载插件的步骤。

2.约束

本系统约束开发语言为C与JavaScript。

### 3．需求规定

3.1对功能的规定

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 输入 | 处理 | 输出 |
| 客户端动态ID生成算法 | 用户请求链接 | 对链接加上ID处理 | 改装后的请求链接 |
| 服务器端动态ID匹配算法 | 改装后的请求链接 | 提取ID进行匹配 | 匹配结果 |
| JavaScript代码混淆 | 用户请求页面中携带的JavaScript文件 | 对JavaScript文件进行混淆 | 混淆后的JavaScript文件 |

3.2 对性能的规定

3.2.1精度

|  |  |  |
| --- | --- | --- |
| 功能 | 输入 | 输出 |
| 客户端动态ID生成算法 | 完整、正确的用户请求链接 | 完整、正确的改装后的请求链接 |
| 服务器端动态ID匹配算法 | 完整、正确的改装后的请求链接 | 匹配结果 |
| JavaScript代码混淆 | 用户请求页面中携带的JavaScript文件字符串流 | 混淆后的JavaScript文件字符串流 |

3.2.2时间特性要求

要求最小化对用户的体验感的影响。其他可由Nginx的高性能特性补充。

3.2.3灵活性

当需求需要增添新的功能模块时，该系统可在第三方模块中继续加入新的功能模块。

3.3输入输出要求

|  |  |  |
| --- | --- | --- |
| 功能 | 输入 | 输出 |
| 客户端动态ID生成算法 | 完整、正确的用户请求链接 | 完整、正确的改装后的请求链接 |
| 服务器端动态ID匹配算法 | 完整、正确的改装后的请求链接 | 匹配结果 |
| JavaScript代码混淆 | 用户请求页面中携带的JavaScript文件字符串流 | 混淆后的JavaScript文件字符串流 |

**范例1：客户端动态ID生成算法**

输入：http://fdinfo.scu.edu.cn/WFManager/login.jsp

输出：http://fdinfo.scu.edu.cn/WFManager/login.jsp\*\*\*ID

说明：\*\*\*ID表示添加的ID元素

**范例2：服务器端动态ID匹配算法**

输入：http://fdinfo.scu.edu.cn/WFManager/login.jsp\*\*\*ID

输出：人机验证页面或客户端所请求的经过混淆后完整的文件，包括html/css/javascript。

**范例3：JavaScript代码混淆**

输入：用户请求页面中携带的JavaScript文件字符串流

输出：混淆后的JavaScript文件字符串流

3.4数据管理能力要求（针对软件系统）

能把下发给客户端的ID和客户端本身的IP对进行存储、查询、修改。

数据规模的大小依网站访问的规模而定。

3.5故障处理要求

系统运行时间长需要重启：Nginx提供了一整套系统重启的解决方案，对使用影响很小。

系统ID有可能被截取，通过隐藏ID的真实功能，或定期更新ID生成算法实现。

3.6其他专门要求

Nginx的并发能力在同类型的网页服务器中表现较好，因此，其多用来搭建反向代理服务器。中国大陆使用Nginx网站有：百度、新浪、网易、腾讯、淘宝等。目前新兴的网站也多用Nginx进行部署。

因此，若使用单位有其他的功能需求，可以非常方便地向其中添加具有新的功能的第三方模块。可补充性强。

同时，在成熟的Nginx上的开发可靠性高。

此外，该系统两个模块之间耦合性非常小，可维护性高。

### 4．运行环境规定

4.1设备

运行该软件所需要的硬设备、新型设备及其专门功能，包括：

1. 处理器型号及内存容量

处理器型号：Core i5以上

内存容量：8G以上

1. 外存容量、联机或脱机、媒体及其存储格式，设备的型号及数量

外存容量：5G以上

联机或脱机：联机

1. 输入及输出设备的型号和数量，联机或脱机；

输入设备为任意一台可以访问网站的终端设备，联机。

高性能服务器，数量依据访问的流量进行变动，联机。

1. 数据通信设备的型号和数量

正常的数据通信设备即可，数量依据访问的流量进行变动，联机。

1. 功能键及其他专用硬件

暂无。

4.2支持软件

系统架构：Nginx第三方模块开发

开发环境：Cygwin、Windows系统、Eclipse-cdt

编程语言：C语言、JavaScript、html、css

4.3接口

接口：动态ID生成算法插件、Nginx反向代理服务

数据通信协议：TCP协议、HTTP协议

4.4控制

只需要正常的部署前将本项目开发的第三方模块一同编译进去即可。

附录6 用户指导手册

基于代码混淆的Web站点动态防御系统

用户手册概要

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月六日]

### 1．引言

1.1编写目的

本文档为“基于代码混淆的Web站点动态防御系统”的使用人员提供指导。

1.2背景

本系统名称：基于代码混淆的Web站点动态防御系统

本系统基于Nginx开发，它的使用遵循Nginx模块的编译，部署，启动。

1.3定义

本系统物理表现为两个部分，即两个单独文件。其一，浏览器插件确定为Chrome Extension，名称：protect；其二，Nginx 的一个是第三方模块，名称为ngx\_http\_ipslab\_module。

本文档目标人群有访问网站的用户和部署网站的网站开发人员。对于网站开发人员，本文档介绍如何使用本系统提供的功能，如何部署，启动，配置本系统到您的网站。对于访问网站的用户，本文档介绍当你访问一个使用了本系统进行反向代理的网站时，需要您配合的操作。

1.4参考资料

[1] Netcraft.January 2019 Web Server Survey [EB/OL] . (2019-01-24) [2019-04-25] .https://news.netcraft.com/archives/2019/01/24/january-2019-web-server-survey.html

### 2．用途

2.1功能

过滤掉一些自动化攻击。包括恶意爬虫，薅羊毛等自动化请求。

2.2性能

2.2.1精度

系统中使用的ID的长度设置为32位。

2.2.2时间特性

本系统部署在Nginx反向代理的环境下，对于需要处理的HTML,JS文件，需要存储，处理，再转发，其响应时间至少是直接访问目标网站的2倍，但依托于Nginx本身的高性能在文本型的使用过程中不会产生较长延迟。对于图片等不需要处理的响应文件，直接转发，也不会产生长延迟的问题。

2.2.3灵活性

Nginx整体设计架构是模块化的，分为核心模块，标准HTTP模块和第三方模块等，本系统开发结果中的是第三方模块，在使用时，可以通过指令添加到Nginx中去，在不需要使用时，完全不影响Nginx其他的使用。

本系统的依赖除了Nginx的核心模块，所涉及到的Gumbo，Duktape都包含在自己模块中，集成度好。

本系统的客户端依赖用户市场占比高达六成的Chrome浏览器，覆盖面广；本系统在目前阶段的开发成果来看，实践上可能存在一些没有考虑的情况需要继续完善。但它的设计目标，理想状态，理论上对目标网站没有要求在实现ID验证之外要求对响应进行透传。在继续升级系统，更趋于成熟是，灵活易用。

2.3安全保密

系统开源，无安全保密需要

本系统作为反向代理处的处理，起过滤作用，不涉及用户，网站相关业务信息。

### 3．运行环境

3.1硬设备

Linux操作系统，或Windows系统+Cygwin环境

3.2支持软件

Nginx，Chrome

语言：C

3.3数据结构

支持C语言开发中的数据结构

涉及Gumbo，Duktape中的数据结构

### 4．使用过程

4.1安装与初始化

对于访问网站的浏览器用户：

安装Chrome Extension “protect”：

1. 第一步，解压Chrome Extension “protect”

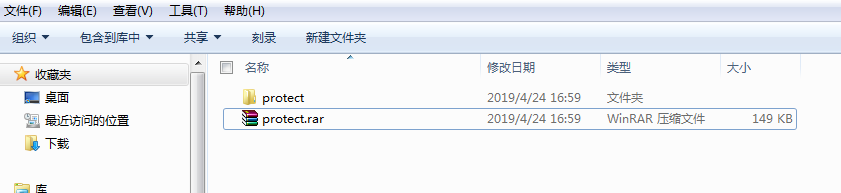


图6- 1 解压

1. 第二步，鼠标点击Chrome浏览器右上角竖三点的按钮浏览器，点开后选择更多工具选项，然后，点击扩展程序来启动Chrome浏览器的扩展管理器页面。

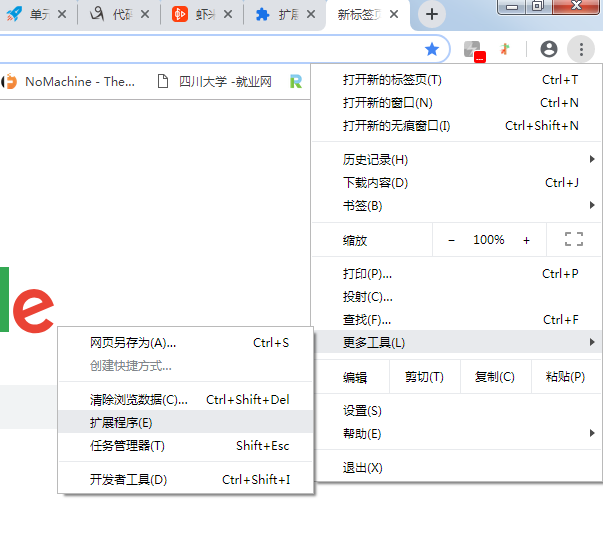


图6- 2 打开扩展程序管理

1. 第三步，在打开的Chrome浏览器的扩展管理器中我们就看到了一些已经安装的程序Chrome扩展程序，或者一个Chrome插件也没有。然后，我们要保证已经勾选了开发者模式。在左边选择 “加载已解压的扩展程序”。选中解压好的文件夹。选中打开。

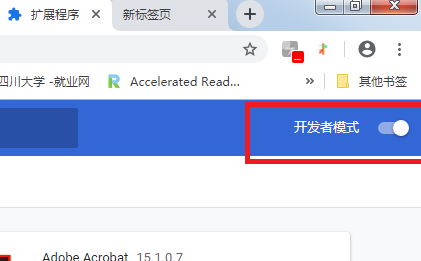


图6- 3 开启开发者模式

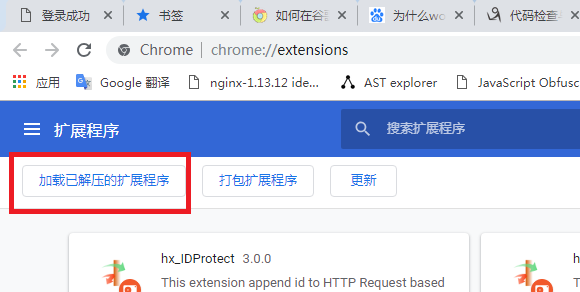


图6- 4 安装扩展程序

1. 安装成功。

对于部署目标网站的网站开发人员：

首先介绍常规Nginx第三方模块使用相关操作

1. 下载Nginx源码

在Nginx官网上面下载源码包。将下载的nginx-1.x.xx.tar.gz源代码压缩包放到准备好的Nginx源代码目录中，解压。解压指令如下：tar –zxvf nginx-1.x.xx.tar.gz。在本项目的开发中使用了当前较稳定版本1.12.2。本项目涉及所有都以此为基准。

1. 编译

本项目的使用基础是“编译安装Nginx”，如果你掌握这部分内容，按照Nginx规范即可以将本模块安装好。对于不熟悉的网站开发人员，这里简单介绍安装步骤：

1. 执行指令

./configure --prefix=path --add-module=s\_3rd\_mdl/ngx\_http\_ipslab\_module --with-debug

上述命令在一行，且将path换成你希望最后Nginx可运行文件套装的安装位置，它的形式可能会是这样”/mynginx/nginx-1.12.2”

1. 执行命令make，执行完后，执行make install。然后再你所设置的path下就生成了Nginx可运行的服务器。
2. 配置

在path(/mynginx/nginx-1.12.2)下找到conf文件夹下的nginx.conf文件进行配置

1. 注意事项

过程中错误请在Nginx相关网站查询，您会获得丰富的信息。

接下来介绍本模块相关指令

使用本模块的配置指令很简单 只有

ipslab arg1

其中，arg1表示一个参数。但在本模块配置指令外，还需要其他模块配合才能更好使用。

下面的例子详细描述使用的配置。

使用ngx\_http\_ipslab\_module模块自定义指令配置slab共享内存 8\*pagesize(4096)；

配置proxy\_buffer\_size 125K；

配置Keepalive\_time 0；

配置监听80端口；

配置符合”/tmpdir/”的location块代理到目标服务器8080端口。

具体在nginx.conf文件的对应的指令分别如下：

ip\_slab 8;

proxy\_buffer\_size 125k;

keepalive\_timeout 0;

listen 80;

location /tmpdir/{proxy\_pass 'http://127.0.0.1:8080//';}

注意：'http://127.0.0.1:8080//'替换成被代理的真正目标服务器网址。

4.2输入

4.2.1输入数据的现实背景

浏览器用户在浏览器扩展中输入反向代理服务器的域，即作用网站可能会用到的网址的正则表达式。例如要作用在”[www.baidu.com](http://www.baidu.com)”的网站上时，可以输入”\*://www.baidu.com/\*”

网站部署人员在Nginx配置中

指明要代理的服务器网址；

需要缓存时，用于缓存上游服务器响应内容的大小；

用于存放(IP,ID)的slab大小；

关闭keep-alive。

4.2.2输入格式

网址域的正则表达式

4.2.3输入举例

“\*://127.0.0.1/\*”

4.3输出

4.3.1输出数据的现实背景

输出已注入ID的目标网站的响应页面；

输出已经过混淆的JavaScript的脚本文件。

4.3.2输出格式

输出对用户来说对业务没有影响的页面响应。

4.3.3输出举例

对用户来说没有区别的页面响应

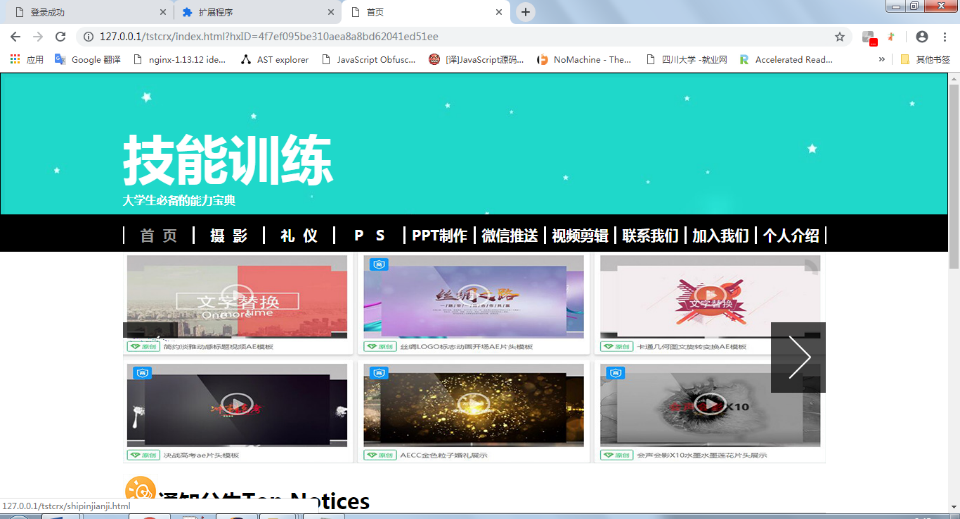


图6- 5 正确响应

但实际已经过ID注入，JavaScript混淆



图6- 6 对目标网站注入ID的效果

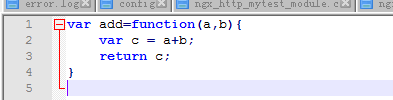


图6- 7 原目标网站JavaScript脚本

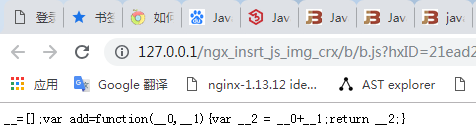


图6- 8 浏览器用户收到的经过混淆的JavaScript脚本

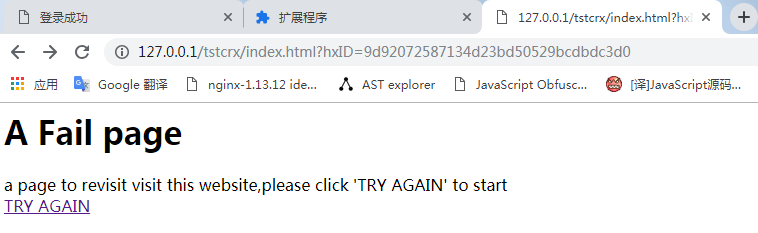
4.4文卷查询

暂无

4.5出错处理与恢复

本系统设计了“Fail Page”页面用来解决动态ID验证模块没有通过的请求。这个HTML页面（1）反馈未通过验证机制的信息（2）以链接形式反馈给用户未通过验证的请求，用户通过点击链接可以再次发出相同的资源请求。

这个设计提供了实际使用中未预料到的情况的错误处理。提高了容错性。



4.6终端操作

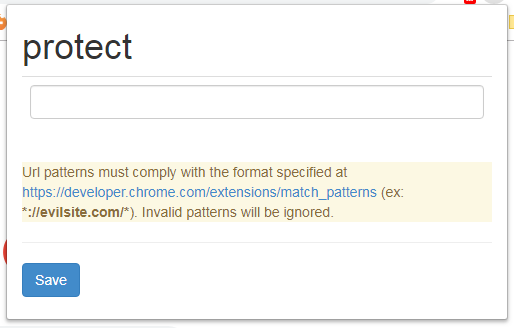


图6- 9 浏览器扩展界面：设置作用网站

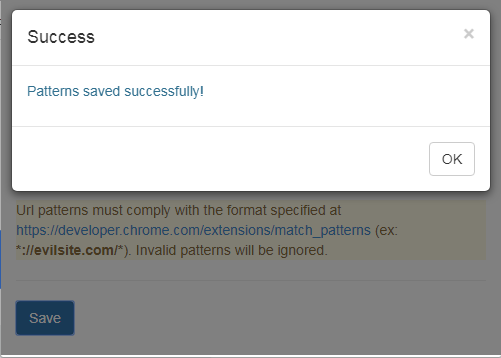


图6- 10 设置成功提示界面

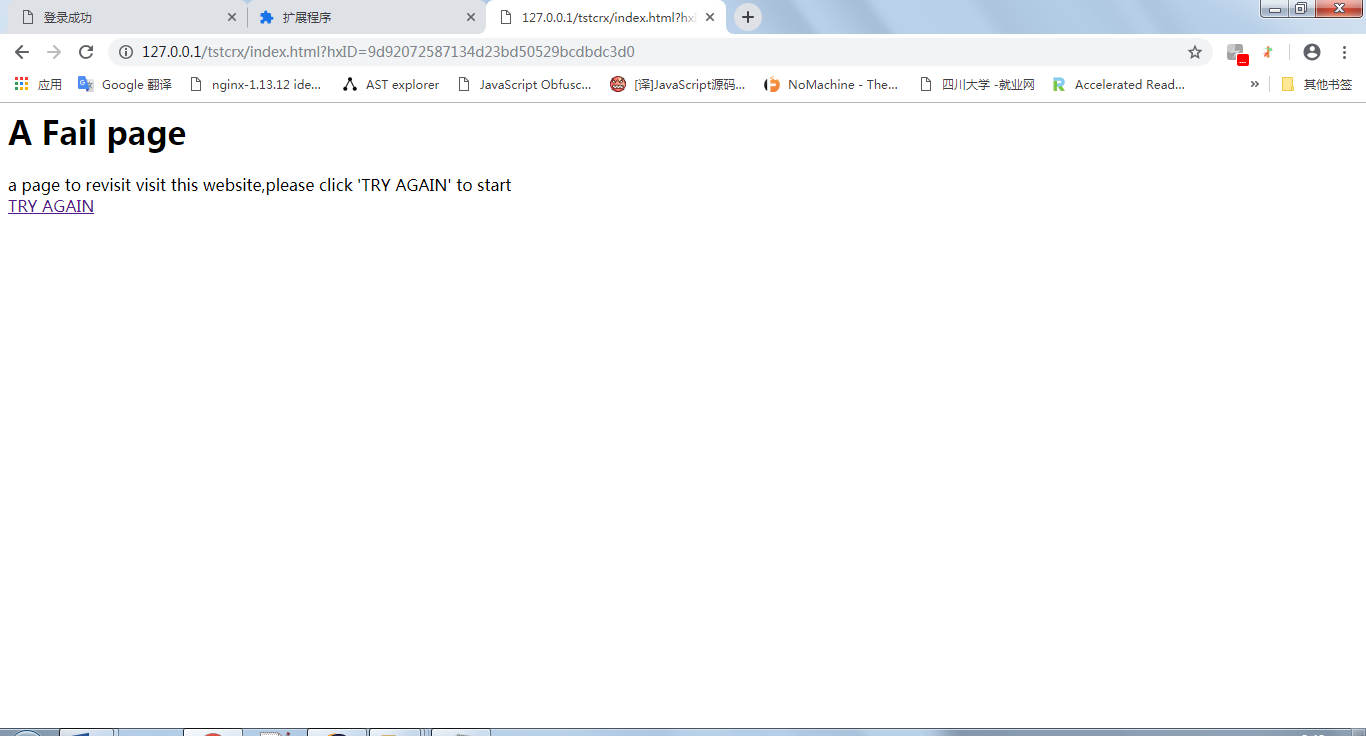


图6- 11 收到验证页面

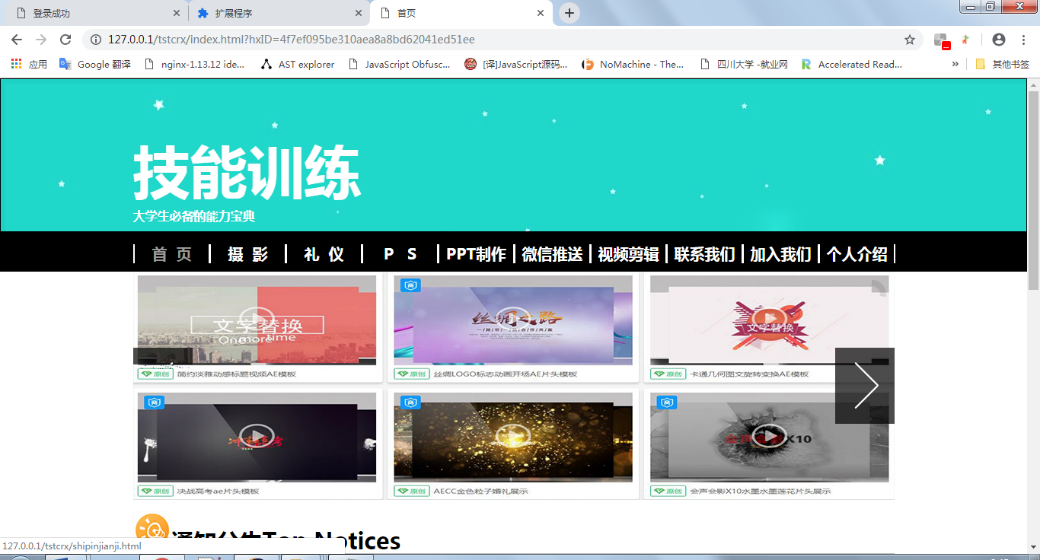


图6- 12 收到响应

附录7 概要设计说明书

基于代码混淆的Web站点动态防御系统

概要设计说明书

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二十六日]

### 1．引言

1.1编写目的

本详细设计说明书是为了实现系统的功能，进行的概要设计说明。具体针对功能的实现进行概要设计。

1.2背景

a. 基于代码混淆的Web站点动态防御系统

b.本项目的任务提出者：邵欣欣；开发者：邵欣欣；用户：网站开发人员及网站使用者。

1.3定义

Chrome Extension：Chrome浏览器的扩展，扩展浏览器的功能，例如：捕捉特定网页的内容，捕捉HTTP报文，捕捉用户浏览动作，改变浏览器地址栏/起始页/书签/Tab等界面元素的行为，与别的站点通信，修改网页内容等。

HTML：Hyper Text Markup Language的缩写，超文本标记语言。HTML 不是一种编程语言，而是制作网页必需的标记语言。“超文本”指页面可以包含图片，链接，音乐和程序等非文字元素。

JavaScript：JavaScript是因特网上最流行的脚本语言，一种直译式脚本语言。

Gumbo：一个C语言编写的开源的HTML解析器。

Duktape：一个C语言编写的可以嵌入C/C++工程的JavaScript引擎。

反向代理服务器：反向代理服务器是服务器的一种，它连接一组或多组后端服务器。用户只知道反向代理服务器的IP地址，对于用户请求，由反向代理服务器向后端服务器进行请求，在发送给用户。它与前向代理的明显区别是，前向代理作为客户端的代理，可能会将资源发送给一个或多个客户端。

AngularJS：是一个JavaScript框架，使用MVC 架构在HTML与JavaScript见提供良好用户体验。

slab共享内存：Nginx的设计架构是一个多进程的程序，收到的请求分布在不同的进程上，当请求的完成需要各进程相互配合时，进程间需要通信，但我们知道，进程间通信开发往往会很困难，为此，Nginx实现了一套共享内存管理机制-slab。

红黑树：一种数据结构，自平衡二叉查找树。

subrequest:Nginx模块中处理子请求的模块。

DOM：Document Object Model，文档对象模型。简单来说，就是一个HTML文件对应结构的数据结构。DOM是表示和处理一个HTML或XML文档的常用方法。是以面向对象的方式描述文档的模型。每一个标签都是树上的一个节点。

AST：Abstract Syntax Tree，抽象语法树，简称语法树，是源代码语法结构的抽象表示。它以树的形式表示编程语言的语法结构，分析源代码更直观，更形象，更贴近人的感性认知。

代码混淆：本系统中涉及到的代码混淆均不涉及到加密解密，对于人来说，通过改变形式让脚本代码变得晦涩难懂，但对于机器来说，并没有实质不同。

MD5：MD5 Message-Digest Algorithm，MD5消息摘要算法。MD5将文本信息，通过其不可逆的字符串变换算法，产生了这个唯一的MD5信息。

1.4参考资料

### 2．总体设计

2.1需求规定

2.1.1系统功能

2.1.1.1动态ID匹配进行人机识别

在slab共享内存中ID的更新，存储，修改，查看

浏览器与代理服务器之间ID交流的逻辑功能

2.1.1.2混淆功能

在返回目标服务器的响应内容前， 对发给用户的文件进行混淆操作。混淆使用随机代码混淆技术对受保护站点页面关键元素和客户端ID生成算法进行随机化混淆处理，阻止攻击者对网页的逆向分析。

2.1.2系统性能

2.1.2.1精度：无

2.1.2.2时间特性要求：保证一定的性能，不能在反向代理处做无故的滞留影响响应速度。具体度量不超过原目标站点响应时间的两倍。

2.1.2.4可靠性：系统运行可靠。系统应该有规避错误的机制。

2.1.2.5灵活性：耦合度低

2.1.3输入输出要求

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 数据名称 | 数据类型 | 存储媒体 | 格式（示例） | 数值范围 | 精度 |
| 1 | 网站正则表达式 | 字符串 | 本地村塾 | \*://127.0.0.1/\* | - | - |
| 2 | 网址 | 字符串 | - | http://127.0.0.1/tstcrx/index.html | - | - |

2.1.4数据管理能力要求

对于一定的访问量，保持ID库的淘汰，不需要很高的数据存储和管理能力。

2.1.5故障处理要求

软件故障包括联网失败、数据读取错误等。后果是系统延宕、崩溃。需专业人员解决。

2.1.6其他专门要求

2.2运行环境

2.2.1设备

PC

2.2.2支持软件

Windows 7,Chrome,Cygwin

2.2.3接口

Nginx 模块开发接口。

Chrome 扩展开发接口。

2.2.4控制

开发人员决定

2.3基本设计概念和处理流程

[说明本系统的基本设计概念和处理流程，尽量使用图表的形式。]

2.4结构

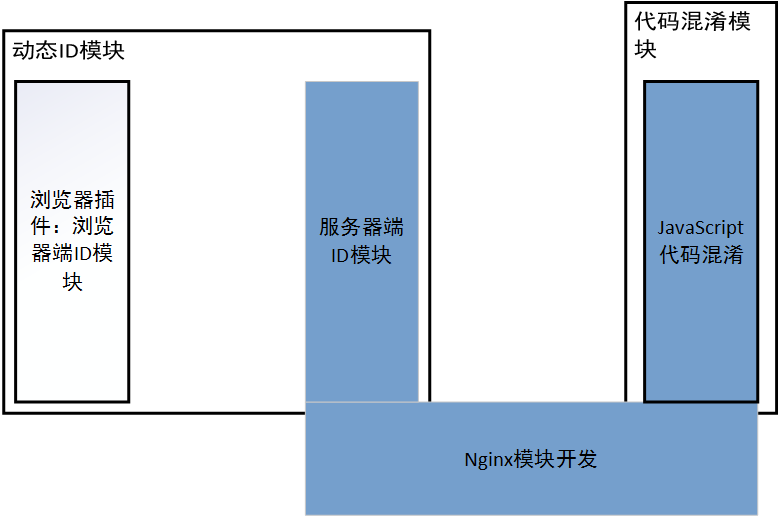


图6- 13 系统开发模块分布图

2.5功能需求与系统模块的关系

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nginx模块开发 | 动态ID模块 | 代码混淆 |
| 过滤自动化攻击 | √ | √ |  |
| 代码保护 | √ |  | √ |

2.6人工处理过程

混淆程序所在的磁盘位置设置需要手动

2.7尚未解决的问题

具体的混淆算法。

### 3．接口设计

3.1用户接口

Pattern

Ipslab

3.2外部接口

无

3.3内部接口

ngx\_http\_ipslab\_lookup\_ID\_update\_forward

duktape相关接口

gumbo相关接口

### 4．运行设计

4.1运行模块组合

动态ID模块：ID分发模块，动态ID模块：浏览器扩展模块

动态ID模块：浏览器扩展模块，动态ID模块：ID验证模块和代码混淆模块

4.2运行控制

外接通过输入控制：

请求携带合法ID或者不合法ID

4.3运行时间

### 5．系统数据结构设计

5.1逻辑结构设计要点

满足接口需要的数据结构

5.2物理结构设计要点

利用C内存结构进行数据结构更灵活的分配。

5.3数据结构与程序的关系

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nginx模块开发 | 动态ID模块 | 代码混淆 |
| ngx\_http\_ipslab\_conf\_t | √ |  |  |
| ngx\_http\_ipslab\_node\_ | √ | √ |  |
| ngx\_http\_ipslab\_shm\_t | √ |  |  |
| ngx\_http\_ipslab\_ctx\_t | √ |  |  |
| Append\_flag |  | √ |  |

### 6．系统出错处理设计

6.1出错信息

“pattern is not valid”

6.2补救措施

1. 重新设置

6.3系统维护设计

代码安全概念贯穿整个开发过程中。

附录8 数据库设计说明

基于代码混淆的Web站点动态防御系统

数据库设计说明书

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二十六日]

### 1.引言

1.1编写目的

本文档对应项目：基于代码混淆的web站点动态防御系统。

本详细设计说明书是为了实现系统的功能，进行的数据库说明。详细描述数据库设计。

本“数据库设计说明书”主要用于：指导数据库实现，指定数据库标准，指导软件开发。

1.2背景

1. 基于代码混淆的Web站点动态防御系统
2. 任务提出者：邵欣欣；开发者：邵欣欣；用户：网站开发人员

1.3定义

无

1.4参考资料

无

### 2．外部设计

1.1标识符的状态

数据单位元：ngx\_http\_ipslab\_node\_t

数据库对数据单元的组织结构：ngx\_rbtree\_t

数据库本身的管理结构：ngx\_http\_ipslab\_shm\_t

2.2使用它的程序

函数ngx\_http\_ipslab\_lookup\_ID\_update\_forward

2.3约定

数据库内数据更新法则：MD5算法从上一个数据得到下一个数据

2.4专门指导

参考文献：[1] 陶辉.深入理解Nginx-模块开发与架构解析第二版[M].北京：机械工业出版社，2016:590-605

2.5支持软件

Nginx的内部模块：slab是必需的支持。

### 3．结构设计

本系统的数据库设计并非一般意义上的外部数据库。它使用Nginx内部提供的slab共享内存技术来开辟出一块区域来提供数据访问，优点是解决了Nginx架构中各模块之间的通信问题。缺点是共享内存提供的区域是有限的，当数据太多时，频繁淘汰再新建会影响性能。

### 4．运用设计

4.1数据字典设计

ngx\_http\_ipslab\_module模块内的结构体设计如下：

表8-1 ngx\_http\_ipslab\_module模块配置结构体ngx\_http\_ipslab\_conf\_t

|  |  |  |
| --- | --- | --- |
| **变量名称** | **变量类型** | **含义** |
| shmsize\_int | ngx\_int\_t | 共享内存的大小，以pagesize=4096为单位大小 |
| shpool | ngx\_slab\_pool\_t | 共享内存结构体 |
| sh | ngx\_http\_ipslab\_shm\_t | 控制红黑树和链表 |

表8-2 存储(IP,ID)对的节点ngx\_http\_ipslab\_node\_t结构体

|  |  |  |
| --- | --- | --- |
| **变量名称** | **变量类型** | **含义** |
| rbtree\_node\_data | u\_char | 实际存储信息开始的位置 |
| slab\_ID | u\_char[33] | 下发给浏览器端的ID |
| last\_slab\_ID | u\_char[33] | 上一次下发给浏览器端的ID |
| queue | ngx\_queue\_t | 一条一定长度保持访问记录的链表，用来淘汰过期节点 |

表8-3 ngx\_http\_ipslab\_shm\_t结构体

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| rbtree | ngx\_rbtree\_t | 红黑树用于快速检索 |
| sentinel | ngx\_rbtree\_node\_t | 使用红黑树必须定义的哨兵节点 |
| queue | ngx\_queue\_t | 淘汰链表 |

表8-4 ngx\_http\_ipslab\_ctx\_t表

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| out\_ctx | ngx\_chain\_t \* | 传递子请求返回结果 |
| hxID | ngx\_str\_t | 父子请求间传递ID |

表8-5枚举类型Append\_flag定义

|  |  |  |
| --- | --- | --- |
| **字段名称** | **存储类型** | **含义** |
| pure\_chrctr | 0 | 标识添加“hxID=xxxx” |
| ampersand\_chrctr | 1 | 标识添加“&hxID=xxxx” |
| question\_chrctr | 2 | 标识添加“?hxID=xxxx” |

4.2安全保密设计

无

附录9 组装测试计划

基于代码混淆的Web站点动态防御系统

组装测试计划

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月六日]

### 1.引言

1.1编写目的

本测试计划用来指导测试，制定详细的测试计划，使用测试执行结果确保软件开发正确的功能，确保功能的正确实现。保证软件开发在预期轨道。预期读者是需求分析人员，项目经理，软件开发人员和网站安全运维人员。

1.2背景

1. 基于代码混淆的Web站点动态防御系统
2. 任务提出者：邵欣欣；开发者：邵欣欣；用户：网站安全运维人员。

1.3定义

Chrome Extension：Chrome浏览器的扩展，扩展浏览器的功能，例如：捕捉特定网页的内容，捕捉HTTP报文，捕捉用户浏览动作，改变浏览器地址栏/起始页/书签/Tab等界面元素的行为，与别的站点通信，修改网页内容等。

HTML：Hyper Text Markup Language的缩写，超文本标记语言。HTML 不是一种编程语言，而是制作网页必需的标记语言。“超文本”指页面可以包含图片，链接，音乐和程序等非文字元素。

JavaScript：JavaScript是因特网上最流行的脚本语言，一种直译式脚本语言。

Gumbo：一个C语言编写的开源的HTML解析器。

Duktape：一个C语言编写的可以嵌入C/C++工程的JavaScript引擎。

反向代理服务器：反向代理服务器是服务器的一种，它连接一组或多组后端服务器。用户只知道反向代理服务器的IP地址，对于用户请求，由反向代理服务器向后端服务器进行请求，在发送给用户。它与前向代理的明显区别是，前向代理作为客户端的代理，可能会将资源发送给一个或多个客户端。

AngularJS：是一个JavaScript框架，使用MVC 架构在HTML与JavaScript见提供良好用户体验。

slab共享内存：Nginx的设计架构是一个多进程的程序，收到的请求分布在不同的进程上，当请求的完成需要各进程相互配合时，进程间需要通信，但我们知道，进程间通信开发往往会很困难，为此，Nginx实现了一套共享内存管理机制-slab。

红黑树：一种数据结构，自平衡二叉查找树。

subrequest:Nginx模块中处理子请求的模块。

DOM：Document Object Model，文档对象模型。简单来说，就是一个HTML文件对应结构的数据结构。DOM是表示和处理一个HTML或XML文档的常用方法。是以面向对象的方式描述文档的模型。每一个标签都是树上的一个节点。

AST：Abstract Syntax Tree，抽象语法树，简称语法树，是源代码语法结构的抽象表示。它以树的形式表示编程语言的语法结构，分析源代码更直观，更形象，更贴近人的感性认知。

代码混淆：本系统中涉及到的代码混淆均不涉及到加密解密，对于人来说，通过改变形式让脚本代码变得晦涩难懂，但对于机器来说，并没有实质不同。

MD5：MD5 Message-Digest Algorithm，MD5消息摘要算法。MD5将文本信息，通过其不可逆的字符串变换算法，产生了这个唯一的MD5信息。

1.4参考资料

列出有关的参考资料。

### 2．计划

2.1系统说明

本系统：基于代码混淆的Web站点动态防御系统有如下划分方法：

从逻辑上可分为动态ID模块和的代码混淆模块，其中动态ID模块分为浏览器插件模块和动态ID服务器模块。

如下图： A代表动态ID浏览器插件模块，B代表动态ID服务器模块，A和B共同组成动态ID模块，C代表代码混淆模块

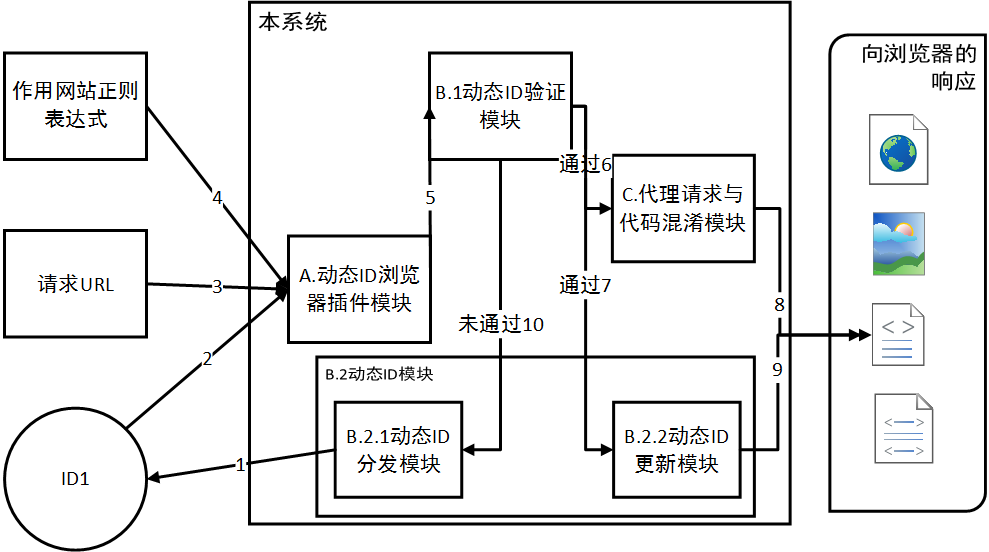


图9- 1 数据流通图

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 测试内容 | 进度安排 | 测试目的 |
| 01 | 浏览器扩展”protect”作用网站的正确设置 | 2019.03.10-2019.03.12 | 确认反向代理服务器网址的正则表达式可以成功设置 |
| 02 | 浏览器插件给请求加ID | 2019.03.10-2019.03.12 | 确认拦截请求，重定向成功 |
| 03 | 服务器ID分配 | 2019.03.10-2019.03.12 | 确认ID新建成功 |
| 04 | 服务器ID进行比较验证 | 2019.03.12-2019.03.13 | 确认ID验证模块功能实现 |
| 05 | 服务器向HTML文件注入ID | 2019.03.12-2019.03.13 | 确认动态ID模块服务器端功能实现 |
| 06 | 服务器混淆JavaScript响应 | 2019.03.13-2019.03.15 | 确认代码混淆模块正常工作 |

2.2测试内容

A代表动态ID浏览器插件模块；

B代表动态ID服务器模块；

C代表代码混淆模块

2.3测试1（模块A，B）

模块A的下述3个单元：

ContentScript单元

Background单元

Popup单元

模块B的下述单元：

ID分发单元

ID验证单元

ID更新单元

2.3.1进度安排

2019.03.10-2019.03.12

2.3.2条件

1. 硬件： PC
2. 软件：Windows操作系统，Chrome浏览器。
3. 人员：邵欣欣

2.3.3测试资料

当前页面是由本系统响应给浏览器的，在浏览器控制台。

2.3.4测试培训

培训的内容：JavaScript的调试，浏览器扩展的调试

受训的人员：邵欣欣。

2.4测试2（模块C）

测试代码混淆模块

2.4.1进度安排

2019.03.10-2019.03.12

2.4.2条件

1. 硬件： PC
2. 软件：Windows操作系统，Nginx服务器。
3. 人员：邵欣欣

2.4.3测试资料

[1] 陶辉.深入理解Nginx-模块开发与架构解析第二版[M].北京：机械工业出版社，2016

2.4.4测试培训

培训的内容：JavaScript的调试，浏览器扩展的调试

受训的人员：邵欣欣。

### 3．测试设计说明

3.1测试1（01）

3.1.1控制

黑盒测试和白盒测试相结合

3.1.2输入

用户配置浏览器扩展作用网站的正则表达式

3.1.3输出

作用网站的正则表达式数据正确存储到Chrome.localstorage对象

3.1.4过程

用户配置浏览器扩展作用网站的正则表达式，作用网站的正则表达式数据正确存储到Chrome.localstorage对象。

3.2测试2（02）

3.2.1控制

黑盒测试

3.2.2输入

用户在浏览器地址栏输入已设置网站的一个网址请求

3.2.3输出

浏览器发出带ID的请求

3.2.4过程

用户在浏览器地址栏输入已设置网站的一个网址请求，浏览器发出带ID的请求

3.3测试3（03）

3.3.1控制

白盒测试

3.3.2输入

一个请求

3.3.3输出

在没有在数据库中找到请求相关信息时，数据库中分配一个新节点记录这个IP对应的ID。

3.3.4过程

服务器接受一个请求，在没有在数据库中找到请求相关信息时，数据库中分配一个新节点记录这个IP对应的ID

3.4测试4（04）

3.4.1控制

白盒测试

3.4.2输入

服务器将请求中的ID和数据库中的ID进行比较

3.4.3输出

比较结果

3.4.4过程

服务器将请求中的ID和数据库中的ID进行比较，输出比较结果。

3.5测试5（05）

3.5.1控制

黑盒测试和白盒测试相结合

3.5.2输入

HTML文件

3.5.3输出

添加了ID信息在其中的HTML文件

3.5.4过程

服务器向HTML文件注入ID

3.6测试6（06）

3.6.1控制

黑盒测试和白盒测试相结合

3.6.2输入

JavaScript脚本文件

3.6.3输出

经过混淆的JavaScript脚本文件

3.6.4过程

打开监控系统，用户点击查看网站接口耗时情况时，监控系统应该展示用户网站接口耗时图表给用户

### 4．评价准则

4.1范围

所选择的测试用例能够检查到大部分所有合法与不合法的输入。

4.2数据整理

输入的测试数据基本上能够满足测试的预期的要求，整个的数据处理基本上可以达到预 期的结果。测试基本通过。

4.3尺度

测试数据采用黑盒、白盒同时进行，输入的数据通过预期的结果来达到最终的测试目的，如测试的数据有偏差，则重新组装，再测试，允许中断或停机的最大数为5次。

附录10 详细设计说明书

基于代码混淆的Web站点动态防御系统

详细设计说明书

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月六日]

### 1.引言

1.1编写目的

本项目将开发一个基于代码混淆的web站点动态防御系统。

本详细设计说明书是为了实现系统的功能，进行的详细设计说明。具体针对功能与技术进行详细的方案设计。

本“设计说明书”主要用于：详细设计本系统的各个功能模块，为程序员写出实际的程序代码提供依据。为测试人员编写测试用例提供指导。为项目经理等人提供系统的说明。它是软件设计阶段所有任务和所有相关人员所需的参考资料。

1.2背景

1. 基于代码混淆的Web站点动态防御系统
2. 任务提出者：邵欣欣；开发者：邵欣欣；用户：网站开发人员

1.3定义

Chrome Extension：Chrome浏览器的扩展，扩展浏览器的功能，例如：捕捉特定网页的内容，捕捉HTTP报文，捕捉用户浏览动作，改变浏览器地址栏/起始页/书签/Tab等界面元素的行为，与别的站点通信，修改网页内容等。

HTML：Hyper Text Markup Language的缩写，超文本标记语言。HTML 不是一种编程语言，而是制作网页必需的标记语言。“超文本”指页面可以包含图片，链接，音乐和程序等非文字元素。

JavaScript：JavaScript是因特网上最流行的脚本语言，一种直译式脚本语言。

Gumbo：一个C语言编写的开源的HTML解析器。

Duktape：一个C语言编写的可以嵌入C/C++工程的JavaScript引擎。

反向代理服务器：反向代理服务器是服务器的一种，它连接一组或多组后端服务器。用户只知道反向代理服务器的IP地址，对于用户请求，由反向代理服务器向后端服务器进行请求，在发送给用户。它与前向代理的明显区别是，前向代理作为客户端的代理，可能会将资源发送给一个或多个客户端。

AngularJS：是一个JavaScript框架，使用MVC 架构在HTML与JavaScript见提供良好用户体验。

slab共享内存：Nginx的设计架构是一个多进程的程序，收到的请求分布在不同的进程上，当请求的完成需要各进程相互配合时，进程间需要通信，但我们知道，进程间通信开发往往会很困难，为此，Nginx实现了一套共享内存管理机制-slab。

红黑树：一种数据结构，自平衡二叉查找树。

subrequest:Nginx模块中处理子请求的模块。

DOM：Document Object Model，文档对象模型。简单来说，就是一个HTML文件对应结构的数据结构。DOM是表示和处理一个HTML或XML文档的常用方法。是以面向对象的方式描述文档的模型。每一个标签都是树上的一个节点。

AST：Abstract Syntax Tree，抽象语法树，简称语法树，是源代码语法结构的抽象表示。它以树的形式表示编程语言的语法结构，分析源代码更直观，更形象，更贴近人的感性认知。

代码混淆：本系统中涉及到的代码混淆均不涉及到加密解密，对于人来说，通过改变形式让脚本代码变得晦涩难懂，但对于机器来说，并没有实质不同。

MD5：MD5 Message-Digest Algorithm，MD5消息摘要算法。MD5将文本信息，通过其不可逆的字符串变换算法，产生了这个唯一的MD5信息。

1.4参考资料

**[**列出有关的参考资料。]

### 2. 系统的结构

本系统架构图如下：

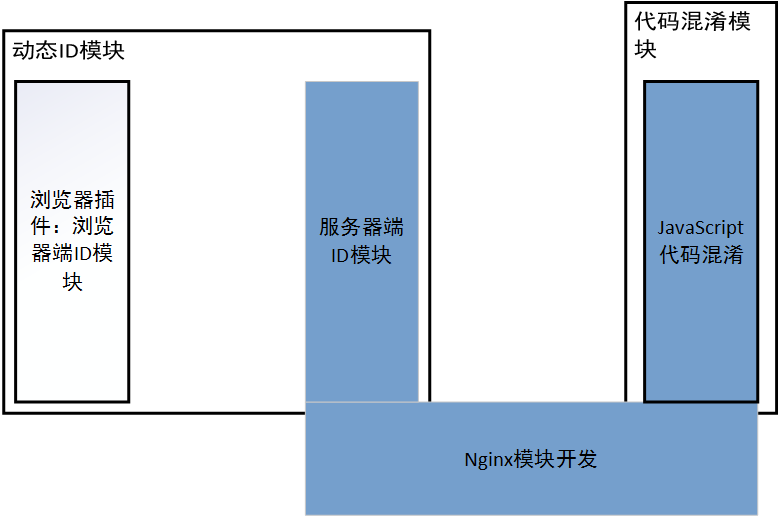


图10- 1 系统架构图

### 3．模块1（01）设计说明

3.1模块描述

Nginx既可以做Web服务器，又可以做反向代理服务器。Nginx整体采用高度模块化的架构设计。我们可以通过开发自定义HTTP模块来实现产品功能，根据Nginx框架定义的数据结构，将自定义HTTP模块介入Nginx 处理HTTP请求的处理流程。

3.2功能

实现Nginx框架定义的HTTP模块的数据结构。

3.3性能

无

3.4输入项

无

3.5输出项

一个自定义HTTP模块

3.6设计方法（算法）

不涉及算法，根据官方定义实现一个标准的HTTP模块

3.7流程逻辑

根据Nginx的定义，这个模块是充满异步调用、无阻塞的HTTP模块

3.8接口

（1）要能作为一个标准HTTP模块介入Nginx处理HTTP请求的处理流程，需要实现的接口有：

Config文件；

ngx\_command\_t 配置项的处理

ngx\_module\_t 模块结构体

ngx\_http\_module\_t Nginx将HTTP请求的处理分成了各个阶段。本结构体提供8个时间点的回调方法。

（2）要为动态ID模块提供内存大小指示

Size

（3）要为代码混淆模块提供混淆程序

程序文件的filePath

3.9存储分配

无

3.10注释设计

3.11限制条件

只针对Nginx服务器

3.12测试计划

通过模块提供的指令来启动模块。

3.13尚未解决的问题

### 4．模块2（02）设计说明

4.1模块描述

动态ID模块的实现由两部分配合完成，分别是浏览器端和服务器端。

4.2功能

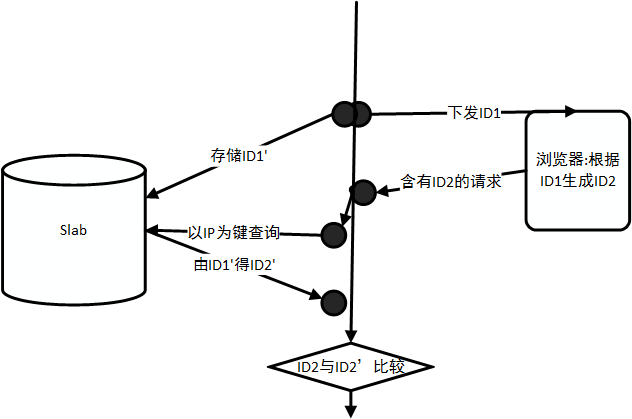


图10- 2 浏览器，服务器ID交互图

4.3性能

无明显阻塞

4.4输入项

一个用户请求

4.5输出项

一个用户请求的拒绝或通过。

4.6设计方法（算法）

MD5是一个安全的散列算法，输入两个不同的明文不会得到相同的输出值，根据输出值，不能得到原始的明文，即其过程不可逆。

4.7流程逻辑

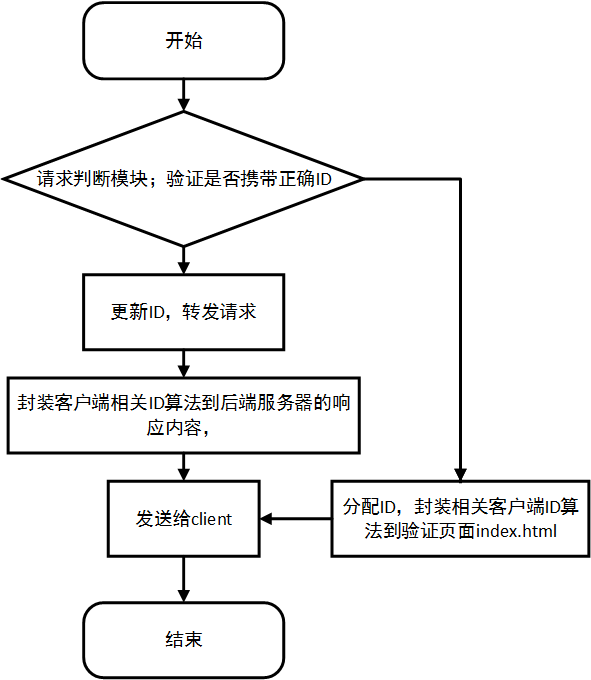


图10- 3 服务器内ID验证机制

4.8接口

（1）slab调用接口。

（2）HTTP请求处理借口

4.9存储分配

存储(IP,ID)对。

4.10注释设计

4.11限制条件

只针对Nginx服务器和Chrome浏览器

4.12测试计划

通过模块提供的指令来启动模块。

4.13尚未解决的问题

### 5．模块3（03）设计说明

5.1模块描述

代码混淆模块使用混淆算法保护JavaScript脚本。

5.3性能

无

5.4输入项

待混淆脚本

5.5输出项

已混淆脚本

5.6设计方法（算法）

正则匹配替换名称

5.7流程逻辑

5.8接口

（1）拿到上游响应。

（2）输出到发送链

5.9存储分配

无

5.10注释设计

5.11限制条件

只针对Nginx服务器

5.12测试计划

通过模块提供的指令来启动模块。

5.13尚未解决的问题

附录11 模块开发说明书

基于代码混淆的Web站点动态防御系统

模块开发说明

[V1.0]

拟 制 人\_\_\_\_\_\_\_\_\_\_邵欣欣\_\_\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_王海舟\_\_\_\_\_\_\_\_\_\_\_\_

[二零一九年三月二十六日]

### 1．标题

系统名称：基于代码混淆的Web站点动态防御系统

模块名称和标识符：Nginx模块开发（01）、动态ID模块（02）、代码混淆模块（03）

程序编制员签名:邵欣欣

修改完成日期：2019年4月27日

编排日期：2019年4月27日

### 2．模块开发情况表

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 模块名称 | 开发开始日期 | 开发完成日期 |
| 01 | Nginx模块开发 | 2018.10.22 | 2018.11.22 |
| 02 | 动态ID模块 | 2018.11.25 | 2019.1.2 |
| 03 | 代码混淆模块 | 2019.1.2 | 2019.3.1 |

### 3．功能说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模块 | 功能 | 输入 | 输出 |
| Nginx模块开发（01） | 介入Nginx | 无 | 无 |
| 动态ID模块（02） | ID增删查 | ID | 树 |
| ID比较验证 | ID | 结果是否相等的判断 |
| 代码混淆模块（03） | JavaScript混淆 | JavaScript脚本 | 混淆过的脚本 |

### 4．设计说明

|  |  |  |
| --- | --- | --- |
| 标识符 | 模块名称 | 设计考虑 |
| 01 | Nginx模块开发 | 较好的扩展性 |
| 02 | 动态ID模块 | 高效查找 |
| 03 | 代码混淆模块 | 好的混淆效果 |

### 5．硬件部分的设计结果

无。

### 6．软件的设计结果

|  |  |  |
| --- | --- | --- |
| 标识符 | 模块名称 | 源代码清单 |
| 01 | Nginx模块开发 | ngx\_http\_ipslab\_module.c |
| 02 | 动态ID模块 | ngx\_http\_ipslab\_module.ctag\_enum.h,gumbo.h,popup.html,popup.js,ContentScript.js,background.js |
| 03 | 代码混淆模块 | Libduktape.a,duktape.h,duk\_config.h,hx.js |

### 7．测试说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模块 | 编号 | 功能 | 输入 | 预期输出 | 实际输出 |
| Nginx模块开发 | a | 介入Nginx | 无 | 模块正确编译进Nginx | 模块正确编译进Nginx |
| 动态ID模块  Nginx模块开发  动态ID模块 | b | 浏览器扩展”protect”作用网站的正确设置 | 网站正则表达式 | 设置成功 | 设置成功 |
| c | 浏览器插件给请求加ID | URL | 加了ID的URL | 加了ID的URL |
| d | 服务器ID分配 | IP | ID | ID |
| e | 服务器ID进行比较验证 | ID1，ID2 | 真/假 | 真/假 |
| f | 服务器向HTML文件注入ID | HTML文件，ID | 注入成功 | 注入成功 |
| Nginx模块开发 | g | 服务器混淆JavaScript响应 | JavaScript混淆 | 成功混淆 | 成功混淆 |

### 8．复审的结论

经过单元测试和集成测试，发现测试的结果符合需求说明以及系统设计说明书中的规定，符合系统预期。

附录12 单元测试报告

单元测试报告

填表日期： 20190423 编号：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | | 第一责任人 | | | | 邵欣欣 | | |
| 单元名称 | | | 浏览器端 | | | 责任人 | | | | 邵欣欣 | | | | 单元所属子系统 | | | | 动态ID浏览器端 | | 开发周期 | | | | 1个月 | |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | | 备注 | | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 边界正常 | | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 接口正常 | | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 数据真实 | | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 代码规范 | | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | | 操作方法 | | | | 结果 | | | | | 建议 | | | | 测试人员 | | | 备注 | |
| 01 | 浏览器扩展”protect”作用网站的设置：正确输入 | | | | | | 用户输入正确的反向代理服务器网址的正则表达式，点击保存，弹出设置成功的提示信息 | | | | 测试结果一切正常 | | | | |  | | | | 邵欣欣 | | |  | |
| 02 | 浏览器扩展”protect”作用网站的设置：错误输入 | | | | | | 用户输入不正确的反向代理服务器网址的正则表达式，点击保存，弹出输入不正确的提示信息 | | | | 测试结果一切正常 | | | | |  | | | | 邵欣欣 | | |  | |
|  |  | | | | | |  | | | |  | | | | |  | | | |  | | |  | |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | 项目第一责任人 | | | | | | | 邵欣欣 | | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | 总工办 | | | | 邵欣欣 | | | | 总工程师 | 邵欣欣 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | | 第一责任人 | | | | 邵欣欣 | | |
| 单元名称 | | | 浏览器插件background控制单元 | | | | 责任人 | | | 邵欣欣 | | 单元所属子系统 | | | | | | 动态ID模块 | | | 开发周期 | | 1个月 | |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | | | 备注 | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 路径正常 | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | - | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 循环正常 | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | - | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 接口正常 | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 界面正常 | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 数据确认正常 | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 代码规范 | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | 操作方法 | | | | | 结果 | | | | | 建议 | | | | | 测试人员 | | | 备注 |
| 01 | 浏览器扩展”protect” 捕捉符合网址正则表达式的请求 | | | | | 用户在地址栏手动输入请求网址发起请求。捕捉到请求后输出到控制台 | | | | | 测试结果一切正常 | | | | |  | | | | | 邵欣欣 | | |  |
| 02 | 浏览器扩展”protect” 在符合条件的请求后添加ID | | | | | 手动输入地址，期望扩展自动加上ID | | | | | 测试结果一切正常 | | | | |  | | | | | 邵欣欣 | | |  |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | | 项目第一责任人 | | | | | | 邵欣欣 | | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | | 总工办 | | | 邵欣欣 | | | 总工程师 | | 邵欣欣 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | | 第一责任人 | | | | 邵欣欣 | |
| 单元名称 | | | 浏览器插件ContentScript单元 | | | | 责任人 | | | 邵欣欣 | | 单元所属子系统 | | | | | | 动态ID模块浏览器插件 | | 开发周期 | | | 1个月 |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | 备注 | | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 路径正常 | | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | - | | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 循环正常 | | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | - | | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 接口正常 | | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 界面正常 | | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 数据确认正常 | | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | 代码规范 | | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | 操作方法 | | | | | 结果 | | | | | 建议 | | | 测试人员 | | 备注 | | |
| 01 | 浏览器扩展”protect” 捕捉页面内ID | | | | | 从反向代理服务器得到的页面中含有ID，输出捕捉到的ID | | | | | 测试结果一切正常 | | | | |  | | | 邵欣欣 | |  | | |
| 02 | ID的MD5加密 | | | | | MD5加密，使用线上MD5工具验证 | | | | | 测试结果一切正常 | | | | |  | | |  | |  | | |
| 03 | 浏览器扩展”protect” 的消息传递 | | | | | 从反向代理服务器得到的响应页面捕捉到ID，ContentScript用消息传递ID给background | | | | | 测试结果一切正常 | | | | |  | | | 邵欣欣 | |  | | |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | | 项目第一责任人 | | | | | | 邵欣欣 | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | | 总工办 | | | 邵欣欣 | | 总工程师 | | | 邵欣欣 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | 第一责任人 | | | | | | 邵欣欣 | | |
| 单元名称 | | | ID管理 | | | | 责任人 | | | 邵欣欣 | | 单元所属子系统 | | | | 动态ID模块服务器端 | | | | | 开发周期 | | | | 1个月 |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | | | 备注 | | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 路径正常 | | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | - | | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 循环正常 | | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | - | | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 接口正常 | | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 界面正常 | | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 数据确认正常 | | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | | 代码规范 | | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | 操作方法 | | | | | 结果 | | | | | | | 建议 | | | 测试人员 | | | 备注 | |
| 01 | 给初次访问的IP分配ID | | | | | 在启动Nginx反向代理服务器后，使用未曾访问过的IP访问反向代理服务器：在地址栏输入网址 | | | | | 测试结果一切正常 | | | | | | |  | | | 邵欣欣 | | |  | |
| 02 | 给携带正确ID的请求准备下一次使用的ID | | | | | 在启动Nginx反向代理服务器后，从反向代理的响应页面点击继续访问 | | | | | 测试结果一切正常 | | | | | | |  | | | 邵欣欣 | | |  | |
| 03 | Slab中(IP,ID)节点的过时淘汰 | | | | | 多个IP多次访问。输出slab中(IP,ID)节点的链表状态，应该根据访问情况发生变化 | | | | | 测试结果一切正常 | | | | | | |  | | | 邵欣欣 | | |  | |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | | 项目第一责任人 | | | | | | | 邵欣欣 | | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | | 总工办 | | | | | 邵欣欣 | | | 总工程师 | 邵欣欣 | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | | 第一责任人 | | | | | 邵欣欣 |
| 单元名称 | | | ID交互机制 | | | | 责任人 | | | 邵欣欣 | | 单元所属子系统 | | | | | | 动态ID模块服务器端 | | 开发周期 | | 1个月 | |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | | 备注 | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 路径正常 | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 循环正常 | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 接口正常 | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 界面正常 | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 数据确认正常 | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 代码规范 | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | 操作方法 | | | | | 结果 | | | | | 建议 | | | | 测试人员 | | 备注 | |
| 01 | Nginx反向代理服务器向转发的响应HTML页面注入ID | | | | | 当响应文件类型是”text/html”时，解析HTML，注入ID | | | | | 测试结果一切正常 | | | | |  | | | | 邵欣欣 | |  | |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | | 项目第一责任人 | | | | | | 邵欣欣 | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | | 总工办 | | | 邵欣欣 | | | | 总工程师 | | 邵欣欣 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 开发项目名称 | | | | | 基于代码混淆的Web站点动态防御系统 | | | | | 开发项目编号 | | | | | 无 | | | 第一责任人 | | | | | 邵欣欣 | |
| 单元名称 | | | 代码混淆 | | | | 责任人 | | | 邵欣欣 | | 单元所属子系统 | | | | | | 代码混淆 | | 开发周期 | | | | 1个月 |
| 代码测试检查： | | | | | | | | | | | | | | | | | | | | | | | | |
| 代码测试内容 | | | | | 测试人员 | | | | | 测试结果 | | | | | | | | | | 备注 | | | | |
| 路径测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 路径正常 | | | | |
| 声明测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 循环测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 循环正常 | | | | |
| 边界测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | - | | | | |
| 接口测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 接口正常 | | | | |
| 界面测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 界面正常 | | | | |
| 数据确认测试 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 数据确认正常 | | | | |
| 代码走查 | | | | | 邵欣欣 | | | | | 测试结果一切正常 | | | | | | | | | | 代码规范 | | | | |
| 功能测试：功能测试一切正常，满足预定功能设定 | | | | | | | | | | | | | | | | | | | | | | | | |
| 序号 | 功能名称 | | | | | 操作方法 | | | | | 结果 | | | | | 建议 | | | | 测试人员 | | 备注 | | |
| 01 | Nginx反向代理服务器向转发的响应js文件进行混淆D | | | | | 斜塞正确ID的JavaScript请求，对JavaScript脚本进行混淆操作 | | | | | 测试结果一切正常 | | | | |  | | | | 邵欣欣 | |  | | |
| 测试结论 | | | | 本模块通过单元测试 | | | | | | | | | | | | | | | | | | | | |
| 责任人 | | 邵欣欣 | | | | | | | | | | | 项目第一责任人 | | | | | | 邵欣欣 | | | | | |
| 审核 | |  | | | | | | | | | | | | | | | | | | | | | | |
| 项目组 | | 本项目组 | | | | | | 测试组 | 本项目组 | | | | | 总工办 | | | 邵欣欣 | | | | 总工程师 | | 邵欣欣 | |

翻译

**Webpage Encryption Based on Polymorphic Javascript Algorithm**

Bai Zhongying , Qin Jiancheng

Beijing University of Posts and Telecommunications

Beijing,China

dragon\_2k@21cn.com , [master@28x28.com](mailto:master@28x28.com)

Abstract—HTML webpages are widely used in computer networks. However, protecting the webpage content is a problem because browsers can obtain the HTML and Javascript source codes. This paper presents an algorithm of HTML code encryption based on polymorphic Javascript programs. The Javascript codes can transmute and protect themselves like polymorphic viruses. Furthermore, this HTML encryption uses compression, permutation and check digits to enhance the security effect. Based on this algorithm, a webpage encrypting shareware named “Athena WebLock” is designed. The experiment results show this encrypting algorithm can protect the webpage content in most cases, so it is valuable in the field of Data Rights Management.

Keywords:

webpageencryption;

polymorphic Javascript codes;

HTML code protection;

data rights management

I. INTRODUCTION

HTML webpages are widely used in computer networks such as Internet and Intranets. Lots of webpages are valuable and need protection. For example, some webpages ought to be prevented from the unauthorized access, copy, falsification and so on. So DRM (Data Rights Management) is needed in World Wide Web.

However, protecting the webpage content is a problem because the HTML source codes have to be downloaded by the browsers, thus the browsers have the opportunity to obtain all of the HTML content. A way to protect the content is using Javascript programs to encrypt the HTML codes, but the Javascript source codes can also be downloaded by the browsers. Crackers can modify the Javascript decrypting program to get the HTML source codes.

This paper presents an algorithm of HTML code encryption based on polymorphic Javascript programs. Executing such programs may decrypt the HTML codes and show the webpages. These Javascript codescan transmute and protect themselves like polymorphic viruses.

The key point is that both encrypted HTML codes and polymorphic Javascript programs are difficult to be cracked. So the webpage content is protected.

Furthermore, this HTML encryption uses compression, permutation and check digits to enhance the security effect.

Based on this algorithm, a webpage encrypting shareware named “Athena WebLock ” is designed. The Chinese version of this shareware can be downloaded in the website:

http://www.28x28.com/index.php?id=soft\_000000002 .

The experiment result shows this encrypting algorithm can protect the webpages in most cases. The remainder of this paper is structured as follows: Section 2 points out the difficulty of webpage

content protection. Section 3 introduces the design of webpage encryption based on polymorphic Javascript algorithm. Section 4 describes the enhancements of encryption such as compression, permutation and check digits. The shareware example and experiment result are given in section 5. Conclusions and references are given in section 6 and 7.

II. DIFFICULTY OF WEBPAGE CONTENT PROTECTION Since HTML webpages are widely used, lots of valueable data have been in World Wide Web. They need protection to defend attacks such as unauthorized access, copy, falsification and so on.

Using Javascript to decrypt the HTML codes is a hopeful DRM solution. Its advantage is that browsers can directly open the encrypted webpages without any additional client software, so this protection is easy to be used all over the World Wide Web.

But the difficulty of such protection also exists. In order to show the webpages, all HTML codes have to be downloaded by the browsers, so the browsers have the opportunity to obtain the HTML content.

The HTML codes can be encrypted by C/C++ or other programs so that the codes are not recognizable. The browsers can execute the Javascript to do the decryption. But the Javascript codes have to be downloaded and executed by the browsers, so the Javascript can also be cracked.

If the crackers can understand the Javascript codes, they can easily modify them to get the HTML source codes.

Even if the HTML source codes are not gained, crackers can also modify the encrypted HTML codes directly. Then the decrypted HTML codes are not the same as the original source codes.

Also, crackers can copy the encrypted webpages to their own websites, or even clone the whole encrypted website without any changes. How to defend such attack by the Javascript program is also a problem.

III. DESIGN OF WEBPAGE ENCRYPTING ALGORITHM To protect the webpages, we have designed the

webpage encryption based on the polymorphic Javascript algorithm.

According to our design, the way to show an encrypted webpage is:

Let the browser download the encrypted HTML and Javascript codes. The Javascript program is executed to decrypt the HTML codes. Then the browser shows the webpage.

We find others also study this way to protect webpages [1]. The difference is that we use random keys, polymorphic Javascript codes and enhancements of encryption to improve the security.

For each encryption, we use a random key. The encrypted HTML codes have the format of Base64 coding such as “flJ02RpQ9t72iATxnPMkhTBQinv4 ”. Thus it is difficult for crackers to decrypt the Base64 codes.

We design the Javascript decrypting program. And the Javascript source codes also need protection. We have written another paper presenting the polymorphic algorithm of Javascript code protection [2]. This algorithm with the name “parhelion” comes from the idea of program code obfuscators [3][4] in manner of polymorphic viruses [5].

“Parhelion” algorithm to generate polymorphic Javascripts has the following steps. More details are provided in [2].

(1) Standardize the Javascript source codes.

S1 = f1 (S0)

(2) Add some fake functions and variables to the Javascript source codes.

S2 = S1 + C2

(3) Change the sequence of all functions and definition of global variables randomly.

S3 = f3 (S2, V3)

(4) Generate the standard name.

C4 = A0 + Ai

( A0 {O,l}, Ai {O,l,0,1}, i = 1,2,…,n-1 )

(5) Generate other names for the functions and variables.

V5 = f5 (C4, U4)

U4 = <N1,N2,…,Nx> ( Ni=1,2,…,n-1; i=1,2,…,x )

(6) Replace all names in the codes.

S6 = f6 (S3, V0, V5)

V0 = <P1,P2,…,Pm>, V5 = <Q1,Q2,…,Qm> “Parhelion” algorithm ensure the security of

Javascript codes. Then we can focus on the algorithm of HTML code encryption.

This paper just discusses encrypting the HTML codes with a single key. We can also fit this encryption into the public key system if we need it. And we ignore the discussion of key management.

Using standard algorithm such as DES or AES to encrypt the HTML codes is feasible. And trying our own algorithm with the name “Athena ” is also significative. We may make some enhancements to improve the security.

IV. ENHANCEMENTS OF ENCRYPTION

In our design, the HTML codes are encrypted by C/C++ or Java programs, and decrypted by Javascripts. In section 2 we have discussed the difficulty of webpage protection. In the design of “Athena ” algorithm, we consider the ways to defend the attacks.

(1) Compression.

Compressing the HTML codes can reduce the data redundancy and destroy the original code length, so cracking will be more difficult. We use LZSS algorithm [6] to compress the HTML codes.

The formula of enhancement (1) is:

B1 = g1 (H0)

H0 is the original HTML code string. g1 is the compressing function.

(2) Check digits.

Appending check digits to the binary codes is an effective way to defend the copy and falsification attacks. Both binary codes and check digits are encrypted, so crackers cannot make correct check digits before they get the decrypted codes. If the cracker modify the encrypted codes blindly, the check digits will be incorrect and the Javascript decrypting program will stop working.

We also put the hash value of the webpage URL (Uniform Resource Locator) into check digits. Thus copying the webpage to another URL will get an invalid webpage.

The formula of enhancement (2) is:

B2 = B1 + g2 (B1 + U)

g2 is the hash function to generate check digits. U is the webpage URL.

(3) Permutation.

In discrete mathematics, permutation is a function f:X X . X is a set. The core encryption may be regarded as an operation in the permutation group. This operation can change a binary string into another one. In the single key system, permutation is familiar.

The formula of enhancement (3) is:

B3 = g3 (B2, K)

g3 is the function of permutation. K is the random key.

(4) Base64 codes.

After the encryption enhancement (1) to (3), the HTML source codes are changed into binary data. Then we change them into Base64 codes so that Javascripts can recognize them. So the encrypted webpages are mainly made up of Base64 codes.

The formula of enhancement (4) is:

B4 = g4 (B3)

g4 is the function of Base64 encoding.

In section 3, “Parhelion ” algorithm generates the polymorphic Javascript codes S6. In this section, “Athena ” algorithm generates the encrypted HTML codes B4. A complete webpage encryption is finished.

On the other hand, the HTML codes of the two demo webpages were cracked 5 times. But it is not because “Athena ” encrypting algorithm is weak. The reason is that the decrypting Javascript program is not strong enough. Each time the demo webpages were cracked, we upgraded the Javascript program later.

Recently we found some powerful Javascript debugger can get the HTML content directly. But we have new idea to deal with it. And we will upgrade the program again.

V. SHAREWARE AND EXPERIMENT RESULT

Based on “parhelion” and “Athena ” algorithms, we develop the webpage encrypting shareware “Athena WebLock” with C/C++ language. The simplified Chinese version can be downloaded in the following webpage:

http://www.28x28.com/index.php?id=soft\_000000002 There are two encrypted demo webpages:

http://www.28x28.com/doc/demo1.html

http://www.28x28.com/doc/demo2.html

In “Athena WebLock ”, “Parhelion” algorithm protects the Javascript codes in the webpages, and “Athena ” algorithm protects the HTML codes. Fig. 1 shows part of the encrypted HTML codes. And Fig. 2 shows part of the original source HTML codes. We can see the codes are throughly changed.

We fetch HTML files from some hompages to do the encrypting experiment. Tab. 1 and Fig. 3 shows the HTML file size changes before and after encryption. The results show that “Athena” algorithm brings about 50% compression ratio. Compression makes the encryption stronger.

“Athena WebLock” is downloaded near 36,000 times. Those two demo webpages have been in the website for over two years. Nobody declares the encrypted Javascript codes have been cracked. That means “Parhelion” algorithm is not proved to be insecure.



Figure 1 Encrypted HTML codes sample

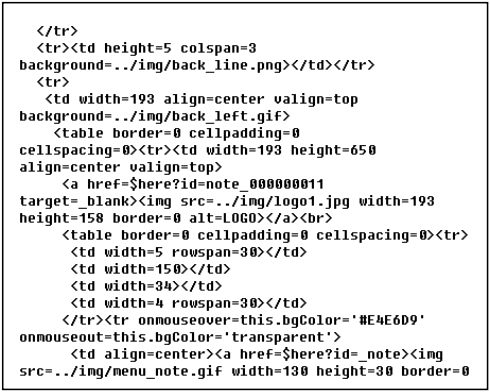
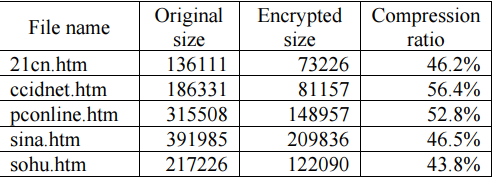


Figure 2 Original HTML codes sample

TABLE 1. HTML FILE SIZE (BYTES)



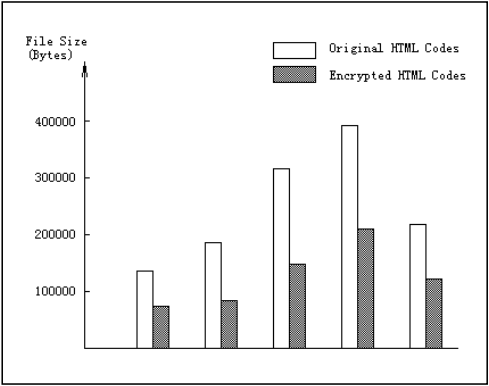


Figure 3 HTML file size changes

VI. CONCLUSIONS

This paper provides a way to solve this problem. “Athena ” algorithm can encrypt the HTML codes, and “parhelion” algorithm can protect the Javascript decrypting program. The two algorithms cooperate to keep the webpage security.

In practice, the shareware “Athena WebLock ” is developed. The experiment results show this encryption can protect the webpages in most cases.

Furthermore, we have the idea of upgrading the Javascript program to prevent the Javascript debugger. The debugger will not get the complete webpage. This is our future work.

REFERENCES

[1] Moser, and Marvin, “Method, system, and apparatus for encrypting a web browser script”, EI village , http://www.engineeringvillage2.org.cn/ , 7 Nov 2002.

[2] Qin Jiancheng, Bai Zhongying, and Bai Yuan, “Polymorphic Algorithm of JavaScript Code Protection”, ISCSCT '08, International Symposium on Volume 1, 22 Dec 2008, pp: 451 – 454.

[3] Nomair A. Naeem, Michael Batchelder, and Laurie Hendren, “Metrics for Measuring the Effectiveness of Decompilers and Obfuscators ”, ICPC’07, 26-29 June 2007, pp: 253 – 258.

[4] T.W. Hou, H.Y. Chen, and M.H. Tsai, “Three control flow obfuscation methods for Java software”, IEE Proceedings, April 2006, Vol. 153, pp: 80 – 86.

[5] John A. Adam, “Threats and countermeasures”, IEEE Press, 445 Hoes Lane PO Box 1331 Piscataway, NJ USA. 1992, pp: 21 – 28.

[6] James A. Storer, and Thomas G. Szymanski, “Data compression via textual substitution”, Journal of the ACM (JACM), Oct 1982, Vol. 29, Issue

**基于多态Javascript算法的网页加密**

摘要：HTML网页广泛用于计算机网络。 但是，网页内容的保护是一个棘手的问题，因为浏览器可以获取HTML和Javascript源代码。 本文提出了一种基于多态Javascript程序的HTML代码加密算法。 Javascript代码可以像多态病毒一样转化和保护自己。 此外，此HTML加密使用压缩，置换和校验数字来增强安全性效果。 基于该算法，设计了一个名为“Athena WebLock”的网页加密共享软件。 实验结果表明，该加密算法在大多数情况下可以保护网页内容，因此在数据权限管理领域具有重要价值

关键词：网页加密; 多态Javascript代码; HTML代码保护; 数据权利管理

一、导论

HTML网页广泛用于诸如因特网和内联网之类的计算机网络中。很多网页很有价值，需要保护。例如，某些网页应该被禁止进行未经授权的访问，复制，伪造等。因此万维网需要DRM（数据权限管理）。

然而，保护网页内容是一个问题，因为HTML源代码必须由浏览器下载，因此浏览器有机会获得所有HTML内容。保护内容的一种方法是使用Javascript程序加密HTML代码，但浏览器也可以下载Javascript源代码。破解者可以修改Javascript解密程序以获取HTML源代码。

本文提出了一种基于多态Javascript程序的HTML代码加密算法。执行这样的程序可以解密HTML代码并显示网页。这些Javascript代码可以像多态病毒一样转化和保护自己。

关键是加密的HTML代码和多态Javascript程序都难以破解。因此，网页内容受到保护。此HTML加密使用压缩，置换和校验位来增强安全性。基于此算法，设计了一个名为“Athena WebLock”的网页加密共享软件。该共享软件的中文版可以在以下网站下载：http：//www.28x28.com/index.php？id = soft\_000000002。实验结果表明，该加密算法在大多数情况下可以保护网页。本文的其余部分结构如下：第2节指出了网页内容保护的难度。第3节介绍了基于多态Javascript算法的网页加密设计。第4节描述了加密的增强功能，如压缩，置换和校验位。共享软件示例和实验结果在第5节中给出。结论和参考文献在第6节和第7节中给出。

二、网页内容保护的难点

由于HTML网页被广泛使用，因此万维网中有许多有价值的数据。 他们需要保护来防御未经授权的访问，复制，伪造等攻击。

使用Javascript解密HTML代码是一个有希望的DRM解决方案。 它的优点是浏览器可以直接打开加密的网页而无需任何其他客户端软件，因此这种保护很容易在万维网上使用。

但是这种保护的困难也存在。 为了显示网页，浏览器必须下载所有HTML代码，因此浏览器有机会获取HTML内容。

HTML代码可以通过C / C ++或其他程序加密，这样代码就无法识别。浏览器可以执行Javascript来进行解密。但是Javascript代码必须由浏览器下载和执行，因此Javascript也可以被破解。

如果破解者可以理解Javascript代码，他们可以轻松修改它们以获取HTML源代码。

即使没有获得HTML源代码，破解者也可以直接修改加密的HTML代码。然后解密的HTML代码与原始源代码不同。

此外，破解者可以将加密的网页复制到他们自己的网站，甚至可以克隆整个加密的网站而不做任何更改。如何通过Javascript程序防御此类攻击也是一个问题。

三、网页加密算法的设计为了保护网页，我们设计了网页

基于多态Javascript算法的网页加密。

根据我们的设计，显示加密网页的方式是：

让浏览器下载加密的HTML和Javascript代码。执行Javascript程序以解密HTML代码。然后浏览器显示网页。

我们发现其他人也研究这种方式来保护网页[1]。不同之处在于我们使用随机密钥，多态Javascript代码和加密增强来提高安全性。

对于每个加密，我们使用随机密钥。加密的HTML代码具有Base64编码的格式，例如“flJ02RpQ9t72iATxnPMkhTBQinv4”。因此，破解者很难解密Base64代码。

我们设计了Javascript解密程序。而且Javascript源代码也需要保护。我们写了另一篇论文，介绍了Javascript代码保护的多态算法[2]。这个名为“parhelion”的算法来自程序代码混淆器[3] [4]的多态病毒[5]。

用于生成多态Javascripts的“Parhelion”算法具有以下步骤。更多细节见[2]。

（1）标准化Javascript源代码。

S1 = f1（S0）

（2）在Javascript源代码中添加一些假函数和变量。

S2 = S1 + C2

（3）随机改变所有函数的顺序和全局变量的定义。

S3 = f3（S2，V3）

（4）生成标准名称。

C4 = A0 + Ai

（A0 {O，l}，Ai {O，l，0,1}，i = 1,2，...，n-1）

（5）为函数和变量生成其他名称。

V5 = f5（C4，U4）

U4 = <N1，N2，...，Nx>（Ni = 1,2，...，n-1; i = 1,2，...，x）

（6）替换代码中的所有名称。

S6 = f6（S3，V0，V5）

V0 = <P1，P2，...，Pm>，V5 = <Q1，Q2，...，Qm>“Parhelion”算法确保安全性

Javascript代码。然后我们可以专注于HTML代码加密的算法。

本文仅讨论使用单个密钥加密HTML代码。如果需要，我们还可以将此加密适用于公钥系统。我们忽略了关键管理的讨论。

使用诸如DES或AES的标准算法来加密HTML代码是可行的。尝试使用名为“Athena”的自己的算法也很有意义。我们可能会进行一些改进以提高安全性。

IV.增强

在我们的设计中，HTML代码由C / C ++或Java程序加密，并由Javascripts解密。在第2节中，我们讨论了网页保护的难度。在“Athena”算法的设计中，我们考虑了防御攻击的方法。

（1）压缩。

压缩HTML代码可以减少数据冗余并破坏原始代码长度，因此破解将更加困难。我们使用LZSS算法[6]来压缩HTML代码。

增强（1）的公式是：

B1 = g1（H0）

H0是原始的HTML代码字符串。 g1是压缩功能。

（2）检查数字。

将校验位附加到二进制代码是防止复制和伪造攻击的有效方法。二进制代码和校验位都是加密的，因此破解者在获得解密代码之前无法生成正确的校验位。如果破解者盲目修改加密代码，则校验位数将不正确，Javascript解密程序将停止工作。

我们还将网页URL（统一资源定位符）的哈希值放入校验位。因此，将网页复制到另一个URL将获得无效的网页。

增强（2）的公式是：

B2 = B1 + g2（B1 + U）

g2是生成校验位的哈希函数。 U是网页网址。

（3）排列。

在离散数学中，置换是函数f：X X. X是一组。核心加密可以被视为置换组中的操作。此操作可以将二进制字符串更改为另一个字符串。在单一密钥系统中，排列是熟悉的。

增强（3）的公式是：

B3 = g3（B2，K）

g3是置换的功能。 K是随机密钥。

（4）Base64代码。

在加密增强（1）到（3）之后，HTML源代码被改变为二进制数据。然后我们将它们更改为Base64代码，以便Javascripts可以识别它们。因此加密的网页主要由Base64代码组成。

增强（4）的公式是：

B4 = g4（B3）

g4是Base64编码的功能。

在第3节中，“Parhelion”算法生成多态Javascript代码S6。在本节中，“Athena”算法生成加密的HTML代码B4。完成完整的网页加密。

另一方面，两个演示网页的HTML代码被破解了5次。但这不是因为“Athena”加密算法很弱。原因是解密Javascript程序不够强大。每次演示网页破解时，我们都会在以后升级Javascript程序。

最近我们发现一些强大的Javascript调试器可以直接获取HTML内容。但我们有新的想法来处理它。我们将再次升级该计划。

V.SHAREWARE和实验结果

基于“parhelion”和“Athena”算法，我们使用C / C ++语言开发网页加密共享软件“Athena WebLock”。简体中文版可以在以下网页下载：

http://www.28x28.com/index.php?id=soft\_000000002有两个加密的演示网页：

http://www.28x28.com/doc/demo1.html

http://www.28x28.com/doc/demo2.html

在“Athena WebLock”中，“Parhelion”算法保护网页中的Javascript代码，“Athena”算法保护HTML代码。图1显示了部分加密的HTML代码。图2显示了原始源HTML代码的一部分。我们可以看到代码彻底改变了。

我们从一些hompages中获取HTML文件来进行加密实验。标签。图1和图3显示了加密前后的HTML文件大小变化。结果表明，“Athena”算法带来了约50％的压缩比。压缩使加密更强大。

“Athena WebLock”的下载量接近36,000次。这两个演示网页已经在网站上存在了两年多。没有人声明加密的Javascript代码已被破解。这意味着“Parhelion”算法并未被证明是不安全的

另一方面，两个演示网页的HTML代码被破解了5次。 但这不是因为“Athena”加密算法很弱。 原因是解密Javascript程序不够强大。 每次演示网页破解时，我们都会在以后升级Javascript程序。

最近我们发现一些强大的Javascript调试器可以直接获取HTML内容。 但我们有新的想法来处理它。 我们将再次升级该计划。



图1.加密的HTML代码示例

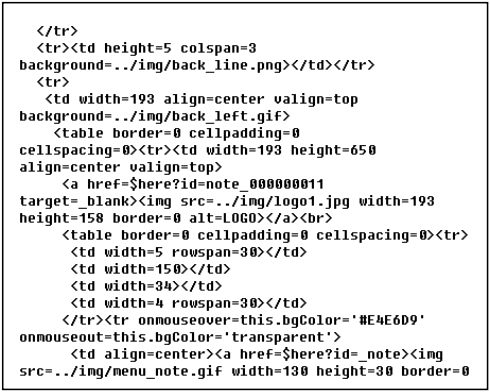
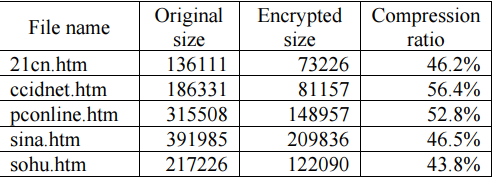


图2.原始HTML代码示例

表1. HTML文件大小（BYTES）



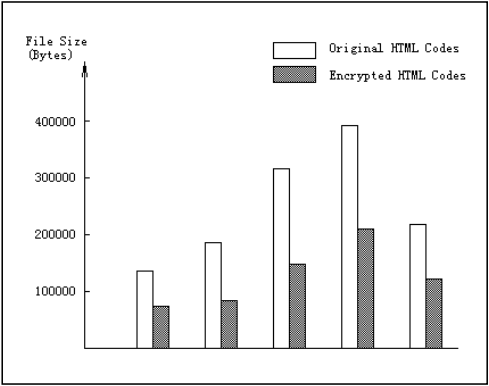


 图3. HTML文件大小更改

六、结论

网页加密很有价值，但实际上保持安全并不容易。 保护网页的问题是所有HTML和Javascript代码都必须下载到浏览器中，因此浏览器有机会破解。

本文提供了一种解决这个问题的方法。 “Athena”算法可以加密HTML代码，“parhelion”算法可以保护Javascript解密程序。 这两种算法相互配合以保持网页安全。

在实践中，开发了共享软件“Athena WebLock”。 实验结果表明，在大多数情况下，此加密可以保护网页。

此外，我们有想法升级Javascript程序以防止Javascript调试器。 调试器无法获得完整的网页。 这是我们未来的工作。

参考文献

[1] Moser, and Marvin, “Method, system, and apparatus for encrypting a web browser script”, EI village, http://www.engineeringvillage2.org.cn/, 7 Nov 2002.

[2] Qin Jiancheng, Bai Zhongying, and Bai Yuan, “Polymorphic Algorithm of JavaScript Code Protection”, ISCSCT '08, International Symposium on Volume 1, 22 Dec 2008, pp: 451 – 454.

[3] Nomair A. Naeem, Michael Batchelder, and Laurie Hendren, “Metrics for Measuring the Effectiveness of Decompilers and Obfuscators ”, ICPC’07, 26-29 June 2007, pp: 253 – 258.

[4] T.W. Hou, H.Y. Chen, and M.H. Tsai, “Three control flow obfuscation methods for Java software”, IEE Proceedings, April 2006, Vol. 153, pp: 80 – 86.

[5] John A. Adam, “Threats and countermeasures”, IEEE Press, 445 Hoes Lane PO Box 1331 Piscataway, NJ USA. 1992, pp: 21 – 28. [6] James A. Storer, and Thomas G. Szymanski, “Data compression via textual substitution”, Journal of the ACM (JACM), Oct 1982, Vol. 29, Issue 4.

**Polymorphic Algorithm of JavaScript Code Protection**

Jiancheng Qin

Beijing University of Posts and Telecommunications

Beijing , China

dragon\_2k@21cn.com

[master@28x28.com](mailto:master@28x28.com)

Abstract —As a common language in the websites, Javascript helps to make various dynamic webpages. But it’s more difficult to protect Javascript source codes than to protect Java or C/C++ programs, because Javascript codes can’t be compiled into byte or binary codes. This paper discusses the obfuscation method to protect source codes of script languages like Javascript, and provides a algorithm by using the reference design of the polymorphic viruses to make the random encryption. This algorithm named “parhelion” has been applied to a set of software of webpage encryption, and a web-based information security system. The paper also gives the examples to show the effect of the protection.

Keywords-Javascript obfuscators; source code protection; polymorphic viruses

I. INTRODUCTION

Zhongying Bai

Beijing University of Posts and Telecommunications

Beijing , China

Yuan Bai

Beijing University of Posts and Telecommunications

Beijing , China

snowbaiyuan@163.com

Source code protection is valuable in most situations. The source codes reveal the key ideas

of programs, and act as an important role in fields of software copyrights and information security.

Javascript is a common language in the websites. This script language helps to make various dynamic webpages. It’s more difficult to protect Javascript source codes than to protect Java or C/C++ programs, because Javascript source codes can’t be complied into byte or binary codes.

This paper discusses the obfuscation method to protect source codes of script languages like Javascript, and provides a protecting algorithm by using the reference design of the polymorphic viruses to make the random encryption.

This algorithm named “parhelion” has been applied to a set of software of webpage encryption, and a web-based information security system. The paper also gives the examples to show the effect of the code protection.

The remainder of this paper is structured as follows: Section 2 points out the difficulty of Javascript source

code protection. Section 3 discusses the idea of obfuscators in manner of polymorphic viruses. Section 4 and 5 describes an encryption algorithm design. Two application examples are given in section 6. Conclusions are given in section 7.

II. D IFFICULTY OF P ROTECTING J AVASCRIPT CODES Most programs are written in Java or C/C++ language. A

well-known way to protect these source codes is compiling them into byte or binary codes. The reverse process, decompiling them into source codes, is difficult.

Yet Javascript source codes can’t be compiled because they have to be explained and executed by the web browsers, such as Internet Explorer and Firefox. Neither binary nor byte codes can be recognized by the web browsers.

All of the web server side programs, for example, ASP, PHP, JSP and so on, are executed in the server side. Even if they are not complied, the servers give them sufficient protection. Source codes can’t be seen by the browsers. What the browsers get are just HTML codes generated by the server side programs.

Yet Javascript source codes have to be downloaded by the browsers, otherwise they can’t be executed. So, each browser has the chance to see Javascript source codes. That’s the difficulty of protecting Javascript codes, too.

III. OBFUSCATORS AND P OLYMORPHIC V IRUSES Javascript programs are valuable applications, too. The source codes need some protection in any possible way. Encryption is considerable, which can translate the original source codes into another ones.

The encrypted codes are also in the form of Javascript source codes, therefore they can be executed by the browsers. There is no difference between the original and new codes when the browsers execute them, but the new codes are difficult to be understood by human beings.

In order to design such encryption algorithm, there are helpful references: obfuscators and polymorphic viruses.

Obfuscators are popular in the field of Java codes protection [1][2]. Because the decompilers can change the Java byte codes back into source codes [3], the obfuscators are useful to make this work difficult. To some extent, the obfuscators encrypt the Java codes.

Although Javascript source codes can’t be compiled into binary or byte codes, a well-designed obfuscator can still change the source codes into hard-to-understand codes.

Polymorphic viruses have the ability of changing themselves randomly [4]. This is a self-protection characteristic. It helps the polymorphic viruses to avoid the anti-virus softwares. In some sense, the codes of polymorphic viruses are encrypted by themselves.

If the Javascript source codes have this ability, so that each time the browser downloads different random codes, it’s more difficult to get the original codes.

IV. ENCRYPTION ALGORITHM DESIGN

The encryption algorithm is named “parhelion” by the paper authors. It uses the following ways to protect Javascript source codes:

(1) The names of functions and variables are changed into random strings. For example, the Javascript code

“var myName,myAddress;”

may be changed into

“var ll0lllOl, ll000llO;”

so that the encrypted codes are like compiled machine language codes.

(2) The sequence of defining functions and variables is randomly changed. For example, the Javascript codes

“var a; var b; function c() {…} function d() {…}”

may be changed into

“function d() {…} var b; function c() {…} var a;”

and finally into

“function Ol000llO() {…} var ll0lllOl; function Ol0lllOl() {…} var ll000llO;”

so that the encrypted codes have fully different location.

(3) Some special Javascript source codes are changed into encrypted strings. For example, “document.write” may be changed into “e355b6104c2019aa8e12faa1ec51e44e”, so that it’s hidden in the whole encrypted codes. When the special code is needed, decrypt the string and use the function “eval()” instead.

(4) Fake functions and variables are mixed into the encrypted codes. They can puzzle the reverse engineers. They aren’t executed by the browsers indeed, so they won’t affect the correct Javascript programs.

“Parhelion” algorithm can be implemented by C/C++ or web server side programs such as ASP, PHP, JSP and so on. The main description of “parhelion” algorithm is given below:

All names of Javascript functions and variables are changed into strings made up of characters “0”, “1”, “O” and “l”, and the first character should be “O” or “l”. The strings should be long enough. Typically, “parhelion” algorithm uses 16-character strings. Then, a name in the encrypted codes may have the appearance like “lll1OOO01l0llOl1”.

Furthermore, in order to make all names alike and difficult to be distinguished, “parhelion” algorithm uses a random string as the standard name, and changes its characters randomly to generate new names.

V. ENCRYPTION ALGORITHM DESCRIPTION

The algorithm has the following steps:

(1) Standardize the Javascript source codes. Each program has one and only one entrance like the “main()” function in a C/C++ program. All Javascript codes, except the definition of global variables, should be inside the functions.

If there are any executable codes outside the functions, put them into a new function, and place the entrance on it.

The formulate of step (1) is:

S1 = f1 (S0 )

S0 is the original source codes. f1 is the standardizing function.

(2) Add some fake functions and variables to the Javascript source codes. Generate some random codes in the fake functions, or copy real function codes. For example, the following codes are added:

“var fake1,fake2; function fake3() {…}”

The formulate of step (2) is:

S2 = S1 + C2

C2 is the random fake codes. The operator “+” is connecting two strings.

(3) Change the sequence of all functions and definition of global variables randomly. Place all initialization codes at the beginning of the entrance.

The browsers accept random sequence definition of Javascript functions. They can execute the codes from the entrance, and follow the program flow.

The formulate of step (3) is:

S3 = f3 (S2 , V3)

f3 is the confusing function. V3 is the random sequence vector.

(4) Generate the standard name. It ’s a 16-character random string mentioned above. For example, “lll1OOO01l0llOl1” is generated.

The formulate of step (4) is:

C4 = A0 + ∑ Ai

( A0∈{O,l}, Ai∈{O,l,0,1}, i = 1,2,…,n-1 )

Ai is the character of No. i. The operator “+” and “∑” are connecting strings. n=16.

(5) Generate other names for the functions and variables. Each character in the standard name can be changed, except the first one. For example, the 2 character of the standard name is “l ”, so it can be changed into “O”, “0” or “1”. If the 2 , 5 and 8 characters are changed, “lll1OOO01l0llOl1” may become “l1l11OOO1l0llOl1”. These two strings are alike.

In order to generate all names of a Javascript program, from the standard name, randomly choose x characters to be changed. The value of x is between 1 and 15. It depends on the amount of all names. The maximum amount is:

m =3

For example, If 3 characters are changed, 27 names can be generated. If 4 characters are changed, 81 names can be generated.

Use a vector to store the generated names.

The formulate of step (5) is:

V5 = f5 (C4, U4 )

U4 = <N1,N2,…,Nx> ( Ni =1,2,…,n-1; i=1,2,…,x )

f5 is the function that changes C4 into a vector with m elements. Each element is a string like C4. U4 is a vector storing the selection. For example, U4=<2,5,8>.

(6) Replace all names in the codes. Scan the codes once. When a new name appears, get a new name from the array and replace it. If the current name is met before, replace it with what it was replaced.

The formulate of step (6) is:

S6 = f6 (S3 , V0, V5)

V0 = <P1,P2,…,Pm>, V5 = <Q1,Q2,…,Qm>

f6 is the string-replacing function. V0 is the vector storing original names P 1 to P m. V5 is the vector storing new names Q1 to Qm.

After using this encryption algorithm, from step (1) to (6), the Javascript source codes (S0) are changed into random ones (S6), just like the polymorphic virus codes. The difference is that polymorphic viruses are usually programmed by the assembly language, and they are in the M3 level of the computer multi-level architecture, while Javascript is a high-level language, and the encrypted codes are in the M4 level.

Because this encryption algorithm only changes the appearance of Javascript source codes, and doesn’t affect their original action, the browsers can still download and execute the encrypted codes correctly.

The original Javascript source codes (S0) and the encrypted codes (S6) are comparable. The string length:

L(S6 ) = L(S0) + L(C2 ) + ∑ Mi (L(Qi ) - L(Pi ))

= L(S0 ) + L(C2 ) + ∑ Mi (n - L(Pi))

( i=1,2,…,m )

Mi is the times of each name Pi appears in S0. n=16. The program complexity:

O(S0) = O(S6 )

Because from S0 to S6, the names are replaced, the definition sequence is changed, and some fate codes are added, but the control flow is unchanged. In fact S0 and S6 are the same program with different appearance.

VI. APPLICATION EXAMPLES

“Parhelion” algorithm has been applied to the practice projects. The following are two examples carried out by the paper authors:

A. “Athena ” WebLock

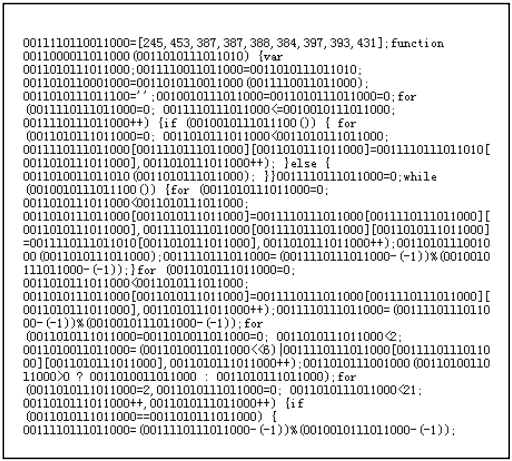
This is a set of webpage encrypting software. It’s a set of shareware made by C language. It can encrypt the HTML source codes in the webpages. This software has simplified Chinese versions. It can be downloaded in the following webpage:

http://www.28x28.com/index.php?id=soft\_000000002 There are two encrypted demo webpages:

http://www.28x28.com/doc/demo1.html

http://www.28x28.com/doc/demo2.html

“Parhelion” algorithm in this software protects the Javascript codes in the HTML pages. Fig. 1 shows part of the encrypted Javascript codes.



Figure\_ 1 Encrypted Javascript codes

“Athena” WebLock is downloaded over 24,000 times. Those two demo webpages have been in the website for over two years. Nobody declares the encrypted Javascript codes have been cracked.

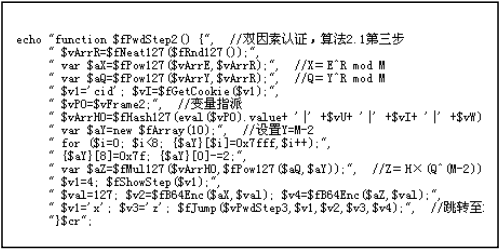
B. “Hephaestus” information system

This is a web-based information system with multi-layer deep-defense architecture. This system is made by PHP, MySQL and C language. It’s a customized information system for different organizations. There is a simplified Chinese demo system:

http://edu.co.2288.org/bg/

“Parhelion” algorithm in this system also protects the Javascript codes. Some corporations have used this system for over 5 years, and the system has withstood over 100,000 attacks successfully.

Fig. 2 shows part of the PHP codes with “parhelion” algorithm, which generate random encrypted Javascript codes shown in Fig. 1.



Figure\_ 2 PHP “parhelion ” codes.

VII. CONCLUSIONS

The problem of protecting Javascript source codes is that Javascript codes have to be downloaded and executed by the browsers, so they can’t be compiled into binary or byte codes. This paper gives a way to solve this problem. “Parhelion” algorithm can encrypt the Javascript source codes randomly, in manner of obfuscators and polymorphic viruses.

In practice, the webpage encryption software and the web-based information system prove that “parhelion” algorithm can protect Javascript codes efficiently.

REFERENCES

[1] Nomair A. Naeem, Michael Batchelder, and Laurie Hendren, “Metrics for Measuring the Effectiveness of Decompilers and Obfuscators ”, Program Comprehension, 2007. ICPC’07. 15th IEEE International Conference, 26-29 June 2007, pp: 253 – 258.

[2] T.W. Hou, H.Y. Chen, and M.H. Tsai, “Three control flow obfuscation methods for Java software”, Software, IEE Proceedings, April 2006, Vol. 153, pp: 80 – 86.

[3] Nomair A. Naeem, and Laurie Hendren, “Programmer-friendly Decompiled Java”, Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference. 14-16 June 2006, pp: 327 – 336.

[4] John A. Adam, “Threats and countermeasures”, IEEE Press, 445 Hoes Lane PO Box 1331 Piscataway, NJ USA. 1992, pp: 21 – 28

JavaScript代码保护的多态算法

摘要 - 作为网站的通用语言，Javascript有助于制作各种动态网页。但保护Javascript源代码比保护Java或C / C ++程序更困难，因为Javascript代码无法编译成字节或二进制代码。本文讨论了保护Javascript等脚本语言源代码的混淆方法，并利用多态病毒的参考设计进行了随机加密。这种名为“parhelion”的算法已应用于一套网页加密软件和一个基于Web的信息安全系统。本文还给出了示例来说明保护的效果。

关键词 - Javascript混淆器;源代码保护;多态病毒

一，导言

在大多数情况下，源代码保护很有价值。源代码揭示了程序的关键思想，并在软件版权和信息安全领域发挥着重要作用。

Javascript是网站中的通用语言。此脚本语言有助于制作各种动态网页。保护Javascript源代码比保护Java或C / C ++程序更困难，因为Javascript源代码无法编译成字节或二进制代码。

本文讨论了保护Javascript等脚本语言源代码的混淆方法，并利用多态病毒的参考设计进行了随机加密的保护算法。

这种名为“parhelion”的算法已应用于一套网页加密软件和一个基于Web的信息安全系统。本文还给出了示例来说明代码保护的效果。

本文的其余部分结构如下：第2节指出了Javascript源代码的难度

代码保护。第3节以多态病毒的方式讨论了混淆器的概念。第4节和第5节描述了加密算法设计。第6节给出了两个应用实例。第7节给出了结论

II. D编写J AVASCRIPT代码的问题大多数程序都是用Java或C / C ++语言编写的。一个

众所周知的保护这些源代码的方法是将它们编译成字节或二进制代码。反向过程很难将它们反编译成源代码。

然而，Javascript源代码无法编译，因为它们必须由Web浏览器（如Internet Explorer和Firefox）进行解释和执行。 Web浏览器无法识别二进制代码和字节代码。

所有Web服务器端程序，例如ASP，PHP，JSP等，都在服务器端执行。即使它们没有被遵守，服务器也会给予足够的保护。浏览器无法看到源代码。浏览器获得的只是服务器端程序生成的HTML代码。

然而，浏览器必须下载Javascript源代码，否则无法执行。因此，每个浏览器都有机会看到Javascript源代码。这也是保护Javascript代码的难度。

三、 OBFUSCATORS和P OLYMORPHIC V IRUSES Javascript程序也是有价值的应用程序。该源代码需要以任何可能的方式进行一些保护。加密相当可观，可以将原始源代码转换为另一种源代码。

加密代码也是Javascript源代码的形式，因此它们可以由浏览器执行。当浏览器执行它们时，原始代码和新代码之间没有区别，但新代码很难被人类理解。

为了设计这样的加密算法，有一些有用的参考：混淆器和多态病毒。

混淆器在Java代码保护领域很流行[1] [2]。因为反编译器可以将Java字节代码更改回源代码[3]，所以混淆器对于使这项工作变得困难很有用。在某种程度上，混淆器会加密Java代码。

虽然Javascript源代码无法编译成二进制或字节代码，但精心设计的混淆器仍然可以将源代码更改为难以理解的代码。

多态病毒具有随机改变自身的能力[4]。这是一种自我保护的特征。它有助于多态病毒避免使用反病毒软件。在某种意义上，多态病毒的代码本身是加密的。

如果Javascript源代码具有此功能，那么每次浏览器下载不同的随机代码时，获取原始代码就更加困难。

IV。加密算法设计

加密算法被论文作者命名为“parhelion”。它使用以下方法来保护Javascript源代码：

（1）函数和变量的名称被更改为随机字符串。例如，Javascript代码

“var myName，myAddress;”

可能会变成

“var ll0lllOl，ll000llO;”

这样加密的代码就像编译的机器语言代码。

（2）定义函数和变量的顺序是随机改变的。例如，Javascript代码

“var a; var b; function c（）{...} function d（）{...}“

可能会变成

“function d（）{...} var b; function c（）{...} var a;“

最后进入

“function Ol000llO（）{...} var ll0lllOl; function Ol0lllOl（）{...} var ll000llO;“

这样加密的代码就有了完全不同的位置。

（3）一些特殊的Javascript源代码被更改为加密字符串。例如，“document.write”可能会更改为“e355b6104c2019aa8e12faa1ec51e44e”，因此它隐藏在整个加密代码中。当需要特殊代码时，解密字符串并使用函数“eval（）”代替。

（4）伪函数和变量混合到加密代码中。他们可以让逆向工程师感到困惑它们确实不是由浏览器执行的，因此它们不会影响正确的Javascript程序。

“Parhelion”算法可以通过C / C ++或Web服务器端程序实现，如ASP，PHP，JSP等。 “parhelion”算法的主要描述如下：

Javascript函数和变量的所有名称都被更改为由字符“0”，“1”，“O”和“l”组成的字符串，并且第一个字符应为“O”或“l”。字符串应该足够长。通常，“parhelion”算法使用16个字符的字符串。然后，加密代码中的名称可以具有类似“lll1OOO01l0llOl1”的外观。

此外，为了使所有名称相似且难以区分，“parhelion”算法使用随机字符串作为标准名称，并随机更改其字符以生成新名称。

V.加密算法描述

该算法具有以下步骤：

（1）标准化Javascript源代码。每个程序都有一个且只有一个入口，如C / C ++程序中的“main（）”函数。除了全局变量的定义之外，所有Javascript代码都应该在函数内部。

如果函数外部有任何可执行代码，请将它们放入新函数中，并将入口放在其上。

步骤（1）的表述是：

S1 = f1（S0）

S0是原始源代码。 f1是标准化功能。

（2）在Javascript源代码中添加一些假函数和变量。在伪函数中生成一些随机代码，或复制实际功能代码。例如，添加以下代码：

“var fake1，fake2; function fake3（）{...}“

步骤（2）的表述是：

S2 = S1 + C2

C2是随机假码。运算符“+”连接两个字符串。

（3）随机改变所有函数的顺序和全局变量的定义。将所有初始化代码放在入口的开头。

浏览器接受Javascript函数的随机序列定义。他们可以从入口执行代码，并遵循程序流程。

步骤（3）的表述是：

S3 = f3（S2，V3）

f3是令人困惑的功能。 V3是随机序列载体。

（4）生成标准名称。它是上面提到的16个字符的随机字符串。例如，生成“lll10000101OllOl1”。

步骤（4）的表述是：

C4 = A0 +ΣAi

（A0∈{O，l}，Ai∈{O，l，0,1}，i = 1,2，...，n-1）

艾是第i的角色。运算符“+”和“Σ”是连接字符串。 N = 16。

（5）为函数和变量生成其他名称。除第一个字符外，可以更改标准名称中的每个字符。例如，标准名称的2个字符是“l”，因此可以将其更改为“O”，“0”或“1”。如果改变2,5和8个字符，则“l1110001101 0111”可以变为“l11111001101llOl1”。这两个字符串是相似的。

为了生成Javascript程序的所有名称，从标准名称中随机选择要更改的x字符。 x的值介于1和15之间。这取决于所有名称的数量。最高金额为：

m = 3

例如，如果更改了3个字符，则可以生成27个名称。如果更改了4个字符，则可以生成81个名称。

使用向量存储生成的名称。

步骤（5）的公式是：

V5 = f5（C4，U4）

U4 = <N1，N2，...，Nx>（Ni = 1,2，...，n-1; i = 1,2，...，x）

f5是将C4变为具有m个元素的向量的函数。每个元素都是一个像C4的字符串。 U4是存储选择的向量。例如，U4 = <2,5,8>。

（6）替换代码中的所有名称。扫描代码一次。出现新名称时，从阵列中获取新名称并替换它。如果之前满足当前名称，请将其替换为更换的名称。

步骤（6）的表述是：

S6 = f6（S3，V0，V5）

V0 = <P1，P2，...，Pm>，V5 = <Q1，Q2，...，Qm>

f6是字符串替换功能。 V0是存储原始名称P 1至P m的向量。 V5是存储新名称Q1到Qm的向量。

在使用该加密算法之后，从步骤（1）到（6），Javascript源代码（S0）被改变为随机的（S6），就像多态病毒代码一样。不同之处在于多态病毒通常由汇编语言编程，并且它们处于计算机多级架构的M3级别，而Javascript是高级语言，加密代码是M4级别。

由于此加密算法仅更改Javascript源代码的外观，并且不影响其原始操作，因此浏览器仍可正确下载和执行加密代码。

原始Javascript源代码（S0）和加密代码（S6）是可比较的。字符串长度：

L（S6）= L（S0）+ L（C2）+Σmi（L（Qi） - L（Pi））

= L（S0）+ L（C2）+ΣMi（n - L（Pi））

（i = 1,2，...，m）

Mi是每个名称Pi出现在S0中的时间。 N = 16。程序复杂性：

O（S0）= O（S6）

因为从S0到S6，名称被替换，定义序列被改变，并且添加了一些命运代码，但是控制流程没有改变。实际上S0和S6是具有不同外观的相同程序。

VI．应用示例

“Parhelion”算法已应用于实践项目。以下是论文作者的两个例子：

A.“Athena”WebLock

这是一组网页加密软件。它是由C语言制作的一组共享软件。它可以加密网页中的HTML源代码。该软件简化了中文版本。它可以在以下网页下载：

http://www.28x28.com/index.php?id=soft\_000000002有两个加密的演示网页：

http://www.28x28.com/doc/demo1.html

http://www.28x28.com/doc/demo2.html

此软件中的“Parhelion”算法保护HTML页面中的Javascript代码。图1显示了部分加密的Javascript代码。

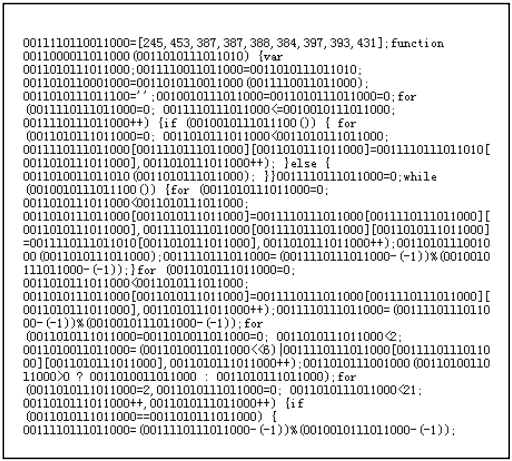


图- 1 加密的javaScript代码

“Athena”WebLock下载超过24,000次。 这两个演示网页已经在网站上存在了两年多。 没有人声明加密的Javascript代码已被破解。

B.“赫菲斯托斯”信息系统

这是一个基于Web的信息系统，具有多层深度防御架构。 该系统由PHP，MySQL和C语言编写。 它是针对不同组织的定制信息系统。 有一个简化的中文演示系统：

http://edu.co.2288.org/bg/

该系统中的“Parhelion”算法也保护Javascript代码。 一些公司已使用该系统超过5年，该系统已经成功抵御了超过100,000次攻击。

图2显示了部分带有“parhelion”算法的PHP代码，它生成了图1所示的随机加密Javascript代码。

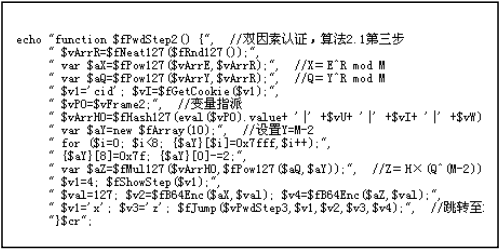


图- 2 PHP “parhelion” codes

七、结论

保护Javascript源代码的问题是Javascript代码必须由浏览器下载和执行，因此它们不能编译成二进制或字节代码。 本文给出了解决这个问题的方法。 “Parhelion”算法可以以混淆器和多态病毒的方式随机加密Javascript源代码。

实际上，网页加密软件和基于网络的信息系统证明“parhelion”算法可以有效地保护Javascript代码。

参考文献

[1] Nomair A. Naeem, Michael Batchelder, and Laurie Hendren, “Metrics for Measuring the Effectiveness of Decompilers and Obfuscators ”, Program Comprehension, 2007. ICPC’07. 15th IEEE International Conference, 26-29 June 2007, pp: 253 – 258.

[2] T.W. Hou, H.Y. Chen, and M.H. Tsai, “Three control flow obfuscation methods for Java software”, Software, IEE Proceedings, April 2006, Vol. 153, pp: 80 – 86.

[3] Nomair A. Naeem, and Laurie Hendren, “Programmer-friendly Decompiled Java”, Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference. 14-16 June 2006, pp: 327 – 336.

[4] John A. Adam, “Threats and countermeasures”, IEEE Press, 445 Hoes Lane PO Box 1331 Piscataway, NJ USA. 1992, pp: 21 – 28