一种基于预警信息的漏洞自动化快速防护方法

徐其望[1], 陈震杭[2], 彭国军[1], 张焕国[1]

[1]武汉大学空天信息安全与可信计算教育部重点实验室, 武汉大学国家网络安全学院 武汉 中国 430072
[2]北京神州绿盟信息安全科技股份有限公司 北京 中国 100089

摘要 针对当前 Web 应用防护方法无法有效应对未知漏洞攻击、性能损耗高、响应速度慢的问题, 本文从漏洞预警公告中快速提取漏洞影响范围和细节, 然后对目标系统存在风险的访问请求与缺陷文件和函数调用关系进行自动化定位, 构建正常访问模型, 从而形成动态可信验证机制, 提出并实现了基于预警信息的漏洞自动化快速防护方法, 最后以流行 PHP Web 应用的多个高危漏洞对本文方法进行验证测试, 结果表明本文方法能够自动化快速成功阻止最新漏洞攻击, 平均性能损耗仅为 5.31%。

关键词 漏洞预警; 漏洞防护; 动态可信; 调用分析; Web 安全
中图法分类号 TP393.0 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2020.01.07

A rapid and automatic vulnerability protection scheme based on warning information

XU Qiwang[1], CHEN Zhenhang[2], PENG Guojun[1], ZHANG Huanguo[1]

[1]Key laboratory of space information security and trusted computing, ministry of education, wuhan university, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
[2]Nsfocus Information Technology Co., Ltd., Beijing 100089, China

Abstract Aiming at the problem that current Web application protection schemes can not effectively deal with unknown vulnerability attacks, with high performance loss and slow response speed, this paper extracts the scope and details of vulnerabilities from vulnerability warning announcements, and then automatically locates the relationship between access requests and defective files and function calls that exist risks in the target system, and constructs a normal access model, thus forming dynamic trusted authentication mechanism, proposed and implemented a rapid vulnerability automation protection scheme based on warning information. Finally, several high-risk vulnerabilities of popular PHP Web applications were tested to verify the scheme. The results show that the scheme can automatically and successfully prevent the latest vulnerability attacks, with an average performance loss of only 5.31%.

Key words vulnerability warning; vulnerability protection; dynamic trustworthiness; invocations analysis; web security

1 引言

面对日益严峻的网络攻防形势[1], 安全厂商推出不少相应的 IDS(Intrusion Detection Systems, 入侵检测系统)和 WAF(Web Application Firewall, 网络应用防火墙)等安全防护产品。然而, 这些安全防护产品大部分都只能阻止已知的漏洞利用攻击, 既便对于最新出现的已知漏洞攻击, 也难以及时应对。在发生漏洞预警后, 如何迅速进行响应, 快速修复漏洞或者采取合适的临时处理措施, 已经成为一个亟需解决的问题。目前业界对漏洞预警和漏洞公告的处理都是交由运维部门人工分析, 应急响应的研究主要集中在系统已经发生了实际的入侵事件。Ben Stock[2]对大规模 Web 漏洞通报的可行性进行了研究, 但是漏洞信息及修复建议发送给受影响的 Web 应用所有者后, 具体修复措施还是需要人工进行, 并且在收到漏洞通知后仍有大量厂商未能及时修复漏洞, 之后 Ben Stock 进一步对该问题的原因进行了探讨[3], 但是对于厂商如何迅速响应漏洞预警并未提出根本性解决方法。连一峰等[4]提出的应急响应体系中, 安全服务层包括了发布和转发安全公告, 但是没有下一步的操作。方玉昕等[5]设计了一个针对电网的安全漏洞报告与预警平台, 但

通讯作者: 彭国军, 博士, 教授, Email: guojpeng@whu.edu.cn。
本课题得到 NSFC-通用技术基础研究联合基金(No.U1636107), 国家自然科学基金(No.61972297)资助。
收稿日期: 2019-09-06; 修改日期: 2019-10-22; 定稿日期: 2019-12-16

其中的预警模块只有收集漏洞公告的功能, 之后也是转交给运维部门进行处理。

在实际的企业应用中, 漏洞预警公告也是交由人工处理。龙海[6]提到 SAP 系统中有专门的系统预警报告工具 EWA(Early Watch Alert)会定时扫描检查系统中的组件状态, 把发现的安全问题交给运维人员。在 Joomla!的高危远程代码执行漏洞被披露出来后, 相关安全产品需要安全工程师对突发漏洞制定相应的防护策略, 只有在第一时间更新安全策略才能保护部署在云端的应用免于漏洞利用攻击的影响[7]。

但是, 依赖于安全工程师的人工分析而临时制定的防护策略可能与实际业务产生冲突, 同时也存在误报漏报的问题。此外, 人工响应漏洞预警反应滞后, 无法在第一时间进行响应。根据 Tenable Research 发布的报告, 新漏洞一旦公开, 攻击者需要花 6 天的时间(中位数)将漏洞武器化, 然而, 安全团队则可能需要 13 天的时间(中位数)才能启动对新漏洞的初步评估工作。

针对以上问题, 本文提出了一种基于预警信息的漏洞自动化防护方法, 以正常行为的调用分析和实时调用拦截作为解决问题的突破口, 以漏洞预警公告作为出发点。本方法通过定时遍历各大安全网站的漏洞公告提取预警关键信息并确定漏洞的细节和影响范围, 通过实时调用拦截可能存在攻击行为的访问请求, 结合目标系统的正常访问模型进行可信验证和响应, 并在此基础上设计实现了能够根据预警信息来对 Web 应用进行自动防护的系统原型。

本文方法的难点在于:

1) 怎样根据公开的预警信息准确定位漏洞细节以及影响范围, 同时和所防护的 Web 应用进行关联, 做好针对性的防护。

2) 怎样迅速对预警信息进行自动响应, 识别攻击行为, 同时对受漏洞影响的文件或函数调用进行实时拦截与检查, 并保持较低的性能损耗和误报率。

本文的贡献主要包括:

1) 提出了一种基于预警信息的漏洞自动化防护方法, 能够根据预警信息快速有效自动进行漏洞防护响应。

2) 提出了针对目标 Web 系统的正常访问模型自动化构建方法, 形成了动态可信验证机制, 在发生漏洞预警后可准确对缺陷类函数的调用及时进行拦截和检查, 有效降低了系统损耗。

3) 设计并实现了基于预警信息的漏洞自动化快速防护系统原型, 并用多个 PHP Web 应用漏洞验证了防护方法的可行性。

2 系统关键技术研究

2.1 预警信息的解析与提取

利用预警信息来引导网站防护, 其中的难点在于漏洞信息的理解和与保护系统及其配置的关联, 语义的多样性、关联过程的不确定性等等。本文方法使用爬虫遍历预警公告, 首先需要提取的漏洞公告中存在漏洞的应用名称和版本号。根据应用名称和版本号可以确定当前应用是否受到漏洞影响。此外, 存在漏洞的文件名或函数名以及插件名也是重要信息, 可用于缩小排查范围。一般而言, 存在漏洞的文件中包含有一个或多个有漏洞的函数插件作为 Web 应用的扩展模块, 其文件或函数中也可能出现漏洞。

在具体的处理和判断过程中, 如果漏洞公告中明确列出了所有受影响的版本号, 则只需要进行匹配即可判断是否受到影响。考虑到复杂情况, 大部分漏洞公告会给出一个影响范围, 例如“ThinkPHP 5.*”、“ <=5.0.23”等, 对于前者, 只需要将“*”视为通配符验证能否匹配防护的应用版本号, 对于“<=5.0.23”, 将防护的应用版本号与之比较即可。特别的, “ThinkPHP3.2.4 之前版本”会理解为<3.2.4 受影响, 而“ThinkPHP3.2.4 及之前版本”则会理解为<=3.2.4 受影响。另外, 为了提高漏洞细节提取的准确性, 本文方法还会对多个安全网站中提取的信息进行比较, 对不同的字段进行语义分析, 再进行归一化处理, 例如“影响产品”和“影响版本”、“before 3.2.4”和“<3.2.4”都可以理解为同样的含义。此外, 考虑到应用插件也有可能存在漏洞, 本文方法也会针对插件漏洞预警信息进行提取。

成功提取漏洞信息后, 除了通过版本号来判断要防护的应用系统是否受到漏洞影响外, 还可以直接通过检查应用系统中是否存在预警信息中提到的有漏洞的文件、函数和插件来判断是否受到漏洞影响, 从而提高漏洞响应的准确性与效率。

2.2 风险模块定位与正常访问模型构建

本文提出的基于预警信息的漏洞自动化防护方法首先需要分析 Web 应用正常运行时的各个文件、类和函数之间的调用关系, 当从漏洞预警信息中提取到存在缺陷的文件和函数信息后, 就可以找到受影响的文件和 URL, 从而确定存在缺陷的文件对于当前 Web 应用的影响范围。

本方法以类函数和敏感函数作为节点的最小单位, 在之上包括类、文件以及对应的 URL。敏感函数及其参数则用于正常访问模型的构建和训练。参

考 Victor Prokhorenko[8] 对 PUF(Potential Unsafe Function)的定义, 本文所指的敏感函数主要包括以下四类: 1)与主机文件系统进行交互的函数, 如 PHP 中的 “file_get_contents()” 、 “file_put_contents()” 、 “system()”; 2)与数据库进行交互的函数; 3)动态代码执行函数; 4)可能泄露信息的函数。

本方法需要生成 Web 应用正常运行时的行为记录文件, 使用 Python 脚本将记录文件转换为对应的节点和关系, 在 Neo4j[9]中生成对应的图。在发生漏洞预警后, 从存在漏洞的函数节点出发, 遍历调用这个函数的文件节点[10], 再查找涉及这个缺陷文件的 URL 请求, 这样就能通过调用关系分析找出所有受影响的 URL, 再单独对这些请求进行严格的参数检查。

考虑到 Web 应用的服务端会处理大量类似的 URL 请求, 但其参数不尽相同, 本文还进一步记录处理不同 URL 请求时关键的参数特别是敏感函数的参数和执行上下文环境, 记录调用敏感函数的类和类函数作为判定依据。经过一段时间的记录和分析, 系统还能够对不同敏感函数的参数根据其执行上下文提取规则, 比如某个类函数在被某些类调用时执行的文件读取函数只会读取固定的几个文件; 某个类函数在被某些类调用时执行的数据库函数只会查询固定的几个表等。这样就完成了对 Web 应用正常访问模型的构建工作。当发生漏洞预警时, 正常访问模型能够为异常行为的判定检查提供参考。

正常访问模型本质上是一个集合。该模型以敏感函数及其参数作为关注的重点, 在集合内根据正常行为记录对行为进行概况和描述, 描述某个类文件在响应某个 URL 请求时被某个文件调用执行, 调用执行了某个类函数, 类函数内还调用了某个敏感函数, 这个敏感函数的参数有一定的特征和限制。

正常访问模型通过文件结构来进行分类, 以 “URL/执行环境文件名/敏感函数名”的结构保存数据。敏感函数文件夹内就会保存所执行敏感函数的参数记录以及根据记录所生成的规则。当进行异常判定时, 先检查对应“URL/执行环境文件名/敏感函数名”文件夹是否存在, 如果不存在, 肯定是异常行为; 如果存在对应文件夹, 再读取敏感函数文件夹内的规则文件, 对参数进行检查。

以 PHP Web 应用的 CNVD-2017-04180 任意文件上传漏洞为例, 攻击者利用该漏洞时会通过 “/phpcms/index.php?m=member&c=index&a=register&siteid=1”构造一个畸形的数据请求。该攻击会调用敏感函数“copy”, 将指定的文件复制到专门的上传目录。由于过滤函数的缺陷, 导致后缀名为“.php”的文件也能被复制到上传目录, 并且会返回文件地址, 导致攻击者能够利用该漏洞上传后门和木马。在自动防护系统所构建的正常行为规范模型中, 正常行为中 “/phpcms/index.php?m=member&c=index&a=register&siteid=1”这个 URL 所对应的敏感函数操作只有“fwrite”,而且执行环境位于“client.class.php”而不是缺陷文件“attachment.class.php”, 因此借助正常行为规范模型能够识别该攻击。

2.3 访问验证与调用实时拦截

对于 Web 应用调用拦截的方式有两种。一种最直接的方式就是对 Web 应用源码进行修改, 在所有函数调用之前加入判断的代码块;另一种是编写 HOOK 函数来实现拦截调用。考虑到修改源码的时间成本较大, 不利于用户升级, 而编写 HOOK 函数兼容性较好, 不会受到应用升级的影响。所以, 本文选择通过编写 HOOK 函数的方式来进行对 Web 应用调用的实时拦截。

基于预警信息的漏洞自动化防护方法需要实现最核心的功能之一, 即拦截对缺陷文件的引入和调用, 并进行访问请求的动态验证。动态验证流程如图 1 所示:

具体来说, 在 HOOK 函数中, 先检查当前是否发现预警, 如果有预警信息才检查每个引入的文件是否属于预警信息中存在缺陷的文件。如果缺陷文件属于无关文件, 在当前的系统业务中不会涉及, 则可以直接阻止其引入和调用。但更多时候为了维持 Web 应用功能的完整性, 系统还需要拦截缺陷文件中出现问题的某个具体函数和类, 并且拦截之后对函数执行所用到的参数进行严格检查, 从而发现并阻断攻击行为。

对参数的检查策略可以通过调用关系分析所构建和训练的正常访问模型进行检查, 检查当前参数和正常运行的参数的差异性, 进行动态验证, 从而判定是否存在攻击。此外, 还可以针对预警信息中不同的漏洞类型采用不同的检查策略。例如, WordPress freshmail-newsletter 插件 CNVD-2019-38067 SQL 注入漏洞的预警公告中并没有提及存在缺陷的函数名或者文件名, 只提到“缺少对外部输入 SQL 语句的验证”, 针对这种验证不严导致的 SQL 注入漏洞, 可以检查是否存在 SQL 命令或者明显的 SQL 注入特征[11, 12]; CNVD-2018-26822 预警公告中提示 “PHPCMS 存在 xss 漏洞。攻击者可利用漏洞获取用户 cookie 等敏感信息。” 预警信息通知是 XSS 漏洞, 但是没有漏洞细节信息, 则可以检查参数中是否存在特殊字符或 JavaScript 命令[13], 等等。

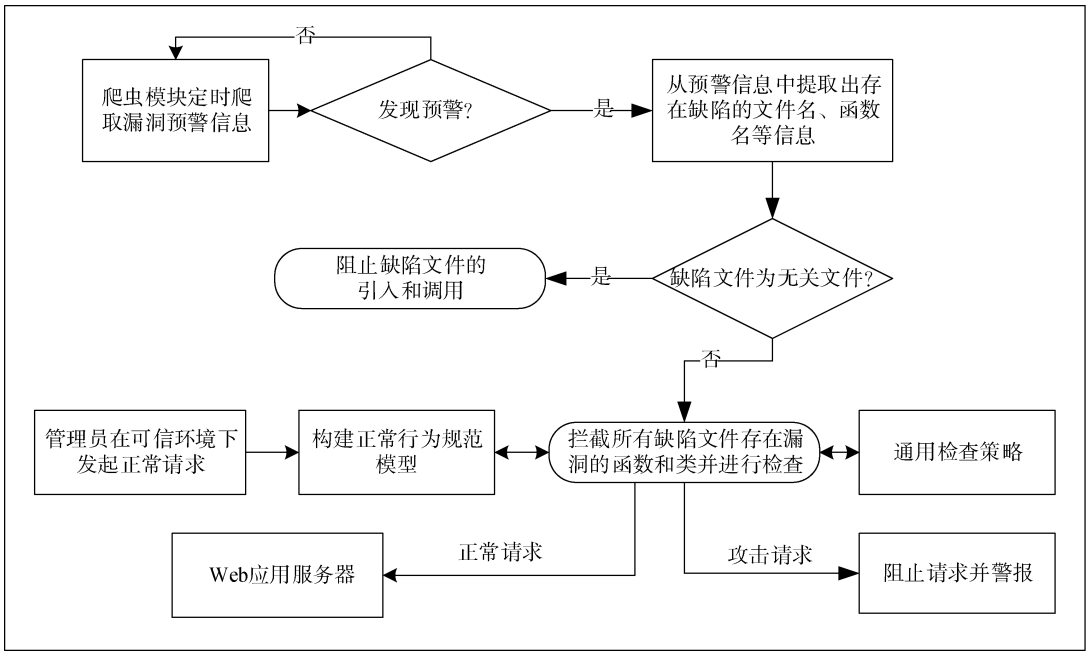

图 1　动态验证流程

Figure 1　Dynamic verification process

3　系统模块的设计和实现

3.1　系统结构及运作流程

基于上述防护方法, 本文进一步设计实现了基于预警信息的漏洞自动化防护系统, 主要由 5 个模块构成, 分别是“爬虫模块”、“负载均衡模块”、“动态可信验证模块”、“调用关系分析模块”、“报警模块”。系统由三个服务器节点构成: 反向代理服务器节点、部署自动防护系统的服务器节点、普通服务器节点。其中反向代理服务器负责定期从各大安全网站上爬取预警公告并提取关键信息, 将所有受影响的 URL 请求转发到部署有自动防护系统的服务器节点上; 普通服务器节点对正常 URL 请求进行响应, 同时阻止对缺陷文件的引用(防止攻击者从其他页面间接触发缺陷文件调用); 部署有自动防护系统的服务器节点负责构建正常访问模型, 同时对所有受影响的 URL 请求进行检查, 识别出攻击请求并拦截。整体设计如图 2 所示。

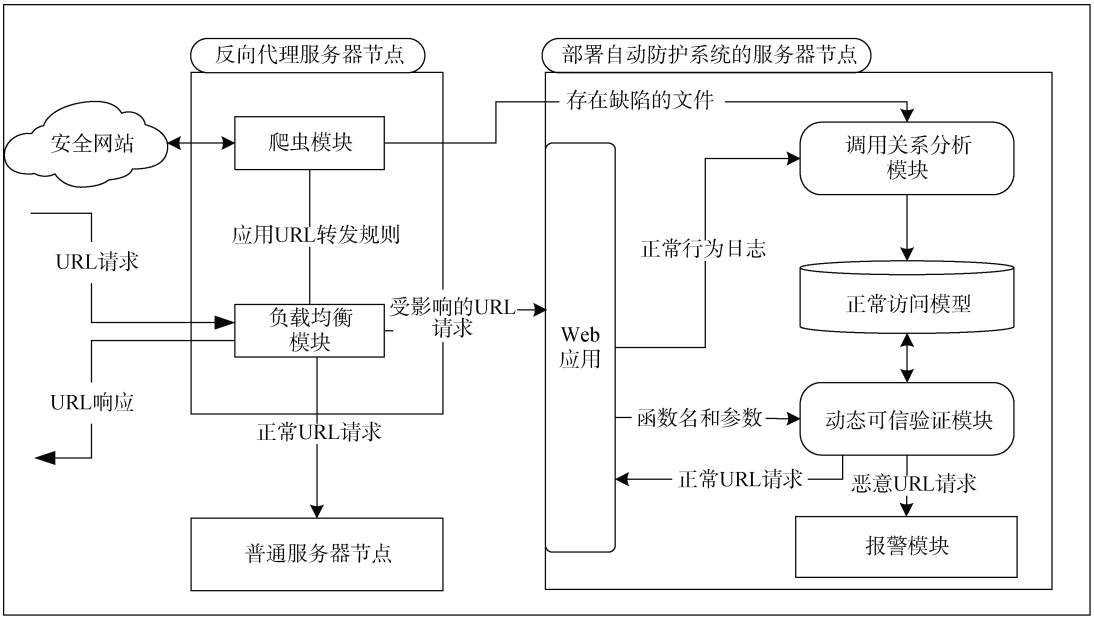

图 2　系统整体架构

Figure 2　System architecture

防护系统需要从如下这三个方面来展开工作:

(1) 分析 Web 应用行为日志。这部分工作在 Web 应用上线前进行。由管理员在可信环境下模拟正常用户对 Web 应用发起访问请求, 确保覆盖 Web 应用的所有页面以及各个功能模块, 搭载自动防护系统的服务器节点在响应这些可信请求后, 生成详细的调用行为日志; “调用关系分析模块”对行为日志进行分析和预处理, 提取关键信息并理清各个文件、类和函数之间的调用关系, 经过预处理后会生成两个文件。一个是函数调用的模版文件, 用于关系调用分析, 另一个则是敏感函数记录文件, 保存所有敏感函数的函数名、参数内容和执行上下文环境, 用于正常访问模型的构建和训练。

以 PHP Web 应用为例, 需要提取的关键信息包括如下三类[14]:

1) 第一类是“include”文件引用, 主要记录发起引用的 PHP 脚本文件名和被引用的 PHP 脚本文件名;

2) 第二类的类函数的调用, 需要记录类名和类函数名以及执行时所在的 PHP 脚本文件名;

3) 第三类是类函数对于 PHP 内建敏感函数的调用, 需要记录被调用的敏感函数名称, 敏感函数的参数内容、类名、类函数名以及执行时所在的 PHP 脚本文件名。

(2) 搜集分析漏洞预警信息。系统爬虫模块需要定期爬取各大安全网站上的漏洞公告信息, 并从中提取存在缺陷的文件名和函数名, 将信息传递给“调用关系分析模块”, 自动判断是否需要进行漏洞响应。

(3) 拦截敏感函数。“调用关系分析模块”根据存在缺陷的文件名找出所有受影响的 URL 请求, 返回给“爬虫模块”, 同时将存在缺陷的文件名发给“动态可信验证模块”, 通知对敏感函数进行拦截。“动态可信验证模块”收到存在缺陷的文件名后, 根据管理员所配置的策略决定直接对文件调用进行拦截还是对参数进行检查。

3.2 模块实现

3.2.1 动态可信验证模块

“动态可信验证模块”主要负责在发生漏洞预警时, 对存在缺陷的文件调用进行拦截, 并对敏感函数的调用进行参数检查, 识别并阻止攻击请求。

以 PHP Web 应用为例, 可以通过编写 Hook 函数的方式来对 PHP 的调用进行拦截。PHP 的体系结构如图 3 所示:

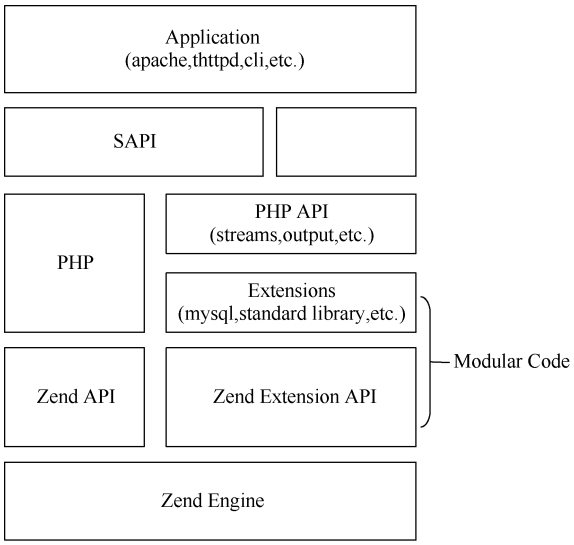

图 3 PHP 体系结构

Figure 3 PHP architecture

从上到下主要分为四个层次: 应用层(Application)、SAPI 层、扩展层(Extensions)和 Zend 引擎。应用层就是由 Web 应用开发人员编写的被运行加载的 PHP 脚本。SAPI(Server Application Programming Interface, 服务端应用编程接口)的作用在于抽象接口, 把 PHP 和实际的服务器应用分开, 让 PHP 专注于解释执行的工作而不需要适配不同的服务器。扩展层允许用户通过安装 PHP 扩展程序的方式为 PHP 添加功能, 扩展层的功能和 PHP 核心的 API 功能属于同一层。最底层 Zend 引擎是 PHP 的内核, 负责实现 PHP 核心的所有功能, 如 PHP 的语法实现、内存管理、垃圾回收、对 PHP 的解释执行以及实现扩展程序的运行。在编译阶段, PHP 解释器先对 PHP 脚本文件的代码进行词法分析, 将源代码按照词法规则切成一个个标记(token), 再使用语法分析器对标记进行规则匹配, 最后将代码编译成 opcode。opcode 是 Zend 引擎的中间语言, 类似于汇编代码, 在执行阶段由 Zend 引擎来执行。Zend 引擎执行 opcode 时如果遇到像“eval()”这种动态执行的函数, 则还需要继续进行编译操作。

Zend 引擎向上提供了一套 Zend API, 供 PHP 扩展程序调用来实现 PHP 的调用拦截。Zend API 提供了一个函数“zend_set_user_opcode_handler()”来实现对某个 opcode 进行拦截。通过“zend_set_user_opcode_handler()” 对某个指定的 opcode 的设置 Hook 函数, 当 Zend 引擎执行到指定的 opcode 时, 处理流程会跳转到 Hook 函数, 由 Hook 函数对操作数进行操作, 操

作完毕后 Zend 引擎可以根据 Hook 函数所返回的宏决定是继续由原有的处理函数对操作数进行处理, 还是跳过原有的处理函数, 甚至是直接中断该 PHP 脚本代码的执行。

基于以上背景, 以 PHP Web 应用作为防护对象为例, 当发现漏洞预警信息后, 系统首先确认存在缺陷的 PHP 脚本文件名或类函数名。动态可信验证模块直接对存在缺陷的 PHP 脚本文件进行修改, 在特定的位置插入 PHP 全局函数。不同的全局函数分别负责拦截文件调用、类调用和函数调用。拦截文件调用即拦截对缺陷 PHP 脚本文件的引入和调用, 这在 PHP Web 应用中表现为某个功能被临时禁用; 为了维持 Web 应用功能和业务的完整性, 对于有缺陷的类和函数, 则需要针对性地拦截类调用和函数调用。

对于存在缺陷的 PHP 脚本文件, 如果要对整个文件的调用进行拦截, 则在 PHP 脚本文件中的“<?php”的下一行插入动态可信验证提供的文件调用拦截函数。如果是 PHP 脚本文件中的类存在缺陷, 则需要在 PHP 脚本文件中查找类的定义范围, 再查找定义范围内的“__construct()”函数。如果存在缺陷的类没有“__construct()”函数, 则动态可信验证可以主动新增该函数, 并在函数内调用动态可信验证的类拦截函数。当其他 PHP 脚本文件引入这个缺陷文件后, 在开始实例化存在缺陷的类的对象时, 动态可信验证就能先收到类拦截函数的调用, 从而进行拦截。如果是函数出现缺陷, 则直接在函数声明的代码块的第一行插入动态可信验证的拦截函数。在程序开始执行缺陷函数时, 就会先调用动态可信验证的拦截函数。

对于所有受影响的 URL 请求, 动态可信验证模块都需要对即将执行的函数进行检查, 在收到函数名和具体参数后, 可以根据“调用关系分析模块”构建的正常访问模型, 检查函数的执行和参数活动范围是否处于正常水平, 也可以根据当前预警的漏洞信息和函数的位置决定不同的判定策略, 得到判定结果。

如果动态可信验证模块将当前即将执行的函数判定为正常行为, 则直接放行, 对于异常行为则需要阻止该函数的执行, 对缺陷 PHP 脚本文件的引用和对缺陷类的实例化也需要进行阻止调用。具体工作分为两步, 第一步是中断当前 PHP 程序的执行, 让 Zend 引擎不再执行下一个 opcode, 第二步是对页面进行跳转, 跳转到某个情况说明页面, 或者是 PHP Web 应用的首页。在动态可信验证模块中, 先通过 Zend API 调用“zend_call_function()”函数, 通过这个函数调用 PHP 全局函数“header()”, 在对 URL 请求的响应中的 Header 设置“302”跳转地址。设置完毕后, 再通过 Zend API 调用“zend_error()”函数, 中断 Zend 引擎的执行, PHP 程序会立刻将响应返回给客户端。

3.2.2 其他模块

“调用关系分析模块”主要负责在响应管理员发起的可信请求后, 从正常行为日志中对各个文件的调用关系以及类函数的调用关系进行分析, 一方面是当发生漏洞预警时, 能够根据调用关系分析找到所有受影响的 URL 请求, 以供负载均衡模块实施 URL 规则转发; 另一方面是构建和训练一个 Web 应用的正常访问模型。

“爬虫模块”负责定时抓取各大安全相关网站的所发布漏洞预警公告, 并从公告中提取相关信息, 如受影响的 Web 应用名称、受影响的版本以及存在缺陷的文件名等, 并根据提取的信息判断当前 Web 应用是否受到漏洞影响。运维人员也可以通过隐藏端口, 主动指明存在缺陷的文件名和漏洞类型。

“负载均衡模块”作为服务器集群的反向代理服务器, 主要负责在 Web 应用正常运行时动态将 URL 请求交给不同的服务器节点处理, 合理分配计算资源, 从而提高系统的响应速度与可靠性。当发生漏洞预警时, 根据调用关系分析得到的 URL 列表, 应用 URL 规则转发, 将这些受影响的 URL 请求转发到部署针对预警信息的漏洞自动化防护系统的服务器节点上。

“报警模块”接收从动态可信验证模块传递过来的信息, 包括判定为恶意的 URL 请求、客户端 IP 地址、即将执行的函数名和函数参数等, 通过发送邮件和短信的方式通知运维人员。

4 实验及分析

4.1 测试环境及系统分析

以 PHP Web 应用作为防护对象为例, 本文使用 C 语言和 Python 语言来编写基于预警信息的漏洞自动化防护系统的原型, 使用 VMware 虚拟机来模拟服务器集群环境和用户访问。本文创建了三个 VMware 虚拟机, 其中一个节点作为反向代理服务器, 一个是正常的服务器节点, 一个是部署了基于预警信息的漏洞自动化防护系统的服务器节点。宿主机作为普通用户和攻击者对服务器集群进行访问。相关环境配置如表 1 所示。

表 1 测试环境说明

Table 1 Test environment specification

配置项	内容
宿主机系统	Windows 8
宿主机 CPU	i3-3220
虚拟机系统	Ubuntu 14
虚拟机分配处理器核心数	1
虚拟机分配内存	1GB
PHP 版本	5.6.12
Nginx 版本	2.4.6
Apache 版本	2.4
Mysql 版本	5.6
Neo4j 版本	3.2
Python 版本	2.7

4.2 功能测试

本文选择了 3 个流行的 PHP Web 应用的 7 个实际漏洞来对自动防护系统进行功能测试。3 个 PHP Web 应用分别是 WordPress、Joomla!和 PHPCMS, 这些应用市场占比较高, 漏洞都具有比较高的危害性, 类型也不尽相同, 具有良好的代表性和研究价值, 能够验证本文方法的防护效果和适用范围。

在功能测试之前, 先关闭爬虫模块, 测试人员先模拟正常的用户对 PHP Web 应用所提供的所有功能运行多次, 保证自动防护系统的调用关系分析模块能够记录到所有正常行为的调用。当打开爬虫模块收到漏洞预警通知后, 测试人员先对 PHP Web 应用发起漏洞利用攻击, 测试拦截结果; 之后测试人员再对 PHP Web 应用提供的不受影响的功能重新运行一遍, 测试误报率。应急响应的方式取决于预警通告所提供的信息, 如果涉及范围过广或者没有指明出现缺陷的类函数, 则只能采用临时拦截的方式, 将可能存在攻击的功能禁用; 部分漏洞预警的信息比较详细, 则可以使用参数检查的应急响应方式, 影响的范围更小。

测试结果如表 2 所示:

表 2 功能测试结果

Table 2 Functional test results

漏洞编号	应用名称	受影响版本	漏洞危害	误报率/%	拦截结果
CVE-2017-1001000	WordPress	4.7.0<=4.7.2	内容注入	3.7	成功
CVE-2017-5487	WordPress	4.7.0<=4.7.1	信息泄露	1.8	成功
CVE-2016-8869	Joomla!	3.4.4-3.6.3	注册提权	0	成功
CVE-2016-8870	Joomla!	3.4.4-3.6.3	绕过注册限制	0	成功
CVE-2017-8917	Joomla!	3.7.0	SQL 注入	0	成功
CVE-2015-7297	Joomla!	3.2-3.4.4	SQL 注入	0	成功
CNVD-2017-04180	PHPCMS	9.6	任意文件上传	0	成功

测试结果中的误报率同样依赖于预警通告所提供的信息, Joomla!的 CVE-2016-8869 和 CVE-2016-8870 漏洞的预警通告指明了具体的缺陷类函数, 因此可以实现精准的拦截。对于在预警信息中缺乏细节信息的漏洞, 可以采用 2.2 节中提到的方法, 即利用正常访问模型判断请求合法性、或者根据不同的漏洞类型采用不同的检查策略来识别攻击请求, 但是这样判断存在一定的误报率, 例如 WordPress 的 CVE-2017-1001000 和 CVE-2017-5487 的预警通告只指明了出现问题的文件名, 临时的拦截会影响部分正常功能, 故存在误报率。总体而言, 本系统具有较高的拦截成功率, Stefan Prandl 对三个开源 Web 应用防火墙(WAF)ModSecurity、WebKnight 以及 Guardian 的测试研究表明, 漏洞攻击拦截率最高为 85.6%[15], 说明了本系统相对于 WAF 具有更好的拦截效果。

4.3 性能测试

本文选择服务器节点内 Apache 对应进程 httpd 和 Python 脚本的 CPU 占比和内存占比作为自动防护系统的性能指标。CPU 总占比指的是正常服务器节点和部署自动防护系统的 CPU 占比相加。内存总占比同理。测试情景分为不启用自动防护系统运行、启用系统进行行为分析和发生预警并且存在黑客进行攻击三种情况。不同情况下, 本文使用压力测试工具 JMeter[16]模拟 10 个用户对 PHP Web 应用不断发起随机 URL 请求, 同时在服务器节点内记录 Apache 对应进程 httpd 和 Python 脚本的 CPU 占比和内存占比, 取平均值。测试结果如表 3 所示:

表 3 性能测试结果

Table 3 performance test results

漏洞编号	不启用系统		启用系统		遭受攻击		性能消耗增长
	CPU 总占比/%	内存总占比/%	CPU 总占比/%	内存总占比/%	CPU 总占比/%	内存总占比/%	
CVE-2017-1001000	21.4	59.1	27.0	43.9	9.5	53.73	5.96
CVE-2017-5487	22.14	58	34.8	55.19	13.43	55	12.66
CVE-2016-8869	2.87	50.16	9.39	46.33	4.65	56.3	6.52
CVE-2016-8870	2.88	50.23	7.9	44	4.71	55.79	5.02
CVE-2017-8917	6.22	54.23	7.16	45.14	6.70	55.58	0.94
CVE-2015-7297	6.8	54.06	8.63	47.38	7.38	56.62	1.83
CNVD-2017-04180	8.22	57.6	12.49	54.5	11.3	57.44	4.27

在测试结果中可以发现, 启用了系统后内存的占用反而降低, CPU 的损耗则是上升。发生该现象的关键在于 URL 请求的吞吐量, Apache 处理的请求越多, 内存占用越高。部署自动防护系统的服务器节点由于需要记录行为调用还有其他计算任务, 单位时间内能够处理的 URL 数量少, 大部分 URL 请求由正常服务器节点处理。正常服务器节点遇到大量 URL 请求时, 吞吐率也下降, 所以内存占比反而降低。当发生漏洞预警后, 部署防护系统的节点处理的请求就更少了, 再加上采用的应急响应方式是临时拦截的方式, 直接将页面跳转, 所需性能更少。最后取所有漏洞测试的性能消耗增长的平均值作为自动防护系统的性能消耗增长, 为 5.31%。相比之下, Zhongxu Yin 等提出的用于防范脚本注入攻击的方法, 防护漏洞单一, 且平均性能损耗为 20%[17]。

4.4 响应时间测试

响应时间方面, 当发生漏洞预警后, 防护系统可以迅速做出响应。以 PHPCMS 的 CNVD-2017-04180 任意文件上传漏洞为例, 测试中爬虫模块发现预警并获取缺陷文件名需要 31.8 秒, 找到受影响的 URL 并对缺陷文件插入全局拦截函数耗时 4.2 秒, 再对 Nginx 的配置文件进行修改以拦截受影响的请求耗时 0.05 秒, 整个过程共计耗时 36.05 秒, 其余漏洞响应耗时也在 60 秒内, 而人工响应漏洞预警耗时一般从数天到数十天不等, 部分网站甚至会忽略漏洞, 表明防护系统能够显著提高漏洞响应的速度与效率。

4.5 结果分析

功能测试的结果说明该防护系统的有效性, 能够对不同类型的漏洞攻击进行防护, 拦截成功率高。误报率则和预警信息有关: 当预警信息中没有明确指出存在缺陷的文件或函数时, 存在一定的误报率。防护系统在运行中保证了其他功能的正常运转, 并能够成功把攻击行为与其他正常调用区分出来。对于其他类型的 Web 应用, 同样可以基于本文提出的防护方法设计对应的自动防护系统, 说明了该防护方法的兼容适配性。

性能测试的结果说明了基于预警信息的漏洞自动化防护系统具有一定的可用性。平均性能为 5.31%, 最高不超过 13%。在遭受黑客攻击的极端情况下, 该系统能够保证其它业务正常运行, 所增加的性能消耗在 6%, 处于能够接受的范围内。

5 结束语

本文针对现有 Web 防护系统难以有效检测未知漏洞攻击、响应速度慢, 以及性能损耗大等问题, 提出了一种基于预警信息的漏洞自动化防护方法, 并在此基础上推出一个能够根据预警信息来对 Web 应用进行自动防护的系统原型,能够以较低的性能损耗拦截漏洞攻击, 不影响系统其他业务的正常运行。但目前的工作中仍存在一些不足之处, 目前所构建的正常访问模型只记录了敏感函数的参数和执行上下文环境, 如果能在调用关系中找到执行业务的关键调用路径, 并对这些路径上的函数调用参数进行记录, 则能增大正常访问模型的覆盖范围, 也能避免记录所有函数参数所需空间过大的问题。

参考文献

[1] The National Computer Network Emergency Response Technical Team. A review of China's Internet security situation in 2018[EB/OL]. https://www.cert.org.cn/publish/main/upload/File/2018situation.pdf.(in Chinese)
(国家互联网应急中心. 2018 年我国互联网网络安全态势综述[EB/OL]. https://www.cert.org.cn/publish/main/upload/File/2018situation.pdf.)

[2] B. Stock, G. Pellegrino, C. Rossow, M. Johns, et al. Hey, you have a problem: On the feasibility of large-scale web vulnerability notification[C]. *USENIX Security Symposium*, 2016:234-245.

[3] Stock B, Pellegrino G, Li F, et al. Didn't You Hear Me? - Towards more Successful Web Vulnerability Notifications[C]. *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA. Reston, VA: Internet Society, 2018: 345-352.

[4] Lian yifeng, Dai yingxia. System research of computer emergency response system[J]. *Journal of graduate school of Chinese academy of sciences*, 2004, 20(2):202-209. DOI:10.3969/j.issn.1002-1175.2004.02.009.(in Chinese)
(连一峰, 戴英侠. 计算机应急响应系统体系研究[J]. *中国科学院研究生院学报*, 2004, 20(2): 202-209. DOI: 10.3969/j.issn.1002-1175.2004.02.009.)

[5] Fang Y X, Chen L, Wang J S, et al. Research on smart grid security vulnerability report and early warning platform[J]. *Netinfo Security*, 2016,15(1): 81-84.
(方玉昕, 陈亮, 王劲松, 等. 智能电网安全漏洞报告与预警平台研究[J]. 信息网络安全, 2016,15(1): 81-84.)

[6] Longhai. Operation and maintenance method of ERP system in collectivized enterprises[J]. *Journal of north China electric power university (social science edition)*,2015,16(6):92-94.DOI:10.3969/j.issn.1008-2603.2015.06.016. (in Chinese)
(龙海. 集团化企业 ERP 系统运维方法浅谈[J].*华北电力大学学报(社会科学版)*, 2015, 16(6):92-94. DOI: 10.3969/j.issn.1008-2603.2015.06.016.)

[7] Newdefend. Newdefend takes the lead in defense to intercept Joomla's high-risk 0Day vulnerability[EB/OL]. https://www.newdefend.com/news/detail/51.(in Chinese)
(牛盾云. 牛盾云安全率先防御拦截 Joomla 的高危 0Day 漏洞[EB/OL]. https://www.newdefend.com/news/detail/51.)

[8] Prokhorenko V, Choo K K Raymond, Ashman H. Intent-Based Extensible Real-Time PHP Supervision Framework[J]. *IEEE Transactions on Information Forensics and Security*, 2016, 11(10): 2215-2226.[LinkOut]

[9] Lu peng.. *The Research and Application of Big Data Retrieval Organization Based on Neo4j*[D]. Nanjing: Southeast University, 2015.
(陆鹏. 基于 Neo4j 的大数据组织检索研究与应用[D]. 南京: 东南大学, 2015.)

[10] Goldsmith S, O'Callahan R, Aiken A. Relational Queries over Program Traces[J]. *ACM SIGPLAN Notices*, 2005, 40(10): 385.

[11] Das D, Sharma U, Bhattacharyya D K. Defeating SQL Injection Attack in Authentication Security: An Experimental Study[J]. *International Journal of Information Security*, 2019, 18(1): 1-22.

[12] Kao D Y, Lai C J, Su C W. A Framework for SQL Injection Investigations: Detection, Investigation, and Forensics[C]. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, October 7-10, 2018. Miyazaki, Japan. Piscataway, NJ: IEEE, 2018.

[13] Marius Steffens, Christian Rossow, Martin Johns, et al. Don't Trust The Locals:Investigating the Prevalence of PersistentClient-Side Cross-Site Scripting in the Wild[J]. *NDSS* 2019

[14] Nunes P J C, Fonseca J, Vieira M. PhpSAFE: A Security Analysis Tool for OOP Web Application Plugins[C]. *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, June 22-25, 2015. Rio de Janeiro, Brazil. Piscataway, NJ: IEEE, 2015: 299-306.

[15] Prandl S, Lazarescu M, Pham D S. A Study of Web Application Firewall Solutions[M]. Information Systems Security. Cham: Springer International Publishing, 2015: 501-510.

[16] Srinivasa Shenoy, Nur Asyikin Abu Bakar, Rajashekara Swamy, et al. Denial of Service Attack Generator in Apache JMeter[C]. *ICUMT* 2018: 1-4.

[17] Yin Z X, Li Z F, Cao Y. A Web Application Runtime Application Self-protection Scheme Against Script Injection Attacks[M]. Cloud Computing and Security. Cham: Springer International Publishing, 2018: 566-577.

徐其望 于 2017 年在湖北大学信息安全专业获得学士学位。现在武汉大学网络空间安全专业攻读硕士学位。研究领域为网络安全。研究兴趣包括: Web 安全、智能硬件安全。Email: yangxiaozhi@whu.edu.cn

陈震杭 于 2018 年在武汉大学信息安全专业获得硕士学位。现任绿盟科技有限公司研究员。研究领域为网络安全。研究兴趣包括: Web 安全、入侵检测。Email: 5284542@qq.com

彭国军 于 2008 年在武汉大学信息安全专业获得博士学位。现任武汉大学国家网络安全学院教授。研究领域为网络与信息系统安全。Email: guojpeng@whu.edu.cn

张焕国 教授, 博士生导师, CCF 高级会员, 主要研究领域为信息安全, 可信计算, 密码学。Email: liss@whu.edu.cn