信息安全技术丛书

代码审计：企业级 Web 代码安全架构

尹　毅　编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

代码审计：企业级 Web 代码安全架构 / 尹毅编著 . —北京：机械工业出版社，2015.12
（信息安全技术丛书）

ISBN 978-7-111-52006-1

I. 代… II. 尹… III. 计算机网络－安全技术 IV. TP393.08

中国版本图书馆 CIP 数据核字（2015）第 260798 号

代码审计是企业安全运营的重要步骤，也是安全从业者必备的基础技能。本书详细介绍代码审计的设计思路以及所需要的工具和方法，不仅用大量案例介绍了实用方法，而且剖析了各种代码安全问题的成因与预防方案。无论是应用开发人员还是安全技术人员都能从本书获益。

本书共分为三个部分。第一部分为代码审计前的准备，包括第 1 ～ 2 章，第 1 章详细介绍代码审计前需要了解的 PHP 核心配置文件以及 PHP 环境搭建的方法；第 2 章介绍学习 PHP 代码审计需要准备的工具，以及这些工具的详细使用方法。第二部分着重介绍 PHP 代码审计中的漏洞挖掘思路与防范方法，包括第 3 ～ 8 章，第 3 章详细介绍 PHP 代码审计的思路，包括根据关键字回溯参数、通读全文代码以及根据功能点定向挖掘漏洞的三个思路；第 4 ～ 6 章则介绍常见漏洞的审计方法，分别对应基础篇、进阶篇以及深入篇，涵盖 SQL 注入漏洞、XSS 漏洞、文件操作漏洞、代码 / 命令执行漏洞、变量覆盖漏洞以及逻辑处理等漏洞；第 7 章介绍二次漏洞的挖掘方法；第 8 章介绍代码审计过程中的一些重要技巧。第三部分主要介绍 PHP 安全编程规范，从攻击者的角度来告诉你应该怎么写出更安全的代码，包括第 9 ~ 12 章，第 9 章介绍参数的安全过滤；第 10 章介绍 PHP 中常用的加密算法；第 11 章从设计安全功能的角度出发，从攻击者的角度详细分析常见功能通常会出现的安全问题以及解决方案；第 12 章介绍企业的应用安全体系建设，介绍横向细化策略和纵深防御策略的具体实施方法与典型案例。

代码审计：企业级 Web 代码安全架构

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）

责任编辑：吴　怡	责任校对：董纪丽
印　　刷：	版　　次：2016 年 1 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16	印　　张：15.25
书　　号：ISBN 978-7-111-52006-1	定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88379426　88361066	投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259	读者信箱：hzit@hzbook.com

Preface 序　言

我第一次见到尹毅是 2013 年在北京中关村。那时候我正在安全宝创业，我们需要招募到最好的人才。这时候尹毅的博客吸引了我，在一个技术分享逐渐枯竭的时代，他的博客令人眼前一亮。然后我试图联系到了他，并邀请到北京来聊一聊。

让我大吃一惊的是，尹毅当时还是一个孩子模样，但是时不时能从生涩的脸庞里看到成熟。在这个年纪就出来工作，我想他一定吃过很多苦。在之后的工作中，尹毅展现出了惊人的天赋。交给他的工作总是能迅速并出色地完成，并时不时会在工作中有一些创新性成果令人惊喜。他的自驱力极强，总是不满足于简单的工作，于是我不得不想出一些更复杂和艰难的挑战交给他。

2014 年 9 月，安全宝分拆了部分业务被阿里收购，我带着尹毅一起到了阿里。此时他已经成为一个安全团队的 Leader，在中国最大的互联网公司里贡献着力量。

尹毅学东西很勤奋，他平时的业余时间就是写代码，或者看技术文章，因此进步迅速。他很快就在 Web 漏洞挖掘能力方面有了长足的进步，并取得了不错的成绩，他陆续发现了好些开源软件的高危漏洞。最难能可贵的是，他开始逐步总结这些经验，并且沉淀在自己开发的一个漏洞挖掘工具里。这让他学会了如何从重复的体力劳动中解放出来，把精力用在更有价值的地方。这是一个优秀黑客应具有的特质：厌倦重复性的体力劳动，而对创新充满着无限的热情和旺盛的精力。

尹毅认为，一个好的黑客，必须要懂编程。这也是他在这本书里所倡导的理念。在他看来，不懂编程、没挖过漏洞的黑客，充其量只能算“脚本小子”。所以，尹毅在本书的出发点是从代码审计开始，通过代码审计，去发现和挖掘漏洞。

漏洞挖掘是一门艺术，同时也是信息安全的核心领域。安全技术发展到今天，常见的漏洞挖掘技术有代码审计、黑盒测试、Fuzzing、逆向分析等。每一种技术都有独到之处，而其中，代码审计又是最基本、最直接的一种方式，是每一个安全专家都应该

掌握的技能。

但时至今日，全自动化的代码审计仍然存在很多困难，主要难点在于理解编程语言的语法、跨文件之间的关联调用、理解开发框架、业务逻辑等地方。因为这些困难在短期内难以克服，所以通过代码审计来挖掘漏洞，仍然是一种极具技巧性和需要丰富经验的工作。在本书中，尹毅根据他自身的经验和学习成果，对这些知识技巧做了一个很好的总结。

本书虽然主要讲述的是 PHP 代码安全问题，但其中的很多思想和案例都非常具有代表性。同时，因为互联网上大量的 Web 应用都是由 PHP 写成的，因此研究 PHP 代码安全对于整个互联网 Web 安全的研究具有至关重要的作用。对于新人来说，非常建议从本书中讲述的内容开始学习。

吴翰清，阿里云云盾负责人，《白帽子讲 Web 安全》作者

2015.9.20

Preface 前　言

代码审计是指对源代码进行检查，寻找代码中的 bug，这是一项需要多方面技能的技术，包括对编程的掌握、漏洞形成原理的理解，系统和中间件等的熟悉。

为什么需要代码审计

代码审计是企业安全运营以及安全从业者必备的基础能力。代码审计在很多场景中都需要用到，比如企业安全运营、渗透测试、漏洞研究等。目前已经有不少公司在推广微软的软件 SDL（Security Development Lifecycle，安全开发周期），它涵盖需求分析→设计→编码→测试→发布→维护，安全贯穿整个软件开发周期，其中设计、编码和测试是整个 SDL 的核心，安全问题大多在这里被解决掉。其中在安全设计这块，必须要非常了解漏洞形成原理，纵观全局。而在代码实现也就是编码阶段，安全依靠于编程人员的技术基础以及前期安全设计的完善性。然后是测试，测试包括白盒测试。黑盒测试以及灰盒测试。黑盒测试也叫功能测试，是指在不接触代码的情况下，测试系统的功能是否有 bug，是否满足设计需求。而白盒测试就是我们说的代码审计，以开放的形式从代码层面寻找 bug，如果发现有 bug 则返回修复，直到没有 bug 才允许软件发布上线。

渗透测试人员掌握代码审计是非常重要的，因为我们在渗透过程中经常需要针对目标环境对 payload 进行调试。另外，如果通过扫描器扫描到 Web 目录下的一个源码备份包，通常攻击者都会利用源码包找一些配置文件，因为里面有数据库、API 等一类配置。如果环境有限制，比如目标站数据库限制连接 IP 等，那么工具小子可能在源码包进行的漏洞利用也就到此为止。对于懂代码审计的人，结果完全不一样，他可以对源码包进行安全审计，发现网站代码里存在的漏洞，然后利用挖掘到的漏

洞进行渗透。

编程能力要求

代码审计对编程语言的基础有一定要求，至少要能看得懂代码，这里说的看懂代码不是简单地理解几个 if...else 语句和 for 循环，而是能看懂代码的逻辑，即使有很多函数没见过，也是可以到 Google 去查的。都说编程在语言这块是一通百通，只要我们对编程思想理解得非常透彻，重新接触一种编程语言也是非常快就能上手的，所以不管你之前写过 Java 还是 C# 程序，想玩一玩 PHP 的代码审计都应该不是什么大问题。

代码审计环境准备

代码审计首先要准备的是审计环境工具，不同的环境会影响漏洞的利用，所以建议 Linux 和 Windows 系统下的 PHP 环境都搭建一套，并且需要多个 PHP 版本。关于版本切换这块，建议安装 phpStudy，phpStudy 是一套 Apache+Nginx+LightTPD+PHP+MySQL+phpMyAdmin+Zend Optimizer+Zend Loader 的集成环境，可以很方便地安装和切换环境。代码审计的工具有很多个，这里推荐使用笔者开发的 Seay 源代码审计系统以及 RIPS，二者都是免费开源工具。

除了自动化审计工具外，还有一些像 Burp Suite、浏览器扩展以及编码工具等审计辅助工具也都是必备的。

代码审计思路

通常做代码审计都是检查敏感函数的参数，然后回溯变量，判断变量是否可控并且没有经过严格的过滤，这是一个逆向追踪的过程。而代码审计并非这一种手段，还可以先找出哪些文件在接收外部传入的参数，然后跟踪变量的传递过程，观察是否有变量传入到高危函数里面，或者传递的过程中是否有代码逻辑漏洞，这是一种正向追踪的方式，这样的挖掘方式比逆向追踪挖掘得更全。还有一种方式是直接挖掘功能点漏洞，根据自身的经验判断该类应用通常在哪些功能中会出现漏洞，直接全篇阅读该功能代码。

可能不少新手对于学习 PHP 代码审计还有一些迷茫，或许之前尝试过学习，但一

直没有很好的进展，因为代码审计是一门很专的技术活，要学好 PHP 代码审计，需要掌握以下几点：

- PHP 编程语言的特性和基础。
- Web 前端编程基础。
- 漏洞形成原理。
- 代码审计思路。
- 不同系统、中间件之间的特性差异。

导　读 *Introduction*

本书总共分为三个部分。第一部分为代码审计前的准备，包括第 1 章以及第 2 章，第 1 章详细介绍我们在学习代码审计前需要了解的 PHP 核心配置文件以及 PHP 环境搭建的方法，第 2 章介绍学习 PHP 代码审计需要准备的工具，以及这些工具的详细使用方法。

第二部分包括第 3 ～ 8 章，着重介绍 PHP 代码审计中的漏洞挖掘思路与防范方法。

第 3 章详细介绍 PHP 代码审计的思路，包括根据关键字回溯参数、通读全文代码以及根据功能点定向挖掘漏洞的三个思路。

第 4 ～ 6 章讲述常见漏洞的审计方法，分别对应基础篇、进阶篇以及深入篇，涵盖 SQL 注入漏洞、XSS 漏洞、文件操作漏洞、代码 / 命令执行漏洞、变量覆盖漏洞以及逻辑处理等漏洞。

第 7 章介绍二次漏洞的挖掘方法，二次漏洞在逻辑上比常规漏洞要复杂，所以我们需要单独拿出来，以实例来进行介绍。

在经过前面几章的代码审计方法学习之后，相信大家已经能够挖掘不少有意思的漏洞。第 8 章将会介绍代码审计中的更多小技巧，利用这些小技巧可以挖掘到更多有意思的漏洞。每类漏洞都有多个配套的真实漏洞案例分析过程，有助于读者学习代码审计的经验。不过，该章不仅介绍漏洞的挖掘方法，还详细介绍这些漏洞的修复方法，对开发者来说，这是非常有用的一部分内容。

第三部分包括第 9 ～ 12 章，主要介绍 PHP 安全编程的规范，从攻击者的角度来告诉你应该怎么写出更安全的代码，这也是本书的核心内容：让代码没有漏洞。第 9 章主要介绍参数的安全过滤，所有的攻击都需要有输入，所以我们要阻止攻击，第一件要做的事情就是对输入的参数进行过滤，该章详细分析 discuz 的过滤类，用实例说明什么样的过滤更有效果。

第 10 章主要介绍 PHP 中常用的加密算法。目前 99% 以上的知名网站都被拖过库，泄露了大量的用户数据，而这一章将详细说明使用什么样的加密算法能够帮助你增强数据的安全性。

第 11 章涉及安全编程的核心内容。所有的应用都是一个个功能堆砌起来的，该章从设计安全功能的角度出发，从攻击者的角度详细分析常见功能通常会出现的安全问题，在分析出这些安全问题的利用方式后，再给出问题的解决方案。如果你是应用架构师，这些内容能够帮助你在设计程序功能的时候避免这些安全问题。

第 12 章介绍应用安全体系建设的两种策略以及实现案例：横向细化和纵深策略，企业的应用安全应把这两种策略深入到体系建设中去。

以上就是本书的全部内容，看到介绍之后你是不是有点儿兴奋呢？赶紧来边读边试吧。

感言和致谢

这本书断断续续写了一年多，期间也发生了很多事情。在 2014 年 9 月的时候从创新工场旗下项目安全宝离职，加入到阿里巴巴安全部。到了一个新的环境，工作上面倒是很快就融入了进去，只是从北京到杭州，是从一个快节奏的城市转到一个慢节奏的城市，感觉整个人变懒了，没有以前在北京那样每天激情澎湃。曾经一度想过放弃，因为心里总感觉像是被捆住了一样，想去做一些事情却因为还有这本书没写完而不能去做。因为写这本书是我必须要做的事情，一是算是给我自己在安全领域的一个交代，二是我承诺过吴怡编辑，一定会努力写好这本书，在这个事情上我看得很严肃，承诺了就一定要做到。

为什么说这算是给自己在安全领域的一个交代呢？记得跟不少朋友说过，创业是我必定要做的一个事情，或许哪天转行创业了，在这个行业里留下了点东西也心安了。回想自己从最初迷恋上网络安全到现在，中间的一些转折点和小插曲还挺有意思，比如以全校第一的成绩考上重点高中之后，读了一年就退学去离家很远的软件开发培训学校，花 500 块钱在网吧淘了一台放酷狗都卡得不行的台式机，在重庆连续通宵读书快一年，等等，这些都已经是美好的回忆。在这些美好回忆中遇到很多美好的人，想对他们说声谢谢。

感谢父母和姐姐、姐夫，最早去重庆的时候，姐姐还怀着马上要出生的外甥女，跟姐夫开车送我去重庆。学校一个学期一两万的学费，父母预支薪水供我读书，感谢他

们的付出。

感谢机械工业出版社的吴怡编辑，如果没有她的鼓励和指导，也不会有这本书的面世，真心感谢她。

感谢吴瀚清（网名：刺、道哥、大风），在安全宝的时间里，刺总给了我很多帮助，不管是工作上还是个人成长上都给予引导和包容，他是一位真正的好老板。

感谢 safekey team 的兄弟们，他们是晴天小铸、tenzy、x0h4ck3r、zvall、yy520 以及 cond0r，本书里面有多个影响非常大的 0day 出自他们之手，我们因为喜欢代码审计而聚集在一起。

感谢曾经陪我熬了无数个通宵的好哥们 Snow、小软，我们曾经一起渗透，一起研究，一起写代码，无不分享。

感谢工作中同我一起奋斗的同事们，没有他们的辛苦战斗就没有今天我们攻城拔寨的辉煌战绩，他们包括但不限于：翁国军、李翼、全龙飞、曾欢。

感谢喜付宝的林能（ID：矢志成谜）对本书提出的建设性建议。

尹毅

微信：seayace

邮箱：root@cnseay.com

博客：www.cnseay.com

微博：http://weibo.com/seayace

Contents 目　录

第一部分 Part 1

代码审计前的准备

漏洞的利用依赖 PHP 版本、Web 中间件版本与类型、操作系统类型和版本以及这些软件的配置等多因素，所以我们在代码审计前需要做不少的准备工作，最重要的是环境搭建和代码审计辅助工具的使用，这一部分将从代码审计环境的搭建和这些工具的使用来展开介绍。

第 1 章主要介绍环境的搭建，包括 wamp/wnmp 环境以及 lamp/lnmp 环境。这些环境搭建是简单的。这里要重点理解的是 PHP 的核心配置，大多数情况下 PHP 的配置可以决定一个漏洞能否利用。

在代码审计过程中，需要用到很多额外的辅助工具，比如编辑器、代码审计系统以及正则表达式工具，等等。借助这些辅助工具，可以大大提高审计效率，所以第 2 章中将着重介绍这些辅助工具的使用。

Chapter 1 第 1 章

代码审计环境搭建

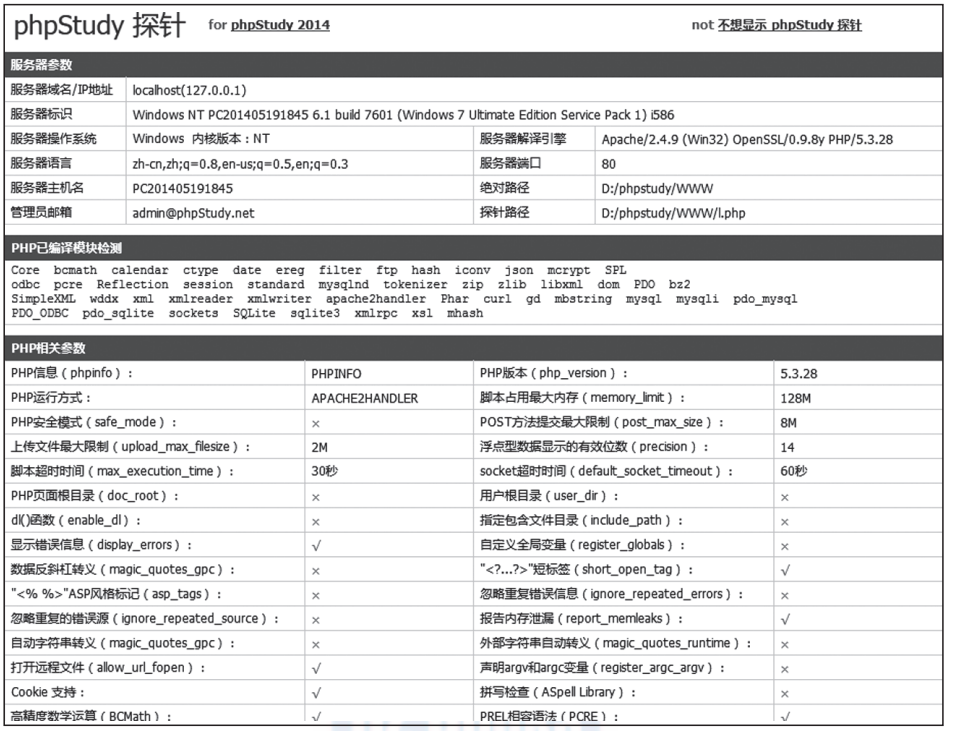
在搭建 PHP 代码审计环境时，因为不是线上环境，为了方便配置环境，所以尽量使用最简单的搭建方法，通常代码审计师都选择安装 wamp/wnmp 或者 lamp/lnmp 等环境集成包，可以快速构建我们所需要的 PHP 运行环境。在选择集成包的时候必须要考虑的是集成环境版本问题，对于 PHP、MySQL、Apache 等服务软件版本，尽量使用目前使用最多的版本，比如 PHP 5.2.X、MySQL 5.0 以上，在针对特殊的漏洞测试时可能还需要安装不同的版本进行测试，还需要在不同的操作系统下测试。

1.1 wamp/wnmp 环境搭建

wamp 组合是使用最多的测试环境，常用的集成环境包有 phpStudy、WampServer、XAMPP 以 AppServ。其中使用最方便且功能最强大的是 phpStudy，该程序包集成最新的 Apache+Nginx+Lighttpd+PHP+MySQL+phpMyAdmin+Zend Optimizer+Zend Loader，一次性安装，不需要配置就可以直接使用，是非常方便、好用的 PHP 调试环境。并且它支持 26 种环境组合随意更改，截至目前，它支持 Apache、Nginx、Lighttpd、IIS6/7/8 中任意一种 WebServer 随时在 PHP 5.2、PHP 5.3、PHP 5.4、PHP 5.5、PHP 5.6 中切换组合使用。我们可以在 phpStudy 官网 www.phpstudy.net 直接下载 phpStudy 安装程序。

我们通过官网链接 http://www.phpstudy.net/phpstudy/phpStudy-x64.zip 下载最新版的

phpStudy。安装后，双击系统桌面上 phpStudy 图标即可启动服务，默认是 Apache+PHP 5.3。这时候访问 http:// localhost/ 即可看到 phpStudy 探针，如图 1-1 所示。



phpStudy 探针　for phpStudy 2014　not 不想显示 phpStudy 探针

服务器参数			
服务器域名/IP地址	localhost(127.0.0.1)		
服务器标识	Windows NT PC201405191845 6.1 build 7601 (Windows 7 Ultimate Edition Service Pack 1) i586		
服务器操作系统	Windows 内核版本：NT	服务器解译引擎	Apache/2.4.9 (Win32) OpenSSL/0.9.8y PHP/5.3.28
服务器语言	zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3	服务器端口	80
服务器主机名	PC201405191845	绝对路径	D:/phpstudy/WWW
管理员邮箱	admin@phpStudy.net	探针路径	D:/phpstudy/WWW/l.php

PHP已编译模块检测

Core bcmath calendar ctype date ereg filter ftp hash iconv json mcrypt SPL
odbc pcre Reflection session standard mysqlnd tokenizer zip zlib libxml dom PDO bz2
SimpleXML wddx xml xmlreader xmlwriter apache2handler Phar curl gd mbstring mysql mysqli pdo_mysql
PDO_ODBC pdo_sqlite sockets SQLite sqlite3 xmlrpc xsl mhash

PHP相关参数			
PHP信息（phpinfo）：	PHPINFO	PHP版本（php_version）：	5.3.28
PHP运行方式：	APACHE2HANDLER	脚本占用最大内存（memory_limit）：	128M
PHP安全模式（safe_mode）：	×	POST方法提交最大限制（post_max_size）：	8M
上传文件最大限制（upload_max_filesize）：	2M	浮点型数据显示的有效位数（precision）：	14
脚本超时时间（max_execution_time）：	30秒	socket超时时间（default_socket_timeout）：	60秒
PHP页面根目录（doc_root）：	×	用户根目录（user_dir）：	×
dl()函数（enable_dl）：	×	指定包含文件目录（include_path）：	×
显示错误信息（display_errors）：	√	自定义全局变量（register_globals）：	×
数据反斜杠转义（magic_quotes_gpc）：	×	"<?...?>"短标签（short_open_tag）：	√
"<% %>"ASP风格标记（asp_tags）：	×	忽略重复错误信息（ignore_repeated_errors）：	×
忽略重复的错误源（ignore_repeated_source）：	×	报告内存泄漏（report_memleaks）：	√
自动字符串转义（magic_quotes_gpc）：	×	外部字符串自动转义（magic_quotes_runtime）：	×
打开远程文件（allow_url_fopen）：	√	声明argv和argc变量（register_argc_argv）：	×
Cookie 支持：	√	拼写检查（ASpell Library）：	×
高精度数学运算（BCMath）：	√	PREL相容语法（PCRE）：	√

图　1-1

我们可以点击界面上的“其他选项”菜单按钮，在菜单中找到“PHP 版本切换”项，更改配置和切换 Web 服务组合，如图 1-2 所示。

图　1-2

点开选项中的“ PHP 版本切换”我们看到 26 种环境组合可以供我们随意切换，如图 1-3 所示。

图 1-3

当我们需要 Nginx 环境时，只需选中 Nginx+PHP*，然后点击“应用”按钮即可。

然而，在启动 Web 服务时偶尔也会遇到服务启动失败的情况，最常见的是 WebServer 服务端口被占用以及 WebServer 配置文件错误。对于端口占用，解决方案有两种，第一种是更换 WebServer 的服务端口，在配置文件中更改监听端口号即可；第二种则是结束占用端口的进程。

如果 Apache 的配置文件 httpd.conf 出错，用命令行模式启动 Apache，并带上参数，Apache 会提示你配置文件哪里有错误，然后就可以针对性地解决，命令是：httpd.exe -w -n "Apache2" -k start ，其中 Apache2 表示服务名。

1.2 lamp/lnmp 环境搭建

在不同的操作系统下，漏洞的测试结果也可能会不一样。简单举例，像文件包含截断，在 Windows 下与 Linux 下截断也有不一样的地方。为了更好地测试漏洞，我们还需要搭建 Linux 下的 PHP 环境。跟 Windows 一样，在 Linux 下也有 PHP 集成环境包，常用的有 phpStudy for Linux、lanmp 以及 XAMPP。因为 phpStudy 支持 Apache、

Nginx、Lighttpd 中任意一种 WebServer 在 PHP 5.2、PHP 5.3、PHP 5.4、PHP 5.5 中 12 种组合的简单切换，为了更方便测试环境调整，所以我们依旧选择 phpStudy 来搭建 lanmp 测试环境，phpStudy 支持 CentOS、Ubuntu、Debian 等 Linux 系统。

我们通过官网 http://lamp.phpstudy.net/ 下载最新版的 phpStudy 到虚拟机并进行安装。安装过程很简单，如果你选择的是下载版，只需要执行如下命令：

```
wget -c http://lamp.phpstudy.net/phpstudy.bin?
chmod +x phpstudy.bin  #权限设置
./phpstudy.bin         #运行安装
```

按提示安装自己所需要的环境组合，如图 1-4 所示。

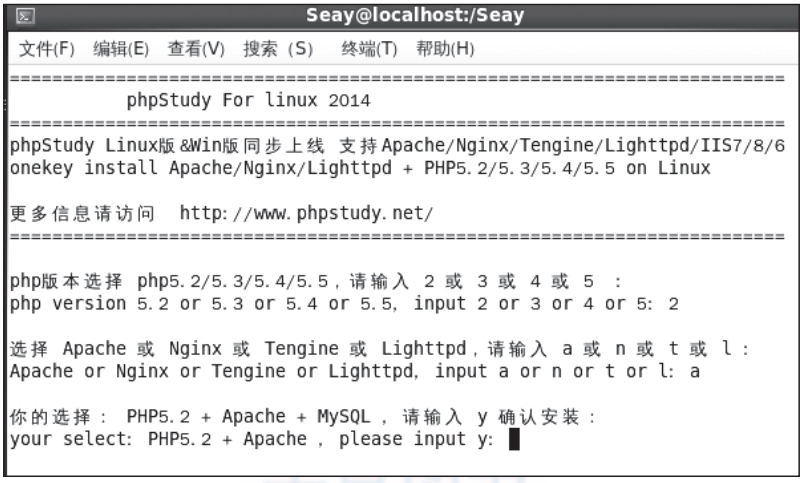


图　1-4

访问 http://localhost（如图 1-5 所示），说明安装成功。

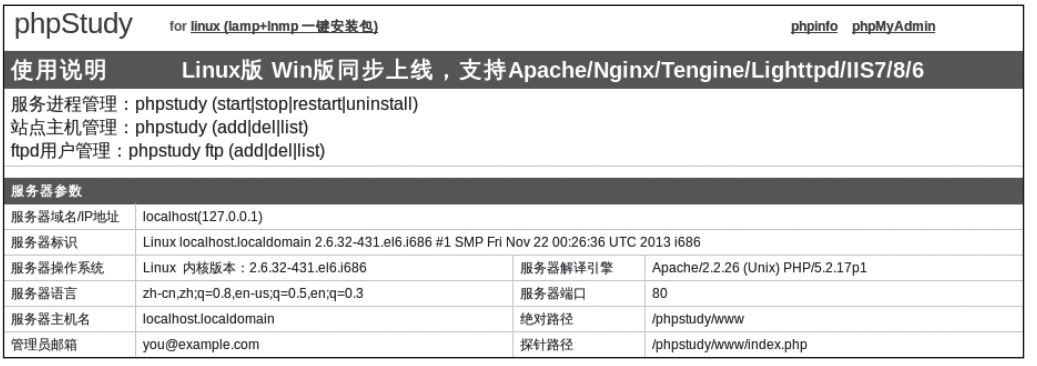


图　1-5

假如你先安装了 Apache+PHP 5.3，想切换成 Nginx+PHP 5.4，只需再运行一次./phpstudy.bin，你会发现有一行是否安装 MySQL 提示，选择“不安装”，这样只需要编译 Nginx+PHP 5.4，从而节省时间，这样只需要几分钟即可。

1.3 PHP 核心配置详解

代码在不同环境下执行的结果也会大有不同，可能就因为一个配置问题，导致一个非常高危的漏洞能够利用；也可能你已经找到的一个漏洞就因为你的配置问题，导致你鼓捣很久都无法构造成功的漏洞利用代码。然而，在不同的 PHP 版本中配置指令也有不一样的地方，新的版本可能会增加或者删除部分指令，改变指令默认设置或者固定设置指令，因此我们在代码审计之前必须要非常熟悉 PHP 各个版本中配置文件的核心指令，才能更高效地挖掘到高质量的漏洞。

我们在阅读 PHP 官方配置说明（http://www.php.net/manual/zh/ini.list.php）之前需要了解几个定义值，即 PHP_INI_* 常量的定义，参见表 1-1。

表 1-1 PHP_INI_* 常量的定义

常 量	含 义
PHP_INI_USER	该配置选项可在用户的 PHP 脚本或 Windows 注册表中设置
PHP_INI_PERDIR	该配置选项可在 php.ini. .htaccess 或 httpd.conf 中设置
PHP_INI_SYSTEM	该配置选项可在 php.ini 或 httpd.conf 中设置
PHP_INI_ALL	该配置选项可在任何地方设置
php.ini only	该配置选项可仅可在 php.ini 中配置

PHP 配置文件指令多达数百项，为了节省篇幅，这里不一一对每个指令进行说明，只列出会影响 PHP 脚本安全的配置列表以及核心配置选项。

1. register_globals（全局变量注册开关）

该选项在设置为 on 的情况下，会直接把用户 GET、POST 等方式提交上来的参数注册成全局变量并初始化值为参数对应的值，使得提交参数可以直接在脚本中使用。register_globals 在 PHP 版本小于等于 4.2.3 时设置为 PHP_INI_ALL，从 PHP 5.3.0 起被废弃，不推荐使用，在 PHP 5.4.0 中移除了该选项。

当 register_globals 设置为 on 且 PHP 版本低于 5.4.0 时，如下代码输出结果为 true。

测试代码：

```
<?php
if($user=='admin') {
echo 'true';
//do something
}
```

执行结果如图 1-6 所示。

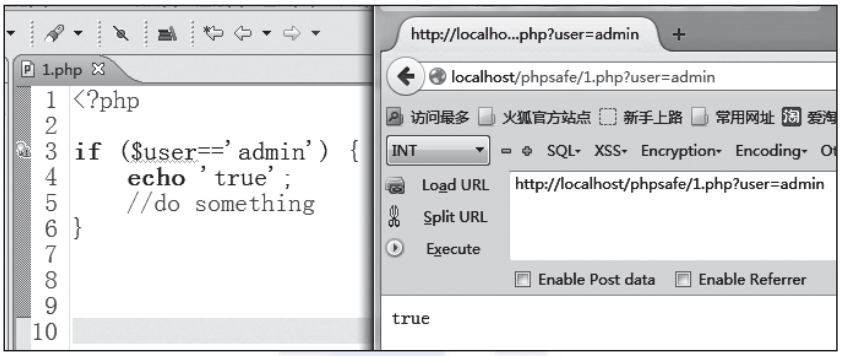


图　1-6

2. allow_url_include（是否允许包含远程文件）

这个配置指令对 PHP 安全的影响不可小觑。在该配置为 on 的情况下，它可以直接包含远程文件，当存在 include ($var) 且 $var 可控的情况下，可以直接控制 $var 变量来执行 PHP 代码。allow_url_include 在 PHP 5.2.0 后默认设置为 off，配置范围是 PHP_INI_ALL。与之类似的配置有 allow_url_fopen，配置是否允许打开远程文件，不过该参数对安全的影响没有 allow_url_include 大，故这里不详细介绍。

配置 allow_url_include 为 on，可以直接包含远程文件。测试代码如下：

```
<?php
include $_GET['a'];
```

测试截图如图 1-7 所示。

3. magic_quotes_gpc（魔术引号自动过滤）

magic_quotes_gpc 在安全方面做了很大的贡献，只要它被开启，在不存在编码或者其他特殊绕过的情况下，可以使得很多漏洞无法被利用，它也是让渗透测试人员很头疼的一个东西。当该选项设置为 on 时，会自动在 GET、POST、COOKIE 变量中的单引号（'）、双引号（"）、反斜杠（\）及空字符（NULL）的前面加上反斜杠（\），但

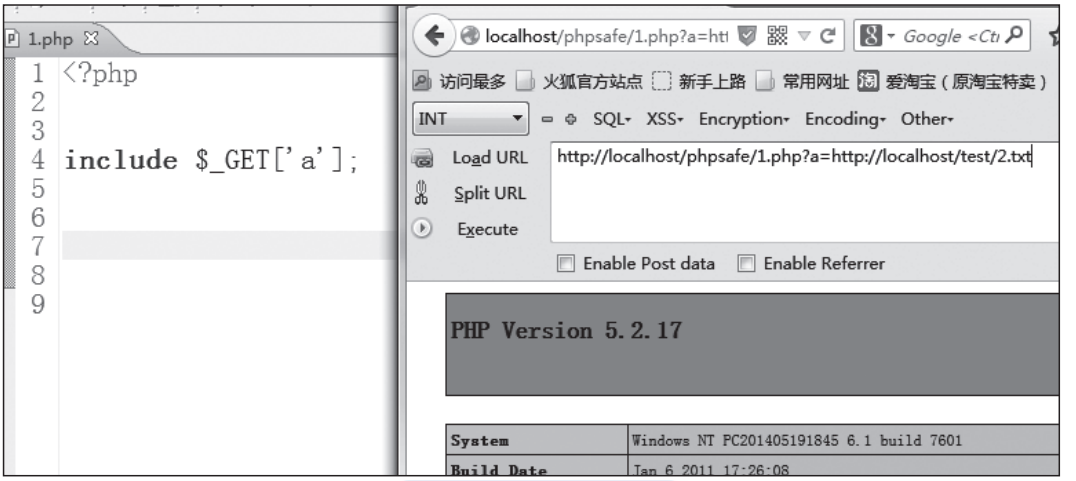


图 1-7

是在 PHP 5 中 magic_quotes_gpc 并不会过滤 $_SERVER 变量，导致很多类似 client-ip、referer 一类的漏洞能够利用。在 PHP 5.3 之后的不推荐使用 magic_quotes_gpc，PHP 5.4 之后干脆被取消，所以你下载 PHP 5.4 之后的版本并打开配置文件会发现找不到这个配置选项。在 PHP 版本小于 4.2.3 时，配置范围是 PHP_INI_ALL；在 PHP 版本大于 4.2.3 时，是 PHP_INI_PERDIR。

测试代码如下：

```
<?php
echo $_GET['seay'];
```

测试结果如图 1-8 所示。

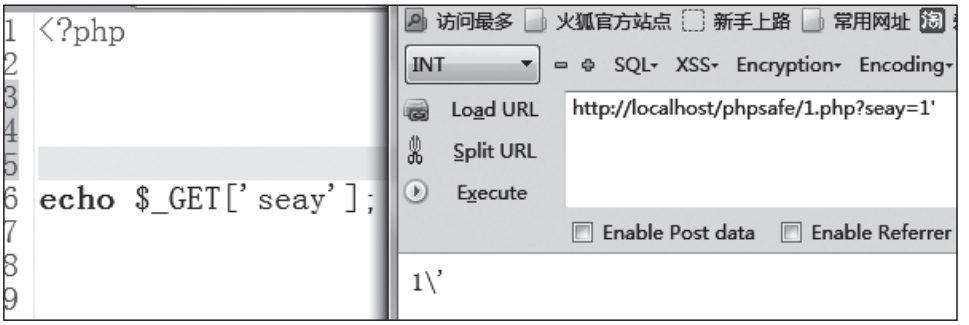


图 1-8

4. magic_quotes_runtime（魔术引号自动过滤）

magic_quotes_runtime 也是自动在单引号（'）、双引号（"）、反斜杠（\）及空字符（NULL）的前面加上反斜杠（\）。它跟 magic_quotes_gpc 的区别是，处理的对象不一样，magic_quotes_runtime 只对从数据库或者文件中获取的数据进行过滤，它的作用也非常大，因为很多程序员只对外部输入的数据进行过滤，却没有想过从数据库获取的数据同样也会有特殊字符存在，所以攻击者的做法是先将攻击代码写入数据库，在程序读取、使用到被污染的数据后即可触发攻击。同样，magic_quotes_runtime 在 PHP 5.4 之后也被取消，配置范围是 PHP_INI_ALL。

有一个点要记住，只有部分函数受它的影响，所以在某些情况下这个配置是可以绕过的，受影响的列表包括 get_meta_tags()、file_get_contents()、file()、fgets()、fwrite()、fread()、fputcsv()、stream_socket_recvfrom()、exec()、system()、passthru()、stream_get_contents()、bzread()、gzfile()、gzgets()、gzwrite()、gzread()、exif_read_data()、dba_insert()、dba_replace()、dba_fetch()、ibase_fetch_row()、ibase_fetch_assoc()、ibase_fetch_object()、mssql_fetch_row()、mssql_fetch_object()、mssql_fetch_array()、mssql_fetch_assoc()、mysqli_fetch_row()、mysqli_fetch_array()、mysqli_fetch_assoc()、mysqli_fetch_object()、pg_fetch_row()、pg_fetch_assoc()、pg_fetch_array()、pg_fetch_object()、pg_fetch_all()、pg_select()、sybase_fetch_object()、sybase_fetch_array()、sybase_fetch_assoc()、SplFileObject::fgets()、SplFileObject::fgetcsv()、SplFileObject::fwrite()。

测试代码如下：

```
#文件1.txt
1'2"3\4

#文件1.php
<?php
ini_set("magic_quotes_runtime", "1");
echo file_get_contents("1.txt");
```

测试结果如图 1-9 所示。

5. magic_quotes_sybase（魔术引号自动过滤）

magic_quotes_sybase 指令用于自动过滤特殊字符，当设置为 on 时，它会覆盖掉 magic_quotes_gpc=on 的配置，也就是说，即使配置了 gpc=on 也是没有效果的。这个指令与 gpc 的共同点是处理的对象一致，即都对 GET、POST、Cookie 进行处理。而它

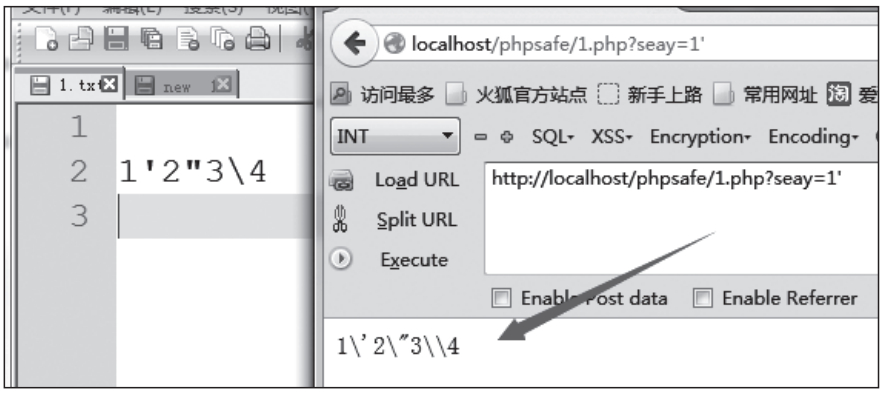


图 1-9

们之前的区别在于处理方式不一样，magic_quotes_sybase 仅仅是转义了空字符和把单引号（'）变成了双引号（"）。与 gpc 相比，这个指令使用得更少，它的配置范围是 PHP_INI_ALL，在 PHP 5.4.0 中移除了该选项。

测试代码如下：

```
<?php
echo $_GET['a'];
?>
```

执行结果如图 1-10 所示。

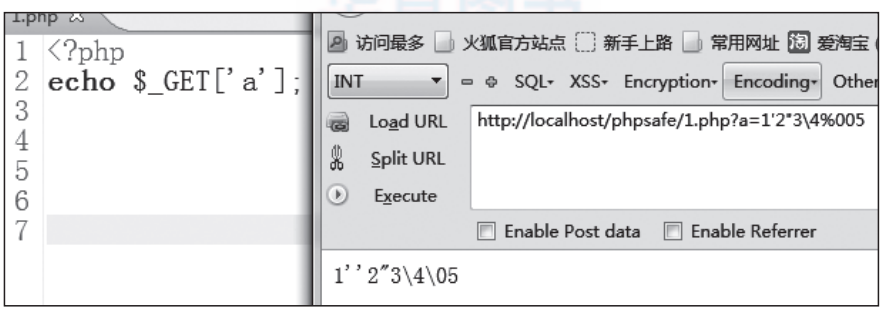


图 1-10

6. safe_mode（安全模式）

安全模式是 PHP 内嵌的一种安全机制，当 safe_mode=on 时，联动可以配置的指令有 safe_mode_include_dir、safe_mode_exec_dir、safe_mode_allowed_env_vars、safe_mode_

protected_env_vars。safe_mode 指令的配置范围为 PHP_INI_SYSTEM，PHP 5.4 之后被取消。

这个配置会出现下面限制：

1）所有文件操作函数（例如 unlink()、file() 和 include()）等都会受到限制。例如，文件 a.php 和文件 c.txt 的文件所有者是用户 a，文件 b.txt 的所有者是用户 b 并且与文件 a.php 不在属于同一个用户的文件夹中，当启用了安全模式时，使用 a 用户执行 a.php，删除文件 c.txt 可成功删除，但是删除文件 b.php 会失败。对文件操作的 include 等函数也一样，如果有一些脚本文件放在非 Web 服务启动用户所有的目录下，需要利用 include 等函数来加载一些类或函数，可以使用 safe_mode_include_dir 指令来配置可以包含的路径。

2）通过函数 popen()、system() 以及 exec() 等函数执行命令或程序会提示错误。如果我们需要使用一些外部脚本，可以把它们集中放在一个目录下，然后使用 safe_mode_exec_dir 指令指向脚本的目录。

下面是启用 safe_mode 指令时受影响的函数、变量及配置指令的完整列表：

apache_request_headers()、ackticks()、hdir()、hgrp()、chmode()、chown()、copy()、dbase_open()、dbmopen()、dl()、exec()、filepro()、filepro_retrieve()、ilepro_rowcount()、fopen()、header()、highlight_file()、ifx_*、ingres_*、link()、mail()、max_execution_time()、mkdir()、move_uploaded_file()、mysql_*、parse_ini_file()、passthru()、pg_lo_import()、popen()、posix_mkfifo()、putenv()、rename()、zmdir()、set_time_limit()、shell_exec()、show_source()、symlink()、system()、touch()。

安全模式下执行命令失败的提示，如图 1-11 所示。

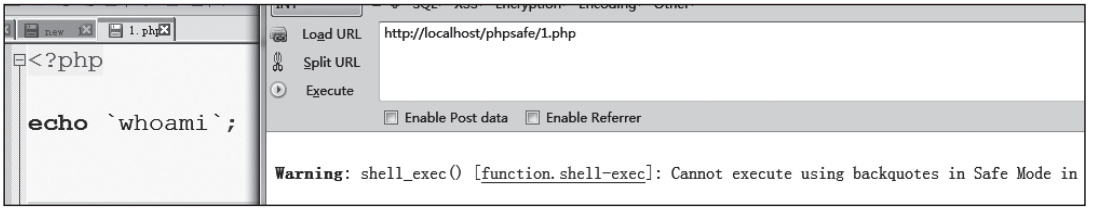


图　1-11

7. open_basedir PHP 可访问目录

open_basedir 指令用来限制 PHP 只能访问哪些目录，通常我们只需要设置 Web 文

件目录即可，如果需要加载外部脚本，也需要把脚本所在目录路径加入到 open_basedir 指令中，多个目录以分号 (;) 分割。使用 open_basedir 需要注意的一点是，指定的限制实际上是前缀，而不是目录名。例如，如果配置 open_basedir =/www/a，那么目录 /www/a 和 /www/ab 都是可以访问的。所以如果要将访问仅限制在指定的目录内，请用斜线结束路径名。例如设置成：open_basedir = /www/a/。

当 open_basedir 配置目录后，执行脚本访问其他文件都需要验证文件路径，因此在执行效率上面也会有一定的影响。该指令的配置范围在 PHP 版本小于 5.2.3 时是 PHP_INI_SYSTEM，在 PHP 版本大于等于 5.2.3 是 PHP_INI_ALL。

8. disable_functions（禁用函数）

在正式的生产环境中，为了更安全地运行 PHP，也可以使用 disable_functions 指令来禁止一些敏感函数的使用。当你想用本指令禁止一些危险函数时，切记要把 dl() 函数也加到禁止列表，因为攻击者可以利用 dl() 函数来加载自定义的 PHP 扩展以突破 disable_functions 指令的限制。

本指令配置范围为 php.ini only。配置禁用函数时使用逗号分割函数名，例如：disable_functions=phpinfo,eval,passthru,exec,system。

9. display_errors 和 error_reporting 错误显示

display_errors 表明是否显示 PHP 脚本内部错误的选项，在调试 PHP 的时候，通常都把 PHP 错误显示打开，但是在生产环境中，建议关闭 PHP 错误回显，即设置 display_errors=off，以避免带来一些安全隐患。在设置 display_errors=on 时，还可以配置的一个指令是 error_reporting，这个选项用来配置错误显示的级别，可使用数字也可使用内置常量配置，数字格式与常量格式的详细信息如表 1-2 所示。

表 1-2 数字格式与常量格式

数字格式	常量格式	数字格式	常量格式
1	E_ERROR	128	E_COMPILE_WARNING
2	E_WARNING	256	E_USER_ERROR
4	E_PARSE	512	E_USER_WARNING
8	E_NOTICE	1024	E_USER_NOTICE
16	E_CORE_ERROR	2047	E_ALL
32	E_CORE_WARNING	2048	E_STRICT
64	E_COMPILE_ERROR		

这两个指令的配置范围都是 PHP_INI_ALL。

会影响到安全的指令大致就介绍到这里，表 1-3 列出一些常用指令以及对应的说明。

表 1-3　常用指令及说明

指　令	可配置范围	说　明
safe_mode_gid	PHP_INI_SYSTEM	以安全模式打开文件时默认使用 UID 来比对；设置本指令为 on 时使用 GID 做宽松的比对
expose_php	php.ini only	是否在服务器返回信息 HTTP 头显示 PHP 版本
max_execution_time	PHP_INI_ALL	每个脚本最多执行秒数
memory_limit	PHP_INI_ALL	每个脚本能够使用的最大内存数量
log_errors	PHP_INI_ALL	将错误输入到日志文件
log_errors_max_len	PHP_INI_ALL	设定 log_errors 的最大长度
variables_order	PHP_INI_PERDIR	此指令描述了 PHP 注册 GET、POST、Cookie、环境和内置变量的顺序，注册使用从左往右的顺序 , 新的值会覆盖旧的值 .
post_max_size	PHP_INI_PERDIR	PHP 可以接受的最大的 POST 数据大小
auto_prepend_file	PHP_INI_PERDIR	在任何 PHP 文档之前自动包含的文件
auto_append_file	PHP_INI_PERDIR	在任何 PHP 文档之后自动包含的文件
extension_dir	PHP_INI_SYSTEM	可加载的扩展 (模块) 的目录位置
file_uploads	PHP_INI_SYSTEM	是否允许 HTTP 文件上传
upload_tmp_dir	PHP_INI_SYSTEM	对于 HTTP 上传文件的临时文件目录
upload_max_filesize	PHP_INI_SYSTEM	允许上传的最大文件大小

Chapter 2 第 2 章

审计辅助与漏洞验证工具

在代码审计和开发中，我们都需要一些代码编辑器来编辑代码，或者调试代码，也需要一些工具来验证漏洞是否存在。而各个编辑器也有所差异，所谓宝刀配英雄，使用一款好的编辑器能帮助你所向披靡，更简单轻松地写代码。而对于审计师来说；代码审计软件也是如此，一款好的代码审计工具可以使审计师在短时间内快速发现代码问题。本章将详细介绍几款常用的代码编辑器和代码审计软件以及一些常用的漏洞验证辅助工具。

2.1 代码编辑器

不管是做开发还是代码审计，一款顺手的代码编辑器必不可少，代码编辑器从轻量级到功能复杂强大的完备型，从免费到商业，都有很多款供我们选择，我们可以根据需要选择最适合的一款，常用的轻量级代码编辑器有 Nodepad++、Editplus、UltraEdit、PSPad、Vim、Gedit，等等，这些都是都是通用型文本编辑器，支持多种编程语言代码高亮，优点是操作简单，启动快并且对文本操作很方便。常用的完备型 PHP 开发软件也不少，这类编辑器主要的优点是功能全，对代码调试、代码提示等都支持得比较好，使我们在开发的时候 bug 更少，开发效率更高，常用的有 Zend Studio、PhpStorm、PhpDesigner 以及 NetBeans 等。

如果你用编辑器来做开发，并且代码量比较大，建议你使用 Zend Studio。如果用

来做代码审计或者少量代码的开发，建议使用 Nodepad++ 这类轻量级文本编辑器。

2.1.1　Notepad++

Notepad++ 是一套非常有特色的开源纯文字编辑器（许可证：GPL），运行于 Windows 系统，有完整的中文接口及支持多国语言撰写的功能（UTF8 技术）。它的功能比 Windows 中的 Notepad（记事本）强大，除了可以用来编辑一般的纯文字文件之外，也十分适合轻量开发的编辑器。Notepad++ 不仅有语法高亮显示功能，也有语法折叠功能，并且支持宏以及扩充基本功能的外挂模组。

Notepad++ 可以安装免费使用。支持如下语言的代码高亮显示：C、C++、Java、C#、XML、HTML、PHP、ASP、AutoIt、DOS 批处理、CSS、ActionScript、Fortran、Gui4Cli、Haskell、JSP、Lisp、Lua、Matlab、NSIS、Objective-C、Pascal、Python、JavaScript 等。

Notepad++ 拥有非常多强大的功能，特别是对文本操作非常灵活，这是笔者用得最多的一个文本编辑器，经常用来做一些有特定格式的文本批量替换、搜索、去重，等等。当然，它的强大不止如此。下面简单介绍下它的核心功能：

1）内置支持多达 27 种语法高亮显示（包括各种常见的源代码、脚本，能够很好地支持 .nfo 文件查看），还支持自定义语言。

2）可自动检测文件类型，根据关键字显示节点，节点可自由折叠 / 展开，还可显示缩进引导线，代码显示得很有层次感。

3）可打开双窗口，在分窗口中又可打开多个子窗口，显示比例。

4）提供了一些有用工具，如 邻行互换位置、宏功能等。

5）可显示选中的文本的字节数（而不是一般编辑器所显示的字数，这在某些情况下很方便，比如软件本地化）。

6）正则匹配字符串及批量替换，也支持批量文件操作。

7）强大的插件机制，扩展了编辑能力，如 Zen Coding。

我们可以在官网 Notepad++ 官网（notepad-plus-plus.org）下载最新版。主界面如图 2-1 所示。

2.1.2　UltraEdit

UltraEdit（官网 www.ultraedit.com）是一款功能强大的文本编辑器，不过它不是开源软件，官网售价 79.95 美元，可以完美运行在 Windows、Linux 以及 Mac 系统上。

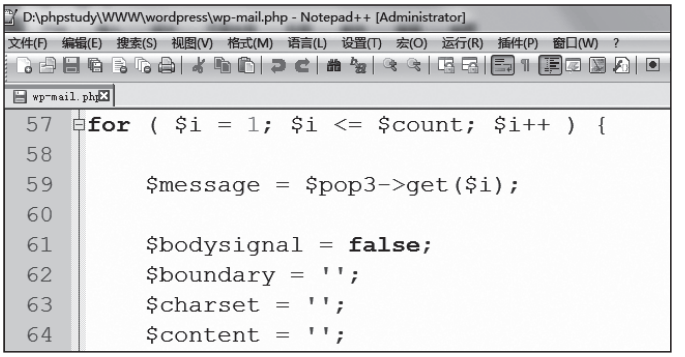


图 2-1

这款编辑器不仅可以编辑文本，还支持十六进制查看以及编辑。可以直接在上面修改 exe 等文件，如图 2-2 所示。

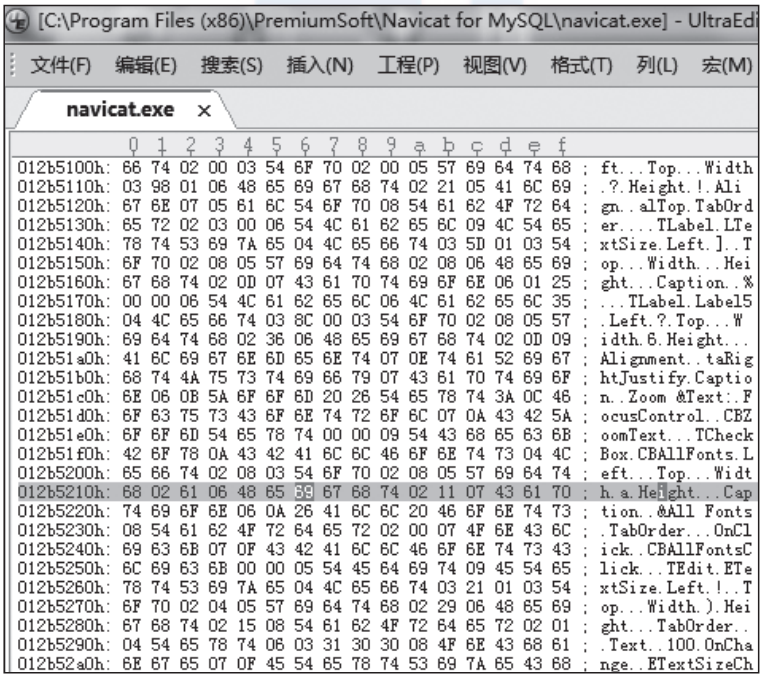


图 2-2

该编辑器支持将近二十种编程语言的语法高亮显示，可同时编辑多个文件，支持打开超过 4GB 以上的文件，支持多种编码转换、排序去重。通过配置使用的脚本运行程序路径，比如 php.exe 的路径，就可以在使用 UltraEdit 编辑 PHP 代码的时候直接执行

代码。再结合它的代码补全功能，它也算得上一款不错的代码编辑器。要实现这个功能，首先在“高级→工具栏配置”中配置一些执行环境参数，在“命令行”的位置填入你的 PHP 文件路径，在“菜单项目名称”上写你想填的菜单栏名称，这里写的是 php.exe，在“工作目录”中写上你的 PHP exe 路径，然后点击“确定”按钮，即可新建一个文件。在“高级”菜单里面点一下添加的 php.exe（菜单栏名称）即可执行代码，如图 2-3 所示。

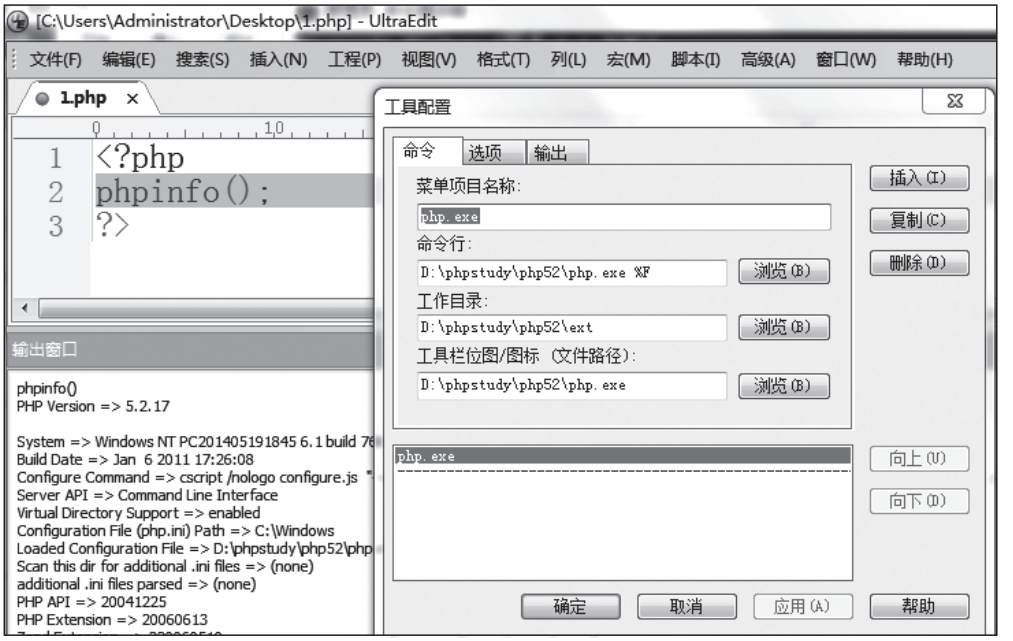


图　2-3

另外一个比较好的功能是文件对比。这个功能也是经常会用到的，特别是我们在分析开源程序发布的官方补丁时，比如 Phpcms 某天发布了一个代码执行漏洞修补补丁，那么我们就可以在官网下载补丁文件，然后利用 UltraEdit 的文件对比功能来快速找到修改了哪段代码，修改的部分是不是成功修补了这个漏洞，或者未公开的漏洞。也可以根据这个方法快速找到漏洞在哪里。

这个功能可以在菜单栏“文件→比较文件”中找到，然后选择要对比的两个以上文件，勾选“比较选项”里面以忽略开头的所有选项，点击“比较”按钮即可，如图 2-4 所示。

如果比较的文件有不同的地方，它会用红色标出，如图 2-5 所示。

UltraEdit 被公认为程序员必备的编辑器，是能够满足你一切编辑需要的编辑器。


比较文件/文件夹
比较模式
文件(I)
文件夹(L)
比较(O)
取消(C)
帮助(H)
多方比较
2 方比较(W)
3 方比较(R)
文件比较
比较的第 1 个文件(F):
E:\php安全研究\code\2.php
比较的第 2 个文件(S):
E:\php安全研究\code\1.php
比较的第 3 个文件(T):
比较选项
临时文件与原始文件比较(M)
高亮文本比较(D)
忽略大小写 (比较模式)(G)
忽略空格 (文本模式)(P)
忽略空行 (文本模式)(B)
忽略行尾符(E)
深层文件夹比较(U)
文件比较类型
文本(X)
二进制(Y)
智能二进制(A)
编辑器平铺
不平铺(N)
垂直平铺(V)
水平平铺(Z)
提示: 你正使用 UltraCompare Pro。可以进行完整的文件/文件夹/合并/比较操作。
在此下载 UltraCompare Professional

图 2-4

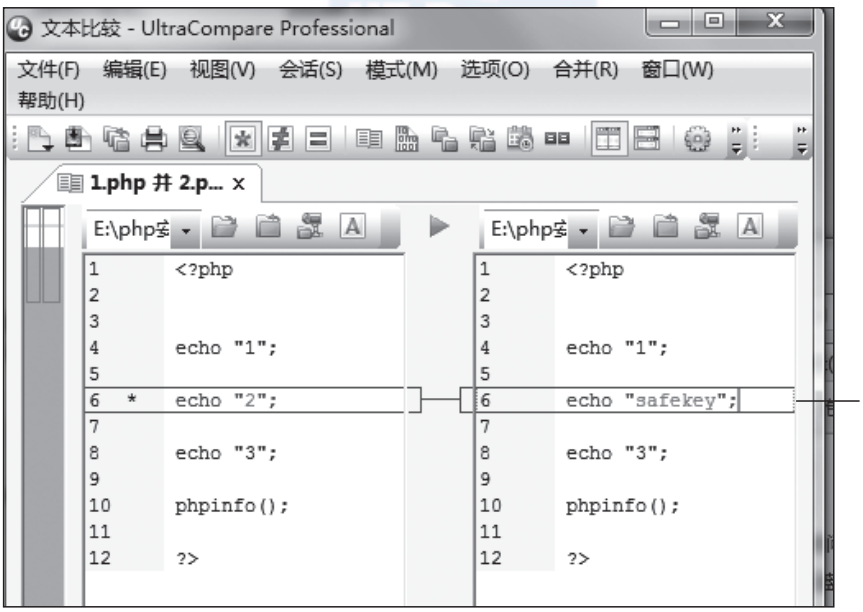

文本比较 - UltraCompare Professional
文件(F) 编辑(E) 视图(V) 会话(S) 模式(M) 选项(O) 合并(R) 窗口(W)
帮助(H)
1.php 并 2.p... x
<?php
echo "1";
echo "2";
echo "3";
phpinfo();
?>
<?php
echo "1";
echo "safekey";
echo "3";
phpinfo();
?>
红色

图 2-5

2.1.3　Zend Studio

Zend Studio 与 PHP 出自同一家公司，也可以说 Zend Studio 是 PHP 官方专门开发出来用来编写 PHP 代码的代码编辑器。Zend Studio 是目前用户量最大的 PHP 开发工具，也是屡获大奖的专业 PHP 集成开发环境，具备功能强大的专业编辑工具和调试工具，支持 PHP 语法高亮显示，支持语法自动填充功能，支持书签功能，支持语法自动缩排和代码复制功能，内置一个强大的 PHP 代码调试工具，支持本地和远程两种调试模式，支持多种高级调试功能，可以完美运行在目前主流的 Windows、Linux 以及 Mac 操作系统上。官网是 http://www.zend.com/en/products/studio。

Zend Studio 10.6 版本的界面截图如图 2-6 所示。

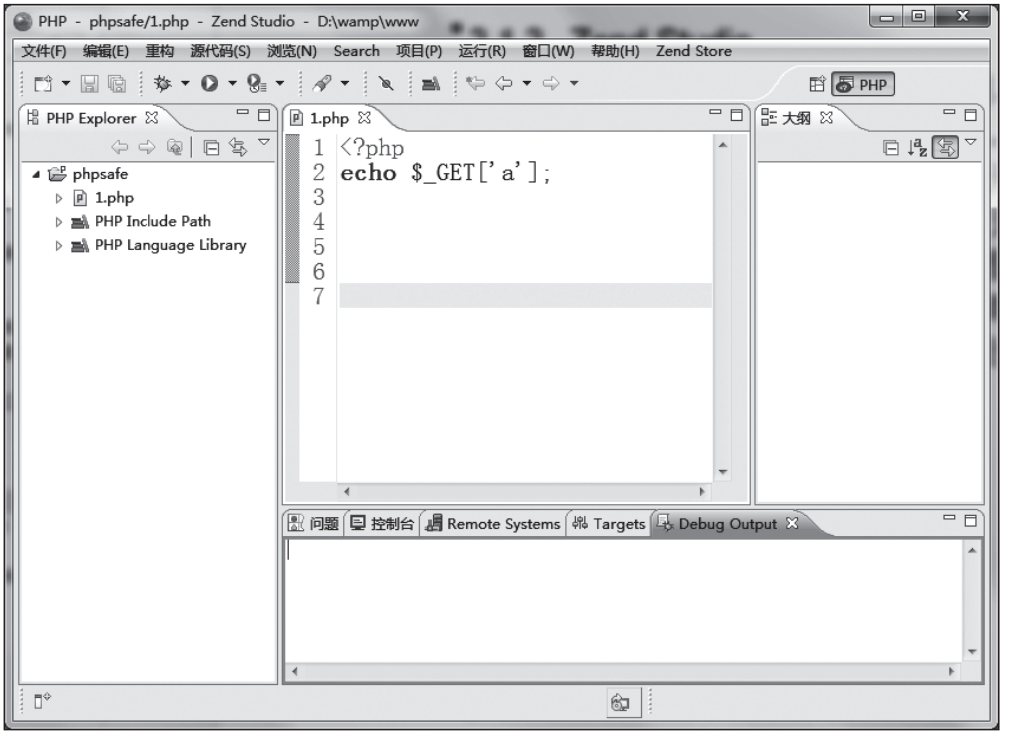


图　2-6

Zend Studio 最令笔者最喜欢的功能是代码提示功能，实际上，只要这个功能做得好的编辑器，笔者都非常喜欢，因为这非常人性化，可以让我们不用去记那么多函数，等你经常用的编程语言超过了 6 种以上，你就会深有感触。代码提示功能如图 2-7 所示。

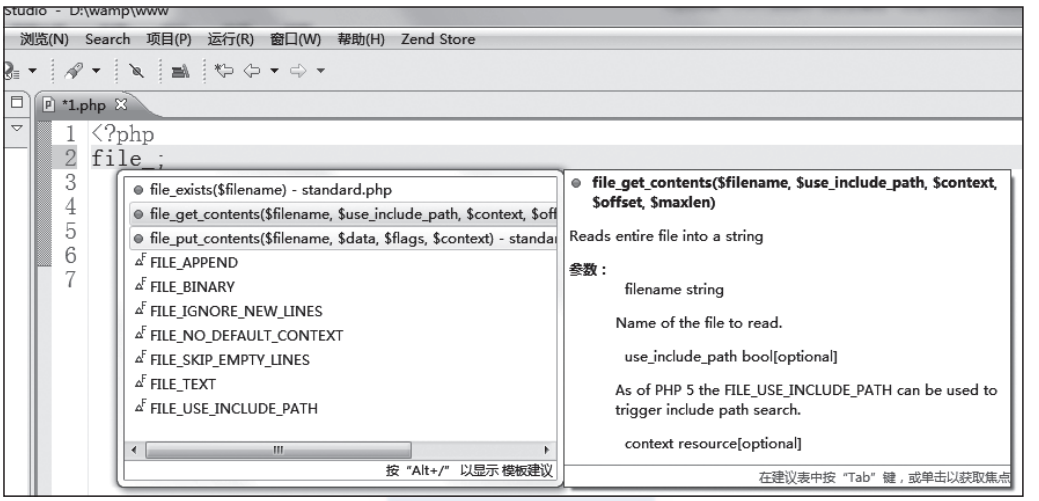


图 2-7

另外 Zend Studio 在代码调试方面也非常强大，支持多种调试模式，利用它的调试功能，可以让我们非常快地发现 bug 位置，监控数据传递过程和函数运行情况，如图 2-8 所示。

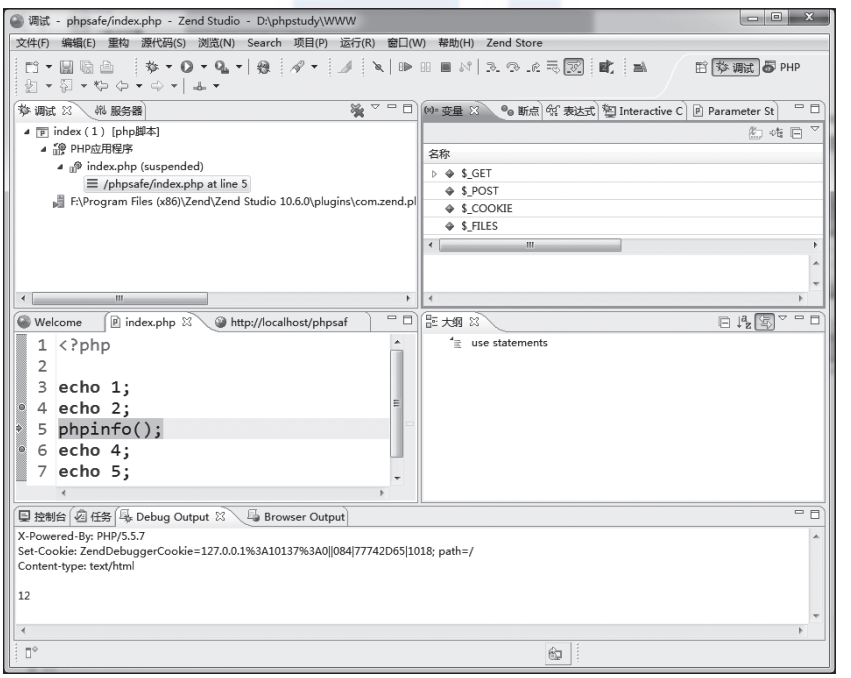


图 2-8

2.2　代码审计工具

代码审计工具是一类辅助我们做白盒测试的程序，它可以分很多类，例如安全性审计以及代码规范性审计，等等。当然，也可以按它能审计的编程语言分类，目前商业性的审计软件大多支持多种编程语言，也有个人或团队开发的免费开源审计软件，像笔者的“Seay 源代码审计系统”就是开源程序。使用一款好的代码审计软件可以极大地降低审计成本，可以帮助审计师快速发现问题所在，同时也能降低审计门槛，但也不能过分依赖审计软件。目前常用的代码安全审计软件还有 Fortify SCA、RIPS、FindBugs、Codescan 等。下面介绍几款常用代码安全审计工具。

2.2.1　Seay 源代码审计系统

这是笔者基于 C# 语言开发的一款针对 PHP 代码安全性审计的系统，主要运行于 Windows 系统上。这款软件能够发现 SQL 注入、代码执行、命令执行、文件包含、文件上传、绕过转义防护、拒绝服务、XSS 跨站、信息泄露、任意 URL 跳转等漏洞，基本上覆盖常见 PHP 漏洞。另外，在功能上，它支持一键审计、代码调试、函数定位、插件扩展、自定义规则配置、代码高亮、编码调试转换、数据库执行监控等数十项强大功能。主界面如图 2-9 所示。

图　2-9

Seay 源代码审计系统主要特点如下：

1）一键自动化白盒审计，新建项目后，在菜单栏中打开“自动审计”即可看到自动审计界面。点击“开始”按钮即可开始自动化审计。当发现可疑漏洞后，则会在下方列表框显示漏洞信息，双击漏洞项即可打开文件跳转到漏洞代码行并高亮显示漏洞代码行，如图 2-10 所示。

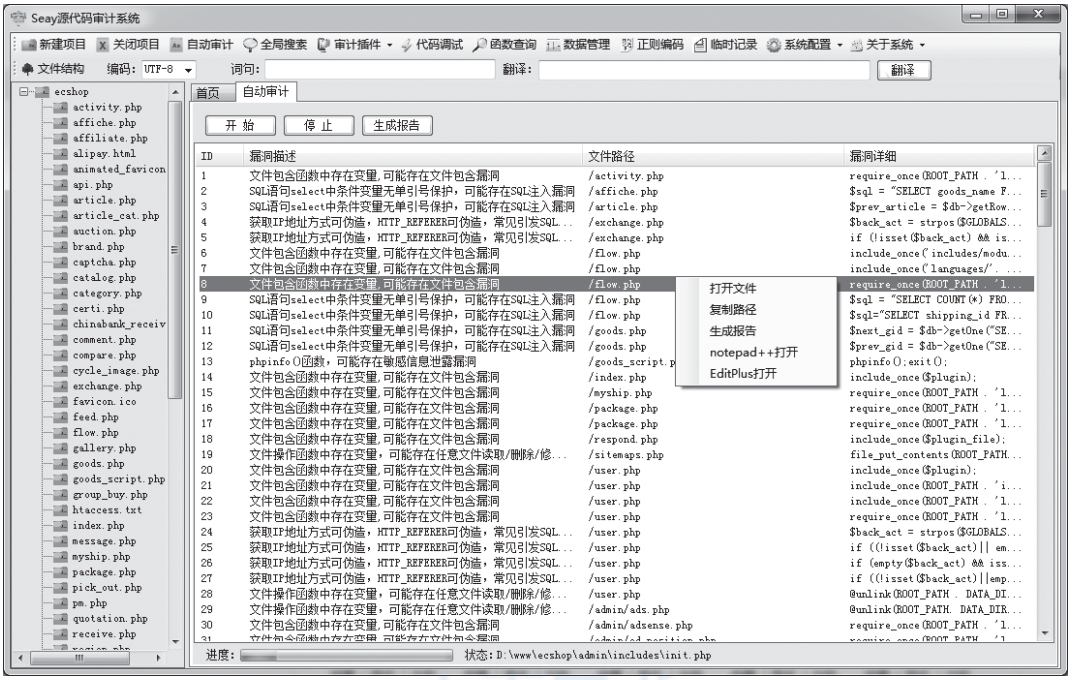


图 2-10

2）代码调试，代码调试功能极大地方便了审计师在审计过程中测试代码。可以在编辑器中选中代码，然后点击右键选择“调试选中”即可将代码在调试界面打开，如图 2-11 所示。

3）正则编码，Seay 源代码审计系统集成了实时正则调试功能，考虑到特殊字符无法直接在编辑框进行输入，在实时正则调试功能中还支持对字符串实时解码后调试。另外，支持 MD5、URl、Base64、Hex、ASCII、Unicode 等多种编码解码转换功能，如图 2-12 所示。

4）自定义插件及规则，Seay 源代码审计系统支持插件扩展，并且插件的开发非常简单，只需要将插件的 dll 文件放入到安装目录下的 plugins 文件夹内即可自动加载插件。目前自带插件包括黑盒 + 白盒的信息泄露审计以及 MySQL 数据库执行监控。


Seay源代码审计系统 D:\www\ecshop\exchange.php
新建项目
关闭项目
自动审计
全局搜索
审计插件
代码调试
函数查询
数据管理
正则编码
临时记录
文件结构
编码: GBK
词句:
翻译:
ecshop
activity.php
affiche.php
affiliate.php
alipay.html
animated_favicon
api.php
article.php
article_cat.php
auction.php
brand.php
captcha.php
catalog.php
category.php
certi.php
chinabank_receiv
comment.php
compare.php
cycle_image.php
exchange.php
favicon.ico
feed.php
flow.php
gallery.php
goods.php
goods_script.php
group_buy.php
htaccess.txt
index.php
首页
自动审计
代码调试
PHP代码
<?php
phpinfo();
?>
执 行
执行结果
phpinfo()
PHP Version => 5.2.1
System => Windows NT PC201407182221 6.1 build 7601
Build Date => Feb 7 2007 23:10:31
Configure Command => cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API => Command Line Interface
Virtual Directory Support => enabled
Configuration File (php.ini) Path => C:\Windows
PHP API => 20041225
PHP Extension => 20060613

图　2-11

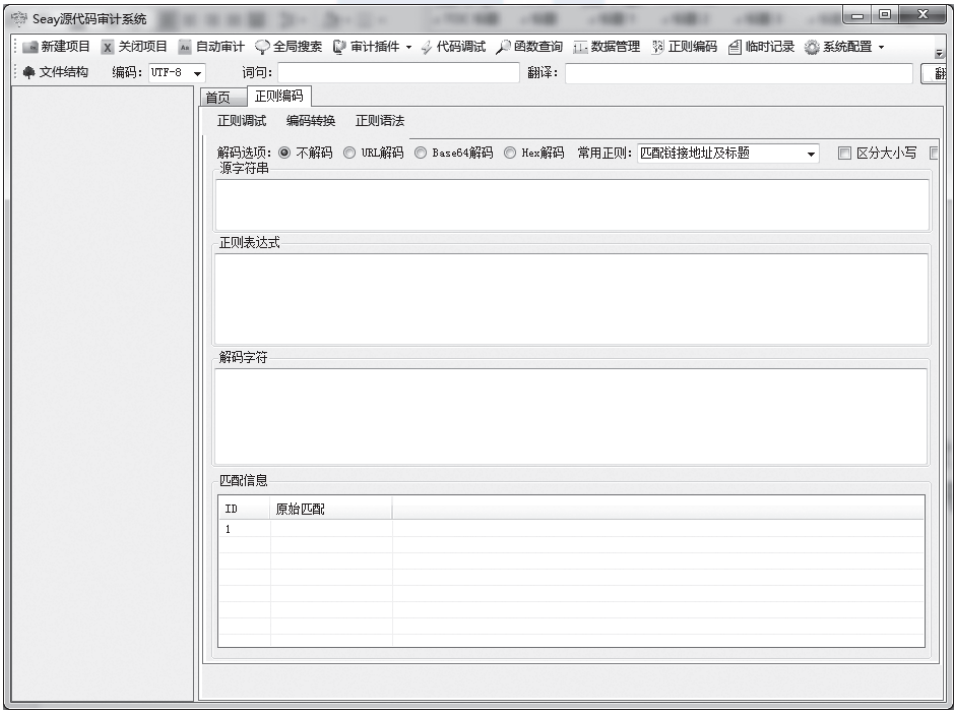

Seay源代码审计系统
新建项目
关闭项目
自动审计
全局搜索
审计插件
代码调试
函数查询
数据管理
正则编码
临时记录
系统配置
文件结构
编码: UTF-8
词句:
翻译:
首页
正则编码
正则调试
编码转换
正则语法
解码选项: 不解码 URL解码 Base64解码 Hex解码
常用正则: 匹配链接地址及标题
区分大小写
源字符串
正则表达式
解码字符
匹配信息
ID
原始匹配
1

图　2-12

除了上述功能外，它还支持自定义审计规则，在规则配置界面中即可添加或修改以及禁用、删除规则，还可针对审计过程做很多审计习惯优化，使得程序简单容易上手。

2.2.2 Fortify SCA

Fortify SCA 是由惠普研发的一款商业软件产品，针对源代码进行专业的白盒安全审计，当然，它是收费的，而且这种商业软件一般都价格不菲。它有 Windows、Linux、UNIX 以及 Mac 版本，通过内置的五大主要分析引擎（数据流、控制流、语义、结构以及配置）对应用软件的源代码进行静态分析。关于这五大分析引擎的介绍如表 2-1 所示。

表 2-1 五大分析引擎概述

分 析 器	描 述
数据流	数据流分析器可以检测涉及将被感染数据（用户控制的输入）用于危险用途的潜在漏洞。数据流分析器使用全局的、程序间的感染繁殖分析，检测 source（用户输入的站点）与 sink（危险的函数调用或者操作）之间的数据流。例如，数据流分析器可以检测一个用户控制的特别长的字符串输入是否正被复制到一个固定长度的缓冲区中，还可以检测一个用户控制的字符串是否正被用来构建 SQL 查询文本
控制流	控制流分析器可以检测潜在的危险操作的执行顺序。通过分析程序中的控制流路径，控制流分析器能确定在执行一系列操作时是否遵循了特定的顺序。例如，控制流分析可以检测 check/time 的时间和未经初始化的变量，并检查实用程序，如 XML 阅读器，是否在使用前做了正确的配置
语义	语义分析器可以在程序内部层面检测可能会引发潜在危险的函数和 API 的各种使用情况。其特定的逻辑会搜索 buffer overflow、format string 和各种执行路径的问题，但并不局限于这几个类别。任何存在潜在危险的函数调用都可以通过语义分析器进行标记。例如，语义分析器可以检测 Java 中的过时函数和 C/C++ 中的不安全函数，如 gets()
结构	结构分析器用于检测可能存在危险的结构缺陷或程序定义。通过理解程序构建的方式，结构分析器能够识别出以其他方式难以检测到的问题，原因是这些技术涉及的范围很广，包括有关声明和变量函数的使用。例如，结构分析器检测在 Java Servlet 中成员变量的赋值，识别未被声明为 static final 的记录器的使用，并标记那些由于断言为始终错误而永不被执行的 dead code 实例
配置	配置分析器可以在应用程序的配置文件中搜索错误、缺陷和违反规则的策略漏洞。例如，配置分析器可检查网络应用程序的某一用户会话的超时是否合理

Fortify SCA 是目前支持最多编程语言的审计软件。它支持的编程语言如下所示：

ASP.NET	VB6
VB.NET	Java
C#.NET	JSP
ASP	JavaScript
VBScript	HTML
Action Script	XML
Objective-C	C/C++

ColdFusion 5.0	PHP
Python	T-SQL (MSSQL)
COBOL	PL/SQL (Oracle)
SAP-ABAP	

分析的过程中与它特有的软件安全漏洞规则集进行全面的匹配、搜索，在最终的漏洞结果中，包括详细的漏洞信息，以及漏洞相关的安全知识说明和修复意见。

2.2.3　RIPS

RIPS 是一款基于 PHP 开发的针对 PHP 代码安全审计的软件。另外，它也是一款开源软件，由国外安全研究员 Johannes Dahse 开发，程序只有 450KB，目前能下载到的最新版是 0.54，笔者发现这款程序在 2013 年 2 月已经暂停更新。在写这段文字之前笔者特意读过它的源码，它最大的亮点在于调用了 PHP 内置解析器接口 token_get_all，并且使用 Parser 做了语法分析，实现了跨文件的变量及函数追踪，扫描结果中非常直观地展示了漏洞形成及变量传递过程，误报率非常低。RIPS 能够发现 SQL 注入、XSS 跨站、文件包含、代码执行、文件读取等多种漏洞，支持多种样式的代码高亮。比较有意思的是，它还支持自动生成漏洞利用。

图 2-13 为 RIPS 截图。

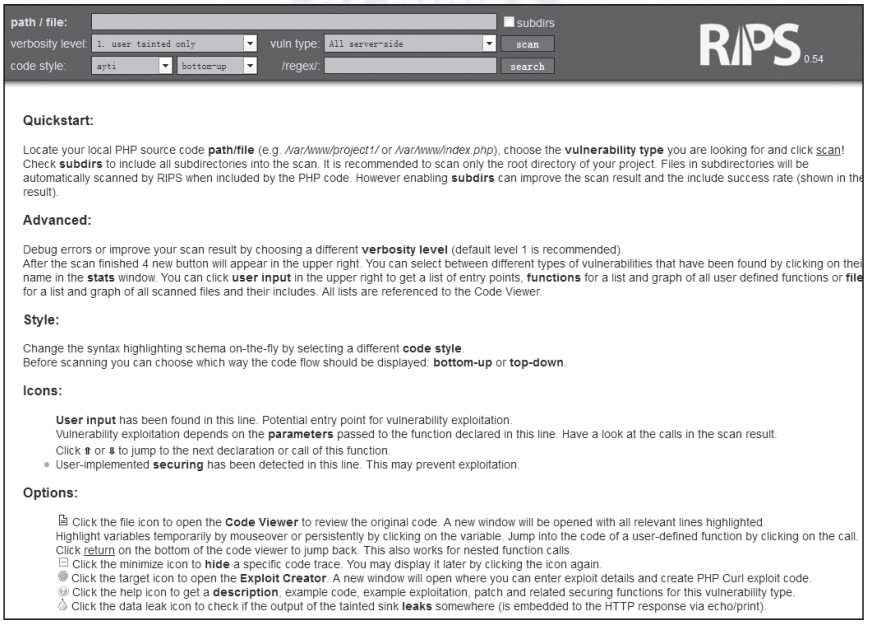


图　2-13

RIPS 的使用非常简单，只需在主界面填入我们要扫描的路径，其余配置可根据自己的需要设置。完成设置后点击 scan 按钮即可开始自动审计。扫描结束后，程序会显示漏洞数量、漏洞比例等信息。查看漏洞详情时，只需点击提示漏洞处的“ –”即可显示漏洞源代码和变量过程，如图 2-14 所示。

笔者通过 RIPS 发现 ecshop 文件包含漏洞，图 2-15 所示为 RIPS 漏洞扫描详细结果。

代码查看也非常方便，只需要点击“ review code”即可跳转到漏洞代码处，将鼠标指针悬浮在变量上，同文件的变量会高亮显示，如图 2-16 所示。

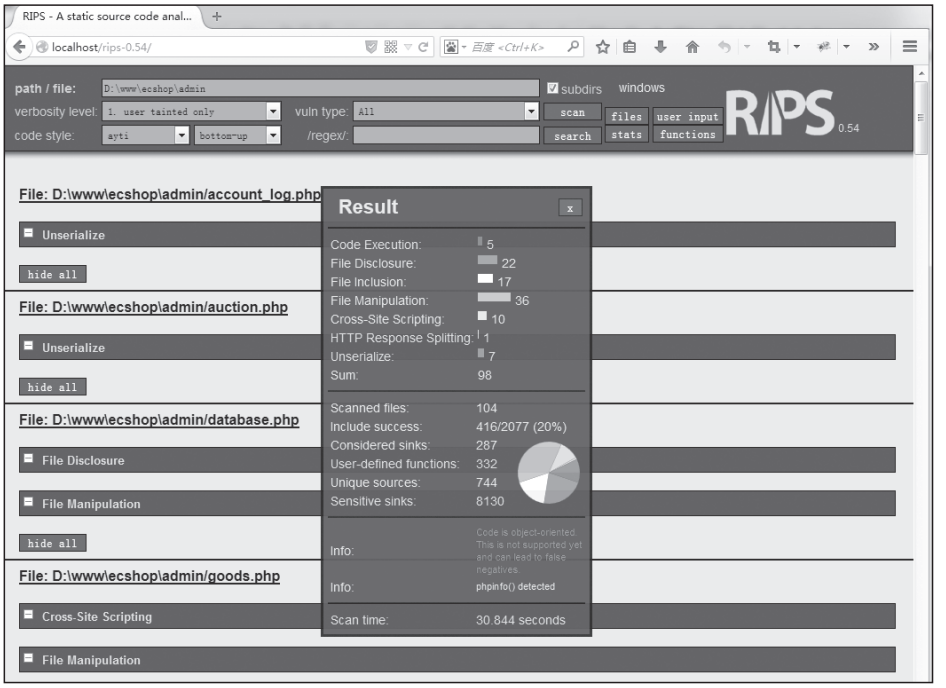


图 2-14

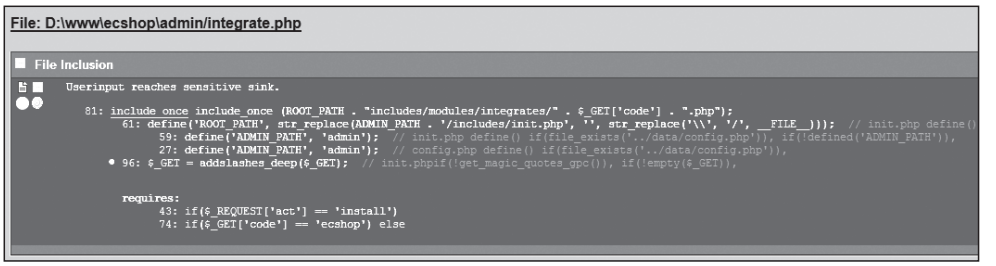


图 2-15

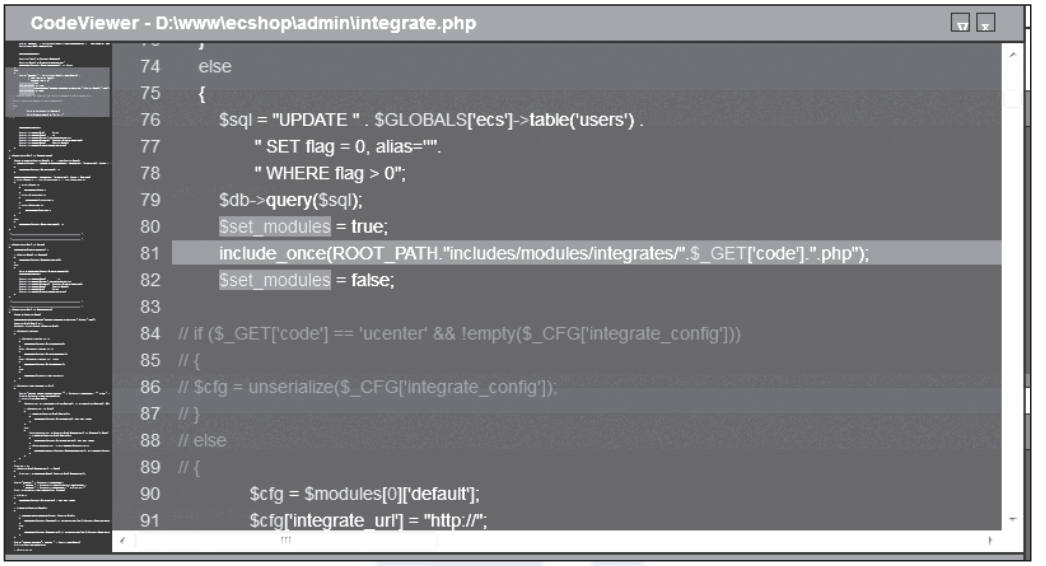


图　2-16

2.3　漏洞验证辅助

不管是借助代码审计工具还是读 PHP 文件发现的漏洞，我们都需要验证漏洞是否真实可用。这就需要借助一些工具来帮助我们快速测试漏洞，或者在某些情况下，比如部分代码不可读时，我们可以在不继续往下读代码的情况下测试漏洞，做基于模糊测试的漏洞验证。主要的辅助工具分为数据包请求工具类、暴力枚举类、编码转换及加解密类。当然，还有一些正则调试和 SQL 执行监控等软件。下面只列举一些常用的，根据不同的漏洞和环境需要搭配不同的工具来测试。

2.3.1　Burp Suite

Burp Suite 是一款基于 Java 语言开发的安全测试工具，使用它需要安装 Java 运行环境。这款软件只有不到 10MB 的大小，但是其强大的功能受到几乎所有安全人员的青睐。Burp Suite 主要分为 Proxy、Spider、Scanner、Intruder、Repeater、Sequencer、Decoder 和 Comparer 几个大模块。下面简单介绍下各个模块：

- Proxy（代理） Burp Suite 的代理抓包功能是这款软件的核心功能，当然也是使用最多的功能。使用这个代理，可以截获并修改从客户端到 Web 服务器的 HTTP/HTTPS 数据包。

- Spider（蜘蛛） 用来分析网站目录结构，爬行速度非常不错，爬行结果会显示在 Target 模块中，支持自定义登录表单，让它自动提交数据包进行登录验证。
- Scanner（扫描器） 用于发现 Web 程序漏洞，它能扫描出 SQL 注入、XSS 跨站、文件包含、HTTP 头注入、源码泄露等多种漏洞。
- Intruder（入侵） 用来进行暴力破解和模糊测试。它最强大的地方在于高度兼容的自定义测试用例，通过 Proxy 功能抓取的数据包可以直接发送到 Intruder，设置好测试参数和字典、线程等，即可开始漏洞测试。
- Repeater（中继器） 用于数据修改测试，通常在测试一些像支付等逻辑漏洞的时候经常需要用到它，只需要设置好代理拦截数据包，然后发送到 Repeater 模块即可对数据随意修改之后再发送。
- Sequencer（会话） 用于统计、分析会话中随机字符串的出现概率，从而分析 Session、Token 等存在的安全风险。
- Decoder（解码） 用于对字符串进行编码和解码，支持百分号、Base64、ASCII 等多种编码转换，还支持 Md5、sha 等 Hash 算法。
- Comparer（比较器） 用于比较两个对象之间的差异性，支持 text 和 hex 形式的对比，通常用来比较两个 request 或者 response 数据包之间不同的地方，功能类似于网上常见的文本或者文件对比软件。

以上是 Burp Suite 的几乎所有功能。当然，不同的使用者有不同的需求，很少会用到上面介绍的所有功能。笔者再详细介绍一下常用的几个功能。

1. Proxy 功能

这是使用最多的功能，因为其他的几个常用功能也依赖于代理功能抓到的数据包。

它的使用非常简单，打开 Burp Suite 后，点击菜单栏的 Proxy 即可看到 Proxy 功能界面。首先需要设置监听 IP 和端口，在 Proxy Listeners 区域选中代理项，然后点击左边的 Edit 按钮。它有三种监听模式，如图 2-17 所示。

如果你只需要监听本地的数据，绑定地址的地方设置 Loopback only 即可。如果需要监听到本机的所有 HTTPS/HTTP 流量，则选中 All interfaces。默认监听 8080 端口，在这里可以自行修改，点击 OK 按钮完成设置。

接下来的设置要在需要被代理的客户端完成。这里的设置如图 2-18 所示。

由于这里要监听本地浏览器的数据，所以设置代理服务器地址为 127.0.0.1，端口为 8080，也就是 Burp Suite 中之前设置的监听端口。

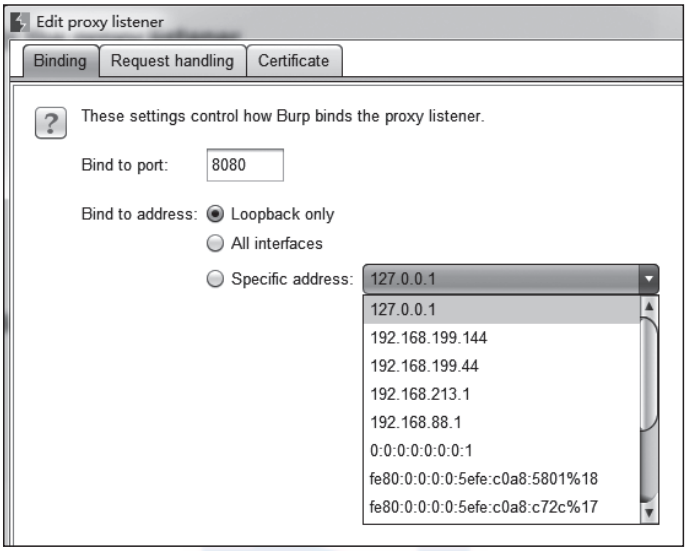


图　2-17

图　2-18

现在就可以开始抓包测试了。如图 2-19 所示，已经可以成功抓到浏览器的 Request 和 Response 数据。

2. Intruder 功能

该功能主要用于模糊测试。在模糊测试分类里，用得最多的是暴力破解登录用户密码，笔者非常喜欢这个功能，因为它有非常强大的兼容性来支持各种数据格式爆破。下面让我们来见识下 Intruder 到底是有多强大。这里演示用它来爆破 Discuz 后台

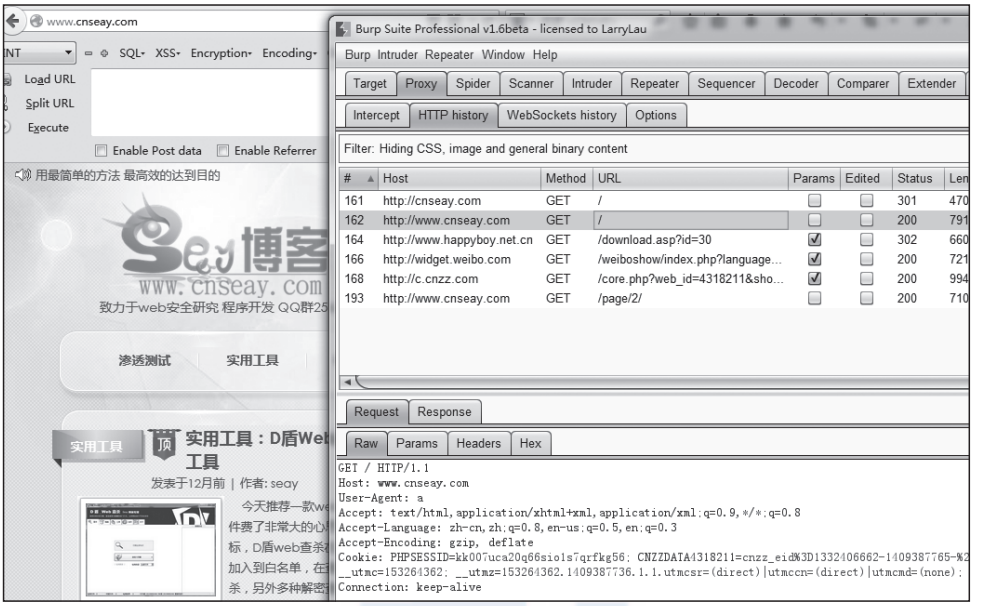


图 2-19

登录密码。注意，Discuz 登录是有登录验证限制的，每个 IP 只有 5 次登录失败的尝试机会。我们利用 Burp Suite 绕过这个登录限制进行密码爆破，绕过原理是，由于 Discuz 采用的是 HTTP_CLIENT_IP 的方式来获取 IP，而这个值可以在发送请求时伪造，于是我们可以利用 Burp Suite 来伪造这个登录 IP 绕过错误次数限制。

首先，按上面的介绍设置好代理抓包，在浏览器中打开 Discuz 后台登录页面，在账号及密码输入框输入任意字符，点击“提交”按钮，回到 Burp Suite，将抓到的数据项发送到 Intruder。接下来，就需要对登录数据包进行修改。

在 Intruder 的 Positions 中需要设置攻击类型为 Pitchfork，并且全选数据包，点击 Clear $ 按钮清除全部标识。然后在 HTTP 头中添加“ client-ip:1.2.3.4”，并且将 1234 这个四个数字单独打上 $ 标识，为 Post 数据里面的 admin_password 字段的值也打上 $ 标识。最后修改效果如图 2-20 所示。

接下来就需要设置 payload 了，点击菜单栏的 Payloads，为 Payload set 下拉框里面的第 1、2、3、4 个选项设置一样的 Numbers payload，包括其他参数设置的最终界面如图 2-21 所示。

设置 Payload set 为 5 的 Payload type 为 simple list，并且点击下面的 Load 按钮载入你的密码字典，然后我们的所有配置就完成了。点击菜单栏的“ Intruder”里的“Start Attack”按钮即可开始爆破。爆破成功的效果如图 2-22 所示。


1 × 2 × ...
Target Positions Payloads Options
Payload Positions
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.
Attack type: Pitchfork
POST /discuz/admin.php? HTTP/1.1
Host: localhost
User-Agent: a
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost/discuz/admin.php
Cookie: tjpctrl=1409394910636; lJb5_2132_saltkey=V74z22QR; lJb5_2132_lastvisit=1409389102;
lJb5_2132_sid=Z6Izvi; lJb5_2132_lastact=1409392833%09admin.php%09; iFsr_2132_sid=HhNh88;
iFsr_2132_saltkey=UYzTZiVV; iFsr_2132_lastvisit=1409389514; iFsr_2132_lastact=1409393119%09admin.php%09;
iFsr_2132_onlineusernum=2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 120
client-ip: §1§.§2§.§3§.§4§
sid=HhNh88&frames=yes&admin_username=seay&admin_password=§seay§&admin_questionid=0&admin_answer=&submit=%E6%8F%90%E4%BA%A4
Add §
Clear §
Auto §
Refresh

图　2-20

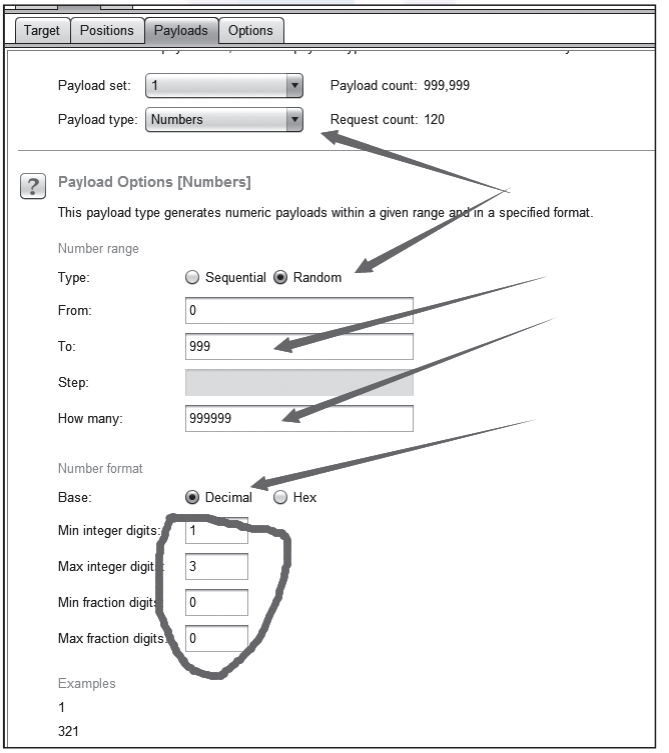

Target Positions Payloads Options
Payload set: 1
Payload count: 999,999
Payload type: Numbers
Request count: 120
Payload Options [Numbers]
This payload type generates numeric payloads within a given range and in a specified format.
Number range
Type: Sequential Random
From: 0
To: 999
Step:
How many: 999999
Number format
Base: Decimal Hex
Min integer digits: 1
Max integer digits: 3
Min fraction digits: 0
Max fraction digits: 0
Examples
1
321

图　2-21

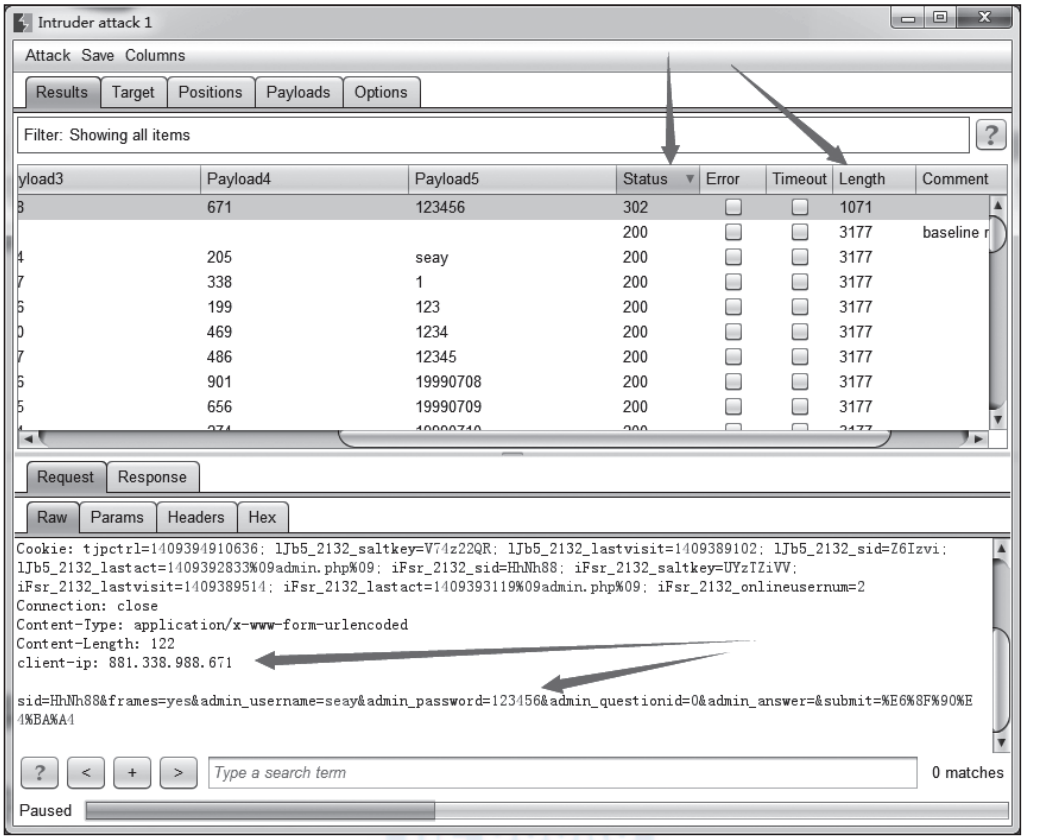


图 2-22

3. Repeater 功能

这个功能经常用于修改数据包以及重放请求，测试漏洞的时候经常需要使用到 Repeater。下面来详细看看它的使用方法。

首先设置好 proxy，然后通过浏览器触发请求查看 Burp Suite 捕捉到的数据包，在“ http history”里选中监听到的数据包项，右键发送到 Repeater。接着我们就可以对请求数据做任意修改了，修改完成后点击 Go 按钮即可发送数据包，如图 2-23 所示。

2.3.2 浏览器扩展

说到浏览器扩展，肯定要先说一下浏览器。通常优先选择的是 Firefox，再就是 Chrome 浏览器，原因很简单，Firefox 的扩展是目前浏览器里面最多的，也许是因为 Firefox 开源的原因，喜欢鼓捣它的人也就多了，自然而然各种插件和扩展也多了。另外不少扩展是专门做安全测试使用的，常用的像 HackBar、FireBug、Live HTTP Headers、

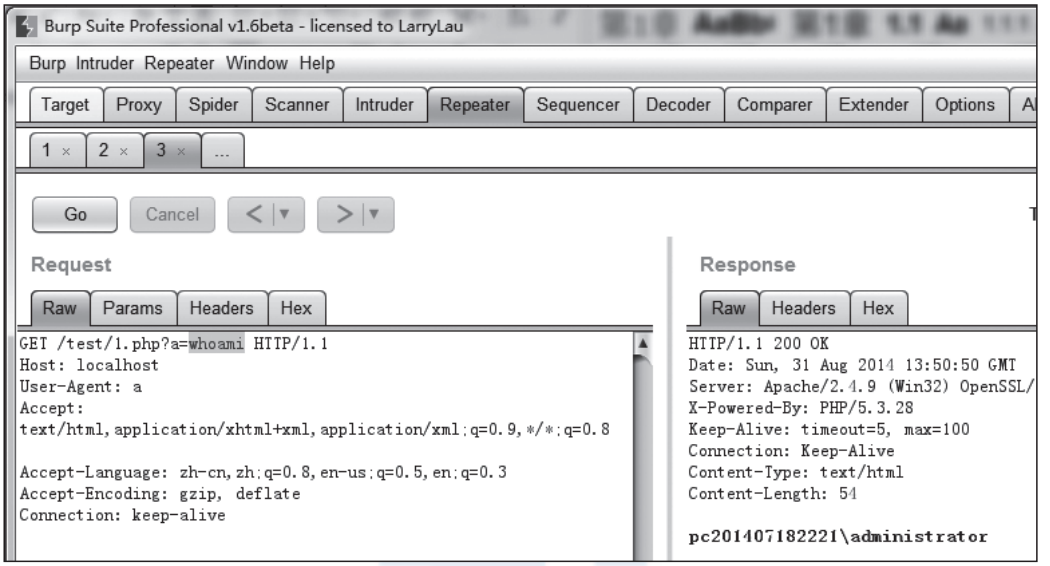


图　2-23

Modify 以及 Tamper Data，等等，稍后会详细介绍这几款扩展。同时，Chrome 浏览器的扩展也非常多，不过方便用来做安全测试的比 Firefox 少，常用的有 Http Headers、EditThisCookie、ModHeader 等。其次就是一些扩展更少的浏览器，这里就不详细列举，不过建议常见内核的浏览器都应该安装一款，笔者电脑上就一直装着 4 款浏览器。

在下面的浏览器扩展介绍里，笔者会着重介绍 Firefox 的扩展。通常不涉及浏览器特性的漏洞测试，在 Firefox 下测试会比较方便；涉及浏览器特性的漏洞测试，则需要安装不同的浏览器。这里推荐一个浏览器测试软件 IEtester，利用它可以切换 IE 浏览器内核版本，而不用安装所有版本的 IE 浏览器。接下来介绍常用的一些扩展的功能和使用方法。

1. HackBar

Hackbar 是安全测试最常用的一款 Firefox 扩展，主要作用是非常方便安全人员对漏洞进行手工测试。它有三个输入框，分别是 URL、Post 数据以及 Referer 的参数设置，在输入框上部还提供了一个菜单栏，有各种各样编码、解码的小功能。Hackbar 的整体界面如图 2-24 所示，箭头所指的标注框里面就是 Hackbar 了。

点击 Load URL 即可从 Firefox 地址栏获取当前 URL，点击 Execute 之后即可发送我们设置好的请求数据。

图 2-24

2. Firebug

Firebug 是一款开发者工具，功能与火狐自带的开发者工具差不多，支持直接对网页 HTML、CSS 等元素进行编辑，其中的“网络”功能可以直接嗅探 Request 和 Response 数据包。通常在利用一些支付漏洞或者 SQL 注入漏洞的时候，我们只需要把鼠标指针定位到要修改的网页区域，右键点击“使用 Firebug 查看元素”即可对网页进行修改测试漏洞，Firebug 的界面图如图 2-25 所示。

图 2-25

3. Live HTTP Headers

Live HTTP Headers 主要的功能是抓取浏览器 Request 和 Response 数据包，也支持对 Request

数据进行修改后再次请求。不好的一点在于它只能抓取到 HTTP 的数据，对 HTTPS 无效，不过用来分析简单页面数据它已经足够。Live HTTP Headers 的界面如图 2-26 所示。

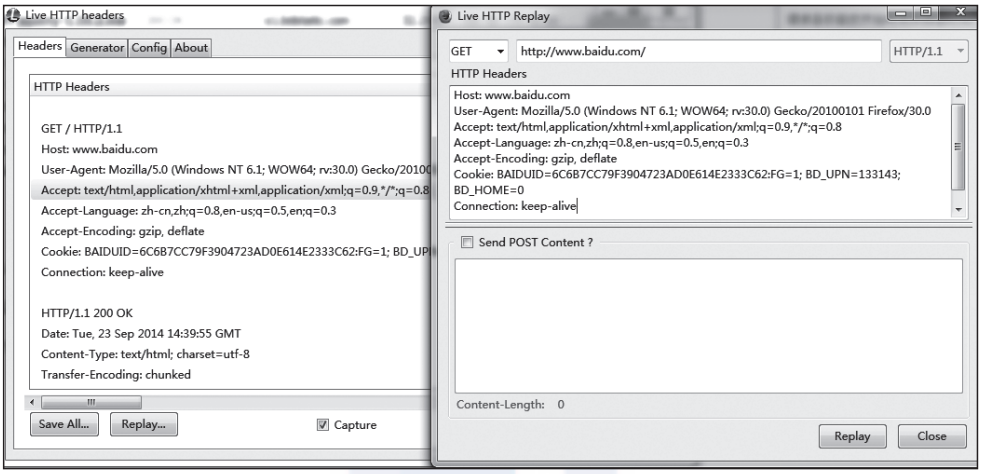


图　2-26

首先勾选 Capture 复选框，然后开始在浏览器中请求页面即可，选中抓到的数据包，点击 Replay 按钮可对数据进行编辑和重新发送。

4. Modify

Modify 是一款火狐扩展工具，顾名思义，这是一款用来修改的扩展，Modify 仅支持添加和修改 Request 中 HTTP Header 的字段，而且它是做全局修改，即开启 Modify 之后，它会把浏览器对任何网站的所有请求中对应字段进行修改。下面就介绍它的使用方法，图 2-27 就是 Modify 的主界面。

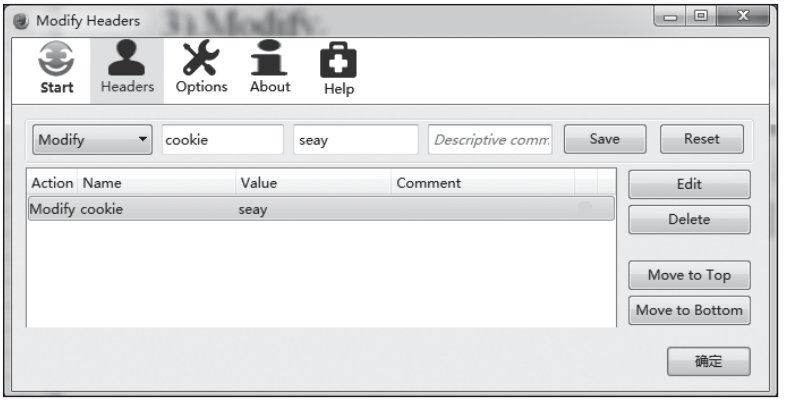


图　2-27

使用非常简单，在图中 Modify 的下拉框中选择要执行的模式，有 Modify、Add 以及 Filter，然后在后面的两个输入框输入参数名以及参数值，点击 Save 再点击左上角的 Start 按钮即可启动 Modify。

通过抓包可以看到以及将访问百度的请求中 cookie 的值修改成了“ seay”，如图 2-28 所示。

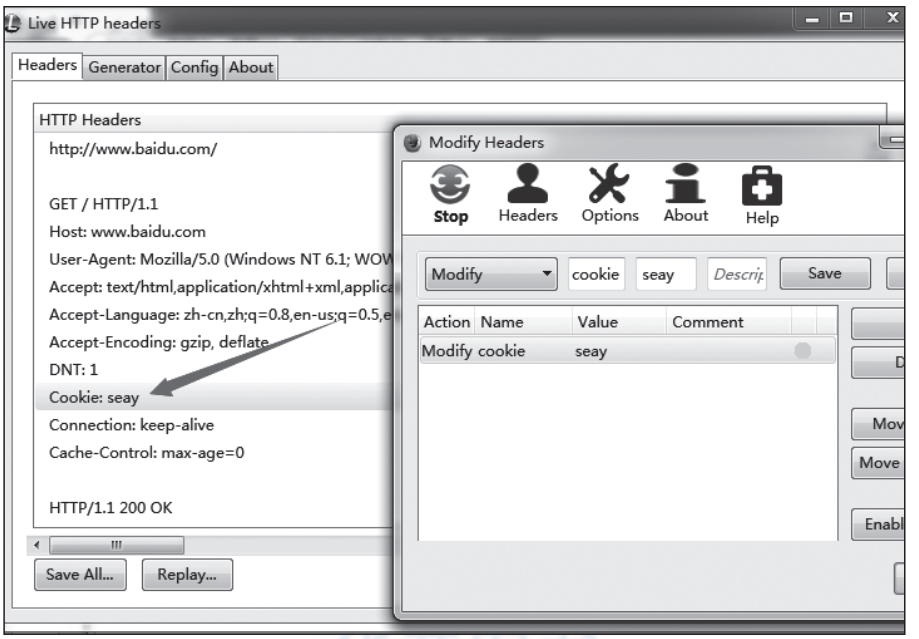


图 2-28

2.3.3 编码转换及加解密工具

代码审计必然要接触到编码相关的知识，历史上很多高危的漏洞是由编码问题导致的，比如在 XSS 漏洞中可以利用浏览器对不同编码的支持来绕过过滤触发漏洞，另外我们也经常需要用到不同的编码转码来进行模糊测试漏洞。另外是加解密以及 Hash 算法，在代码审计中，我们经常遇到程序对特定字符进行加密或者 Hash 后用作 Cookie 和 Session，或者是用户密码的保存通常也会加密，所以我们必须要了解常用的加解密方式，在后面会详细列举常见加解密方式及原理，这里就不详细介绍。下面笔者推荐几款编码转换和加解密的工具。

1. Seay 代码审计系统自带的编码功能

主界面菜单栏点击“正则编码”即可打开该功能，目前支持 Md5 算法、URL、

Base64、Hex、ASCII、Unicode 多种常用编码方式转换。另外还针对 MySQL 与 MSSQL 注入等做利用格式做针对性处理，主界面如图 2-29 所示。

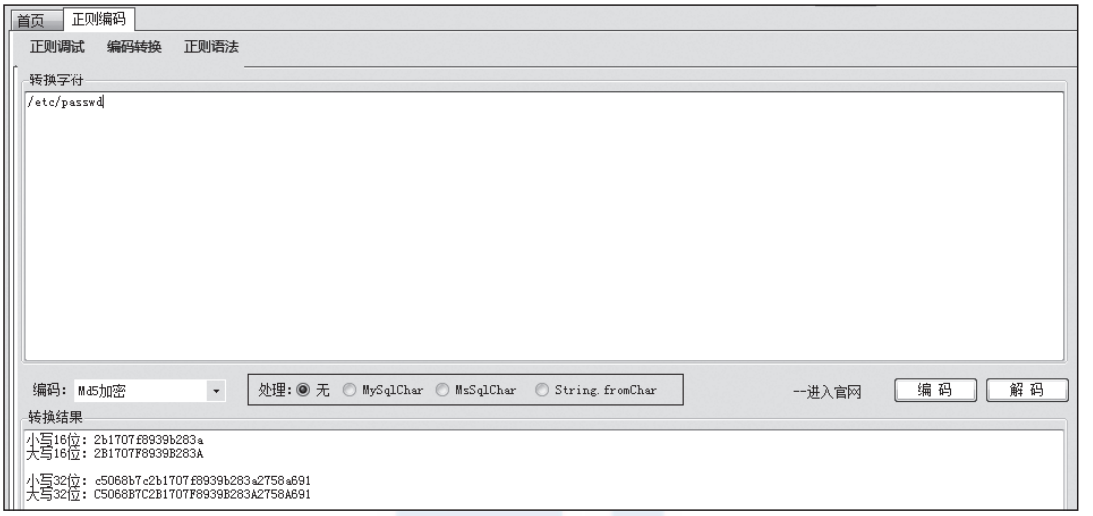


图　2-29

2. Burp Suite 上有一个 Decoder 功能

这个功能可对字符串进行编码和解码，支持百分号、Base64、ASCII 等多种编码转换，还支持 Md2、Md5、Sha 系列等 Hash 算法。它的使用也非常简单，只需要输入要转换要的字符，选择需要转换成的编码即可，如图 2-30 所示。

图　2-30

3. 超级加解密转换工具

另外介绍一个专门用来加解密的国产小工具，支持的算法比较多，独立的 exe 文件轻巧干净，在百度搜索“超级加解密转换工具”即可找到这款小软件，使用界面如图 2-31 所示。

图 2-31

加密：使用的时候只要在软件右侧的“方法分类选择”中选择加密方式，在“明文”输入框中输入需要“加密”的字符串，点击“加密”即可在“密文”框中看到加密后的结果。

解密：使用的时候只要在软件右侧的“方法分类选择”中选择解密方式，在“密文”输入框中输入需要解密的字符串，点击“解密”即可在“明文”框中看到解密后的结果。

2.3.4 正则调试工具

正则表达式是用自定义好的特定字符组合，在正则解析引擎内进行字符匹配。正则表达式有非常强的灵活性，在很多不同的场景都会用到正则表达式，比如验证注册用户名、邮箱等格式是否合格，再比如用来搜索文件内容，很大一部分的 WAF（Web 应用防火墙）的规则也是基于正则表达式，等等，可谓无处不在，然而如果正则表达式写

得不严谨，就经常会导致各种 bug 出现，像防火墙被绕过，等等。

首先要熟悉正则表达式的用法，熟悉各个符号的含义，这样我们才能写出严谨的正则表达式，才能在代码审计中发现别人正则表达式的问题所在。

笔者就不在这里详细的介绍正则表达式的详细用法，网上可以找到很多关于正则表达式的详细教程，这里仅介绍几个常用的正则表达式调试工具。

1. Seay 代码审计系统中自带的正则调试功能

在 Seay 代码审计系统的菜单栏点击“正则编码”项，即可看到正则调试功能的主界面，它支持正则实时预览，即你在输入框修改正则表达式或者要匹配的源字符的时候，调试的结果会实时显示在下方的信息栏中，非常的直观和方便。效果图如图 2-32 所示。

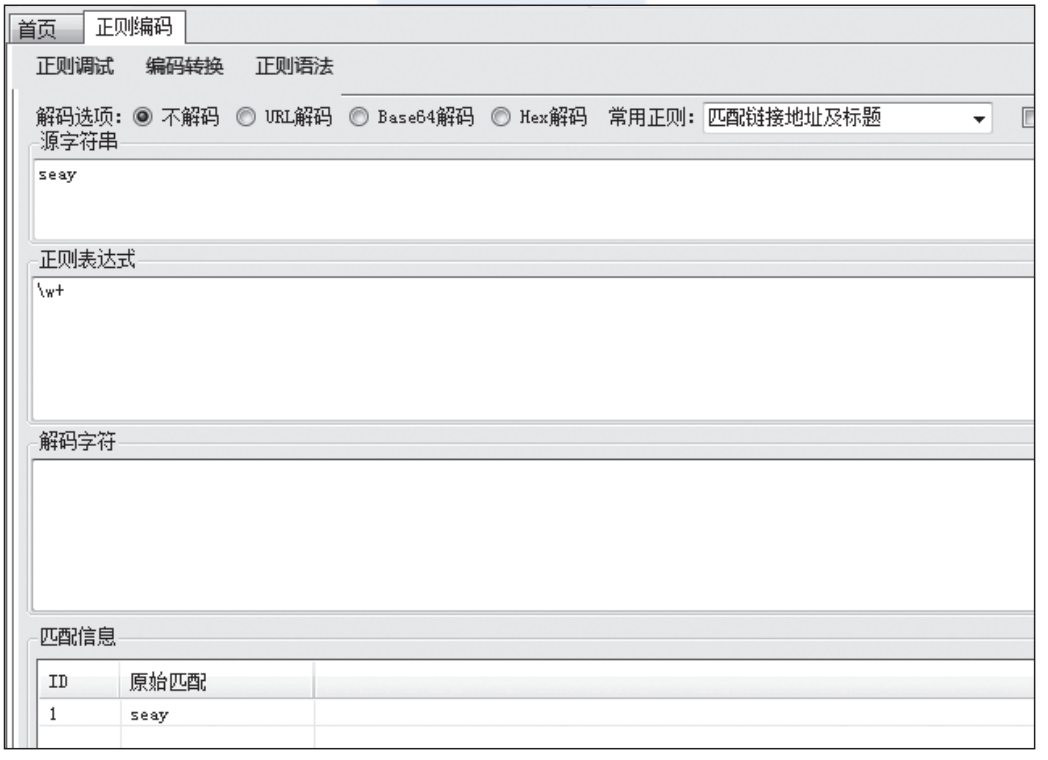


图　2-32

除了实时调试，它还支持实时解码调试，一些特殊字符我们无法在输入框输入，但是可以利用编码处理后输入，只要设置解码选项，程序在用正则匹配前会先将源字符进行指定的编码转换后在进入正则引擎匹配，目前支持 URL 编码、Base64 编码以及

Hex 编码，使用效果如图 2-33 所示。

图 2-33

2. 灵者正则调试

这是一款从 RenGod（灵者更名）中提取出来的绿色版正则调试工具，单文件的大小只有 400 多 K，支持正则搜索、正则替换，也同样支持实时预览，在正则性能上也做的相当不错。如图 2-34 所示是它的主界面。

这款工具使用起来也非常简单，比如正则搜索功能的使用，将要搜索的文本放入到“要搜索的文本”输入框中，在“正则表达式”输入框中输入正则表达式，即可实时的在下方看到匹配上的结果。

2.3.5 SQL 执行监控工具

SQL 执行监控可以非常高效地帮助我们发现一些 SQL 注入和 XSS 等问题，帮助我们非常方便地观察到数据在 Web 程序与数据库中的交互过程，在做模糊测试时，只需利用模拟测试工具爬取页面的 URL 及表单，提交特定的参数如带单引号（‘）等，通

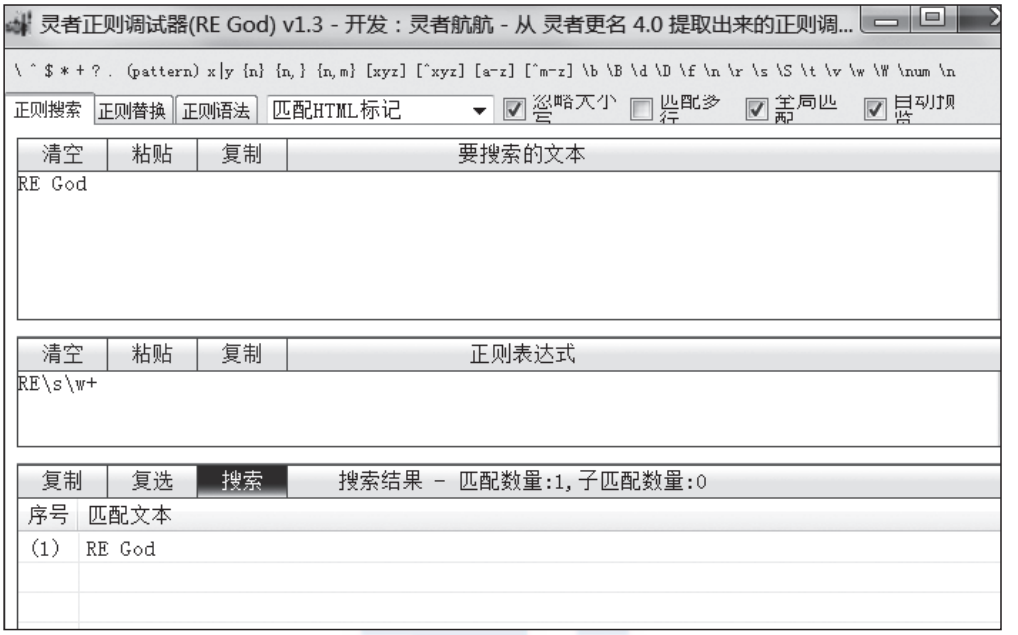


图　2-34

过分析 SQL 执行日志则可以非常准确地判断出 SQL 注入漏洞是否存在，同样的注入 <>”等符合也可以用来测试 XSS 漏洞。

这里分别针对 MySQL 和 MSSQL 列举 SQL 执行监控的工具和使用方法，其中着重介绍 MySQL 的监控，因为 PHP 大多都是使用的 MySQL 作为存储。

针对 MySQL 的执行监控，笔者没有找到比较好的工具，于是在自己的代码审计系统上加入了这么一个插件，主要原理是开启 MySQL 的 general_log 来记录 MySQL 的历史执行语句，它有两种记录方式，默认是通过记录到文件方式，另外一种是通过直接记录到 MySQL 库的 general_log 表中，为了更方便地查询，我选择的是记录到 MySQL 数据库的方式。另外这个功能的开启方式也有两种，一种是直接用 MySQL 的 SQL 语句开启，SQL 语句如下：

```
set global general_log=on;
SET GLOBAL log_output='table';
```

执行结果如图 2-35 所示。

不过这些步骤在笔者的工具中都自动完成了，同时它还支持快速过滤实时预览。只要点击下断，操作完之后点击更新即可看到这段时间内在 MySQL 执行过的所有 SQL 语句，主界面如图 2-36 所示。

```
mysql> show global variables like "%genera%";
+------------------+----------------------------------------+
| Variable_name    | Value                                  |
+------------------+----------------------------------------+
| general_log      | OFF                                    |
| general_log_file | D:\phpStudy\MySQL\data\ali-081518n.log |
+------------------+----------------------------------------+
2 rows in set (0.00 sec)

mysql> set global general_log=on;
Query OK, 0 rows affected (0.03 sec)

mysql> SET GLOBAL log_output='table';
Query OK, 0 rows affected (0.00 sec)
```

图 2-35

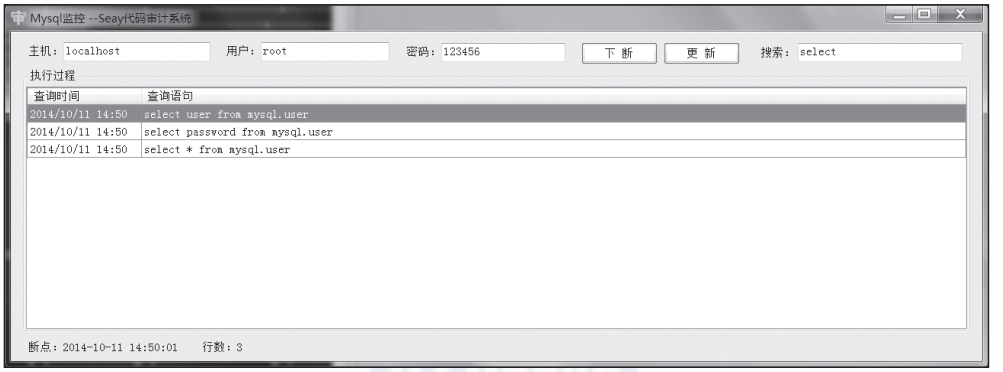


图 2-36

另外一种开启方法是在 MySQL 配置文件中修改，在 [mysqld] 配置中加入如下代码：

```
general_log=ON
general_log_file={日志路径}/query.log
```

重启 MySQL 后可以看到所有的 SQL 查询语句都会记录在设置的这个文件中。

SQL Server 执行监控也很简单，在 SQL Server 上自带有一个性能监控的工具 SQL Server Profiler，在开始菜单里可以找到它，使用 SQL Server Profiler 可以将 SQL 执行过程保存到文件和数据库表，同时它还支持实时查看和搜索。

下面我们来看看怎么使用它，打开 SQL Server Profiler 后，在左上角的菜单栏里选择“文件→新建跟踪”，在常规栏输入跟踪名（随意）后，点击“事件选择”标签，我们只需要 SQL 执行过程，所以要过滤掉一些干扰的东西，比如登录、退出等，在事件选择里只保留 TSQL 下面的 SQL:BatchCompleted 事件，然后点击“运行”，如图 2-37 所示。

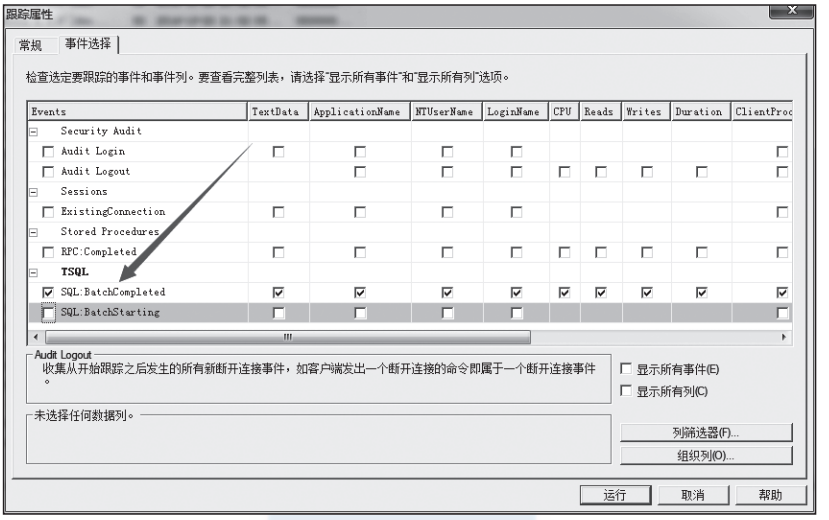


图　2-37

运行后监控的到 SQL 语句如图 2-38 所示。

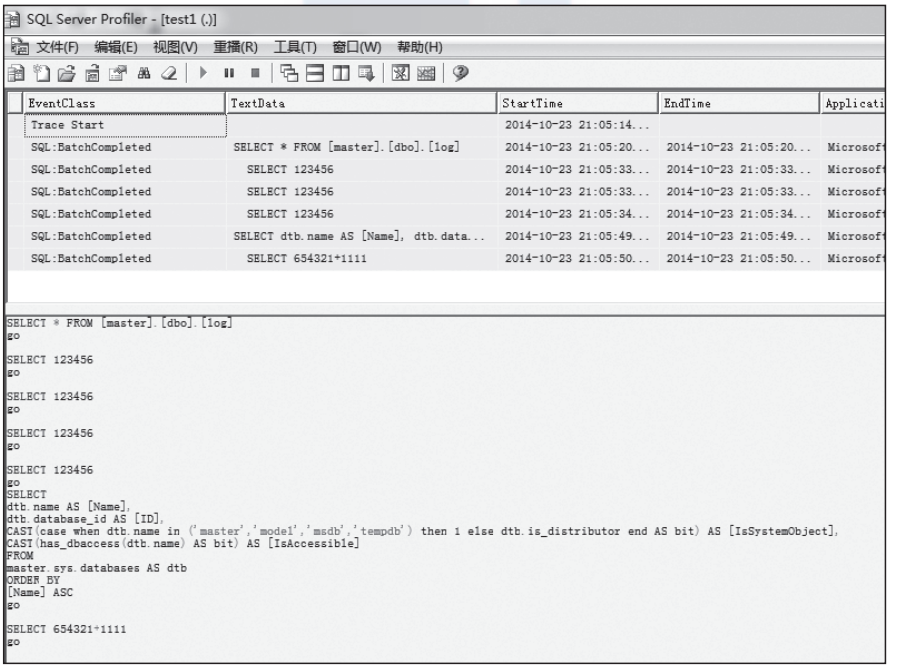


图　2-38

从图中监控结果可以非常清楚第看到之前执行的 SQL 语句以及开始执行时间、结束时间。

HZ BOOKS
华章图书

第二部分 Part 2

漏洞发现与防范

第二部分包括第 3 ～ 8 章一共六章，将着重介绍 PHP 代码审计的中漏洞挖掘思路与防范方法。

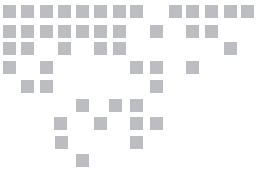
其中第 3 章详细介绍 PHP 代码审计的思路，包括根据关键字回溯参数、通读全文代码以及根据功能点定向挖掘漏洞的三个思路。

第 4 ～ 6 章介绍常见漏洞的审计方法，共分为基础篇、进阶篇以及深入篇，涵盖 SQL 注入漏洞、XSS 漏洞、文件操作漏洞、代码 / 命令执行漏洞、变量覆盖漏洞以及逻辑处理等漏洞。

第 7 章介绍二次漏洞的挖掘方法，二次漏洞在逻辑上比常规漏洞要复杂，所以我们需要单独拿出来以实例来进行介绍。

在经过前面几章的代码审计方法学习之后，相信大家已经能够挖掘不少有意思的漏洞，在第 8 章，将会介绍更多代码审计中的小技巧，利用这些小技巧可以挖掘到更多有意思的漏洞。

每类漏洞都有多个真实漏洞案例的分析过程，可以真正帮助大家学习代码审计的经验，不过这些内容不仅仅是介绍了漏洞的挖掘方法，还详细介绍了这些漏洞的修复方法，对开发者来说是非常有用的一部分内容。



Chapter 3 第 3 章

通用代码审计思路

代码审计工具的实现都是基于代码审计经验开发出来的优化工作效率的工具，我们要学好代码审计就必须要熟悉代码审计的思路，只有在了解了这些思路之后，我们才知道如何下手，如何最高效地挖掘到最有质量的漏洞，常见的代码审计思路有以下四种：

1）根据敏感关键字回溯参数传递过程。

2）查找可控变量，正向追踪变量传递过程。

3）寻找敏感功能点，通读功能点代码。

4）直接通读全文代码。

下面我们就来看看这几种代码审计思路在实际场景中的应用。

3.1 敏感函数回溯参数过程

根据敏感函数来逆向追踪参数的传递过程，是目前使用得最多的一种方式，因为大多数漏洞是由于函数的使用不当造成的。另外非函数使用不当的漏洞，如 SQL 注入，也有一些特征，比如 Select、Insert 等，再结合 From 和 Where 等关键字，我们就可以判断这是否是一条 SQL 语句，通过对字符串的识别分析，就能判断这个 SQL 语句里面的参数有没有使用单引号过滤，或者根据我们的经验来判断。像 HTTP 头里面的 HTTP_CLIENT_IP 和 HTTP_X_FORWORDFOR 等获取到的 IP 地址经常没有安全过

滤就直接拼接到 SQL 语句中，并且由于它们是在 $_SERVER 变量中不受 GPC 的影响，那我们就可以去查找 HTTP_CLIENT_IP 和 HTTP_X_FORWORDFOR 关键字来快速寻找漏洞。

这种方式的优点是只需搜索相应敏感关键字，即可以快速地挖掘想要的漏洞，具有可定向挖掘和高效、高质量的优点。

其缺点为由于没有通读代码，对程序的整体框架了解不够深入，在挖掘漏洞时定位利用点会花费一点时间，另外对逻辑漏洞挖掘覆盖不到。

espcms 注入挖掘案例

我们来举一个根据关键字回溯的例子，用 PHP 程序 espcms 举例，使用 Seay 源代码审计系统做演示，首先载入程序，然后点击自动审计，得到一部分可能存在漏洞的代码列表，如图 3-1 所示。

图　3-1

我们挑其中的一条代码，如图 3-2 所示。

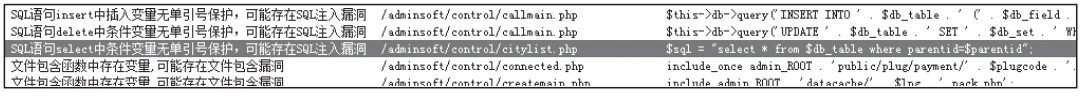


图　3-2

双击该项直接定位到这行代码，如图 3-3 所示。

在选中该变量后，在下方可以看到该变量的传递过程，并且点击下方的变量传递过程也可以直接跳转到该项代码处，可以非常直观地帮助我们看清整个变量在该文件的传递过程。另外我们可以看到 $parentid 变量是在如下代码段获得的：

```
$parentid = $this->fun->accept('parentid', 'R');
```

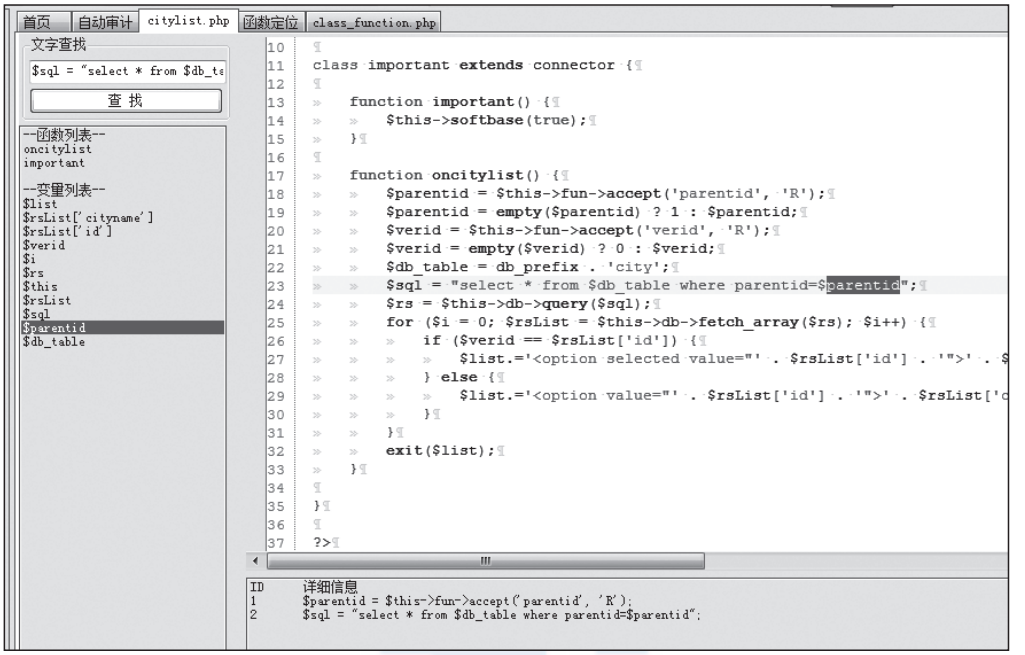


图 3-3

右键选中该代码定位该函数主体，如图 3-4 所示。

图 3-4

可以看到跳转到了 class_function.php 文件的 314 行，代码如下：

```
function accept($k, $var = 'R', $htmlcode = true, $rehtml = false) {
   switch ($var) {
     case 'G':
```

```
            $var = &$_GET;
            break;
        case 'P':
            $var = &$_POST;
            break;
        case 'C':
            $var = &$_COOKIE;
            break;
        case 'R':
            $var = &$_GET;
            if (empty($var[$k])) {
                $var = &$_POST;
            }
            break;
    }
    $putvalue = isset($var[$k])? $this->daddslashes($var[$k], 0) : NULL;
    return $htmlcode ?($rehtml?$this->preg_htmldecode($putvalue):$this->
        htmldecode($putvalue)): $putvalue;
}
```

可以看到这是一个获取 GET、POST、COOKIE 参数值的函数，我们传入的变量是 parentid 和 R，则代表在 POST、GET 中都可以获取 parentid 参数，最后经过了一个 daddslashes() 函数，实际上是包装的 addslashes() 函数，对单引号等字符进行过滤，不过注意看前面的 SQL 语句是这样的：

```
$sql = "select * from $db_table where parentid=$parentid";
```

并不需要单引号来闭合，于是可以直接注入。

在 citylist.php 文件看到 oncitylist() 函数在 important 类中，选中该类名右键点击“全局搜索”功能，如图 3-5 所示。

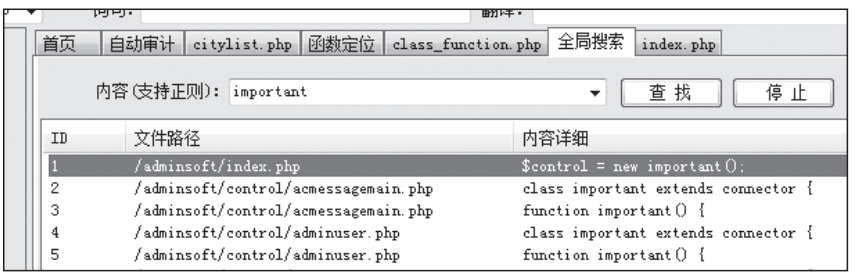


图　3-5

可以看到 index.php 文件有实例化该类，代码如下：

```
$archive = indexget('archive', 'R');
$archive = empty($archive) ? 'adminuser' : $archive;
$action = indexget('action', 'R');
$action = empty($action) ? 'login' : $action;
include admin_ROOT . adminfile . "/control/$archive.php";
$control = new important();
$action = 'on' . $action;
if (method_exists($control, $action)) {
   $control->$action();
} else {
   exit('错误：系统方法错误！');
}
```

这里可以看到一个 include 文件的操作，可惜经过了 addslashes() 函数无法进行截断使其包含任意文件，只能包含本地的 PHP 文件，如果你有 MySQL 的 root 权限，能导出文件到 tmp 目录，不能导出到 Web 目录，这种场景才用得到这个文件包含，再往下走就是实例化类并且调用函数的操作了，根据代码可以构造出利用 EXP：

```
http://127.0.0.1/espcms/adminsoft/index.php?archive=citylist&action=cit
    ylist&parentid=-1 union select 1,2,user(),4,5
```

漏洞截图如图 3-6 所示。

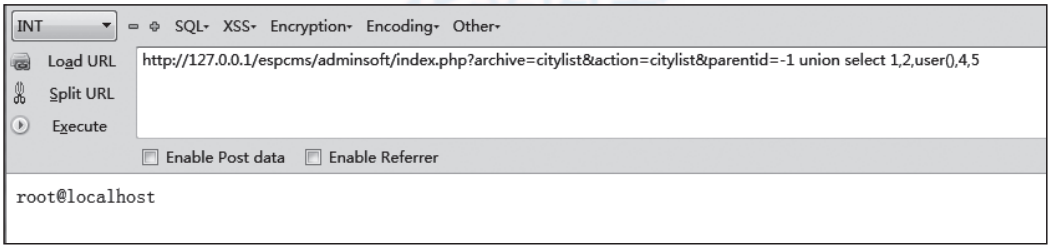


图 3-6

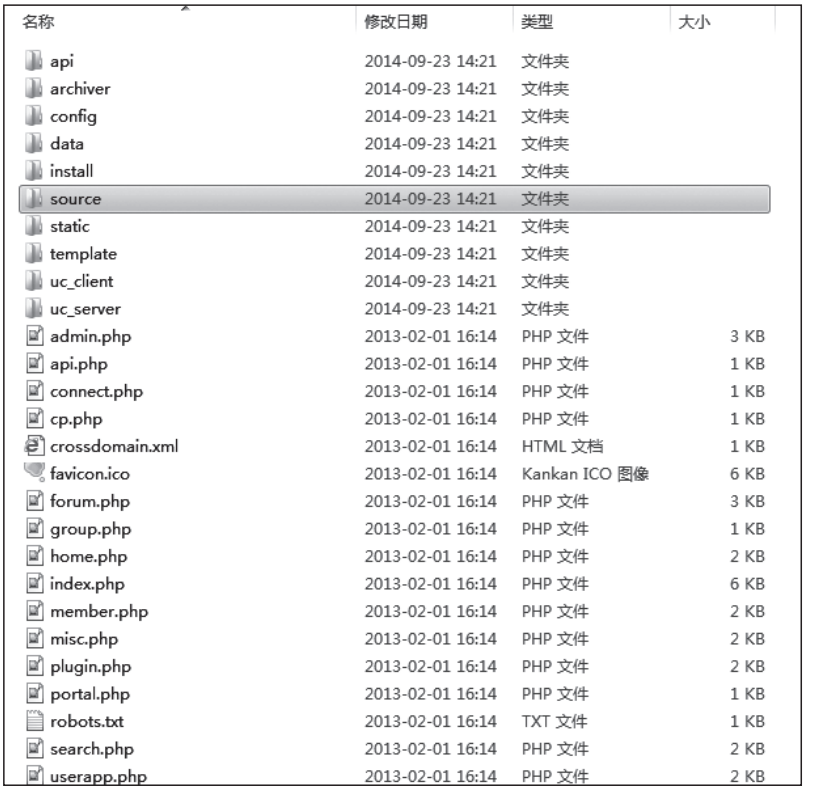
这个案例非常真实地演示了如何根据关键字回溯变量来进行代码审计。

3.2 通读全文代码

前面提到的根据敏感关键字来回溯传入的参数，是一种逆向追踪的思路，我们也提到了这种方式的优缺点，实际上在需要快速寻找漏洞的情况下用回溯参数的方式是非

常有效的，但这种方式并不适合运用在企业中做安全运营时的场景，在企业中做自身产品的代码审计时，我们需要了解整个应用的业务逻辑，才能挖掘到更多更有价值的漏洞。

通读全文代码也有一定的技巧，并不是随便找文件逐个读完就可以了，这样你是很难真正读懂这套 Web 程序的，也很难理解代码的业务逻辑，首先我们要看程序的大体代码结构，如主目录有哪些文件，模块目录有哪些文件，插件目录有哪些文件，除了关注有哪些文件，还要注意文件的大小、创建时间。我们根据这些文件的命名就可以大致知道这个程序实现了哪些功能，核心文件是哪些，discuz 的程序主目录如图 3-7 所示。



名称	修改日期	类型	大小
api	2014-09-23 14:21	文件夹	
archiver	2014-09-23 14:21	文件夹	
config	2014-09-23 14:21	文件夹	
data	2014-09-23 14:21	文件夹	
install	2014-09-23 14:21	文件夹	
source	2014-09-23 14:21	文件夹	
static	2014-09-23 14:21	文件夹	
template	2014-09-23 14:21	文件夹	
uc_client	2014-09-23 14:21	文件夹	
uc_server	2014-09-23 14:21	文件夹	
admin.php	2013-02-01 16:14	PHP 文件	3 KB
api.php	2013-02-01 16:14	PHP 文件	1 KB
connect.php	2013-02-01 16:14	PHP 文件	1 KB
cp.php	2013-02-01 16:14	PHP 文件	1 KB
crossdomain.xml	2013-02-01 16:14	HTML 文档	1 KB
favicon.ico	2013-02-01 16:14	Kankan ICO 图像	6 KB
forum.php	2013-02-01 16:14	PHP 文件	3 KB
group.php	2013-02-01 16:14	PHP 文件	1 KB
home.php	2013-02-01 16:14	PHP 文件	2 KB
index.php	2013-02-01 16:14	PHP 文件	6 KB
member.php	2013-02-01 16:14	PHP 文件	2 KB
misc.php	2013-02-01 16:14	PHP 文件	2 KB
plugin.php	2013-02-01 16:14	PHP 文件	2 KB
portal.php	2013-02-01 16:14	PHP 文件	1 KB
robots.txt	2013-02-01 16:14	TXT 文件	1 KB
search.php	2013-02-01 16:14	PHP 文件	2 KB
userapp.php	2013-02-01 16:14	PHP 文件	2 KB

图　3-7

在看程序目录结构的时候，我们要特别注意以下几个文件：

1）函数集文件，通常命名中包含 functions 或者 common 等关键字，这些文件里面是一些公共的函数，提供给其他文件统一调用，所以大多数文件都会在文件头部包含

到其他文件。寻找这些文件一个非常好用的技巧就是去打开 index.php 或者一些功能性文件，在头部一般都能找到。

2）配置文件，通常命名中包括 config 关键字，配置文件包括 Web 程序运行必须的功能性配置选项以及数据库等配置信息。从这个文件中可以了解程序的小部分功能，另外看这个文件的时候注意观察配置文件中参数值是用单引号还是用双引号括起来，如果是双引号，则很可能会存在代码执行漏洞，例如下面 Kuwebs 的代码，只要我们在修改配置的时候利用 PHP 可变变量的特性即可执行代码。

```
<?php
/*网站基本信息配置*/
$kuWebsiteURL       = "http://www.kuwebs.com";
$kuWebsiteSupportEn         = "1";
$kuWebsiteSupportSimplifiedOrTraditional       = "0";
$kuWebsiteDefauleIndexLanguage                 = "cn";
$kuWebsiteUploadFileMax                        = "2";
$kuWebsiteAllowUploadFileFormat   = "swf|rar|jpg|zip|gif";

/*邮件设置*/
$kuWebsiteMailType         = "1";
$kuWebsiteMailSmtpHost              = "smtp.qq.com";
```

3）安全过滤文件，安全过滤文件对我们做代码审计至关重要，关系到我们挖掘到的可疑点能不能利用，通常命名中有 filter、safe、check 等关键字，这类文件主要是对参数进行过滤，比较常见的是针对 SQL 注入和 XSS 过滤，还有文件路径、执行的系统命令的参数，其他的则相对少见。而目前大多数应用都会在程序的入口循环对所有参数使用 addslashes() 函数进行过滤。

```
private static function _do_query_safe($sql) {
   $sql = str_replace(array('\\\\', '\\\'', '\\"', '\'\''), '', $sql);
   $mark = $clean = '';
   if (strpos($sql, '/') === false && strpos($sql, '#') === false &&
       strpos($sql, '--') === false && strpos($sql, '@') === false &&
       strpos($sql, '`') === false) {
          $clean = preg_replace("/'(.+?)'/s", '', $sql);
   } else {
```

4）index 文件，index 是一个程序的入口文件，所以通常我们只要读一遍 index 文件就可以大致了解整个程序的架构、运行的流程、包含到的文件，其中核心的文件又

有哪些。而不同目录的 index 文件也有不同的实现方式，建议最好先将几个核心目录的 index 文件都简单读一遍。

上面介绍了我们应该注意的部分文件，可以帮助我们更有方向地去读全部的代码，实际上在我们真正做代码审计的时候，经常会遇到各种框架，这时候就会被搞得晕头转向，所以在学习代码审计的前期建议不要去读开源框架或者使用开源框架的应用，先去 chinaz、admin5 之类的源码下载网站下载一些小应用来读，并且一定要多找几套程序通读全文代码，这样我们才能总结经验，等总结了一定的经验，对 PHP 也比较熟悉的时候，再去读一些像 thinkphp、Yii、Zend Framework 等开源框架，才能快速地挖掘高质量的漏洞。

通读全文代码的好处显而易见，可以更好地了解程序的架构以及业务逻辑，能够挖掘到更多、更高质量的逻辑漏洞，一般老手会比较喜欢这种方式。而缺点就是花费的时间比较多，如果程序比较大，读起来也会比较累。

骑士 cms 通读审计案例

我们已经介绍了代码审计中通读全文代码审计方式的思路，下面我们用案例来说明这种通读方式。

为了方便大家理解，笔者找了一款相对简单容易看懂的应用骑士 cms 来介绍，版本是 3.5.1，具体的审计思路我们在上文中已经有过介绍。

3.2.1.1　查看应用文件结构

首先来看一下骑士 cms 的大致文件目录结构，如图 3-8 所示。

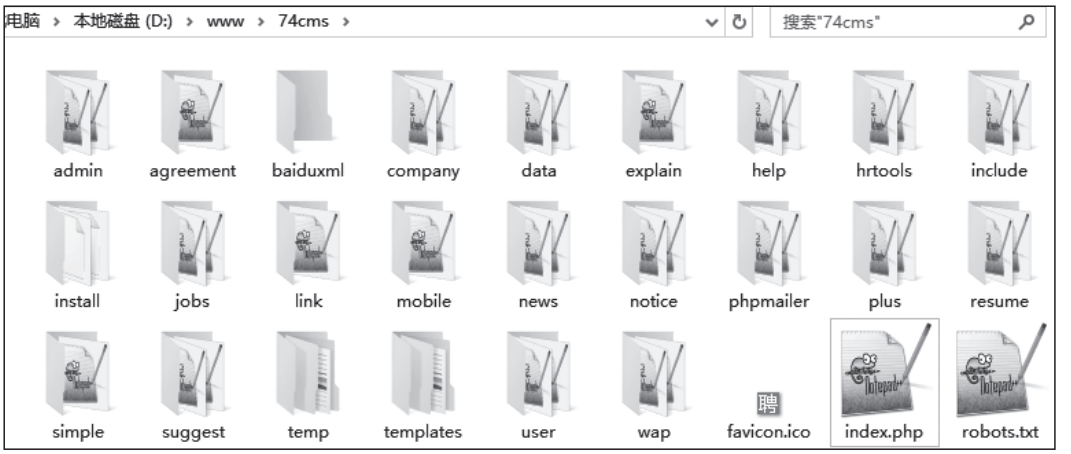


图　3-8

首先需要看看有哪些文件和文件夹，寻找名称里有没有带有 api、admin、manage、include 一类关键字的文件和文件夹，通常这些文件比较重要，在这个程序里，可以看到并没有什么 PHP 文件，就一个 index.php，看到有一个名为 include 的文件夹，一般比较核心的文件都会放在这个文件夹中，我们先来看看大概有哪些文件，如图 3-9 所示。

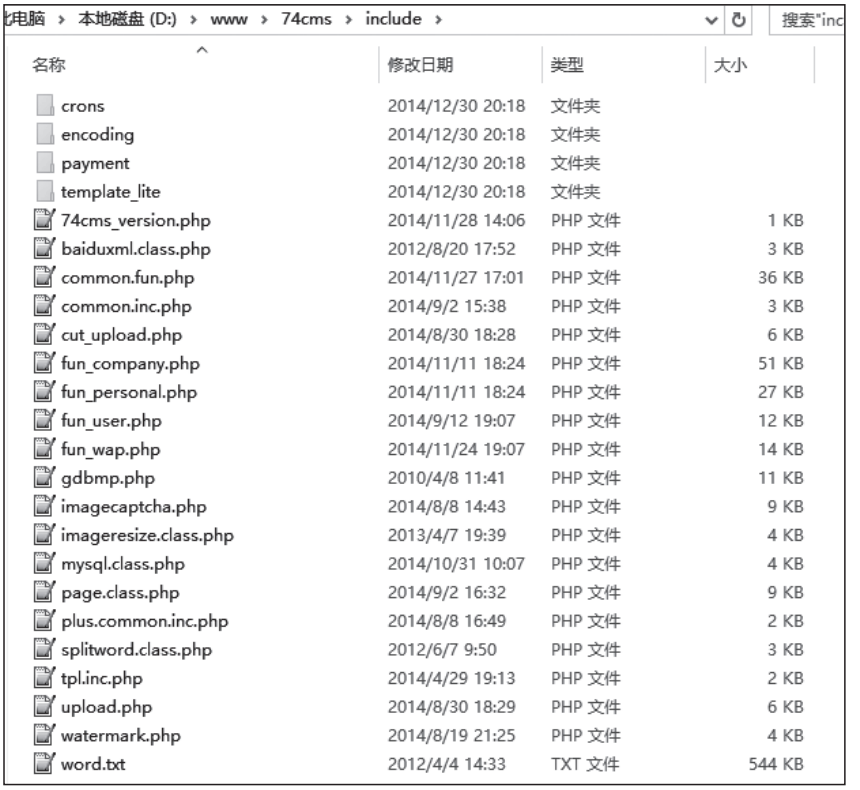


图 3-9

3.2.1.2 查看关键文件代码

在这个文件夹里面我们看到了多个数十 K 的 PHP 文件，比如 common.fun.php 就是本程序的核心文件，基础函数基本在这个文件中实现，我们来看看这个文件里有哪些关键函数，一打开这个文件，立马就看到一大堆过滤函数，这是我们最应该关心的地方，首先是一个 SQL 注入过滤函数：

```
function addslashes_deep($value)
{
    if (empty($value))
```

```
    {
        return $value;
    }
    else
    {
        if (!get_magic_quotes_gpc())
        {
        $value=is_array($value)?array_map('addslashes_deep', $value):
            mystrip_ tags(addslashes($value));
        }
        else
        {
        $value=is_array($value)?array_map('addslashes_deep', $value):
            mystrip_tags($value);
        }
        return $value;
    }
}
```

该函数将传入的变量使用 addslashes( ) 函数进行过滤，也就过滤掉了单引号、双引号、NULL 字符以及斜杠，现在我们要记住，在挖掘 SQL 注入等漏洞时，只要参数在拼接到 SQL 语句前，除非有宽字节注入或者其他特殊情况，否则使用了这个函数就不能注入了。

再往下走是一个 XSS 过滤的函数 mystrip_tags()，代码如下：

```
function mystrip_tags($string)
{
   $string = new_html_special_chars($string);
   $string = remove_xss($string);
   return $string;
}
```

这个函数调用了 new_html_special_chars() 和 remove_xss() 函数来过滤 XSS，就在该函数下方，代码如下：

```
function new_html_special_chars($string) {
   $string = str_replace(array('&amp;', '&quot;', '&lt;', '&gt;'),
       array('&', '"', '<', '>'), $string);
   $string = strip_tags($string);
   return $string;
```

```
}
function remove_xss($string) {
    $string = preg_replace('/[\x00-\x08\x0B\x0C\x0E-\x1F\x7F]+/S', '',
        $string);

    $parm1 = Array('javascript', 'union','vbscript', 'expression',
        'applet', 'xml', 'blink', 'link', 'script', 'embed', 'object',
        'iframe', 'frame', 'frameset', 'ilayer', 'layer', 'bgsound',
        'title', 'base');

    $parm2 = Array('onabort', 'onactivate', 'onafterprint',
        'onafterupdate', 'onbeforeactivate', 'onbeforecopy',
        'onbeforecut', 'onbeforedeactivate', 'onbeforeeditfocus',
        'onbeforepaste', 'onbeforeprint', 'onbeforeunload',
        'onbeforeupdate', 'onblur', 'onbounce', 'oncellchange',
        'onchange', 'onclick', 'oncontextmenu', 'oncontrolselect',
        'oncopy', 'oncut', 'ondataavailable', 'ondatasetchanged',
        'ondatasetcomplete', 'ondblclick', 'ondeactivate', 'ondrag',
        'ondragend', 'ondragenter', 'ondragleave', 'ondragover',
        'ondragstart', 'ondrop', 'onerror', 'onerrorupdate',
        'onfilterchange', 'onfinish', 'onfocus', 'onfocusin',
        'onfocusout', 'onhelp', 'onkeydown', 'onkeypress', 'onkeyup',
        'onlayoutcomplete', 'onload', 'onlosecapture', 'onmousedown',
        'onmouseenter', 'onmouseleave', 'onmousemove', 'onmouseout',
        'onmouseover', 'onmouseup', 'onmousewheel', 'onmove',
        'onmoveend', 'onmovestart', 'onpaste', 'onpropertychange',
        'onreadystatechange', 'onreset', 'onresize', 'onresizeend',
        'onresizestart', 'onrowenter', 'onrowexit', 'onrowsdelete',
        'onrowsinserted', 'onscroll', 'onselect', 'onselectionchange',
        'onselectstart', 'onstart', 'onstop', 'onsubmit', 'onunload',
        'style','href','action','location','background','src','poster');

    $parm3= Array('alert','sleep','load_file','confirm','prompt','bench-
        mark', 'select','update','insert','delete','alter','drop','tru
        ncate','script','eval');

    $parm = array_merge($parm1, $parm2, $parm3);

   for ($i = 0; $i < sizeof($parm); $i++) {
       $pattern = '/';
```

```
            for ($j = 0; $j < strlen($parm[$i]); $j++) {
                if ($j > 0) {
                    $pattern .= '(';
                    $pattern .= '(&#[x|X]0([9][a][b]);?)?';
                    $pattern .= '|(&#0([9][10][13]);?)?';
                    $pattern .= ')?';
                }
                $pattern .= $parm[$i][$j];
            }
            $pattern .= '/i';
            $string = preg_replace($pattern, '****', $string);
        }
        return $string;
    }
```

在 new_html_special_chars() 函数中可以看到，这个函数对 & 符号、双引号以及尖括号进行了 html 实体编码，并且使用 strip_tags() 函数进行了二次过滤。而 remove_xss() 函数则是对一些标签关键字、事件关键字以及敏感函数关键字进行了替换。

再往下走有一个获取 IP 地址的函数 getip()，是可以伪造 IP 地址的：

```
function getip()
{
    if (getenv('HTTP_CLIENT_IP') and strcasecmp(getenv('HTTP_CLIENT_IP'),
        'unknown')) {
        $onlineip=getenv('HTTP_CLIENT_IP');
    }elseif (getenv('HTTP_X_FORWARDED_FOR') and strcasecmp(getenv('HTTP_
        X_FORWARDED_FOR'),'unknown')) {
        $onlineip=getenv('HTTP_X_FORWARDED_FOR');
    }elseif (getenv('REMOTE_ADDR') and strcasecmp(getenv('REMOTE_
        ADDR'),'unknown')) {
        $onlineip=getenv('REMOTE_ADDR');
    }elseif (isset($_SERVER['REMOTE_ADDR']) and $_SERVER['REMOTE_ADDR']
        and strcasecmp($_SERVER['REMOTE_ADDR'],'unknown')) {
        $onlineip=$_SERVER['REMOTE_ADDR'];
    }
    preg_match("/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/",$onlineip,$match);
    return $onlineip = $match[0] ? $match[0] : 'unknown';
}
```

很多应用都会由于在获取 IP 时没有验证 IP 格式，而存在注入漏洞，不过这里还只

是可以伪造 IP。

再往下看可以看到一个值得关注的地方，SQL 查询统一操作函数 inserttable() 以及 updatetable() 函数，大多数 SQL 语句执行都会经过这里，所以我们要关注这个地方是否还有过滤等问题。

```
function inserttable($tablename, $insertsqlarr, $returnid=0, $replace = false,
    $silent=0)

{
    global $db;
    $insertkeysql = $insertvaluesql = $comma = '';
    foreach ($insertsqlarr as $insert_key => $insert_value) {
        $insertkeysql .= $comma.'`'.$insert_key.'`';
        $insertvaluesql .= $comma.'\''.$insert_value.'\'';
        $comma = ', ';
    }
    $method = $replace?'REPLACE':'INSERT';
    // echo $method." INTO $tablename ($insertkeysql) VALUES
        ($insertvaluesql)", $silent?'SILENT':'';die;
    $state = $db->query($method." INTO $tablename ($insertkeysql) VALUES
        ($insertvaluesql)", $silent?'SILENT':'');
    if($returnid && !$replace) {
        return $db->insert_id();
    }else {
        return $state;
    }
}
```

再往下走则是 wheresql() 函数，是 SQL 语句查询的 Where 条件拼接的地方，我们可以看到参数都使用了单引号进行包裹，代码如下：

```
function wheresql($wherearr='')
{
   $wheresql="";
   if (is_array($wherearr))
       {
       $where_set=' WHERE ';
           foreach ($wherearr as $key => $value)
           {
           $wheresql .=$where_set. $comma.$key.'="'.$value.'"';
```

```
            $comma = ' AND ';
            $where_set=' ';
            }
        }
    return $wheresql;
}
```

还有一个访问令牌生成的函数 asyn_userkey()，拼接用户名、密码 salt 以及密码进行一次 md5，访问的时候只要在 GET 参数 key 的值里面加上生成的这个 key 即可验证是否有权限，被用在注册、找回密码等验证过程中，也就是我们能看到的找回密码链接里面的 key，代码如下：

```
function asyn_userkey($uid)
{
    global $db;
    $sql = "select * from ".table('members')." where uid = '".intval($uid)."'
        LIMIT 1";
    $user=$db->getone($sql);
    return md5($user['username'].$user['pwd_hash'].$user['password']);
}
```

同目录下的文件如图 3-10 所示。

fun_company.php	2014/11/11 18:24	PHP 文件	51 KB
fun_personal.php	2014/11/11 18:24	PHP 文件	27 KB
fun_user.php	2014/9/12 19:07	PHP 文件	12 KB
fun_wap.php	2014/11/24 19:07	PHP 文件	14 KB

图　3-10

图中是具体功能的实现代码，我们这时候还不需要看，先了解下程序的其他结构。

3.2.1.3　查看配置文件

接下来我们找找配置文件，上面我们介绍到配置文件的文件名通常都带有“config”这样的关键字，我们只要搜索带有这个关键字的文件名即可，如图 3-11 所示。

在搜索结果中我们可以看到搜索出来多个文件，结合文件所在目录这个经验可以判断出 data 目录下面的 config.php 以及 cache_config.php 才是真正的配置文件，打开 /data/config.php 查看代码，如下所示：

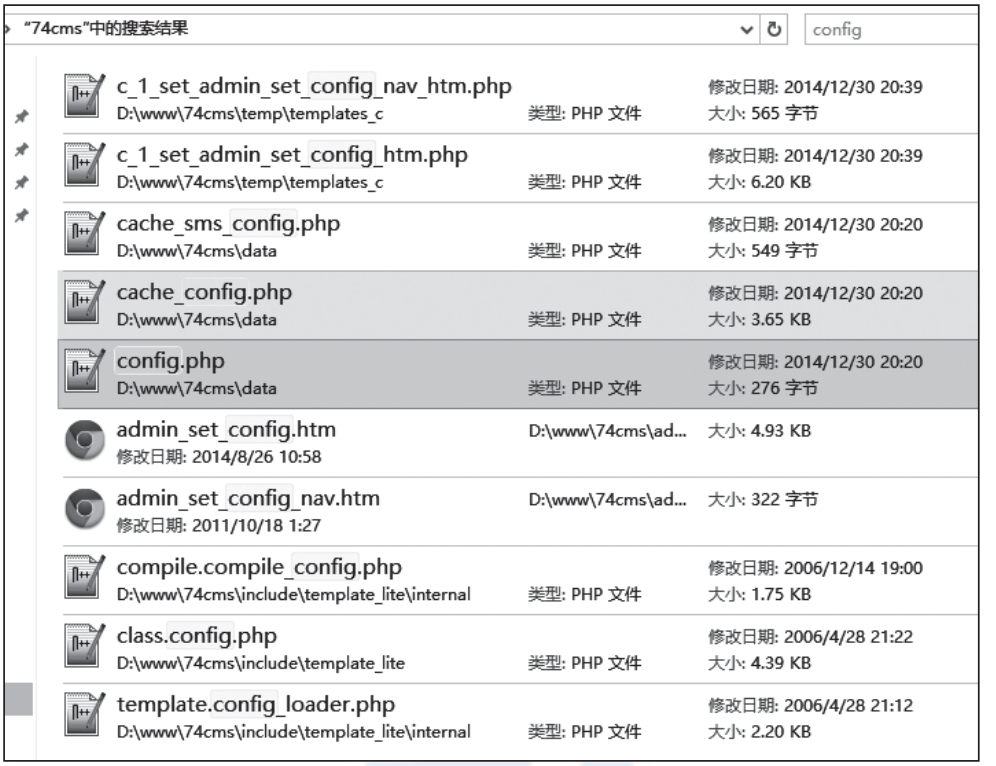


图 3-11

```
<?php
$dbhost   = "localhost";
$dbname   = "74cms";
$dbuser   = "root";
$dbpass   = "123456";
$pre   = "qs_";
$QS_cookiedomain = '';
$QS_cookiepath =  "/74cms/";
$QS_pwdhash = "K0ciF:RkE4xNhu@S";
define('QISHI_CHARSET','gb2312');
define('QISHI_DBCHARSET','GBK');
?>
```

很明显看到，很有可能存在我们之前说过的双引号解析代码执行的问题，通常这个配置是在安装系统的时候设置的，或者后台也有设置的地方。另外我们还应该记住的一个点是 QISHI_DBCHARSET 常量，这里配置的数据库编码是 GBK，也就可能存在宽字节注入，不过需要看数据库连接时设置的编码，不妨找找看，找到骑士 cms 连接

MySQL 的代码在 include\mysql.class.php 文件的 connect() 函数，代码如下：

```
function connect($dbhost, $dbuser, $dbpw, $dbname = '', $dbcharset =
    'gbk', $connect=1){
   $func = empty($connect) ? 'mysql_pconnect' : 'mysql_connect';
   if(!$this->linkid = @$func($dbhost, $dbuser, $dbpw, true)){
      $this->dbshow('Can not connect to Mysql!');
   } else {
      if($this->dbversion() > '4.1'){
         mysql_query( "SET NAMES gbk");
         if($this->dbversion() > '5.0.1'){
            mysql_query("SET sql_mode = ''",$this->linkid);
            mysql_query("SET character_set_connection=".$dbcharset.",
                character_set_results=".$dbcharset.", character_set_
                client=binary", $this-> linkid);
          }
      }
   }
   if($dbname){
      if(mysql_select_db($dbname, $this->linkid)===false){
         $this->dbshow("Can't select MySQL database($dbname)!");
      }
   }
}
```

这段代码里面有个关键的地方，见加粗代码，这里存在安全隐患。

代码首先判断 MySQL 版本是否大于 4.1，如果是则执行如下代码：

```
mysql_query( "SET NAMES gbk");
```

执行这个语句之后再判断，如果大于 5 则执行如下代码：

```
mysql_query("SET character_set_connection=".$dbcharset.",
    haracter_set_results=".$dbcharset.", character_set_client=binary",
    $this->linkid);
```

也就是说在 MySQL 版本小于 5 的情况下是不会执行这行代码的，但是执行了“set names gbk”，我们在之前介绍过“set names gbk”其实干了三件事，等同于：

```
SET character_set_connection=’ gbk’, haracter_set_results=’ gbk’,
    character_set_client=’gbk’
```

因此在 MySQL 版本大于 4.1 小于 5 的情况下，基本所有跟数据库有关的操作都存在宽字节注入。

3.2.1.4 跟读首页文件

通过对系统文件大概的了解，我们对这套程序的整体架构已经有了一定的了解，但是还不够，所以我们得跟读一下 index.php 文件，看看程序运行的时候会调用哪些文件和函数。

打开首页文件 index.php 可以看到如下代码：

```
if(!file_exists(dirname(__FILE__).'/data/install.lock'))
    header ("Location:install/index.php");
define('IN_QISHI', true);
$alias="QS_index";
require_once(dirname(__FILE__).'/include/common.inc.php');
```

首先判断安装锁文件是否存在，如果不存在则跳转到 install/index.php，接下来是包含 /include/common.inc.php 文件，跟进该文件查看：

```
require_once(QISHI_ROOT_PATH.'data/config.php');
header("Content-Type:text/html;charset=".QISHI_CHARSET);
require_once(QISHI_ROOT_PATH.'include/common.fun.php');
require_once(QISHI_ROOT_PATH.'include/74cms_version.php');
```

/include/common.inc.php 文件在开头包含了三个文件，data/config.php 为数据库配置文件，include/common.fun.php 文件为基础函数库文件，include/74cms_version.php 为应用版本文件。接着往下看：

```
if (!empty($_GET))
{
$_GET  = addslashes_deep($_GET);
}
if (!empty($_POST))
{
$_POST = addslashes_deep($_POST);
}
$_COOKIE   = addslashes_deep($_COOKIE);
$_REQUEST  = addslashes_deep($_REQUEST);
```

这段代码调用了 include/common.fun.php 文件里面的 addslashes_deep() 函数对

GET/POST/COOKIE 参数进行了过滤，再往下走可以看到又有一个包含文件的操作：

```
require_once(QISHI_ROOT_PATH.'include/tpl.inc.php');
```

包含了 include/tpl.inc.php 文件，跟进看看这个文件做了什么：

```
include_once(QISHI_ROOT_PATH.'include/template_lite/class.template.
    php');
$smarty = new Template_Lite;
$smarty -> cache_dir = QISHI_ROOT_PATH.'temp/caches/'.$_CFG['template_
    dir'];
$smarty -> compile_dir =  QISHI_ROOT_PATH.'temp/templates_c/'.$_CFG
    ['template_dir'];
$smarty -> template_dir = QISHI_ROOT_PATH.'templates/'.$_CFG['template_
    dir'];
$smarty -> reserved_template_varname = "smarty";
$smarty -> left_delimiter = "{#";
$smarty -> right_delimiter = "#}";
$smarty -> force_compile = false;
$smarty -> assign('_PLUG', $_PLUG);
$smarty -> assign('QISHI', $_CFG);
$smarty -> assign('page_select',$page_select);
```

首先看到包含了 include/template_lite/class.template.php 文件，这是一个映射程序模板的类，由 Paul Lockaby paul 和 Mark Dickenson 编写，由于该文件较大，我们这里不再仔细分析，继续往下跟进，可以看到这段代码实例化了这个类对象赋值给 $smarty 变量，继续跟进则回转到 index.php 文件代码：

```
if(!$smarty->is_cached($mypage['tpl'],$cached_id))
{
require_once(QISHI_ROOT_PATH.'include/mysql.class.php');
$db = new mysql($dbhost,$dbuser,$dbpass,$dbname);
unset($dbhost,$dbuser,$dbpass,$dbname);
$smarty->display($mypage['tpl'],$cached_id);
}
else
{
$smarty->display($mypage['tpl'],$cached_id);
}
```

判断是否已经缓存，然后调用 display() 函数输出页面，审计到这里是否对整个程

序的框架比较熟悉了？接下来像审计 index.php 文件一样跟进其他功能入口文件即可完成代码通读。

3.3 根据功能点定向审计

在有了一定的代码审计经验之后，一定会知道哪些功能点通常会有哪些漏洞，在我们想要快速挖掘漏洞的时候就可以这样来做，首先安装好并且运行程序，到处点点，浏览一下，看下程序有哪些功能，这些功能的程序文件分别是怎么样的，是独立的模块还是以插件形式存在，或者是写在一个通用类里面，然后多处调用。在了解这些功能的存在形式后，可以先寻找经常会出问题的功能点，简单黑盒测试一下，如果没有发现很普通、很常见的漏洞，再去读这个功能的代码，这样我们读起来就可以略过一些刚才黑盒测试过的点，提高审计速度。

根据经验，我们来简单介绍几个功能点经常会出现的漏洞，如下所示：

1）**文件上传功能**。这里说的文件上传在很多功能点都会出现，比如像文章编辑、资料编辑、头像上传、附件上传，这个功能最常见的漏洞就是任意文件上传了，后端程序没有严格地限制上传文件的格式，导致可以直接上传或者存在绕过的情况，而除了文件上传功能外，还经常发生 SQL 注入漏洞。因为一般程序员都不会注意到对文件名进行过滤，但是又需要把文件名保存到数据库中，所以就会存在 SQL 注入漏洞。

2）**文件管理功能**。在文件管理功能中，如果程序将文件名或者文件路径直接在参数中传递，则很有可能会存在任意文件操作的漏洞，比如任意文件读取等，利用的方式是在路径中使用 ../ 或者 ..\ 跳转目录，如图 3-12 所示。

除了任意文件操作漏洞以外，还可能会存在 XSS 漏洞，程序会在页面中输出文件名，而通常会疏忽对文件名进行过滤，导致可以在数据库中存入带有尖括号等特殊符号的文件名，最后显示在页面上的时候就会被执行。

3）**登录认证功能**。登录认证功能不是指一个登录过程，而是整个操作过程中的认证，目前的认证方式大多是基于 Cookie 和 Session，不少程序会把当前登录的用户账号等认证信息放到 Cookie 中，或许是加密方式，是为了保持用户可以长时间登录，不会一退出浏览器或者 Session 超时就退出账户，进行操作的时候直接从 Cookie 中读取出当前用户信息，这里就存在一个算法可信的问题，如果这段 Cookie 信息没有加 salt 一类的东西，就可以导致任意用户登录漏洞，只要知道用户的部分信息，即可生成认证

图　3-12

令牌，甚至有的程序会直接把用户名明文放到 Cookie 中，操作的时候直接取这个用户名的数据，这也是常说的越权漏洞。

ESPCMS 就多次被曝光存在这个漏洞，具体的漏洞分析在乌云上面可以直接看到，其中一个漏洞信息如下，感兴趣的读者可以研究一下：

缺陷编号：　WooYun-2015-90324

漏洞标题：　ESPCMS 所有版本任意用户登录

相关厂商：　易思 ESPCMS 企业网站管理系统

漏洞作者：　路人甲

4）**找回密码功能**。找回密码虽然看起来不像任意文件上传这种可以危害到服务器安全的漏洞，但是如果可以重置管理员的密码，也是可以间接控制业务权限甚至拿到服务器权限的。而找回密码功能的漏洞有很多利用场景，最常见的是验证码爆破，目前特别是 APP 应用，请求后端验证码的时候大多是 4 位，并且没有限制验证码错误次数和有效时间，于是就出现了爆破的漏洞。除此之外，还有验证凭证的算法，这需要在代码中才能看到，所以我们做代码审计的时候可以看看这个算法是否可信。

这些功能点上的漏洞需要我们多读代码才能积累经验。

BugFree 重装漏洞案例

针对功能点的审计是相对简单的，不过在使用这种方式审计之前建议先了解整个程序的架构设计和运行流程，程序重装漏洞在早期是比较常见的，我们来看看 BugFree 的程序安装功能，该程序之前被 papaver 爆出存在多个漏洞，其中就有一个重装漏洞。

BugFree 安装文件在 install\index.php，代码如下：

```
<?php
require_once('func.inc.php');

set_time_limit(0);
error_reporting(E_ERROR);
// 基本路径
define('BASEPATH', realpath(dirname(dirname(__FILE__))));
// upload path
define('UPLOADPATH', realpath(dirname(dirname(dirname(__FILE__)))).
   DIRECTORY_SEPARATOR.'BugFile');
// 配置样本文件路径
define('CONFIG_SAMPLE_FILE', BASEPATH . '/protected/config/main.sample.
    php');
// 配置文件路径
define('CONFIG_FILE', BASEPATH . '/protected/config/main.php');
```

看这段代码首先包含了 'func.inc.php' 文件，跟进这个文件可以看到一些读取配置文件、检查目录权限以及服务器变量等功能的函数，下方则是定义配置文件的路径，继续往下走，真正进行程序逻辑流程的地方如下代码所示：

```
$action = isset($_REQUEST['action']) ? $_REQUEST['action'] : CHECK;
if(is_file("install.lock") && $action != UPGRADED && $action != INSTALLED)
{
   header("location: ../index.php");
}
```

这段代码存在一个逻辑漏洞，首先判断 install.lock 文件是否存在以及 action 参数值是否升级完成和安装完成，如果是，则跳转到程序首页，这里仅仅使用了

```
header("location: ../index.php");
```

并没有使用 die( ) 或者 exit( ) 等函数退出程序流程，这个跳转只是 HTTP 头的跳转，下方代码依然会继续执行，这时候如果使用浏览器请求 install/index.php 文件则会跳转到

首页，我们用 burp 试试，效果如图 3-13 所示。

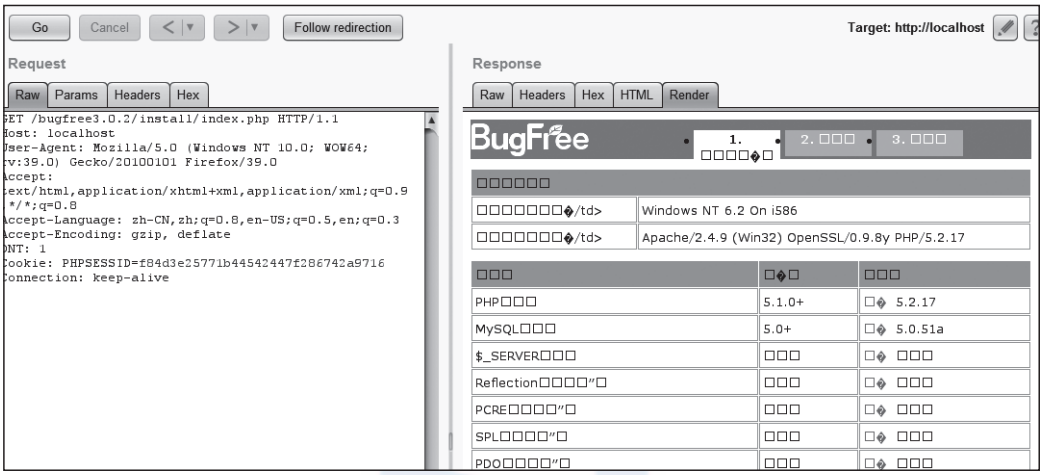


图　3-13

可以看到程序又可以再次安装。