

网络对抗原理与技术

——漏洞

—

尹钰 yinyu@xidian.edu.cn

漏洞

仍然要从概念说起
#

软件漏洞 v.s. 缺陷

配置缺陷 v.s. 设计缺陷

通用软件漏洞 v.s. 应用漏洞

OWASP TOP 10

从应用漏洞说起

##

注入

失效的身份认证和会话管理

跨站脚本（XSS）

不安全的直接对象引用

安全配置错误

敏感信息泄漏

功能级访问控制缺失

跨站请求伪造（CSRF）

使用含有已知漏洞的组件

未验证的重定向和转发

注入

都是越界惹的祸

###

缓冲区溢出

命令行注入

Struts2 的一系列 OGNL 注入

SQL 注入

针对 NoSQL 的注入

甚至 XSS 都可以理解为注入

怎么学

以 SQL 注入为例
###

场景

迈出第一步

细节

SQL 注入

从最典型最普遍的开始
####

```
SELECT * FROM t
```

```
WHERE c = '$c'
```

SQL 注入

万能密码

```
SELECT * FROM user WHERE
```

```
    uname = '$u'
```

```
AND password = '$p'
```

u : admin

p : 'a' OR 'a'='a'

```
SELECT * FROM user WHERE
```

```
    uname = 'admin'
```

```
AND password = 'a' OR 'a'='a'
```

SQL 注入

获取更多的信息

本地实例

尝试控制参数并看结果

工具 - sqlmap

抓包

日志

数据库验证

SQL 注入

sqlmap 的几个基本参数

```
-u -p --data --cookie -r  
  
--level --risk  
  
--dbs --users --tables  
  
--current-user --current-db  
  
-D -T -C --dump --sql-shell  
  
--os-shell --os-cmd
```

SQL 注入

sqlmap 的几个进阶参数

--technique BEUSTQ

--string --not-string --regexp

--code --titles

--where --start --stop

--flush-session --fresh-queries

--hex --time-sec --dbms

SQL 注入

sqlmap tamper

apostrophemask.py
apostrophencode.py
appendnullbyte.py
base64encode.py
base64encode.pyc
between.py
bluecoat.py
chardoubleencode.py
charencode.py
charunicodeencode.py
comma2wide.py
comma2wide.pyc
concat2concatws.py
equaltolike.py
greatest.py
halfversionedmorekeywords.py
ifnull2ifisnull.py
ifnull2ifisnull.pyc

__init__.py
lowercase.py
modsecurityversioned.py
modsecurityzeroverSIONED.py
msjs.pcap
multiplespaces.py
nonrecursivereplacement.py
overlongutf8.py
percentage.py
randomcase.py
randomcomments.py
securesphere.py
space2comment.py
space2dash.py
space2hash.py
space2morehash.py
space2mssqlblank.py

SQL 注入

攻击向量

a' AND UNION ALL SELECT 'X' #

闭合部分

注入向量

封闭部分

SQL 注入

封闭

AND 'a' = 'a

-- 注意后面有个空格

注意编码问题

-- # 最保险

SQL 注入

注入技术

Union query

boolean based

time base

NoSQL 注入

结构溢出
####

```
http://127.0.0.1/1.php?u=u1&p=1  
23456
```

```
db.user.find({u:'u1',p:'123456'});
```

```
http://127.0.0.1/1.php?  
u=u1&p[$ne]=1
```

```
db.user.find({u:'u1',p: {'$ne': '1'}});
```

命令行注入

与 SQL 注入类似
####

```
$c = _GET["c"];
```

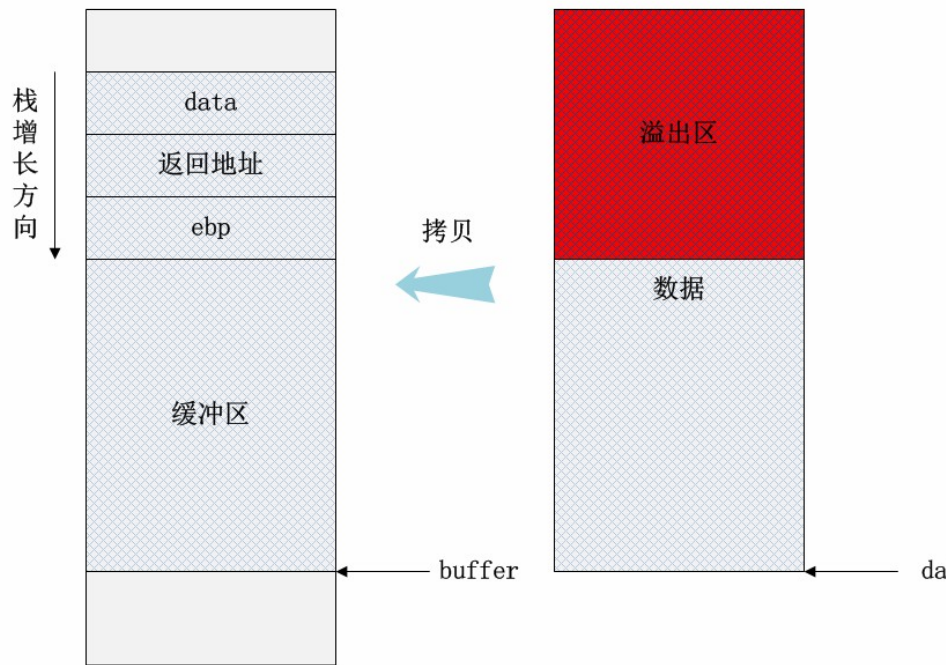
```
$cmd = "ping -c $c 127.0.0.1";
```

```
system($cmd);
```

缓冲区溢出

古老而新鲜

```
void fun(unsigned char *data)
{
    unsigned char buffer[BUF_LEN];
    strcpy((char*)buffer, (char*)data); // 溢出点
}
```



Struts2 OGNL

直接从 string 到 命令
####

```
(c)(('#rt.exec("calc")')  
  (#rt=@java.  
    lang.Runtime@getRuntime()))
```

```
java.lang.Runtime.getRuntime().  
  exec("calc");
```

XSS

半没落的漏洞之王
###

HTML 模板 + 变量

Hello \$i

\$i = `<script>alert(1)</script>`

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

XSS

闭合、越界、封闭
###

```
<input value="$n"/>
```

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

```
<input value="">  
  <script>alert(1)</script>  
  <a="a"/>
```

XSS

挡不住作死

###

```
<script>
```

```
.....
```

```
var a=$a;
```

```
.....
```

```
</script>
```

XSS

存储型
###

攻击行为 和 受害行为 分开

空间 时间 深度

XSS

限制
###

http_only

客户端限制

上下游的限制与弱化

URL 任意跳转

钓鱼
###

原理很简单

白名单 v.s. 黑名单

各种绕过

未验证的转发

突破内网边界
###

SSRF

翻译 & 解码

代理

服务路由

任意文件读取

XXE

CSRF

不盗号而达到目的
###

投票的例子

a.com/vote?id=121

转钱？ 刷票？ 授权？ 蠕虫？

a.com/vote?id=121&token=ac12cf

其他办法： POST？ Refer？

水平权限漏洞

认为用户不会修改请求
###

/getAddressList

/getAddress?aid=1231

不止是查询

修改密码

1 元订单

测试、利用简单，难以统一避免

访问控制缺失

把钥匙藏在窗台下
###

靠不知道路径

靠不知道请求格式

各种空密码

开放目录访问

失效的身份认证 和会话管理

被盗号
###

弱密码

没有防爆破限制

Session 伪造

Session 泄漏

其他安全问题

OWASP TOP 10

##

安全配置错误

使用含有已知漏洞的组件

敏感信息泄漏
