文章中涉及的代码都会开源，欢迎大家一起加入。

简介

在一个嘈杂的环境中，怎样才能尽可能的发现异常？不外乎黑白名单。
黑名单，又可以总结出两种方式：

> 1.基于特征的检测，2.基于行为的检测

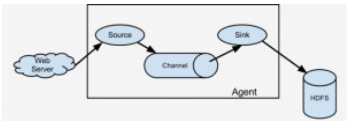
基于特征，是一种立竿见影的手段，对于一般的攻击很有效，但是永远不可能做到百分百，并且实效性极强，需要强大的响应队伍，对新漏洞尽可能快地做成特征库。
基于行为，是一种较为复杂的方式，是通过数学统计的方式来寻找异常，通过模式学习来寻找异常，但缺点是准确度不确定，可以做到很高，但误报率也很高。

为了能够防范未知漏洞和实时性的要求，同时能够灵活变通，我开始建立以hadoop，mongodb，python为基础的大数据分析平台，用来从公司的流量以及web日志中进行数据挖掘。

在进行了一段时间的思考和调研后，初步确定了下面的架构：

采集部分：

> IIS-》flume-》hdfs
> snort-》flume-》hdfs



统计分析模块：

> M/R python 分析程序-》hdfs

白名单模块：

> M/R python 过滤-》 hdfs

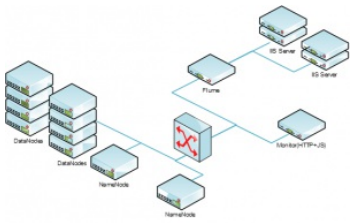
规则分析模块：

> M/R python 规则-》mongodb

可视化模块：

> php jquery -》监控平台

其他联动模块：
ips
扫描器
防火墙



通过这个架构，可以让我对整个公司web应用的访问情况，服务器的负载情况，攻击情况，异常访问进行直观的展示。
同时为整个互联网区域提供基础的分析平台。

Flume采集部分

由于公司使用的是IIS，所以在采集的时候遇到了一些困难：

> 1.Flume在linux下工作良好，但是在windows下用的人相对较少，版本也不够新；
> 2.IIS日志格式较为复杂，一个日志目录下，多个子站点的文件夹，每个文件夹的名字是随机的，并且设置了日志切分，文件会不断刷新，而flume的日志采集是基于tail的；
> 3.windows下的tail(gnu32)工作不稳定，出现各种崩溃错误。
> 为了解决这些问题，需要对flume进行一些配置和编写自己的程序。
> 1)使用静态编译的flume1.31-bin版本，测试过后发现没问题。
> 2)编写了pathtail.py，编译成exe来代替tail.exe，兼顾目录监控，同时整合所有的web日志成一个flume进程输出，还可以跟踪最新的日志文件。。
> 3)将flume打包注册成service，方便管理。
> 4.扩展pathtail.py,编写监控模块，用来监控实时的访问流量和负载。

针对某个业务系统的IIS服务器，总共使用12个Agent，1到2个汇聚服务器，1个Sink连接。每天的日志量为，15G（单台）*12 = 180G 一年为180*365 = 65.7 TB，按照HADOOP的冗余架构，整体数据量为65.7*3 = 197.1TB。压缩后的IIS一天IIS日志约为400M，12台为4.8G，一年为1752G，冗余后为5.256T。

为了实现IIS日志的准实时性分析，需要计算每分钟负载，设定每天高峰交易时间为早上8点至晚上9点，共13个小时，计算得到每5分钟负载约为：180G/24*13/60*5 约为10G。

根据目前在实验环境进行的测试得结果：

计算节点数量	计算量	消耗时间	结果统计	总耗时
3	73.5G	约15分钟	约30秒	约16分钟

推测计算时间为：

计算节点数量	计算量	消耗时间	结果统计	总耗时
3	10G	约3分钟	约10秒	约4分钟

勉强能够完成任务。

因此为了保证实时性，计划部署的初期HADOOP集群为：

计算节点数量	计算量	消耗时间	结果统计	总耗时
6	10G	约3分钟	约10秒	约4分钟

代码在这里 http://linxinsnow.me/?p=108
flume 配置 http://linxinsnow.me/?p=119

IIS日志分析

根据flume的配置，我们将IIS的日志进行一分钟切割，按照每分钟一个文件夹，每个文件夹按照10秒钟/128M文件进行切割，然后通过M/R框架，对文件夹的日志进行1初步统计分析。这里的统计内容为日志内字段的关联结果。
IIS的日志字段，根据配置的不同，可以多达22个字段，分别是：

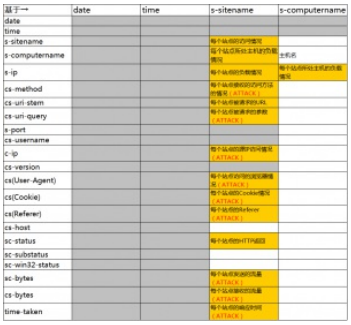
> date：发出请求时候的日期。time：发出请求时候的时间。
> 注意：默认情况下这个时间是格林威治时间，比我们的北京时间晚8个小时，下面有说明。
> c-ip：客户端IP地址。
> cs-username：用户名，访问服务器的已经过验证用户的名称，匿名用户用连接符-表示。

s-sitename：服务名，记录当记录事件运行于客户端上的Internet服务的名称和实例的编号。
s-computername：服务器的名称。
s-ip：服务器的IP地址。
s-port：为服务配置的服务器端口号。
cs-method：请求中使用的HTTP方法，GET/POST。
cs-uri-stem：URI资源，记录做为操作目标的统一资源标识符（URI），即访问的页面文件。
cs-uri-query：URI查询，记录客户尝试执行的查询，只有动态页面需要URI查询，如果有则记录，没有则以连接符-表示。即访问网址的附带参数。
sc-status：协议状态，记录HTTP状态代码，200表示成功，403表示没有权限，404表示找不到该页面，具体说明在下面。
sc-substatus：协议子状态，记录HTTP子状态代码。
sc-win32-status：Win32状态，记录Windows状态代码。
sc-bytes：服务器发送的字节数。
cs-bytes：服务器接受的字节数。
time-taken：记录操作所花费的时间，单位是毫秒。
cs-version：记录客户端使用的协议版本，HTTP或者FTP。
cs-host:记录主机头名称，没有的话以连接符-表示。注意：为网站配置的主机名可能会以不同的方式出现在日志文件中，原因是HTTP.sys使用Punycode编码格式来记录主机名。
cs(User-Agent)：用户代理，客户端浏览器、操作系统等情况。

这些字段就是我们寻找用户行为和攻击行为的切入点。
一个简单的例子：
从IIS字段中，我们选取几个要素IP，返回码，访问URL，即

```
c-ip,sc-status,cs-uri-stem
```

如果对这三个字段进行分别统计分析，我们可能可以得到一定时间范围内，服务器的访问情况，URL请求情况，返回码情况，是无法检测出一定异常的，因此我会对这三个字段进行关联分析，如：
1.每个IP在访问服务器的时候，他的状态码比例是如何的，如果是相同的业务访问，状态码的比例波动应该变化不大
2.对于一个IP，访问的URL应该是离散的，而不是聚合（收束）的，就是说，如果出现一个IP访问特定的几个URL的频率很高，那么很有可能这个就是个采集器，或者正在进行漏洞的验证。
3.对于特定的URL，其相关请求，如URL之间的关联关系，应该是类似的，因为所有页面请求基本有一样，同理同样的URL请求，它的状态码应该也是一样的，不应该出现较大波动，比如访问index.aspx,那肯定会访问index.css,如果没有，那可能就是自动化提交的工具。
4.URL的rank值在很大程度上也是有规律的，即IP用户的入口点是有规律的，在总结了所有的高可能性入口点后，如果出现某个IP突然访问一个新的入口点，那么这个IP可能是被XSS或者CSRF或者是webshell。



我们对IIS日志的所有22个字段都进行了这样的单独分析和关联分析，通过1维，二维，三维，多维的关联，进行统计分析，得出异常行为和用户的访问模式，变成插件的形式加载入M/R框架进行分析。
代码例子 http://linxinsnow.me/?p=98

项目还在进行中，后续会不断更新进展..

iis日志分析 # 安全日志分析

被以下专辑收录，发现更多精彩内容

+ 收入我的专辑

评论

按热度排序

请登录 / 注册 后在FreeBuf发布内容哦

相关推荐

关 注

0
文章数
0
评论数
0
关注者