目录

[0x01 MHN的安装及注意事项 2](#_Toc496972730)

[0x02 MHN的排错 2](#_Toc496972731)

[0x03 MHN的配置事项 4](#_Toc496972732)

[0x04 MHN的使用 5](#_Toc496972733)

[0x05 MHN蜜罐安装 6](#_Toc496972734)

[0x06 MHN蜜罐调试及排错： 6](#_Toc496972735)

[0x07 MHN框架解释 6](#_Toc496972736)

[0x08 MHN开发及拓展 7](#_Toc496972737)

[0x09 MHN参考： 8](#_Toc496972738)

# 0x01 MHN的安装及注意事项

因为MHN有部署脚本，所以安装MHN很简单，安装步骤如下：

yum update

yum install –y git sqlite-devel

cd /opt

sudo git clone https://github.com/threatstream/mhn

其中install.sh 即为项目的安装脚本，脚本是通过自身以及调用script文件夹下一系列的安装脚本实现安装服务的。

执行安装脚本之前要注意根据自己的机器环境更改一些相关内容，例如

机器自身的源是否稳定，是否执行update，python版本是否有冲突，supervisor安装的配置文件是否一样，安装脚本中有些组件的安装方式是否有变化，自己是否需要安装所有项目等确认事宜。

确认并更改完成之后运行安装脚本。

cd mhn

./install.sh

安装完成之后使用 sudo supervisorctl status 查看mhn服务各组件的运行状态。

# 0x02 MHN的排错

安装过程中碰到的一些报错问题及解决方案：

报错内容：

You must not use 8-bit bytestrings unless you use a text\_factory that can interpret 8-bit bytestrings (like text\_factory = str).

It is highly recommended that you instead just switch your application to Unicode strings.

[SQL: u'INSERT INTO rule\_references (text, rule\_id) VALUES (?, ?)']

[parameters: ('url,blogs.forcepoint.com/security-labs/trojanized-adobe-installer-used-install-dragonok\xe2\x80\x99s-new-custom-backdoor', 6673)]

解决方案：

编辑 /mhn/server/mhn/api/models.py 文件，在开头加入:

import sys

reload(sys)

sys.setdefaultencoding('utf-8')

将所有的 ref.text = r 替换成 ref.text = r.decode('utf-8')

保存文件。删除产生的 mhn/server/mhn.db文件。

注释掉安装脚本中成功的部分代码再新运行安装脚本，这样可以避免冲突，节省大量时间。

报错内容：

src/github.com/jt6211/sockjs-go/sockjs/rawwebsocket.go:4:2: cannot find package "golang.org/x/net/websocket" in any of:

/usr/lib/golang/src/golang.org/x/net/websocket (from $GOROOT)

/opt/honeymap/server/src/golang.org/x/net/websocket (from $GOPATH)

解决方案：

cd /opt/honeymap/server

export GOPATH=/opt/honeymap/server

go get github.com/golang/net

mkdir -p golang.org/x

cp -rf src/github.com/golang/net/ ./golang.org/x/

cp -rf golang.org/ /usr/local/go/src/

go build

sudo supervisorctl restart all

报错内容：

提示python sqlite 找不到依赖，或者存库时有编码问题。

此问题多为机器上本来就有python环境并且已经安装了sqlite依赖。但是两者之间版本不兼容导致。

解决方案：

查找当前系统python的版本，在另外的地方下载其源码包，重新编译。将编译得到的\_sqlite3.so文件复制到当前系统的python包lib目录下。

cp \_sqlite3.so /usr/local/lib/python2.7/sqlite3/

cp \_sqlite3.so /usr/local/lib/python2.7/lib-dynload/

新系统中安装不会出现此类问题，因为相关组件安装正确的话会自动安装到相关 lib 目录。

安装完成后执行sudo supervisorctl status查看服务运行状态有ERROR或者FATAL的情况：

可以使用supervisorctl命令和排查一些日志来判断问题出在哪里。

命令介绍：

supervisorctl update

#更新配置文件

supervisorctl status

#列出所有进程的状态

supervisorctl　restart [process|all]

#重启单个或者全部进程

supervisorctl start [process|all]

#开始单个进程或者全部进程

supervisorctl stop [process|all]

#停止单个进程或者全部进程

如果更新配置重启服务没有作用的话，就需在 /etc/supervisor/conf.d/查看对应进程的配置文件，找到并查看日志进行分析。

Supervisorctl命令报错：

解决方案：

查看/etc/supervisord.conf文件并根据系统重新配置supervisor进程位置。

# 0x03 MHN的配置事项

机器位于公网，就可以跳过这一步，如果只是放在内网里面，则需要配置mnemosyne的配置文件

sudo vim /opt/mnemosyne/mnemosyne.cfg

ignore\_rfc1918 = False

允许节点使用私有地址和服务器进行通信

重启服务

sudo supervisorctl restart Mnemosyne

如果想修改smtp服务的配置，可以编辑/opt/mhn/server/config.py

cd /opt/mhn/server

sudo vim config.py

sudo supervisorctl restart mhn-uwsgi

手动更改用户的管理密码

#!bash

sudo su

cd /opt/mhn/

source env/bin/activate

cd server

python manual\_password\_reset.py

deactivate

exit

假如要非匿名的将收集到得攻击数据回传到ThreatStream，要做以下操作

#!bash

cd /opt/mhn/scripts

sudo ./disable\_collector.sh

执行enable\_collector.sh可以开启

Web页面中Map 页面是显示的地图，这里一开始点击会显示说 404，需要在配置文件把页面地址更改一下 > vim /opt/mhn/server/config.py

把 > HONEYMAP\_URL = ‘:3000’

改成

HONEYMAP\_URL = ‘[http://ip:3000'](http://ip:3000'/)

ip 即为该服务器 ip

# 0x04 MHN的使用

运行：supervisorctl start all

一旦安装了基础的服务，就能够部署蜜罐节点了，通过web来展示相关数据等等，可以根据具体的境况做一些简单的调整。

更新机器防火墙配置，访问服务地址：[http://103.218:8080](http://103.235.226.218:8080) ，登录可以看到首页的数据汇总页面 。

这里介绍一下其他页面的作用：

Map页面即攻击数据实时展示页面，他会根据攻击者IP所在位置标出攻击点并动态展示。内网攻击无法在map页面展示。

Deploy 页面即保存部署蜜罐的脚本，可以下拉选择不同的蜜罐，也可以自定义脚本。它提供的脚本不一定完全适用，有些情况需要自己更改一下。复制上面的 Deploy Command 到另外的机器上执行，完成后一般就会在 Sensors 页面显示新安装的蜜罐节点。Sensors页面中有蜜罐名称与mongo数据库中的字段对应关系。这里要注意蜜罐是通过hpfeed服务向mhn服务器注册的，通过的端口是服务器的1000端口，安装蜜罐之前要注意防火墙策略是否打开。

蜜罐部署脚本的位置是 mhn/ scripts。更改部署页面上的脚本并不能改本地脚本。

Acctack 页面是攻击数据的详情页面，可以选择不同蜜罐的数据进行查看，信息有攻击时间，攻击蜜罐，攻击者国家，攻击者IP，目标端口，攻击协议，蜜罐类型。

Payloads 页面是蜜罐捕获的http服务 payload 数据展示页面。

Rules 页面是snort 捕获规则页面，可以查看规则详情并选择激活撤销规则。默认规则都是激活状态。

Sensors页面是蜜罐管理页面，可以添加删除蜜罐，展示信息有蜜罐名称，对应机器名称，机器IP，蜜罐类型，数据库对应ID，受攻击次数。

Charts 页面是ssh蜜罐捕获的数据展示页面，展示信息有爆破账号密码。

Settings 页面是设置页面，可以添加账号，更改密码，还显示了自己的API KEY。

# 0x05 MHN蜜罐安装

安装蜜罐节点比较容易， MHN上对每个蜜罐都有对应的安装脚本，所以安装起来比较容易，只需要根据自身系统环境对安装脚本进行一些微小的修改。

定义的安装脚本在MHN服务器的web界面中，在"Deploy"这个选项下有所有蜜罐的安装脚本。例如：

http://192.168.5.11/ui/manage-deploy/?script\_id=11

注意根据实际情况更新安装脚本，点击update按钮保存。然后使用系统给出的安装命令进行安装。

要格外注意服务器url地址要加上端口，以及mhn服务器的hpfeed以及web端口要开放给蜜罐IP。例如

wget "http://10218:8080/api/script/?text=true&script\_id=2" -O deploy.sh && sudo bash deploy.sh http://103.218:8080 GLPCsrAh

wget "http://106.218：8080/api/script/?text=true&script\_id=17" -O deploy.sh && sudo bash deploy.sh http://106.218：8080 GLPCsrAh

# 0x06 MHN蜜罐调试及排错：

脚本安装的蜜罐也是通过supervisor方式启动的，排错方式跟上面排查supervisor 服务错误的方式一样。

当出现mhn没有收到蜜罐消息的情况时，可以找到蜜罐的配置文件，

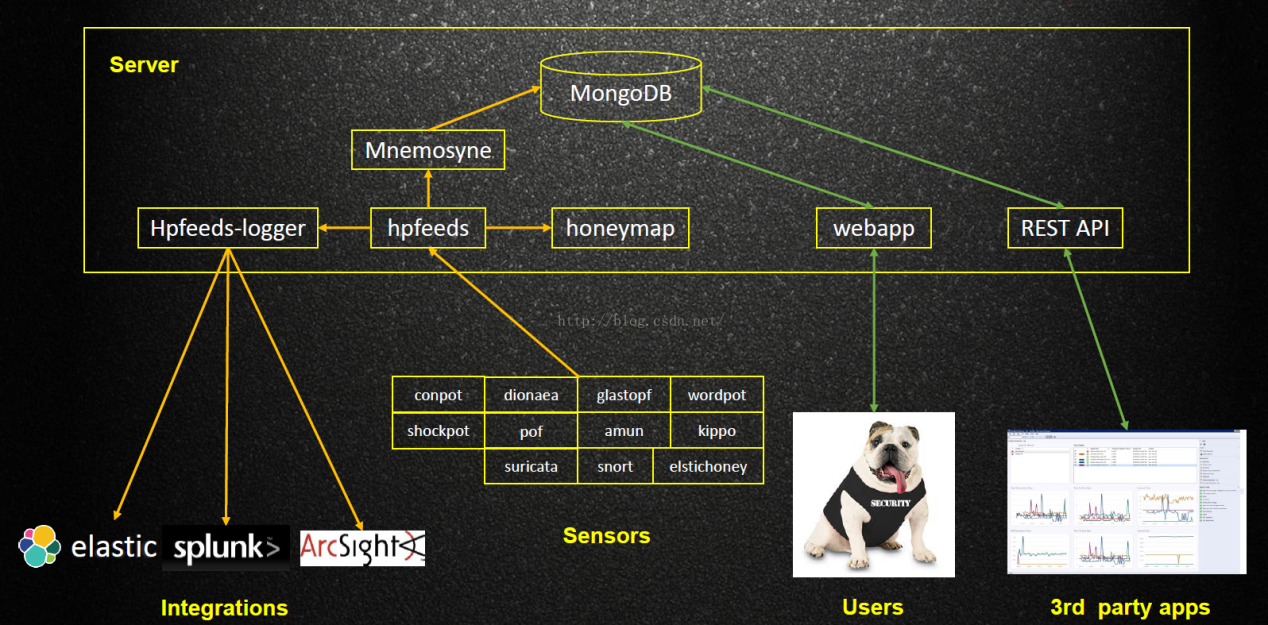
查看配置文件中自己蜜罐的名字，Hpfeed 的地址及端口是否正确，还可以排查两者之间的防火墙设置，蜜罐端配置以及hpfeed服务情况。

使用docker方式启动的蜜罐，可以在supervisor配置文件中查看其启动方式，端口映射，文件映射等信息。Dionaea蜜罐指纹修改的方法可以查看Dionaea安装及测试文档，其方法在这里同样有效。这里进入docker容器的命令可以用 docker exec -it a3f009163c25 /bin/bash ，a3f009163c25 是容器名，docker ps。

进入容器中，测试网络联通性。

# 0x07 MHN框架解释

MHN框架解释图



如图所示，mhn服务器即是上面框里的部分，蜜罐通过hpfeed协议传输数据给server端。

所以可以接入hpfeed的蜜罐理论上都可以接入到mhn框架中来。

Webapp即是服务器的web展示

REST API 并没有详细文档，只有几个示例用法，获取的数据即是web首页的汇总数据。

有一个第三方python包，<https://github.com/hrbrmstr/mhn>，里面的接口有限，获取的数据跟上边一样，没有拓展。

# 0x08 MHN开发及拓展

蜜罐报警机制设置：

5分钟查一次数据库，如果发现今天的记录并且没有重复，就发送报警邮件。

将同一个月的报警邮件全都写入到一个文件中。实现脚本为/opt/mhn/alarm.py

写一个脚本判断一个文件是否存在，存在则发送月统计邮件，不存在则发送上月没有蜜罐报警事件。计划任务每月1日运行这个脚本。实现脚本为/opt/mhn/summary.py

服务组件日志位置

/var/log/mhn/

日志位置：/var/dionaea

配置位置：/etc/Dionaea

配置文件中有自己蜜罐的名字，Hpfeed 的地址及端口

docker ps

进入docker命令：docker exec -it a3f009163c25 /bin/bash

进入mongo数据库：mongo 127.0.0.1:27017

use mnemosyne

show collections

daily\_stats

db.getCollection(‘daily\_stats').find({})

db.getCollection('hpfeed').find({})

新蜜罐接入的步骤：

1. 在新蜜罐添加hpfeed 支持，示例：https://github.com/threatstream/wordpot/pull/1/files
2. 在hpfeed中添加新蜜罐honey map支持，示例：<https://github.com/threatstream/hpfeeds/pull/9/files>
3. 更新[install\_honeymap.sh](https://github.com/threatstream/mhn/blob/master/scripts/install_honeymap.sh) 添加新蜜罐类型
4. 在Mnemosyne 中添加新蜜罐的数据格式化支持,示例：<https://github.com/threatstream/mnemosyne/pull/7/files>
5. 在mhn中添加新蜜罐注册权限并编写自动部署脚本，示例：<https://github.com/threatstream/mhn/pull/67/files>
6. 在数据库中新建hpfeed通信频道

# 0x09 MHN参考：

MHN支持的蜜罐及介绍

• Sort: https://www.snort.org/

• Suricata: http://suricata-ids.org/

• Dionaea: http://dionaea.carnivore.it/, 它是一个低交互式的蜜罐，能够模拟 MSSQL, SIP, HTTP, FTP, TFTP 等服务 drops 中有一篇介绍： http://drops.wooyun.org/papers/4584

• Conpot: http://conpot.org/

• Kippo: https://github.com/desaster/kippo，它是一个中等交互的蜜罐，能够下载任意文件。 drops 中有一篇介绍： http://drops.wooyun.org/papers/4578

• Amun: http://amunhoney.sourceforge.net/，它是一个低交互式蜜罐，但是已经从 2012 年之后不在维护了。

• Glastopf： http://glastopf.org/

• Wordpot： https://github.com/gbrindisi/wordpot

• ShockPot： https://github.com/threatstream/shockpot，模拟的 CVE-2014-6271，即破壳漏洞

• p0f： https://github.com/p0f/p0f

其他web蜜罐

https://github.com/mushorg/glastopf 8月前

https://github.com/dustyfresh/HoneyPress 7个月

https://github.com/mushorg/tanner 最近

https://shadowd.zecure.org/overview/introduction/

推荐蜜罐组合

Dionaea + kippo + snort + p0f

Amun + kippo + snort + p0f

Elastichoney + shockpot + snort + p0f

Conpot + snort + p0f

Elastichoney + Glastopf + snort + p0f

p0f和snort(suricata)结合部署在每天机器上可以提供实用的威胁情报，包括攻击者的操作系统，网络连接类型，payload类型等。

https://github.com/threatstream/mhn/wiki/Howto%3A-Add-Support-for-New-Sensors-to-the-MHN