实验：snort 与单台防火墙联动实验

预备知识

Snort 简述

Snort 是一个强大的轻量级网络入侵检测系统，它能够检测到各种不同的攻击方式，对攻击进行实时报警。此外，Snort 具有很好的扩展性和可移植性，并且这个软件遵循 GPL，这意味着只要遵守 GPL 的任何组织和个人均可以自由使用这个软件。

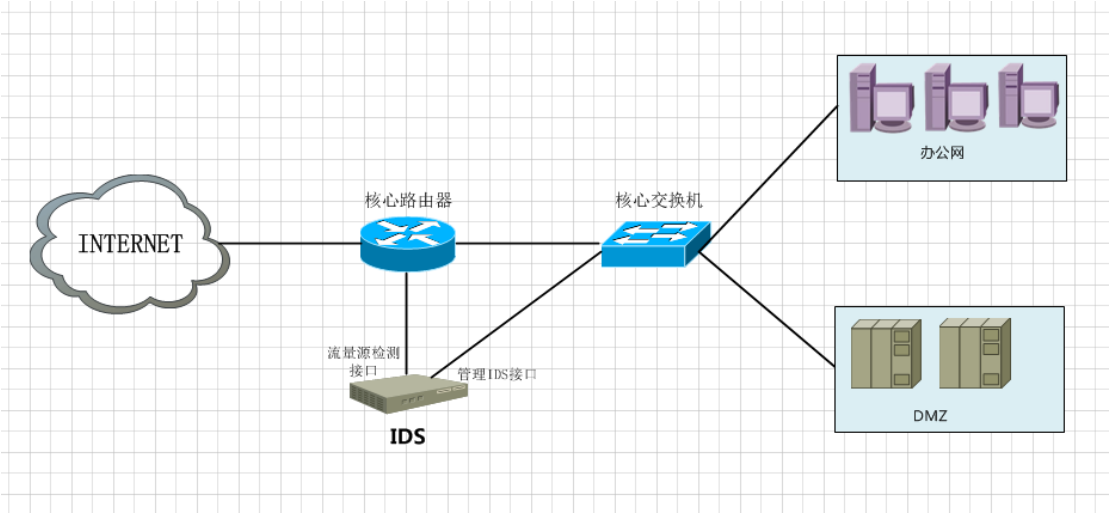
Snort 具有实时流量分析和日志 IP 网络数据包的能力，能够快速地检测网络攻击，及时地发出报警。Snort 的报警机制很丰富，例如：Syslog、用户指定的文件、一个 Unix 套接字，还有使用 samba 协议向 Windows 客户端发出 WinPoup 消息。利用 XML 插件，Snort 可以使用 SNML 把日志存放到一个文件或者实时报警。Snort 能够进行协议分析、内容的搜索/匹配。目前 Snort 能够分析的协议有 TCP、UDP、ICMP，将来可能支持 ARP、OSPF、IPX、RIP 等协议，它能够检测多种方式的攻击和探测，例如：缓冲区溢出、CGI 攻击、端口暴力破解、SMB 探测以及 web 应用程序的攻击现在也已经有简单规则的支持。Snort 自带的检测攻击的规则数量有限，但 Snort 支持用户自定义规则的加载，这对有能力的大型企业而言是个不错的 IDS 选择。

Snort 体系架构

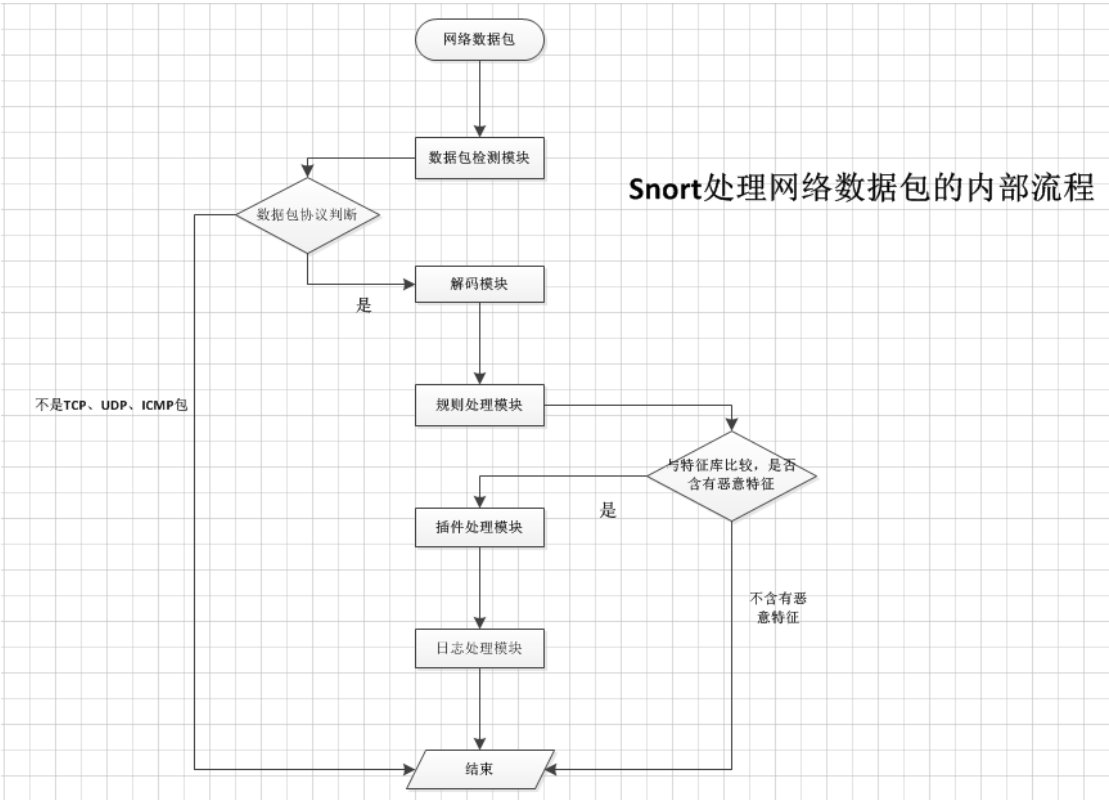
1.Snort 工作原理与应用场景

Snort 是一个基于模式匹配的的网络入侵检测系统，实际上目前现在市场上的大多商业入侵检测系统都是基于模式匹配的，即将恶意行为和恶意代码预定成入侵规则特征库，然后将实际数据源于规则库的特征码进行匹配，以判断其中是否包含了入侵行为。

IDS 的应用场景一般如下图所示：



Snort 的大致处理流程如下图所示：



2.Snort 的主体架构

Snort 系统总体上是由规则集及 Snort 可执行程序两大部分组成。

1）Snort 规则集

Snort 规则集是 Snort 的攻击特征库，每条规则是一条攻击标识，Snort 通过它来识别攻击行为。

2）Snort 可执行程序

可执行程序由 4 个重要的子系统构成：

数据包捕获和解码子系统、检测引擎、日志/报警子系统、预处理器。

Snort 的总体结构：

序号	模块名称	源文件名称	备注
1	主控模块	Snort.cPlugbase.c…	主控模块，完成插件的管理和服务功能。
2	解码模块	Decode.c…	完成报文解码功能，把网络数据包解码成 snort 定义的 Packet 结构，用于后续分析
3	规则处理模块	Rules.cParser.c…	完成所有与规则相关的功能
4	预处理插件模块	Spp_defrag.c…	模拟 tcp、ip 堆栈功能的插件、各种解码插件
5	处理插件模块	Sp_IP_fragbits.c…	辅助完成基于规则的匹配过程
6	输出插件模块	Spo_alert_fast.c…	系统日志和告警日志的输出处理
7	日志模块	Logs.c…	完成所有与日志记录有关的功能
8	辅助模块	Ubi_Bintree.c…	辅助功能

3. Snort 的插件机制

1）预处理插件

预处理插件在规则匹配之前运行，完成的功能主要为：

（1）模拟 tcp、ip 堆栈功能的插件，如 IP 碎片重组、TCP 流重组插件；

（2）各种解码插件：http 解码插件、unicode 解码插件、rpc 解码插件、Telnet 解码插件等；

（3） 规则匹配无法进行攻击检测时所用的插件：端口扫描插件、spade 异常入侵检测插件、bo 检测插件等

2）处理插件

处理插件在规则匹配阶段的 parse rule options 中被调用，辅助完成基于规则的匹配库。每个规则处理函数通常对应规则选项中的一个关键字，实现对这个关键字的解释。其主要功能为：

（1）检查协议各字段，如 TCPflag、ICMPtype、Fragbits、RPC、Dsize 等；

（2）辅助功能，例如关闭连接、会话记录、攻击响应等

3）输出插件

输出插件在规则匹配过程中和匹配过程结束后调用，以便记录日志和告警。

4.总体流程

Snort 的入侵检测流程分成两大步：

第一步是规则的解析流程，包括从规则文件中读取规则和在内存中组织规则。其过程为：

（1）读取规则文件；

（2）依次读取每条规则；

（3）解析规则；

（4）在内存中对规则进行组织，建立规则语法树；

第二步是使用这些规则进行匹配的入侵流程。其过程为对从网络上捕获的每一条数据报文和在第一步建立的规则树进行匹配，若发现存在一条规则匹配该报文，就表示检测到一个攻击，然后安装规则规定的行为进行处理；若搜索完所有的规则都没有找到匹配的规则，则视此报文正常。

Snort 与 Iptables 的联动

1. 概述

为什么要让 Snort 与 Iptables 联动呢？根据前面的介绍我们已经了解了 Snort 的工作方式与原理，聪明的你一定发现了 Snort 最致命的密码——不能阻断攻击！Snort 的主要作用是对整个网络起到预警作用，从它的旁路部署模式也可以看出，它并不能阻断网络里的攻击行为。谁能阻断攻击呢？——Iptable。可为什么不采用 Iptables 呢？因为 Iptables 的规则过于固定，并且 Iptables 并不能识别网络里的攻击行为。那能不能综合一下它们二者的优点互补对方的缺点，而达到检测到攻击即切断攻击连接这样的效果呢。答案是可以！

2. 实现方式和实现原理

通过前面的知识我们了解到，Snort 有个插件机制提供了预处理插件和处理插件等方式。而这种插件在 Snort 里是支持自定义开发并加载的。因此第一种实现方式就是自定义开发插件，当检测到规则匹配时则调用远程或对应主机的防火墙，将有入侵行为的 ip 和端口，建立对应的一条 Iptables 规则丢弃这个连接、端口的数据包或将此 ip 的所有包都丢弃。

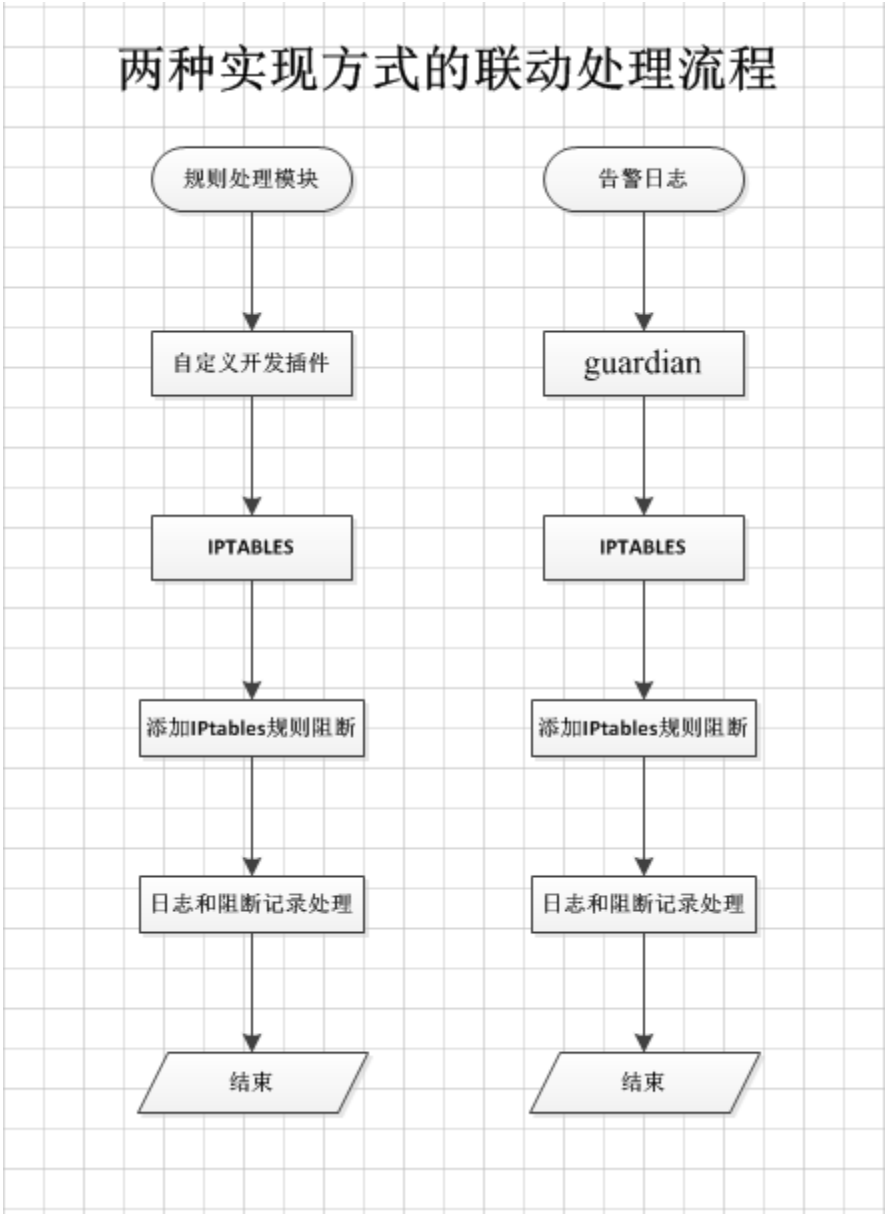
相对于 Snort 的插件方式，第二种的实现方式非常简单且易于实现，很适合本次实验。这种方式就是利用一个简单的脚本实时读取告警日志，将记录到的 Ip 和端口，创建对应的一条 Iptables 规则，加入到远程或对应主机的防火墙规则中，也就是实现了同第一种方式相同的功能，虽然后者在处理速度上没有第一种方式及时，但整体防护能力上并未有太大什么区别。

实现方式总结如下：

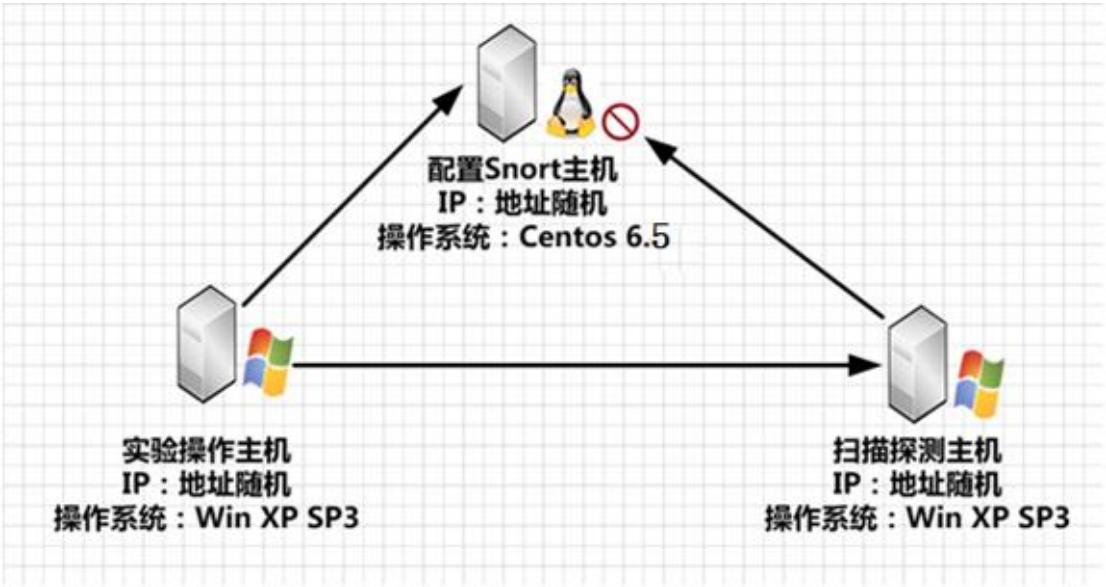
(1).利用 Snort 的扩展功能，自定义开发集成插件(目前有 snortsam)。

(2).利用 Snort 的告警日志，自定义开发脚本。(本次实验所采用，脚本为 guardian)

实现原理大致流程图：


两种实现方式的联动处理流程
规则处理模块
自定义开发插件
IPTABLES
添加IPtables规则阻断
日志和阻断记录处理
结束
告警日志
guardian
IPTABLES
添加IPtables规则阻断
日志和阻断记录处理
结束

实验环境



服务器:snort-host(Centos6.5)，IP 地址： 10.1.1.12

Snort 版本： 2.9.7.6(最新) Guardian 版本：1.7(最新)

操作主机:host(WinXp)， IP 地址：随机

测试主机:test(WinXp)， IP 地址：随机

下载路径：http://tools.hetianlab.com/tools/X-Scan-v3.3-cn.rar

下载路径：http://tools.hetianlab.com/tools/Xshell.rar

实验步骤一

Snort 与 guardian 的安装和配置

1. 操作指导（实验过程中操作命令综述）

Vim 编辑器是 linux 下的一款相当于记事本的编辑器。

命令：vi 某文件名即进入编辑状态，输入:i 即可插入或删除文字

按下 ESC 可退出编辑。

shift 键+冒号(:) 输入 q! 即不保存退出输入 x 即可保存退出

2. Snort 安装与配置

/********更新系统、安装 snort 依赖包、下载 snort、规则库*********/**

SSH 登录 centos6.5 主机

用户名：root

password:123456

查看网卡

ifconfig（如果网卡没启动）

ifup eth0

更新系统

yum –y update

安装 snort 依赖包

yum -y install pcre pcre-devel gcc gcc-c++ zlib zlib-devel libpcap libpcap-devel make autoconf flex byacc bison libxml2-devel wget tcpdump

下载 Snort、Snort 规则库、DAQ 、libdnet、guardian

(外网下载十分缓慢，因此安装包已经提前下载到实验环境服务器（http://tools.hetianlab.com/tools/）中)

下载 xshell 登陆 linux
下载地址 ：http://tools.hetianlab.com/tools/Xshell.rar

cd /usr/local/src/

wget http://tools.hetianlab.com/tools/snort+IPtables.tar.gz

tar zxvf snort+IPtables.tar.gz

安装 libdnet

cd /usr/local/src/snort+IPtables

tar zxvf libdnet-1.12.tgz

cd libdnet-1.12

./configure --with-pic

make && make install

cd /usr/local/lib/

ldconfig -v /usr/local/lib

安装 DAQ

cd /usr/local/src/snort+IPtables

tar zxvf daq-2.0.6.tar.gz

cd daq-2.0.6

./configure

make && make install

cd /usr/local/lib/

ldconfig -v /usr/local/lib

安装 snort

cd /usr/local/src/snort+IPtables/

tar zxvf snort-2.9.7.6.tar.gz

cd snort-2.9.7.6

./configure –enable-sourcefire

make && make install

cd /usr/local/lib/

ldconfig –v /usr/local/lib

Snort 配置

创建 snort 系统必要的文件夹，导入规则库，建立黑名单和白名单文件

mkdir /etc/snort

mkdir /var/log/snort

cd /etc/snort

cp /usr/local/src/snort+IPtables/snort-2.9.7.6/etc/* . (注意有个点号跟*号中间有空格)

tar zxvf /usr/local/src/snort+IPtables/snortrules-snapshot-2976.tar.gz

cp ./etc/* .

touch /etc/snort/rules/white_list.rules /etc/snort/rules/black_list.rules

为 snort 添加一个用户和组

groupadd -g 40000 snort

useradd snort -u 40000 -d /var/log/snort -s /sbin/nologin -c SNORT_IDS -g snort

cd /etc/snort

chown -R snort.snort *

chown -R snort.snort /var/log/snort

vi /etc/snort/snort.conf

ipvar HOME_NET 10.1.1.0/24（本次实验为此 ip 段地址）

ipvar EXTERNAL_NET any

var RULE_PATH /etc/snort/rules

var SO_RULE_PATH /etc/snort/so_rules

var PREPROC_RULE_PATH /etc/snort/preproc_rules

var WHITE_LIST_PATH /etc/snort/rules

var BLACK_LIST_PATH /etc/snort/rules

preprocessor sfportscan: proto　{ all } memcap { 10000000 } sense_level { low }(检测端口扫描，不去注释也可以，去掉注释用 nmap 扫描即可看到扫描日志)

include $PREPROC_RULE_PATH/preprocessor.rules(注释去掉)

include $PREPROC_RULE_PATH/decoder.rules(注释去掉)

include $PREPROC_RULE_PATH/sensitive-data.rules(注释去掉)

为 snort 的文件在另一个位置创建一个同步链接

ln -s /usr/local/bin/snort /usr/sbin/snort

修改 snort 及 daq 等相关目录和文件权限

cd /usr/local/src/snort+IPtables

chown -R snort.snort daq-2.0.6

chmod -R 700 daq-2.0.6

chown -R snort.snort snort-2.9.7.6

chmod -R 700 snort-2.9.7.6

cd /usr/local/src/

chown -R snort.snort snort_dynamicsrc

chmod -R 700 snort_dynamicsrc

cd /var/log

chown -R snort.snort snort

chmod -R 700 snort

cd /usr/local/bin

chown -R snort.snort daq-modules-config

chmod -R 700 daq-modules-config

chown -R snort.snort u2*

chmod -R 700 u2*

cd /etc

chown -R snort.snort snort

chmod -R 700 snort

配置动态规则

mkdir -p /usr/local/lib/snort_dynamicrules

cp /etc/snort/so_rules/precompiled/RHEL-6-0/x86-64/2.9*/*.so /usr/local/lib/snort_dynamicrules (实验环境是 64 位系统)

如果在 32 位系统上，这样导入动态规则:

cp /etc/snort/so_rules/precompiled/RHEL-6-0/i386/2.9*/*.so /usr/local /lib/snort_dynamicrules/

cd /usr/local/lib

chown -R snort.snort snort*

chmod -R 700 snort*

chown -R snort.snort pkgconfig

chmod -R 700 pkgconfig

导出动态规则文件

snort -c /etc/snort/snort.conf --dump-dynamic-rules=/etc/snort/so_rules

如报错，很有可能是导入动态规则时未能导入对应系统架构的规则文件。

设定告警文件权限

touch /var/log/snort/alert

cd /var/log/snort

chown snort.snort alert

chmod 700 alert

snort -T -c /etc/snort/snort.conf -i eth0 检测配置文件是否有错误

启动 snort

snort -c /etc/snort/snort.conf -i eth0

停止 snort

ps -ef |grep snort

kill -9 pid 号 即可结束进程

实验步骤二

安装配置 guardian

安装配置

cd /usr/local/src/snort+IPtables

tar zxvf guardian-1.7.tar.gz

cd guardian-1.7

touch /etc/snort/guardian.ignore

touch /etc/snort/guardian.target

touch /var/log/snort/guardian.log

cp guardian.pl /usr/local/bin/

cp scripts/iptables_block.sh /usr/local/bin/guardian_block.sh

cp scripts/iptables_unblock.sh /usr/local/bin/guardian_unblock.sh

cp guardian.conf /etc/snort

编辑 guardian 配置文件

vi /etc/snort/guardian.conf

Interface eth0

LogFile /var/log/snort/guardian.log

AlertFile /var/log/snort/alert //alert 文件的位置

IgnoreFile /etc/snort/guardian.ignore //白名单

targetFile /etc/snort/guardian.target //黑名单

TimeLimit 120 //阻断时间,以秒为单位

guardian 启动

/usr/bin/perl /usr/local/bin/guardian.pl -c /etc/snort/guardian.conf

启动成功会出现如下示例：

```
[root@root guardian-1.7]# /usr/bin/perl /usr/local/bin/guardian.pl -c /etc/snort/guardian.conf
OS shows Linux
Warning! HostIpAddr is undefined! Attempting to guess..
Got it.. your HostIpAddr is 10.1.1.12
My ip address and interface are: 10.1.1.12 eth0
Loaded 0 addresses from /etc/snort/guardian.ignore
Loaded 0 addresses from /etc/snort/guardian.target
Becoming a daemon..
[root@root guardian-1.7]#
```

Guardian 停止

ps -ef|grep guardian

kill -9 pid 号即可杀死该进程

实验步骤三

联动测试（在执行以下操作之前，请先停止 guradian，否则会导致无法登陆试验机。）

Snort 与本地 iptables 联动

测试规则是否加载生效

vi /etc/snort/rules/local.rules

添加下面两条规则

alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"OUT"; sid:5000005)

alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"IN"; sid:5000006)

规则说明：

告警外网和内网之间的所有 tcp 流量，用来测试你的 snort.conf 配置是否有问题

启动 snort，查看 alert 是否有日志

snort -c /etc/snort/snort.conf -i eth0

cd /var/log/snort

tail -f alert

如果可以看到日志，则表明 snort.conf 配置没有问题，可以继续下面的测试，否则请检查 snort.conf 的配置，以及 snort 目录和文件权限设置。

联动测试

如果之前的测试没有问题，请将/etc/snort/rules/local.rules 你所添加的两条规则删除或注释掉。

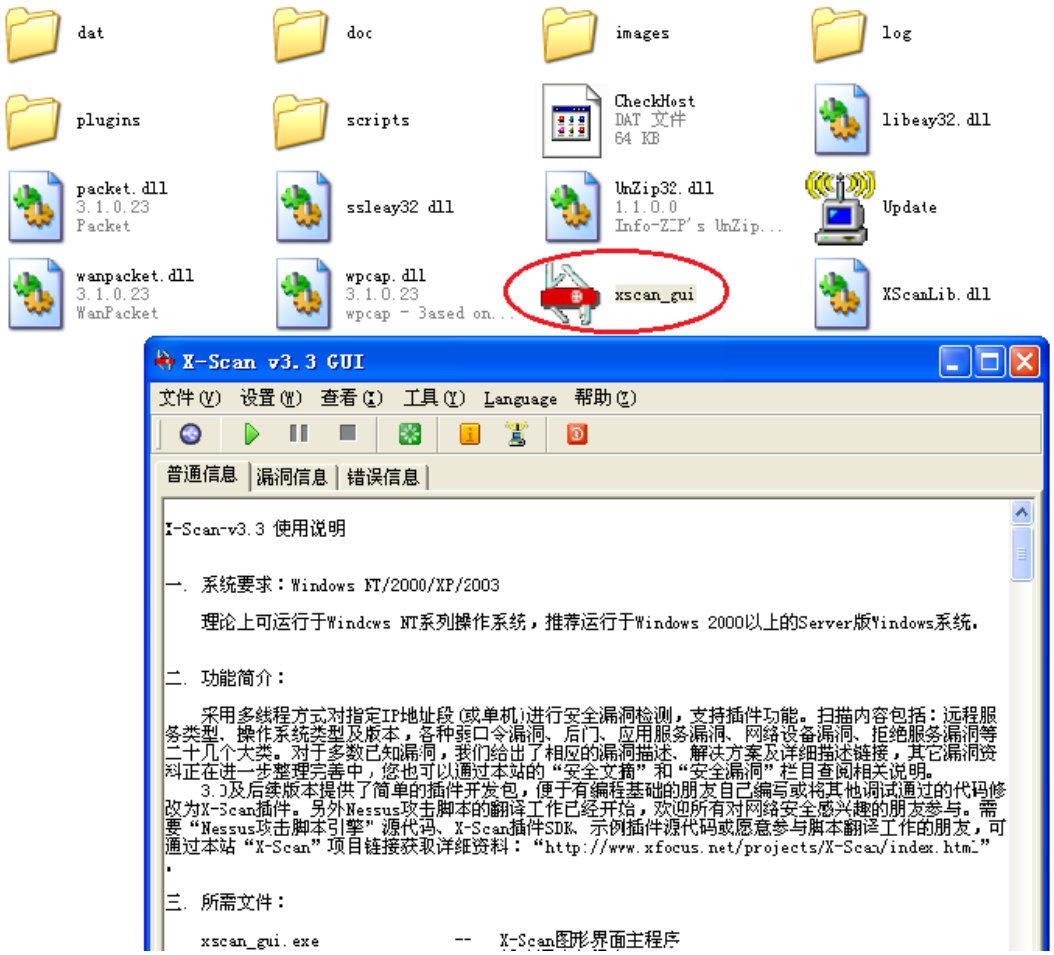
vi /etc/snort/rules/local.rules

#alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"OUT"; sid:5000005)

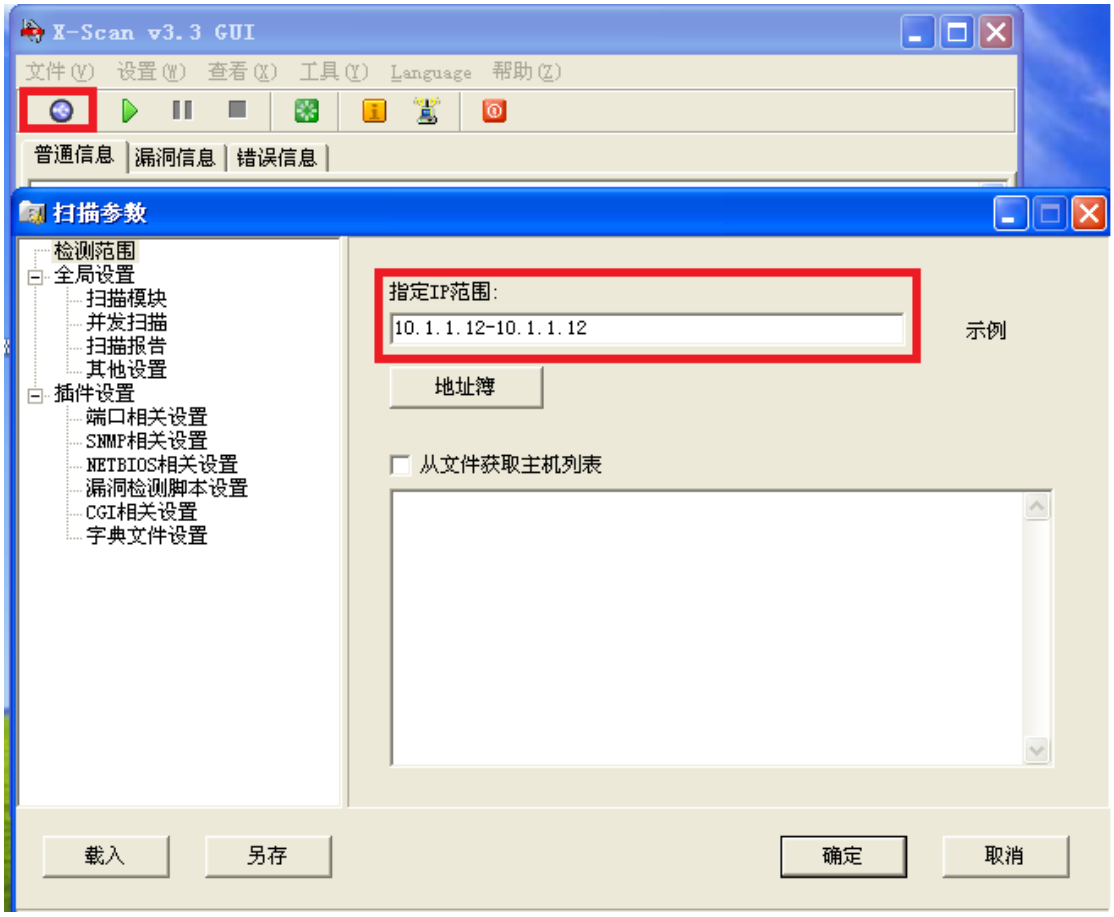
#alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"IN"; sid:5000006)

登录另一台 windows 测试主机(注意:如果用本机扫描很有可能导致 snort 主机无法登陆)，打开浏览器，登录 http://tools.hetianlab.com/tools/X-Scan-v3.3-cn.rar 下载 x-scan 并解压，双击文件夹

打开 x-scan

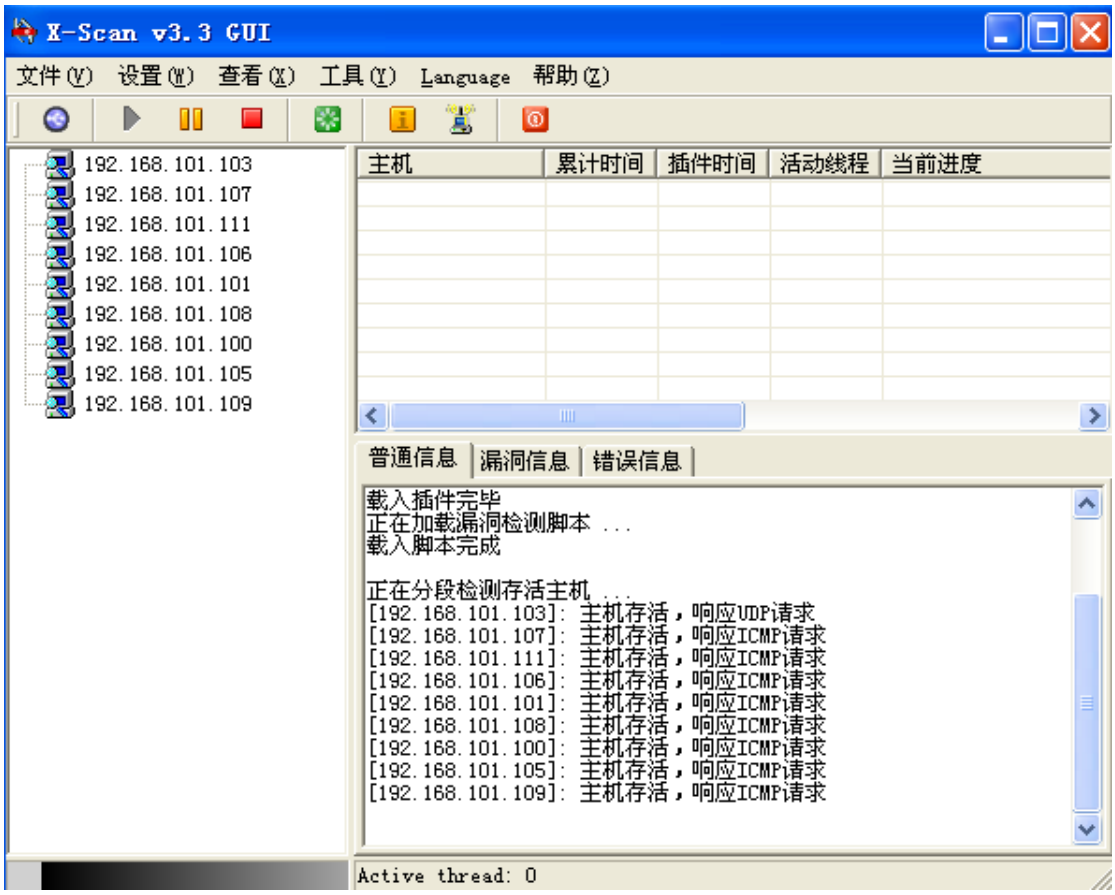

dat
doc
images
log
plugins
scripts
CheckHost
DAT 文件
64 KB
libeay32.dll
packet.dll
3.1.0.23
Packet
ssleay32 dll
UnZip32.dll
1.1.0.0
Info-ZIP's UnZip...
Update
wanpacket.dll
3.1.0.23
WanPacket
wpcap.dll
3.1.0.23
wpcap - 3ased on...
xscan_gui
XScanLib.dll
X-Scan v3.3 GUI
文件(V) 设置(W) 查看(I) 工具(Y) Language 帮助(Z)
普通信息 漏洞信息 错误信息
X-Scan-v3.3 使用说明
一、系统要求：Windows NT/2000/XP/2003
理论上可运行于Windows NT系列操作系统，推荐运行于Windows 2000以上的Server版Windows系统。
二、功能简介：
采用多线程方式对指定IP地址段(或单机)进行安全漏洞检测，支持插件功能。扫描内容包括：远程服务类型、操作系统类型及版本，各种弱口令漏洞、后门、应用服务漏洞、网络设备漏洞、拒绝服务漏洞等二十几个大类。对于多数已知漏洞，我们给出了相应的漏洞描述、解决方案及详细描述链接，其它漏洞资料正在进一步整理完善中，您也可以通过本站的“安全文摘”和“安全漏洞”栏目查阅相关说明。
3.1及后续版本提供了简单的插件开发包，便于有编程基础的朋友自己编写或将其他调试通过的代码修改为X-Scan插件。另外Nessus攻击脚本的翻译工作已经开始，欢迎所有对网络安全感兴趣的朋友参与。需要“Nessus攻击脚本引擎”源代码、X-Scan插件SDK、示例插件源代码或愿意参与脚本翻译工作的朋友，可通过本站“X-Scan”项目链接获取详细资料：“http://www.xfocus.net/projects/X-Scan/index.html”
三、所需文件：
xscan_gui.exe -- X-Scan图形界面主程序

设置扫描参数



点击“扫描参数”按钮，将指定 IP 范围设置成你的 snort 主机 ip 地址，在“全局设置”的“扫描模块”中选中“全选”，“插件设置”的“SNMP 相关设置”、“NETBIOS 相关设置”、“漏洞检测脚本设置”均选择全部选中。

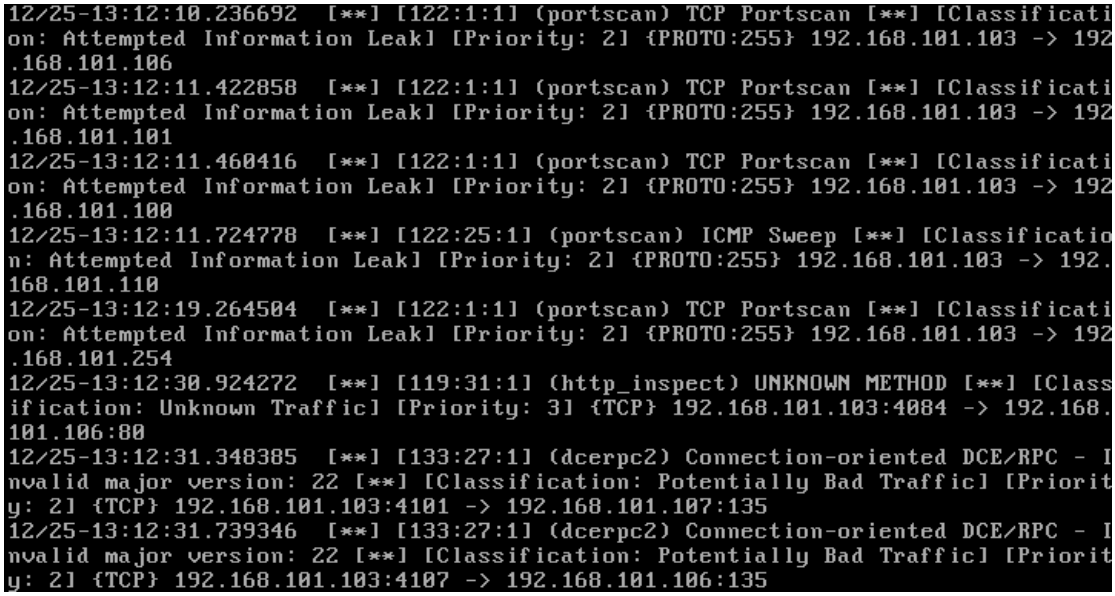
点击开始按钮，开始扫描



观察 **alert** 是否有告警日志

cd /var/log/snort

tail -f alert

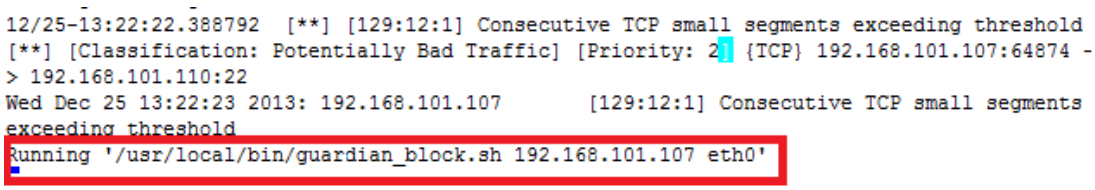


检测到的扫描日志

发现告警日志，启动 guardian 与 iptables 联动

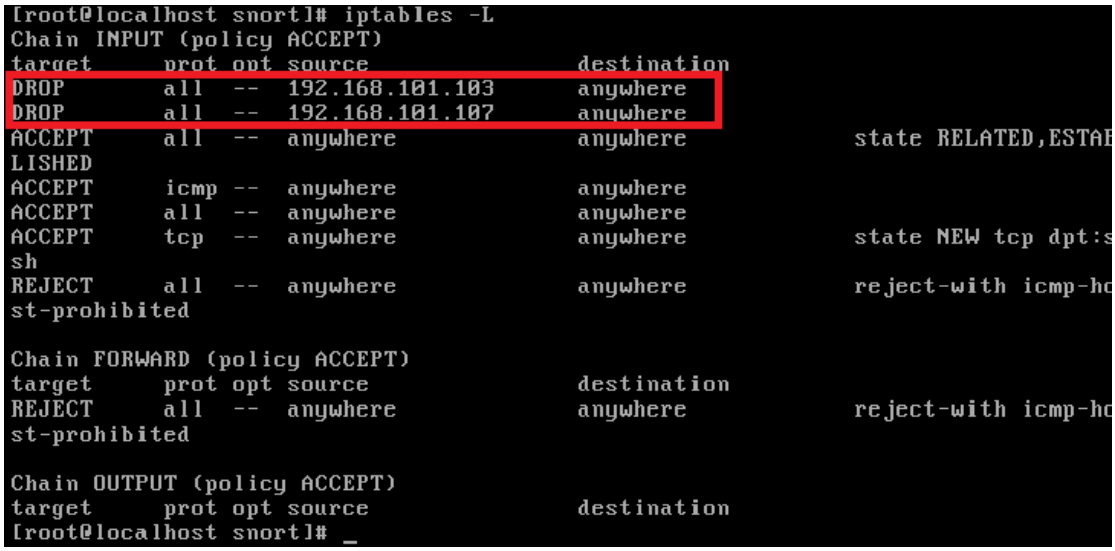
/usr/bin/perl /usr/local/bin/guardian.pl -c /etc/snort/guardian.conf

在 snort 主机上执行 iptables -L 观察是否有规则加入，有规则即实验成功。



可以看到，执行了 block 脚本

在 snort 主机上执行，iptables -L 即可看到新加入的规则。



至此，单台防火墙联动已经成功实现。

扩展思考——Snort 与其他主机或路由器联动

这里给出一些思路，如果您对此感兴趣可以自行去实验。

思路一：

将 snort 安装在拥有两个网卡的主机上，将网卡进行桥接使之处于透明模式，将 Snort 串联部署在网络出口路由器的后面，即可监控内网所有流量，实现联动本地防火墙实现 IPS 功能。

思路二：

写一个脚本实现如下功能，将告警日志里其他主机的告警，写一条对应的联动防火墙的规则脚本，将其存储在本地，使用脚本登录该主机后登录 snort 主机下载 并执行规则脚本，实现联动功能。此实现方式的弱点是无法达到及时响应的状态，且联动状况取决于外界因素过多。

实验报告要求

参考实验原理与相关介绍，完成实验任务，并对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

分析与思考

1、请查阅相关资料了解 snortsam 插件，分析并简述 snortsam 与 guardian 的优缺点。
2、你认为 snortsam 和 guardian 的这种联动方式有哪些弊端？