Сергей Яремчук

Агент системной безопасности

По данным институтов, занимающихся безопасностью (например, CERT, www.cert.org), число инцидентов в Интернете постоянно растет. Действительно, данные с описаниями взломов, а также готовые программы и эксплоиты сегодня найти довольно просто. Поэтому даже неопытный пользователь может вообразить себя хакером и попробовать в действии весь доступный арсенал.

Для обнаружения атак в состав Unix-систем включается целый ряд полезных приложений, имеющих определенные задачи и нацеленных на конкретный класс атак. Здесь и межсетевые экраны, закрывающие ненужные порты, и антивирусные программы, осуществляющие поиск вирусов, и сетевые сканеры, и сканеры безопасности, определяющие уязвимые участки сети, которые могут быть использованы злоумышленником для атаки. Пассивные атаки, цель которых — сбор информации без воздействия на работающие службы вроде перехвата трафика, определяются программами-антиснифферами (AntiSniff, Sentinel). Средства контроля целостности файловых систем (Tripwire, AIDE) и обнаружения закладок (chkrootkit, rkdet) позволяют безошибочно определить несанкционированное изменение важных системных файлов или установку suid/guid на пользовательские приложения. Всех программ не перечесть. Особое место в этом списке занимают IDS (Intrusion Detection Systems) – системы обнаружения атак и вторжений.

Системы IDS, в свою очередь, также подразделяются на классы. Например, сетевые системы обнаружения атак (Network Intrusion Detection System — NIDS) и «индивидуальные» системы host-based. Первые контролируют проходящий сетевой трафик, а вторые, анализируя данные на конкретном узле, пытаются обнаружить злонамеренные действия. В последнее время начали быстро развиваться так называемые гибридные IDS, сочетающие в себе возможности обоих типов – сетевых и узловых. Есть еще и Application Based IDS, выявляющие атаки, направленные на конкретные приложения. По типу определения атак системы IDS делятся на сигнатурные, работа которых подобно антивирусам заключается в поиске заранее известных признаков атаки, и системы, реагирующие на аномалии в контролируемой системе или сети. Имеются и подклассификации, например статистические и адаптивные системы, строящие профиль защищаемой среды по разным правилам. Кроме того, сегодня быстрыми темпами развиваются Intrusion Prevention System, способные не только обнаруживать, но и останавливать атаки. Каждый тип подобных систем имеет свои достоинства и недостатки, о них мы говорить не будем, классификация же приведена для того, чтобы была понятна сложность этой задачи,

а также показано конкретное место в общей системе обнаружения телекоммуникационных атак, которое занимает NIDS Snort (www.snort.org). Еще одно отступление, которое необходимо сделать: дословно IDS переводится как «система обнаружения вторжений», поэтому в литературе чаще всего используется именно этот термин, что, в принципе, не всегда правильно. Например, система IDS может обнаружить неудачную атаку, которая не привела к вторжению, поэтому более логичным будет использование термина «система обнаружения атак» — СОА.

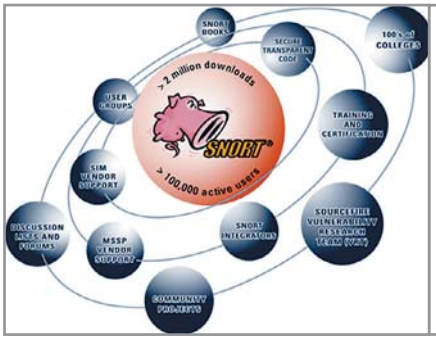
Возможности Snort

Первые упоминания о СОА относятся к 1980 году, а именно к публикации Джона Андерсона «Computer Security Threat Monitoring and Surveillance». Но активно развиваться это направление стало гораздо позже, в 90-х годах прошлого столетия.

О Snort впервые заговорили в 1998 году, когда ее основатель Мартин Роеш решил создать некие правила, которые могли бы логично описывать информацию перехваченных сетевых пакетов в созданном им пакетном снифферe. Snort очень быстро учился обнаружению атак, а Мартин постоянно совершенствовал свой продукт, привлекая все большее количество сторонников. Постепенно Snort обретал способность работать с фрагментированными пакетами, учился обрабатывать TCP-заголовки, SLIP- и PPP-пакеты, правила становились несколько неудобными в написании, но зато выигрывали по части читаемости, росло количество операционных систем, в которых можно было запустить Snort, появлялись препроцессоры.

Итак, Snort является сетевой системой обнаружения атак с открытым исходным кодом, способной выполнить в реальном времени анализ IP-пакетов, передаваемых на контролируемых интерфейсах, с целью обнаружения атак или попыток поиска уязвимостей. Snort обнаруживает атаки, комбинируя два метода — сигнатурный и анализ протоколов. Для описания событий, которые могут считаться злонамеренными или аномальными, используются гибкий язык правил и модульная система анализа. Кстати, сегодня существует два типа правил: официальные сертифицированные и строго протестированные Sourcefire VRT Certified Rules, распространяющиеся по лицензии VRT Certified Rules License Agreement, ограничивающей их коммерческое использование. Эти правила доступны в двух вариантах — для зарегистрированных и незарегистрированных пользователей. Регистрация абсолютно бесплатна. Все изменения в первую очередь распространяются по подписчикам (subscription release с буквой s в названии пакета), затем становятся доступными для зарегистрированных пользователей (без буквы s). Те же, кто не зарегистрировался, довольствуются статистическими правилами, обновляемыми только к выходу очередного релиза Snort и, естественно, отстающими от жизни (они имеют префикс pr в названии). Например, на момент написания статьи (середина ноября) эти правила датировались июлем 2005 года. Второй тип правил называется Community Rules. Они созданы добровольцами, но еще не прошли проверку и распространяются под лицензией GPL. Решать, использовать их или нет, вам.

Но это еще не все. Начиная с версии 2.3.0 RC1, в Snort включен код проекта Snort-inline, тем самым он получил возможность не только выявлять, но и останавливать предпринятую атаку, перестраивая правила iptables. И теперь Snort можно по праву назвать полноценной системой предотвращения атак.



Более 100 тысяч пользователей, более двух миллионов загрузок

Сегодня Snort может работать в четырех режимах:

- как пакетный сниффер, который подобно tcpdump, также используя libpcap, отлавливает пакеты в сети и выводит на экран информацию о них;
- как регистратор пакетов, записывающий данные на диск;
- как комплексная перестраиваемая система обнаружения атак, анализирующая сетевой трафик, следуя правилам, и выполняющая на их основе определенные действия;
- как система предотвращения атак, получающая вместо libpcap пакеты из iptables, способная отвергать или пропускать пакеты, основываясь на специфических правилах.

Всю собранную информацию детектор Snort позволяет сохранить в файлах журналов различных форматов (обычные ASCII, текстовые файлы или так называемые бинарные, совместимые с tcpdump). Кроме того, для удобства анализа всю собранную информацию можно занести в базу данных: Postgresql, MySQL, MS SQL Server, Oracle или unixODBC. Хотя, в принципе, можно использовать любую другую базу данных, не представленную в этом списке, но тогда таблицы придется формировать вручную. Часто в литературе Snort называют легкой IDS, подразумевая, что она предназначена для работы в сетях с небольшой нагрузкой. Это утверждение несколько устарело, учитывая хотя бы то, что сегодня даже домашние сети могут быть построены на гигабитных адаптерах, которые называть легкими язык уже не поворачивается. Система, построенная на датчиках Snort, способна собирать и обрабатывать информацию с нескольких сетевых локаций. Все в дело в производительности компьютеров, используемых в качестве сенсоров. Для того чтобы улучшить производительность, разделяя быструю работу IDS по захвату пакетов и относительно медленную по занесению информации, необходимо использовать Barnyard, который доступен на странице закачки проекта Snort. В этом случае Snort создает двоичный выходной, так называемый «унифицированный» формат, с которым в дальнейшем и работает Barnyard.

Установка Snort

На момент написания статьи актуальной была версия 2.4.3. В качестве варианта можно использовать последний снимок snort-snapshots-CURRENT.tar.gz. Хотелось бы отметить, что подкаталог contrib, содержащий различные дополнения к Snort, начиная с версии 2.2.0 пустует. Скрипты для создания баз данных переместились в подкаталог schemas, а скрипты для создания RPM-

пакетов — в одноименный подкаталог. Остальные же расширения можно найти на странице www.snort.org/dl/contrib. Все самое важное сказано, теперь можно начинать установку. Для выполнения большинства операций потребуются права root.

Распаковываем архив:

```
$ tar –xzvf snort-snapshots-CURRENT.tar.gz
$ cd HEAD
```

Внутри вы не найдете привычного конфигурационного скрипта, его необходимо создать:

```
$ ./autojunk.sh
```

Теперь конфигурируем. В самом простом случае скрипту никаких параметров передавать не надо. Если же необходимо использовать базу данных, то, например, для MySQL добавляем опцию --with-mysql, для включения режима остановки атак добавляем --enable-inline. Остальные опции можно просмотреть с помощью -help:

```
$ ./configure --with-mysql
```

Так как Snort является, по своей сути, сниффером, перехватывающим пакеты, то для компиляции требуется библиотека libpcap, которая обычно идет вместе с tcpdump. Поэтому, если во время конфигурирования вы получили такую ошибку:

```
ERROR! Libpcap header not found, go get it from
www.tcpdump.org/ or use the --with-libpcap-*
options, if you have it installed in an unusual place
```

Необходимо зайти на сайт www.tcpdump.org, скачать и затем установить библиотеку libpcap. Или другой пример. При конфигурировании с опцией --enable-inline я получил следующую ошибку:

```
checking "for libnet.h version 1.0.x"... /sw/include
./configure: line 1: dnet-config: command not found
./configure: line 1: dnet-config: command not found
checking dnet.h usability... no
checking dnet.h presence... no
checking for dnet.h... no
ERROR! Libdnet header not found, go get it from
http://libdnet.sourceforge.net or use the --with-dnet-*
options, if you have it installed in an unusual place
```

В принципе, в подсказке написано, что необходима библиотека libnet, но в этом случае я знал, что она у меня есть. Поэтому я просто нашел, где лежит нужный заголовочный файл:

```
# find /usr -name libnet.h
```

И добавил при конфигурировании параметр --with-dnet-includes=/usr/include/.

Также при установке в режиме inline вам обязательно потребуется наличие заголовочных файлов iptables. Если конфигуратор не может их найти, то, для того чтобы преодолеть эту проблему (если не помогают вышеописанные методы), необходимо скачать и распаковать исходные тексты iptables (www.iptables.org) и в образовавшемся каталоге ввести команду make install-devel. Когда конфигурация завершится без ошибок, нужно ввести традиционные make и make install и приступить к настройке.

Настройка Snort

Не знаю, с чем это связано, но все каталоги, необходимые для работы Snort, до сих пор приходится создавать вручную.

```
# mkdir /etc/snort
```

Сюда мы будем складывать все конфигурационные файлы и правила.

```
# mkdir /var/log/snort
```

А здесь будет вестись журнал работы.

Теперь в каталог /etc/snort копируем все, что лежит в подкаталоге etc дистрибутива.

```
# cp –R /home/sergej/work/HEAD/etc/* /etc/snort/
```

Далее распаковываем файл правил и помещаем их в /etc/snort/rules. В принципе, место для них можно выбрать любое, но так удобнее, к тому же это считается традиционным:

```
# tar –xzvf snortrules-snapshot-CURRENT.tar.gz
# mv rules /etc/snort
```

Рассмотрим одно правило, для того чтобы стало ясно, как они пишутся. Например, одно-единственное правило в файле virus.rules содержит такую запись:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET 25
(msg:"VIRUS OUTBOUND bad file attachment"; flow:to_serv-
er,established; content:"Content-Disposition|3A|"; nocase;
pcre:"/filename\s*=\s*.*?\.(?=[abcdehijlmnoprsvwx])(a(d[ep]|s[
dfx])|c([ho]m|li|md|pp)|d(iz|ll|ot)|e(m[fl]|xe)|h(lp|sq|ta)|jse?|m(d[abe
w]|s[ip])|p(p[st]|if|[lm]|ot)|r(eg|tf)|s(cr|[hy]s|wf)|v(b[es]?|cf|xd)|w(m
[dfsz]|p[dmsz]|s[cfh])|xl[tw]|bat|ini|lnk|nws|ocx)[\x27\x22\n\r\s]/
iR"; classtype:suspicious-filename-detect; sid:721; rev:8;)
```

Несмотря на довольно внушительный вид, правило очень простое, и если разобрать его по частям, то все становится на свои места. Первая строка говорит о том, что все сообщения по протоколу TCP, направленные из домашней сети с любого порта во внешнюю сеть на порт 25 (что говорит о почтовых сообщениях), имеющие в своем составе присоединенный файл с определенными расширениями, определяются как подозрительные. Директива alert указывает на действия, которые должен производить Snort при обнаружении пакета, попадающего под это правило. По умолчанию имеется пять действий: alert, log, pass, activate и dynamic. Кроме того, в режиме inline доступны еще три — drop, reject и sdrop. В некоторых случаях в состав правила включены комментарии (reference), позволяющие найти более подробную информацию об уязвимости на специальных ресурсах. В нашем же случае таких указателей нет, так как это правило можно отнести к общим, но в самом файле содержится подробное объяснение. Правило может быть односторонним (->) и двусторонним (< >), когда направление движения пакета роли не играет. Также в правиле может использоваться директива priority, указывающая на приоритет. При появлении событий с определенным приоритетом они могут быть обработаны сторонними утилитами, такими как swatch (Simple Watcher) или syslog-ng (syslog-next generation), и выполнять какие-либо действия, например отправлять e-mail.

Файл конфигурации snort.conf

И последний шаг, который осталось сделать, — отредактировать конфигурационный файл /etc/snort/snort.conf. В дистрибутиве уже имеется готовый шаблон, поэтому с нуля его писать не придется. В файле используются переменные, в том числе встречающиеся и в правилах. Это довольно удобно — при смене какого-либо параметра не придется его переписывать несколько раз. Кроме того, некоторые опции вынесены во внеш-

ние файлы, которые подключаются директивой include с именем файла. Все параметры щедро снабжены комментариями, которые традиционно начинаются со знака решетки. Для удобства восприятия файл разбит на пять частей:

- установка переменных сети;
- настройка препроцессоров;
- настройка вывода информации;
- установка дополнительных директив;
- модификация правил.

Для нормальной работы достаточно настроить первые три пункта, остальные можно пока не трогать.

Переменная HOME_NET определяет IP-адреса, которые Snort будет считать адресами домашней сети. Возможно задание отдельного адреса или диапазона. Если требуется указать несколько адресов, они перечисляются через запятую. Ключевое слово any означает любой адрес. Например:

```
var HOME_NET 10.1.1.0/24
var HOME_NET [10.1.1.0/24,192.168.1.0/24]
```

Переменная EXTERNAL_NET указывает на внешние узлы. По умолчанию выставлено значение any. Его можно оставить как есть, а более логичным будет указать, что все, не являющееся домом, будет внешним:

```
var EXTERNAL_NET !$HOME_NET
```

Ниже в файле идет список серверов (DNS, SMTP, web, SQL, telnet и SNMP), используемых в сети. Можно оставить как есть, то есть $HOME_NET, или указать конкретный IP-адрес, но, с другой стороны, если у вас нет веб-сервера, то зачем отслеживать специфические для него атаки? Поэтому лишнее можно смело отключить. Далее задаются номера портов, используемых серверами. Это позволяет Snort не распылять ресурсы, а искать атаку более конкретно. Принцип тот же: если нет Oracle, то соответствующую строку лучше закомментировать. Обратите внимание, что номер порта может быть задан как единичный [80] и как непрерывный [80:8080]. Перечисление портов через запятую работать не будет (это обещается исправить в будущем). Поэтому, если веб-сервер использует два порта, необходимо написать следующее:

```
var HTTP_PORTS 80
var HTTP_PORTS 8080
```

Также обратите внимание, что переменную RULE_PATH нужно определить как

```
var RULE_PATH rules
```

А не как

```
var RULE_PATH ../rules
```

Препроцессоры, подключаемые во второй секции «Configure preprocessors», — штука довольно серьезная и полезная, но требующая некоторого времени, для того чтобы разобраться с назначением и особенностями работы. Обратите внимание, что некоторые препроцессоры дублируют друг друга, поэтому включать все сразу также не имеет смысла. Так, вместо Portscan и Flow-Portscan разработчики рекомендуют использовать sfPortscan, разработанный в Sourcefire и предназначенный для тех же целей, то есть для определения сканирования портов. Более быстрый в работе модуль Frag3, предназначенный для дефрагментации IP-пакетов, пришел на замену устаревшему Frag2. Кроме того, некоторые препроцессоры направлены на определение аномалий в работе определенных сервисов. Так, X-Link2State предназначен для определения уязвимости в Microsoft Exchange Server, HTTPInspect изучает аномалии в HTTP-трафике.

В третьей секции «Configure output plugins», как уже говорилось, настраиваются выходные параметры. В общем случае строка параметров имеет такой вид:

```
output <name_of_plugin>: <configuration_options>
```

В настоящее время Snort может использовать десять плагинов для вывода информации (каждый из которых имеет дополнительные опции):

- alert_syslog – для вывода информации используется демон syslog; модуль настраивает приоритеты сообщений и уровень;
- alert_fast – информация о возможной атаке выводится в указанный в качестве дополнительного параметра файл в сокращенном формате, без подробностей;
- alert_full – модуль, подходящий для небольших сетей, так как сильно затормаживает работу Snort; заголовок пакета выводится полностью, в лог-каталоге будет создан подкаталог, в который по каждому IP будут записываться пакеты, вызвавшие предупреждение;
- alert_unixsock – похож на предыдущий, только информация в реальном времени передается в Unix-сокет, откуда может быть считана любой другой программой;
- log_tcpdump — записывает в указанный файл (к его имени будет добавляться метка времени, поэтому затереть его при перезапуске не получится) перехваченные пакеты в формате утилиты tcpdump;
- database – модуль, позволяющий заносить информацию в базу данных;
- csv – вывод в файл формата csv, который может быть использован для занесения информации в базу данных; кроме имени файла необходимо перечислить параметры, которые в него заносятся;
- unified – выводит данные в специальном формате, оптимизированном для обработки внешними утилитами, которые затем будут заниматься регистрацией события; в универсальном формате используются два файла — alert_unified и log_unified, первый содержит IP-адрес, порт, протокол, ID сообщения, во второй записывается дамп пакета;
- alert_prelude – доступен при конфигурировании с опцией --enable-prelude, в этом случае Snort используется как датчик гибридной IDS Prelude (www.prelude-ids.org);
- log_null – в этом случае Snort способен реагировать на указанные предупреждения без регистрации пакетов.

Готовые примеры имеются в файле. Для работы их достаточно раскомментировать, поэтому останавливаться на них мы не будем. Можно создать и собственные правила и привязать к ним определенный тип действий. Например, изменив alert на suspicious в приведенном выше правиле и записав следующее действие, подозрительные пакеты можно захватывать в формат tcpdump:

```
ruletype suspicious
{
type log
output log_tcpdump: suspicious.log
}
```

В конце секции с помощью директивы include подключаем файл classification.config, содержащий описание классификаций и приоритетов атак, а также файл reference.config, содержащий URL для сообщений об обнаруженных уязвимостях.

И, наконец, в конце файла вы найдете секцию «Customize your rule set», в которой необходимо убрать комментарии, указывающие на файлы с правилами:

```
include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
....
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
include $RULE_PATH/experimental.rules
```

Названия правил говорят сами за себя. Оставьте то, что действительно нужно (если сомневаетесь, лучше включите все). По умолчанию файл local.rules пуст, в него заносит свои правила сам пользователь. В качестве варианта можно создать отдельный файл, вынести в него необходимые параметры и подключить с помощью директивы include.

Запуск Snort

После того как все готово, можно запускать Snort. Для работы в режиме сниффера Snort запускается с флагом -v. При этом на экран будут выводиться заголовки пакетов. Если же вы хотите видеть и данные, используйте команду:

```
# snort –vd
*** interface device lookup found: eth0
Running in packet dump mode
Initializing Network Interface eth0
      --== Initializing Snort ==--
Initializing Output Plugins!
Decoding Ethernet on interface eth0
      --== Initialization Complete ==--
  ,,_     -*> Snort! <*-
 o"  )~   Version current (Build 29)
  ''''    By Martin Roesch & The Snort Team:
http://www.snort.org/team.html
        (C) Copyright 1998-2005 Sourcefire Inc., et al.
11/14-18:06:25.638963 192.168.0.1:2291 -> 192.168.0.20:114
TCP TTL:128 TOS:0x0 ID:17112 IpLen:20 DgmLen:48 DF
***S* Seq: 0x3AB82ED6  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
11/14-18:06:25.639014 192.168.0.20:114 -> 192.168.0.1:2291
TCP TTL:64 TOS:0x0 ID:2523 IpLen:20 DgmLen:40 DF
***A*R** Seq: 0x0  Ack: 0x3AB82E
Snort received 1714 packets
   Analyzed: 1157(67.503%)
   Dropped: 557(32.497%)
Breakdown by protocol:
    TCP: 451       (26.313%)
   ICMP: 71        (4.142%)
ETHLOOP: 0         (0.000%)
DISCARD: 0         (0.000%)
Action Stats:
ALERTS: 0
LOGGED: 0
PASSED: 0
```

Если в системе один интерфейс, то программа сама разберется, с чем ей работать. В противном случае его требуется указать при помощи -i:

```
# snort –vd -i eth0
```

Можно указать на конкретную информацию, которую требуется захватить. Например, устанавливаем в качестве домашней сети 192.168.1.0 и захватываем пакеты с узла 192.168.1.1:

```
# snort -h 192.168.1.0/24 -d -v host 192.168.1.1
```

Для регистрации пакетов в общем случае указываем каталог, в который надо записывать информацию:

```
# snort -l ./log
```

Если на выходе требуется файл формата tcpdump, то добавляем параметр -b.

И, наконец, работа в режиме обнаружения атак. Так как файл snort.conf уже создан, то поступаем просто:

```
# snort -c /etc/snort/snort.conf
```

Для тестирования набираем ping -s 65507. После чего, если выбран соответствующий режим ведения журнала, в каталоге /var/log/snort появится файл с предупреждением о потенциально зловредном пакете:

```
[**] [1:499:3] ICMP Large ICMP Packet [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
15/11-18:21:2.1131991802 192.168.0.1 -> 192.168.0.20
ICMP TTL:255 TOS:0x0 ID:18479 IpLen:20 DgmLen:63028
Type:0 Code:0 ID:512 Seq:19456 ECHO REPLY
[Xref => arachnids 246]
```

При более всестороннем тестировании COA следует использовать специальные утилиты вроде IDSwakeup (www.hsc.fr/ressources/outils/idswakeup/download).

Для автоматического запуска Snort при загрузке системы необходимо использовать скрипт snortd, который находится в подкаталоге RPM-дистрибутива. Копируем его в /etc/rc.d/init.d/ и даем команду:

```
# chkconfig snortd on
```

Snort создан для того, чтобы выполнять одну задачу — определение атак, и выполняет он ее хорошо. Анализ файлов журналов отдан на откуп сторонним разработчикам. Некоторые утилиты, предназначенные для этих целей, вы найдете на сайте проекта. Например, при помощи Perl-скрипта SnortALog (http://jeremy.chartier.free.fr/snortalog) можно отобрать необходимую информацию и вывести ее в удобочитаемом виде. Вот так можно вывести топ-лист атак, сгруппированный по времени, и отослать его по почте:

```
# cat /var/log/snort/snort.log.1131990681 | ./snortalog.pl
-hour_attack | /usr/sbin/sendmail -f admin@domain.com
```

Несмотря на то что описано уже многое, за кадром осталось еще немало вопросов, таких как работа с базой данных и автоматическое обновление правил. Почти ничего не сказано о средствах анализа собранной информации. Обо всем этом мы поговорим в следующий раз.