электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.056.53

Расширенная настройка межсетевого экрана Netfilter/Iptables для защиты от сетевых атак

Якимов М. А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Защита информации»

Научный руководитель: Старчак С.Л., д.т.н, профессор
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный консультант: Морозов Ю.В., к.т.н., начальник отдела,
НИИЦ (г. Москва) «ЦНИИ ВВКО» МО РФ, г. Москва, Россия
starchak@bmstu.ru

Классификация сетевых атак

Главная цель практически при любой атаке — получение несанкционированного доступа к информации. Существуют два принципиальных варианта получения информации: искажение и перехват. Вариант перехвата информации означает получение к ней доступа без возможности ее изменения. Возможность подмены информации следует понимать либо как полный контроль над потоком информации между объектами сети, либо возможность передачи различных сообщений от чужого имени. Атаки на хосты или сети можно классифицировать по следующим признакам.

По характеру воздействия

- Пассивное (атаки не оказывают прямого влияния на работу системы, но способны нарушить её политику безопасности, например прослушивать каналы связи в сети (Sniffing))
- Активное (атаки оказывают прямое влияние на работу самой системы, которое нарушает политику безопасности)

По цели воздействия

- нарушение функционирования системы (либо доступа к системе)
- нарушение целостности информационных ресурсов
- нарушение конфиденциальности информационных ресурсов

По наличию обратной связи с атакуемым объектом

- с обратной связью
- без обратной связи (ответы на запросы атакующему не нужны, например, при DoS-атаке)

По условию начала осуществления воздействия

- атака по запросу от атакуемого объекта (например, DNS и ARP-запросы)
- атака по наступлению ожидаемого события на атакуемом объекте (например, при прерывании сеанса работы с сервером)
- безусловная атака (атакующий является инициатором начала атаки, например, при DoS-атаке)

По расположению субъекта атаки относительно атакуемого объекта

- внутрисегментное (субъект и объект атаки располагаются, например, в общей локальной сети (актуально для прослушивания Sniffing)
- межсегментное

По уровню эталонной модели ISO/OSI, на котором осуществляется воздействие

- физический
- канальный
- сетевой
- транспортный
- сеансовый
- представительный
- прикладной

Общие сведения и принцип работы межсетевого экрана Netfilter/Iptables

Netfilter — межсетевой экран, встроенный в ядро Linux с версии 2.4 и управляемый утилитой Iptables (для протокола IPv6 используется Ip6tables). До Netfilter/Iptables существовал межсетевой экран Ipchains, который входил в состав ядер Linux 2.2. До Ipchains в Linux применялся межсетевой экран Ipfw (IPV4 firewall), перенесенный из BSD. Ipfw управлялся утилитой Ipfwadm. Проект Netfilter/Nptables был основан в 1998 году Расти Расселлом, который также руководил и прошлыми разработками. В 1999 году образовалась команда NetfilterCoreTeam (сокращено coreteam). В августе 2003 года руководителем coreteam стал Харальд Вельте (Harald Welte).

Проекты Ipchains и Ipfwadm изменяли работу стека протоколов ядра Linux, поскольку до появления Netfilter в архитектуре ядра не существовало возможностей для подключения дополнительных модулей управления пакетами. Iptables сохранил основную идею Ipfwadm — список правил, состоящих из критериев и действия, которое выполняется если пакет соответствует критериям.

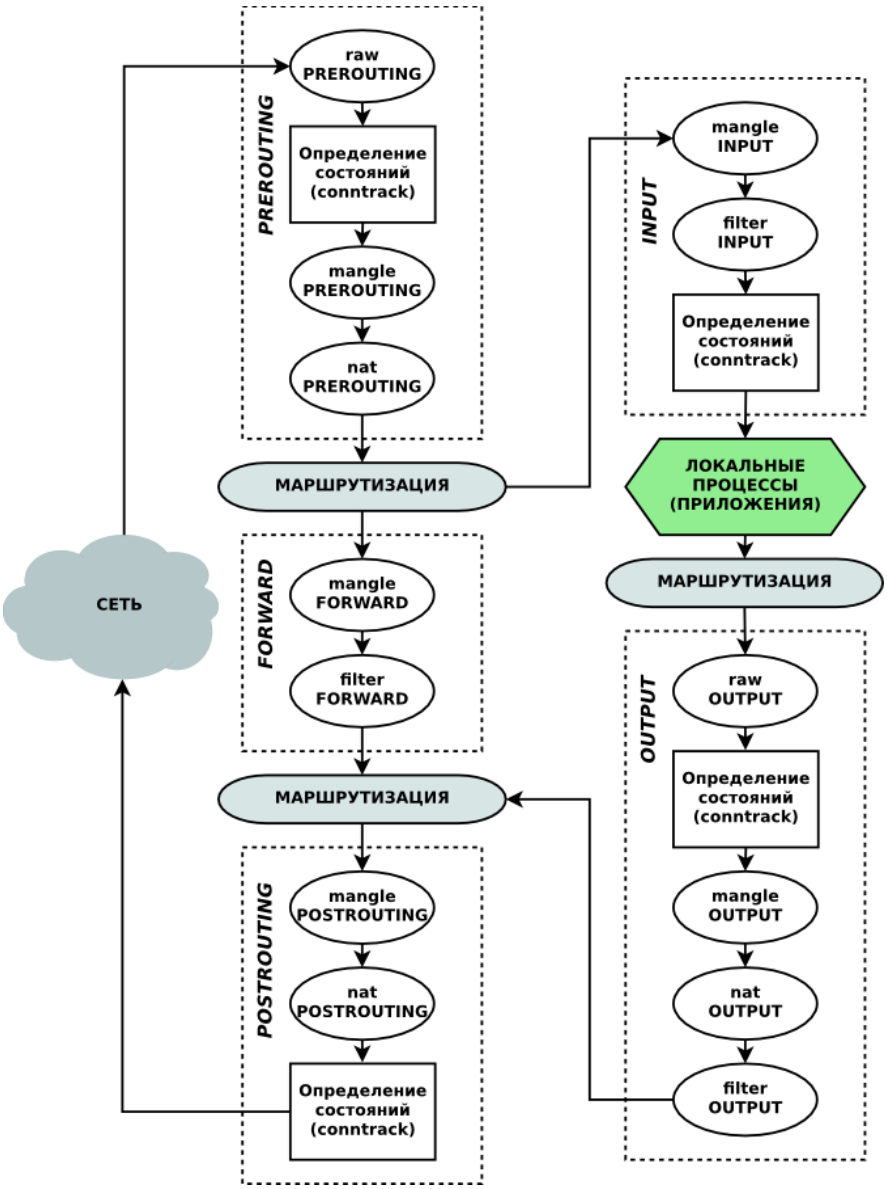
Цепочка — упорядоченная последовательность правил. Цепочки можно разделить на пользовательские и базовые.

Базовая цепочка — цепочка, создаваемая по умолчанию при инициализации таблицы. Каждый пакет должен пройти положенный ему набор базовых цепочек различных таблиц. Кроме того, базовая цепочка отличается от пользовательской наличием «действия по умолчанию» (default policy).

Пользовательская цепочка — цепочка, созданная пользователем. Может использоваться только в пределах своей таблицы.

Таблица — совокупность базовых и пользовательских цепочек, объединенных общим функциональным назначением.

Возможность создавать новые цепочки правил и переход пакетов между цепочками впервые была представлена в Ipchains, а в Iptables данная концепция была расширена до четырёх таблиц, разграничивающих цепочки правил по задачам: фильтрация, NAT и модификация пакетов. Также Iptables расширил возможности Linux в области определения состояний, позволяя создавать межсетевые экраны работающие на сеансовом уровне. На рисунке представлен в общем виде принцип работы межсетевого экрана Netfilter/Iptables.



Базовые цепочки Netfilter:

PREROUTING — для начальной обработки входящих пакетов

INPUT — для входящих пакетов, адресованных непосредственно локальному компьютеру

FORWARD — для проходящих (маршрутизируемых) пакетов

OUTPUT — для пакетов, создаваемых локальным компьютером

POSTROUTING— для окончательной обработки исходящих пакетов

Данные цепочки содержат следующие таблицы:

raw — пакет проходит данную таблицу до передачи системе определения состояний. Используется в основном для маркировки пакетов, которые не должны

обрабатываться системой определения состояний. Данная таблица содержится в цепочках PREROUTING и OUTPUT.

mangle — содержит правила модификации (обычно полей заголовка) IP-пакетов, поддерживает действия TTL, TOS и MARK. Данная таблица содержится во всех пяти стандартных цепочках.

nat — предназначена для подмены адреса отправителя или получателя. Данную таблицу проходят только первые пакеты из потока, так как подмена адреса отправителя или получателя применяются ко всем последующим пакетам в потоке автоматически. Данная таблица поддерживает действия DNAT, SNAT, MASQUERADE, REDIRECT и содержится в цепочках PREROUTING, OUTPUT и POSTROUTING.

filter — основная таблица (по умолчанию), которая используется для фильтрации пакетов. Данная таблица содержится в цепочках INPUT, FORWARD и OUTPUT.

Сетевые пакеты поступают в сетевой интерфейс, настроенный на стек TCP/IP и после некоторых простых проверок ядром (например, контрольная сумма) проходят последовательность цепочек. Вначале пакет обязательно проходит цепочку PREROUTING. После цепочки PREROUTING, в соответствии с таблицей маршрутизации, проверяется кому принадлежит пакет и, в зависимости от назначения пакета, определяется в какую цепочку он дальше попадет. Если пакет не адресован локальной системе, то он направляется в цепочку FORWARD, иначе, он направляется в цепочку INPUT и после прохождения данной цепочки отдается локальным процессам (демонам). После обработки локальной программой, при необходимости формируется ответ. Пакет отправляемый локальной системой в соответствии с правилами маршрутизации направляется на соответствующий хост из локальной сети или адрес маршрутизатора и направляется в цепочку OUTPUT. После цепочки OUTPUT (или цепочки FORWARD, если пакет был проходящим) пакет снова сверяется с правилами маршрутизации и отправляется в цепочку POSTROUTING.

Проходя через серию цепочек пакет последовательно проходит каждую таблицу в указанном порядке (см. рис. 1) и в каждой таблице последовательно сверяется с каждым набором критериев всех правил, и если пакет соответствует какому-либо критерию, то выполняется заданное действие над пакетом. При этом, в каждой таблице (кроме пользовательских) существует заданная по-умолчанию политика. Данная политика определяет действие над пакетом, в случае, если пакет не соответствует ни одному из правил в таблице. Чаще всего используется действие ACCEPT, чтобы принять пакет и

передать в следующую таблицу или действие DROP, чтобы отбросить пакет. В случае, если пакет не был отброшен, он завершает свое путешествие по ядру системы и отправляется в сетевую карту сетевой интерфейс, которая подходит по правилам маршрутизации.

Механизм определения состояний Conntrack (connection tracking)

Conntrack является частью пакетного фильтра и позволяет определить, к какому соединению/сеансу принадлежит пакет. Conntrack анализирует состояние всех пакетов, кроме тех, которые помечены как NOTRACK в таблице raw. На основе этого состояния определяется принадлежит пакет новому соединению (состояние NEW), уже установленному соединению (состояние ESTABLISHED), или дополнительному к уже существующему (RELATED), либо к неопределяемому соединению (состояние INVALID). Состояние пакета определяется на основе анализа заголовков передаваемого TCP-пакета. Модуль conntrack позволяет реализовать межсетевой экран сеансового уровня модели ISO/OSI. Для управления данным механизмом так же можно использовать параметр утилиты iptables «-m conntrack» или «-m state». Состояния текущих соединений conntrack хранит в ядре. Их можно просмотреть в файле «/proc/net/nf_conntrack» (или «/proc/net/ip_conntrack»).

Для расширения возможностей межсетевого экрана Netfilter/Iptables созданы утилиты nDPI и Suricata, реализующие функции накопления статистических данных, проверки и фильтрации сетевых пакетов по их содержимому на основе технологии Deep Packet Inspection. Кроме того, для более эффективного распределения нагрузки на Netfilter/Iptables существует стандартная утилита Ipset.

Стандартная утилита Ipset

Вследствие того, что каждому сетевому пакету, обрабатываемому ядром, приходится проходить сравнение по всему списку правил каждой цепочки, то при большом количестве правил возникает значительная нагрузка на систему.

Листинг №1

```
-A INPUT -s 10.1.2.3/32 -p tcp --dport 8080 -j DROP
```

Например, правило (см. Листинг №1) означает отбрасывание всех пакетов с IP-адреса 10.1.2.3 на порт 8080 локальной машины. Действие DROP выполняется далеко не всегда,

однако, сравнение IP-адреса источника будет проходить любой входящий на локальную машину пакет. Таким образом, при большом количестве подобных правил нагрузка существенно увеличивается. Кроме того, сокращение количества правил (с учётом изначально грамотной оптимизации) приведёт к уязвимостям в межсетевом экране.

Более эффективным способом является использование ветвлений правил. Принцип работы заключается в том, что однотипные правила группируются в отдельную цепочку, а в основной цепочке остается одно правило, которое перенаправляет пакет в отдельную цепочку в зависимости от какого-то общего признака пакетов. Например, имеются три правила (см. Листинг №2).

Листинг №2

```
-A INPUT -s 10.0.1.1/32 -p icmp -j ACCEPT
-A INPUT -s 10.0.1.2/32 -p icmp -j ACCEPT
-A INPUT -s 10.0.1.3/32 -p icmp -j ACCEPT
```

Данные правила объединяет одинаковый протокол — icmp. По этому признаку можно сгруппировать правила. Для этого необходимо создать новую цепочку PROT_ICMP, например « iptables -N PROT_ICMP».

Затем нужно создать правила, которые будут отправлять пакеты в эту цепочку (см. Листинг №3).

Листинг №3

```
-A PROT_ICMP -s 10.0.1.1/32 -j ACCEPT
-A PROT_ICMP -s 10.0.1.2/32 -j ACCEPT
-A PROT_ICMP -s 10.0.1.3/32 -j ACCEPT
```

Как можно заметить, протокол «-p» в этой цепочке уже не проверяется, так как отправляться в эту цепочку будут только ICMP пакеты. Далее необходимо отправлять все пакеты ICMP в эту цепочку (см. Листинг №4).

Листинг №4

```
-A INPUT -p icmp -g PROT_ICMP
```

Теперь входящий пакет (если он не ICMP) будет проходить лишь через одно правило вместо трех. Однако, данный способ не всегда реализуем. Например, имеются несколько однотипных правил, но сгруппировать их или сделать дополнительную цепочку не представляется возможным (см. Листинг №5).

Листинг №5

```
-A INPUT -s 10.1.1.1/32 -j DROP
```

```
-A INPUT -s 10.1.1.2/32 -j DROP
-A INPUT -s 10.1.1.3/32 -j DROP
```

В таких случаях, когда необходимо делать проверку по большому количеству IP-адресов и/или портов, рекомендуется воспользоваться стандартной утилитой Ipset. Сам модуль Ipset представляет из себя модуль ядра Ip_set, ряд вспомогательных библиотек и утилиту Ipset для задания параметров.

Для рассмотренного выше примера настройка осуществляется следующим образом. Сначала создается список (см. Листинг №6).

Листинг №6

```
ipset -N dropips iphash
```

В данном списке dropips — название списка, а iphash — тип списка. Типы списков можно посмотреть в manipset (например для работы с ip-адресами, подсетями, портами, mac-адресами). iphash служит для хранения IP-адресов, а использование хеширования предотвращает добавление в список дублирующих IP-адресов. Далее IP-адрес добавляется в список (см. Листинг №7).

Листинг №7

```
ipset -A dropips 10.1.1.1
```

Создается правило для использования списка (см. Листинг №8).

Листинг №8

```
iptables -A INPUT -m set --setdropipssrc -j DROP
```

«-m set» указывает на использование модуля Ipset, «--setdropips» указывает список IP-адресов, «src» указывает на то, что сверять нужно только IP-источника. Таким образом, будут отбрасываться все пакеты с IP-адресов, указанных в списке dropips.

Утилита nDPI

Межсетевые экраны, работающие на втором и третьем уровнях модели ISO/OSI, перестали удовлетворять современным требованиям обеспечения безопасности. Только проанализировав данные сеансового уровня и выше, можно действительно ограничить нежелательный трафик, а также обнаружить и блокировать вирусы. Данная технология накопления статистических данных, проверки и фильтрации сетевых пакетов по их содержимому называется Deep Packet Inspection (сокращённо DPI). Поэтому для межсетевого экрана Netfilter/Iptables была разработана утилита OpenDPI, распространяемая по лицензии LGPLv3, построенная на коде коммерческого продукта

PACE компании Ipoque. OpenDPI был модулем для Iptables и умел фильтровать большое количество различных типов пакетов. Однако проект прекратил свое существование. Но появилась компания Ntop, которая стала использовать исходники OpenDPI как основу для своего проекта, который получил название nDPI. Кроме того данный проект был переработан в модуль для Iptables, таким образом восстановив и дополнив проект OpenDPI новыми протоколами. Список доступных протоколов можно посмотреть, введя следующую команду (см. Листинг №9).

Листинг №9

```
iptables -m ndpi –help
```

В данный список входит большинство распространённых на сегодняшний день протоколов. Поэтому можно выбрать нужный протокол и создать на его основе правило в Iptables. Например, можно блокировать проходящие (маршрутизируемые) пакеты для программы Skype (см. Листинг №10).

Листинг №10

```
iptables -A FORWARD -m ndpi --skype -j DROP
```

Или использовать промаркированные пакеты Bittorrent для ограничения пропускной способности до 1 кБ/с (см. Листинг №11).

Листинг №11

```
iptables -A INPUT -m ndpi --bittorrent -j MARK --set-mark 1
```

Система обнаружения/предотвращения вторжений (IDS/IPS) Suricata

Система предотвращения вторжений (Intrusion Prevention System) — программная или аппаратная система сетевой и компьютерной безопасности, обнаруживающая вторжения или нарушения безопасности и автоматически защищающая от них. Системы IPS можно рассматривать как расширение Систем обнаружения вторжений (IDS), так как задача отслеживания атак остается одинаковой. Однако, они отличаются в том, что IPS должна отслеживать активность в реальном времени и быстро реализовывать действия по предотвращению атак. Возможные меры — блокировка потоков трафика в сети, сброс соединений, выдача сигналов оператору. Также IPS могут выполнять дефрагментацию пакетов, переупорядочивание пакетов TCP для защиты от пакетов с измененными SEQ и ACK номерами.

Suricata — open-source IPS/IDS система. Основана разработчиками, которые трудились над IPS версией Snort. Основное отличие Suricata от Snort — возможность использования GPU в режиме IDS, более продвинутая система IPS, многозадачность, и

как следствие высокая производительность, позволяющая обрабатывать трафик до 10Gbit на обычном оборудовании, и многое другое, в том числе полная поддержка формата правил Snort. Режим IDS позволяет проверять практически весь пакет трафика, так как нет строгого ограничения по времени обработки, а также для анализа используется GPU. Поэтому предлагается проверить работоспособность режима IPS, который теоретически имеет больше уязвимостей. В Suricata используется два режима IPS: NFQ и AF_PACKET.
NFQ IPS режим работает следующим образом:

1) Пакет попадает в Iptables
2) Правило Iptables направляет его в очередь NFQUEUE, для чего существует правило «iptables -I INPUT -p tcp -j NFQUEUE»
3) Из очереди NFQUEUE пакеты могут обрабатываться на уровне пользователя, что и делает Suricata
4) Suricata прогоняет пакеты по настроенным правилам и в зависимости от них может вынести один из трех вердиктов: NF_ACCEPT, NF_DROP и NF_REPEAT
5) Пакеты, попадающие в NF_REPEAT, могут быть промаркированы в системе, и направлены обратно в начало текущей таблицы Iptables, что дает огромный потенциал для влияния на дальнейшую судьбу пакетов с помощью правил Iptables.

Начиная с версии 1.4, Suricata умеет работать в качестве IPS, используя режим «zero copy» системы AF_PACKET, но с некоторыми ограничениями. Система должна работать в качестве шлюза с двумя сетевыми интерфейсами. Если пакет попадает под DROP правило, то он просто не пересылается на второй интерфейс. Основное преимущество режима «zero copy» — скорость обработки пакетов.

Пример настройки режима NFQ на WEB сервере

Сначала настраивается первоначальное правило Iptables, например, в очередь направляются пакеты, которые идут на 80-й порт и не попадают под маску «0x1/0x1» для исключения бесконечного цикла (см. Листинг №12).

Листинг №12

```
iptables -t mangle -I PREROUTING -p tcp -m tcp --dport 80 -m mark ! --mark 0x1/0x1 –j
NFQUEUE --queue-num 0
```

В данном примере используется таблица mangle, так как она является одной из первых на пути пакетов. Опция «--queue-bypass» позволяет пропускать все пакеты в очереди при отсутствии слушающего NFQUEUE приложения. Таким образом, если Suricata не запущена, то все пакеты, попадающие под правила, просто пойдут дальше. Опция «--

queue-num» задаёт номер очереди. «-m mark! --mark 0x1/0x1» игнорирует все пакеты, которые уже были обработаны Suricata. Далее файл конфигурации Suricata настраивается в режиме IPS (см. Листинг №13).

Листинг №13

```
nfq:
mode: repeat
repeat-mark: 1
repeat-mask: 1 ...
default-rule-path: /etc/suricata
rule-files: - test.rules
```

Правило Suricata, которое реагирует на текст TEST в пакете «/etc/suricata/test.rules» приведено в Листинге №14.

Листинг №14

```
pass tcp any any -> any any (content: "TEST"; msg: "TEST was marked!";
nfq_set_mark:0x2/0xffffffff; sid:2455);
```

Sid в данном случае должен быть уникальным. В совокупности с настройкой Suricata и правилом, маркировка и маска нежелательного пакета будут «0x02/0xfe» и «0xff XOR 0x01=0xfe». Затем следующей командой запускается Suricata (см. Листинг №15).

Листинг №15

```
suricata -q 0 -c /etc/suricata/suricata.yaml
```

Дальнейший разбор пакетов будет осуществляться следующими правилами iptables (см. Листинг №16).

Листинг №16

```
iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 80 -m mark --mark 0x2/0xfe -j
LOG --log-prefix "TEST packet detected"
```

После выполнения на удаленном клиенте команды «curl http://221.141.200.189/TEST» в логе «/var/log/syslog» появится запись (см. Листинг №17).

Листинг №17

```
Sep  9 14:23:06 server kernel: [ 2897.581561] TEST packet detectedIN=eth0 OUT=
MAC=c5:d5:08:8f:2d:be:ce:df:3e:af:8c:06:08:00 SRC=97.17.34.191
DST=221.141.200.189 LEN=133 TOS=0x00 PREC=0x00 TTL=64 ID=57685 DF PROTO=TCP
SPT=33949 DPT=80 WINDOW=115 RES=0x00 ACK PSH URGP=0 MARK=0x3
```

Также стоит заметить, что Suricata маркирует лишь пакеты. Чтобы правило сработало на всё соединение в целом, необходимо его промаркировать следующими командами (см. Листинг №18).

Листинг №18

```
iptables -t mangle -A PREROUTING -m mark --mark 0x2/0xfe -j CONNMARK --save-mark
iptables -t mangle -A PREROUTING -m connmark --mark 0x2/0xfe -j LOG --log-prefix
"TEST connection detected"
iptables -t mangle  -A PREROUTING -m connmark --mark 0x2/0xfe -j CONNMARK --
restore-mark
```

Также при помощи Suricata можно направлять разные типы трафика на разные адреса назначения, используя дополнение к Iptables в виде «RAW DNAT/SNAT». Но тогда возможна потеря целостности соединения. Однако, это можно решить, используя проксирующее программное обеспечение, которое сможет сразу восстанавливать соединения. Кроме того, Suricata умеет модифицировать пакеты (см. Листинг №19).

Листинг №19

```
pass tcp any any -> any any (content: "TEST"; replace:"SETS"; msg: "TEST was marked!";
nfq_set_mark:0x2/0xffffffff; sid:2455;)
```

Например, команда (см. Листинг №21) заменит в пакете текст «TEST» на «SETS», но при условии, что заменяющие данные должны быть точно такого же размера, что и оригинал. В данном случае команда «curl -v http://221.141.200.189/TEST» сохранит в лог WEB сервера следующую запись (см. Листинг №20).

Листинг №20

```
97.17.34.191 - - [09/Sep/2013:14:51:04 +0400] "GET /SETS HTTP/1.1" 200 151 "-"
"curl/7.26.0"
```

Пример настройки режима AF_PACKET на шлюзе

В данном случае конфигурация suricata.yaml должна выглядеть приблизительно следующим образом (см. Листинг №21).

Листинг №21

```
af-packet:
- interface: eth0
threads: 1
defrag: yes
```

cluster-type: cluster_flow

cluster-id: 98

copy-mode: ips

copy-iface: eth1

buffer-size: 64535

use-mmap: yes

- interface: eth1

threads: 1

cluster-id: 97

defrag: yes

cluster-type: cluster_flow

copy-mode: ips

copy-iface: eth0

buffer-size: 64535

use-mmap: yes

Количество потоков обработчика должно быть не более единицы для ядер старше 3.6, иначе увеличение количества потоков вызовет бесконечный цикл. MTU на обоих сетевых интерфейсах должно быть идентичным. Затем командой «suricata -c /etc/suricata/suricata.yaml --af-packet» запускается Suricata.

Анализ результатов тестирования и рекомендации по настройке утилит nDPI и Suricata

При тестировании межсетевого экрана Netfilter/Iptables с настроенными утилитами nDPI и Suricata применялась программа проверки уязвимостей Metasploit, а также программма-сниффер трафика Wireshark.

В результате тестирования данного комплекса можно сделать следующие выводы.

Использование утилиты nDPI совместно с Ipset позволяет блокировать, модифицировать (например, подменять поля заголовка) или маркировать (например, для дальнейшего ограничения пропускной способности) пакеты практически всех распространённых на сегодня протоколов и подпротоколов (например: skype, bittorrent, MySQL, Flash, Gmail и другие). Данное решение может применяться интернет провайдерами или системными администраторами компаний для распределения нагрузки на сеть, однако оно не обеспечивает должный уровень защиты. Они способны защитить от

некоторых простых атак, например: Bruteforce, IP-spoofing, Ping flooding, smurf, DNS-spoofing, WinNuke и т.д. Однако для профессиональных DoS-атак злоумышленники редко используют стандартные подпротоколы, а пакеты стараются тщательно замаскировать для усложнения работы межсетевого экрана. Кроме того данные утилиты не могут защитить от атак на более высоком уровне (уровне приложений).

Использование системы обнаружения/предотвращения вторжений (IDS/IPS) Suricata позволяет как детектировать атаки (в режиме IDS), так и отслеживать активность в реальном времени и быстро реализовывать действия по предотвращению атак (в режиме IPS). Режим IDS при грамотной настройке с использованием дополнительных программ (например: Barnyard2, Snortsam и другие) позволит детектировать многие распространённые DoS-атаки, атаки на уровне приложений. Однако в ходе тестирования было выявлено, что режим IPS не обеспечивает должный уровень защиты. Ниже приведена таблица тестирования на эксплоиты утилиты Suricata в режиме IPS.

Эксплоиты (Alert)	Количетсво (обнаруженые/пропущенные)
ms05_040_pnp	9/0
ms05_047_pnp	11/0
ms05_039_pnp	4/3
ms03_026_dcom	15/1
ms01_033_1dq	5/0
ms05_017_msmq	4/1

Кроме того, при тестировании Suricata в режиме IPS пропускает следующие shell-коды (см. Листинг №22 и Листинг №23).

SHELLCODE IRIX SGI + NOOP

Листинг №22

```
shellcode =  "\x30\x0b\xff\xff"          #/* andi   $t3,$zero,0xffff  */
shellcode += "\x24\x02\x04\x01"          #/* li     $v0,1024+1        */
shellcode += "\x20\x42\xff\xff"          #/* addi   $v0,$v0,-1        */
shellcode += "\x03\xff\xff\xcc"          #/* syscall                  */
shellcode += "\x30\x44\xff\xff"          #/* andi   $a0,$v0,0xffff  */
shellcode += "\x31\x65\xff\xff"          #/* andi   $a1,$t3,0xffff  */
shellcode += "\x24\x02\x04\x64"          #/* li     $v0,1124          */
```

```
shellcode += "\x03\xff\xff\xcc"          #/* syscall          */
shellcode += "\x24\x0f\x12\x34"          #NOOP
```

Cisco: Create a new VTY, allocate a password then set the privilege level to 15

Листинг №23

```
shellcode = "\x3c\x80\x81\x83"           # lis    4,vty_info@ha
shellcode += "\x38\x84\xda\x60"          # la     4,vty_info@l(4)
shellcode += "\x7d\x08\x42\x78"          # xor    8,8,8
shellcode += "\x7c\xe4\x40\x2e"          # lwzx   7,4,8
shellcode += "\x91\x07\x01\x74"          # stw    8,372(7)
shellcode += "\x39\x08\xff\xff"          # subi   8,8,1
shellcode += "\x38\xe7\x09\x1a"          # addi   7,7,233
shellcode += "\x91\x07\x04\xca"          # stw    8,1226(7)
shellcode += "\x7d\x03\x43\x78"          # mr     3,8
shellcode += "\x3c\x80\x80\xe4"          # lis    4,terminate@ha
shellcode += "\x38\x84\x08\x6c"          # la     4,terminate@l(4)
shellcode += "\x7c\x89\x03\xa6"          # mtctr  4
shellcode += "\x4e\x80\x04\x20"          # bctr
        # exits cleanly without adversely affecting the FTP server
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x61\x61\x61\x61"          # padding
shellcode += "\x80\x06\x23\xB8"          # return address
shellcode += "\x0d\x0a"
```

В ходе тестирования системы обнаружения/предотвращения вторжений (IDS/IPS) Suricata были выявлены некоторые уязвимости режима IPS, что связано, скорее всего, с недоработкой самого режима, так как он появился лишь в последней версии Suricata 1.4. Однако при тестировании режима IDS не было выявлено уязвимостей. Таким образом

рекомендуется основную фильтрацию производить в режиме IDS, а режим IPS использовать для «быстрой» фильтрации трафика.

В итоге, можно отметить, что стандартные возможности межсетевого экрана Netfilter/Iptables можно расширить набором дополнительных утилит, которые блокируют сетевые атаки на сеансовом уровне (модели ISO/OSI) и выше. Однако Netfilter/Iptables с любыми утилитами не способен обнаруживать прослушивание сетевых каналов (Sniffing), а также различные вирусы, передаваемые по сети.

Корректная настройка Netfilter/Iptables в сочетании с использованием дополнительных утилит nDPI и Suricata позволяют детектировать большинство сетевых атак, но не способны блокировать многие из них в реальном времени. Для повышения уровня защиты необходимо детектировать сетевые атаки в реальном времени.

Разработать эффективную утилиту, осуществляющая фильтрацию всех пакетов напрямую через себя, практически невозможно, так как при сканировании содержимого пакета на предмет различных атак затрачивается много времени и производственных мощностей.

Поэтому предлагается доработать существующий комплекс следующим способом. Утилита Suricata выдаёт специальные логи (обычно располагаются в директории «/var/log/suricata»), которые представляют собой текстовые файлы с описанием обнаруженных атак. Пример лога с описанием атаки (в данном случае был обнаружен shell-код) представлен в Листинге №25.

Листинг №25

```
ET SHELLCODE Possible TCP x86 JMP to CALL Shellcode Detected [**]
[Classification: Executable code was detected] [Priority: 3] {TCP}
ET SHELLCODE Possible TCP x86 JMP to CALL Shellcode Detected [**]
[Classification: Executable code was detected] [Priority: 3] {TCP}
GPL SHELLCODE x86 setuid 0 [**]
[Classification: A system call was detected] [Priority: 3] {TCP}
GPL SHELLCODE x86 setgid 0 [**]
[Classification: A system call was detected] [Priority: 3] {TCP}
```

На основании данных логов можно разработать утилиту, которая будет сначала анализировать логи Suricata, а потом – составлять правила для Netfilter/ Iptables. Данные правила будут содержать фильтры для обнаружения пакетов, которые могут относиться к обнаруженной атаке (для этого достаточно проверять заголовки пакетов). При

обнаружении пакетов, удовлетворяющих данным правилам, Netfilter/Iptables блокирует их (действие DROP).

Таким образом, можно блокировать многие атаки почти в реальном времени. Однако, если для осуществления атаки будет достаточно передать небольшое количество пакетов (например, при SQL-инъекции или PHP-инъекции), то Netfilter просто не успеет их заблокировать. То есть, пока Suricata будет анализировать первый пакет, Netfilter может пропустить остальные вредоносные пакеты (так как останавливать весь трафик нельхя, иначе скорость передачи трафика сильно упадёт). Поэтому данное усовершенствование комплекса позволит эффективно блокировать только атаки, состоящие из значительного числа пакетов (хотя подобных сетевых атак большинство, например, различные виды DoS-атак).

Список литературы

1. Огненный щит: Изучаем популярные надстройки для iptables. Режим доступа: http://www.xakep.ru/post/58089/ (дата обращения 05.03.2014).
2. Жуков Ю.В. Основы веб-хакинга. Нападение и защита; 2 изд. СПб.: Питер, 2012. 206 с.
3. Удаленные сетевые атаки. Режим доступа: http://ru.wikipedia.org/wiki/Удалённые_сетевые_атаки (дата обращения 05.03.2014).
4. Iptables. Режим доступа: (дата обращения 05.03.2014).
5. Suricata как IPS. Режим доступа: http://habrahabr.ru/post/192884/ (дата обращения 05.03.2014).
6. OpenDPI. Режим доступа: http://habrahabr.ru/post/108280/ (дата обращения 05.03.2014).
7. Deep packet inspection. Режим доступа: http://ru.wikipedia.org/wiki/Deep_packet_inspection (дата обращения 05.03.2014).