4. СИСТЕМЫ ОБНАРУЖЕНИЯ АТАК

Система обнаружения атак — это программный или программно-аппаратный комплекс, предназначенный для выявления и по возможности предупреждения действий, угрожающих безопасности информационной системы.

Первые прототипы СОА появились в начале 1980-х годов и были ориентированы в первую очередь на защиту автономных ЭВМ, не объединенных в сеть. Обнаружение атак производилось путем анализа журналов регистрации событий постфактум. Современные системы в основном ориентированы на защиту от угроз, направленных из сети, поэтому их архитектура существенным образом поменялась. Вместе с тем основные подходы к обнаружению атак остались прежними. Рассмотрим классификацию и принципы работы СОА более подробно.

4.1. Сигнатурный анализ и обнаружение аномалий

Основные подходы к обнаружению атак практически не изменились за последнюю четверть века, и, несмотря на громкие заявления разработчиков, можно с уверенностью утверждать, что концептуально обнаружение атак базируется либо на методах *сигнатурного анализа*, либо на методах *обнаружения аномалий*. Возможно также совместное использование указанных выше методов.

Сигнатурный анализ основан на предположении, что сценарий атаки известен и попытка ее реализации может быть обнаружена в журналах регистрации событий или путем анализа сетевого трафика. В идеале администратор информационной системы должен устранить все известные ему уязвимости. На практике, однако, данное требование может оказаться невыполнимым, так как в результате может существенным образом пострадать функциональность ИС. Не исключено также, что людские и материальные затраты, необходимые для устранения этих уязвимостей, могут превысить стоимость информации, обрабатываемой системой. Системы обнаружения атак, использующие методы сигнатурного анализа, предназначены для решения обозначенной проблемы, так как в большинстве случаев позволяют не только обнаружить, но и предотвратить реализацию атаки на начальной стадии ее выполнения.

Процесс обнаружения атак в данных системах сводится к поиску *заранее известной* последовательности событий или строки символов в упорядоченном во времени потоке информации. Механизм поиска определяется способом описания атаки.

Наиболее простым является описание атаки при помощи набора правил (условий). Применительно к анализу сетевых пакетов эти правила могут включать определенные значения отдельных полей заголовка пакета (IP-адрес и порт источника или получателя, установленные флаги, размер пакета

и т. д.). При анализе журналов регистрации событий правила могут ограничивать время регистрации пользователя в системе, количество попыток неправильного ввода пароля в течение короткого промежутка времени, а также наличие изменений в критических файлах системы. Таким образом, описание атаки отражает, во-первых, характер передаваемой информации и, во-вторых, совокупность реакций системы на реализацию атаки.

Если описать текущее состояние информационной системы совокупностью пар атрибут-значение, а события представить как действия, связанные с изменением этих атрибутов, то для описания атаки может использоваться теория конечных автоматов. В этом случае реализации каждой атаки соответствует последовательность переходов из «исходного» состояния системы в ее «конечное» состояние, характеризующее реализацию данной атаки. Условия и направление перехода определяются набором правил, как было описано выше. Такой подход к описанию сценария атаки является более точным, чем описание при помощи набора правил, так как позволяет учитывать динамику развития атаки и выявлять попытки реализации атак, скрытых в интенсивном потоке событий, сгенерированных злоумышленником для прикрытия своих истинных намерений.

Применение методов сигнатурного анализа требует от разработчика СОА выбора или создания специального языка, позволяющего описывать регистрируемые системой события, а также устанавливать соответствия между ними. Универсальность и полнота этого языка являются определяющими факторами эффективной работы системы обнаружения, так как в конечном счете на этом языке будут сформулированы правила, по которым выявляется атака.

Реагирование на попытку реализации атаки может включать как простое извещение администратора информационной системы, так и более активные меры: разрыв установленного соединения, отключение уязвимой службы, перепрограммирование межсетевого экрана на отклонение пакетов, полученных от выявленного системой злоумышленника, а также другие меры, препятствующие «успешному» завершению атаки.

Все СОА, использующие метод обнаружения атак по сигнатуре, имеют в своем составе базу данных известных атак (их сигнатур). Очевидно, что к принципиальным недостаткам рассматриваемого класса СОА относится невозможность обнаружения атак, сигнатуры которых отсутствуют в базе данных. Поэтому, чтобы обеспечить эффективную работу системы обнаружения, эта база должна регулярно обновляться. Обычно возможность обновления, в том числе автоматического, предусмотрена разработчиками системы. Преимуществами систем, использующих сигнатурный анализ, являются низкая вероятность «ложной тревоги» (ошибочного обнаружения атаки при ее фактическом отсутствии), а также относительная простота настройки.

Подход к обнаружению атак, основанный на попытке обнаружения аномального поведения системы, также был впервые предложен в 1980-х годах. Основной предпосылкой применения указанного подхода является то, что

в процессе «штатного» функционирования информационная система находится в некотором равновесном состоянии. Попытка реализации атаки ведет к выходу системы из этого состояния, причем факт выхода может быть зафиксирован. При создании СОА, работающих по принципу обнаружения аномалий, должны быть решены три задачи. Во-первых, необходимо разработать способ описания состояния информационной системы. Это нетривиальная задача, так как должна быть учтена как статическая, так и динамическая составляющие. Например, должны быть описаны типичные для системы потоки данных, передаваемых по сети. Во-вторых, необходимо разработать алгоритмы, при помощи которых будет автоматически (или с вмешательством администратора) составляться описание реальной работающей системы — ее «профиль». Это нужно для того, чтобы «научить» СОА различать штатный режим работы информационной системы. В-третьих, необходимо выбрать математические методы, которые будут использоваться для обнаружения аномалий в процессе функционирования системы. Другими словами, должны быть определены механизмы принятия решения о попытке атаки защищаемой системы. Очевидно, что используемые механизмы принятия решения в первую очередь зависят от того, как была описана система.

Рассмотрим простой пример. Пусть одним из параметров информационной системы является количество отклоненных входящих TCP-соединений, а точнее — среднее значение и дисперсия указанного параметра. В случае штатной работы системы количество отклоняемых соединений должно быть незначительным. Если злоумышленник начнет исследовать уязвимость системы при помощи сканирования портов, то есть попытается реализовать атаку «сканирование портов», количество отклоненных TCP-соединений резко возрастет. Такой скачок может быть обнаружен различными способами. Во-первых, может быть применен статистический критерий равенства средних значений двух случайных величин. Для его использования, правда, необходимо сделать два достаточно неоднозначных предположения: о нормальности распределений случайных величин и о равенстве их дисперсий. Во-вторых, что представляется более уместным, могут быть применены математические методы, известные под общим названием «методов обнаружения разладки». Эти методы специально разрабатывались для решения подобного класса задач, первоначально связанных с контролем систем слежения и управления. В-третьих, могут применяться математические методы распознавания образов. Известны также разработки, использующие нейросетевые методы анализа, однако о практическом внедрении этих методов в коммерческих программных продуктах до сих пор не сообщается.

Основным преимуществом использования подхода, основанного на обнаружении аномального поведения системы, является теоретическая возможность обнаружения новых, не описанных ранее, атак. Данная возможность основана на предположении, что атака по определению представляет собой набор действий, нехарактерных для штатного режима работы системы. На-

сколько эффективно будут выявляться новые атаки, определяется опять же способом описания состояния системы и количеством анализируемых параметров. Большинство математических методов обнаружения, используемых в рассматриваемом классе СОА, не являются детерминированными. Это означает, что все решения принимаются на основе статистического анализа и, следовательно, могут содержать ошибки. Возможны два класса ошибок: «пропуск атаки» и «ложная тревога». Вероятность пропуска определяется характером атаки и степенью адекватности описания текущего состояния системы. «Ложная тревога» может иметь место в том случае, если в информационной системе наблюдается нетипичная активность, являющаяся следствием действий законных пользователей (или процессов). Например, если на всех компьютерах локальной сети, имеющей подключение к Интернет, будет установлена антивирусная программа, запрограммированная на обновление антивирусных баз в одно и то же время каждые два дня, то одновременная попытка всех компьютеров подключиться к одному серверу Интернет будет интерпретироваться СОА как вирусная атака. Поэтому именно высокую вероятность «ложной тревоги» обычно называют главным недостатком систем обнаружения атак, основанных на обнаружении аномальной активности. Еще одним недостатком принято считать сложность настройки («обучения») системы, так как этот процесс требует от администратора глубоких знаний базовых принципов взаимодействия элементов информационной системы. В связи с этим полноценных коммерческих продуктов, использующих принципы обнаружения аномальной активности, на рынке практически нет, хотя разработки этих систем непрерывно ведутся ввиду их перспективности.

4.2. Обнаружение в реальном времени и отложенный анализ

По типу обрабатываемых данных системы обнаружения атак подразделяются на «системы реального времени» и «системы отложенной обработки». Системы отложенной обработки анализируют содержимое журналов регистрации событий или массив предварительно записанного трафика, а системы реального времени — входящий поток событий от программных датчиков. Очевидно, что адекватное реагирование на попытку реализации атаки, включая ее предотвращение, возможно только при использовании систем реального времени. В то же время это не означает, что СОА реального времени «лучше», чем системы отложенной обработки. Так, СОА реального времени, не имеющая функций по предотвращению атак, заведомо менее эффективна, чем аналогичная система с отложенной обработкой, поскольку в системе реального времени одним из основных критериев эффективности является простота используемых алгоритмов, а не их оптимальность с позиций надежности обнаружения атак. Поэтому выбор того или иного типа СОА должен делаться исходя из анализа задач, которые ставятся перед системой обнаружения.

Апостериорный анализ может использоваться со следующей целью:

− расследование информационных преступлений и инцидентов;
− выявление атак, не являющихся информационным преступлением (сбор информации об инфраструктуре сети, сканирование портов и пр.);
− сбор информации об уязвимостях информационной системы с целью их устранения;
− анализ активности отдельных пользователей;
− минимизация системных требований к СОА.

Основной целью использования систем обнаружения атак реального времени является быстрое реагирование на попытки реализации атак, в том числе пресечение этих попыток. В связи с этим типичными процедурами для данных систем является анализ и фильтрация трафика на сетевом и транспортном уровнях модели OSI. Чтобы сократить производительные затраты, часто рассматриваются лишь заголовки пакетов, а их содержимое «отбрасывается». Это, очевидно, значительно сокращает перечень обнаруживаемых атак.

4.3. Локальные и сетевые системы обнаружения атак

СОА, использующие информацию, получаемую от персонального компьютера, на который они установлены, обычно называют *локальными* (*host-based*). В противоположность им системы обнаружения, ориентированные на анализ всего доступного сетевого трафика, называют *сетевыми* (*network-based*).

Рассмотрим, какая информация может использоваться локальными СОА для выявления попыток атаки.

− Отслеживание попыток входящих и исходящих TCP- и UDP-соединений. В результате могут обнаруживаться и пресекаться попытки несанкционированных подключений к отдельным портам, а также попытки сканирования портов.
− Анализ входящего и исходящего сетевого трафика на предмет наличия «подозрительных» пакетов. «Досмотру» могут подлежать как поля заголовков пакетов, так и их содержимое.
− Отслеживание попыток регистрации на локальной ЭВМ. В случае интерактивной регистрации может накладываться ограничение на время регистрации, а в случае регистрации по сети можно ограничить перечень сетевых адресов, с которых разрешается вход в систему. Отдельное внимание уделяется множественным неудачным попыткам регистрации, которые могут иметь место в случае атаки «подбор пароля методом перебора».
− Отслеживание активности пользователей, наделенных повышенными полномочиями в системе (суперпользователь root в UNIX-системах, пользователи группы Администраторы в ОС Windows). Так как в широком классе случаев атака направлена на получение полномочий суперпользователя, могут

отслеживаться попытки регистрации этого пользователя в системе в неразрешенное время, попытки сетевой регистрации суперпользователя и т. д.

− Проверка целостности отдельных файлов или ключей реестра (для Windows-систем). Несанкционированные действия взломщика могут заключаться в попытках внесения изменений в базу данных пользователей или в модификации отдельных настроек системы. Контроль целостности критичных областей данных позволит СОА обнаружить попытку реализации атаки.

Сетевые СОА собирают и анализируют все доступные им сетевые пакеты на предмет наличия «подозрительного» содержимого или несанкционированных потоков информации от одного узла сети к другому. В связи с этим точка подключения СОА должна обеспечивать максимальный охват трафика, циркулирующего в сегменте сети. Обычно такие системы подключаются к специальному порту коммутатора либо устанавливаются непосредственно на маршрутизаторе сети. СОА данного класса гораздо эффективнее, чем локальные, способны обнаруживать факт сканирования портов, а также выявлять попытки атак на «отказ в обслуживании». Кроме того, если система обнаружения установлена на шлюзе, обеспечивающем доступ из локальной сети в Интернет, то путем фильтрации нежелательных пакетов может обеспечиваться защита этой локальной сети от внешних атак. Получается, что СОА выполняет в этом случае функции межсетевого экрана (либо управляет им). Таким образом, сетевые системы обнаружения атак находят свое применение в информационных системах, где установка специализированного программного обеспечения на компьютеры пользователей затруднительна, и там, где требуется изолировать сетевой сегмент от внешней угрозы. Необходимо отметить, что анализ интенсивного потока данных требует существенных вычислительных затрат, поэтому аппаратные требования к узлу, на котором устанавливается такая СОА, могут быть очень высокими. Наиболее критичной эта проблема становится при попытке защиты сети, содержащей несколько сотен компьютеров и имеющей выход в Интернет. К этому классу относятся большинство сетей крупных предприятий.

4.4. Распределенные системы обнаружения атак

Отдельным классом систем обнаружения атак являются распределенные системы. Их основным отличием является перераспределение функций сбора данных между несколькими «агентами» — программными датчиками, установленными на узлах информационной системы. Агенты могут собирать информацию непосредственно с компьютера, на который они установлены, или анализировать данные, передаваемые по сети. Наиболее принципиальным моментом при внедрении распределенных СОА является организация информационного обмена между отдельными агентами системы. Конечной целью этого обмена является принятие решения о факте атаки.

Преимуществом использования распределенных систем является возможность собирать и анализировать значительно больший объем информации, что позволяет обнаруживать широкий спектр атак. В этом отношении наиболее эффективным является решение, объединяющее методологию локальных и сетевых СОА в единое целое. С другой стороны, распределенные системы обладают рядом недостатков, наиболее существенным из которых является меньшая защищенность их компонентов. Во-первых, сбор данных с нескольких узлов создает дополнительную нагрузку на сеть, которая, в случае полномасштабной атаки, может превысить ее пропускную способность. Во-вторых, обмен информацией по сети подразумевает наличие открытых портов, потенциально доступных для атаки на отказ в обслуживании (переполнение очереди входящих TCP-соединений). В-третьих, на узлах информационной системы возможно внедрение вредоносного программного обеспечения, которое будет блокировать работу агента или пытаться подделывать информацию, им передаваемую.

Отдельной задачей, возникающей при проектировании распределенных СОА, является выбор идеологии процедуры принятия решения. Возможны несколько вариантов процедуры, отличающиеся степенью централизации. Наиболее простым является вариант процедуры с предельной степенью централизации, когда агенты занимаются лишь сбором данных и передачей их центральному узлу СОА — модулю принятия решения. Этот модуль анализирует поступающую информацию и выносит решение о факте атаки. Данный вариант характеризуется большим объемом передаваемых по сети данных, что повышает вероятность обнаружения факта работы СОА злоумышленником и делает систему уязвимой к атакам на отказ в обслуживании. Наиболее очевидным решением по использованию такого типа СОА является их установка в рамках подсетей небольшого размера, что позволит обойти проблему перегрузки сети пакетами, сгенерированными системой. Увеличение масштабов сети требует многоступенчатого подхода к принятию решения. Он заключается в том, что выделяются промежуточные модули принятия решения, которые собирают данные только с ограниченного числа «своих» агентов и передают на верхний уровень гораздо меньший объем информации, дополненный промежуточным решением. При любом варианте реализации процедуры принятия решения очевидно, что первоочередной становится задача обеспечения защищенности самой СОА.

4.5. Система обнаружения атак Snort

4.5.1. Общие сведения

Snort — это бесплатная система обнаружения атак с открытым исходным кодом, разработанная Мартином Рошем (Martin Roesch). Доступны версии программы, работающие под управлением операционных систем Win-

dows NT, Linux, BSD, Mac OS X, а также некоторых других. В соответствии с предложенной выше классификацией, Snort является сетевой СОА, основанной на сигнатурном анализе. Сигнатуры атак описываются при помощи *правил* — специальных синтаксических конструкций, позволяющих выявлять интересующую администратора информацию в полях заголовков и содержимом передаваемых по сети пакетов. Кроме того, в Snort реализовано несколько препроцессоров, выполняющих более сложные операции по анализу трафика, такие, например, как дефрагментация IP-пакетов, отслеживание TCP-соединений и выявление попыток сканирования портов.

4.5.2. Установка и запуск программы

Для обеспечения возможности перехвата сетевых пакетов программой Snort необходима предварительная установка и запуск службы WinPcap (WinPcap_3_1.exe).

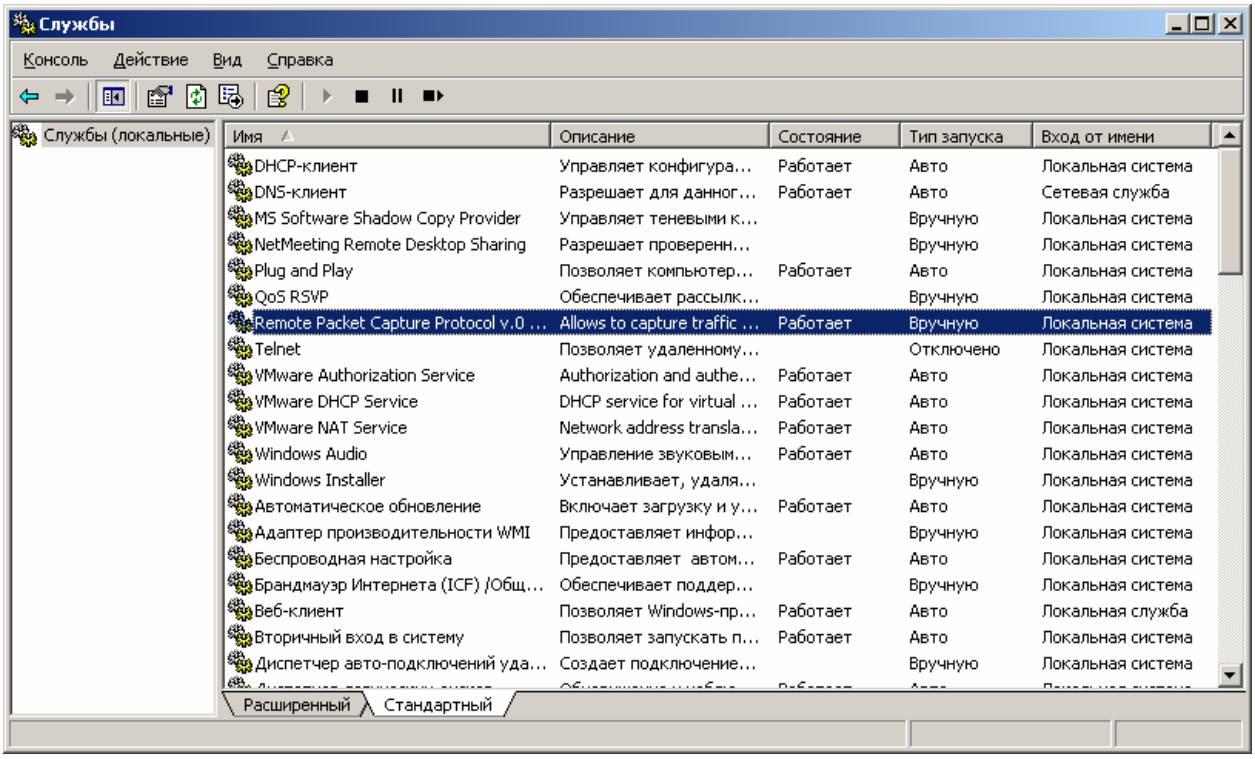

Рис. 4.1. Служба WinPcap в перечне служб

Для установки СОА Snort версии 2.4.3 необходимо запустить файл «Snort_243_Installer.exe» и ответить на интересующие программу-инсталлятор вопросы. По умолчанию Snort устанавливается в каталог «C:\snort», а его исполняемый файл располагается в каталоге «C:\snort\bin». Запуск СОА Snort осуществляется из командной строки, в табл. 4.1 приведен неполный список параметров, с которыми может производиться запуск. Чтобы вывести этот список на экран во время работы, необходимо выполнить команду

```
snort -?
```

Таблица 4.1

Список параметров СОА Snort

Параметр	**Описание**
`-c <rules>`	Использовать файл правил `<rules>`.
`-E`	Добавлять предупреждения (alerts) в журнал регистрации событий Windows NT (не создавая log-файла).
`-h <hn>`	Задать домашнюю сеть (home network) `<hn>`.
`-i <if>`	Подключиться к сетевому интерфейсу номер `<if>` (список интерфейсов можно получить при помощи команды `snort -W`).
`-I`	Добавлять к предупреждению наименование интерфейса.
`-K <mode>`	Режим регистрации предупреждений: `pcap` (по умолчанию) — в двоичном формате, `ascii` — в текстовом формате, `none` — без регистрации.
`-l <ld>`	Сохранять результаты регистрации в каталог `<ld>`.
`-L <file>`	Сохранять результаты регистрации в указанный файл формата tcpdump (будет располагаться в каталоге, предварительно указанном параметром `-l`).
`-n <cnt>`	Завершить работу программы после получения `<cnt>` пакетов.
`-N`	Не сохранять в файлах регистрации содержимого пакетов (сохраняется лишь текст предупреждений).
`-p`	Анализировать только пакеты, отправленные на локальный адрес, и широковещательные пакеты.
`-r <tf>`	Прочитать и обработать файл `<tf>`, записанный в формате tcpdump.
`-S <n=v>`	Установить значение переменной `n` файла правил, равной `v`.
`-U`	Записывать временные метки в универсальном скоординированном времени (UTC).
`-v`	Отображать на экране заголовки всех перехваченных пакетов.
`-V`	Показать версию Snort.
`-W`	Показать доступные сетевые интерфейсы.
`-X`	Сохранять в файле журнала регистрации событий содержимое перехваченных пакетов в «сыром» виде, начиная с уровня link модели OSI.

Параметр	Описание
`-y`	Добавить месяц, день и год в отображаемые и сохраняемые временные отметки.
`-?`	Показать помощь по параметрам командной строки.

Для проверки работоспособности СОА Snort рекомендуется выполнить следующие действия.

ВЫПОЛНИТЬ!

1. Установить СОА Snort.
2. Вывести на экран список доступных сетевых интерфейсов командой

   ```
   snort –W
   ```

3. Запустить Snort на выбранном интерфейсе в режиме анализатора пакетов с выводом информации на экран, указав программе завершить работу после приема третьего пакета:

   ```
   snort –v –i 1 –n 3
   ```

4. Выполнить любые действия, которые приведут к отправке или приему сетевых пакетов (например, отправить эхо-запрос на любой IP-адрес командой ping). Убедиться, что пакеты перехватываются и отображаются на экране.

4.5.3. Описание языка правил

Рассмотрим краткое описание языка правил, на котором задаются сигнатуры атак, обнаруживаемых СОА Snort. Полное описание языка правил содержится в файле документации «c:\snort\doc\snort_manual.pdf»

Правила записываются в одну строку, если возникает необходимость перенести текст правила на следующую строку, необходимо добавить в конце строки символ обратной косой черты «\».

Правила состоят из двух частей: заголовка и набора атрибутов. Заголовок, в свою очередь, состоит из:

1. Указания действия, которое необходимо выполнить (`alert`, `log`, `pass` и др.).
2. Протокола (`tcp`, `udp`, `icmp`, `ip`).
3. IP-адреса и маски подсети источника и приемника информации, а также информации о портах источника и приемника.

Действие `alert` заключается в генерации предупреждающего события и сохранении содержимого пакета для дальнейшего анализа. Действие `log` предполагает сохранение пакета без генерации предупреждения. Действие

pass означает пропуск пакета (его игнорирование). Существует также ряд более сложных действий, которые здесь не рассматриваются.

Текст атрибутов располагается в скобках, каждая пара атрибут — значение имеет вид *<атрибут>: <значение>;*. Значения строковых атрибутов записываются в кавычках.

Рассмотрим пример простого правила (одна строка):

```
alert <протокол> <адрес_подсети1>[/маска_подсети1]
<порт1> <направление> <адрес_подсети2>[/маска_подсети2]
<порт2> ([msg:"Текст сообщения";] [другие_атрибуты])
```

где

- alert — действие, которое необходимо выполнить при обнаружении пакета, удовлетворяющего данному правилу, и которое заключается в генерации «предупреждения» — записи в журнале регистрации
- <протокол> — наименование протокола (tcp, udp, icmp, ip)
- <адрес_подсети>[/маска_подсети] — IP-адрес и маска подсети, либо IP-адрес узла участника обмена в формате: 192.168.247.0/24, либо 192.168.247.1
- <порт> — номер порта либо диапазон портов в формате 1:1024 для обозначения диапазона портов от 1 до 1024, 1024: — с номерами больше или равными 1024, или :1024 — меньше или равными 1024 соответственно
- <направление> — обозначение направления в виде ->, <- или <>

Вместо IP-адресов и номеров портов могут использоваться псевдонимы any, являющиеся заменителем любого значения.

Атрибуты являются наиболее значимой частью правил, так как позволяют искать интересующую информацию в полях заголовков и содержимом пакетов. Существует четыре категории атрибутов:

- meta-data — предоставляют информацию о правиле, но не влияют на процесс обнаружения;
- payload — атрибуты данного типа предназначены для поиска информации в «полезной нагрузке» (содержимом) пакета;
- non-payload — предназначены для поиска информации в заголовках пакетов;
- post-detection — определяют поведение системы после обнаружения пакета, удовлетворяющего правилу.

В табл. 4.2 приведен неполный список атрибутов, которые могут быть использованы при написании правил.

Если известно местонахождение интересующей информации в пакете, то целесообразно ограничить область поиска при помощи модификаторов offset и depth, так как это существенно сократит время, затрачиваемое на анализ пакета.

Таблица 4.2

Список атрибутов COA Snort

Атрибут	Описание
meta-data	
`msg: "<текст>";`	Сообщение, которое добавляется в журнал регистрации при активации правила.
`sid: <идентификатор>;`	Уникальный номер, используемый для идентификации правил. Идентификаторы от 100 до 1 000 000 используются для правил, включенных в дистрибутив Snort. Для локальных правил следует использовать значения больше 1 000 000.
`rev: <номер_редакции>;`	Целое число, служащее для обозначения номера редакции правила.
`classtype: <имя_класса>;`	Используется для обозначения класса атаки. Полный список классов приведен в документации по Snort.
`priority: <приоритет>;`	Целое число, используемое для переопределения приоритета, задаваемого указанным ранее классом атаки, или для назначения приоритета новому правилу. Наивысший приоритет — 1, типичное значение атрибута составляет от 1 до 4.
payload	
`content: [!]"<строка>";`	Позволяет искать заданную подстроку в содержимом полезной нагрузки пакета. Восклицательный знак означает отсутствие указанной информации в пакете. По умолчанию данный атрибут является чувствительным к регистру. Для обозначения двоичных данных следует использовать шестнадцатеричные значения, отделенные вертикальными чертами: `|00 5C|`. Атрибут `content` имеет несколько модификаторов, которые могут располагаться следом за ним.
`nocase;`	Модифицирует стоящий ранее атрибут `content`, делая его нечувствительным к регистру.
`depth: <число_байт>;`	Значение атрибута (в байтах) определяет, как далеко в полезной нагрузке пакета должен производиться поиск.

Атрибут	Описание
`offset: <число_байт>;`	Значение атрибута определяет, сколько байтов полезной нагрузки следует пропустить. Поиск будет вестись, начиная с `число_байт+1`-го байта.
`distance: <число_байт>;`	Атрибут похож на `depth`, но указывает, сколько байт необходимо пропустить после предыдущей совпавшей подстроки перед продолжением поиска.
`within: <число_байт>;`	Атрибут указывает системе искать совпадения лишь в первых `число_байт`, начиная с конца предыдущей совпавшей подстроки.
non-payload	
`dsize: <размер>;`	Сравнить размер полезной нагрузки с заданным. Возможно указание диапазонов значений с использованием знаков >, < и <>. Например: `>128`, `300<>500` (от 300 до 500).
`flags: [!|*|+]<флаги>`	Проверить, установлены ли указанные флаги в принятом TCP-пакете. Флаги записываются подряд без пробелов и обозначаются следующим образом: `F` — FIN (LSB в байте флагов) `S` — SYN `R` — RST `P` — PSH `A` — ACK `U` — URG `1` — Резерв 1 (MSB в байте флагов) `2` — Резерв 2 `0` — флаги не установлены Могут быть дополнительно использованы следующие модификаторы: ! — указанные флаги не установлены * — установлен хотя бы один из указанных + — установлены указанные и любые другие
`itype: <тип>;`	Сравнить тип ICMP сообщения с указанным. Возможно указание диапазонов значений с использованием знаков >, < и <> (см. выше).
`icode: <код>;`	Сравнить код ICMP сообщения с указанным.

Вместо IP-адресов могут использоваться переменные, заданные выше по тексту следующим образом:

```
var <имя_переменной> <значение_переменной>
```

Чтобы сослаться на переменную далее в тексте, перед ее именем следует поставить знак доллара `$`.

В текст файла правил можно включать комментарии, которые отделяются знаком `#`. Вся информация справа от этого знака и до конца строки считается комментарием и не интерпретируется системой:

```
#<комментарий>
```

Рассмотрим пример задания двух переменных последующего их использования в правиле, фильтрующем входящие ICMP-пакеты ECHO (тип 8):

```
# Глобальные переменные
var HOME_NET 192.168.247.1
var EXTERNAL_NET !$HOME_NET

# Обнаружение эхо-запросов (ping'ов)
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"Incoming ECHO REQUEST"; itype: 8;)
```

4.5.4. Использование СОА Snort

СОА Snort можно использовать как анализатор трафика, обладающий значительными возможностями по фильтрации пакетов. Например, можно создать файл с правилами, использующими исключительно действия типа `log`. В результате из входящего потока данных будут отобраны и сохранены пакеты, удовлетворяющие указанным правилам. Так как по умолчанию журнал ведется в двоичном формате tcpdump, он может быть импортирован почти всеми специализированными программами анализа трафика. Обычно эти программы позволяют наглядно отображать содержимое пакетов, но не обладают такими возможностями по их фильтрации, как Snort.

Для запуска Snort в режиме анализатора трафика, как и для запуска его в режиме системы обнаружения атак, необходимо выполнить следующую команду в командной строке Windows:

```
snort -i <интерфейс> -c <файл_конфигурации>
-l <путь_к_журналу>
```

где

`<интерфейс>` — номер интерфейса, полученный в результате выполнения команды `snort –W`

<файл_конфигурации> — путь к файлу, в котором хранятся настройки программы и правила обнаружения

<путь_к_журналу> — путь к каталогу, в котором необходимо сохранить файл журнала

Пример:

```
snort -i 3 -c ../etc/my.conf -l ../log
```

Следует обратить внимание, что при записи пути используются не обратные, а прямые «косые черты».

Для завершения работы СОА Snort, необходимо нажать клавиши <Ctrl+C>.

Рассмотрим несколько правил (табл. 4.3), которые позволят обнаруживать атаки, описанные в разделе 2. Текст правил должен записываться в одну строку.

Таблица 4.3

Примеры правил СОА Snort

№	Описание	Правило
1	Обнаружение входящих ECHO-запросов (ping'ов)	`alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "Incoming ECHO REQUEST"; itype: 8;)`
2	Обнаружение исходящих ECHO-ответов	`alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg: "Outgoing ECHO REPLY"; itype: 0;)`
3	Обнаружение больших ICMP-пакетов (атака «Ping of Death»)	`alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg: "Incoming large ICMP packet"; dsize: >800;)`
4	DoS-атака Winnuke	`alert tcp $EXTERNAL_NET any -> $HOME_NET 135:139 (msg: "DoS Winnuke attack"; flags: U+;)`

5	Запрос на подключение к 139 порту (служба SMB) из внешней сети (два варианта)	`alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg: "NETBIOS SMB IPC$ share access"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "\\IPC$|00|"; nocase;)`
		
		`alert tcp $EXTERNAL_NET any -> $HOME_NET 139 (msg:"NETBIOS SMB IPC$ share access (unicode)"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content:`

№	Описание	Правило
		"|5c00|I|00|P|00|C|00|$| 00|"; nocase;)
6	Запрос на подключение к 445 порту (служба SMB) из внешней сети (два варианта)	alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg: "NETBIOS SMB IPC$ share access"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "\\IPC$|00|"; nocase;) alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"NETBIOS SMB IPC$ share access (unicode)"; flags: A+; content: "|00|"; offset: 0; depth: 1; content: "|FF|SMB|75|"; offset: 4; depth: 5; content: "|5c00|I|00|P|00|C|00|$|00|"; nocase;)
6	Обнаружение сканирования портов методом NULL.	alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"NULL port scanning"; flags:!FSRPAU;)
7	Обнаружение сканирования портов методом XMAS.	alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"XMAS port scanning"; flags:FPU+;)

ВЫПОЛНИТЬ!

5. Создать в каталоге «\snort\etc» файл «my.conf», содержащий следующие строки:

   ```
   var HOME_NET <IP-адрес_СОА>
   var EXTERNAL_NET !$HOME_NET
   ```

6. Добавить в файл «my.conf» правило, позволяющее обнаруживать входящие ECHO-запросы. Проверить, происходит ли обнаружение, запустив СОА из каталога «\snort\bin» следующей командой (из «командной строки»):

   ```
   snort -i <интерфейс> -c ../etc/my.conf -l ../log
   ```

   Для проверки выполнить несколько ECHO-запросов с другого компьютера, используя команду:

   ```
   ping <IP-адрес_СОА>
   ```

7. Дополнить файл «my.conf» правилами, указанными в табл. 4.3. Для проверки обнаружения подключений к службе SMB использовать команду:

   ```
   net use \\<IP-адрес_СОА>\IPC$ "" /user:""
   ```

К какому порту производится подключение? Как это зависит от используемой операционной системы?

4.5.5. Выявление факта сканирования портов

В СОА Snort встроен программный модуль, позволяющий выявлять сканирование портов защищаемой системы. Алгоритм, обнаруживающий сканирование, основан на том, что при сканировании портов существенно увеличивается количество исходящих TCP-пакетов с установленным флагом RST. Установка этого флага на отправляемом в ответ пакете означает, что порт, к которому производилось обращение, закрыт. Таким образом, анализируя количество пакетов с установленным флагом RST, можно обнаружить факт сканирования портов системы.

Программный модуль, или препроцессор, как его называют разработчики Snort, инициализируется из файла конфигурации следующим образом. В конфигурационный файл следует добавить следующие строки:

```
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan: proto { <протокол> } scan_type
{ <тип_сканирования> } sense_level
{ <чувствительность> } logfile { <файл_с_отчетом> }
```

Первая строка предназначена для инициализации препроцессора Flow, без которого модуль обнаружения сканирования не работает. Вторая строка инициализирует препроцессор sfPortscan, при этом задаются следующие параметры (жирным шрифтом показаны рекомендуемые значения):

`<протокол>` — анализируемый протокол (`tcp`, `udp`, `icmp`, `ip_proto`, **`all`**)

`<тип_сканирования>` — выявляемые типы сканирования (`portscan`, `portsweep`, `decoy_portscan`, `distributed_portscan`, **`all`**)

`<чувствительность>` — чувствительность (`low`, **`medium`**, `high`)

`<файл_с_отчетом>` — имя файла, в который будет помещен отчет об обнаруженных попытках сканирования портов

Файл с отчетом будет располагаться в каталоге, указанном параметром `-c` при запуске Snort. Чувствительность определяет перечень анализируемой информации и в итоге сказывается на вероятности «ложной тревоги» (для high она наибольшая). За более подробной информацией о параметрах модуля sfPortscan следует обращаться к документации на Snort.

Приведем пример настройки модуля sfPortscan:

```
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan: proto { all } scan_type { all }
sense_level { medium } logfile { portscan.log }
```

При данной настройке Snort будет выявлять все описанные в разделе 2 методы сканирования портов. Необходимо отметить, что две и больше процедуры сканирования портов, выполненные во время одного сеанса работы Snort, будут отражены в файле регистрации событий только один раз. Это остается справедливым даже в том случае, когда используются разные методы сканирования. Таким образом, для поверки возможности обнаружения сканирования, выполняемого разными методами, Snort необходимо закрывать и запускать снова.

ВЫПОЛНИТЬ!

8. Дополнить файл «my.conf» приведенными выше строками для настройки препроцессора sfPortscan. С использованием утилиты nmap проверить, происходит ли обнаружение попыток сканирования портов защищаемого узла. Использовать следующие команды для запуска сканирования:
   `nmap <IP-адрес_СОА> -v -sT -p <диапазон_портов>` — для сканирования методом с полным циклом подключения (метод Connect)
   `nmap <IP-адрес_СОА> -v -sS -p <диапазон_портов>` — для сканирования с неполным циклом подключения (метод SYN)
   `nmap <IP-адрес_СОА> -v -sN -p <диапазон_портов>` — для сканирования при помощи TCP-пакета со сброшенными флагами (метод NULL)
   `nmap <IP-адрес_СОА> -v -sX -p <диапазон_портов>` — для сканирования при помощи TCP-пакета со всеми установленными флагами (метод XMAS)
9. Какие методы сканирования позволяют практически выявлять наличие открытых портов? Как это зависит от используемой операционной системы? Все ли указанные методы сканирования обнаруживает СОА Snort?