Визитка

ЮРИЙ ДЕНИСОВ, начальник технического отдела одного из провайдеров г. Владимира. Сфера профессиональных интересов: сетевые технологии, проектирование распределенных сетей передачи данных, развертывание телематических сервисов, биллинговые системы

Атаки из сети Интернет

Внедряем IDS Snort для защиты корпоративной сети

В статье рассматриваются система обнаружения и предотвращения вторжений Snort, способы ее внедрения в сеть предприятия, а также модули препроцессоров, участвующих в детектировании сетевых атак и аномалий

Для того чтобы эффективно применять тот или иной механизм, мы должны прежде всего четко представлять, как он работает, что может, а на что не способен. Данная статья не является пошаговой инструкцией по установке и базовой настройке этой программы, коих в Интернете великое множество. Напротив, цель ее – провести небольшое исследование возможностей Snort путем последовательного изучения его структуры (с практическими примерами) и получить четкое представление о том, что программа умеет и нужна ли она в вашей корпоративной сети.

В предыдущих статьях мы рассмотрели ряд мер по предотвращению некоторых видов атак на наиболее часто используемые сервисы (SMTP-, DNS-, веб-серверы), таких как явные DDoS-атаки, атаки с подменой IP-адресов. Это делалось прежде всего с помощью настроек файрвола, а также настроек самих программ, предоставляющих данные сервисы, и своевременного их обновления.

Минусы подобных мер очевидны – это лишь базовая защита, не позволяющая противостоять более хитроумным методам взлома. Для того чтобы детектировать последние, необходимо иметь в своем арсенале более интеллектуальную систему, позволяющую опираться не только на информацию, получаемую из полей IP- и TCP/UDP-пакетов, но и на содержание самих пакетов. Кроме того, система должна уметь оценивать не только одиночные пакеты, но и их цепочки в целях выявления аномального трафика. Идеальным было бы, кроме всего прочего, обладать информацией о трафике на более высоком сеансовом уровне. Данный вид задач и призвана решать система обнаружения вторжений (IDS) Snort.

Архитектура Snort

Упрощенно говоря, архитектуру Snort можно представить в виде трех блоков – это снифер, прослушивающий интерфейс и вылавливающий сетевые пакеты, модуль обнаружения (препроцессор и процессор) и модуль реагирования и оповещения. В сети может использоваться одна или несколько систем Snort. Использование нескольких IDS улучшает защиту сети, но и усложняет структуру системы. Кроме того, Snort умеет работать в качестве сниффера, выдавая необходимую прослушиваемую информацию, а также умеет записывать эту информацию на диск в целях анализа ее в будущем.

Варианты включения Snort

Коротко остановимся на вариантах включения Snort в сеть. Вариантов много, но, на мой взгляд, самыми удобными являются два из них.

Первый – установка Snort непосредственно на сервер, выступающий в роли пограничного маршрутизатора. В этом случае он видит 100% приходящего к вам трафика. Удобно то, что исходящий трафик от машин в локальной сети попадет на Snort до того, как будет обработан механизмом NAT (трансляции сетевых адресов).

Второй вариант – включение зеркалирования на свитчах в локальной сети таким образом, чтобы на сервер с запущенным IDS приходил весь трафик, идущий через них. Эту схему удобно применять, когда в качестве пограничного роутера используется не сервер, а обычный маршрутизатор, например, Cisco. Однако тогда не видны атаки на сам маршрутизатор. Чтобы это исправить, необходимо зазеркалировать трафик еще с одной точки, располагающейся до этого самого маршрутизатора. Туда можно будет поставить еще один свитч с функцией зеркалирования. Некоторые маршрутизаторы и сами способны зеркалировать трафик. Все зависит от выбранного бренда и наличия свободных портов на них. В любом случае выбор используемого варианта за вами, главное – учитывать одно: на Snort должен попадать абсолютно весь интересующий нас трафик, и не должно быть потерянных в процессе зеркалирования пакетов.

Сам по себе Snort является по большому счету обычным tcpdump, с той лишь разницей, что он умеет анализировать снимаемую им информацию в соответствии с рядом задаваемых правил. Большие библиотеки правил уже существуют и доступны в Интернете. Информацию по ним можно найти на сайте разработчиков www.snort.org. Кроме того, вы можете самостоятельно добавить в конфигурацию любое необходимое вам правило.

Схема работы

Чтобы понять, что делает Snort и чем он нам может быть полезен, необходимо понять схему его работы. Если мы хотим, чтобы наша система работала стабильно и не приносила неожиданных сюрпризов, мы должны четко представлять все действия по проверке трафика, которые она производит.

Во-первых, Snort снимает информацию о всем трафике на интерфейсе. И здесь есть один момент. Snort действительно видит весь приходящий на интерфейс трафик, даже тот, который был отфильтрован файрволом iptables. Почему? Дело в том, что он основан на библиотеке pcap, которая работает с драйвером сетевой карты. Snort получает информацию о пакете до того момента, как этот пакет попадет в ядро ОС. По понятным причинам, Snort никогда не увидит исходящие с интерфейса пакеты, которые были отфильтрованы пакетным фильтром iptables.

Снятая информация обрабатывается препроцессорами и затем передается модулю бесконтекстной обработки. В чем их разница? Бесконтекстные правила подразумевают обработку каждого пакета независимо от остальных, а, следовательно, опираются на информацию, полученную из заголовков и содержания лишь одного пакета, в то время как контекстные правила (которые в основном и используются в препроцессорах) обрабатывают цепочку пакетов как единое целое, позволяя выявлять информацию по протоколам более высокого уровня. Рассмотрим кратко виды препроцессоров и их назначение.

Прежде всего хочу остановиться на одной опции, которая будет часто встречаться в различных препроцессорах:

```
inspection type <stateful | stateless>
```

В документации смысл опции объяснен, на мой взгляд, недостаточно развёрнуто. Многие не понимают разницы между двумя этими методами и, как следствие, выбирают тот или иной наобум.

Глобально существует два метода проверки трафика. Первый называется stateful, это анализ потока данных, который представляет собой установленное сетевое соединение. В противовес ему метод stateless обрабатывает каждый пакет в сетевом соединении отдельно, не обращая внимания на то, в каком из сетевых соединений участвует текущий пакет. В качестве аналогии можно привести две функции файрвола – трансляцию сетевых адресов (NAT), которая опирается на отслеживание сессий, и простые списки контроля доступа (ACL), которые применяются ко всем входящим пакетам без трассировки существующих сетевых соединений. Метод stateful является более быстродействующим, он также подвержен меньшему количеству ложных срабатываний, но процент выявленных атак и аномалий у него больше.

Например, если необходимо проверять какой-либо из параметров трафика, предположим, наличие шифрования, метод stateful проверит первый пакет, и этого будет достаточно, в то время как stateless станет проверять на этот признак каждый приходящий пакет и, как следствие, приведет к увеличению накладных расходов при анализе сетевого потока.

Существует несколько препроцессоров, которые скорее всего будут включены всегда, поскольку осуществляют ряд очень полезных функций.

Frag3. Позволяет работать с фрагментированными пакетами, дефрагментируя их и передавая дальше по цепочке для анализа. Имеет несколько очевидных параметров, позволяющих более тонко его настроить, таких как максимальное количество фрагментов в цепочке, минимальный размер фрагмента.

Stream5. Позволяет собирать TCP-, UDP- и ICMP-потоки для последующей их обработки. Кроме того, данный модуль позволяет определять ряд аномалий, присутствующих в потоке по протоколу TCP, и генерировать связанные с этим оповещения.

sfPortscan. Назначение этого препроцессора понятно из его названия. Он позволяет определить ситуацию, когда с удаленного компьютера производится сканирование IP-адресов и портов компьютеров в вашей сети. Его можно сконфигурировать на различные типы протоколов (TCP,UDP,ICMP и т.д.) и различные виды атак. На мой взгляд, модуль нужный, поскольку сканирование по факту является так называемой разведкой – первой частью удаленной атаки.

SSL/TLS. Необходим для того, чтобы вовремя детектировать зашифрованный трафик (который не может быть обработан) и отключать его проверку, уменьшая таким образом нагрузку на систему и снижая количество ложных срабатываний.

Следующий ряд препроцессоров необходимо будет включать, только если в вашей сети используются соответствующие сервисы.

Включение нормализации DNS-трафика

Если вы используете кэширующий DNS-сервер, препроцессор DNS поможет детектировать некоторые виды атак. Механизм его работы таков – он просматривает ответный трафик от DNS-сервера в целях выявления следующих ситуаций:

Experimental Record Types – наличие запросов о новых, экспериментальных типах записей в файле зоны.

Obsolete Record Types – наличие запросов об устаревших типах записей в файле зоны.

DNS Client RData Oveflow – эксплойт, при котором DNS-клиент подвергается опасности переполнения буфера при обработке поля RDATA в ответе сервера. Удаленный злоумышленник может воспользоваться этим для выполнения произвольного кода на удаленной системе.

Ваш кэширующий сервер может рассматриваться как клиент по отношению к другим вышестоящим DNS-серверам. Для его работы также необходим Stream5 – для обработки потоков TCP и UDP. Настройки позволяют прослушивать любой порт, с которого приходят ответы от DNS-сервера (по умолчанию это порт номер 53). Пример настройки препроцессора для детектирования всех трех ситуаций:

```
preprocessor dns: ports { 53 } enable_rdata_overflow ↵
        enable_obsolete_types enable_experimental_types
```

Включение защиты SMTP-сервера

Для этого служит препроцессор smtp. Назначение его понятно из названия. Препроцессор анализирует трафик на предмет наличия в нем SMTP-команд и ответов на них, нормализует его и детектирует аномалии, при этом игнорируя зашифрованный (TLS)-трафик.

Он имеет несколько опций, таких как максимальная длина команды, посылаемой серверу, и максимальная длина ответа, детектирование неправильных или неизвестных команд, нормализация команд и т.д. Все они хорошо описаны в документации. По умолчанию препроцессор нормализует некоторые команды и устанавливает для ряда команд максимально допустимую длину:

```
preprocessor SMTP: ports { 25 } inspection_type stateful ↵
        normalize cmds normalize_cmds { EXPN VRFY RCPT } ↵
        alt_max_command_line_len 260 { MAIL } ↵
        alt_max_command_line_len 300 { RCPT } ↵
        alt_max_command_line_len 500 { HELP HELO ETRN } ↵
        alt_max_command_line_len 255 { EXPN VRFY }
```

Хочу заметить, что неправильно настроенные параметры могут частично или полностью блокировать работу вашего SMTP-сервера. Самым правильным будет не спеша изучить каждую опцию, чтобы верно подобрать величину параметра к каждой из них.

Выявление атак на веб-сервер

Для этого служит препроцессор HTTP Inspect. Его основной задачей является декодирование потоков данных, проходящих через нашу сеть, и выявление в них http-трафика в целях его анализа. При этом анализируются оба направления – как трафик от клиента к серверу, так и обратный ответ. В случае если трафик будет зашифрован, он не сможет быть прочтен и обработан. Настройка препроцессора состоит из двух частей. В одной устанавливаются глобальные опции, в другой – опции для конкретных серверов.

Для того чтобы препроцессор правильно отрабатывал трафик, необходимо прежде всего определить тип защищаемого сервера. Для этого служит опция:

```
profile <all|apache|iis|iis5_0|iis4_0>
```

В зависимости от выставленных опций препроцессор будет по-разному нормализовывать http-трафик, приходящий на сервер. Практически это означает, что в зависимости от выбранного профиля будут автоматически установлены значения целого ряда опций, специфичные для того или иного сервера (посмотреть их можно в документации на Snort). Однако при необходимости, любую из этих опций можно переопределить.

```
preprocessor http_inspect: ↵
        global iis_unicode_map unicode.map 1252 ↵
        detect_anomalous_servers
preprocessor http_inspect_server: server 1.2.3.4 ↵
        profile apache ports { 80 }
```

В данном примере присутствует одна полезная, на мой взгляд, опция – detect_anomalous_servers, – которая позволяет выявлять http-активность на портах, обычно не используемых для передачи http-трафика.

Выявление атак на SSH-сервер

Для этого служит препроцессор SSH. Позволяет детектировать ряд эксплоитов, которым могут быть подвержены SSH-серверы. Для ускорения работы можно будет выключить ряд опций, которые вам не нужны. К примеру, если вы используете на своих серверах только SSH версии 2, то можно отключить модуль enable_ssh1crc32, поскольку данному типу уязвимости (ошибка целочисленного переполнения в CRC32) подвержены только серверы SSH версии 1.

Далее опишем препроцессоры, которые используются несколько реже предыдущих.

FTP/Telnet Preprocessor. Детектирует в потоке протокол FTP и escape последовательности Telnet, а затем нормализует их. При этом он может быть сконфигурирован отдельно для каждого FTP-сервера и клиента, позволяя переопределять опции для них. Вообще говоря, использование Telnet-протокола для доступа к вашим устройствам из внешнего мира – очень плохая идея, поскольку вся передаваемая в нем информация никак не зашифровывается, включая и связки логинов-паролей. В связи с этим правильнее будет Telnet не использовать вообще, а данный препроцессор заставить проверять только FTP-трафик.

ARP Spoof Preprocessor. Этот препроцессор является очень удобным для выявления программных снифферов, поскольку детектирует атаки типа ARP-spoofing, задачей которых является перенаправление трафика с некоторого хоста на себя, в целях прослушивания всего трафика, предназначенного ему.

Perfomance monitor. Позволяет определять нагрузку, которую создает Snort, на вашу систему. Идеально было бы наладить проброс данных, поступающих из этого модуля в какую-нибудь систему мониторинга, способную визуально отображать эти параметры. Например, MRTG, Zabbix.

RPC Decode. Собирает множество фрагментов в единую последовательность RPC-команд. Если службы RPC в вашей сети не используются, – этот препроцессор можно отключить.

DCE/RPC 2 Preprocessor. Определяет и обрабатывает трафик протоколов DCE/RPC и Samba, используемых, как правило, для передачи файлов и управления в локальной сети. На самом деле порты, отвечающие за эти протоколы, не должны быть видны в сети Интернет, поэтому я этот препроцессор отключаю вовсе.

Sensitive Data Preprocessor. Реагирует на определенные последовательности символов, передаваемых по сети. Это могут быть любые данные, которые вы считаете приватными, например, номер вашей кредитной карточки.

Normalizer. Нормализует ряд параметров в пакетах на уровнях TCP/IP, в том числе ICMP. Кроме того, позволяет следить за полем TTL в IP-пакете и при необходимости его корректирует.

***

Итак, роли каждого из препроцессоров четко разграничены, что дает нам возможность более тонкой настройки системы, включая либо отключая тот или иной препроцессор. Это важно в том случае, если сетевой поток достаточно велик, ведь каждый из модулей добавляет свой процент в загрузку системы. Отключив ненужные, на ваш взгляд, модули, вы сможете получить оптимальную по быстродействию IDS.

В следующей статье рассмотрим работу Snort после того, как информация прошла обработку препроцессорами, ознакомимся с подсистемой реагирования и регистрации событий. Кроме того, постараемся оценить производительность Snort на некоторых типах оборудования и вклад в нагрузку системы различных модулей программы. EOF