Л7. Инструментальные средства командной строки для анализа пакетов.

Проведение анализа пакетов в ходе расследования случаев нарушения безопасности всегда считалось непростым делом, поскольку оно включает в себя неизвестный элемент устройства, которое подвергается атаке злоумышленника. В самом деле, нельзя же добраться до рабочего места атакующего злоумышленника и подвергнуть его допросу или сравнить его обычный трафик с исходным. Вместо этого приходится иметь дело с перехваченным образцом взаимодействия своей и чужой системы. Правда, для удаленного вторжения в чужие системы злоумышленникам придется так или иначе взаимодействовать с сетью. И это им хорошо известно, поэтому у них нет недостатка в приемах маскирования своей вредоносной деятельности.

В этой главе рассматриваются различные практические аспекты нарушения безопасности системы на уровне сети, включая обследование сети,

переадресацию вредоносного трафика и самые распространенные методики внедрения вредоносного кода. В некоторых случаях будет проведен анализ незаконного вторжения в систему для интерпретации сетевого трафика на основании предупреждений от системы обнаружения вторжений. Проработав материал этой главы, вы получите ясное представление о сетевой безопасности, что может оказаться для вас очень важным, даже если вы и не занимаетесь в настоящий момент вопросами безопасности.

Обследование сети

Злоумышленник первым делом выполняет углубленное обследование целевой системы. Эта стадия обычно называется *получением отпечатка (footprinting)* и нередко осуществляется с помощью различных общедоступных ресурсов, включая веб-сайт целевой системы или компании Google. По окончании подобного обследования злоумышленник, как правило, приступает к сканированию целевой системы по ее IP-адресу (или DNS-имени) на наличие в ней открытых портов или действующих служб.

Сканирование дает злоумышленнику возможность выяснить, активна и доступна ли целевая система. Рассмотрим в качестве примера сценарий, где преступники собираются ограбить самый крупный банк в каком-нибудь городе. Потратив не одну неделю на тщательную проработку плана ограбления и прибыв на место преступления, они в конечном итоге обнаружили, что банк переехал на другую улицу. Представьте еще более скверную ситуацию, когда преступники собирались ограбить банк в обычные часы его работы, украв ценности из банковского хранилища, но, прибыв на место преступления, обнаружили, что банк в этот день закрыт. Таким образом, первое препятствие, которое придется преодолеть при ограблении банка или совершении атаки на сеть, — убедиться, что целевой хост активен и доступен.

Кроме того, сканирование сети позволяет злоумышленнику выяснить, через какие порты целевая система взаимодействует с сетью. Если снова обратиться к аналогии ограбления банка, то нетрудно представить, что произойдет, если преступники заявятся в банк, совершенно не зная планировку его помещения. Им не удастся добраться до банковского хранилища, не зная слабых мест в физической защите банка. В этом разделе будут рассмотрены самые распространенные методики сканирования сети, применяемые для выявления хостов, их открытых портов и уязвимостей в сети.

ПРИМЕЧАНИЕ *До сих пор в этой книге передающая и принимающая стороны сетевого соединения назывались клиентом и сервером. А в этой главе эти стороны называются злоумышленником и целевым хостом соответственно.*

Сканирование пакетами SYN

Файл перехвата `synscan.pcapng`

Зачастую первым видом обследования целевой системы является *сканирование пакетами SYN по протоколу TCP*, иначе называемое *скрытым сканированием* или же *сканированием полуоткрытых соединений*. Сканирование пакетами `SYN` считается самым распространенным по ряду следующих причин.

- Проводится очень быстро и надежно.
- Пригодно для всех платформ независимо от реализации стека протоколов TCP/IP.
- Вносит меньше помех, чем другие методики сканирования.

В основу сканирования пакетами `SYN` положен трехэтапный процесс установки связи по протоколу TCP, что позволяет выяснить, какие именно порты открыты на целевом хосте. В этом случае злоумышленник посылает пакет `SYN` по протоколу TCP на определенный ряд портов целевого хоста, как будто бы пытаясь установить с ним канал для обычного обмена данными через его порты. Как только этот пакет будет получен целевым хостом, может произойти одно из нескольких событий, как показано на рис. 12.1.

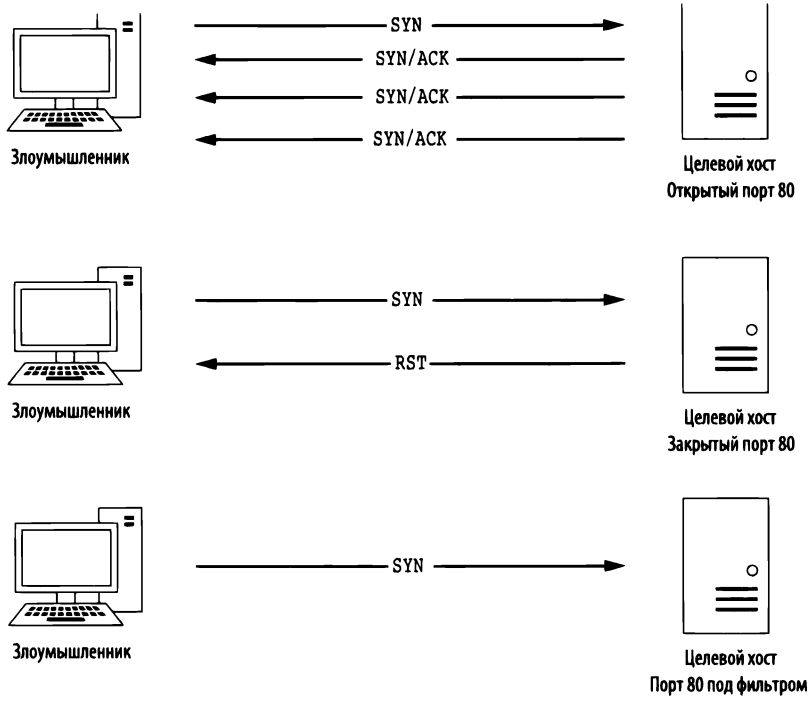


Рис. 12.1. Возможные результаты сканирования пакетами SYN по протоколу TCP

Если служба на целевой машине прослушивает сеть через порт, принимающий пакет SYN, она ответит злоумышленнику пакетом SYN/ACK по протоколу TCP, обычно отправляемом на втором этапе процесса установки связи по этому протоколу. В итоге злоумышленнику становится известно, что порт открыт и служба прослушивает сеть через него. При обычных обстоятельствах окончательный пакет ACK должен быть отправлен по протоколу TCP с целью завершить процесс установки связи. Но в данном случае злоумышленнику совсем не нужно, чтобы это произошло, поскольку ему не требуется больше связь с хостом в данный момент. Следовательно, злоумышленник не будет пытаться завершить процесс установки связи по протоколу TCP.

Если же ни одна из служб не прослушивает сеть через сканируемый порт, злоумышленник не получит пакет SYN/ACK. Но он может получить в ответ пакет RST, указывающий на то, что порт закрыт, хотя это зависит от конкретной конфигурации операционной системы целевого компьютера. С одной стороны, злоумышленник может вообще не получить ничего в ответ. Отсутствие ответа может означать, что сканируемый порт находится под фильтром на промежуточном устройстве, например, брандмауэре или на самом хосте. А с другой стороны, ответ может быть утерян при передаче. Таким образом, результат такого сканирования получается неоднозначным, несмотря на то, что он, как правило, указывает на то, что порт закрыт.

В файле перехвата synscan.pcapng предоставляется характерный пример сканирования пакетами SYN, выполняемого средствами Nmap — надежного приложения для сканирования сети, разработанного Гордоном Лайеном (Gordon Lyon) по прозвищу "Федор". Оно способно выполнять практически любые вообразимые виды сканирования сети. Свободно загрузить приложение Nmap можно по адресу http://www.nmap.com/download.html.

В данном примере перехваченный трафик состоит приблизительно из 2000 пакетов. Это означает, что анализируемое здесь сканирование сети произведено в приемлемых масштабах. Чтобы установить пределы такого сканирования, лучше всего просмотреть его результаты в окне Conversations, как показано на рис. 12.2. В этом окне должен быть показан единственный диалог по протоколу IPv4 ❶ между злоумышленником, находящимся по адресу **172.16.0.8**, и целевым хостом, расположенным по адресу **64.13.134.52**. Кроме того, в окне Conversations можно увидеть, что между указанными выше хостами проведено 1994 диалога по протоколу TCP ❷, а, по существу, для каждой пары портов, задействованных в обмене данными, организован новый диалог.

Сканирование сети происходит очень быстро, и поэтому прокрутка содержимого файла перехвата — далеко не самый лучший способ поиска ответа на каждый первоначальный пакет SYN. Ведь прежде получения ответа на первоначальный пакет может быть отправлен ряд дополнительных пакетов.

Правда, для обнаружения нужного сетевого трафика можно составить вспомогательные фильтры.

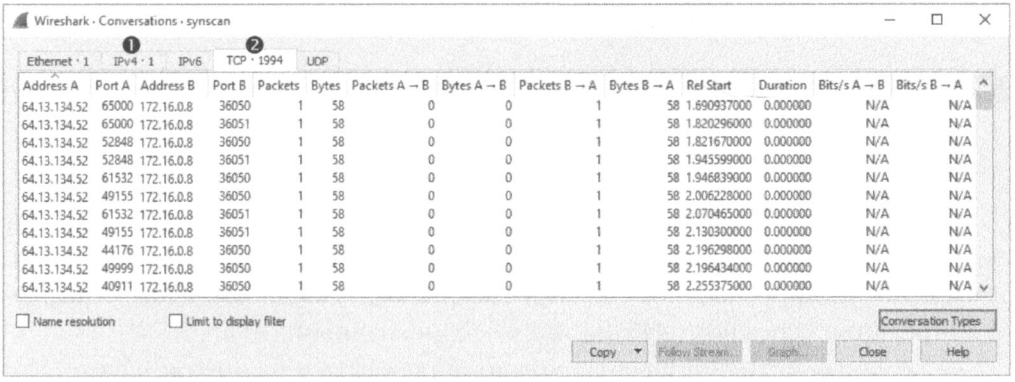


Рис. 12.2. В окне Conversations наглядно демонстрируется разнообразие обменов данными по протоколу TCP

Использование фильтров при сканировании пакетами SYN

В качестве примера фильтрации результатов сканирования сети рассмотрим первый пакет из упомянутого выше файла перехвата. Это пакет `SYN`, отправленный целевому хосту через порт **443** (по протоколу HTTPS). Чтобы выяснить, получен ли ответ на этот пакет, можно создать фильтр для показа всего исходящего трафика через порт **443**. Ниже поясняется, как это сделать очень быстро.

1. Выберите первый пакет из файла перехвата `synscan.pcapng`.
2. Разверните TCP-заголовок этого пакета в панели Packet Details.
3. Щелкните правой кнопкой мыши на поле Destination Port (Порт получателя), выберите команду Prepare as Filter⇨Selected (Подготовить как фильтр⇨Выбрать) из контекстного меню.
4. В итоге выбранный фильтр появится в диалоговом окне фильтров всех пакетов, проходящих через порт **443**. Щелкните в верхней части диалогового окна фильтров и удалите часть **dst** из выбранного фильтра, поскольку нам требуется получить список всех пакетов (как входящих, так и исходящих) по порту **443**.

Получающийся в итоге фильтр отсеет два пакета, которые являются пакетами `SYN`, передаваемыми по протоколу TCP от злоумышленника к целевому хосту, как показано на рис. 12.3.

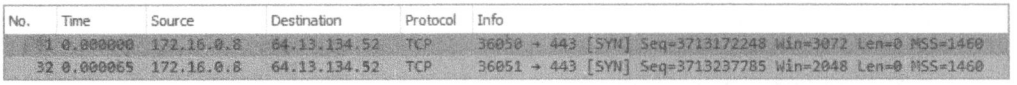


Рис. 12.3. Две попытки установки соединения с помощью пакетов SYN

ПРИМЕЧАНИЕ *В этом разделе при отображении пакетов используется формат времени* Seconds Since Previous Displayed Packet *(Количество секунд, прошедших с момента отображения предыдущего пакета).*

Отсутствие ответа на эти пакеты, вероятно, свидетельствует о том, что ответ был отфильтрован целевым хостом или промежуточным устройством, а, возможно, сканируемый порт оказался закрытым. В конечном счете результат сканирования через порт **443** оказался неоднозначным.

Ту же самую методику анализа можно опробовать и на другом пакете, чтобы попытаться получить иные результаты. С этой целью удалите предыдущий фильтр и выберите из списка пакет 9. Это пакет SYN, отправляемый в порт **53**, который обычно связан со службой DNS. Применяя описанную выше методику или внеся коррективы в предыдущий фильтр, составьте новый фильтр для отображения всего трафика, проходящего через порт **53** по протоколу TCP. Применив этот фильтр, вы должны в итоге обнаружить пять пакетов (рис. 12.4).

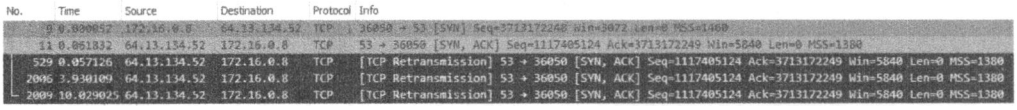


Рис. 12.4. Пять пакетов, указывающих на открытый порт

Первым из них является пакет 9 типа SYN, выбранный вначале перехваченного трафика. А вторым пакетом следует ответ SYN/ACK, посылаемый целевым хостом и вполне ожидаемый при трехэтапном процессе установки связи по протоколу TCP. При нормальных обстоятельствах следующим должен быть пакет ACK, отправляемый хостом, пославшим первоначальный пакет SYN. Но в данном случае злоумышленнику совсем не нужно завершать трехэтапный процесс установки связи, поэтому ответ не посылается. В итоге целевой хост повторно передаст пакет SYN/ACK еще три раза, прежде чем процесс установки связи будет аварийно завершен. А поскольку ответ SYN/ACK получен при попытки связаться с хостом через порт **53**, то благоразумнее предположить, что сканируемая служба прослушивает сеть через данный порт.

Очистите полученный результат и повторите данный процесс еще раз для пакета 13. Это пакет SYN, отправляемый в порт **113**, который обычно связан с протоколом Ident, зачастую применяемым для распознавания служб IRC и

аутентификации в них. Если применить фильтр того же самого типа к порту, перечисленному в данном пакете, то в итоге появятся четыре пакета, как показано на рис. 12.5.

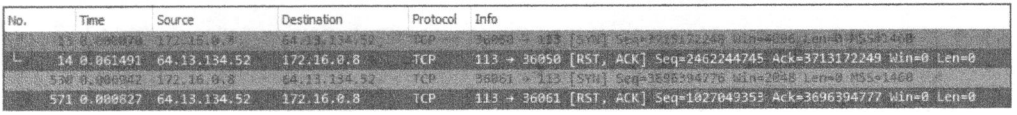


Рис. 12.5. Пакет SYN, вслед за которым посылается пакет RST, указывающий на то, что порт закрыт

Первым в данном случае следует первоначальный пакет SYN, вслед за которым целевой хост сразу же отправляет пакет RST. Это свидетельствует о том, что целевой хост не принимает подключения через свой порт и что служба на нем, вероятнее всего, не запущена.

Выявление открытых и закрытых портов

А теперь, когда стали понятны разные типы ответов, которые способно выявить сканирование пакетами SYN, требуется найти какой-нибудь быстрый способ выявления открытых и закрытых портов. И с этой целью можно снова обратиться к окну Conversations, где диалоги по протоколу TCP нетрудно отсортировать по номерам пакетов, чтобы первыми следовали наибольшие значения. Для этого достаточно щелкнуть на заголовке **Packets**, чтобы появилась стрелка, указывающая вниз, как показано на рис. 12.6.

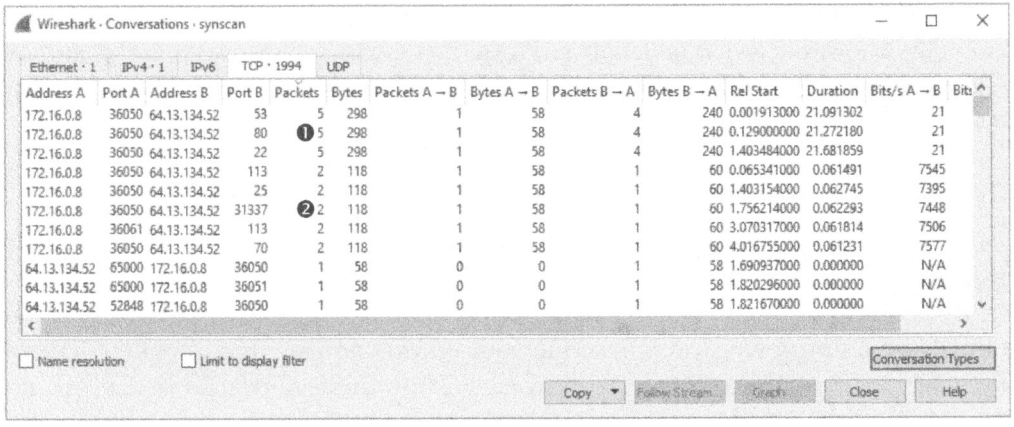


Рис. 12.6. Выявление открытых портов в окне Conversations

В каждом диалоге через три сканируемых порта проходят по пять пакетов ❶. Известно, что порты **53**, **80** и **22** открыты, поскольку эти пять пакетов представляют следующую последовательность установления связи:

первоначальный пакет SYN, соответствующий ему пакет SYN/ACK, а также три пакета SYN/ACK, повторно передаваемых целевым хостом.

Из пяти портов только два пакета были вовлечены в обмен данными ❷. Первым из них является первоначальный пакет SYN, а вторым — пакет RST, отправляемый целевым хостом. Эти пакеты указывают на то, что порты **113**, **25**, **31337** и **70** закрыты.

Остальные записи, отображаемые в окне Conversations, включают в себя только один пакет. Это означает, что целевой хост вообще не ответил на первоначальный пакет SYN. А оставшиеся порты, скорее всего, закрыты, но в этом нет никакой уверенности.

Рассматриваемая здесь методика подсчета пакетов вполне подошла для сканирования данного хоста, хотя она пригодна не для всех сканируемых хостов, а следовательно, полагаться только на них не стоит. Вместо этого особое внимание следует уделить изучению внешнего вида и возможного назначения обычных и необычных реакций на обычные стимулы.

Получение отпечатка операционной системы

Злоумышленник придает большое значение сведениям о целевой операционной системе. Эти сведения помогают ему правильно настроить все свои методы атаки на данную систему, а также позволяют выяснить местоположение определенных критически важных файлов и каталогов в целевой файловой системе, если, конечно, ему удастся получить доступ к ней.

Получение отпечатка операционной системы (operating system fingerprinting) обозначает группу методик, применяемых с целью определить операционную систему, работающую в целевой системе без физического доступа к данной системе. Имеются два способа получения отпечатка операционной системы: пассивный и активный.

Пассивное получение отпечатка операционной системы

Файл перехвата **passiveos-fingerprinting.pcapng**

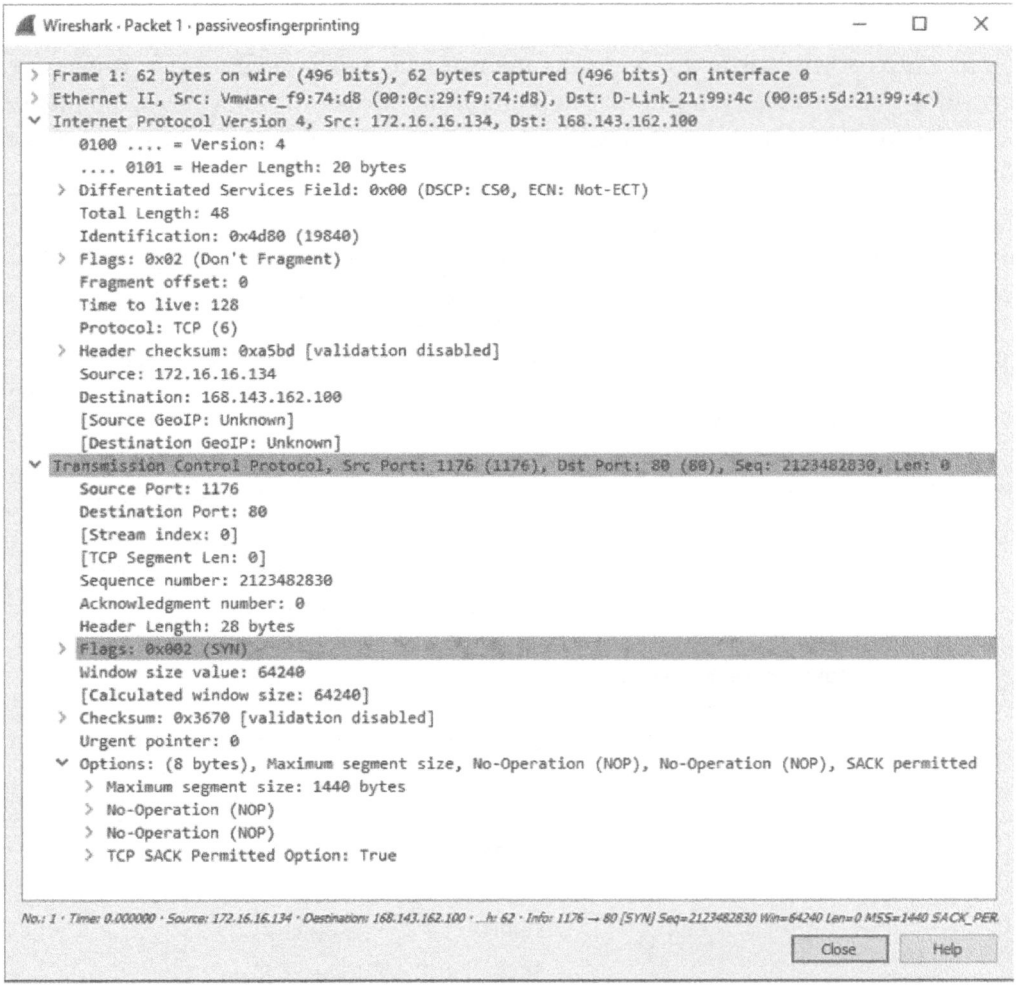
Используя методику *пассивного получения отпечатка*, можно исследовать определенные поля в пакетах, отправленных целевым хостом, чтобы определить применяемую в нем операционную систему. Такой способ считается пассивным, поскольку в этом случае вы принимаете только те пакеты, которые отправляет целевой хост, но сами активно не посылаете этому хосту никаких пакетов. Такой способ получения отпечатка операционной системы идеально подходит для злоумышленников, поскольку он дает им возможность действовать скрытно.

Так как же определить, какая именно операционная система работает на целевом хосте, исходя только из пакетов, которые он отправляет? Это возможно благодаря отсутствию стандартизированных значений в спецификациях, определенных в документах RFC на сетевые протоколы. Несмотря на то что различные поля в TCP-, UDP- и IP-заголовках имеют специальное назначение, устанавливаемые по умолчанию значения, как правило, определены не в каждом поле. Это означает, что реализация стека протоколов TCP/IP в каждой операционной системе требует определения собственных значений, устанавливаемых по умолчанию в этих полях. В табл. 12.1 перечислены наиболее употребительные поля и устанавливаемые в них значения, которые могут быть связаны с различными операционными системами. Следует, однако, иметь в виду, что эти значения могут быть изменены в новых выпусках операционных систем.

Таблица 12.1. Поля заголовков и их значения, используемые при пассивном получении отпечатка операционной системы

Заголовок протокола	Поле	Значение по умолчанию	Платформа
IP	`Initial time to live` (Первоначальное время жизни)	`64`	NMap, BSD, OS X, Linux
		`128`	Novell, Windows
		`255`	Cisco IOS, Palm OS, Solaris
IP	`Don't fragment` (Не фрагментировать)	Флаг установлен	BSD, OS X, Linux, Novell, Windows, Palm OS, Solaris
		Флаг сброшен	Nmap, Cisco IOS
TCP	`Maximum segment size` (Максимальный размер сегмента)	`0`	Nmap
		`1440-1460`	Windows, Novell
		`1460`	BSD, OS X, Linux, Solaris
TCP	`Window size` (Размер окна)	`1024-4096`	Nmap
		`65535`	BSD, OS X
		Переменное	Linux
		`16384`	Novell
		`4128`	Cisco IOS
		`24820`	Solaris
		Переменное	Windows
TCP	`SackOK` (Выборочное подтверждение разрешено)	Флаг установлен	Linux, Windows, OS X, OpenBSD
		Флаг сброшен	Nmap, FreeBSD, Novell, Cisco IOS, Solaris

Пакеты, содержащиеся в файле перехвата `passiveosfingerprinting.pcapng`, служат характерными примерами данной методики. В этом файле содержатся два пакета. Оба пакета относятся к типу `SYN` и отправляются по протоколу TCP в порт **80**, но они исходят из разных хостов. Используя только значения, содержащиеся в этих пакетах и обращаясь за справкой к табл. 12.1, можно определить архитектуру операционной системы, применяемой на каждом хосте. Подробные сведения о каждом пакете приведены на рис. 12.7.



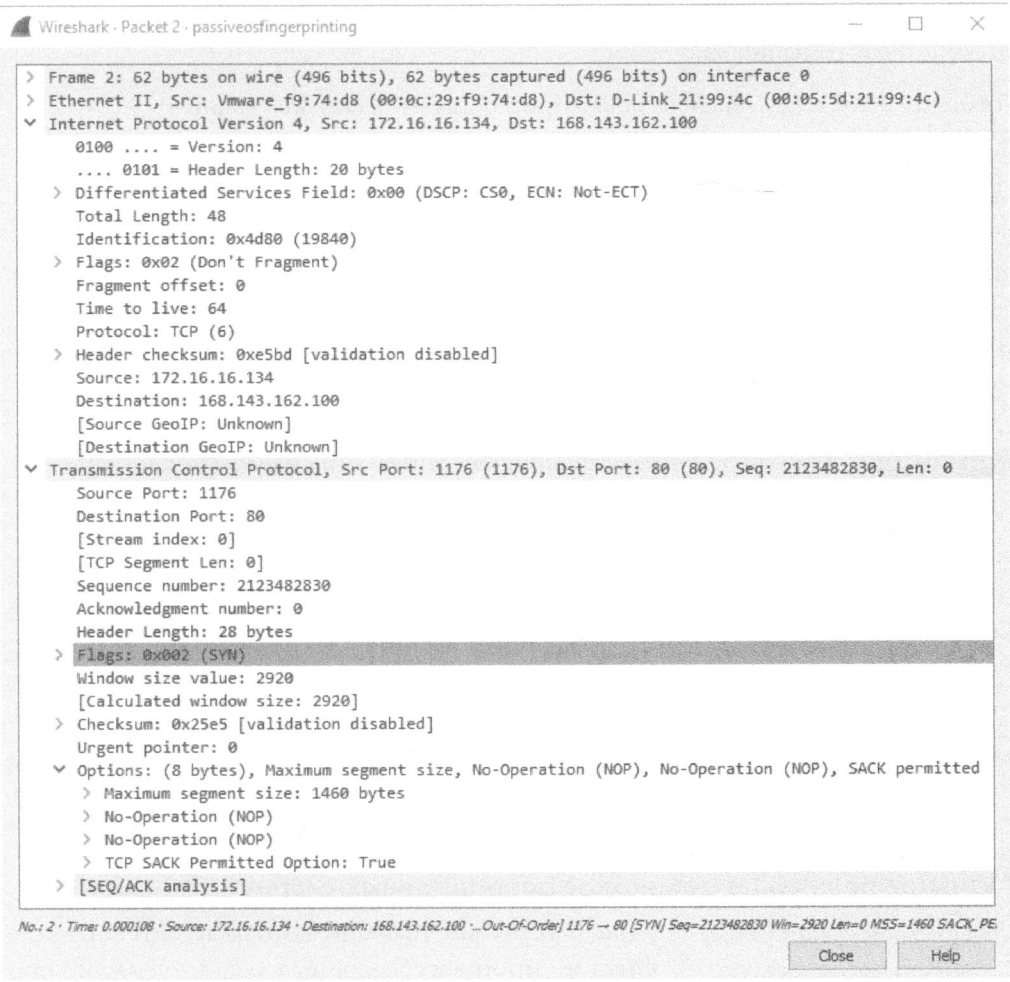


Рис. 12.7. По приведенным здесь пакетам можно узнать, из какой операционной системы они были присланы

Используя табл. 12.1 в качестве справки, можно составить табл. 12.2, где сведены соответствующие поля из этих пакетов. Основываясь на этих значениях, можно прийти к выводу, что пакет 1 был, вероятнее всего, отправлен устройством, работающим под управлением ОС Windows, а пакет 2 — устройством, работающим под управлением Linux.

Следует, однако, иметь в виду, что перечень полей, наиболее употребительных при получении отпечатка операционной системы и приведенных в табл. 12.2, нельзя считать исчерпывающим. Хотя методика применения этих полей может иметь немало неожиданных уверток, способных привести к заметным отклонениям от предполагаемых значений. Следовательно,

полностью полагаться на результаты получения пассивного отпечатка операционной системы нельзя.

Таблица 12.2. Сводка полей из пакетов, используемых при получении отпечатка операционной системы

Заголовок протокола	Поле	Значение из пакета 1	Значение из пакета 2
IP	`Initial time to live`	128	64
IP	`Don't fragment`	Флаг установлен	Флаг установлен
TCP	`Maximum segment size`	1440 байт	1460 байт
TCP	`Window size`	64240 байт	2920 байт
TCP	`SackOK`	Флаг установлен	Флаг установлен

ПРИМЕЧАНИЕ *Зачастую злоумышленники используют автоматизированные инструментальные средства для пассивного выявления типа операционной системы на целевом хосте. К числу таких средств, в которых применяются способы получения отпечатков операционных систем, относится p0f. Это инструментальное средство позволяет анализировать соответствующие поля в перехваченных пакетах и выводить сведения о предполагаемом типе операционной системы. Применяя инструментальные средства вроде p0f, можно получить сведения не только об архитектуре операционной системы, но иногда даже ее версию или уровень установки пакетов обновлений. Инструментальное средство p0f можно загрузить по адресу* **`http://lcamtuf.coredump.cx/p0f.shtml`**.

Активное получение отпечатков операционных систем

Файл перехвата `activeos-fingerprinting.pcapng`

Если пассивный текущий контроль сетевого трафика не приносит желаемых результатов, можно опробовать более прямой способ — *активное получение отпечатков операционных систем.* В этом случае злоумышленник активно посылает специально составленные пакеты целевому хосту, чтобы выявить ответы, позволяющие выяснить тип операционной системы на целевой машине. Безусловно, это не особенно скрытный, хотя и весьма эффективный способ, поскольку он подразумевает установление непосредственной связи с целевым хостом.

Пакеты, содержащиеся в файле перехвата `activeosfingerprinting.pcapng`, служат характерными примерами методики активного получения отпечатков операционных систем, инициируемых утилитой сканирования Nmap. В этом файле несколько пакетов получены в ответ на отправку утилитой Nmap ряда зондирующих пакетов с целью выявить в ответных пакетах характерную информацию, по которой можно будет идентифицировать тип операционной

системы целевого хоста. Утилита Nmap зафиксировала ответы на эти зондирующие пакеты и создала активный отпечаток операционной системы, который затем сравнила с информацией, хранящейся в базе данных для принятия решения.

ПРИМЕЧАНИЕ *Методики, применяемые в Nmap для активного получения отпечатка операционной системы, довольно сложные. Чтобы узнать больше, каким образом в Nmap выполняется активное получение отпечатка операционной системы, обратитесь за справкой к руководству по Nmap под названием Nmap Network Scanning, опубликованному в 2011 году его автором, Гордоном Лайеном по прозвищу "Федор".*

Манипулирование сетевым трафиком

Одна из самых главных идей, которые я попытался раскрыть в этой книге, заключается в том, что, анализируя подходящие пакеты, можно узнать немало о системе и ее пользователях. Следовательно, нет ничего удивительного в том, что злоумышленники нередко ищут возможность перехватить такие пакеты самостоятельно. Анализируя пакеты, сформированные системой, злоумышленник может узнать о типе операционной системы, применяемых в ней приложениях, учетных записях пользователей и много другой полезной для себя информации.

В этом разделе будут рассмотрены две методики, применяемые на уровне пакетов. С их помощью злоумышленник может воспользоваться заражением ARP-кеша для перехвата и фиксации целевого трафика или перехватом cookies-файлов протокола HTTP, с помощью которых можно проводить атаки по перехвату и подмене сеансов связи.

Заражение ARP-кеша

Файл перехвата `arppoison.pcapng`

Как обсуждалось в главе 7, "Протоколы сетевого уровня", протокол ARP служит для преобразования IP-адресов устройств в MAC-адреса в сети, а в главе 2, "Подключение к сети", пояснялось, как воспользоваться методикой заражения ARP-кеша для подключения к сети и перехвата сетевого трафика, исходящего из тех хостов, пакеты которых требуется проанализировать. Если заражение ARP-кеша применяется в благих целях, оно оказывает немалую помощь в диагностике сети. Но если эта методика употребляется со злым умыслом, то она превращается в летальную форму *атаки через посредника* (MITM), иначе *называемой атакой типа "человек посредине"*.

Совершая атаку через посредника, злоумышленник переадресует сетевой трафик между двумя хостами, чтобы перехватить или видоизменить пере-

даваемые данные. Имеется немало форм атак через посредника, включая подмену доменного имени и SSL-перехват. При заражении ARP-кеша составленные особым образом пакеты ARP вынуждают оба хоста действовать так, как будто они связаны друг с другом непосредственно, тогда как на самом деле они обмениваются данным через стороннего посредника, пересылающего им пакеты. Такое противозаконное применение обычных функциональных возможностей протокола ARP может делаться со злым умыслом.

Файл перехвата `arppoison.pcapng` содержит пример заражения ARP-кеша. Открыв этот файл, вы обнаружите, что перехваченный в нем сетевой трафик выглядит на первый взгляд нормально. Но если проанализировать пакеты из этого трафика, то можно заметить, что целевой хост, находящийся по адресу **`172.16.0.107`**, обращается к веб-ресурсу компании Google для выполнения поиска. В результате этого поиска появляется значительный трафик по протоколу HTTP, сочетающийся с некоторыми DNS-запросами.

Как известно, методика заражения ARP-кеша действует на втором уровне модели OSI, и если случайно бросить лишь беглый взгляд на пакеты в панели Packet List, то вряд ли удастся обнаружить какое-нибудь мошенничество. Поэтому в качестве вспомогательной меры необходимо добавить пару столбцов в панели Packet List, выполнив следующие действия.

1. Выберите команду Edit⇨Preferences из главного меню.
2. Щелкните на элементе Columns, который находится у левого края открывшегося окна Preferences.
3. Щелкните на кнопке со знаком "плюс" (+), чтобы добавить новый столбец.
4. Введите **`Source MAC`** (MAC-адрес источника) в области Title (Заголовок) и нажмите клавишу <Enter>.
5. Выберите вариант Hw src addr (resolved) (Аппаратный адрес отправителя (преобразованный)) из раскрывающегося списка Type.
6. Щелкните на вновь введенном столбце и перетащите его, расположив сразу после столбца **`Source`**.
7. Щелкните на кнопке со знаком "плюс" (+), чтобы добавить еще один новый столбец.
8. Введите **`Dest MAC`** (MAC-адрес места назначения) в области Title и нажмите клавишу <Enter>.
9. Выберите вариант Hw dest addr (resolved) (Аппаратный адрес получателя (преобразованный)) из раскрывающегося списка Type.
10. Щелкните на вновь введенном столбце и перетащите его, расположив сразу после столбца **`Destination`**.
11. Щелкните на кнопке OK, чтобы применить внесенные изменения.

Выполнив описанные выше действия, вы должны увидеть на экране то же самое, что и на рис. 12.8. Теперь в вашем распоряжении должны быть два дополнительных столбца, где отображаются MAC-адреса отправителя и получателя из анализируемых пакетов.

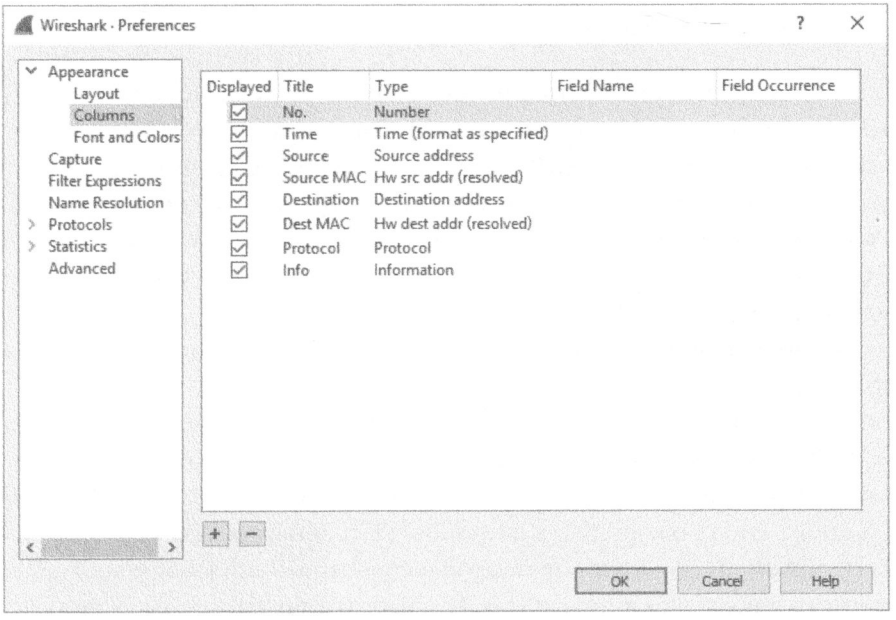


Рис. 12.8. Экран конфигурирования столбцов с двумя вновь введенными столбцами аппаратных адресов отправителя и получателя

Если у вас по-прежнему активизирован режим преобразования MAC-адресов, вы должны увидеть, что обменивающиеся данными устройства обладают MAC-адресами, обозначающими сетевое оборудование компаний Dell и Cisco. Эти сведения очень важно запомнить, поскольку, прокрутив дальше список перехваченных пакетов, можно заметить, что они изменяются в пакете 54, где происходит необычный трафик по протоколу ARP между хостом с оборудованием компании Dell (целевым хостом) и вновь внедренным хостом с оборудованием компании HP (злоумышленником), как показано на рис. 12.9.

No.	Time	Source	Source MAC	Destination	Dest MAC	Protocol	Info
54	4.171500	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	(1) Dell_c0:56:f0	ARP	Who has 172.16.0.107? Tell 172.16.0.1
(2) 55	0.000053	Dell_c0:56:f0	Dell_c0:56:f0	HewlettP_bf:91:ee	HewlettP_bf:91:ee	ARP	172.16.0.107 is at 00:21:70:c0:56:f0
56	0.000013	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	Dell_c0:56:f0	ARP	(3) 172.16.0.1 is at 00:25:b3:bf:91:ee

Рис. 12.9. Необычный трафик по протоколу ARP между сетевыми устройствами компаний Dell и HP

Прежде чем продолжить дальше, обратите внимание на конечные точки рассматриваемого здесь соединения, перечисленные в табл. 12.3.

Таблица 12.3. Конечные точки контролируемого соединения

Роль	Тип устройства	IP-адрес	MAC-адрес
Целевой хост	Dell	`172.16.0.107`	`00:21:70:c0:56:f0`
Маршрутизатор	Cisco	`172.16.0.1`	`00:26:0b:31:07:33`
Злоумышленник	HP	Неизвестен	`00:25:b3:bf:91:ee`

Но в чем необычность анализируемого здесь трафика? Как пояснялось при обсуждении протокола ARP в главе 7, "Протоколы сетевого уровня", имеются два основных типа пакетов ARP с запросами и ответами. В частности, пакет с запросом посылается как широковещательный всем хостам, чтобы обнаружить в сети машину, имеющую MAC-адрес, связанный с конкретным IP-адресом. На это искомая машина посылает запрашивающему устройству ответ в одноадресатном пакете. Принимая эти сведения во внимание, можно выяснить особенности данной последовательности обмена данными, как показано на рис. 12.9.

Прежде всего, пакет 54 содержит ARP-запрос, посылаемый злоумышленником (с MAC-адресом `00:25:b3:bf:91:ee`), в виде одноадресатного пакета непосредственно целевому хосту (с MAC-адресом `00:21:70:c0:56:f0`) ❶. Запрос данного типа должен быть широковещательным для всех хостов в сети, но анализируемый здесь запрос направляется только целевому хосту. Следует также иметь в виду, что в нем указан IP-адрес маршрутизатора, а не собственный адрес злоумышленника, несмотря на то, что данный пакет посылается этим злоумышленником с MAC-адресом его устройства, включенным в ARP-заголовок.

После данного пакета следует ответ от целевого хоста злоумышленнику, содержащему сведения о своем MAC-адресе ❷. Анализируемое здесь жульничество происходит в пакете 56, где злоумышленник посылает пакет целевому хосту, содержащий незатребованный ARP-ответ, в котором сообщается, что устройство с IP-адресом `172.16.0.1` находится по MAC-адресу `00:25:b3:bf:91:ee` ❸. Но дело в том, что IP-адресу `172.16.0.1` соответствует не MAC-адрес `00:25:b3:bf:91:ee`, а MAC-адрес `00:26:0b:31:07:33`. Об этом известно из проведенного ранее анализа пакетов, которыми маршрутизатор, расположенный по адресу `172.16.0.1`, обменивался с целевым хостом. А поскольку протокол ARP, по существу, не является защищенным и допускает незатребованные обновления ARP-таблицы, то целевой хост пошлет теперь свой трафик не маршрутизатору, как это должно быть, а злоумышленнику.

Как только целевой хост и маршрутизатор удастся ввести в заблуждение, обмен данными между ними будет происходить через хост злоумышленника, как показано на рис. 12.10.

ПРИМЕЧАНИЕ *Анализируемые здесь пакеты перехвачены из целевой машины и поэтому не дают полного представления о происходящем. Чтобы атака злоумышленника возымела действие, он должен послать ту же самую последовательность пакетов маршрутизатору с целью вынудить его принять компьютер злоумышленника за целевой хост. Но для того чтобы обнаружить эти пакеты, придется осуществить еще один перехват сетевого трафика, исходящего из маршрутизатора (или хоста злоумышленника).*

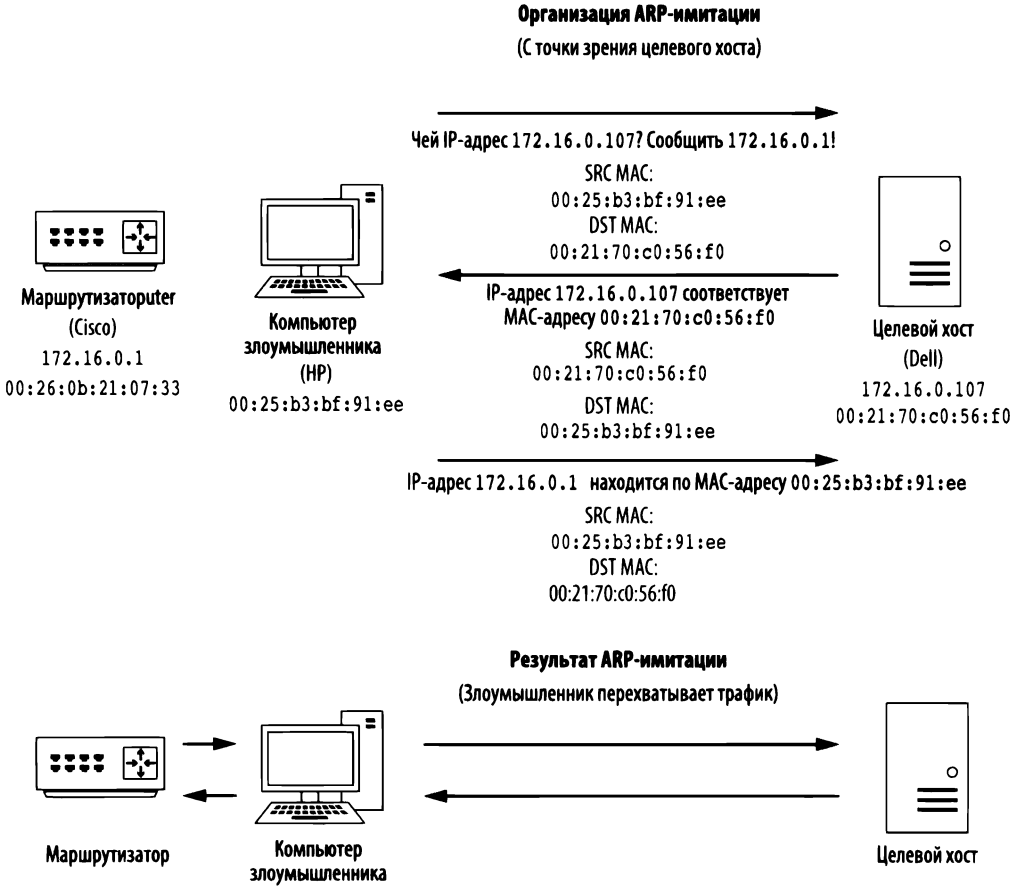


Рис. 12.10. Заражение ARP-кеша при атаке через посредника

Пакет 57 подтверждает удачный исход данной атаки. Если сравнить этот пакет с тем, что был послан прежде таинственного трафика по протоколу ARP (например, с пакетом 40; рис. 12.11), то можно заметить, что IP-адрес удаленного сервера (компании Google) остался тем же самым ❷ и ❹, тогда как MAC-адрес целевого хоста изменился ❶ и ❸. Такое изменение MAC-адреса

свидетельствует о том, что сетевой трафик теперь переадресовывается через атакующего злоумышленника, прежде чем достичь маршрутизатора.

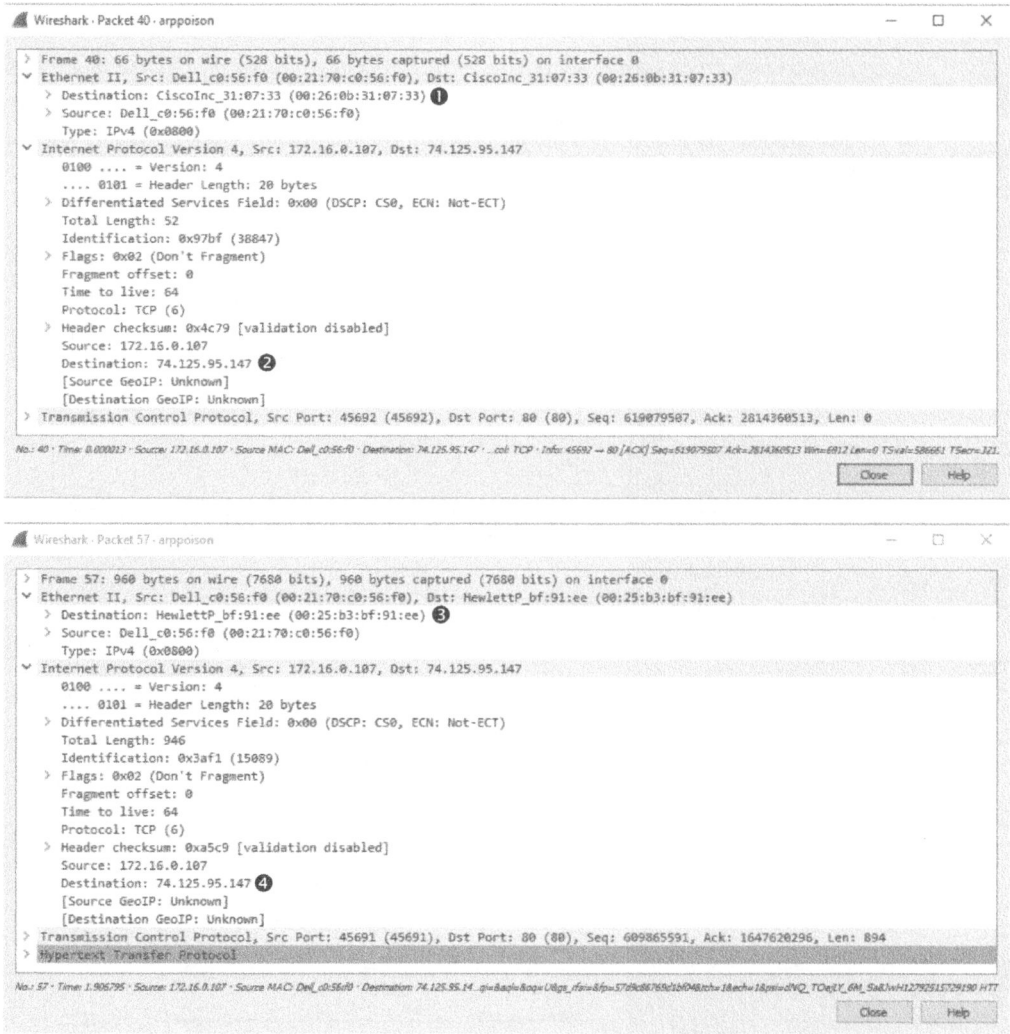


Рис. 12.11. Изменение MAC-адреса целевого хоста свидетельствует об успешном исходе данной атаки через посредника

Анализируемая здесь атака настолько незаметна, что обнаружить ее крайне сложно. Чтобы обнаружить ее, требуется помощь системы обнаружения вторжений, специально настроенной на решение подобной задачи, или программы, работающей на устройствах, предназначенных для обнаружения внезапных изменений в элементах ARP-таблицы. А поскольку вам, скорее всего, придется пользоваться методикой заражения ARP-кеша для перехвата

пакетов в анализируемых сетях, то вы должны знать, как эта методика может быть использована против вас.

Перехват сеансов связи

Файл перехвата **sessionhijacking.pcapng**

Рассмотрев порядок применения со злым умыслом заражения ARP-кеша, продемонстрируем методику *перехвата сеансов связи*, где оно выгодно применяется. При перехвате сеансов связи злоумышленник завладевает копией cookie-файла сеанса связи по протоколу HTTP, как будет показано ниже, чтобы с помощью этого файла выдать себя за другого пользователя. С этой целью злоумышленник может воспользоваться заражением ARP-кеша, чтобы перехватить сетевой трафик целевого хоста и извлечь соответствующую информацию из сеансового cookie-файла. Этой информацией злоумышленник может затем воспользоваться для доступа к веб-приложению целевого хоста как его авторизованный пользователь.

Начнем рассмотрение данного сценария с файла перехвата `sessionhijacking.pcapng`, содержащего сетевой трафик целевого хоста, находящегося по адресу **172.16.16.164** и обменивающегося данными с веб-приложением, действующим по адресу **172.16.16.181**. При этом целевой хост невольно стал жертвой злоумышленника, расположенного по адресу **172.16.16.154** и активно перехватывавшего обмен данными целевого хоста с веб-приложением. Эти пакеты были собраны с точки зрения веб-сервера, которая вероятнее всего будет точно такая же, как и у системы защиты инфраструктуры сервера от атак типа перехвата сеансов связи.

ПРИМЕЧАНИЕ *Упоминаемое здесь веб-приложение называется DVWA (Damn Vulnerable Web Application – Чертовски уязвимое веб-приложение). Оно намеренно сделано уязвимым ко многим видам атак и зачастую применяется в качестве учебного средства. Если у вас возникнет желание ознакомиться подробнее с веб-приложением DVWA, обратитесь по адресу* **http://www.dvwa.co.uk/**.

Анализируемый здесь сетевой трафик состоит, главным образом, из двух диалогов. Первый диалог представляет собой обмен данными между целевого хостом и веб-сервером и может быть селектирован фильтром **ip.addr == 172.16.16.164 && ip.addr == 172.16.16.181**. Этот обмен данными представлен обычным трафиком просмотра веб-содержимого, в котором нет ничего особенного. В анализируемых здесь запросах особый интерес вызывает cookie-значение. Так, если проанализировать запрос по методу `GET` (например, в пакете 14), то можно обнаружить, что cookie-значение

отображено на панели Packet Details (рис. 12.12). В данном случае cookie-значение **ncobrqrb7fj2a2sinddtk567q4** обозначает идентификатор сеанса связи PHPSESSID ❶.

Рис. 12.12. Просмотр сеансового cookie-значения целевого хоста

Веб-сайты пользуются cookie-файлами, чтобы поддерживать сеанс связи с отдельными хостами. Когда на веб-сайте появится новый посетитель, ему будет присвоен однозначно определяющий его идентификатор сеанса связи (PHPSESSID). Многие веб-приложения ожидают до тех пор, пока пользователь не пройдет аутентификацию со своим идентификатором сеанса связи, после чего в базе данных создается запись, где этот идентификатор распознается как обозначающий аутентифицированный сеанс связи. Любой пользователь с этим идентификатором сможет получить доступ к веб-приложению, пройдя такую аутентификацию. Безусловно, разработчикам необходимо обеспечить однозначность идентификатора, формируемого для каждого пользователя. Но такой способ обработки идентификаторов небезопасен, поскольку он дает злонамеренному пользователю возможность похитить идентификатор другого пользователя, чтобы выдать себя за него. И хотя имеются специальные методы, позволяющие предотвратить атаки типа перехвата сеанса связи, тем не менее, многие веб-сайты, включая и DVWA, остаются по-прежнему уязвимыми к подобным атакам.

Целевой хост даже не подозревает, что его сетевой трафик перехватывается злоумышленником или же у этого злоумышленника имеется доступ к сеансовому cookie-файлу (рис. 12.12). И в этом случае злоумышленнику остается лишь установить связь с веб-сервером, используя перехваченное

cookie-значение. Такую задачу можно решить с помощью определенного рода прокси-серверов, но это еще легче сделать, используя такие подключаемые к браузеру модули, как Cookie Manager для Chrome. С помощью этого подключаемого модуля злоумышленник может указать значение идентификатора `PHPSESSID`, полученное из сетевого трафика целевого хоста, как показано на рис. 12.13.

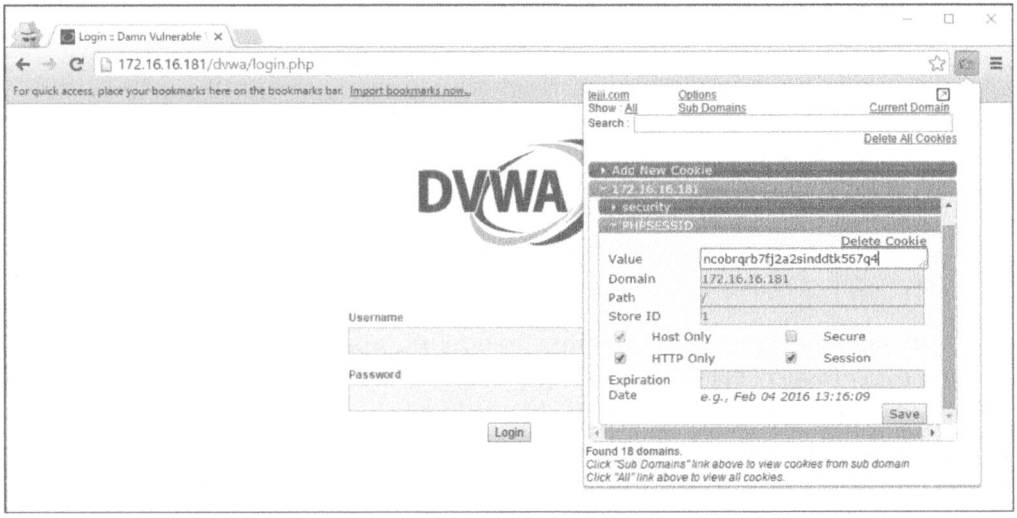


Рис. 12.13. Применение подключаемого модуля Cookie Manager злоумышленником с целью выдать себя за целевой хост

Если удалить фильтр, применявшийся ранее к файлу перехвата, и прокрутить полученный список пакетов вниз, то можно обнаружить IP-адрес компьютера злоумышленника, устанавливающегося связь с веб-сервером. Просмотр этой связи можно выявить с помощью следующего фильтра:

```
ip.addr == 172.16.16.154 && ip.addr == 172.16.16.181
```

Прежде чем углубляться в дальнейший анализ, добавьте еще один столбец на панели Packet List, чтобы отображать в нем cookie-значения. Если вы добавили столбцы в предыдущем разделе, посвященном заражению ARP-кеша, удалите их, прежде чем добавлять этот столбец. После этого выполните действия, описанные в предыдущем разделе, чтобы добавить новый столбец, исходя из имени поля **`http.cookie_pair`**, и разместите его справа от столбца **`Destination`**. В итоге содержимое экрана должно выглядеть так же, как и на рис. 12.14.

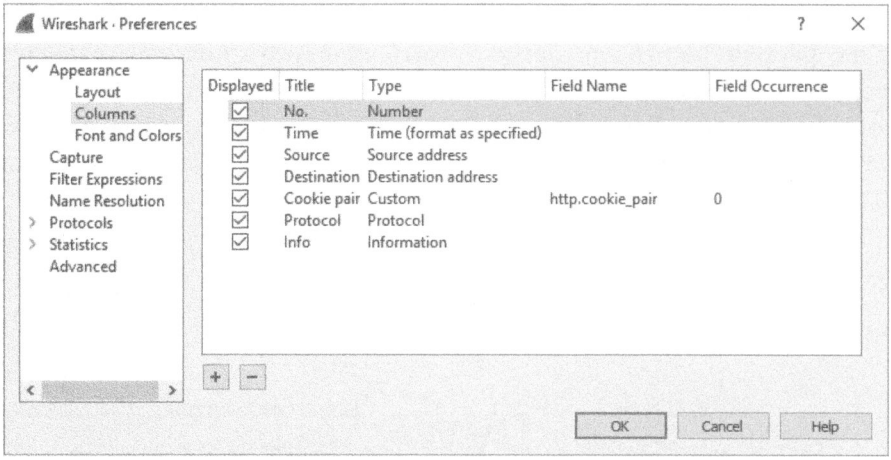


Рис. 12.14. Экран конфигурирования столбцов для исследования перехвата сеансов связи

Итак, сконфигурировав столбцы по-новому, видоизмените фильтр, чтобы отображать только HTTP-запросы, поскольку обмен данными по протоколу TCP для этой цели не подходит. Новый фильтр должен выглядеть так, как показано ниже. Пакеты, селектированные с помощью этого фильтра, приведены на рис. 12.15.

```
(ip.addr==172.16.16.154 && ip.addr==172.16.16.181)
        && (http.request.method || http.response.code)
```

No.	Time	Source	Destination	Cookie pair	Protocol	Info
77	16.563004	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmsjtgrd71	HTTP	❶GET /dvwa/ HTTP/1.1
79	16.565584	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 302 Found❷
80	16.570187	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmsjtgrd71	HTTP	❸GET /dvwa/login.php HTTP/1.1
81	16.575123	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html)❹
115	60.040166	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	❺GET /dvwa/ HTTP/1.1
118	60.042241	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html❻
120	64.292056	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	❼GET /dvwa/setup.php HTTP/1.1
122	64.293401	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html❽

Рис. 12.15. Анализ этих пакетов показывает, что злоумышленник выдает себя за пользователя целевого хоста

А теперь проанализируем обмен данными между компьютером злоумышленника и сервером. В первых четырех пакетах злоумышленник запрашивает каталог `/dvwa/` ❶ и получает в ответ код **302**, что считается обычным для веб-сервера способом переадресации посетителей по другим URL. В данном случае злоумышленник переадресовывается на страницу регистрации по адресу `/dvwa/login.php` ❷. Компьютер злоумышленника запрашивает страницу регистрации ❸, которая успешно возвращается ❹. В обоих запросах применяется идентификатор сеанса связи `lup70ajeuodkrhrvbmsjtgrd71`.

Далее следует новый запрос каталога `/dvwa/`, но на этот раз обратите внимание на другой идентификатор сеанса связи `ncobrqrb7fj2a2sinddtk567q4` ❺, который теперь совпадает с использовавшимся ранее идентификатором целевого хоста. Это означает, что злоумышленник манипулировал сетевым трафиком, чтобы воспользоваться украденным идентификатором. Вместо переадресации на страницу регистрации в ответ на отправленный запрос присылается код состояния **200** по протоколу HTTP, и страница доставляется так, как будто ее должен увидеть пользователь, аутентифицированный на целевом хосте ❻. Далее злоумышленник запрашивает другую страницу по адресу `dvwa/setup.php`, используя идентификатор целевого хоста ❼. И эта страница возвращается успешно ❽. Таким образом, злоумышленник просматривает вебсайт DVWA, как будто он действительно аутентифицирован как пользователь этого целевого хоста. И все это ему удается сделать, не зная даже ни имени пользователя, ни пароля.

Рассмотренный выше сценарий служит еще одним примером того, как злоумышленник может превратить анализ пакетов в наступательное средство. В общем, благоразумнее допустить, что если злоумышленник способен перехватывать и проанализировать пакеты, связанные с определенным обменом данными по сети, то это может привести к определенного рода зловредным действиям. И это одна из причин, по которым специалисты в области безопасности ратуют за то, чтобы защищать передаваемые данные путем шифрования.

Вредоносное программное обеспечение

Если совершенно законное программное обеспечение может быть использовано со злым умыслом, то термином *вредоносное программное обеспечение (malware)* принято обозначать код, специально написанный со злыми намерениями. Вредоносное программное обеспечение может принимать самые разные виды и формы, включая самораспространяющиеся "черви" и "троянские кони", маскирующиеся под вполне законные программы. С точки зрения защиты сетей бо́льшая часть вредоносного программного обеспечения неизвестна и не обнаруживается до тех пор, пока его зловредные действия не будут выявлены и проанализированы. Процесс такого анализа состоит из нескольких стадий, включая и ту, что специально предназначена для поведенческого анализа образцов обмена данными, совершаемого вредоносным программным обеспечением по сети. Иногда такой анализ проводится в специальной судебно-экспертной лаборатории по восстановлению алгоритма работы вредоносного программного обеспечения. Но чаще всего он проводится в естественных условиях, когда специалист по вопросам безопасности обнаруживает заражение вирусом в анализируемой им сети. В этом разделе разбирается ряд примеров настоящего вредоносного программного обеспечения и анализируется его поведение на уровне наблюдаемых пакетов.

Операция "Аврора"

Файл перехвата **aurora.pcapng**

В январе 2010 года был выявлен вредоносный код под названием операция "Аврора" (Operation Aurora), эксплуатировавший неизвестную ранее уязвимость в браузере Internet Explorer. Эта уязвимость позволила злоумышленникам получить удаленный контроль над целевыми серверами, среди прочего, компании Google.

Чтобы этот вредоносный код был выполнен, пользователю было достаточно посетить веб-сайт, используя уязвимую версию браузера Internet Explorer. И тогда злоумышленники получали немедленный доступ к машине пользователя с теми же правами, что и у зарегистрированного на ней пользователя[1]. Чтобы как-то обмануть или чем-нибудь прельстить жертву, злоумышленники применяли *целенаправленный фишинг*, состоявший в том, что они посылали по электронной почте сообщение с целью вынудить получателя щелкнуть на ссылке, ведущей к вредоносному веб-сайту.

Что же касается операции "Аврора", то мы выполнили перехват пакетов начиная с момента, когда целевой пользователь только щелкал на ссылке в сообщении целенаправленного фишинга, которое он получал по электронной почте. Полученные в итоге пакеты сохранялись в файле перехвата `aurora.pcapng`.

Анализируемый здесь перехваченный трафик начинается с трехэтапного процесса установки связи по протоколу TCP между целевого хостом, находящимся по адресу **192.168.100.206**, и компьютером злоумышленника, расположенным по адресу **192.168.100.202**. Первоначальное соединение устанавливается через порт **80**. Это может заставить поверить, что сетевой трафик осуществляется по протоколу HTTP. И такое предположение подтверждается в четвертом пакете, где содержится HTTP-запрос каталога `/info` ❶ по методу `GET`, как показано на рис. 12.16.

Как показано на рис. 12.17, компьютер злоумышленника подтверждает прием запроса по методу `GET` и посылает в ответ код состояния **302** (Moved Temporarily – Документ временно перемещен) в пакете 6 ❶. Этот код состояния обычно применяется для переадресации браузера на другую страницу, как в данном случае. Наряду с кодом состояния **302** в поле **Location** заголовка указано значение **/info?rFfWELUjLJHpP** ❷.

[1] Этот пример демонстрирует тот факт, что учетные записи обычных пользователей на целевом хосте не должны обладать правами администратора. Тогда заражение всей системы станет в принципе невозможным, поскольку для записи на системный диск и в реестр требуются права администратора. – *Примеч. ред.*

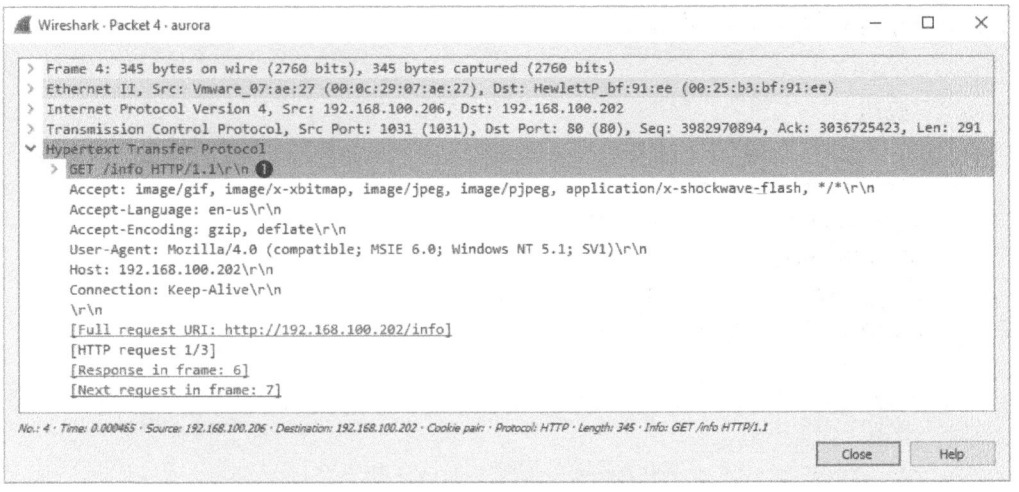


Рис. 12.16. Целевой хост делает запрос каталога **`/info`** *по методу* **`GET`**

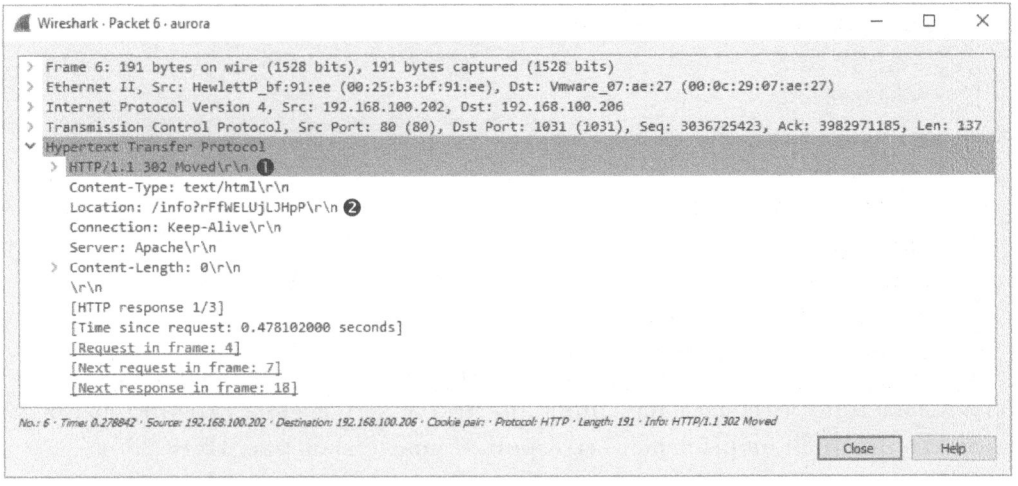


Рис. 12.17. В данном пакете браузер клиента переадресовывается на другую страницу

Получив пакет HTTP с кодом состояния **302**, клиент инициирует еще один запрос по методу `GET` по URL со значением **`/info?rFfWELUjLJHpP`** в пакете 7, на что получает подтверждение в пакете 8 типа `ACK`. После пакета `ACK` следует несколько пакетов, представляющих данные, передаваемые злоумышленником целевому хосту. Чтобы более тщательно проанализировать эти данные, щелкните правой кнопкой мыши на одном из пакетов в потоке данных (например, на пакете 9) и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В этом потоке вывода данных можно обнаружить первоначальный запрос по методу `GET`, переадресацию по коду состояния **302**, а также второй запрос по методу `GET`, как показано на рис. 12.18.

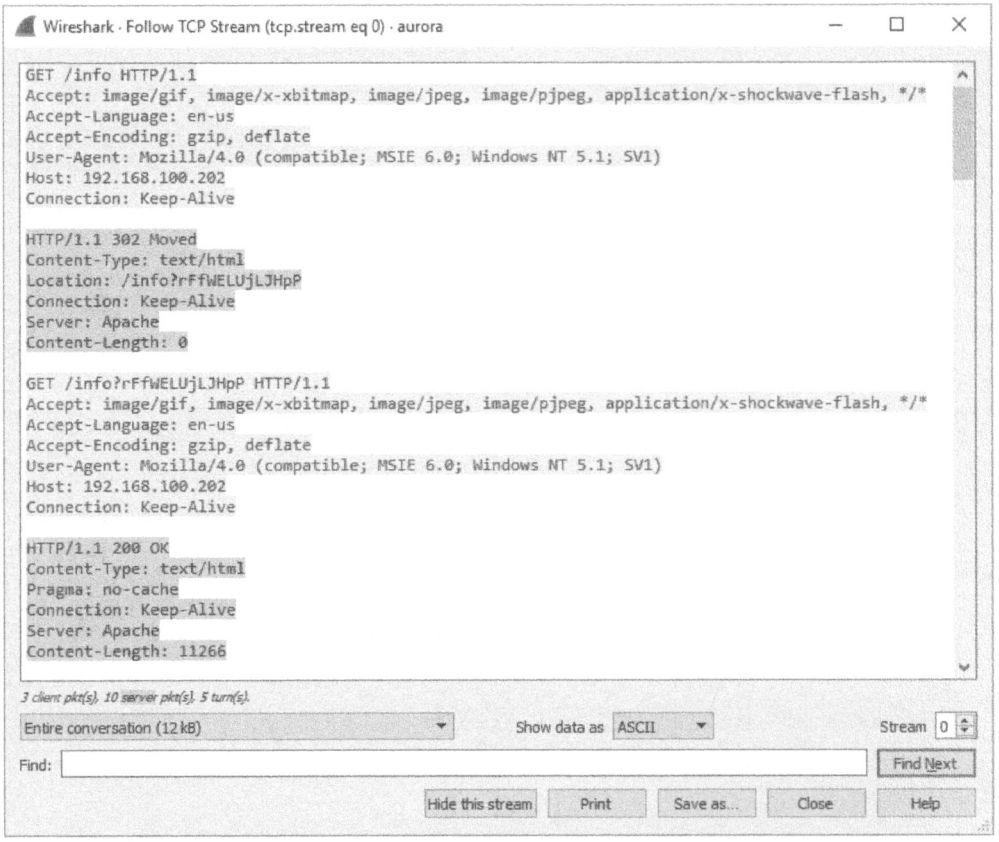


Рис. 12.18. Поток данных, передаваемый клиенту

После этого дело начинает принимать совершенно необычный оборот. Злоумышленник отвечает на запрос по методу `GET` весьма странно выглядящим содержимым, первая часть которого приведена на рис. 12.19.

Это содержимое состоит из целого ряда случайных чисел и букв в дескрипторе `<script>` ❶. Этот дескриптор применяется в HTML-документе для обозначения исходного кода на языке сценариев высокого уровня, предназначенного для выполнения на стороне HTTP-клиента. Как правило, в дескрипторе `<script>` указываются операторы, составляющие исполняемый код сценария. Но бессмысленное содержимое в данном случае зашифровано, чтобы скрыть сам исполняемый код от обнаружения. А поскольку нам известно, что анализируемый здесь трафик служит для эксплуатации уязвимостей в системе безопасности атакуемой службы, то можно предположить, что запутанный текст в дескрипторе `<script>` содержит шестнадцатеричное заполнение и код запуска оболочки со злым умыслом.

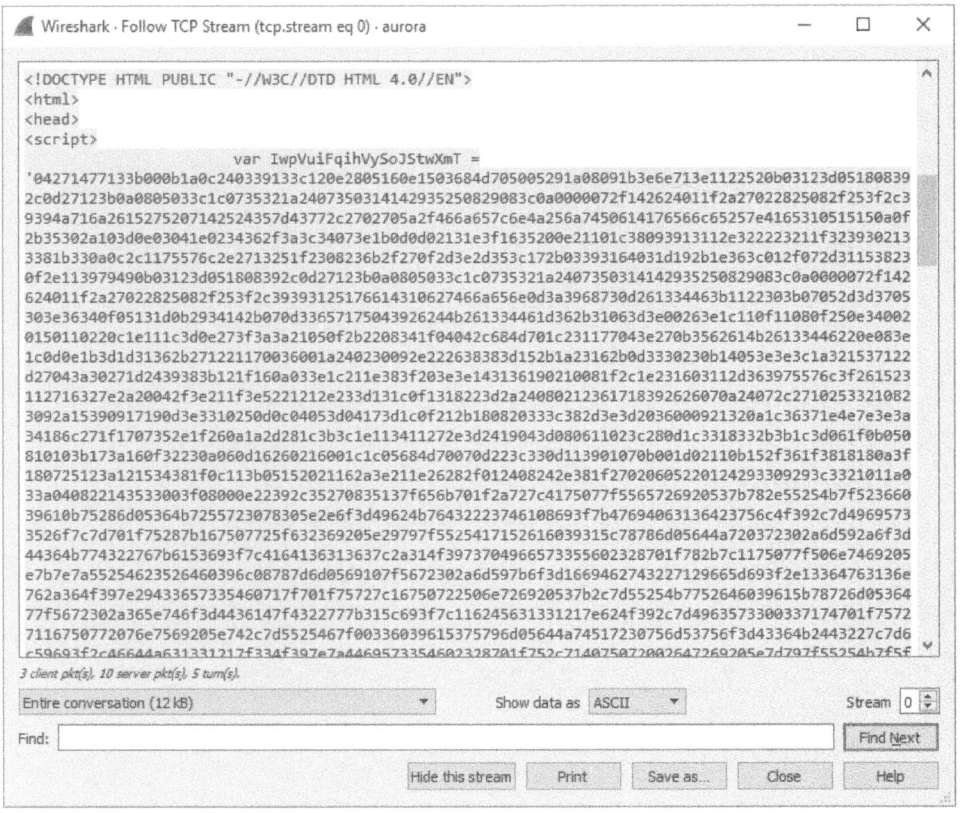


Рис. 12.19. Содержимое дескриптора ***<script>*** *оказывается зашифрованным*

ПРИМЕЧАНИЕ *Запутывание сценариев является весьма распространенной методикой, применяемой во вредоносном программном обеспечении с целью избежать обнаружения и скрыть зловредное содержимое. И хотя распутывание подобных сценариев выходит за рамки данной книги, умение делать это приобретается с опытом в ходе тщательного анализа обмена данными по сети со злым умыслом. Многие опытные специалисты по анализу вредоносного программного обеспечения способны мгновенно распознавать зловредные сценарии, бросив беглый взгляд на перехваченный сетевой трафик. Если вы желаете испытать себя, попробуйте распутать вредоносный сценарий, обнаруженный в данном примере.*

Во второй части содержимого, отправленного злоумышленником и приведенного на рис. 12.20, нам, наконец-то, удается обнаружить хотя бы немного удобочитаемого текста. Даже не имея особого опыта программирования, видно, что это исходный тест программы, где выполняется синтаксический анализ символьной строки на основании значений нескольких переменных.

И этот последний фрагмент текста расположен прямо перед закрывающим дескриптором </script>.

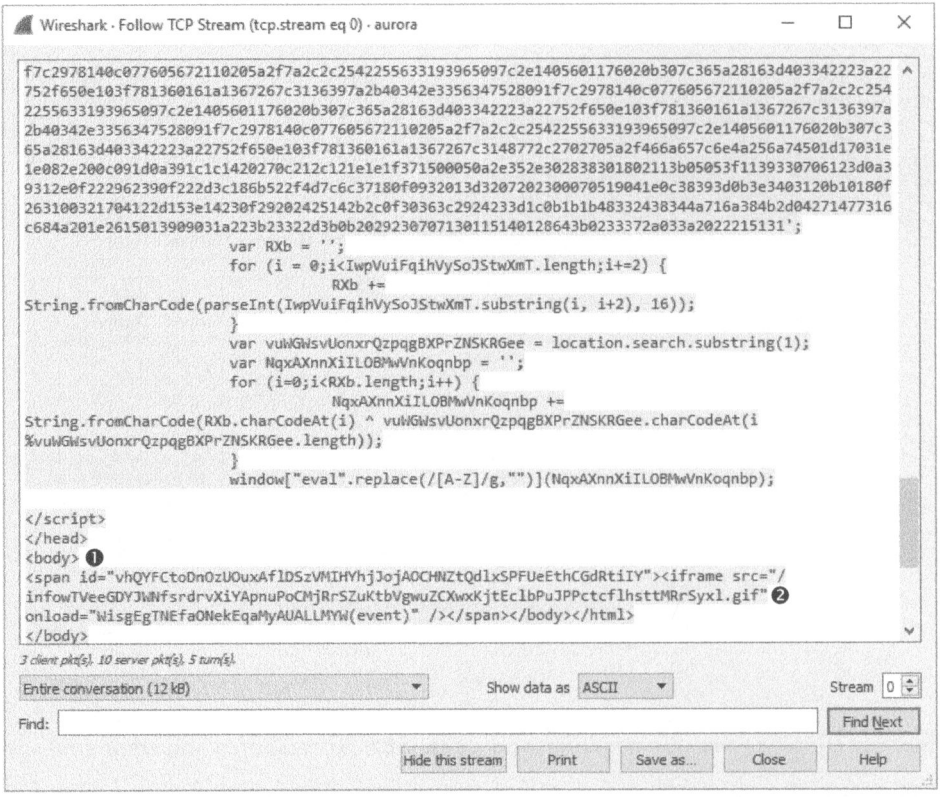


Рис. 12.20. Эта часть содержимого, отправленного сервером, содержит удобочитаемый текст и подозрительный встраиваемый фрейм `iframe`

Последний фрагмент данных, посылаемых злоумышленником целевому хосту, состоит из двух частей (см. рис. 12.20). Первая часть заключена в дескриптор `<span id="vhQYFCtoDnOzUOuxAflDSzVMIHYhjJojAOCHNZtQdlxSPFUeEthCGdRtiIY">` ❶, а вторая — в дескриптор `<iframe src="/infowTVeeGDYJWNfsrdrvXiYApnuPoCMjRrSZuKtbVgwuZCXwxKjtEclbPuJPPctcflhsttMRrSyxl.gif" onload="WisgEgTNEfaONekEqaMyAUALLMYW(event)" />` ❷. И это содержимое может свидетельствовать о злом умысле, если принять во внимание слишком длинные и произвольные строки неудобочитаемого и потенциально запутанного текста.

Часть содержимого, заключенная в дескриптор `<span>`, образует *встраиваемый фрейм*. Злоумышленники часто пользуются этим способом для встраивания дополнительного и не предполагаемого содержимого на HTML-страницу. А в дескрипторе `<iframe>` образуется *внутристрочный фрейм*, чтобы его не смог обнаружить пользователь. В данном случае в дескрипторе `<iframe>`

содержится ссылка на необычно именуемый файл формата GIF. Как показано на рис. 12.21, когда браузер целевого хоста обнаруживает ссылку на этот файл, он делает его запрос по методу `GET` в пакете 21 ❶, и в ответ немедленно посылается запрашиваемый файл формата GIF ❷.

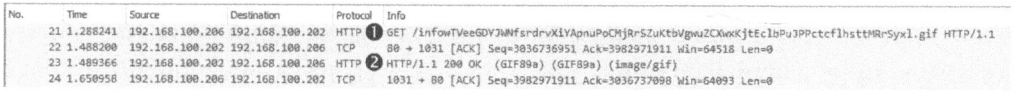


Рис. 12.21. Файл формата GIF, указанный в дескрипторе **<iframe>**, *запрашивается и загружается целевым хостом*

Самое необычное в анализируемом здесь перехваченном трафике происходит в пакете 25, когда целевой хост инициирует обратное соединение с компьютером злоумышленника через порт **4321**. Просмотр этого второго потока обмена данными на панели Packet Details мало что дает, и поэтому нам придется снова отследить поток TCP, чтобы получить более ясное представление об обмениваемых данных. Результат, выводимый в окне Follow TCP Stream, приведен на рис. 12.22.

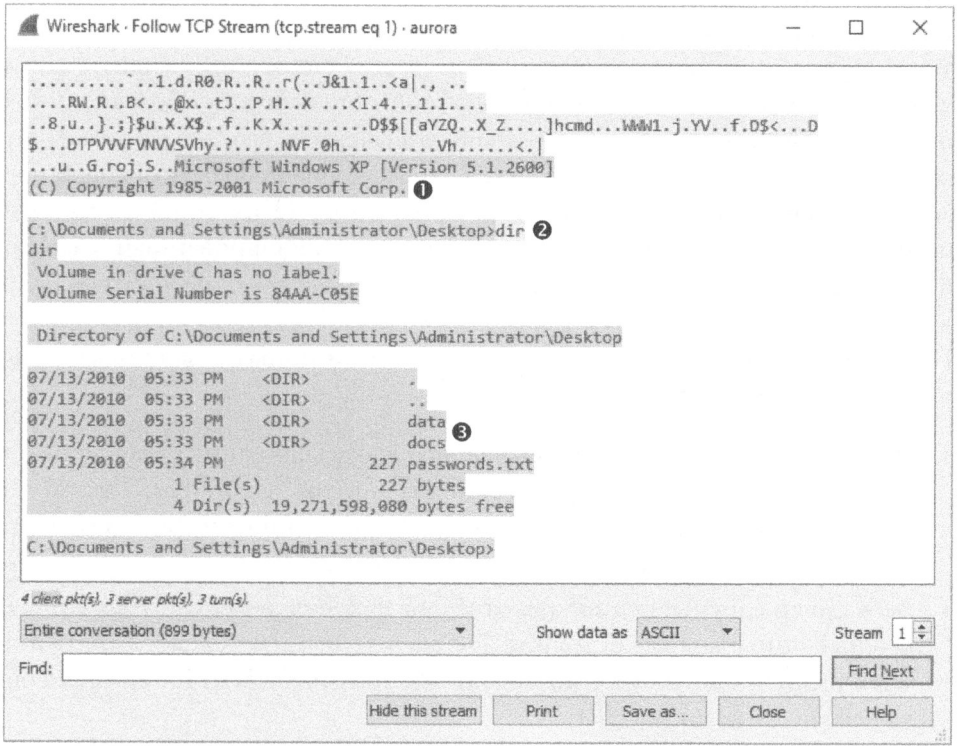


Рис. 12.22. Злоумышленник взаимодействует с командной оболочкой Windows через данное соединение

В данном окне наблюдается нечто такое, что сразу же должно вызвать тревогу: запуск команд из командной оболочки Windows ❶. Эта оболочка посылает данные выполнения команд на целевом хосте серверу, указывая на то, что попытка злоумышленника эксплуатировать уязвимость целевого хоста увенчалась успехом и полезная информация была получена злоумышленником. Целевой хост передал данные выполнения команд в командной оболочке обратно компьютеру злоумышленника сразу же после запуска вредоносного кода. В этом перехваченном трафике можно обнаружить взаимодействие злоумышленника с целевым хостом, которое заключается в выполнении команды **dir** ❷ для просмотра содержимого текущего каталога на машине целевого хоста ❸.

Если предположить, что в результате данной попытки воспользоваться уязвимостью целевого хоста злоумышленнику удалось подключиться к системному процессу, выполняющемуся с правами администратора, и запустить вредоносный код, то в результате злоумышленник может сделать с целевого хостом, все, что угодно. Достаточно один раз щелкнуть на ссылке кнопкой мыши, чтобы за доли секунды полностью отдать целевой хост во власть злоумышленника.

Подобные попытки воспользоваться уязвимостью целевого хоста обычно зашифрованы и их код трудно распознать. Поэтому при распространении по сети система обнаружения вторжений не всегда может их выявить. Следовательно, не зная заранее характер подобной попытки или образец ее кода, очень трудно без дополнительного анализа выяснить, что же на самом деле происходит в системе целевого хоста. Правда, в анализируемом здесь перехваченном трафике все же можно увидеть некоторые характерные признаки наличия вредоносного кода. К их числу относится зашифрованный и запутанный код в дескрипторах `<script>` и `<iframe>`, а также текстовый результат работы командной оболочки.

Ниже приведено краткое описание этапов при попытке воспользоваться уязвимостью в операции "Аврора".

- Целевой хост получает от атакующего злоумышленника сообщение по электронной почте, которое выглядит вполне законным, щелкает на ссылке в этом сообщении и посылает обратно запрос по методу `GET` протокола HTTP на вредоносный сайт злоумышленника.
- Веб-сервер злоумышленника в ответ выдает целевому хосту команду переадресации с кодом состояния **302**, на что браузер целевого хоста автоматически выдает запрос по методу `GET` протокола HTTP, переадресовывая его по указанному URL.
- Веб-сервер злоумышленника передает целевому хосту веб-страницу, содержащую запутанный код сценария на языке JavaScript, включающий в

себя попытку воспользоваться уязвимостью целевого хоста и встраиваемый фрейм, содержащий ссылку на запрашиваемый файл с изображением в формате GIF.

- Переданный ранее запутанный код сценария JavaScript распутывается при визуализации страницы в браузере целевого хоста, а затем выполняется на его машине с целью выявить уязвимость браузера Internet Explorer.
- Как только уязвимость будет выявлена, запустится на выполнение запутанный код, скрытый в переданной полезной информации. В результате будет открыт новый сеанс связи целевого хоста с компьютером злоумышленника через порт **4321**.
- В результате злоумышленник получает полный контроль над целевым хостом через команды, передаваемые по сети и выполняемые командной оболочкой, которая была запущена из вредоносного кода.

С точки зрения защиты сети анализируемый здесь файл перехвата может быть использован для создания сигнатуры, с помощью которой системе обнаружения вторжений удастся в дальнейшем выявлять случаи проявления подобного рода атак. С одной стороны, можно вычленить незапутанную часть перехваченного трафика, например, представленный в виде текста исходный код программы, расположенный в конце запутанного текста в дескрипторе `<script>`. А с другой стороны, можно написать сигнатуру для всего сетевого трафика по протоколу HTTP, в котором выполняется переадресация с кодом состояния **302** на веб-сайт, в URL которого присутствует слово **info**. Такая сигнатура потребует дополнительной настройки, чтобы стать полезной в условиях реальной эксплуатации, хотя ее написание и послужит неплохим началом. Не следует, однако, забывать, что сигнатуры можно побороть. Так, если злоумышленник внесет незначительные изменения в рассмотренные выше символьные строки или доставит вредоносный код через другой механизм, то созданные сигнатуры могут оказаться бесполезными. Это означает, что извечная борьба нападающих и защитников не прекратится никогда.

ПРИМЕЧАНИЕ *Умение создавать сигнатуры из образцов вредоносного трафика имеет решающее значение для тех, кто пытается защитить свою сеть от неизвестных угроз. Анализ перехваченного трафика, аналогичный описанному в этом разделе, дает отличную возможность выработать в себе навыки написания подобных сигнатур. Чтобы узнать больше об обнаружении незаконного вторжения и сигнатурах совершаемых атак, посетите веб-сайт проекта Snort по адресу* **http://www.snort.org/**.

Троянская программа удаленного доступа

Файл перехвата **ratinfected.pcapng**

До сих пор мы исследовали события, заранее зная хотя бы что-нибудь о происходящем. И хотя это отличный способ изучить атаки, на самом деле он далек от практики. Ведь в большинстве реальных сценариев те, кому поручено защищать сети, не будут анализировать каждый пакет, проходящий по сети. Вместо этого они воспользуются определенной формой системы обнаружения вторжений, которая должна предупредить их об аномалиях в сетевом трафике, который требует дальнейшего анализа на основании предопределенных сигнатур совершаемых атак.

В следующем примере мы начнем с простого предупреждения, взяв на себя роль настоящего аналитика. В данном случае система обнаружения вторжений формирует приведенное ниже предупреждение.

```
[**] [1:132456789:2] CyberEYE RAT Session Establishment [**]
[Classification: A Network Trojan was detected] [Priority: 1]
07/18-12:45:04.656854 172.16.0.111:4433 -> 172.16.0.114:6641
TCP TTL:128 TOS:0x0 ID:6526 IpLen:20 DgmLen:54 DF
***AP*** Seq: 0x53BAEB5E Ack: 0x18874922 Win: 0xFAF0 TcpLen: 20
```

На следующей стадии просматривается приведенное ниже правило сигнатуры, инициирующее данное предупреждение.

```
alert tcp any any -> $HOME_NET any
    (msg:"CyberEYE RAT Session Establishment";
content:"|41 4E 41 42 49 4C 47 49 7C|";
    classtype:trojan-activity; sid:132456789; rev:2;)
```

Это правило установлено для выдачи предупреждения всякий раз, когда обнаруживается пакет, поступающий из внутренней сети с шестнадцатеричным содержимым **41 4E 41 42 49 4C 47 49 7C**, которое преобразуется в удобочитаемую фразу **ANA BILGI**, если преобразовать эту последовательность в ASCII-код. Как только это содержимое обнаружится, инициируется предупреждение, указывающее на возможное наличие *троянской программы удаленного доступа* (Remote-Access Trojan – RAT), разработанной средствами CyberEYE. Программы RAT считаются вредоносными и незаметно выполняются на целевой машине, предоставляя злоумышленнику несанкционированный доступ к ней.

ПРИМЕЧАНИЕ *CyberEYE – это распространенное некогда инструментальное средство турецкого происхождения, предназначавшееся для создания исполняемого кода программ RAT и административного управления подвергавшихся угрозам хостов. По иронии судьбы, приведенное выше правило Snort*

инициирует предупреждение при обнаружении в пакете символьной строки **"ANA BILGI"**, *которая по-турецки означает "ОСНОВНАЯ ИНФОРМАЦИЯ".*

А теперь рассмотрим пример сетевого трафика, связанного с предупреждением из файла перехвата `ratinfected.pcapng`. Это предупреждение по правилу Snort обычно приводит к перехвату лишь одного пакета, инициировавшего данное предупреждение, но, к счастью, в нашем распоряжении имеется вся последовательность обмена данными между хостами. Чтобы выделить из анализируемого трафика самое главное, поищите шестнадцатеричную строку, упомянутую в правиле Snort, выполнив следующие действия.

1. Выберите команду Edit⇨Find Packet (Правка⇨Найти пакет) из главного меню или нажмите комбинацию клавиш <Ctrl+F>.
2. Выберите пункт Hex Value (Шестнадцатеричное значение) из раскрывающегося списка, расположенного слева от поля ввода.
3. Введите значение **41 4E 41 42 49 4C 47 49 7C** в текстовой области.
4. Щелкните на кнопке Find (Найти).

Как показано на рис. 12.23, вы должны теперь увидеть первое вхождение шестнадцатеричной строки в информационной части пакета 4 ❶.

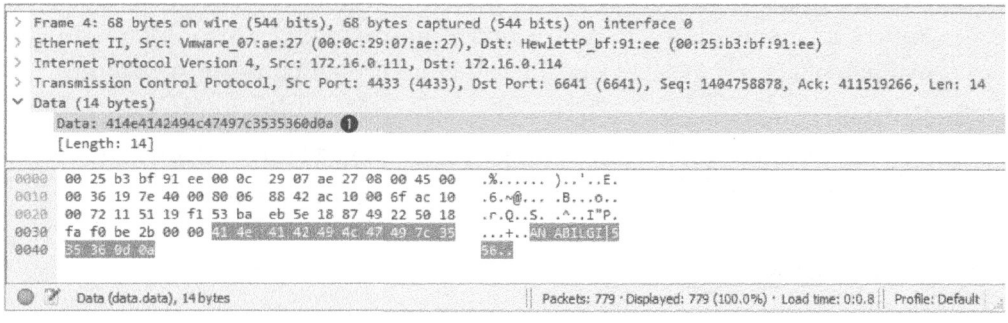


Рис. 12.23. Строка содержимого, на которую реагирует предупреждение по правилу Snort, впервые появляется в пакете 4

Если щелкнуть на кнопке Find еще несколько раз, то можно обнаружить, что данная строка присутствует также в пакетах 5, 10, 32, 156, 280, 405, 531 и 652. И хотя весь обмен данными в анализируемом здесь файле перехвата происходит между компьютером злоумышленника, находящимся по адресу **172.16.0.111**, и целевого хостом, расположенным по адресу **172.16.0.114**, вхождения этой строки оказываются и в других диалогах. В то время как обмен пакетами 4 и 5 происходит через порты **4433** и **6641**, большинство остальных вхождений рассматриваемой здесь строки обнаруживаются из обмена

пакетами между портом **4433** и произвольно выбранными временными портами. Заглянув на вкладку TCP в окне Conversations, можно убедиться в существовании нескольких диалогов, как показано на рис. 12.24.

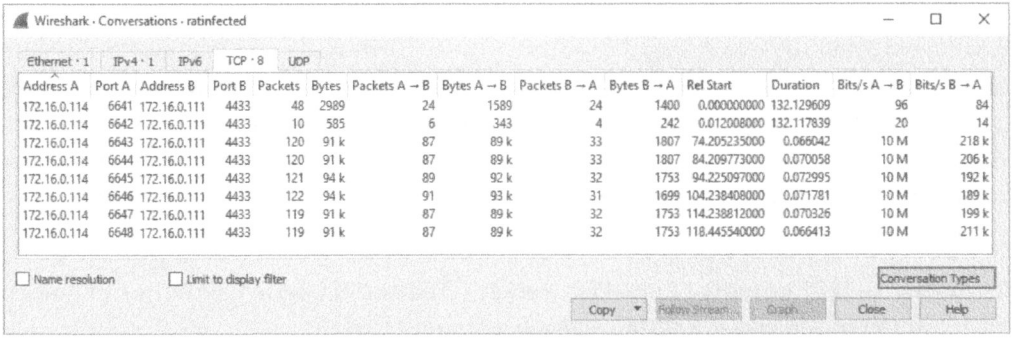


Рис. 12.24. Между компьютером злоумышленника и целевым хостом существуют три отдельных диалога

Различные диалоги можно разграничить визуально в рассматриваемом здесь файле перехвата, выделив их разным цветом. С этой целью выполните следующие действия.

1. Введите правило для фильтра **`(tcp.flags.syn == 1) && (tcp.flags.ack == 0)`** в диалоговом окне, расположенном выше панели Packet List, и нажмите клавишу <Enter>. В результате применения этого фильтра будут выбраны пакеты SYN для каждого диалога в анализируемом трафике.
2. Щелкните правой кнопкой мыши на первом пакете и выберите команду Colorize Conversation (Выделить диалог цветом) из контекстного меню.
3. Выберите сначала вариант TCP, а затем конкретный цвет.
4. Повторите данный процесс для остальных пакетов SYN, выбрав для каждого из них другой цвет.
5. По завершении щелкните на кнопке X, чтобы удалить применяемый фильтр.

Выделив диалоги разными цветами, можно удалить применяемый фильтр, чтобы посмотреть, каким образом они связаны друг с другом. Это поможет нам отследить процесс обмена данными между двумя хостами. Обмен данными (через порты **6641** и **4433**) между этими хостами начинается в первом диалоге, и поэтому начать анализ удобно именно с него. Итак, щелкните правой кнопкой мыши на любом пакете в первом диалоге и выберите команду Follow⇨TCP Stream (Отслеживать⇨Поток TCP) из контекстного меню. В итоге появится результат, приведенный на рис. 12.25.

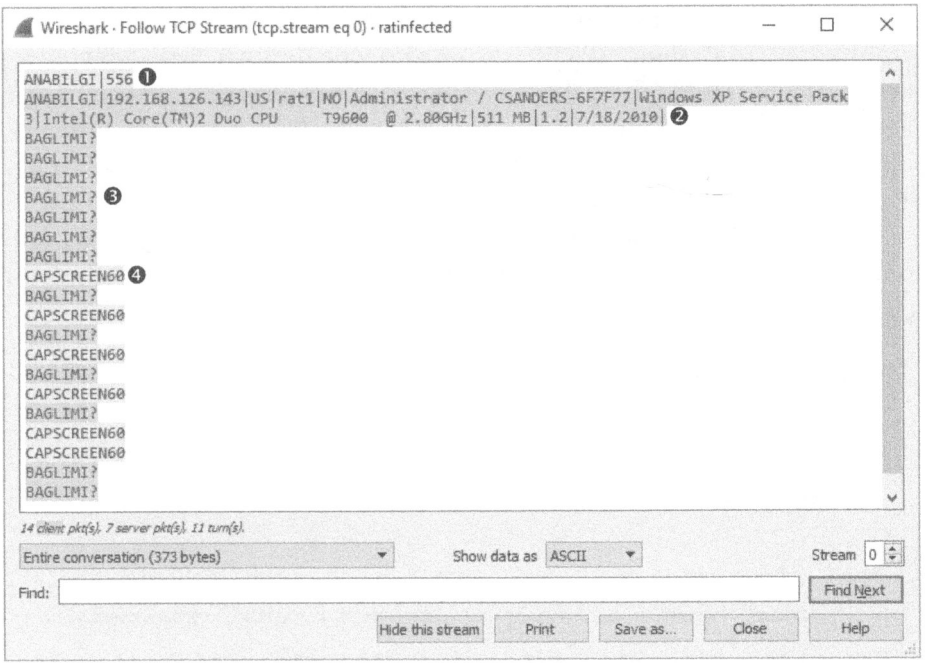


Рис. 12.25. Анализ первого диалога дает интересные результаты

И мы сразу же увидим, что текстовая строка `"ANABILGI|556"` посылается злоумышленником целевому хосту ❶. В итоге целевой хост ответит некоторым количеством основной информации о системе, включая имя компьютера (`CSANDERS-6F7F77`) и наименование применяемой операционной системы (`Windows XP Service Pack 3`) ❷, а затем начнет периодически передавать текстовую строку `"BAGLIMI?"` обратно злоумышленнику ❸. На это злоумышленник ответит лишь текстовой строкой `"CAPSCREEN60"` ❹, которая появляется шесть раз подряд.

Текстовая строка `"CAPSCREEN60"`, возвращаемая злоумышленником, представляет особый интерес, поэтому выясним, к чему она приведет. С этой целью непременно удалите любые фильтры отображения и найдите текстовую строку `"CAPSCREEN60"` в анализируемых здесь пакетах, выбрав опции String (Строка) и Packet bytes (Байты из пакета) из списков, расположенных на панели поиска, чтобы указать, где следует искать эту строку.

По окончании поиска перейдите к первому вхождению искомой строки в пакете 27. Обнаруженная здесь информация примечательна тем, что как только анализируемая здесь строка будет отправлена злоумышленником целевому хосту, последний подтвердит прием пакета, а затем начнется новый диалог в пакете 29. Теперь вам будет легче обнаружить начало нового диалога благодаря установленным ранее правилам выделения диалогов цветом.

Если теперь отследить поток вывода данных из этого нового диалога по протоколу TCP (рис. 12.26), то можно обнаружить уже знакомую текстовую строку "ANABILGI|12", следующую далее строку "SH|556" и строку "CAPSCREEN|C:\WINDOWS\jpgevhook.dat|84972" ❶. Обратите внимание на путь к файлу, указанный вслед за словом CAPSCREEN в последней строке, и следующий далее неудобочитаемый текст. Самое примечательное здесь состоит в том, что неудобочитаемый текст предваряется строкой "JFIF" ❷, которая обычно указывается в начале файлов формата JPG, как показывает быстрый поиск в Google.

Рис. 12.26. Оказывается, что злоумышленник инициирует запрос на загрузку файла изображения в формате JPG

В данный момент можно благоразумно заключить, что злоумышленник инициировал запрос на загрузку указанного файла изображения в формате JPG. Но еще важнее, что постепенно начинает проясняться структура команды, возникающая из сетевого трафика. Оказывается, что **CAPSCREEN** — это команда, посылаемая злоумышленником с целью начать передачу указанного файла изображения в формате JPG. В действительности всякий раз, когда посылается команда **CAPSCREEN**, получается одинаковый результат. Чтобы убедиться в этом, просмотрите поток TCP каждого диалога, где присутствует команда **CAPSCREEN**, или попытайтесь воспользоваться средством графического представления ввода-вывода в Wireshark, выполнив следующие действия.

1. Выберите команду Statistics⇨IO Graph из главного меню.
2. Щелкните на кнопке со знаком "плюс" (+), чтобы ввести пять строк.
3. Введите фильтры **tcp.stream eq 2**, **tcp.stream eq 3**, **tcp.stream eq 4**, **tcp.stream eq 5** и **tcp.stream eq 6** соответственно в пяти строках, вновь введенных в окне Display Filter. Присвойте каждому фильтру подходящее имя.
4. Измените на Bytes/s (Байт/с) единицы измерения, откладываемые по оси *y* при построении графика.
5. Щелкните на кнопках Graph 1, Graph 2, Graph 3, Graph 4 и Graph 5, чтобы активизировать точки данных для указанных фильтров.

Построенный в итоге график приведен на рис. 12.27.

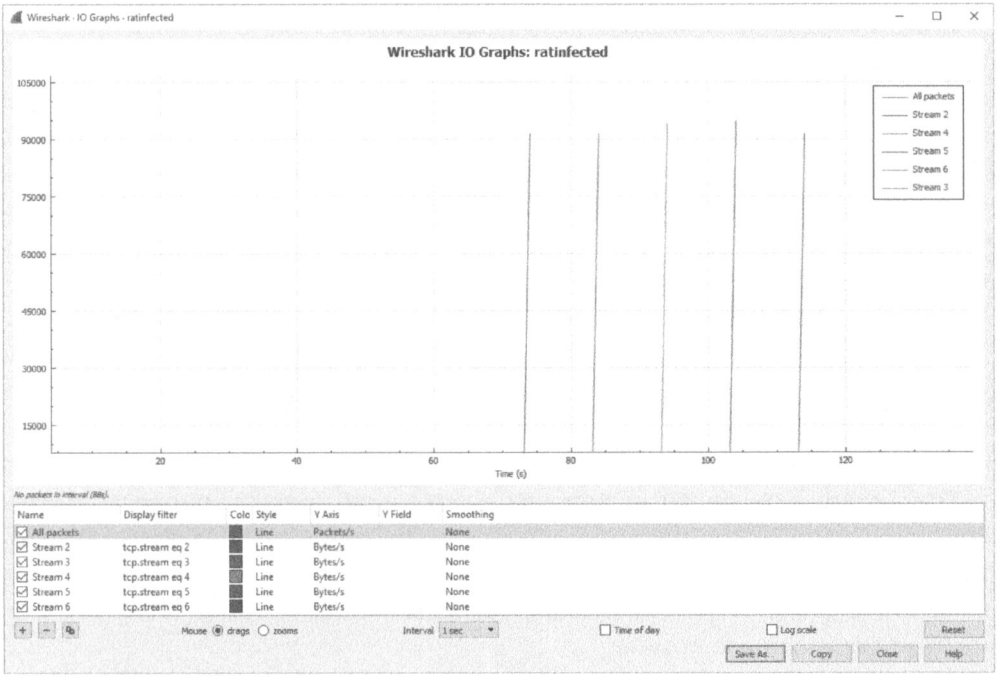


Рис. 12.27. На этом графике наглядно показано повторение одинаковых действий

Основываясь на этом графике, можно сделать вывод, что каждый диалог содержит приблизительно одинаковое количество данных и происходит в течение одного и того же периода времени. И эти действия повторяются несколько раз.

У вас, возможно, уже имеются некоторые догадки относительно содержимого передаваемого файла изображения в формате JPG, поэтому выясним, можно ли просмотреть такой файл. Чтобы извлечь данные из файла изображения в формате JPG в Wireshark, выполните следующие действия.

1. Прежде всего отследите поток TCP из подходящих пакетов, как это было сделано на рис. 12.25. Для это удобно выбрать пакет 29.
2. Обмен данными нужно изолировать, чтобы можно было увидеть только поток данных, отправленный целевым хостом злоумышленнику. С этой целью щелкните на кнопке со стрелкой раскрывающегося списка Entire Conversation (85033 bytes) (Весь диалог (85033 байта)). Выберите в нем нужное направление трафика 172.16.0.114:6643 --> 172.16.0.111:4433 (85 kB).
3. Выберите вариант RAW из раскрывающегося списка Show data as (Показать данные в формате).
4. Сохраните данные, щелкнув на кнопке Save As, чтобы сохранить файл с расширением `.jpg`.

Если теперь попытаться открыть файл изображения, то с удивлением можно обнаружить, что он не открывается. Дело в том, что для этого нужно сделать еще кое-что. В отличие от сценария, описанного в главе 10, "Основные реальные сценарии", где файл был извлечен непосредственно из трафика FTP, анализируемый здесь трафик дополнен, помимо данных, еще некоторым содержимым. В данном случае первые две строки, наблюдаемые в потоке TCP, фактически являются частью последовательности команд из вредоносной программы, а не данных, образующих изображение формата JPG (рис. 12.28). Это постороннее содержимое было сохранено вместе с выбранным потоком данных. В итоге средство просмотра файлов, искавшее заголовок файла изображения формата JPG, обнаружило содержимое, не соответствовавшее тому, что оно предполагало найти, и поэтому оно и не смогло открыть указанный файл изображения.

Устранение подобного недостатка — довольно простой процесс, требующий тем не менее умения работать шестнадцатеричном редакторе файлов. Все, что нужно сделать — вырезать нужную часть данных, составляющих изображение в формате JPG, из массива перехваченных данных. Для этого выполните следующие действия.

1. Просматривая поток TCP, приведенный на рис. 12.28, щелкните на кнопке Save as. Выберите удобное для запоминания имя файла и сохраните его в том месте, где его можно сразу же найти.
2. Загрузите шестнадцатеричный редактор файлов WinHex по адресу `https://www.x-ways.net/winhex/` и установите его.
3. Запустите редактор WinHex и откройте в нем файл, только что сохраненный в Wireshark.

Рис. 12.28. Постороннее содержимое, добавленное вредоносной программой, препятствует правильному открытию файла изображения

4. Выделите все постороннее содержимое в начале открытого файла. Это должны быть все байты, предшествующие байтам **FF D8 FF E0**, обозначающим начало нового файла изображения в формате JPG. Выделенные байты должны быть в итоге отмечены, как показано на рис. 12.29.

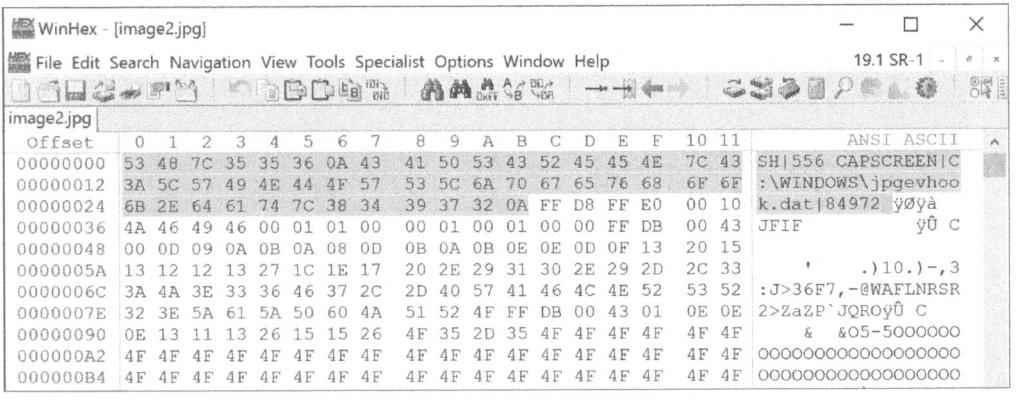


Рис. 12.29. Удаление посторонних байтов из файла изображения в формате JPG

5. Нажмите клавишу <Delete>, чтобы удалить выбранные данные.
6. Щелкните на кнопке Save главной панели инструментов редактора WinHex, чтобы сохранить внесенные изменения.

ПРИМЕЧАНИЕ *Я лично предпочитаю восстанавливать файл из перехваченного массива данных в редакторе WinHex под управлением Windows, но вы вольны воспользоваться для этой цели любым известным вам шестнадцатеричным редактором файлов.*

Итак, удалив ненужные байты данных, вы должны теперь суметь открыть рассматриваемый здесь файл. Теперь должно быть ясно, что троянская программа периодически делает копии экрана рабочего стола в операционной системе целевого хоста и передает их обратно злоумышленнику (рис. 12.30). Как только обмен данными будет завершен, далее последует процесс обычного разъединения соединения по протоколу TCP.

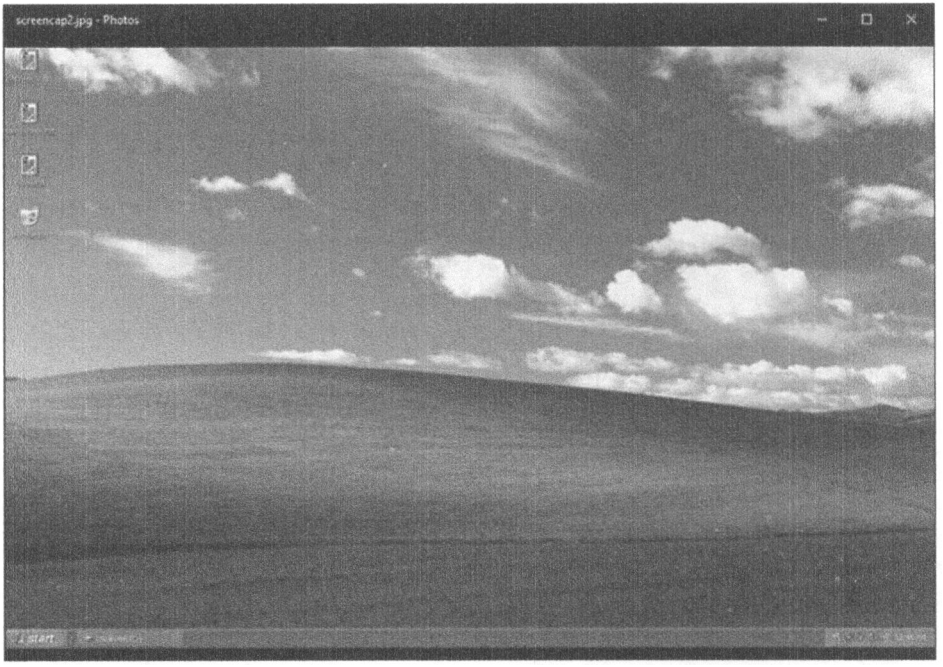


Рис. 12.30. В передаваемом файле в формате JPG содержится копия экрана рабочего стола операционной системы целевого компьютера

Рассматриваемый здесь сценарий служит характерным примером приведенного ниже хода мыслей специалиста, анализирующего сетевой трафик на основании предупреждения, полученного от системы обнаружения вторжений.

- Исследовать предупреждение и сформированную в нем сигнатуру.
- Убедиться, что сигнатура, совпавшая в анализируемом трафике, находится в надлежащем контексте.
- Исследовать трафик, чтобы выяснить, что сделал злоумышленник с поставленной под угрозу машиной.
- Начать процесс нейтрализации уязвимости, чтобы не допустить утечку секретной информации из поставленного под угрозу целевого хоста.

Набор эксплойтов и программы-вымогатели

Файлы перехвата **cryptowall4_c2.pcapng** *и* **ek_to_cryptowall4.pcapng**

И последний сценарий, рассматриваемый в этой главе, начинается с предупреждения от системы обнаружения вторжений. Сначала мы проанализируем пакеты, сформированные зараженной системой, а затем попытаемся отследить источник возникшей угрозы. В данном примере применяется настоящее вредоносное программное обеспечение, которое, вероятнее всего, является причиной заражения устройства в сети.

Рассматриваемый здесь сценарий начинается с предупреждения от системы обнаружения вторжений, которое было сгенерировано средствами Snort на консоли Sguil, как показано на рис. 12.31. Sguil — это инструментальное средство, применяемое для манипулирования, просмотра и исследования предупреждений от одного или нескольких датчиков системы обнаружения вторжений. И хотя это инструментальное средство предоставляет далеко не самый привлекательный пользовательский интерфейс, оно давно и широко применяется специалистами по анализу безопасности.

Об этом предупреждении в Sguil предоставляется немало сведений. Сводка этого предупреждения приведена на верхней панели ❶. Здесь можно увидеть момент времени, когда это предупреждение было сгенерировано, IP-адреса и порты отправителя и получателя, применяемый сетевой протокол, а также сообщение о событии, составленное на основании совпавшей сигнатуры из системы обнаружения вторжений. В данном случае дружественная локальная система, находящаяся по адресу **192.168.122.145**, связывается с неизвестной внешней системой, расположенной по адресу **184.170.149.44**, через порт **80**, который обычно связан с сетевым трафиком протокола HTTP. Внешнюю систему следует рассматривать как вредоносную, поскольку на обмен данными с ней среагировала система обнаружения вторжений. Одна из ее сигнатур свидетельствует о злонамеренном характере обмена данными, кроме того сам IP-адрес оказался никому не известен. Сигнатура, совпавшая с анализируемым здесь трафиком, представляет собой трафик регистрации (check-in),

исходящий из семейства вредоносных вирусных программ CryptoWall, если допустить, что штамм этой вирусной программы установлен на дружественной локальной системе.

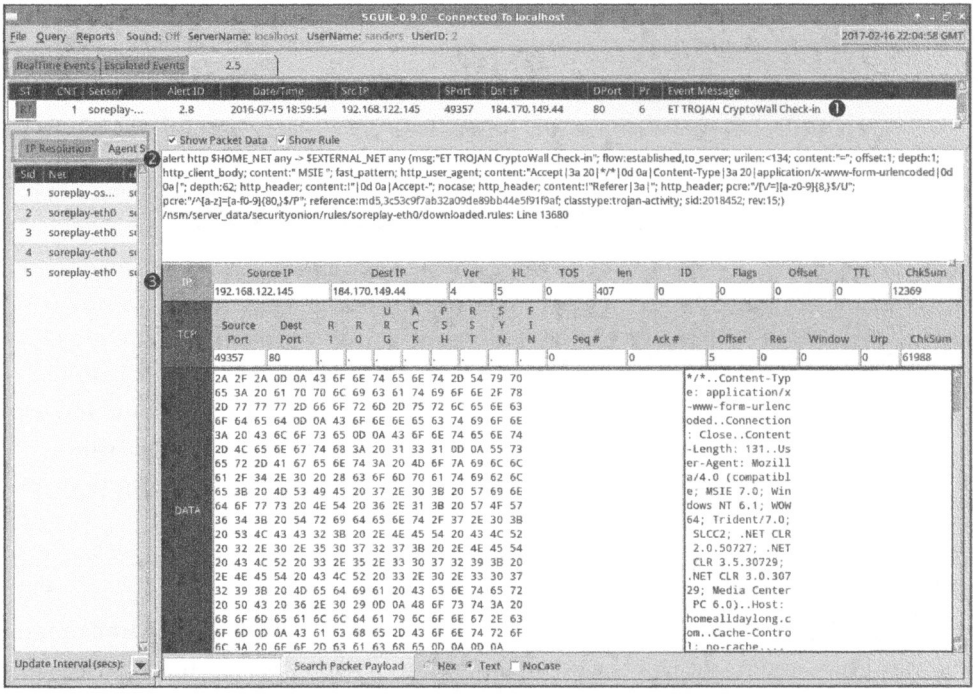


Рис. 12.31. Это предупреждение от системы обнаружения вторжений указывает на заражение вирусом CryptoWall 4

На консоли Sguil отображается синтаксис правила совпадения ❷ и данные из отдельных пакетов, где обнаружено совпадение с этим правилом ❸. Обратите внимание на то, что сведения из пакетов разделяются на заголовок протокола и информационные части аналогично представлению подобных сведений в Wireshark. Но, к сожалению, в Sguil предоставляются сведения только из одного совпавшего пакета, а нам потребуется более углубленный анализ пакетов. Поэтому исследуем далее сетевой трафик, связанный с рассматриваемым здесь предупреждением, непосредственно в Wireshark, чтобы попытаться проверить его на достоверность и выяснить, что же в нем происходит. Этот трафик содержится в файле перехвата `cryptowall4_c2.pcapng`.

Анализируемые здесь перехваченные пакеты содержат обмен данными, происходивший приблизительно в тот момент, когда возникло предупреждение, и он не особенно сложный. Первый диалог происходит в пакетах 1–16, и его можно без труда просмотреть, отследив поток TCP из этого диалога (рис. 12.32). В самом начале анализируемого здесь перехваченного трафика

локальная система устанавливает соединение по протоколу TCP с вредоносным хостом через порт **80** и выдает запрос по методу `POST`, содержащий URL адресата `http://homealldaylong.com/76N1Lm.php?x4tk7t4jo6` ❶ и небольшое количество буквенно-цифровых данных ❷. На это вредоносный хост отвечает буквенно-цифровой строкой ❹ и кодом состояния **`HTTP 200 OK`** ❸, прежде чем данное соединение будет корректно разорвано.

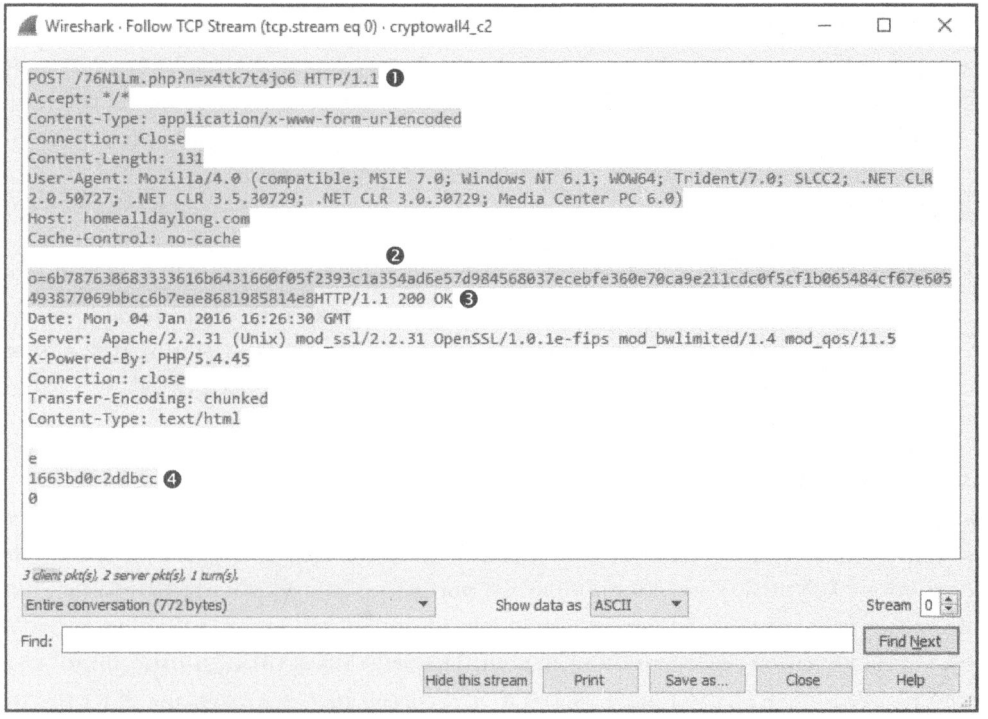


Рис. 12.32. Между этими хостами передается небольшое количество данных по протоколу HTTP

Если проанализировать остальную часть упоминаемого здесь файла перехвата, то можно заметить, что всякий раз повторяется практически одна и та же последовательность обмена данными между этими хостами, хотя и с незначительными изменениями в передаваемых данных. Чтобы посмотреть три разных соединения со сходной структурой URL, можно воспользоваться фильтром **`http.request.method == "POST"`** (рис. 12.33).

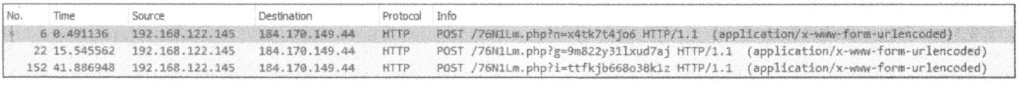


Рис. 12.33. Структура URL, демонстрирующая разные данные, передаваемые одной и той же странице

Часть **76N1Lm.php** структуры URL, относящаяся непосредственно к веб-странице, остается той же самой, тогда как остальная ее часть, относящаяся к параметрам и данным, передаваемым странице, изменяется. Повторяющаяся последовательность обмена данными в сочетании со структурой запросов согласуется с командно-управляющим поведением вредоносного программного обеспечения и сигнатурой, сгенерировавшей предупреждение. Следовательно, анализируемая здесь система, вероятнее всего, заражена вирусом CryptoWall, как и предполагается в самой сигнатуре. Можете убедиться в этом дополнительно, исследовав аналогичный пример, который можно найти на популярной странице CryptoWall Tracker, находящейся по адресу `https://www.cryptowalltracker.org/cryptowall-4.html#networktraffic`.

ПРИМЕЧАНИЕ *В этой книге недостаточно места, чтобы досконально разобрать данные, которыми обмениваются дружественная и вредоносная системы во время командно-управляющей последовательности в рассматриваемом здесь сценарии. Но если вас это заинтересует, дополнительные сведения о данном процессе вы можете получить по адресу* **`https://www.cryptowalltracker.org/communication-protocol.html`**.

Итак, убедившись, что в рассматриваемом здесь сценарии имеет место командно-управляющая последовательность обмена данными с вредоносной программой, было бы неплохо устранить этот недостаток, вылечив зараженную вирусом машину. Это особенно важно сделать, когда машина заражена вирусом вроде CryptoLocker, поскольку он пытается зашифровать пользовательские данные и предоставляет ключ расшифровки только в том случае, если пользователь заплатит солидный выкуп. Именно по этой причине подобные вредоносные программы называются *программами-вымогателями*. Рассмотрение способов излечения от подобного вируса выходит за рамки данной книги, но в реальном сценарии это следующая последовательность действий, которую должен предпринять специалист по анализу безопасности.

В связи с этим возникает следующий уточняющий вопрос: каким образом произошло заражение дружественной системы? Если источник заражения удастся определить, то могут быть обнаружены и другие устройства, зараженные аналогичным образом другими вирусами. А возможно, удастся выработать механизмы защиты или обнаружения, чтобы предупредить заражение.

В проанализированных выше пакетах командно-управляющую последовательность обмена данными удалось выявить лишь после заражения. В тех сетях, где проводится текущий контроль безопасности и постоянный перехват пакетов, многие сетевые датчики настроены на хранение данных из пакетов в течение нескольких часов или дней для судебной экспертизы. Ведь далеко не каждая организация оснащена средствами оперативного реагирования на

предупреждения. Временное хранение пакетов дает возможность проанализировать их данные со стороны дружественного хоста, прежде чем началась командно-управляющую последовательность обмена данными. И такие пакеты зафиксированы в файле перехвата `ek_to_cryptowall4.pcapng`.

Прокрутив содержимое этого файла перехвата, можно обнаружить намного больше пакетов для анализа, хотя все они относятся к протоколу HTTP. Зная принцип действия этого протокола, попробуем перейти сразу к делу, ограничив отображаемые пакеты только запросами с помощью фильтра **`http.request`**. В итоге останется лишь одиннадцать HTTP-запросов, исходящих из дружественного хоста (рис. 12.34).

No.	Time	Source	Destination	Protocol	Info
4	0.534405	192.168.122.145	113.20.11.49	HTTP	GET /index.php/services HTTP/1.1
35	5.265859	192.168.122.145	45.32.238.202	HTTP	GET /contrary/1653873/quite-someone-visitor-nonsense-tonight-sweet-await-gigantic-dance-third HTTP/1.1
39	6.109508	192.168.122.145	45.32.238.202	HTTP	GET /occasional/bXJkeHFlYXhmaA HTTP/1.1
123	9.126714	192.168.122.145	45.32.238.202	HTTP	GET /goodness/1854096/earnest-fantastic-thorough-weave-grotesque-forth-awaken-fountain HTTP/1.1
130	14.020289	192.168.122.145	45.32.238.202	HTTP	GET /observation/enVjZ2dtcnpz HTTP/1.1
441	30.245463	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?v=x4tk7t4jo6 HTTP/1.1 (application/x-www-form-urlencoded)
456	41.772768	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?n=x4tk7t4jo6 HTTP/1.1 (application/x-www-form-urlencoded)
472	45.628284	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?w=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
487	56.827194	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?g=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
619	71.971402	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?h=ttfkjb668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)
634	83.168580	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?i=ttfkjb668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)

Рис. 12.34. В результате фильтрации осталось лишь одиннадцать HTTP-запросов, исходящих из дружественного хоста

Первый запрос направляется дружественным хостом, находящимся по адресу **`192.168.122.145`**, неизвестному внешнему хосту, расположенному по адресу **`113.20.11.49`**. Анализ HTTP-заголовка из пакета с этим запросом показывает (рис. 12.35), что пользователь запросил страницу по адресу `http://www.sydneygroup.com.au/index.php/services/` ❶, которую он нашел в поисковой системе Bing по запросу `sydneygroup.com.au` ❷. До сих пор все вроде бы выглядит нормально.

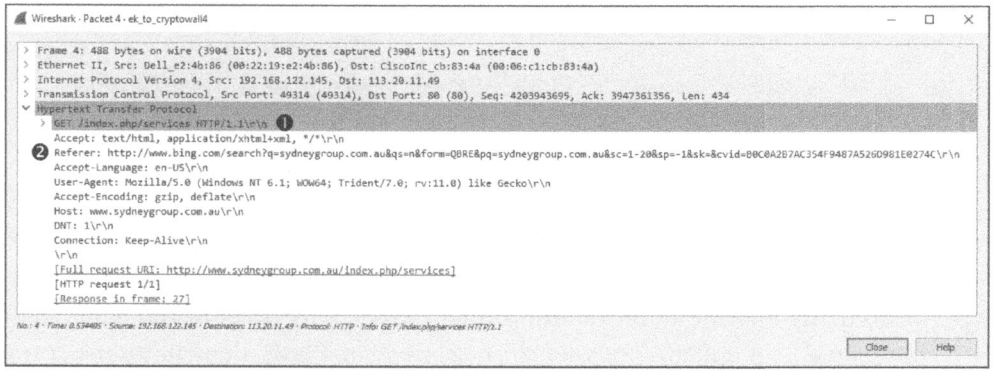


Рис. 12.35. HTTP-запрос, отправленный неизвестному внешнему хосту

Далее дружественный хост посылает в пакетах 35, 39, 123 и 130 четыре запроса на другой неизвестный внешний хост, расположенный по адресу

45.32.238.202. Как демонстрировалось в предыдущих примерах, браузеры обычно извлекают содержимое из дополнительных хостов при просмотре веб-страницы, на которой хранится встраиваемое содержимое или рекламные объявления из сторонних серверов. Само по себе это не вызывает тревогу, хотя домен в этих запросах выглядит так, словно он составлен случайным образом, что вызывает подозрение.

Любопытнее дело обстоит в запросе по методу `GET` в пакете 39. Отследив поток TCP в этом обмене данными (рис. 12.36), можно обнаружить, что в нем запрашивается файл `bXJkeHFlYXhmaA` ❶. Имя этого файла выглядит не совсем обычно и вообще не включает в себя расширение.

Рис. 12.36. Загрузка Flash-файла с не совсем обычным именем

Более тщательный анализ показывает, что веб-сервер обозначает содержимое этого файла как **`x-shockwave-flash`** ❷. Flash является распространенным подключаемым модулем для воспроизведения мультимедийных потоков в браузере, и поэтому вполне естественно обнаружить, что устройство загружает Flash-содержимое. Однако подключаемый модуль Flash печально известен своими уязвимостями, которые зачастую не устраняются. Таким образом, упомянутый выше Flash-файл успешно загружается по запросу.

После загрузки Flash-файла в пакете 130 выдается запрос на другой аналогично именуемый файл. Отследив этот поток TCP (рис. 12.37), можно

обнаружить, что в нем запрашивается файл по имени `enVjZ2dtcnpz` ❶. Тип этого файла здесь не обозначается расширением или сервером. После этого запроса целевой хост загружает фрагмент неудобочитаемых данных объемом 358400 байт ❷.

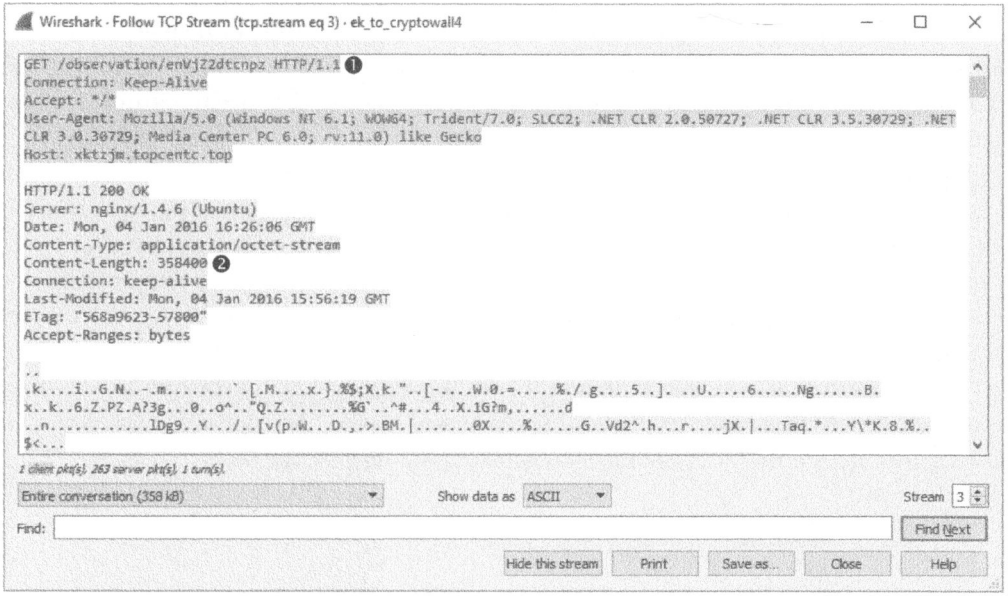


Рис. 12.37. Загрузка еще одного файла с не совсем обычным именем, но на этот раз тип этого файла не определен

Менее чем за 20 секунд после загрузки упомянутого выше файла должно появиться нечто знакомое в списке HTTP-запросов, приведенных на рис. 12.34. Начиная с пакета 441, дружественный хост начнет посылать HTTP-запросы по методу `POST` двум серверам, используя ту же самую командно-управляющую последовательность, что и рассмотренная выше. Очень похоже на то, что мы выявили источник заражения, ответственность на которое возлагается на оба загруженных файла. Первый файл, запрошенный в пакете 39, доставил средство эксплуатации уязвимостей (эксплойт), а второй файл, запрошенный в пакте 130, — вредоносную программу.

ПРИМЕЧАНИЕ *Для декодирования и анализа файлов, указанных в перехваченных пакетах, можно воспользоваться методиками анализа вредоносного программного обеспечения. Если вам интересно узнать больше о методиках восстановления алгоритмов работы вредоносного программного обеспечения, прочитаете книгу Practical Malware Analysis Майкла Сикорски (Michael Sikorski) и Эндрю Хонига (Andrew Honig), которая вышла в издательстве No Starch Press в 2012 году и которой я отдаю предпочтение.*

В рассматриваемом здесь сценарии демонстрируется одна из самых распространенных методик заражения вирусами. Она состоит в том, что, блуждая по Интернету, пользователь попадает на сайт, зараженный вредоносным переадресовывающим кодом из набора эксплойтов. Такие наборы служат для заражения вполне законных и надежных серверов и снятия отпечатков с клиентов для выявления их уязвимостей. Зараженная страница называется *посадочной страницей* эксплойта и служит для переадресации клиента на другой сайт, содержащий средство эксплуатации уязвимостей, определенное в этом наборе как наиболее эффективное для нанесения вреда целевой системе.

Проанализированные здесь пакеты перехвачены из набора эксплойтов Angler, который наиболее часто употреблялся атакующими злоумышленниками в 2015 и 2016 гг. Как только пользователь попадал на сайт, зараженный эксплойтом Angler, этот набор сразу же определял имеет ли подключаемый модуль Flash в браузере пользователя уязвимость. И если да, то тогда в эту систему сначала доставлялся Flash-файл, чтобы воспользоваться ее уязвимостью, а затем в качестве вторичной полезной информации был загружен и установлен вредоносный вирус CryptoWall на целевом хосте. Вся последовательность заражения целевой системы наглядно показана на рис. 12.38.

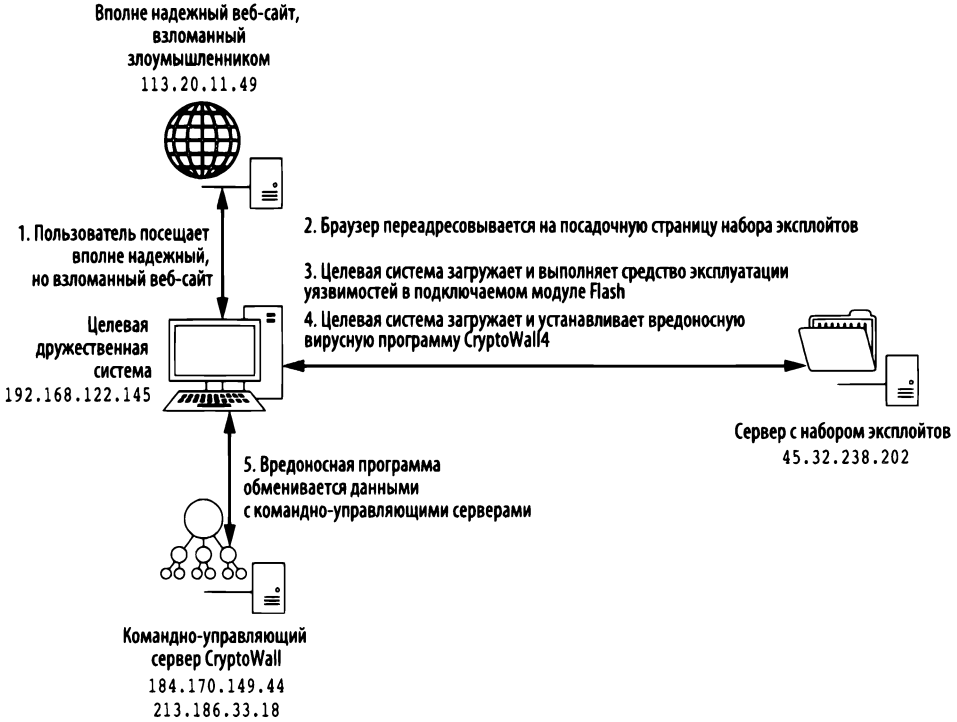


Рис. 12.38. Последовательность заражения целевой системы с помощью набора эксплойтов