

СОДЕРЖАНИЕ

От издательства	11
Об авторе	12
О рецензенте	12
Предисловие	13
Благодарности	16
Введение	18
Глава 1. Знакомство с основами	23
Сетевая архитектура и протоколы	23
Набор интернет-протоколов	24
Инкапсуляция данных	27
Заголовки, концевики и адреса.....	27
Передача данных	28
Сетевая маршрутизация.....	29
Моя модель для анализа сетевых протоколов.....	31
Заключительное слово	33
Глава 2. Перехват трафика	34
Пассивный перехват сетевого трафика	34
Краткое руководство по Wireshark.....	35
Альтернативные методы пассивного перехвата	37
Отслеживание системных вызовов.....	37
Утилита strace для Linux	39

Мониторинг сетевых подключений с помощью DTrace	40
Process Monitor в Windows	41
Преимущества и недостатки пассивного перехвата	43
Активный перехват сетевого трафика	43
Сетевые прокси.....	44
Прокси-сервер с переадресацией портов.....	44
Прокси-сервер SOCKS.....	48
Прокси-серверы HTTP	53
Переадресация	54
Обратный прокси-сервер HTTP	57
Заключительное слово	61

Глава 3. Структура сетевых протоколов

Структура двоичных протоколов	63
Числовые данные.....	63
Логические значения.....	66
Битовые флаги.....	66
Двоичный порядок байтов.....	67
Текстовые и удобочитаемые данные	68
Данные переменной длины	72
Даты и время	75
POSIX/Unix-время.....	75
FILETIME	76
Паттерн TLV	76
Мультиплексирование и фрагментация.....	77
Информация о сетевом адресе	78
Структурированные двоичные форматы	78
Структуры текстового протокола	80
Числовые данные.....	80
Текстовые логические значения.....	81
Даты и время	81
Данные переменной двоичной длины.....	82
Структурированные текстовые форматы	82
Кодирование двоичных данных	85
Шестнадцатеричное кодирование	86
Base64	86
Заключительное слово	88

Глава 4. Расширенный перехват трафика.....

Перенаправление трафика.....	89
Использование traceroute.....	90
Таблицы маршрутизации	91
Настройка маршрутизатора	92
Активируем маршрутизацию в Windows.....	93
Активируем маршрутизацию в Unix-подобных системах	93
Преобразование сетевых адресов	94
Активируем SNAT	94
Настройка SNAT в Linux	95

Активируем DNAT	96
Перенаправление трафика на шлюз.....	98
DHCP-спуфинг.....	98
ARP-спуфинг.....	101
Заключительное слово	105
Глава 5. Анализ на практике	106
Приложение для генерирования трафика: SuperFunkyChat.....	106
Запуск сервера.....	107
Запуск клиентов.....	107
Обмен данными между клиентами.....	108
Экспресс-курс анализа с помощью Wireshark.....	109
Генерация сетевого трафика и захват пакетов.....	110
Базовый анализ.....	111
Чтение содержимого TCP-сеанса.....	112
Определение структуры пакета с помощью шестнадцатеричного дампа.....	113
Просмотр отдельных пакетов.....	114
Определение структуры протокола	115
Проверим свои предположения.....	117
Анализ протокола с помощью Python	118
Разработка диссекторов Wireshark на Lua.....	124
Создание диссектора	126
Разбор	128
Парсинг пакета сообщения	128
Использование прокси-сервера для активного анализа трафика	131
Настройка прокси-сервера.....	132
Анализ протокола с использованием прокси-сервера.....	134
Добавляем базовый парсинг протокола	136
Изменение поведения протокола.....	137
Заключительное слово	139
Глава 6. Обратная разработка	140
Компиляторы, интерпретаторы и ассемблеры	141
Интерпретируемые языки.....	141
Компилируемые языки	142
Статическая и динамическая компоновки	142
Архитектура x86	143
Архитектура набора команд	143
Регистры ЦП.....	145
Порядок выполнения.....	147
Основы операционной системы.....	148
Форматы исполняемых файлов	148
Области.....	149
Процессы и потоки.....	150
Сетевой интерфейс операционной системы	150
Двоичный интерфейс приложений.....	153
Статический анализ.....	154
Краткое руководство по использованию IDA Pro Free Edition.....	155

Анализ переменных и аргументов стека	158
Определение ключевой функциональности	159
Динамический анализ	164
Установка точек останова	165
Отладчик Windows	166
Где установить точки останова?	168
Языки программирования, компилируемые в управляемый код	168
Приложения .NET	168
Использование ILSpy	169
Приложения Java	172
Работа с обфускацией	174
Ресурсы	175
Заключительное слово	176

Глава 7. Безопасность сетевого протокола

Алгоритмы шифрования	178
Шифр подстановки	179
XOR-шифрование	180
Генераторы случайных чисел	181
Симметричное шифрование	182
Блочные шифры	182
Режимы блочного шифрования	185
Дополнение	188
Атака padding oracle	189
Потоковые шифры	192
Криптографическая система с открытым ключом	193
Алгоритм RSA	193
Дополнение с RSA	195
Протокол Диффи–Хеллмана	196
Алгоритмы подписи	197
Алгоритмы криптографического хеширования	198
Асимметричные алгоритмы подписи	199
Коды аутентификации сообщений	200
Инфраструктура открытых ключей	203
Сертификаты X.509	203
Проверка цепочки сертификатов	205
Пример использования: протокол защиты транспортного уровня	206
TLS-рукопожатие	207
Начальное согласование	207
Аутентификация конечной точки	208
Установка зашифрованного соединения	210
Соответствие требованиям безопасности	211
Заключительное слово	212

Глава 8. Реализация сетевого протокола

Воспроизведение существующего перехваченного сетевого трафика	214
Захват трафика с помощью Netcat	215

Использование Python для повторной отправки захваченного UDP-трафика.....	217
Изменяем назначение нашего прокси.....	219
Повторное использование существующего исполняемого кода.....	224
Повторное использование кода в приложениях .NET	225
Повторное использование кода в приложениях Java	230
Неуправляемые исполняемые файлы	232
Шифрование и работа с TLS.....	236
Изучение используемого шифрования	237
Расшифровка TLS-трафика	238
Заключительное слово	243
Глава 9. Основные причины уязвимостей.....	244
Классы уязвимостей	245
Удаленное выполнение кода.....	245
Отказ в обслуживании.....	245
Утечка информации.....	246
Обход аутентификации.....	246
Обход авторизации	246
Уязвимости, связанные с повреждением памяти.....	247
Безопасные и небезопасные языки программирования с точки зрения доступа к памяти	247
Переполнение буфера.....	248
Индексирование буфера за пределами границ.....	253
Атака с расширением данных	255
Сбой при динамическом выделении памяти	255
Учетные данные, используемые по умолчанию или вшитые в код	256
Перечисление пользователей.....	257
Неправильный доступ к ресурсам.....	258
Канонизация.....	258
Подробные сообщения об ошибках	259
Исчерпание памяти	261
Исчерпание хранилища.....	262
Исчерпание ресурсов ЦП.....	263
Алгоритмическая сложность	263
Конфигурируемая криптография	265
Уязвимости строки форматирования	266
Внедрение команд	267
Внедрение SQL-кода.....	268
Замена символов в текстовой кодировке.....	269
Заклучительное слово	271
Глава 10. Поиск и эксплуатация уязвимостей.....	272
Фаззинг.....	273
Простейший тест.....	273
Мутационный фаззер	274
Создание тест-кейсов	275

Сортировка уязвимостей.....	275
Отладка приложений.....	275
Повышаем наши шансы найти первопричину сбоя.....	282
Эксплуатация распространенных уязвимостей.....	285
Эксплуатация уязвимостей, связанных с нарушением целостности памяти.....	285
Произвольная запись в память.....	293
Написание шелл-кода.....	296
Приступим.....	296
Простая техника отладки.....	299
Вызов системных вызовов.....	300
Выполнение других программ.....	305
Генерация шелл-кода с помощью Metasploit.....	306
Устранение уязвимостей, связанных с нарушением целостности памяти.....	308
Предотвращение выполнения данных.....	309
Использование метода возвратно-ориентированного программирования.....	310
Рандомизация размещения адресного пространства.....	312
Обнаружение переполнения стека с помощью предохранителей.....	316
Заключительное слово.....	319
Набор инструментов для анализа сетевых протоколов.....	320
Предметный указатель.....	335

Об авторе

Джеймс Форшоу – известный специалист по компьютерной безопасности из команды Google Project Zero с более чем десятилетним опытом анализа и эксплуатации уязвимостей в сетевых протоколах прикладного уровня. Его навыки варьируются от взлома игровых консолей до выявления сложных проблем проектирования в операционных системах, особенно в Microsoft Windows, что принесло ему награду в размере 100 000 долларов и позволило занять первое место в списке Microsoft Security Response Center (MSRC). Он создал Canare, инструмент для анализа сетевых протоколов, который он разработал, будучи специалистом с многолетним опытом работы в этой области, а также был приглашен принять участие в глобальных конференциях по безопасности, таких как BlackHat, CanSecWest и Chaos Computer Congress, где он представил свои новаторские исследования.

О рецензенте

С первых дней существования Commodore PET и VIC-20 технологии были постоянным спутником (а иногда и навязчивой идеей!) Клиффа Янзена. Клифф обнаружил в себе страсть к этой профессии, когда в 2008 г. после десяти лет работы в ИТ перешел работать в сферу информационной безопасности. С тех пор ему посчастливилось сотрудничать с лучшими специалистами этой отрасли и учиться у них, включая мистера Форшоу и сотрудников из No Starch во время создания этой книги. Он работает консультантом по вопросам безопасности, занимаясь всем – от анализа политик до тестов на проникновение. Ему повезло, что у него есть карьера, которая вместе с тем является его любимым хобби, и жена, которая его поддерживает.

ПРЕДИСЛОВИЕ

Когда я впервые познакомилась с Джеймсом Форшоу, я занималась тем, что в 2007 г. журнал Popular Science описал как одну из десяти худших профессий Microsoft Security Grunt. Это ярлык, который журнал использовал для всех, кто работал в Microsoft Security Response Center (MSRC). Это позиционировало нашу работу хуже, чем «исследователь китовых фекалий», но немного лучше, чем «вазэктомист, лечащий слонов» в этом списке профессий (настолько известном среди тех из нас, кто страдал в Редмонде, штат Вашингтон, что мы сделали футболки), так это непрекращающийся шквал отчетов об ошибках в системе безопасности в продуктах Microsoft.

Именно здесь, в MSRC, Джеймс, с его острым и творческим взглядом на необычное и упускаемое из виду, впервые привлек мое внимание в качестве стратега по безопасности. Джеймс был автором некоторых самых интересных отчетов об ошибках безопасности. Это был немалый подвиг, учитывая, что MSRC получал более 200 000 отчетов об ошибках безопасности в год от исследователей в области ИБ. Джеймс обнаруживал не только простые ошибки – в платформе .NET Framework он нашел проблемы на уровне архитектуры. Хотя их было труднее исправить с помощью простого патча, они были гораздо более ценными для Microsoft и ее клиентов.

Перенесемся к первой программе Bug Bounty от корпорации Microsoft, которую я создала в компании в июне 2013 года. Первоначально у нас было три программы – программы, которые обещали платить исследователям безопасности, таким как Джеймс, наличными в обмен на сообщение о наиболее серьезных ошибках в Microsoft. Я знала: для того чтобы эти программы доказали свою эффективность, нужно было исправлять серьезные ошибки.

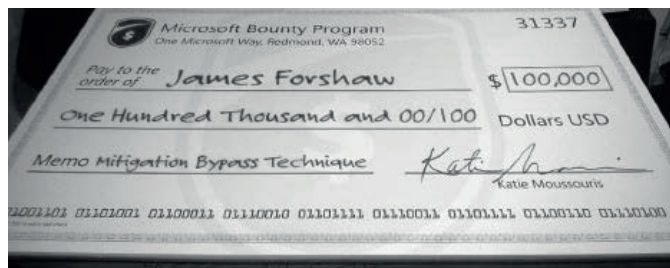
Если бы мы создали ее, не было никакой гарантии, что к нам придут специалисты по поиску ошибок. Мы знали, что соревнуемся за право стать одними из самых высококвалифицированных специалистов по поиску ошибок в мире. Было доступно множество других денежных вознаграждений, но не все вознаграждения назначались именно за защиту. У отдельных государств и преступников был хорошо развитый рынок ошибок и эксплойтов, и Microsoft полагалась на специалистов, которые уже предоставляли по 200 000 отчетов об ошибках в год бесплатно. Награды должны были привлечь внимание этих дружелюбных, альтруистических охотников за ошибками, в устранении которых Microsoft нуждалась больше всего.

Поэтому я, конечно же, позвонила Джеймсу и другим, потому что рассчитывала, что они займутся этим. Мы, специалисты по безопасности из MSRC, действительно хотели получить уязвимости для бета-версии Internet Explorer (IE) 11, и нам нужно было нечто, за что ни один поставщик программного обеспечения никогда не пытался назначить вознаграждение: мы хотели узнать о новых техниках эксплуатации. Эта награда была известна как Mitigation Bypass Bounty и в то время составляла 100 000 долларов. Я помню, как сидела с Джеймсом за кружкой пива в Лондоне, пытаясь увлечь его поиском ошибок в IE, когда он объяснил, что раньше никогда особо не интересовался безопасностью браузера, и предупредил меня, чтобы я не ожидала от него слишком многого.

Тем не менее Джеймс создал четыре уникальных варианта побега из песочницы бета-версии IE 11 Четыре. Они находились в тех областях кода IE, которые наши внутренние команды и добросовестные внешние специалисты по тестированию на проникновение пропустили. Побег из песочницы были необходимы для более надежной эксплуатации других ошибок. Джеймс получил награды за все четыре ошибки, за которые заплатила сама команда IE, плюс дополнительный бонус в размере 5000 долларов из моего бюджета. Оглядываясь назад, я, наверное, должна была дать ему лишние 50 000 долларов. Потому что это супер! Неплохо для охотника за ошибками, который никогда раньше не интересовался безопасностью веб-браузеров.

Всего несколько месяцев спустя я позвонила Джеймсу, находясь рядом с кафетерием Microsoft прохладным осенним днем, совершенно запыхавшись, чтобы сказать ему, что он только что вошел в историю: его заявка на участие в одной из других программ Bu Bount, Microsoft – Mitigation Bypass Bounty, на сумму 100 000 долларов, была принята. Джеймс Форшоу нашел новый уникальный способ обойти все средства защиты платформы, используя недостатки архитектурного уровня в последней версии операционной системы, и выиграл самую первую награду в размере 100 000 долларов от Microsoft.

Во время того телефонного разговора, насколько я помню, он сказал, что представил, как я вручаю ему до смешного огромный чек на сцене во время внутренней конференции Microsoft BlueHat. После этого звонка я отправила в отдел маркетинга записку, и в одно мгновение «Джеймс и гигантский чек» навсегда вошли в историю Microsoft и интернета.



Я уверена, что читатели узнают на страницах этой книги о несравненной гениальности Джеймса – той же гениальности, которую я увидела в одном или четырех отчетах об ошибках много лет назад. Существует очень мало исследователей безопасности, которые могут найти ошибки в одной передовой технологии, и еще меньше тех, кто может последовательно находить их в нескольких технологиях. Кроме того, есть такие люди, как Джеймс Форшоу, которые могут сосредоточиться на более глубоких архитектурных проблемах с точностью хирурга. Я надеюсь, что те, кто читает эту книгу и будет читать все последующие книги Джеймса, воспримут ее как практическое руководство, которое поможет им пробудить ту же гениальность и творческий потенциал в своей работе.

На собрании по Bug Bounty в Microsoft, когда члены команды IE качали головами, гадая, как они могли пропустить эти ошибки, о которых сообщил Джеймс, я просто сказала: «Джеймс видит Женщину в Красном платье, а также код, который ее визуализировал, в Матрице». Все, кто сидел за столом, приняли это объяснение того, как работал у Джеймса ум. Ему все было по плечу; и, изучая его работы, если вы не грешите предвзятостью, то тоже сможете стать такими же.

Всем искателям ошибок в мире – вот ваша планка, и она высока. Несмотря на неисчислимое количество специалистов по безопасности, пусть все ваши отчеты об ошибках будут такими же интересными и ценными, как и те, что предоставлены единственным и неповторимым Джеймсом Форшоу.

Кэти Муссурис,
основатель и генеральный директор Luta Security
Октябрь 2017 г.

ВВЕДЕНИЕ

Когда впервые была представлена технология, позволявшая устройствам подключаться к сети, она была эксклюзивной для крупных компаний и правительств. Сегодня большинство людей носят с собой полностью подключенные к сети вычислительные устройства, а с развитием интернета вещей (IoT) можно добавить в этот взаимосвязанный мир такие устройства, как холодильник и домашняя система безопасности. Безопасность этих устройств становится все более важной. Хотя вы, возможно, и не слишком беспокоитесь о том, что кто-то раскроет подробности того, сколько йогуртов вы покупаете, если ваш смартфон будет взломан в той же сети, что и ваш холодильник, вы можете лишиться всей своей личной и финансовой информации – она будет доступна злоумышленнику.

Эта книга называется *«Атака сетей на уровне протоколов»*, потому что для обнаружения уязвимостей в устройстве, подключенном к сети, необходимо принять образ мыслей злоумышленника, который хочет использовать эти слабые места. Сетевые протоколы обмениваются данными с другими устройствами в сети, и поскольку эти протоколы должны быть доступны в открытой сети и нечасто подвергаются такому же уровню проверки, как другие компоненты устройства, они являются очевидной целью атаки.

Зачем читать эту книгу?

Во многих книгах обсуждается захват сетевого трафика с целью диагностики и базового анализа сети, но в них не говорится об аспектах безопасности протоколов. Эту книгу от других отличает тот факт, что она фокусируется на анализе пользовательских протоколов для поиска уязвимостей в системе безопасности.

Она для тех, кто интересуется анализом и атаками сетей на уровне протоколов, но не знает, с чего начать. Здесь вы познакомитесь с методами обучения захвату сетевого трафика, выполнением анализа протоколов, а также обнаружением и эксплуатацией уязвимостей в системе безопасности. В книге представлена справочная информация о сетях и сетевой безопасности, а также практические примеры протоколов для анализа.

Хотите ли вы атаковать сеть, чтобы сообщить об уязвимостях поставщику приложения, или просто хотите узнать, как ваше IoT-устройство обменивается данными, вы найдете здесь интересующие вас темы.

Что есть в этой книге?

Эта книга состоит из теоретических и практических глав. Для практических глав я разработал и сделал доступной сетевую библиотеку `Canare Core`, которую можно использовать для создания собственных инструментов для анализа и эксплуатации уязвимостей протоколов. Я также представил образец сетевого приложения под названием `SuperFunkyChat`, которое реализует протокол чата между пользователями. Следуя обсуждениям в главах, вы можете использовать это приложение, чтобы изучить навыки анализа протоколов и атаковать образцы сетевых протоколов. Ниже приводится краткое описание каждой главы.

Глава 1. Знакомство с основами

В этой главе описываются основы компьютерных сетей и особый акцент делается на стеке TCP/IP, который составляет основу сетевых протоколов прикладного уровня. В следующих главах предполагается, что вы хорошо разбираетесь в основах построения сетей. В этой главе также представлен подход, который я использую для моделирования протоколов приложений. Эта модель разбивает протокол приложения на гибкие уровни и абстрагирует сложные технические детали, позволяя вам сосредоточиться на отдельных частях протокола, который вы анализируете.

Глава 2. Перехват трафика

В этой главе представлены концепции пассивного и активного перехвата сетевого трафика, и это первая глава, в которой сетевые библиотеки `Canare Core` используются для практических задач.

Глава 3. Структуры сетевых протоколов

В этой главе содержится подробная информация о внутренних структурах, которые распространены в сетевых протоколах, таких как представление чисел или удобочитаемый текст. Когда вы анализируете перехваченный сетевой трафик, то можете использовать эти знания, чтобы быстро определить распространенные структуры, ускоряя тем самым анализ.

Глава 4. Расширенный перехват трафика

В этой главе исследуется ряд более продвинутых методов перехвата, которые дополняют примеры из главы 2. Методы расширенного перехвата включают в себя настройку механизма NAT для перенаправления интересующего вас трафика и спуфинга протокола APR.

Глава 5. Анализ на практике

В этой главе представлены методы анализа перехваченного сетевого трафика с использованием пассивных и активных методов перехвата, описанных в главе 2. Здесь мы используем приложение SuperFunkyChat для генерации трафика.

Глава 6. Обратная разработка

В этой главе описываются методы обратного проектирования программ, подключенных к сети. Обратная разработка позволяет анализировать протокол без необходимости захватывать трафик. Эти методы также помогают определить, как реализовано пользовательское шифрование или запутывание кода, чтобы можно было лучше анализировать перехваченный трафик.

Глава 7. Безопасность сетевого протокола

В этой главе представлена справочная информация о методах и криптографических алгоритмах, используемых для защиты сетевых протоколов. Защита содержимого сетевого трафика от раскрытия или подделки при его передаче по общедоступным сетям имеет первостепенное значение для безопасности сетевых протоколов.

Глава 8. Реализация сетевого протокола

В этой главе объясняются методы реализации сетевого протокола в вашем собственном коде, чтобы вы могли протестировать его поведение и найти слабые места.

Глава 9. Основные причины уязвимостей

В этой главе описаны распространенные уязвимости, с которыми вы можете столкнуться в сетевом протоколе. Когда вы поймете коренные причины уязвимостей, вам будет легче идентифицировать их во время анализа.

Глава 10. Поиск и эксплуатация уязвимостей

В этой главе описываются процессы поиска уязвимостей на базе основных причин, указанных в главе 9, и демонстрируется ряд способов их эксплуатации, включая разработку собственного шелл-кода и обход средств защиты от эксплоитов посредством возвратно-ориентированного программирования.

Приложение. Набор инструментов для анализа сетевых протоколов

В этом приложении вы найдете описания инструментов, которые я обычно использую при выполнении анализа сетевых протоколов. Многие инструменты также кратко описаны в основной части книги.

Как пользоваться этой книгой

Если вы хотите освежить в памяти основы, прочтите сначала главу 1. Когда вы ознакомитесь с основами, переходите к главам 2, 3 и 5, чтобы получить практический опыт в перехвате сетевого трафика и изучить процесс анализа сетевых протоколов.

Зная принципы перехвата сетевого трафика и его анализа, можно перейти к главам с 7 по 10 для получения практической информации о том, как находить и эксплуатировать уязвимости в этих протоколах.

В главах 4 и 6 содержится более подробная информация о дополнительных методах перехвата и обратном проектировании приложений, поэтому если хотите, то можете прочитать их после того, как ознакомитесь с другими главами.

Для выполнения практических примеров вам потребуется установить .NET Core (<https://www.microsoft.com/net/core/>), кросс-платформенную версию среды выполнения .NET от корпорации Microsoft, которая работает в Windows, Linux и macOS. Затем вы можете скачать выпуски для Canape Core на странице <https://github.com/tyranid/CANAPE.Core/releases/> и SuperFunkyChat на странице <https://github.com/tyranid/ExampleChatApplication/releases/>. Они используют .NET Core в качестве среды выполнения. Ссылки на каждый сайт доступны в ресурсах книги на странице <https://www.nostarch.com/networkprotocols/>.

Чтобы выполнить пример сценария Canape Core, необходимо использовать приложение *CANAPE.Cli*, которое будет находиться в пакете релиза, загруженном из репозитория Canape Core на Github. Выполните следующий код в командной строке, заменив *script.csx* именем сценария, который вы хотите выполнить.

```
dotnet exec CANAPE.Cli.dll script.csx
```

Все примеры листингов для практических глав, а также захват пакетов доступны на странице книги по адресу <https://www.nostarch.com/networkprotocols/>.

Лучше всего загрузить эти примеры перед тем, как вы приступите, чтобы можно было следовать за главами без необходимости вводить большой объем исходного кода вручную.

Как связаться со мной

Мне всегда интересно получать как положительные, так и отрицательные отзывы о моей работе, и эта книга не исключение. Вы можете написать мне по адресу attacking.network.protocols@gmail.com, а также подписаться на меня в Twitter – [@tiraniddo](https://twitter.com/tiraniddo) – или подписаться на мой блог на странице <https://tyranidslair.blogspot.com/>, где я публикую некоторые из своих последних передовых исследований в области безопасности.

1

ЗНАКОМСТВО С ОСНОВАМИ

Чтобы атаковать сеть на уровне протоколов, необходимо знать основы. Чем лучше вы понимаете, как устроены и функционируют обычные сети, тем проще будет применить эти знания для перехвата трафика, его анализа и эксплуатации уязвимостей.

В этой главе я познакомлю вас с основными концепциями, с которыми вы сталкиваетесь каждый день при анализе сетевых протоколов. А также заложу основы для их понимания, что упростит поиск ранее неизвестных проблем безопасности во время анализа.

Сетевая архитектура и протоколы

Начнем с обзора базовой терминологии и зададим себе главный вопрос: что такое сеть? *Сеть* – это два или более компьютеров, соединенных между собой для обмена информацией. Обычно каждое подключенное устройство называют *узлом*, чтобы это описание можно было применить к более широкому кругу устройств. На рис. 1.1 приведен очень простой пример.

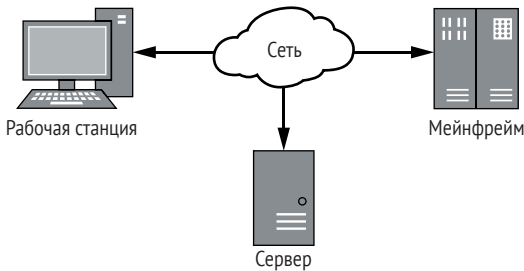


Рис. 1.1. Простая сеть из трех узлов

Здесь показаны три узла, подключенных к общей сети. У каждого узла может быть своя операционная система или оборудование. Но пока каждый узел следует набору правил, или *сетевому протоколу*, он может взаимодействовать с другими узлами в сети. Чтобы обмен данными осуществлялся надлежащим образом, все узлы в сети должны понимать один и тот же сетевой протокол.

Сетевым протоколом выполняется множество функций, включая одну или несколько из перечисленных ниже:

Поддержание состояния сеанса – обычно протоколы реализуют механизмы для создания новых подключений и завершения уже существующих.

Идентификация узлов посредством адресации – данные должны передаваться на правильный узел. Некоторые протоколы реализуют механизм адресации для идентификации конкретных узлов или групп узлов.

Управление потоком – объем данных, передаваемых по сети, ограничен. Протоколы могут реализовывать способы управления потоком данных для увеличения пропускной способности и уменьшения задержки.

Гарантия порядка передачи данных – многие сети не гарантируют, что порядок отправки данных будет соответствовать порядку, в котором они будут получены. Протокол может изменить порядок данных, чтобы убедиться, что они будут доставлены правильно.

Обнаружение и исправление ошибок – многие сети не являются надежными на 100 %, и данные могут быть повреждены. Важно обнаружить повреждение и, в идеале, исправить его.

Форматирование и кодирование данных – данные не всегда находятся в формате, подходящем для передачи их по сети. Протокол может указывать способы кодирования данных, например кодирование текста на английском языке в двоичные значения.

Набор интернет-протоколов

TCP/IP – это протокол де-факто, используемый современными сетями. Хотя можно рассматривать TCP/IP как единый протокол, на са-

мом деле это комбинация двух протоколов: *протокола управления передачей* (TCP) и *интернет-протокола* (IP). Оба они являются частью набора интернет-протоколов (IPS), концептуальной модели того, как сетевые протоколы отправляют сетевой трафик через интернет. Таким образом, обмен данными можно разделить на четыре уровня, как показано на рис. 1.2.

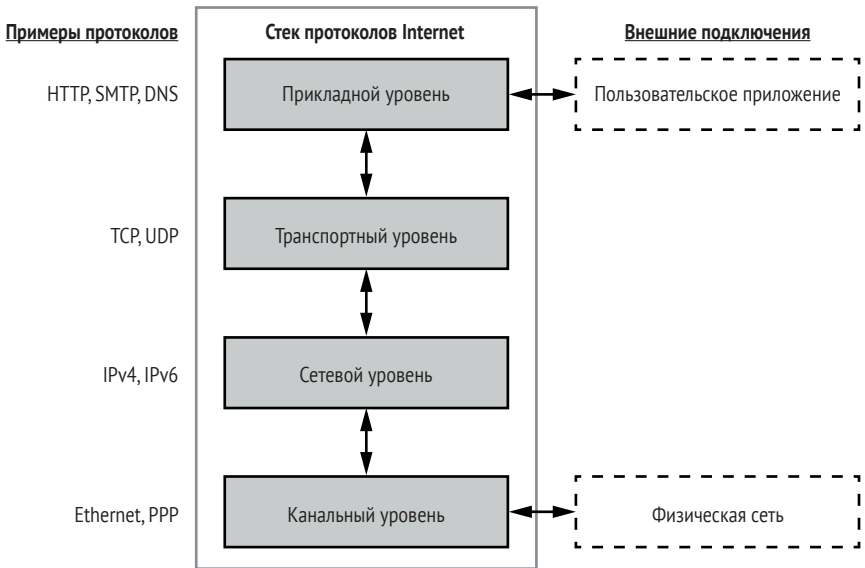


Рис. 1.2. Уровни набора интернет-протоколов

Эти четыре уровня образуют *стек протоколов*. В следующем списке приводится объяснение каждого из этих уровней.

Канальный уровень (уровень 1) – является самым низким уровнем и описывает физические механизмы, используемые для передачи информации между узлами в локальной сети. Хорошо известные примеры включают Ethernet (проводной и беспроводной) и протокол Point-to-Point (PPP).

Сетевой уровень (уровень 2) – предоставляет механизмы для определения пути передачи данных. В отличие от уровня 1, узлы не обязательно должны находиться в локальной сети. Этот уровень содержит IP; в современных сетях фактически используемый протокол может быть либо версией 4 (IPv4), либо версией 6 (IPv6).

Транспортный уровень (уровень 3) – отвечает за соединения между клиентами и серверами, иногда обеспечивая правильный порядок пакетов и предоставляя мультиплексирование сервисов. Мультиплексирование сервисов позволяет одному узлу поддерживать несколько различных сервисов, присваивая каждому сервису разные номера; этот номер называется *портом*. На этом уровне работают протоколы TCP и UDP.

Прикладной уровень (уровень 4) – содержит сетевые протоколы, такие как *протокол передачи гипертекста (HTTP)*, который передает содержимое веб-страниц; *простой протокол передачи почты (SMTP)*, передающий электронную почту; и *протокол системы доменных имен (DNS)*, который преобразует имя в узел в сети. В этой книге мы сосредоточимся главным образом на этом уровне.

Каждый уровень взаимодействует только с уровнем, который располагается выше и ниже него, но должны осуществляться и внешние взаимодействия со стеком. На рис. 1.2 показаны два внешних соединения. Канальный уровень взаимодействует с физическим сетевым соединением, передавая данные в физической среде, например электрические импульсы или импульсы света. Прикладной уровень взаимодействует с пользовательским приложением: *приложение* представляет собой набор связанных функций, которые предоставляют сервис пользователю. На рис. 1.3 показан пример приложения, обрабатывающего электронную почту. Сервис, предоставляемый почтовым приложением, – это отправка и получение сообщений по сети.

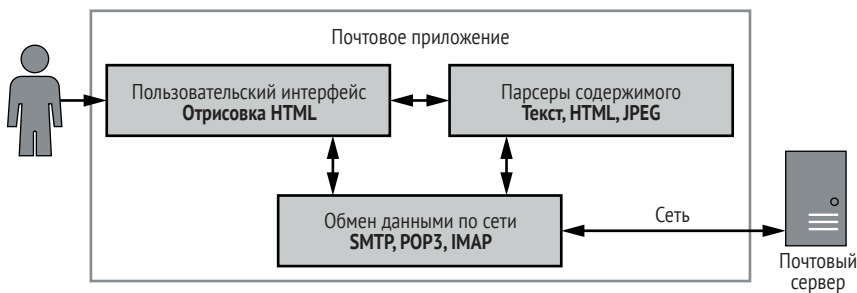


Рис. 1.3. Пример почтового приложения

Обычно приложения содержат следующие компоненты:

Обмен данными по сети – этот компонент обменивается данными по сети и обрабатывает входящие и исходящие данные. Для почтового приложения обмен данными по сети, скорее всего, является стандартным протоколом, таким как SMTP или POP3.

Парсеры содержимого – данные, передаваемые по сети, обычно включают в себя содержимое, которое необходимо извлечь и обработать. Это могут быть текстовые данные, такие как тело электронного письма, или изображения или видео.

Пользовательский интерфейс – позволяет пользователю просматривать полученные электронные письма и создавать новые письма для передачи. В почтовом приложении пользовательский интерфейс может отображать электронные письма с использованием HTML в веб-браузере.

Обратите внимание, что пользователь, взаимодействующий с пользовательским интерфейсом, не обязательно должен быть человеком.

Это может быть другое приложение, автоматизирующее отправку и получение писем посредством инструмента командной строки.

Инкапсуляция данных

Каждый уровень в IPS построен на уровне, находящемся ниже, и каждый уровень может инкапсулировать данные, полученные от вышестоящего уровня, чтобы те могли перемещаться между уровнями. Данные, передаваемые каждым уровнем, называются *блоком данных протокола (PDU)*.

Заголовки, концевики и адреса

На каждом уровне блок данных протокола содержит передаваемые полезные данные. Обычно к полезным данным добавляется *заголовок*, содержащий необходимую информацию для этих передаваемых данных, такую как *адреса* узлов источника и назначения в сети. Иногда у PDU также есть *концевик*, который добавляется к полезным данным и содержит значения, необходимые для обеспечения правильной передачи, например информацию для проверки ошибок. На рис. 1.4 показано, как блоки данных протокола размещаются в IPS.

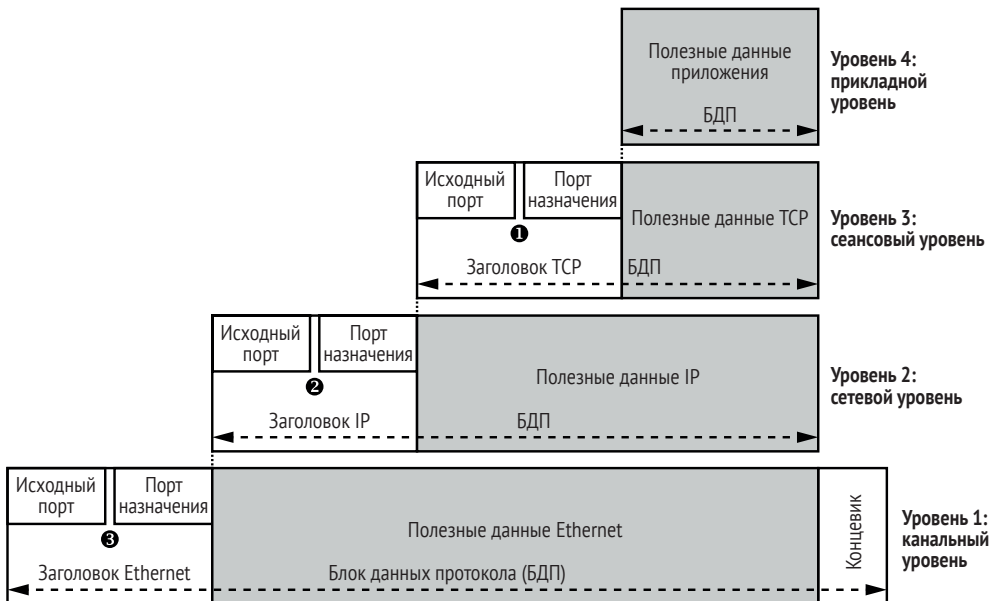


Рис. 1.4. Инкапсуляция данных IPS

Заголовок TCP содержит номер исходного порта и порта назначения 1. Благодаря этим номерам у одного узла может быть несколько уникальных сетевых соединений. Номера портов для протокола TCP (и UDP) находятся в диапазоне от 0 до 65 535.