Email Filtering and Mitigating Circumvention Techniques

Steven McLeod

Michael Cohen

Computer Network Vulnerability Team

Defence Signals Directorate

Department of Defence

May 2004

Agenda

- Introduction
  - Why email filtering?
  - Two goals of email filtering.
- Bypassing email filters, and preventing such activity.
  - Numerous examples that explore the following protocols and file formats while providing context – taking advantage of file formats to include executable code designed to bypass email content filters.
    - Internet Message Format – RFC2822

Agenda

- .bmp
  - bitmap specification explored in detail, including inserting executable code in bitmaps.
- .zip
  - zip specification explored in detail, including hiding zip files.
- pdf/doc/rtf/wri/html/zip
  - Embedding malicious content.

Why email filtering?

- Antivirus software has its place – but that place is not alone on the front line.
  - Viruses are spreading faster than a pattern can be issued and applied, and we have seen originally created malicious content targeted at the victim, hence it doesn't match a known pattern and hasn’t triggered AV software with heuristic detection.
- Mail content rewriting is more effective than just blocking or forwarding email.
  - Unpack email, remove prohibited files, sanitise active content e.g. javascript, repack and forward to recipient.
  - Rewriting circumvents several exploit techniques.

Two goals of email filtering

- Like a firewall, an email filter’s job is to enforce a policy, and is only as good as the policy.
- Consider the following goals when designing the policy:
  - Prevent delivery of emails of an executable nature that only require the non security conscious user to double click for the executable content to run.
    - e.g. attachments ending in exe/com/pif/scr or zip files containing such files.
  - Prevent delivery of emails of an executable nature that are somehow disguised, and require social engineering of the user to extract and run the executable content.
    - e.g. .exe renamed to .txt, password protected zip file.
    - Internal users use such tricks to circumvent traditional email filters. An attacker is therefore an internet or external user who is attempting to circumvent the “no executable content allowed” policy, regardless of their motive.

Bypassing email filters, and preventing such activity

- Bypassing email filters
  - Extracts from a test suite I compiled with ~400 test cases created to test the networks of our government customers.
- Preventing such activity
  - Insights into the email filtering software we wrote to better meet our requirements than the COTS products we have encountered so far.
    - Written in python by my colleague Michael Cohen. A self booting Linux CD that reads its configuration file from a floppy disk.

Bypassing email filters, and preventing such activity

- **Typical block/fail action is to replace the prohibited file with a text file describing the matching policy rule, and forward the original email (but with the replaced file) to the original email recipient.**
  - **Can be configured to not deliver emails matching certain criteria, which is useful for viruses that propagate via email. Users don't want 100 emails that were previously infected but have been neutered.**

Agenda - update

- Blocking attachments by extension.
- Allowing by file type.
- Combining extension, file type, and content-type checks.
- Testing RFC 2822 (Internet message format)
- Detecting malicious content in otherwise allowed files (pdf/doc/rtf/wri/html/zip).
- Operating system dependent problems.
- Email server tests.

Blocking attachments by extension

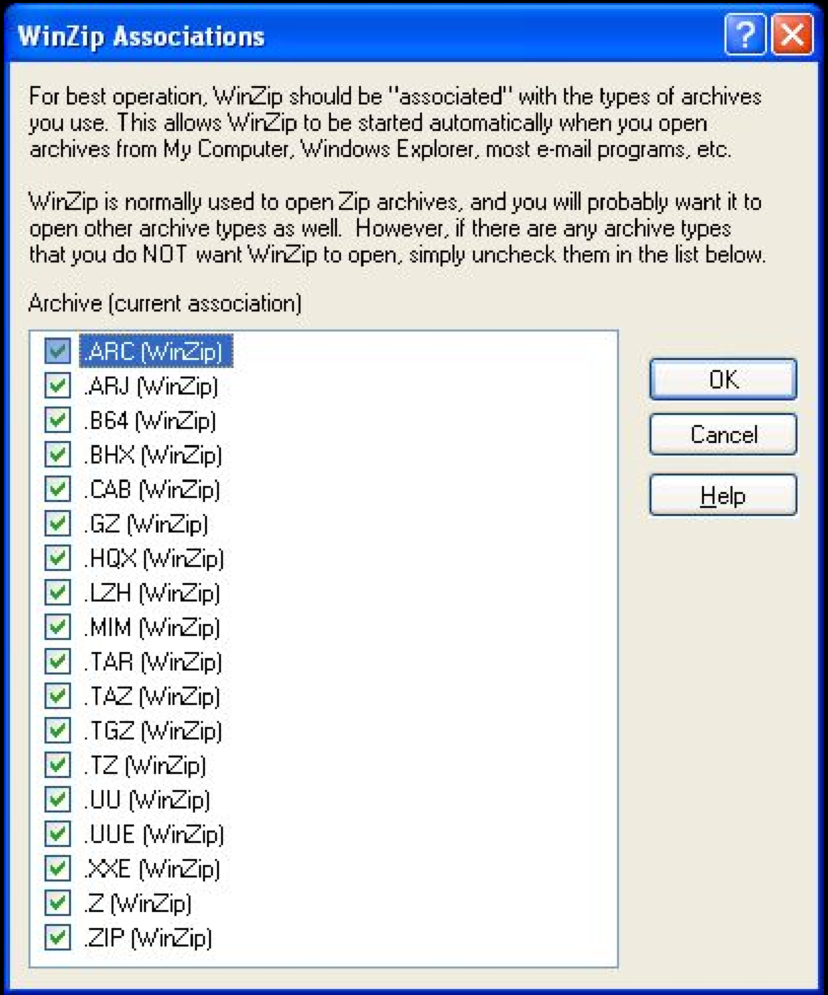
- Like a firewall rule set, define what you allow and reject the rest.
  - e.g. doc/xls/png/jpg/gif/bmp/pdf/…
- If you attempt to list potentially malicious file extensions
  - e.g. bat/cmd/com/exe/hta/js/jse/pif/reg/scr/shs/vbe/vbs/wsh/…

Blocking attachments by extension

- You will inevitably miss some
  - msg/oft/eml attachments are emails themselves and may contain file attachments with potentially malicious file extensions. Furthermore, a saved draft Outlook Express email (.eml) file containing a malicious file attachment is saved in plain text format.
  - msi/msp = Windows Installer Package/Patch

Blocking attachments by extension

– **uue/cab/bhx and other formats that software on the user's workstation can decode/decompress with just a double click.**



WinZip Associations

For best operation, WinZip should be "associated" with the types of archives you use. This allows WinZip to be started automatically when you open archives from My Computer, Windows Explorer, most e-mail programs, etc.

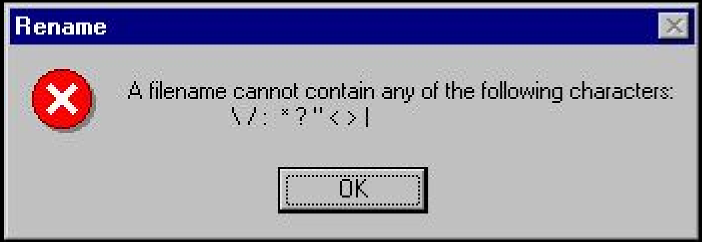
WinZip is normally used to open Zip archives, and you will probably want it to open other archive types as well. However, if there are any archive types that you do NOT want WinZip to open, simply uncheck them in the list below.

Archive (current association)

- ☑ .ARC (WinZip)
- ☑ .ARJ (WinZip)
- ☑ .B64 (WinZip)
- ☑ .BHX (WinZip)
- ☑ .CAB (WinZip)
- ☑ .GZ (WinZip)
- ☑ .HQX (WinZip)
- ☑ .LZH (WinZip)
- ☑ .MIM (WinZip)
- ☑ .TAR (WinZip)
- ☑ .TAZ (WinZip)
- ☑ .TGZ (WinZip)
- ☑ .TZ (WinZip)
- ☑ .UU (WinZip)
- ☑ .UUE (WinZip)
- ☑ .XXE (WinZip)
- ☑ .Z (WinZip)
- ☑ .ZIP (WinZip)

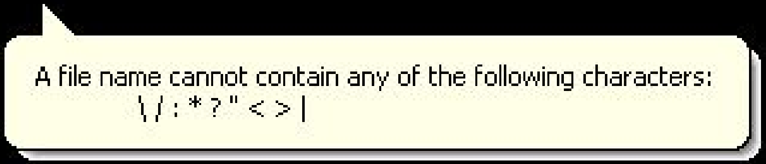
Blocking attachments by extension

– File extensions containing characters reserved by Microsoft



Rename

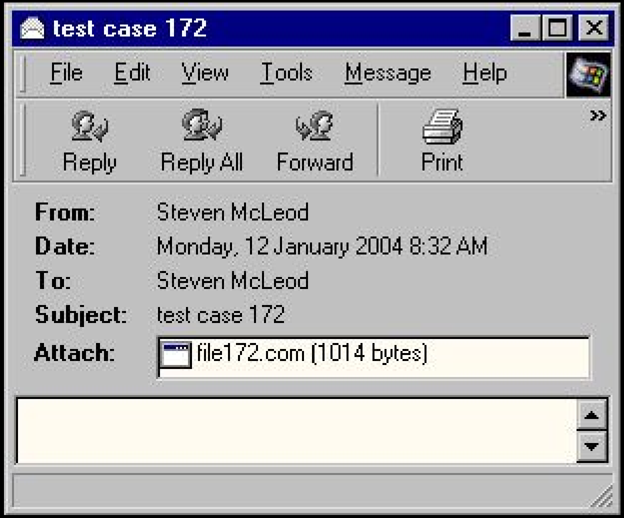
A filename cannot contain any of the following characters:
\ / : * ? " < > |



A file name cannot contain any of the following characters:
\ / : * ? " < > |

- Some email clients silently strip out reserved characters, while other email clients replace them with an underscore:

```
Content-Type: application/octet-stream;
        name="file172.com?"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="file172.com?"
```



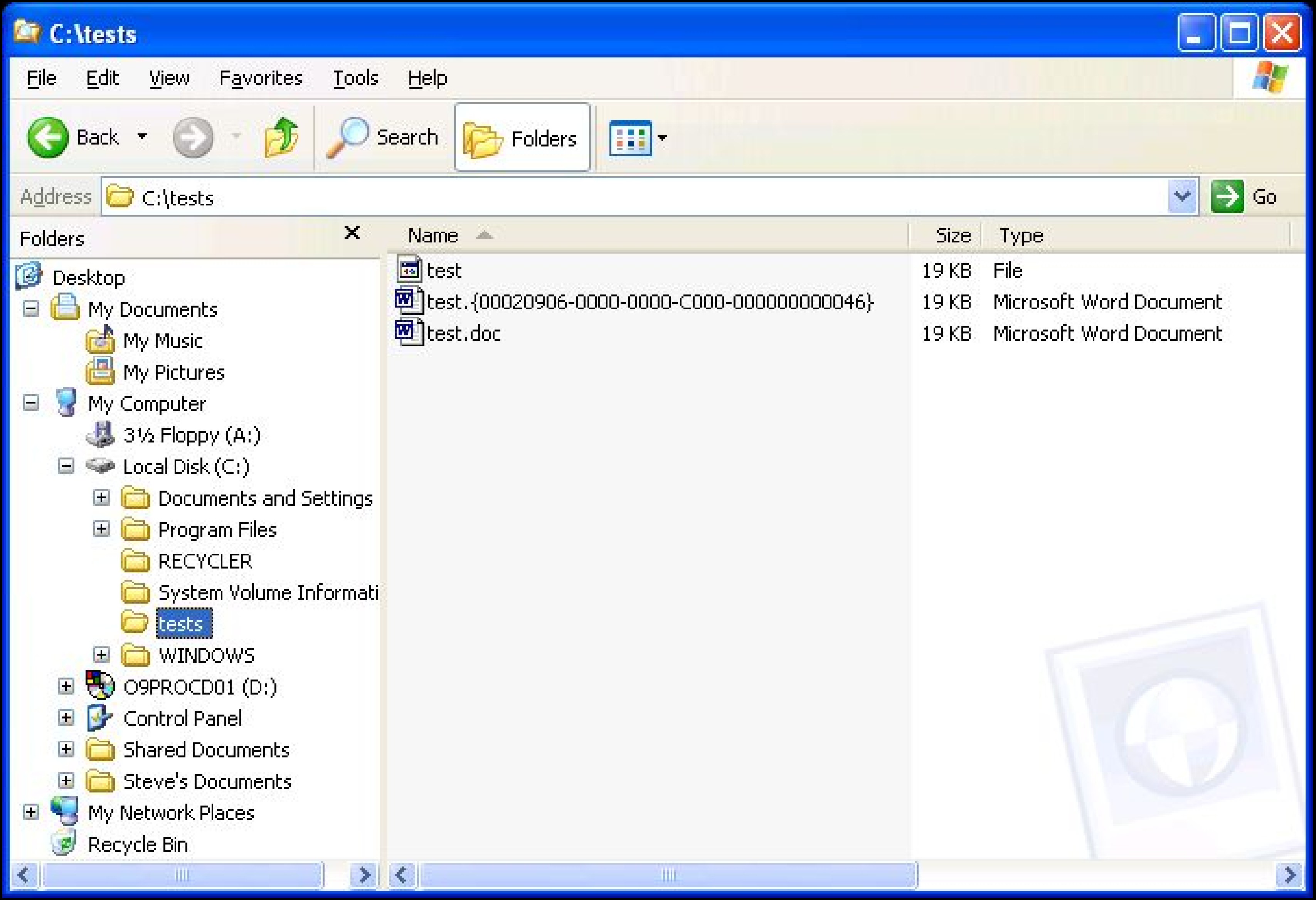
test case 172

From: Steven McLeod
Date: Monday, 12 January 2004 8:32 AM
To: Steven McLeod
Subject: test case 172
Attach: file172.com (1014 bytes)

Blocking attachments by extension

- Files with no filename
- Files with no extension
- Files with uppercase extensions e.g. somefile.eXe
- Files with multiple extensions e.g. somefile.bmp.com
- Files with a classid as an extension
  - .{00020906-0000-0000-C000-000000000046} = .doc

- **Demonstration**
  - Example files with a classid extension or no extension are opened by the appropriate application.



Name	Size	Type
test	19 KB	File
test.{00020906-0000-0000-C000-000000000046}	19 KB	Microsoft Word Document
test.doc	19 KB	Microsoft Word Document

Blocking attachments by extension

- It is evident that allowing file extensions deemed to be safe is preferable to trying to block all the possibly malicious extensions.

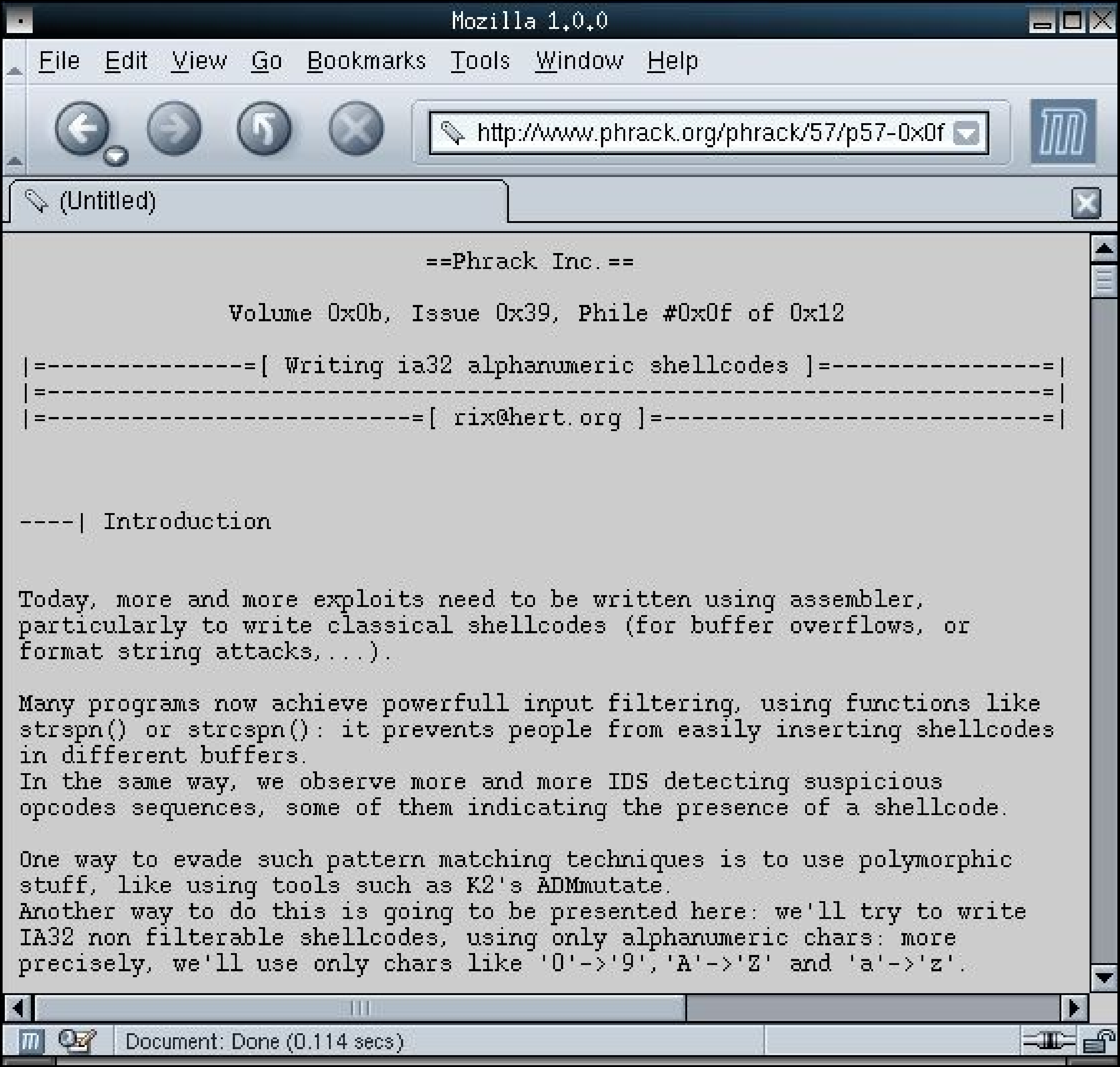
Allowing by file type

- Used to prevent users circumventing the email policy by renaming the cute .scr screen saver they absolutely must share with their friends to a file with a .txt extension.

- A virus may propagate via email with a “non-malicious” file extension, and the email social engineers the user into renaming the file.

Allowing by file type

- File typing does have some limitations.
  - A saved draft Outlook Express email file containing a malicious file attachment is saved in plain text format.
  - A .com file renamed to .txt will typically be allowed by email filters since .com files have no fixed format and no fixed sequence of bytes (signature) that they are guaranteed to begin with.
    - Our mail filter specifies that a .txt file should only contain characters able to be typed on the keyboard (including carriage return and line feed). This will block the vast majority of .com files renamed to .txt though some code consists of only alphanumeric characters.



Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://www.phrack.org/phrack/57/p57-0x0f
(Untitled)
Document: Done (0.114 secs)

```
                         ==Phrack Inc.==

               Volume 0x0b, Issue 0x39, Phile #0x0f of 0x12

|=--------------=[ Writing ia32 alphanumeric shellcodes ]=---------------=|
|=-----------------------------------------------------------------------=|
|=------------------------=[ rix@hert.org ]=-----------------------------=|
```

----| Introduction

Today, more and more exploits need to be written using assembler, particularly to write classical shellcodes (for buffer overflows, or format string attacks,...).

Many programs now achieve powerfull input filtering, using functions like strspn() or strcspn(): it prevents people from easily inserting shellcodes in different buffers.
In the same way, we observe more and more IDS detecting suspicious opcodes sequences, some of them indicating the presence of a shellcode.

One way to evade such pattern matching techniques is to use polymorphic stuff, like using tools such as K2's ADMmutate.
Another way to do this is going to be presented here: we'll try to write IA32 non filterable shellcodes, using only alphanumeric chars: more precisely, we'll use only chars like '0'->'9','A'->'Z' and 'a'->'z'.

Allowing by file type: bitmap files

- More advanced file type evasion techniques.
  - A legitimate bitmap file that is also an executable program. Program can be executed by changing file extension, or possibly automatically executed using a vulnerability in an email client (or web browser) e.g. specify content-type as application/octet-stream instead of image/bmp.
    - Perhaps the email program will present a dialog box to the user, asking them if they want to run/open picture.bmp. The user, believing that bitmap files just contain non-executable pixel data, will click Yes, resulting in the file being executed (as opposed to the bitmap file being displayed by an image viewer).

Allowing by file type: bitmap files

- http://www.coco3.com/text/doc_BMP.txt

```
typedef struct tagBITMAPFILEHEADER {    /* bmfh */
   UINT    bfType;
   DWORD   bfSize;
   UINT    bfReserved1;
   UINT    bfReserved2;
   DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

The BITMAPFILEHEADER structure contains information about the type, size, and layout of a device-independent bitmap (DIB) file.

Member	Description
bfType	Specifies the type of file. This member must be BM.
bfSize	Specifies the size of the file, in bytes.
bfReserved1	Reserved; must be set to zero.
bfReserved2	Reserved; must be set to zero.
bfOffBits	Specifies the byte offset from the BITMAPFILEHEADER structure to the actual bitmap data in the file.

Allowing by file type: bitmap files

Member	Description
bfType	Specifies the type of file. This member must be BM.
bfSize	Specifies the size of the file, in bytes.

```
DWORD   bfSize;
```

- **Originally I overwrote three of the four size bytes with a jump instruction to my code at the end of the bitmap.**
  - **Security is a cat and mouse game.**
  - **Bad news**
    - **No image viewer seemed to use the size value e.g. for allocating memory corresponding to the size of the bitmap file, so this attack vector worked.**
      - **There is a more useful field for image viewers in the following header structure:**
        biSizeImage   Specifies the size, in bytes, of the image.
  - **Good news**
    - **A sophisticated email content filter could check if bfSize = the actual size of the bitmap file.**

Allowing by file type: bitmap files

- Bad news
  - Such a check could be defeated by appending junk to the end of the bitmap file. The junk is ignored by all of the bitmap viewers I tested since the bitmap viewers know how many bytes they need to read.
- Good news
  - A very sophisticated email content filter may be able to detect the extra data (executable code and padding) if it understands the bitmap file format. For example, check the picture height, width and *image* size values in the header, and ensure there is just enough data in the bitmap file to describe each pixel, with no data (i.e. executable code or padding) left over.
- Bad news
  - An attacker could modify the the picture height, width and image size values in the header.... etc.. etc.. etc.. etc..

Allowing by file type: bitmap files

– Steve, how come you are *still* talking about bitmap files??

- Users are constantly looking for ways to defeat the “overly restrictive” security mechanisms implemented by the “bastard” system administrator.

- They can use this technique to send uudecode.com inside a bitmap file from their home Internet account to their work email address. Then they can send the uuencoded version of uuencode.com from home to their work account. Afterwards, they can send/receive any file of any type to/from their friends or home account by uuencoding/uudecoding it, resulting in them emailing “safe” plain text .txt file attachments or embedding the uuencoded data as plain text in the email itself with no attachment.

- EICAR test virus is represented as the following plain text:

```
begin 644 eicar.com
M6#5/(5`E0$%06S1<4%I8-30H4%XI-T-#*3=])$5)0T%2+5-404Y$05)$+4%.
95$E625)54RU415-4+49)3$4A)$@K2"H-"@``
`
end
```

Allowing by file type: bitmap files

– Bad news

- I devised another method to include executable code in a bitmap file designed to evade the email content filter. Let’s have another look at the bitmap specifications.

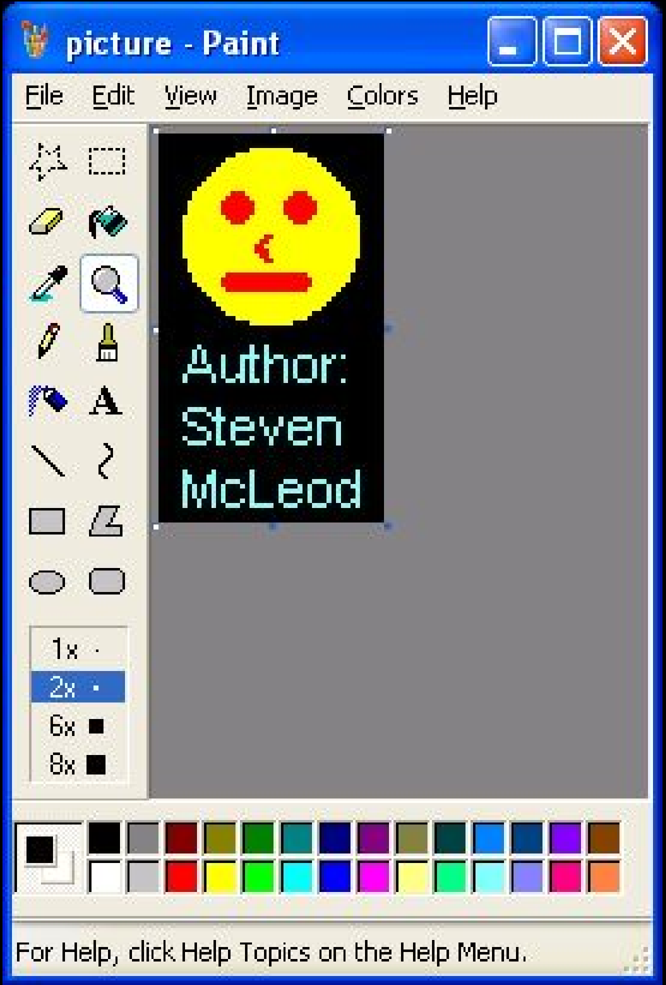
```
typedef struct tagBITMAPFILEHEADER {    /* bmfh */
   ....
   DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

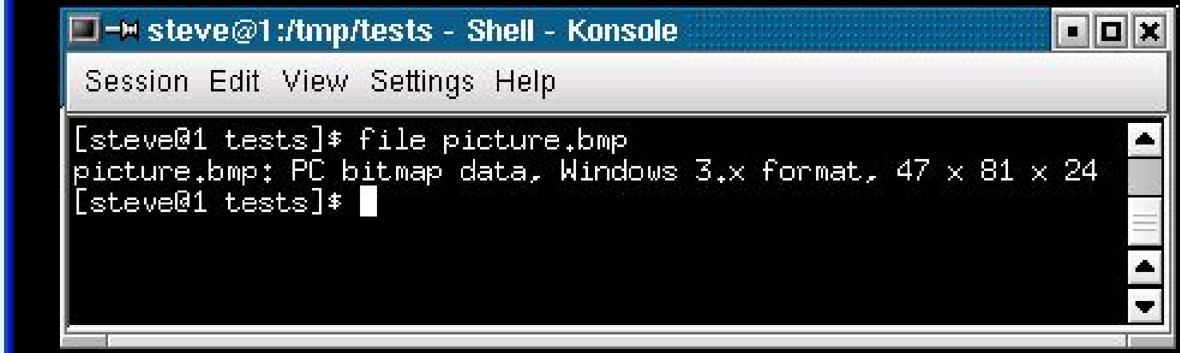
bfOffBits Specifies the byte offset from the BITMAPFILEHEADER structure to the actual bitmap data in the file.

- This allows me to create a bitmap file in the format of BITMAPFILEHEADER, then the BITMAPINFOHEADER part of BITMAPINFO, executable content, followed by the “actual bitmap data in the file”.
  - In this example, the code is close to the start of the bitmap file instead of at the end, and only requires two bytes to jump to it, resulting in the ability to create a smaller (less than 65k) bitmap file which conforms entirely with the bitmap specifications and may not require padding to ensure that the total file size is the same size as the jump instruction.

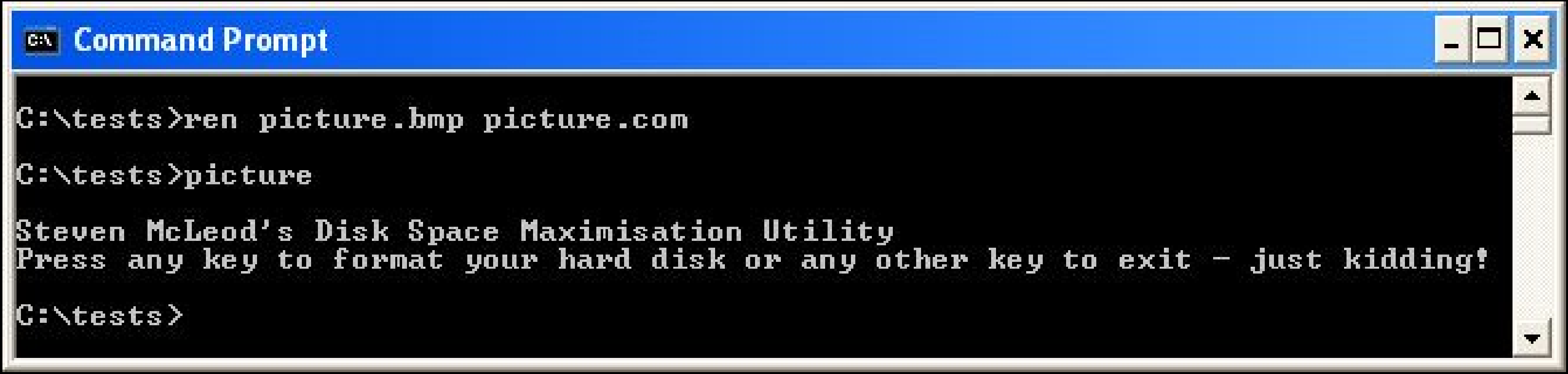
Allowing by file type: bitmap files

- **Demonstration**
  - **Bitmap file containing potentially malicious executable code.**

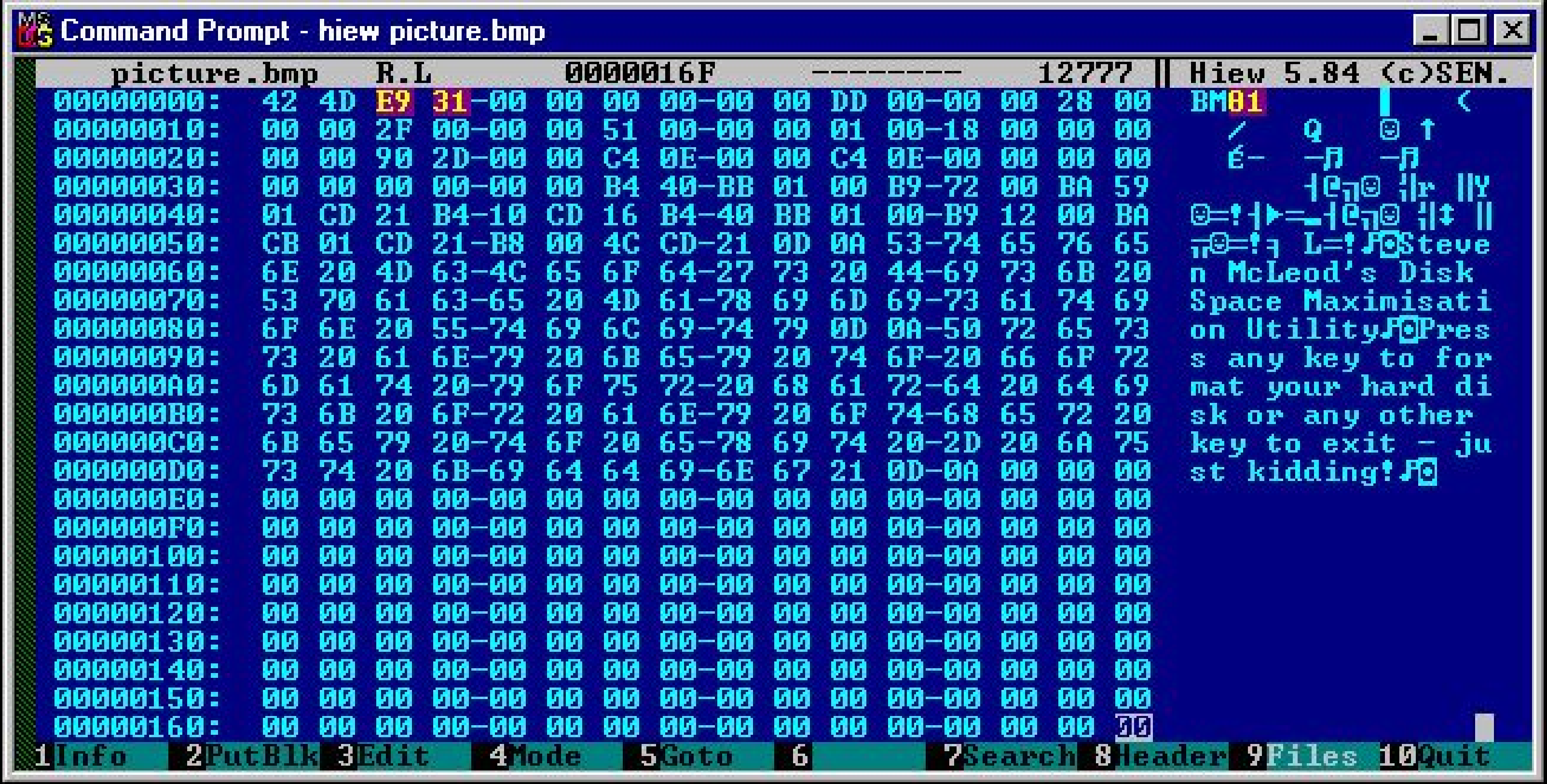

picture - Paint
File Edit View Image Colors Help
Author:
Steven
McLeod
1x
2x
6x
8x
For Help, click Help Topics on the Help Menu.


steve@1:/tmp/tests - Shell - Konsole
Session Edit View Settings Help

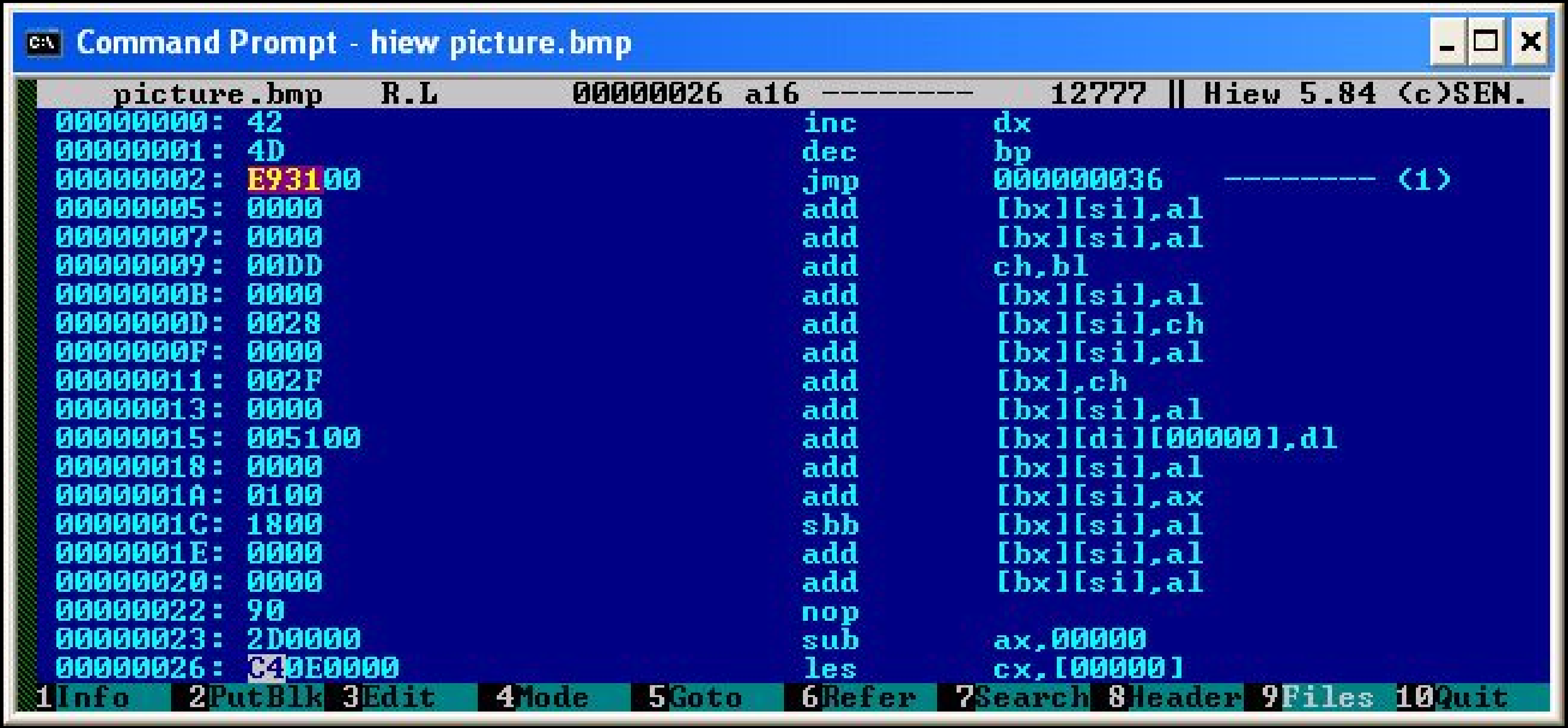
```
[steve@1 tests]$ file picture.bmp
picture.bmp: PC bitmap data, Windows 3.x format, 47 x 81 x 24
[steve@1 tests]$
```



```
C:\tests>ren picture.bmp picture.com

C:\tests>picture

Steven McLeod's Disk Space Maximisation Utility
Press any key to format your hard disk or any other key to exit - just kidding!

C:\tests>
```



```
   picture.bmp   R.L      0000016F      ---------      12777 ‖ Hiew 5.84 (c)SEN.
 00000000:  42 4D E9 31-00 00 00 00-00 00 DD 00-00 00 28 00  BMΘ1      ▌   (
 00000010:  00 00 2F 00-00 00 51 00-00 00 01 00-18 00 00 00    /   Q   ☺ ↑
 00000020:  00 00 90 2D-00 00 C4 0E-00 00 C4 0E-00 00 00 00    É-  ─♫  ─♫
 00000030:  00 00 00 00-00 00 B4 40-BB 01 00 B9-72 00 BA 59        ┤@╗☺ ╣r ║Y
 00000040:  01 CD 21 B4-10 CD 16 B4-40 BB 01 00-B9 12 00 BA  ☺═!┤►═▬┤@╗☺ ╣↕ ║
 00000050:  CB 01 CD 21-B8 00 4C CD-21 0D 0A 53-74 65 76 65  ╦☺═!╕ L═!♪◙Steve
 00000060:  6E 20 4D 63-4C 65 6F 64-27 73 20 44-69 73 6B 20  n McLeod's Disk
 00000070:  53 70 61 63-65 20 4D 61-78 69 6D 69-73 61 74 69  Space Maximisati
 00000080:  6F 6E 20 55-74 69 6C 69-74 79 0D 0A-50 72 65 73  on Utility♪◙Pres
 00000090:  73 20 61 6E-79 20 6B 65-79 20 74 6F-20 66 6F 72  s any key to for
 000000A0:  6D 61 74 20-79 6F 75 72-20 68 61 72-64 20 64 69  mat your hard di
 000000B0:  73 6B 20 6F-72 20 61 6E-79 20 6F 74-68 65 72 20  sk or any other
 000000C0:  6B 65 79 20-74 6F 20 65-78 69 74 20-2D 20 6A 75  key to exit - ju
 000000D0:  73 74 20 6B-69 64 64 69-6E 67 21 0D-0A 00 00 00  st kidding!♪◙
 000000E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 000000F0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000100:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000110:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000120:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000130:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000140:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000150:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 00000160:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
 1Info   2PutBlk 3Edit   4Mode   5Goto   6       7Search 8Header 9Files 10Quit
```

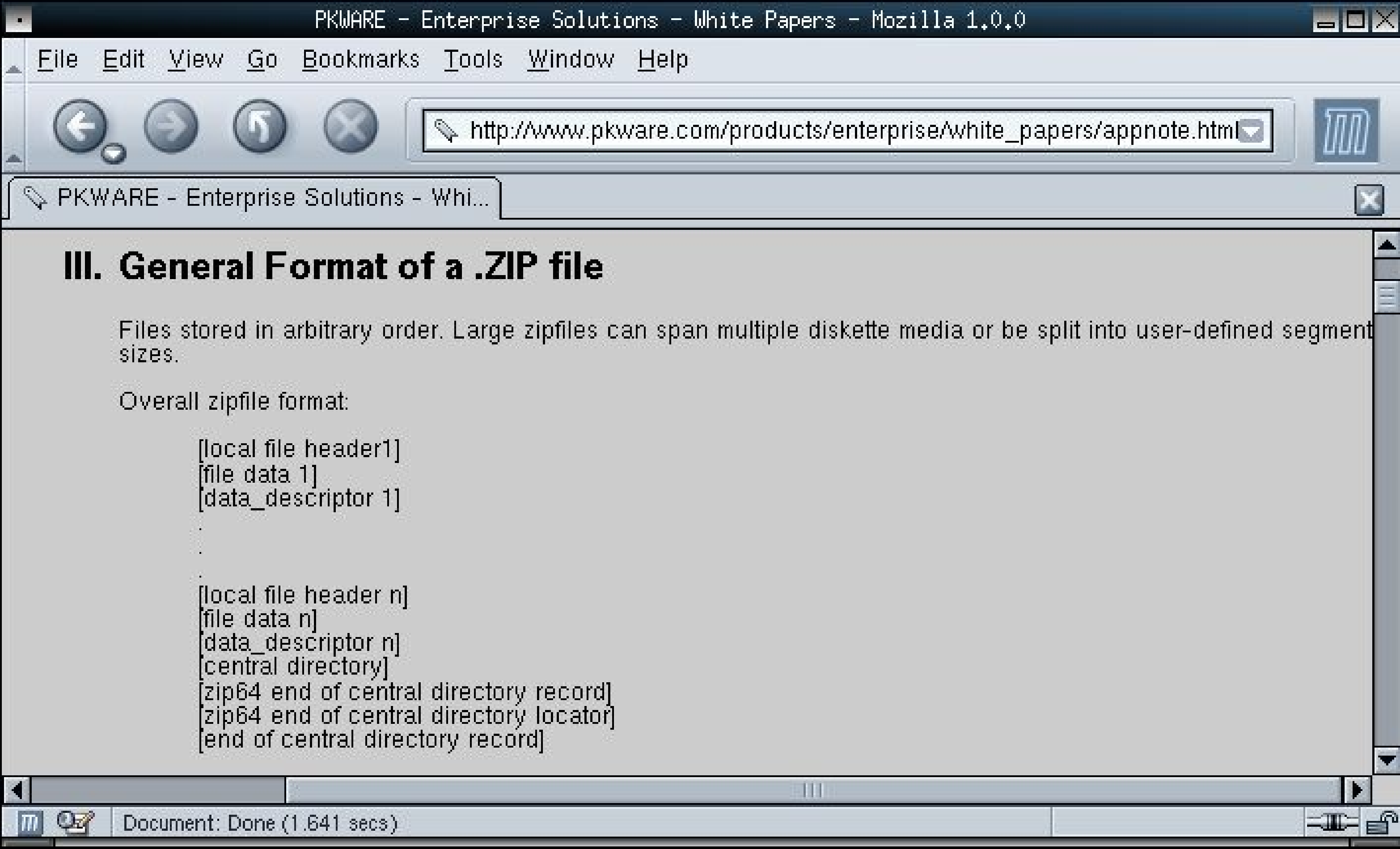


```
    picture.bmp   R.L      00000026 a16 ---------      12777 || Hiew 5.84 (c)SEN.
 00000000: 42                                    inc        dx
 00000001: 4D                                    dec        bp
 00000002: E93100                                jmp        000000036   --------- (1)
 00000005: 0000                                  add        [bx][si],al
 00000007: 0000                                  add        [bx][si],al
 00000009: 00DD                                  add        ch,bl
 0000000B: 0000                                  add        [bx][si],al
 0000000D: 0028                                  add        [bx][si],ch
 0000000F: 0000                                  add        [bx][si],al
 00000011: 002F                                  add        [bx],ch
 00000013: 0000                                  add        [bx][si],al
 00000015: 005100                                add        [bx][di][00000],dl
 00000018: 0000                                  add        [bx][si],al
 0000001A: 0100                                  add        [bx][si],ax
 0000001C: 1800                                  sbb        [bx][si],al
 0000001E: 0000                                  add        [bx][si],al
 00000020: 0000                                  add        [bx][si],al
 00000022: 90                                    nop
 00000023: 2D0000                                sub        ax,00000
 00000026: C40E0000                              les        cx,[00000]
```

1Info 2PutBlk 3Edit 4Mode 5Goto 6Refer 7Search 8Header 9Files 10Quit

Allowing by file type: bmp+zip

- Change focus from bmp to zip, starting with an example that includes both file formats.
  - A zip file appended to a bitmap file will have a file type of bitmap, and will therefore be allowed by traditional email content filters.
  - Tested zip utilities (Winzip, linux unzip, pkunzip) ignore the bitmap data and see a valid zip file. Why is this?
    - Reviewing the source code to linux unzip shows that it reads at least 65557 bytes (in case there is a comment in the zip file) *backwards* from the end of the zip file, searching for the end of central directory record. This record includes the offset in the zip file where the central directory record is located.

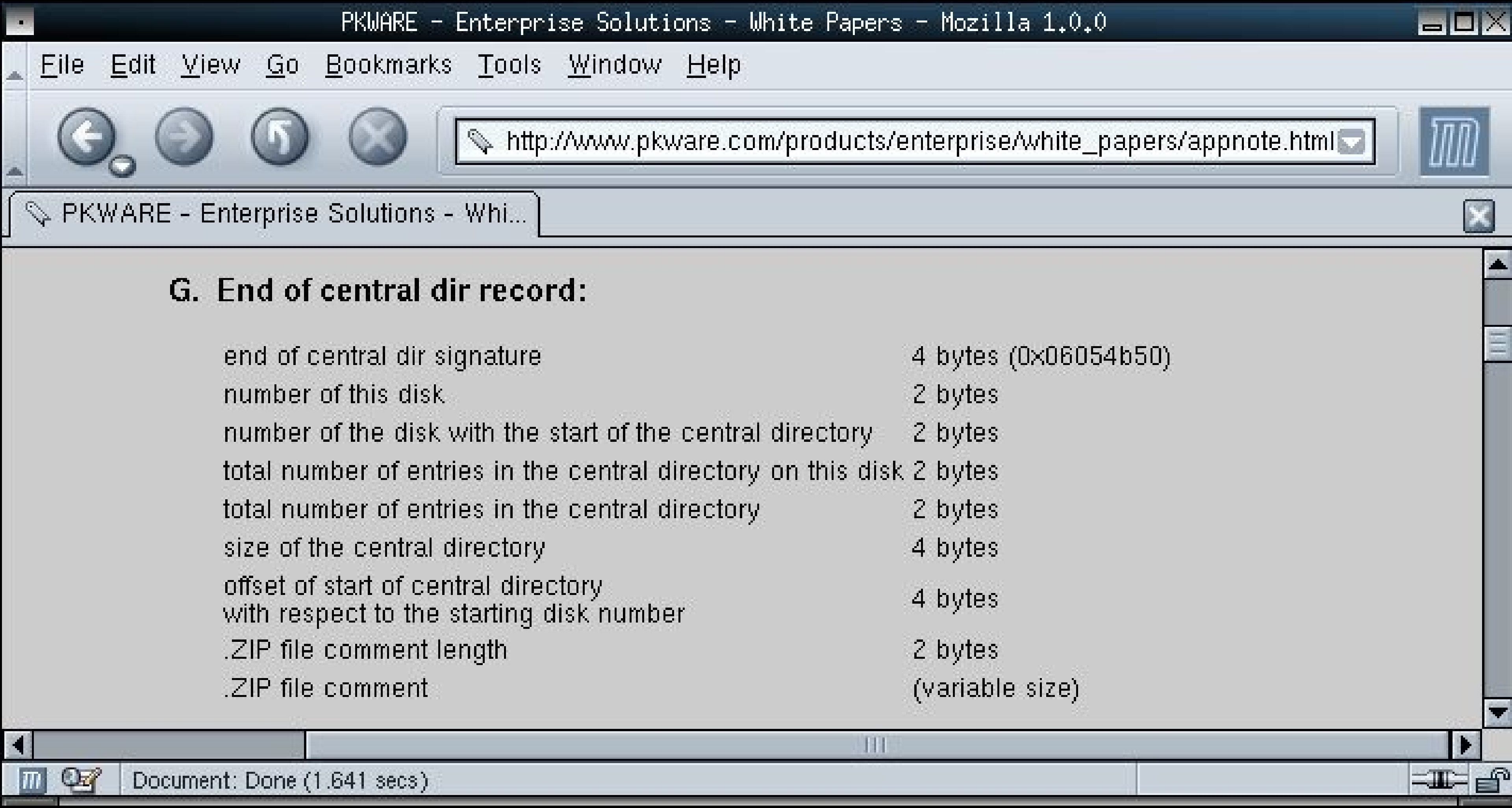

PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

III. General Format of a .ZIP file

Files stored in arbitrary order. Large zipfiles can span multiple diskette media or be split into user-defined segment sizes.

Overall zipfile format:

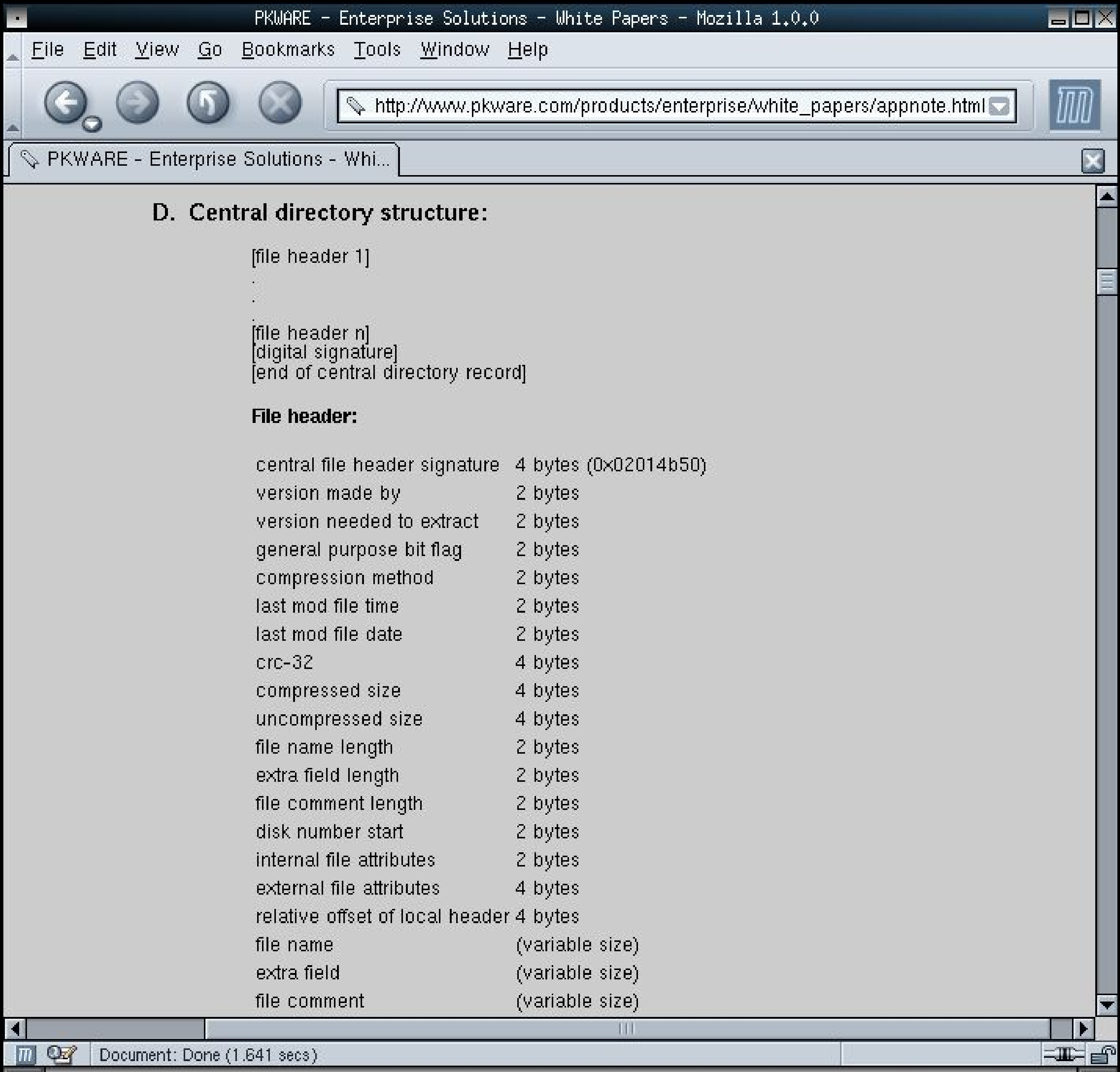
[local file header1]
[file data 1]
[data_descriptor 1]
.
.
.
[local file header n]
[file data n]
[data_descriptor n]
[central directory]
[zip64 end of central directory record]
[zip64 end of central directory locator]
[end of central directory record]



PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File
Edit
View
Go
Bookmarks
Tools
Window
Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

G. End of central dir record:

end of central dir signature	4 bytes (0x06054b50)
number of this disk	2 bytes
number of the disk with the start of the central directory	2 bytes
total number of entries in the central directory on this disk	2 bytes
total number of entries in the central directory	2 bytes
size of the central directory	4 bytes
offset of start of central directory with respect to the starting disk number	4 bytes
.ZIP file comment length	2 bytes
.ZIP file comment	(variable size)


PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

D. Central directory structure:

[file header 1]
.
.
.
[file header n]
[digital signature]
[end of central directory record]

File header:

central file header signature	4 bytes (0x02014b50)
version made by	2 bytes
version needed to extract	2 bytes
general purpose bit flag	2 bytes
compression method	2 bytes
last mod file time	2 bytes
last mod file date	2 bytes
crc-32	4 bytes
compressed size	4 bytes
uncompressed size	4 bytes
file name length	2 bytes
extra field length	2 bytes
file comment length	2 bytes
disk number start	2 bytes
internal file attributes	2 bytes
external file attributes	4 bytes
relative offset of local header	4 bytes
file name	(variable size)
extra field	(variable size)
file comment	(variable size)

Extract of source code to linux unzip utility.

Overview:

process_zipfiles()

calls do_seekable

do_seekable calls find_ecrec

do_seekable then calls uz_end_central

Source code extract:

```
/*--------------------------------------------------------------------------

  process.c

  This file contains the top-level routines for processing multiple zipfiles.

  ---------------------------------------------------------------------------*/

[..]

/**************************/

/* Function do_seekable() */

/**************************/

[..]
```

```
/*---------------------------------------------------------------------------
    Open the zipfile for reading in BINARY mode to prevent CR/LF translation,
    which would corrupt the bit streams.
  ---------------------------------------------------------------------------*/
[..]
/*---------------------------------------------------------------------------
    Find and process the end-of-central-directory header.  UnZip need only
    check last 65557 bytes of zipfile:  comment may be up to 65535, end-of-
    central-directory record is 18 bytes, and signature itself is 4 bytes;
    add some to allow for appended garbage.  Since ZipInfo is often used as
    a debugging tool, search the whole zipfile if zipinfo_mode is true.
  ---------------------------------------------------------------------------*/
[..]
```

calls find_ecrec() which consists of:

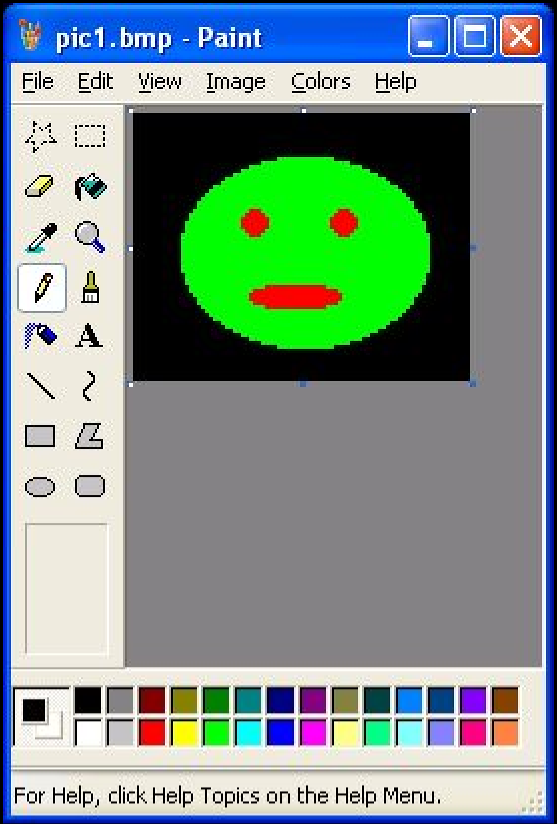
```
    /*-----------------------------------------------------------------------
        Loop through blocks of zipfile data, starting at the end and going
        toward the beginning.  In general, need not check whole zipfile for
        signature, but may want to do so if testing.
      -----------------------------------------------------------------------*/
[..]
```

calls uz_end_central() which consists of:

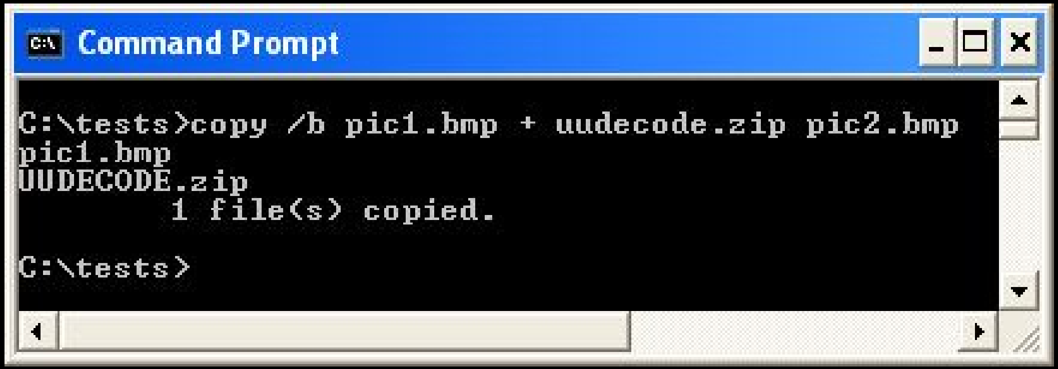
```
    /*---------------------------------------------------------------------------
        Get the zipfile comment (up to 64KB long), if any, and print it out.
        Then position the file pointer to the beginning of the central directory
        and fill buffer.
      ---------------------------------------------------------------------------*/
[..]
/*-----------------------------------------------------------------------
    list, extract or test member files as instructed, and close the
    zipfile.
  -----------------------------------------------------------------------*/
[..]
} /* end function do_seekable() */
```

Allowing by file type: bmp+zip

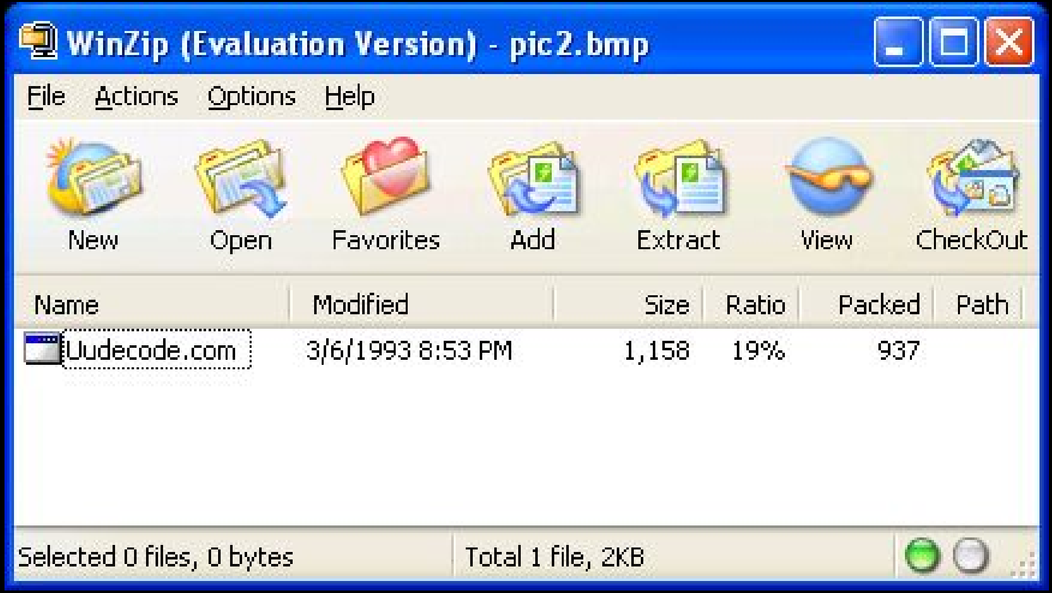
- Demonstration
  - Bitmap file prepended to a zip file containing potentially malicious executable code.



For Help, click Help Topics on the Help Menu.



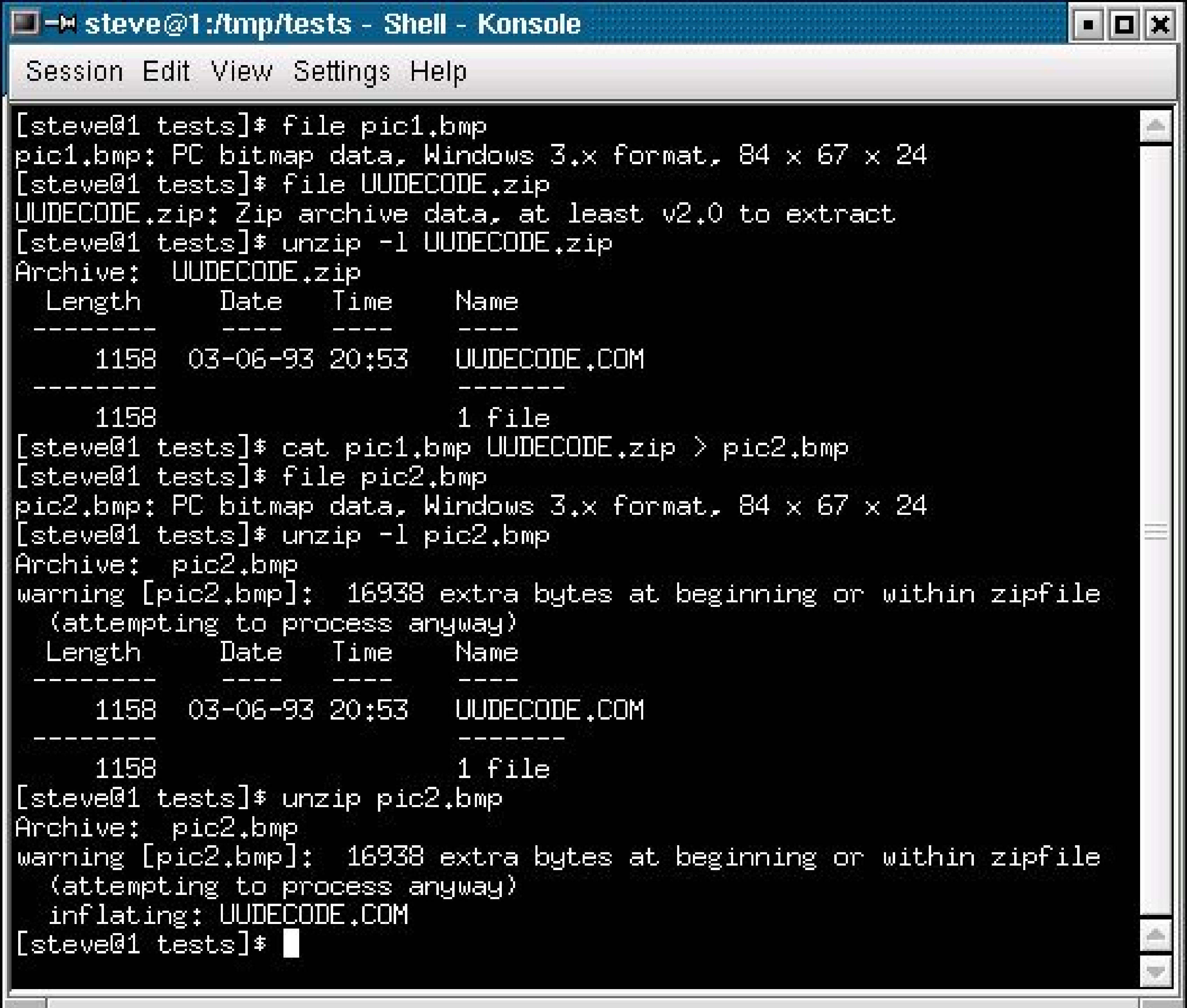
```
C:\tests>copy /b pic1.bmp + uudecode.zip pic2.bmp
pic1.bmp
UUDECODE.zip
        1 file(s) copied.

C:\tests>
```


WinZip (Evaluation Version) - pic2.bmp
File Actions Options Help
New
Open
Favorites
Add
Extract
View
CheckOut

Name	Modified	Size	Ratio	Packed	Path
Uudecode.com	3/6/1993 8:53 PM	1,158	19%	937	

Selected 0 files, 0 bytes

Total 1 file, 2KB



steve@1:/tmp/tests - Shell - Konsole
Session Edit View Settings Help

```
[steve@1 tests]$ file pic1.bmp
pic1.bmp: PC bitmap data, Windows 3.x format, 84 x 67 x 24
[steve@1 tests]$ file UUDECODE.zip
UUDECODE.zip: Zip archive data, at least v2.0 to extract
[steve@1 tests]$ unzip -l UUDECODE.zip
Archive:  UUDECODE.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
     1158  03-06-93 20:53   UUDECODE.COM
 --------                   -------
     1158                   1 file
[steve@1 tests]$ cat pic1.bmp UUDECODE.zip > pic2.bmp
[steve@1 tests]$ file pic2.bmp
pic2.bmp: PC bitmap data, Windows 3.x format, 84 x 67 x 24
[steve@1 tests]$ unzip -l pic2.bmp
Archive:  pic2.bmp
warning [pic2.bmp]:  16938 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  Length     Date   Time    Name
 --------    ----   ----    ----
     1158  03-06-93 20:53   UUDECODE.COM
 --------                   -------
     1158                   1 file
[steve@1 tests]$ unzip pic2.bmp
Archive:  pic2.bmp
warning [pic2.bmp]:  16938 extra bytes at beginning or within zipfile
  (attempting to process anyway)
  inflating: UUDECODE.COM
[steve@1 tests]$ 
```

Allowing by file type: bmp+zip

- Good news
  - Our email content filter addresses this problem by running unzip on all file attachments regardless of their extension or file type.
- Bad news
  - Two variants of combining a bitmap file with a zip file containing an executable file.
    - Combine a zip file containing a text file with a zip file containing an executable file.
    - Combine a zip file containing an executable file with a zip file containing a text file.

Allowing by file type: zip+zip

- In these cases where a zip file is actually two zip files combined, some unzip implementations will see the zip file with the text file, while others will see the zip file with the executable file. As long as the mail filter unzip utility behaves differently to the unzip utility on the email recipient's workstation, executable content can be emailed to a user and subsequently executed, and the email content scanner only saw a text file.

– Good news

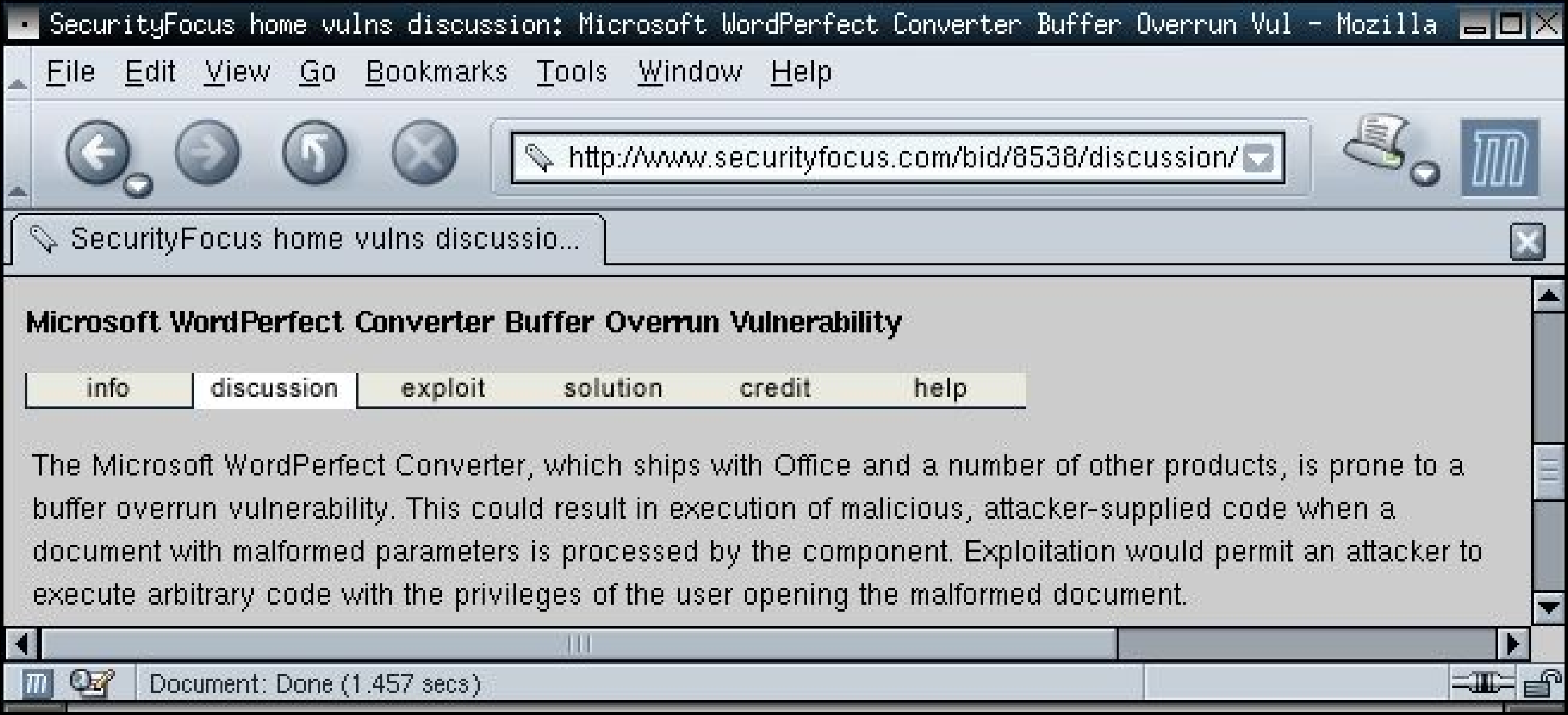
- Solution: our email content filter repackages zip files, removing the original zip file attachment and replacing it with a newly created zip file.
  - If it sees a zip file containing a text file, it creates a new zip file, places the text file in it, attaches it to the email and forwards it to the email recipient.
  - If it sees a zip file containing an executable file, it creates a new zip file, places the text file blocked_message.txt in it, attaches it to the email and forwards it to the email recipient.

Combining extension, file type, and content-type checks.

- **Combine checking techniques to create a complete rule set.**
- **Don’t just allow:**
  “files with a .doc extension” or
  “files of type Microsoft Office Document” or
  “file attachments with a content-type of type application/msword” (especially since the content-type can be set to an arbitrary value by an attacker.

Combining extension, file type, and content-type checks.

- **For increased accuracy, combine the rules – only allow a file attachment with .doc extension if it is also of type Microsoft Office Document and (preferably) has a content-type of application/msword.**
  - **e.g. this combination provided proactive protection against the publicly circulated exploit for the recent serious vulnerability in the Microsoft WordPerfect Converter used by Word and other Microsoft products.**



SecurityFocus home vulns discussion: Microsoft WordPerfect Converter Buffer Overrun Vul - Mozilla

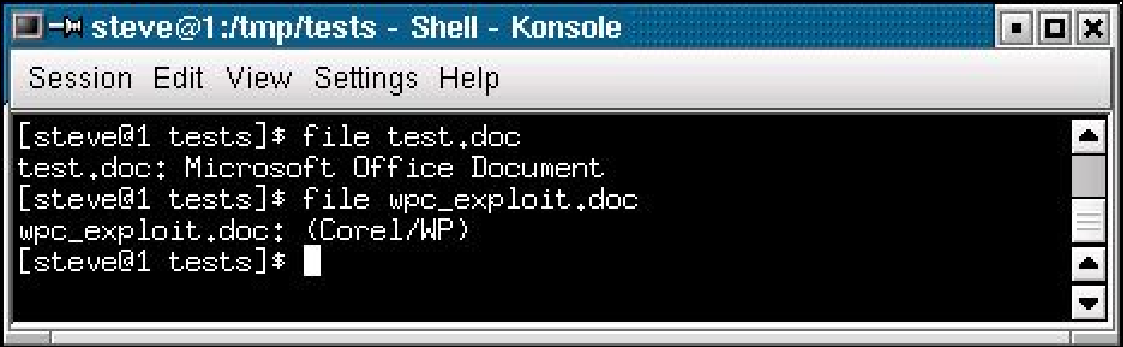
http://www.securityfocus.com/bid/8538/discussion/

SecurityFocus home vulns discussio...

Microsoft WordPerfect Converter Buffer Overrun Vulnerability

The Microsoft WordPerfect Converter, which ships with Office and a number of other products, is prone to a buffer overrun vulnerability. This could result in execution of malicious, attacker-supplied code when a document with malformed parameters is processed by the component. Exploitation would permit an attacker to execute arbitrary code with the privileges of the user opening the malformed document.

Document: Done (1.457 secs)



steve@1:/tmp/tests - Shell - Konsole

```
[steve@1 tests]$ file test.doc
test.doc: Microsoft Office Document
[steve@1 tests]$ file wpc_exploit.doc
wpc_exploit.doc: (Corel/WP)
[steve@1 tests]$
```

Testing RFC 2822 (Internet message format)

- Test various email encoding combinations
  - uuencoded, mime, quoted printable, base64, 8 bit headers
  - The file name file100.com can be represented as
    - Content-Type: application/octet-stream;
      name="=?ISO-8859-1?Q?=66=69=6C=65=31=30=30=2E=63=6F=6D?="
      Content-Transfer-Encoding: base64
      Content-Disposition: attachment;
      filename="=?ISO-8859-1?Q?=66=69=6C=65=31=30=30=2E=63=6F=6D?="
  - Consider file.bmp.com encoded as
    - name="=?ISO-8859-1?Q?= 66=69=6C=65=2E=62=6D=70=2E=63=6F=6D?="
    - name="=?ISO-8859-1?Q?= 66=69=6C=65=2E=62=6D=70=0D=2E=63=6F=6D?="
    - name="=?ISO-8859-1?Q?= 66=69=6C=65=2E=62=6D=70=0A=2E=63=6F=6D?="
    - name="=?ISO-8859-1?Q?= 66=69=6C=65=2E=62=6D=70=0D=0A=2E=63=6F=6D?="

Testing RFC 2822 (Internet message format)

- **Constructs with implementation dependent handling. Attackers want the email content filter to behave differently to the email client on the email recipient’s workstation.**
  - **File names such as:**
    - **file40.com<CR>.txt   where <CR> is carriage return**
    - **file41.com<LF>.txt   where <LF> is line feed**
    - **file40.txt<CR>.com**
    - **file41.txt<LF>.com**
    - **file112.bmp".com   email looks like: filename="file112.bmp".com"**

Testing RFC 2822 (Internet message format)

– File name can be specified in four different places.

-
  ```
  Content-Type: application/msword;
          name="file347.com"; filename="file347.doc"
  Content-Transfer-Encoding: base64
  Content-Disposition: attachment;
          filename="file347.doc"; name="file347.doc"
  ```

-
  ```
  Content-Type: application/msword;
          name="file348.doc"; filename="file348.com"
  Content-Transfer-Encoding: base64
  Content-Disposition: attachment;
          filename="file348.doc"; name="file348.doc“
  ```
- etc....

Testing RFC 2822 (Internet message format)

– Introduced errors and non-conformance

- 2.2. Header Fields
  Header fields are lines composed of a field name, followed by a colon (":"), followed by a field body, and terminated by CRLF. A field name MUST be composed of printable US-ASCII characters (i.e., characters that have values between 33 and 126, inclusive), except colon. A field body may be composed of any US-ASCII characters, except for CR and LF.

-
  ```
  Subject: test case 360
  MIME-Version: 1.0
  Content-Type: multipart/mixed;
          boundary="----=_NextPart_000_000A_01C3A6F9.17A0FD81"
  Content-Type : multipart/mixed;
          boundary="----=_NextPart_000_000A_01C3A6F9.17A0FD80"
  X-MimeOLE: Just another header
  [..]
  ------=_NextPart_000_000A_01C3A6F9.17A0FD80
  Content-Type: application/octet-stream;
          name="file360.com"
  Content-Transfer-Encoding: base64
  Content-Disposition: attachment;
          filename="file360.com"

  62SQDQpJbnB1dCBwYXRoL2ZpbGU6ICBJbnB1dCBmaWxlIGVycm9yLk91dHB1dCBmaWxlIGVy
  cm9y
  ```

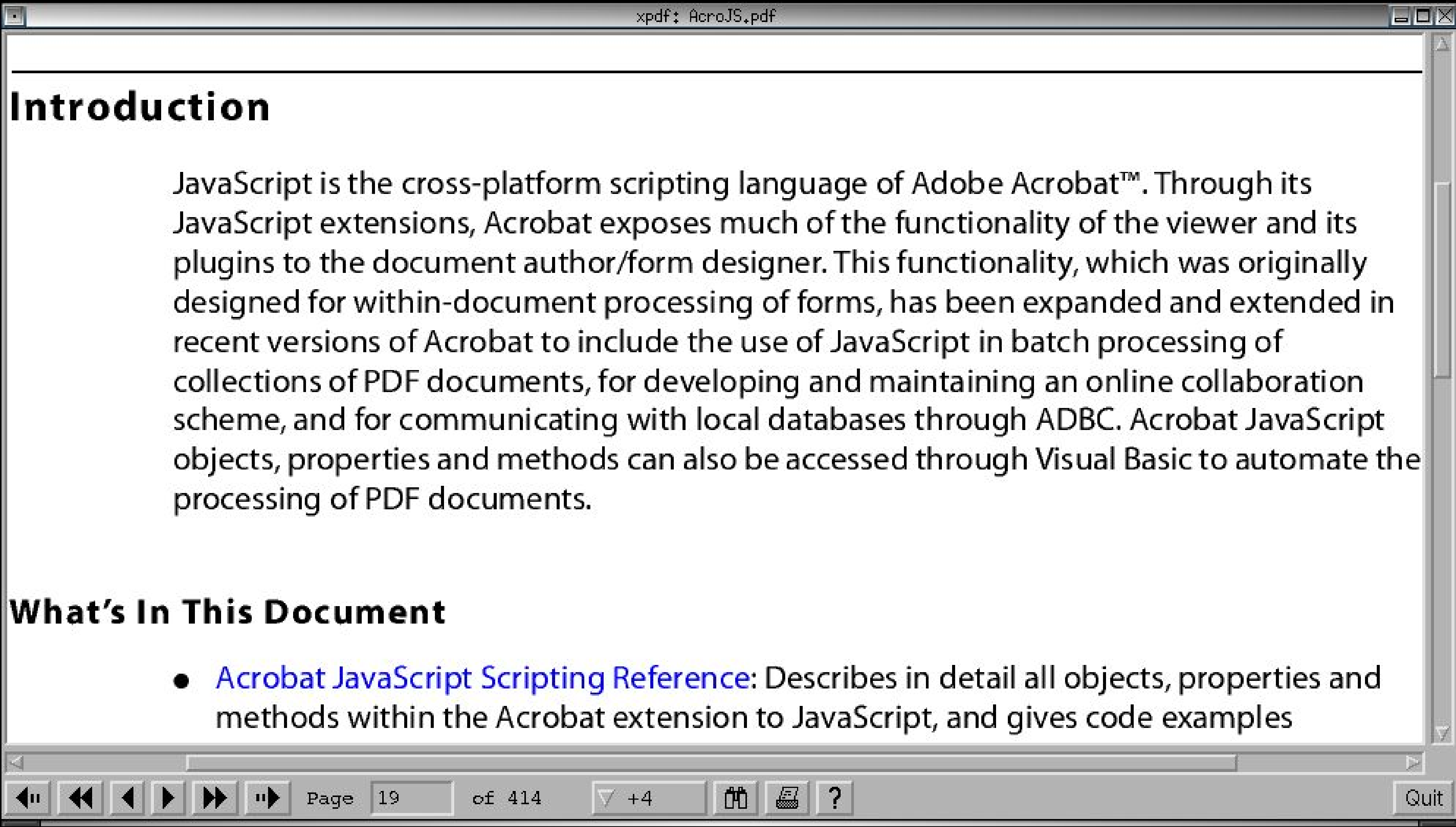
Testing RFC 2822 (Internet message format)

– A small com file with no file name, content-disposition set to inline, content-type set to

- application/octet-stream
- image/bmp
- message/rfc822

Detecting malicious content in otherwise allowed files: pdf

- **Additional testing is required for some types of files that have an allowed file extension and an allowed file type.**
  - **Check if a .pdf file contains potentially malicious active content.**
    - **Depending on the version of Acrobat installed, active content in a . pdf file can send emails without user interaction using Adobe's extension to javascript.**
      - **Acrobat JavaScript Scripting Reference http://partners.adobe.com/asn/acrobat/sdk/public/docs/AcroJS.pdf**

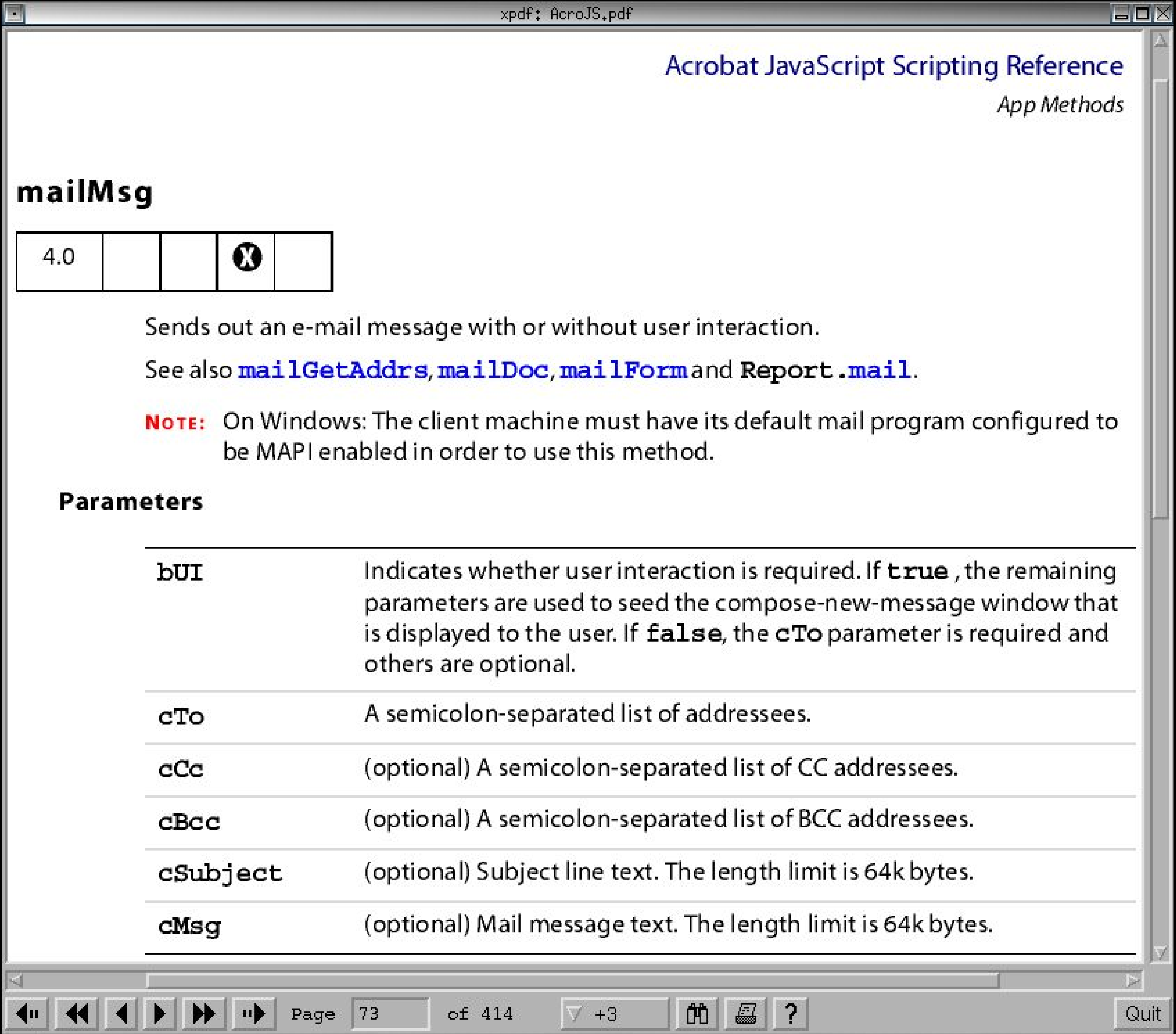


Introduction

JavaScript is the cross-platform scripting language of Adobe Acrobat™. Through its JavaScript extensions, Acrobat exposes much of the functionality of the viewer and its plugins to the document author/form designer. This functionality, which was originally designed for within-document processing of forms, has been expanded and extended in recent versions of Acrobat to include the use of JavaScript in batch processing of collections of PDF documents, for developing and maintaining an online collaboration scheme, and for communicating with local databases through ADBC. Acrobat JavaScript objects, properties and methods can also be accessed through Visual Basic to automate the processing of PDF documents.

What's In This Document

- Acrobat JavaScript Scripting Reference: Describes in detail all objects, properties and methods within the Acrobat extension to JavaScript, and gives code examples



mailMsg

4.0			ⓧ	

Sends out an e-mail message with or without user interaction.

See also `mailGetAddrs`, `mailDoc`, `mailForm` and `Report.mail`.

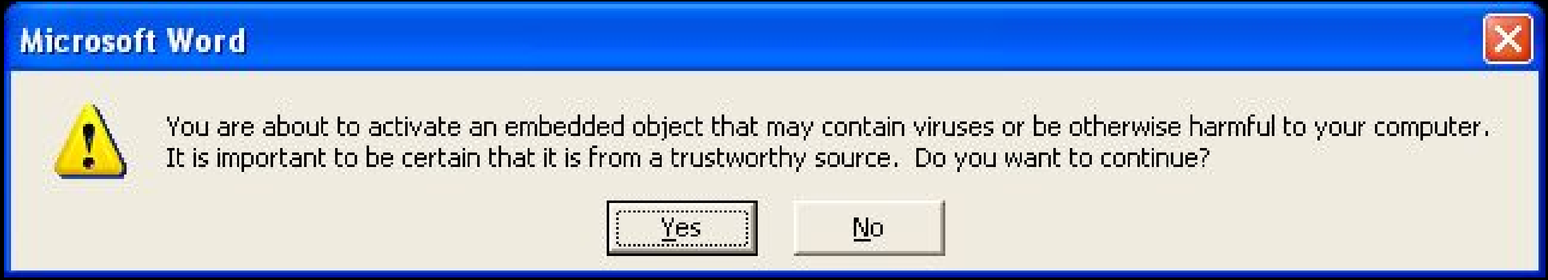
NOTE: On Windows: The client machine must have its default mail program configured to be MAPI enabled in order to use this method.

Parameters

`bUI`	Indicates whether user interaction is required. If `true` , the remaining parameters are used to seed the compose-new-message window that is displayed to the user. If `false`, the `cTo` parameter is required and others are optional.
`cTo`	A semicolon-separated list of addressees.
`cCc`	(optional) A semicolon-separated list of CC addressees.
`cBcc`	(optional) A semicolon-separated list of BCC addressees.
`cSubject`	(optional) Subject line text. The length limit is 64k bytes.
`cMsg`	(optional) Mail message text. The length limit is 64k bytes.

Detecting malicious content in otherwise allowed files: doc/rtf/wri

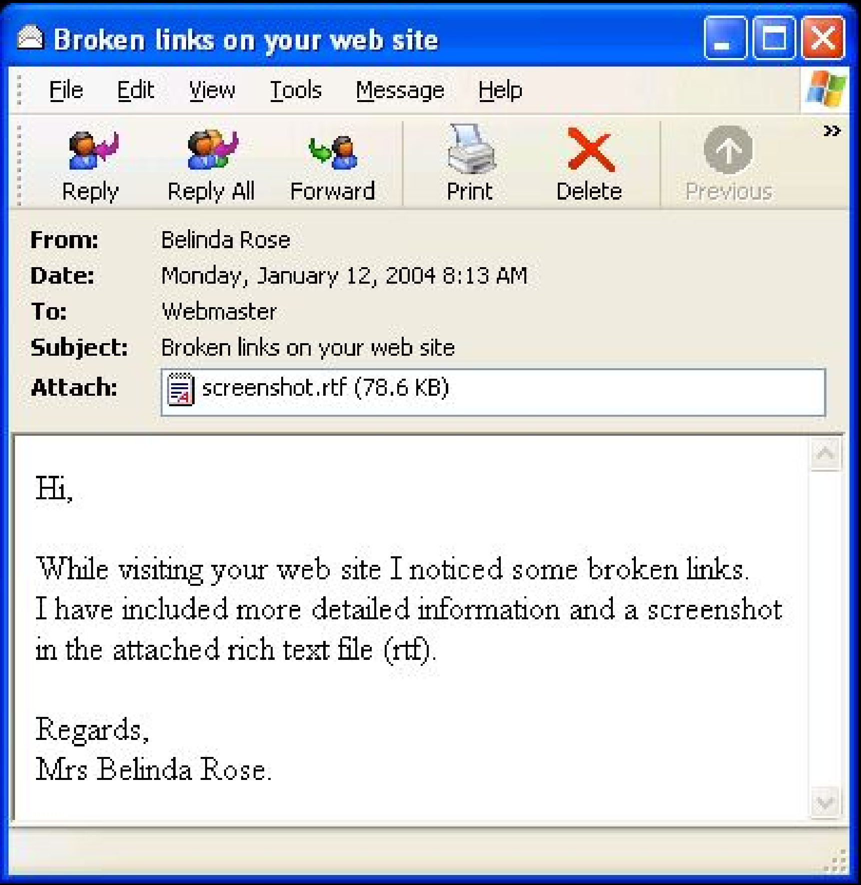
- Check if a .doc file contains potentially malicious active content such as macros.
- Check if a doc/rtf/wri contains an embedded/packaged object consisting of executable code.
  - Microsoft Word warns the user if they double click on an embedded file.



You are about to activate an embedded object that may contain viruses or be otherwise harmful to your computer. It is important to be certain that it is from a trustworthy source. Do you want to continue?

  - Microsoft WordPad does not have this security functionality. Double clicking executes the embedded file without prompting the user.
  - .wri files are therefore very dangerous, especially since embedded objects can be given an arbitrary icon and a displayed file name to socially engineer a user into double clicking. .rtf files are similarly very dangerous unless Word is installed and configured to handle .rtf files.

- Demonstration
  - .rtf with embedded potentially malicious executable code.

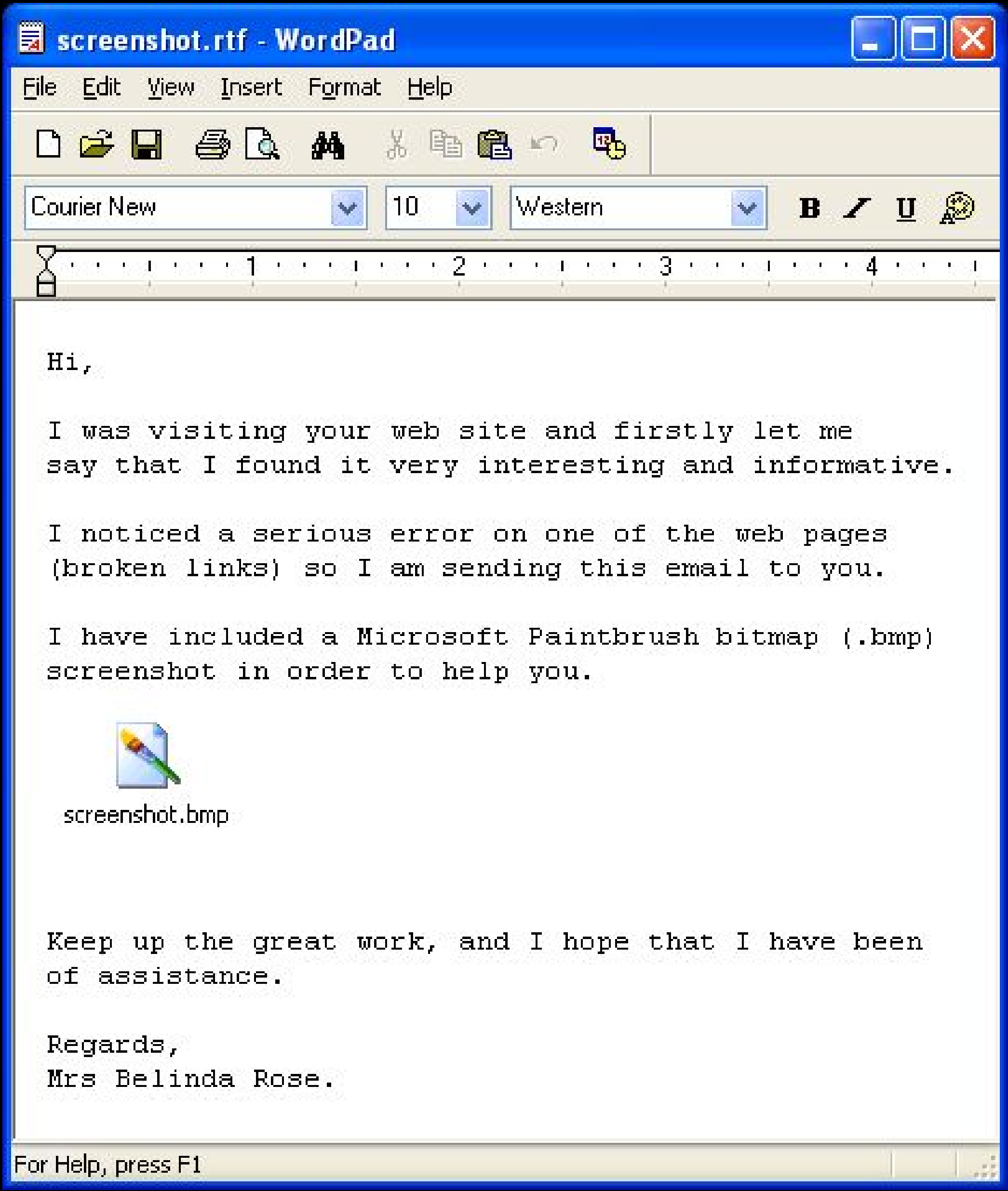

Broken links on your web site
File Edit View Tools Message Help
Reply
Reply All
Forward
Print
Delete
Previous

From: Belinda Rose
Date: Monday, January 12, 2004 8:13 AM
To: Webmaster
Subject: Broken links on your web site
Attach: screenshot.rtf (78.6 KB)

Hi,

While visiting your web site I noticed some broken links.
I have included more detailed information and a screenshot in the attached rich text file (rtf).

Regards,
Mrs Belinda Rose.



Hi,

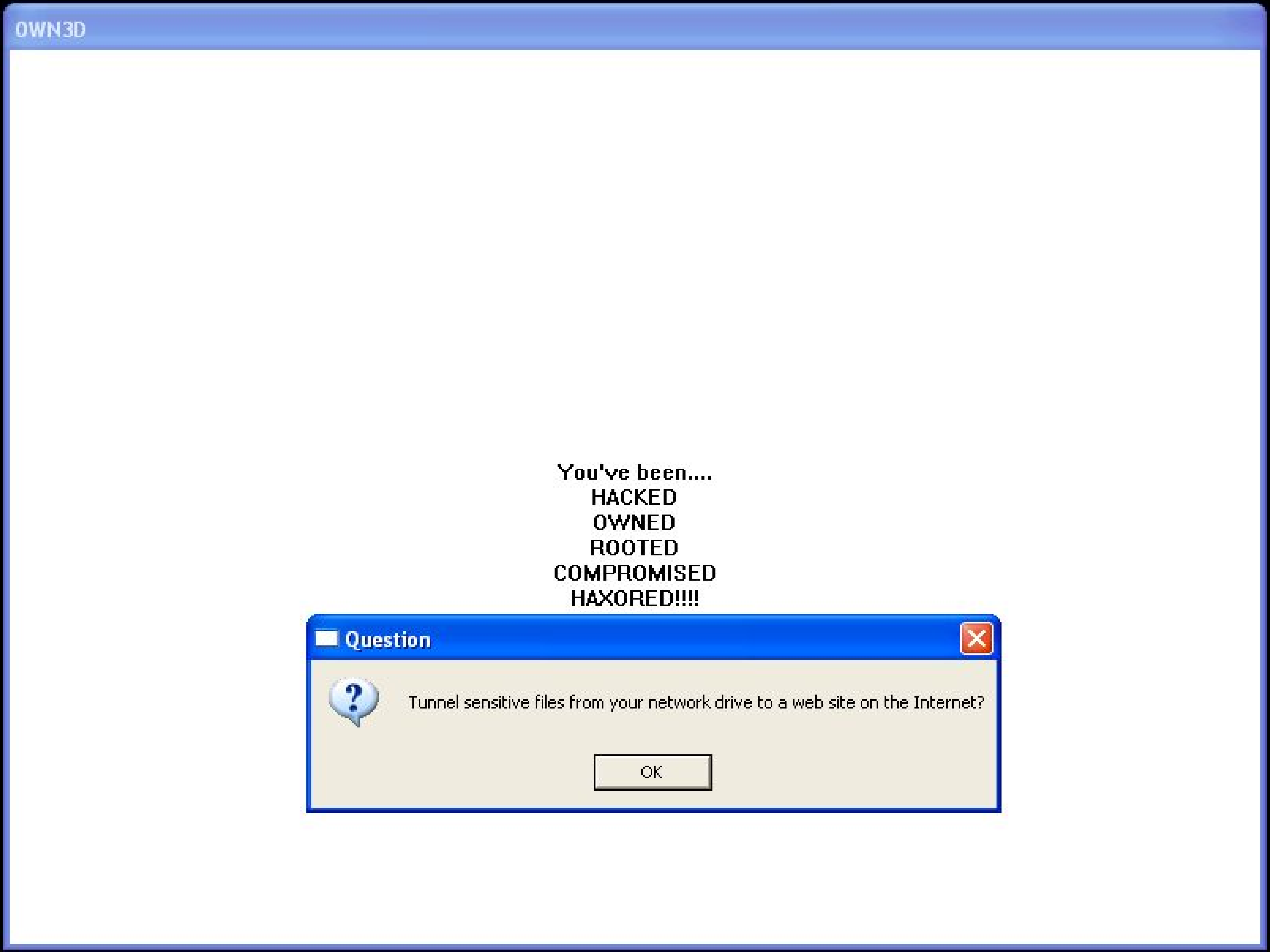
I was visiting your web site and firstly let me
say that I found it very interesting and informative.

I noticed a serious error on one of the web pages
(broken links) so I am sending this email to you.

I have included a Microsoft Paintbrush bitmap (.bmp)
screenshot in order to help you.

Keep up the great work, and I hope that I have been
of assistance.

Regards,
Mrs Belinda Rose.



You've been....
HACKED
OWNED
ROOTED
COMPROMISED
HAXORED!!!!

Tunnel sensitive files from your network drive to a web site on the Internet?

Detecting malicious content in otherwise allowed files: html

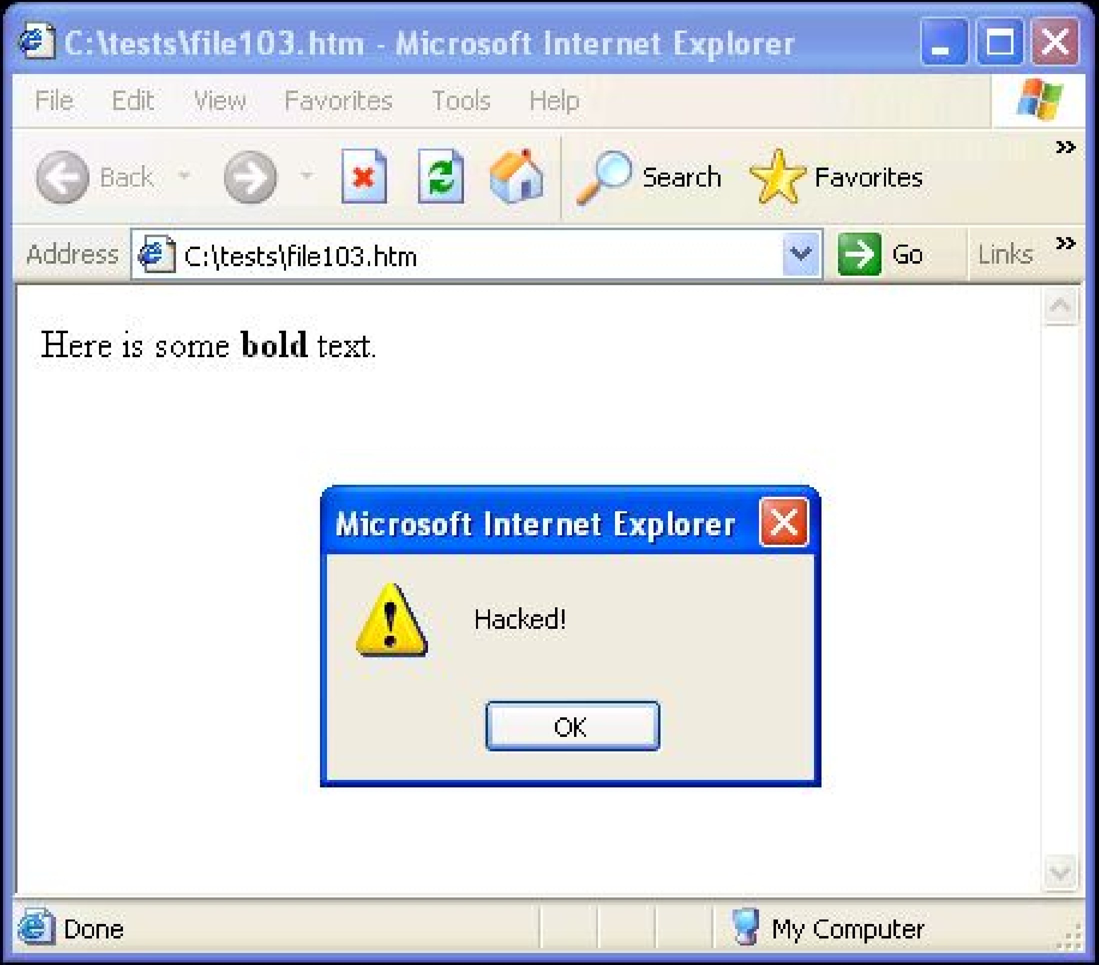
- **Does a htm/html email or file attachment contain potentially malicious active content?**
  - **HTML emails and attachments are just plain evil, mostly due to a never ending stream of vulnerabilities (and a very slow vendor response time for the associated patches) in a particular web browser that many Windows email clients use to render HTML emails. Unfortunately, blocking HTML emails outright tends to result in user outcry.**
  - **Some email clients can be configured to disallow the execution of script, but the html attachments will be opened by the default web browser.**

Detecting malicious content in otherwise allowed files: html

- Not good enough to just search for / sanitise tags such as applet/object/script since javascript can be executed without script tags.

- Demonstration
  - HTML without script tags executing javascript.

```
<html>
<body>
Here is some <b onmouseover=alert('Hacked!');>bold</b> text.
</body>
</html>
```



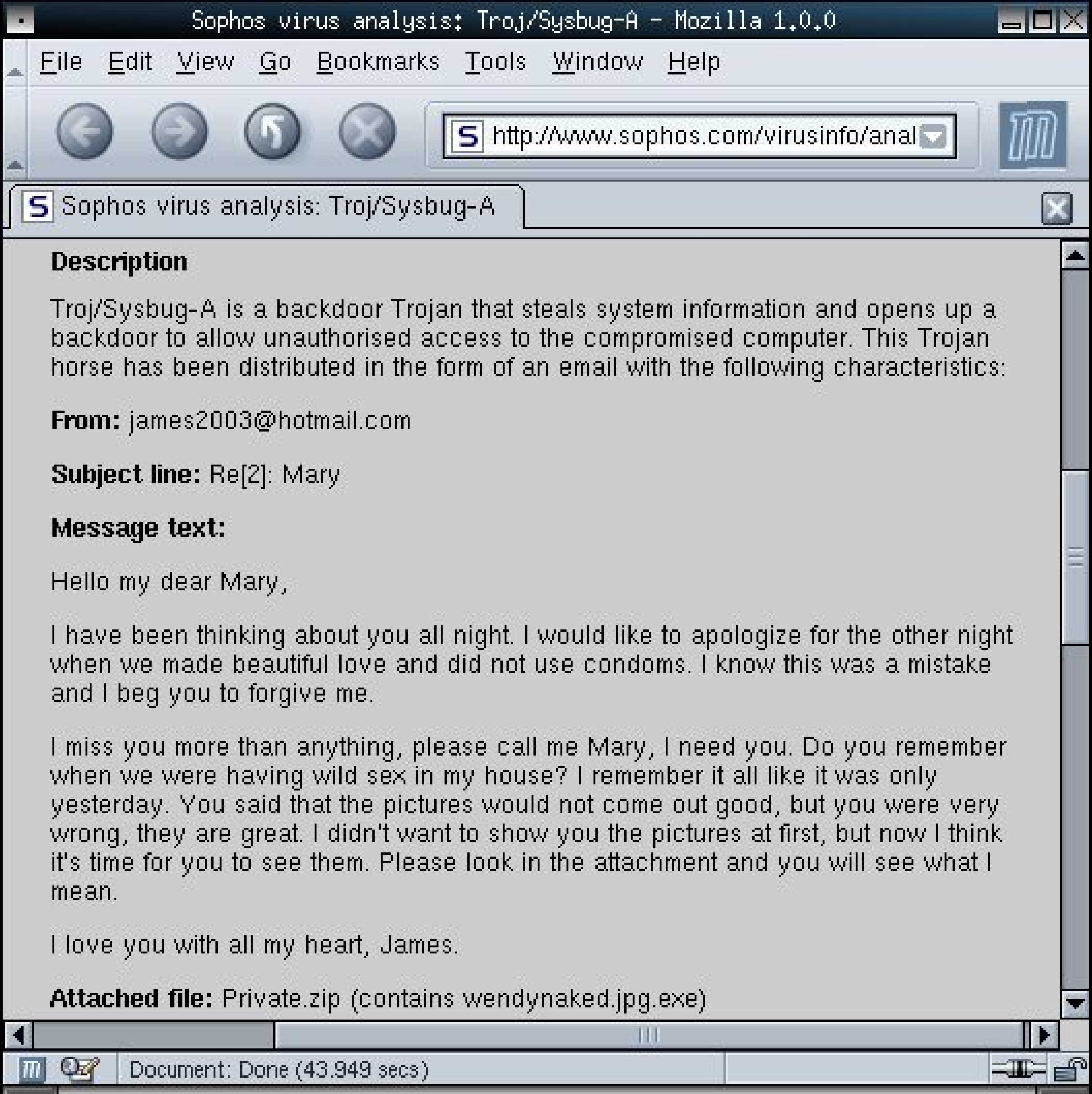
Here is some **bold** text.

Microsoft Internet Explorer

Hacked!

Detecting malicious content in otherwise allowed files: zip

- **Zip files are commonly used in business, so you typically have to allow them as an accepted file type.**
- **However, including the ability to process zip files opens up many additional attack vectors.**
- **Standard testing involves determining if the zip file only contains files of an accepted type and extension.**
  - **Several viruses/worms/trojans consist of executable content inside a zip file. This defeats email content filters that can't process zip files and only requires an additional double click by the user to run the executable content.**


Sophos virus analysis: Troj/Sysbug-A - Mozilla 1.0.0
File
Edit
View
Go
Bookmarks
Tools
Window
Help
http://www.sophos.com/virusinfo/anal
Sophos virus analysis: Troj/Sysbug-A
Document: Done (43.949 secs)

Description

Troj/Sysbug-A is a backdoor Trojan that steals system information and opens up a backdoor to allow unauthorised access to the compromised computer. This Trojan horse has been distributed in the form of an email with the following characteristics:

From: james2003@hotmail.com

Subject line: Re[2]: Mary

Message text:

Hello my dear Mary,

I have been thinking about you all night. I would like to apologize for the other night when we made beautiful love and did not use condoms. I know this was a mistake and I beg you to forgive me.

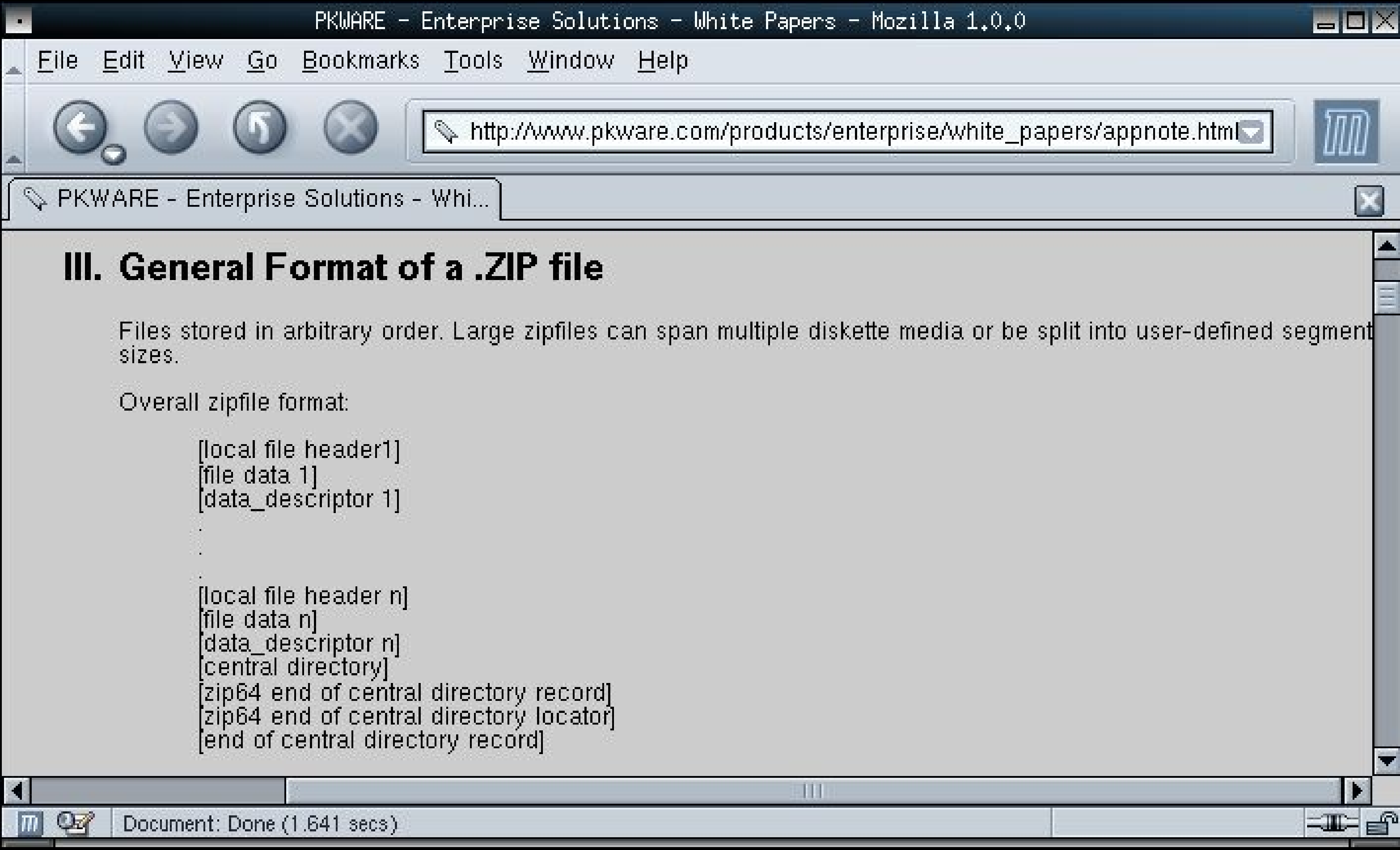
I miss you more than anything, please call me Mary, I need you. Do you remember when we were having wild sex in my house? I remember it all like it was only yesterday. You said that the pictures would not come out good, but you were very wrong, they are great. I didn't want to show you the pictures at first, but now I think it's time for you to see them. Please look in the attachment and you will see what I mean.

I love you with all my heart, James.

Attached file: Private.zip (contains wendynaked.jpg.exe)

Detecting malicious content in otherwise allowed files: zip

- **Determining if the zip file only contains files of an accepted type and extension - do not just perform a directory listing of the files inside the zip file.**
  - **Files must be extracted so that file typing can be performed.**
  - **What you see is *not* what you get. A directory listing of the files inside a zip file is not an accurate representation of the files actually in the zip file since file attributes such as file name are stored in two places for each file.**

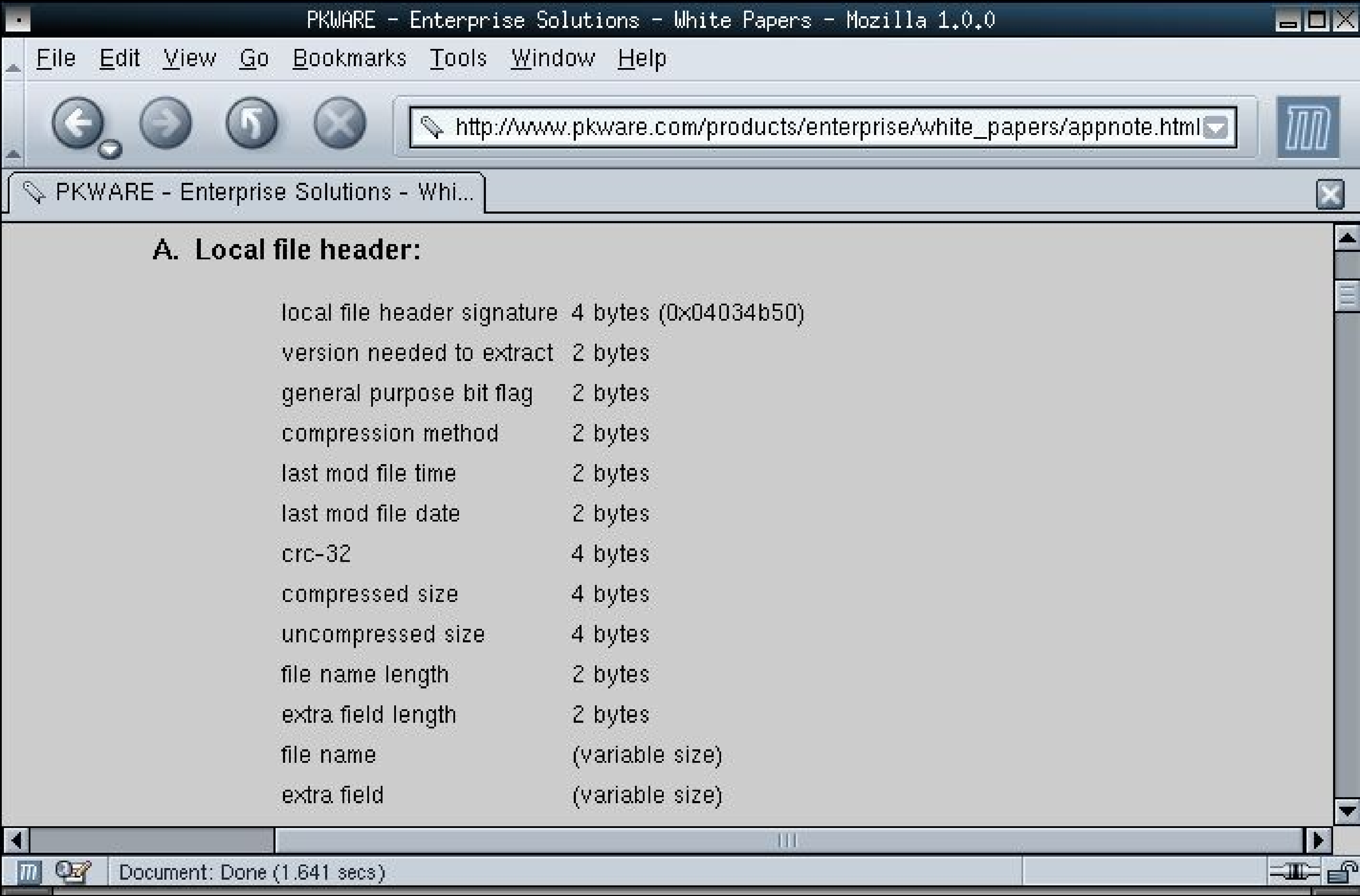

PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File Edit View Go Bookmarks Tools Window Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

III. General Format of a .ZIP file

Files stored in arbitrary order. Large zipfiles can span multiple diskette media or be split into user-defined segment sizes.

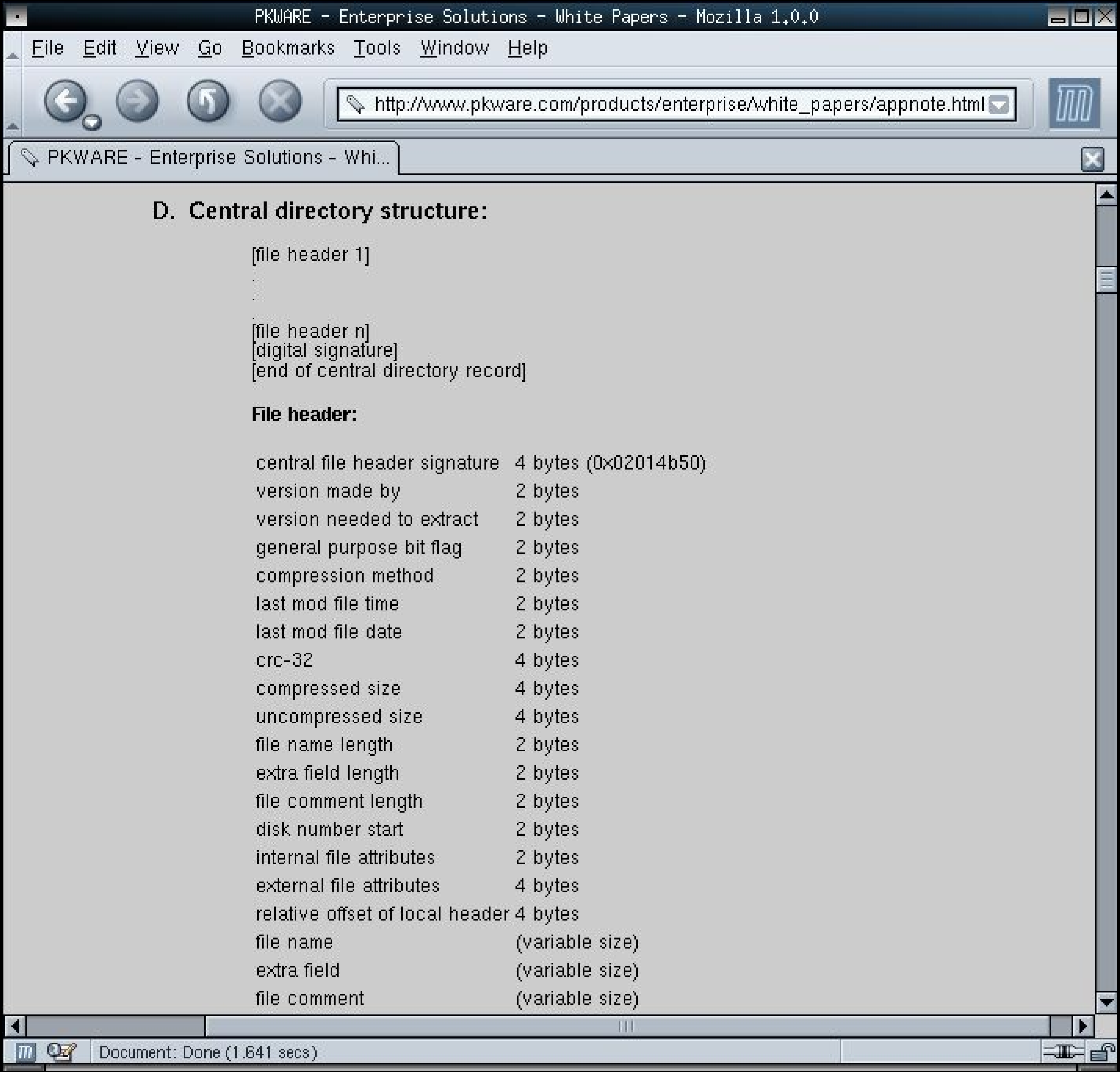
Overall zipfile format:

```
[local file header1]
[file data 1]
[data_descriptor 1]
.
.
.
[local file header n]
[file data n]
[data_descriptor n]
[central directory]
[zip64 end of central directory record]
[zip64 end of central directory locator]
[end of central directory record]
```


PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File
Edit
View
Go
Bookmarks
Tools
Window
Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

A. Local file header:

local file header signature	4 bytes (0x04034b50)
version needed to extract	2 bytes
general purpose bit flag	2 bytes
compression method	2 bytes
last mod file time	2 bytes
last mod file date	2 bytes
crc-32	4 bytes
compressed size	4 bytes
uncompressed size	4 bytes
file name length	2 bytes
extra field length	2 bytes
file name	(variable size)
extra field	(variable size)



PKWARE - Enterprise Solutions - White Papers - Mozilla 1.0.0
File
Edit
View
Go
Bookmarks
Tools
Window
Help
http://www.pkware.com/products/enterprise/white_papers/appnote.html
PKWARE - Enterprise Solutions - Whi...
Document: Done (1.641 secs)

D. Central directory structure:

[file header 1]
.
.
.
[file header n]
[digital signature]
[end of central directory record]

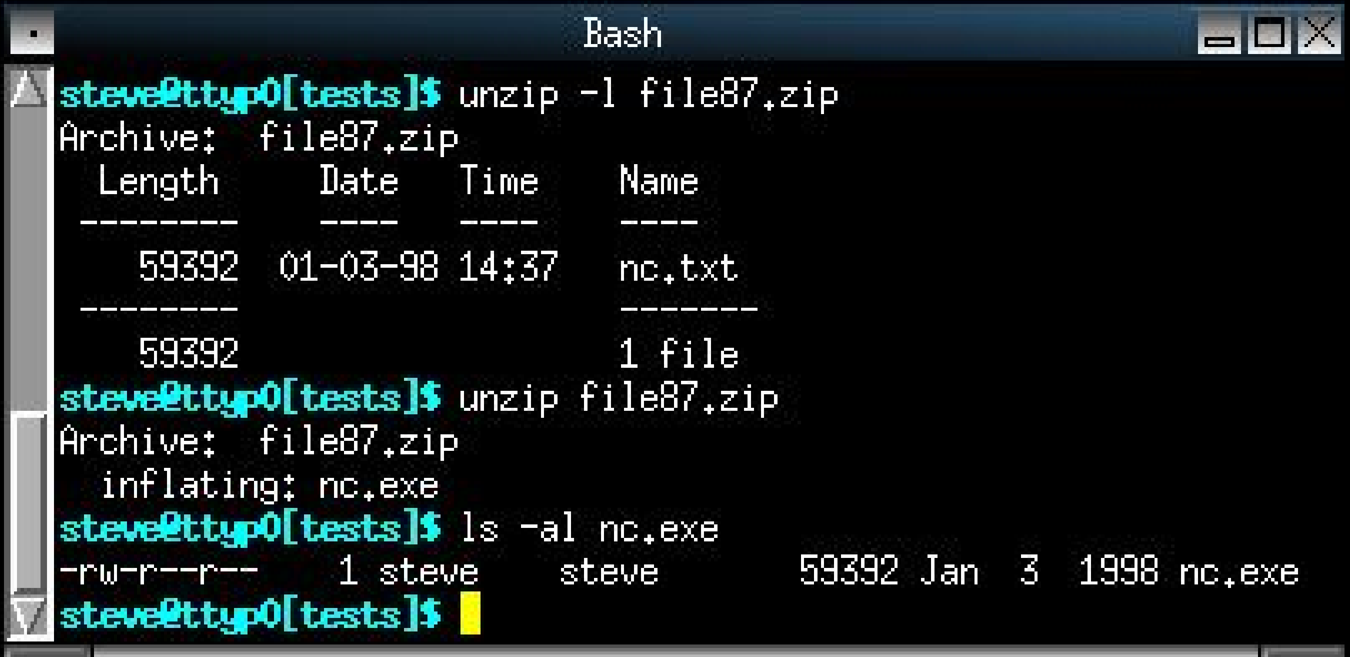
File header:

central file header signature	4 bytes (0x02014b50)
version made by	2 bytes
version needed to extract	2 bytes
general purpose bit flag	2 bytes
compression method	2 bytes
last mod file time	2 bytes
last mod file date	2 bytes
crc-32	4 bytes
compressed size	4 bytes
uncompressed size	4 bytes
file name length	2 bytes
extra field length	2 bytes
file comment length	2 bytes
disk number start	2 bytes
internal file attributes	2 bytes
external file attributes	4 bytes
relative offset of local header	4 bytes
file name	(variable size)
extra field	(variable size)
file comment	(variable size)

Australian Government
Department of Defence

Detecting malicious content in otherwise allowed files: zip

- Our email content filter checks that each file name matches its entry in the central directory record, and disallows the email attachment if this is not the case.
- Demonstration
  - Mismatching file names in a zip file.



```
steve@ttyp0[tests]$ unzip -l file87.zip
Archive:  file87.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
    59392  01-03-98 14:37   nc.txt
 --------                   -------
    59392                   1 file
steve@ttyp0[tests]$ unzip file87.zip
Archive:  file87.zip
  inflating: nc.exe
steve@ttyp0[tests]$ ls -al nc.exe
-rw-r--r--    1 steve    steve       59392 Jan  3  1998 nc.exe
steve@ttyp0[tests]$
```

Detecting malicious content in otherwise allowed files: zip

- Password protected zip files.
  - Disallowed by our email content filter.
  - Although a directory listing of the files inside a password protected zip file can be performed, this is not accurate as previously discussed.
  - The type of files inside a password protected zip file can't be determined.

Detecting malicious content in otherwise allowed files: zip

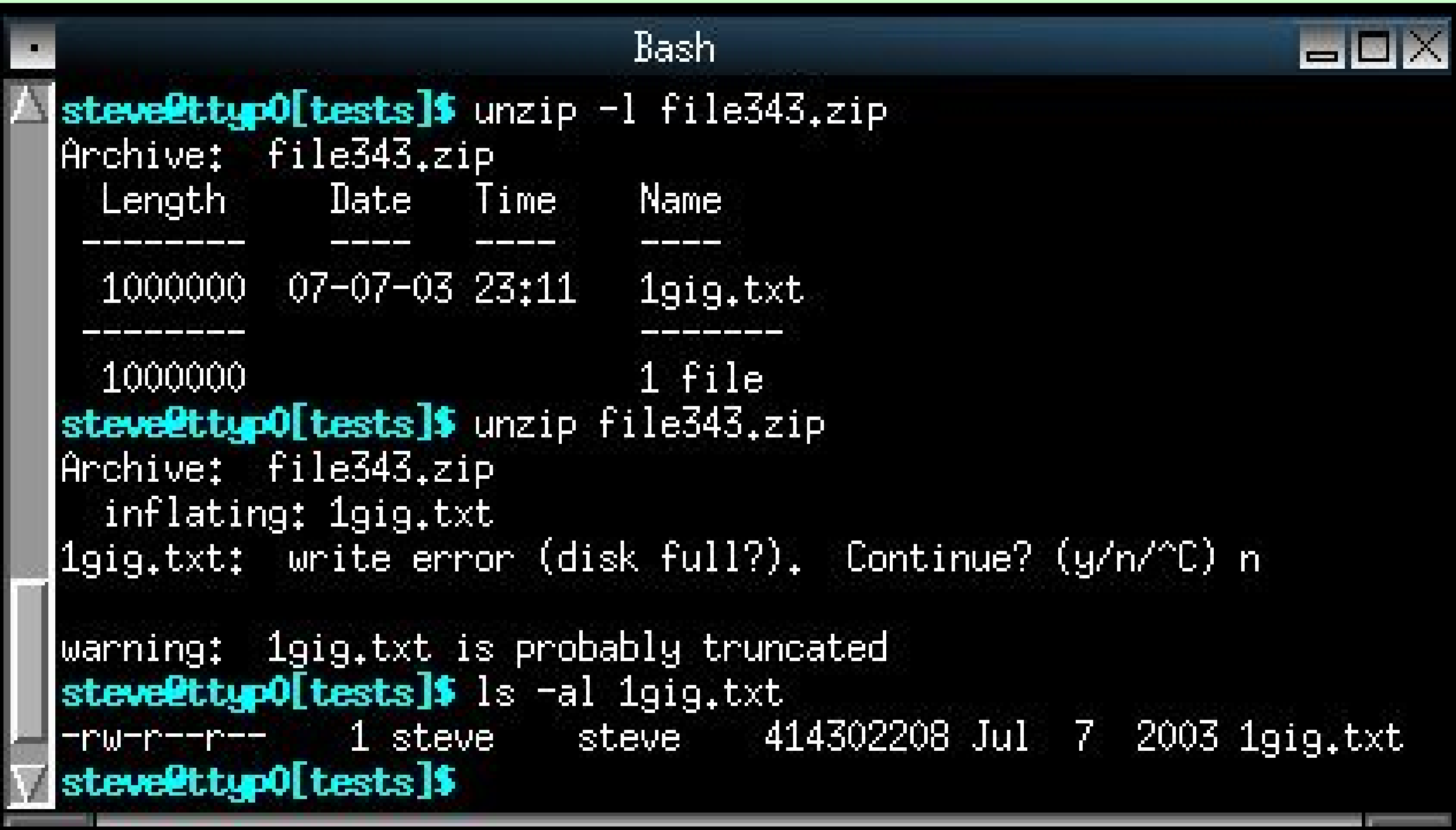
- **Zip files containing zip files.**
  - **Disallowed by our email content filter.**
  - **We have found it trivial to DoS “market leading” email content filters by using a zip file containing 16 zip files, each of which contain 16 zip files, each of which contain 16 zip files… A depth of 5 results in over one million zip files to be processed.**

Detecting malicious content in otherwise allowed files: zip

- Must extract zip file contents in order to inspect them.
  - We have found it trivial to DoS “market leading” email content filters by using a 1GB file containing spaces that compresses to 1MB.
  - Some antivirus programs suffer from the same problem.
  - Before unzipping the file, can we examine the central directory record to determine the size of the file when decompressed? Yes, but it is very unreliable.

Australian Government
Department of Defence

Detecting malicious content in otherwise allowed files: zip



```
steve@ttyp0[tests]$ unzip -l file343.zip
Archive:  file343.zip
  Length     Date   Time    Name
 --------    ----   ----    ----
  1000000  07-07-03 23:11   1gig.txt
 --------                   -------
  1000000                   1 file
steve@ttyp0[tests]$ unzip file343.zip
Archive:  file343.zip
  inflating: 1gig.txt
1gig.txt:  write error (disk full?).  Continue? (y/n/^C) n

warning:  1gig.txt is probably truncated
steve@ttyp0[tests]$ ls -al 1gig.txt
-rw-r--r--    1 steve    steve    414302208 Jul  7  2003 1gig.txt
steve@ttyp0[tests]$
```

- **Our email content filter solves this problem via the use of CPU/memory/disk quotas. If the unzip failed for example because the quota was reached (or the zip file was corrupt or password protected) then the email attachment is disallowed.**

Detecting malicious content in otherwise allowed files: zip

- Directory traversal and specifying the destination directory.
  - An attacker’s dream come true. Just send an email with a zip attachment containing malicious versions of system files, and the email content filter will rootkit itself.
  - File attachments (and files inside zip files) with names:
    - \winnt\system32\cmd.exe
    - /winnt/system32/cmd.exe
    - ..\..\..\..\..\..\..\..\..\..\..\..\winnt\system32\cmd.exe
    - ../../../../../../../../../../../..\winnt\system32\cmd.exe
    - ..\..\..\..\..\..\..\..\..\..\..\..\winnt\profiles\administrator\startm~1\programs\startup\badfile.com
    - .^./.^./.^./.^./.^./.^./.^./.^./bin/sh   (where ^ is hex 7F)
  - Our email content filter addresses this issue by extracting files into memory.

Detecting malicious content in otherwise allowed files: zip

- **Duplicates that test logic and error handling.**
  - **An email with two attachments, both called file106.zip where the first zip file contains a text file called blah.txt and the second zip file contains a malicious executable program called bad.exe.**
    - **Also vice versa (first zip has program, second zip has text file).**
    - **While unpacking the email, will the email content filter overwrite the zip file containing the executable with the zip file containing the text file, and then incorrectly forward the email to the original recipient?**
    - **Our email content filter handles this by analysing file attachments one at a time.**

Detecting malicious content in otherwise allowed files: zip

– **A zip file containing a text file called dir.txt and a malicious program called dir.txt/file268.com where dir.txt is a subdirectory.**

  - **Also test a zip file with the files in the opposite order.**
  - **A directory can’t be created if there is already a file by the same name, therefore the program is not extracted and examined by the email content filter.**
  - **Our email content filter handles this by decompressing and analysing files one at a time.**

Detecting malicious content in otherwise allowed files: zip

- **A zip file containing a text file called file271.com/file271.txt and a malicious program called file271.com**
    - **Also test a zip file with the files in the opposite order.**
    - **A file can't be created if there is already a directory by the same name, therefore the program is not extracted and examined by the email content filter.**
    - **Our email content filter handles this by decompressing and analysing files one at a time.**

Detecting malicious content in otherwise allowed files: zip

- An email containing the file attachment file323.zip
  - This zip file contains a zip file called file323.zip which contains a malicious program.
  - Will the email content filter overwrite the first file323.zip?

Detecting malicious content in otherwise allowed files: zip

– A zip file containing a malicious program called file344.txt and a text file called file344.txt

- Also test a zip file with the files in the opposite order.
- Will the email content filter overwrite the malicious file344.txt with the innocent file344.txt during the unzip phase, then perform extension and file type checks, and finally forward the email to the original recipient?

Detecting malicious content in otherwise allowed files: zip

– A zip file containing a malicious program called null.com^.txt where ^ is hex 00

- Use a hex editor to change ^ to hex 00
- Hex 00 may be interpreted as the end of string terminator.
- The email content filter may see a file called null.com^.txt while the email client sees a file called null.com

Operating System dependent problems

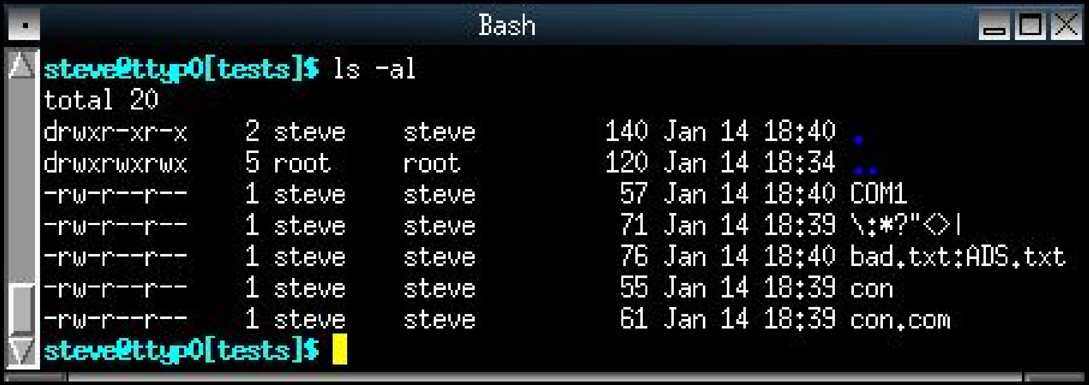
- **Attackers want the email content filter unzip utility to behave differently to the unzip utility on the email recipient's workstation e.g. Windows vs Linux**
  - **A network of Linux workstations should not be protected by an email content filter running on Windows.**
  - **A network of Windows workstations can be protected by an email content filter running on Linux or Windows. If it is running on Windows it will have limitations, but these limitations should be negated by the workstations suffering from the same limitations.**

Operating System dependent problems

- Consider a Windows based email content filter that decodes (and unzips if necessary) file attachments into a directory, and then checks the files in that directory.
  - A variety of files may not be created successfully on the Windows file system, and therefore may not be extracted and examined by the email content filter.
  - The Linux file system does not have such limitations, so a user on a Linux workstation would be able to extract and execute the file.
    - File extensions are of less use in Linux.
    - Users would typically have to go out of their way to assign executable permissions to the file, and may be protected by administrator configured Linux security features such as a non-executable home directory.

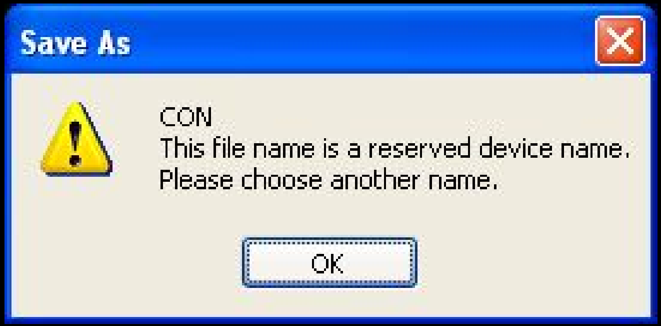
Operating System dependent problems

- **Example files:**
  - **The file con.txt**
  - **A zip file containing the file COM1\malware.txt (i.e. the file malware.txt in the subdirectory COM1)**
  - **Files containing characters reserved by Microsoft \ / : * ? " < > |**

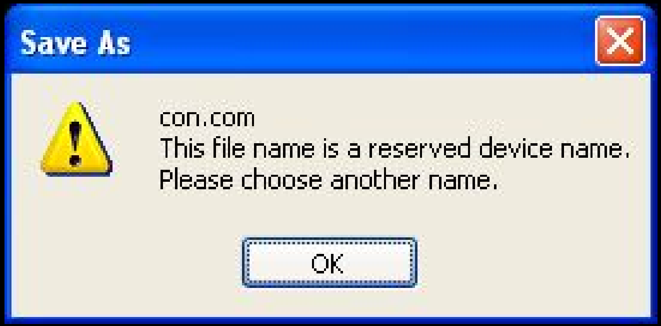


```
steve@ttyp0[tests]$ ls -al
total 20
drwxr-xr-x    2 steve    steve         140 Jan 14 18:40 .
drwxrwxrwx    5 root     root          120 Jan 14 18:34 ..
-rw-r--r--    1 steve    steve          57 Jan 14 18:40 COM1
-rw-r--r--    1 steve    steve          71 Jan 14 18:39 \:*?"<>|
-rw-r--r--    1 steve    steve          76 Jan 14 18:40 bad.txt:ADS.txt
-rw-r--r--    1 steve    steve          55 Jan 14 18:39 con
-rw-r--r--    1 steve    steve          61 Jan 14 18:39 con.com
steve@ttyp0[tests]$
```

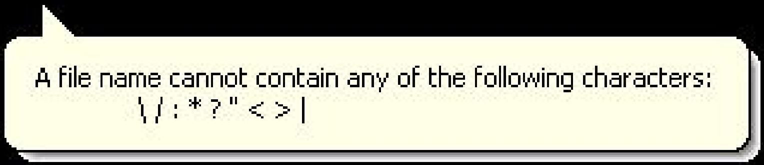
versus



CON
This file name is a reserved device name.
Please choose another name.



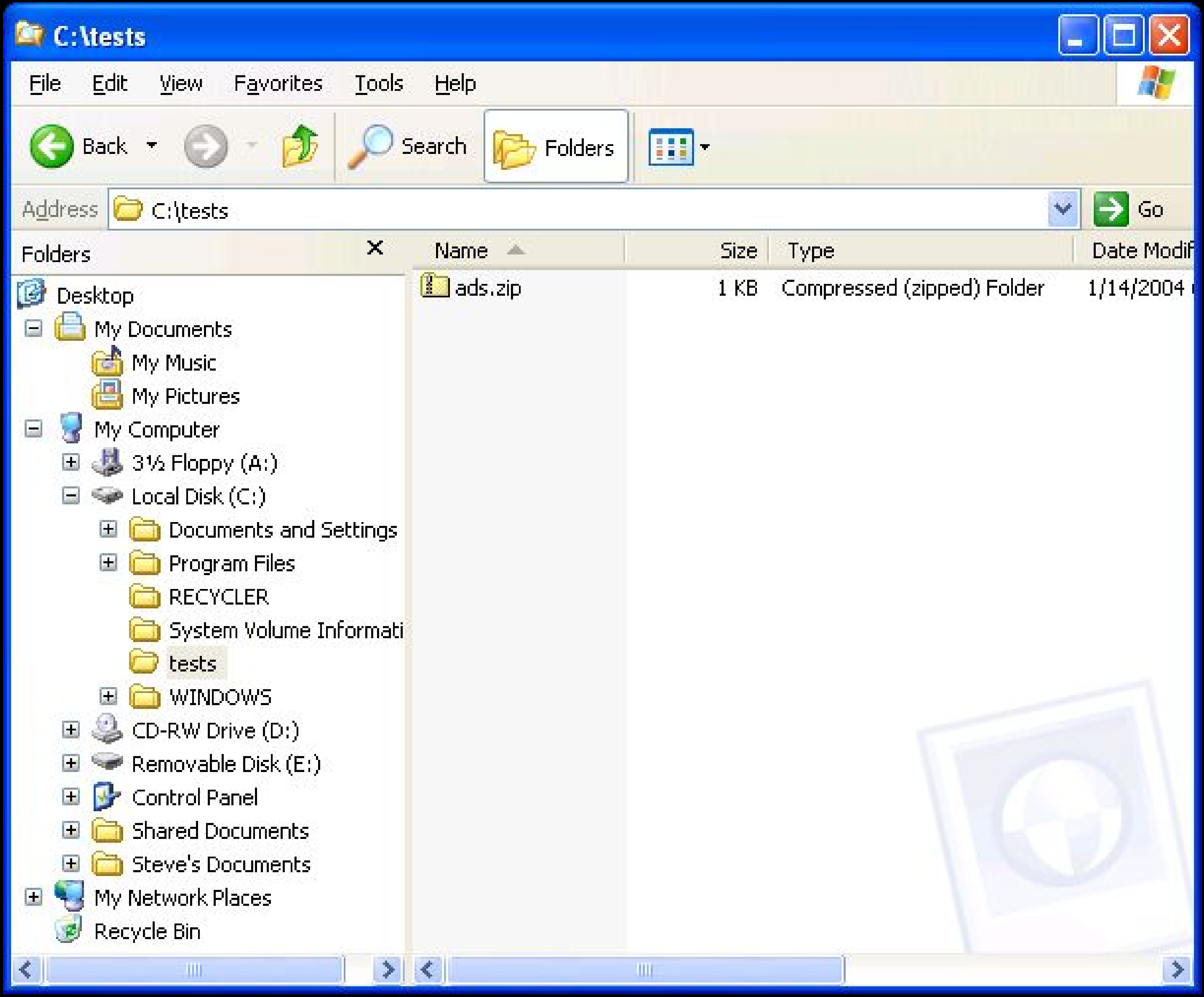
con.com
This file name is a reserved device name.
Please choose another name.



A file name cannot contain any of the following characters:
\ / : * ? " < > |

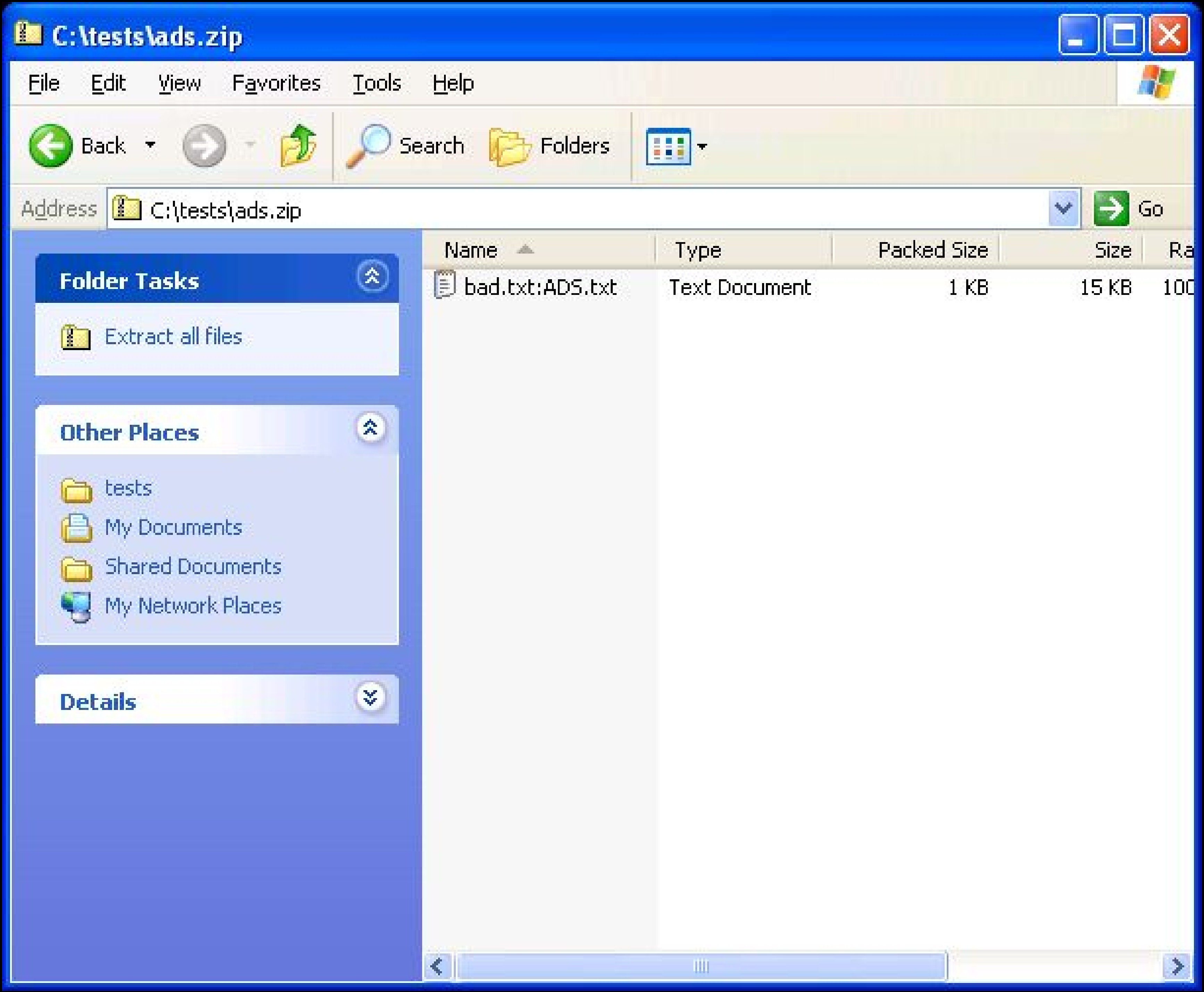
Operating System dependent problems

- A zip file containing the file bad.txt:ADS.txt
  - Will create a 0 byte file called bad.txt, with the contents of the file hidden in an alternate data stream and inaccessible to the email content filter unless it can specifically handle ADS.



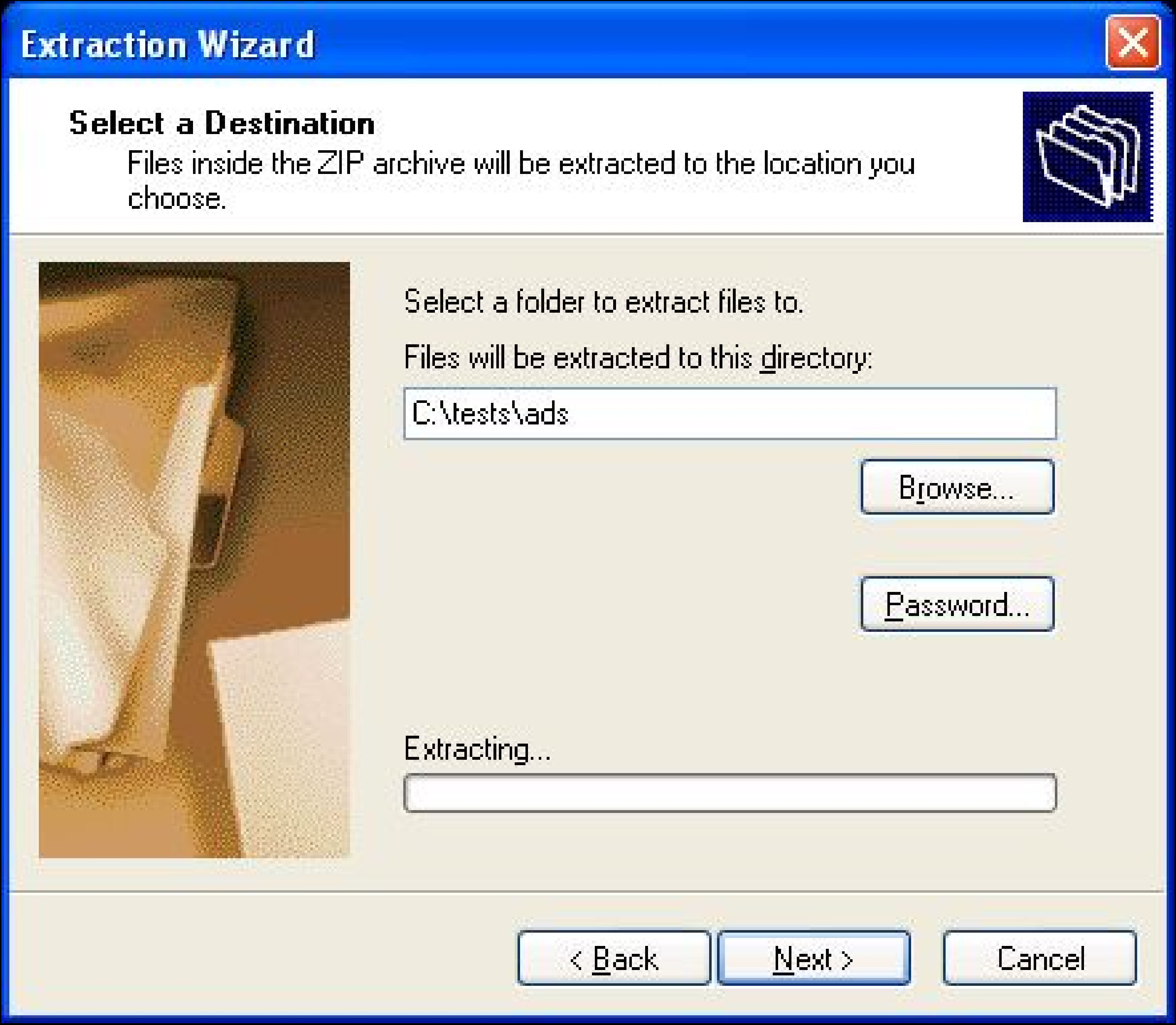
Address: C:\tests

Name	Size	Type	Date Modif
ads.zip	1 KB	Compressed (zipped) Folder	1/14/2004



Address C:\tests\ads.zip

Name	Type	Packed Size	Size	Ra
bad.txt:ADS.txt	Text Document	1 KB	15 KB	100



Select a Destination

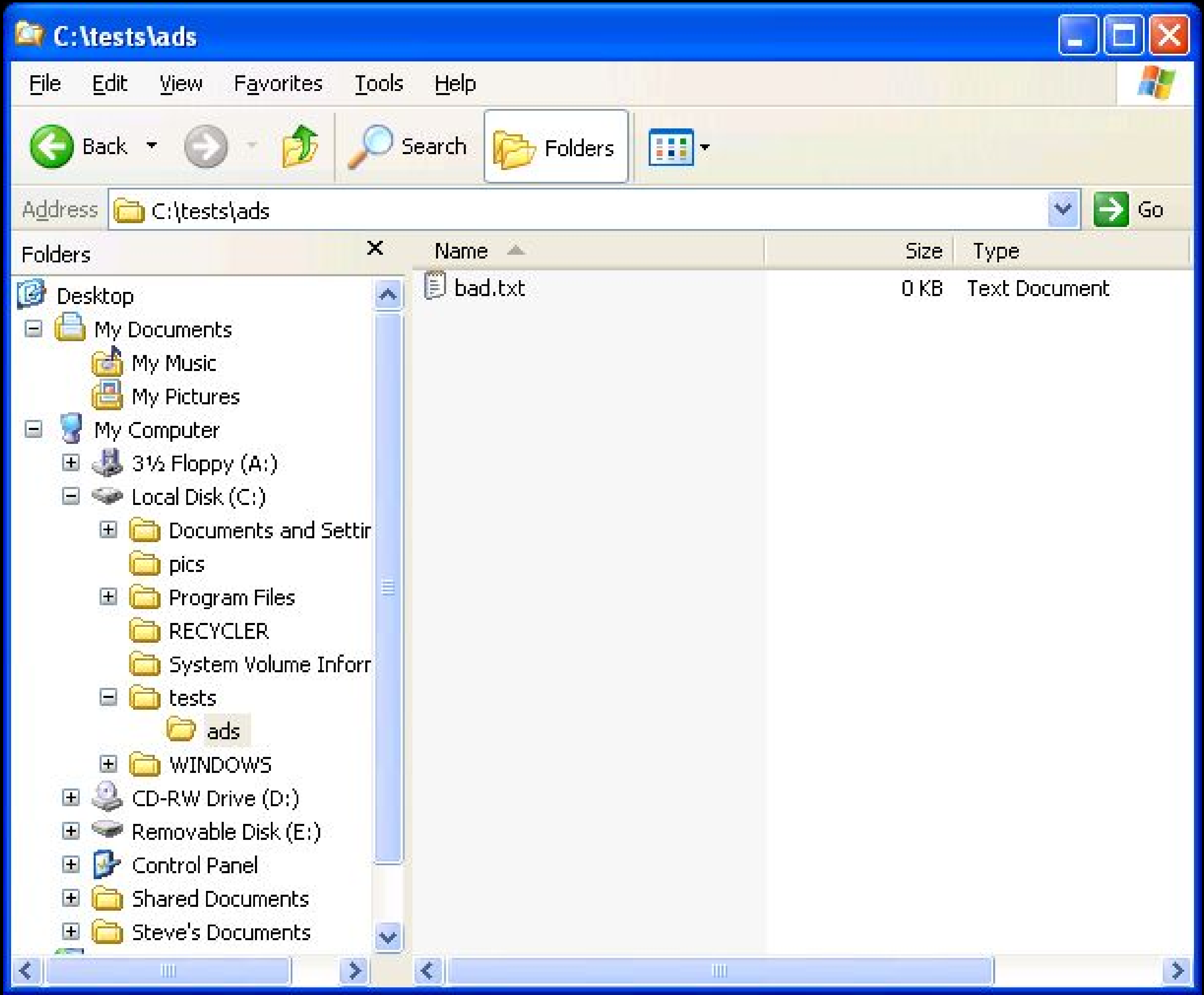
Files inside the ZIP archive will be extracted to the location you choose.

Select a folder to extract files to.

Files will be extracted to this directory:

C:\tests\ads

Extracting...



Address: C:\tests\ads

Name	Size	Type
bad.txt	0 KB	Text Document

Operating System dependent problems

– An email content filter running on Linux that improperly calls unzip may be vulnerable to file attachments with the following names:

- file263.zip;touch /tmp/test263;file263.zip
- file264.zip||touch /tmp/test264||file264.zip
- file265.zip&&touch /tmp/test265&&file265.zip
- -o -q -d /tmp file267.zip

Email server tests

- If the email content filter is also running email server software, test for typical email server vulnerabilities.
  - Email relaying where an external attacker causes the email server to send an email to an external recipient.
    - Standard email relaying. Can attempt to bypass anti-relay measures using a FROM address of root@localhost or root@[127.0.0.1]
    - Specify the target email address as the FROM, and send the email to a non-existent user so that the bounce message carries the spam or malicious attachment.

Email server tests

- Delivery Status Notification (DSN) e.g.
  MAIL FROM: thirdparty@somewhere.org RET=FULL
  RCPT TO: a_legitimate_user@legitimate_domain_name
  NOTIFY=SUCCESS,FAILURE,DELAY
  data
  Hi,
  Please double click on the attachment.....

- Determine the number of listening email connections and their timeout value.
- Send an email to someemaillist@targetdomain.com with a delivery and read receipt - some email server implementations will disclose who is on the mailing list.

Email server tests

- Use EXPN and VRFY verbs to determine if accounts can be enumerated.
  - joe
  - root
  - postmaster
  - skdnjfsdsdjbgsdgj
- Specify the following RCPT TO: values to determine if accounts can be enumerated.
  - joe
  - root
  - postmaster
  - skdnjfsdsdjbgsdgj

Conclusions

- Threats are viruses and attackers sending malicious emails to uneducated users.
  - The educated users are too busy trying to beat the system.
- Allowing "safe" file extensions is preferable to blocking all of the possibly malicious extensions.
- File typing complements file extension checks.
- "Safe" file types and file extensions may require further checks.

Conclusions

- A dedicated user can beat the system. Facilitated by:
  - File formats that allow the inclusion of code (bmp) or extra arbitrary data that is ignored (zip).
  - Windows based email content filters that inherit the limitations of the OS.
  - Utilities such as uuencode.com and uudecode.com.
  - The Windows .com file format - there isn't one!

Questions?