Global Information Assurance Certification Paper

SANS GIAC Program
Incident Handling and Hacker Exploits
Practical Assignment, Version 1.5

Chris Krumme
18 April 2001

Internet Explorer MIME Header Exploit

Introduction:

With its widespread use and powerful add-ons, Microsoft's Internet Explorer (hereafter referred to as "IE") is an obvious target for new and creative exploits. IE's family of frighteningly powerful components tie it closely to the Windows operating system, making it a risk for a variety of potentially dangerous attacks. The reason IE is so vulnerable is an excellent example of what happens when you see system security as secondary to ease of use. In an attempt to make Windows and IE friendlier from the end user's perspective, certain functions are carried out automatically with the intention of simplifying the user experience. If one can figure out how to trick these automated features into allowing actions other than what they were designed for, the system becomes a security risk. When you combine the number of available add-ons, plug-ins, application integration capabilities with a few curious and creative minds, you create a wonderful digital petrie dish to grow exploits in.

High-bandwidth Internet connections, the end-user's desire for more integration between applications, and the goal of simplifying the user experience has driven developers to create more powerful, capable software at an astounding pace. The unfortunate side effect is that since marketing and the almighty dollar come first, we rush new products on to the shelves in the hopes of turning a quick a profit. This cycle constantly introduces more entropy to our systems, and leaves us forever trying to catch up with the mistakes made during development. Many of the serious flaws are patched quickly after an exploit is discovered, but there is always that nagging question about how new the exploit really is, and how many more there are like it that we *don't* know about.

I chose to examine a MIME header exploit in IE both because it was made public recently (March 29, 2001), and because it is one of the nastier HTML-based exploits that I have come across. This vulnerability would allow a malicious e-mail or web page to literally do *anything* on your local computer, provided the current user has permissions to run the exploit on the machine. This means that an attacker could potentially add/delete data, run applications, and issue commands automatically and without the user's knowledge simply by embedding the exploit within a web page or HTML e-mail.

Exploit Details:

This exploit was discovered by Juan Carlos Garcia Cuartango, and made public by Microsoft on March 29, 2001. This exploit involves tricking IE into running arbitrary commands, scripts, or executables, provided the permissions allow them on the target machine. A patch was released simultaneously with the announcement, but oddly enough has not been part of a critical update (at least at the time of this writing).

This exploit is documented at Microsoft's web site as:

- Microsoft Security Bulletin: (MS01-020)
- Title: Incorrect MIME Header Can Cause IE to Execute E-Mail Attachment
- Originally posted: March 29, 2001
- Summary
  Who should read this bulletin: Customers using Microsoft® Internet Explorer
  Impact of vulnerability: Run code of attacker's choice.
  Recommendation: Customers using IE should install the patch immediately.
  Affected Software:
    Microsoft Internet Explorer 5.01
    Microsoft Internet Explorer 5.5
  Note: Internet Explorer 5.01 Service Pack 2 is not affected by this vulnerability

And at BugTraq as:

- BugTraq ID: 2524
- Class: Failure to Handle Exceptional Conditions
- CVE: CAN-2001-0154
- Remote: Yes
- Local: No
- Published: March 29, 2001
- Updated: April 03, 2001
- **Vulnerable**:
- Microsoft Internet Explorer 5.5
- Microsoft Windows 95/98/NT 4.0/2000
- Microsoft Internet Explorer 5.01
  - Microsoft Windows 95/98/NT 4.0/2000
- **Not vulnerable**:
- Microsoft Internet Explorer 5.0.1SP2
  - Microsoft Windows 95/98/NT 4.0/2000

Protocol Description:

The short, technical answer to the protocol that is used for this exploit is Hypertext Transfer Protocol (HTTP). The payload is contained within an HTML-

based e-mail file. These files, although parsed by the computer, are in ASCII text and are readable by humans.

This IE MIME header exploit is taken advantage of specifically via EML files. These files are MIME multipart files that IE is called upon to parse when encountered. In order to fall victim to this vulnerability, a user must open an e-mail message or visit a web site that has the malicious code embedded within it. That said, there could be level of social engineering involved in tricking a target into falling for this. It may or may not be difficult to accomplish, but the fact that you almost have to be targeted for the exploit certainly shouldn't give you a false sense of security. Someone trying really hard to get an e-mail to you or point you to a web page that you're not familiar with *should* throw a red flag in the first place, but the end result of something like this in the hands of a cracker with enough access to a high-volume web site could be disastrous. Most corporate users shouldn't have to worry as much because, in most cases, companies won't give full rights on a machine to the average employee. But those who do have enough rights on a machine, and home users who don't have user accounts on their machines at all are at risk for an exploit like this.

Later on I will discuss several possible scenarios that suggest ways that this exploit might be used. Examples are not difficult to come up with since both the human and network protocols are so common and accessible.

Description of Variants:

There are less damaging variations surfacing that revolve around this particular exploit. Two possible exploits have been proposed, one minor and one medium risk, if they are confirmed. Both of these involve prompting the user to take an action, albeit poorly, so they both boil down to how educated the person at the keyboard is.

The more minor of the two exploits was discovered and posted to BugTraq by Jesus Lopez de Aguileta, and involves changing the file name of the part to be executed. In his example, he changed the extension of the file from EXE to EXXE (he was demonstrating bypassing mail filters searching for potentially harmful content). The resulting message box referred to the content as a file, rather than a program, which he speculated could trick a user into running an executable when they thought they were opening another less risky type of file.

The more serious possible exploit came as a direct result of the patch, and was discovered by the original contributor who reported the flaw to Microsoft, Juan Carlos Garcia Cuartango. On his patched system, he was able to disguise an executable file with a falsified download name. In other words, he demonstrated how to fool the user into thinking that they were downloading or running a file of one type when it was actually another. Microsoft does not consider this or the previously mentioned issue to be a vulnerability because the user is prompted, and therefore involved in the decision on what to do with the file. Obviously, it is quite a bold statement to assume that all users know everything they should know to stay out of trouble, but that is the mentality

behind this particular family of products. It is also one of many contributing factors to the security problems within.

In a more general sense, there are too many IE-related exploits to even count; usually involving unauthorized access of data on the client machine. Even worse, client-side hacking appears to be gaining popularity in the last year or so. You can imagine the implications of anyone being able to read information from the browser client pointed at their site. Stored passwords, account numbers, "inline AutoComplete" data, and cookies could be stolen and used to gain access to accounts, or learn personal information that could be used in another attack.

Components such as ActiveX, Java, JavaScript, and multimedia capabilities combined with the powerful macro capabilities found in many Windows applications create an environment that, if certain conditions are met, have the capability to give anyone unrestricted access to a victim machine. We have seen a number of exploits that rely upon IE's powerful capabilities being loosely held together at the seams. As such, a wide variety of variants have been spawned that range from the annoying to the full-blown destructive. JavaScript can be used to pop up those annoying extra windows when you enter or leave a web site. This trick can even be used to create a denial of service condition on a web client machine. Worms have been invented that take advantage of the power of ActiveX and can propagate themselves across the Internet without any interaction from the user at all. The ones that spread particularly fast, such as the ILOVEYOU worm, are not only annoying, but can have devastating effects on the amount of traffic directed at or leaving ill-prepared networks. Denial of service conditions can result as overstressed mail servers attempt to process the sudden mass of mail activity. But, I digress.

The point is that with all of this interaction and power, there are bound to be ways to exploit it. There are also bound be people who, for whatever reason, go out of their way to find these exploits. IE doesn't seem to become less of a security risk with age, quite the opposite, so it is wise to expect more variants like this one, and more IE-related exploits in the future.

How the exploit works:

In the most basic sense, this exploit works as a result of modular code designed to work with other modular code in order to perform operations that are intended to enhance the usability of the software. In many cases this is partially accomplished by the software handling specific types of content automatically. In order to make some of these operations possible, potentially exploitable commands and function calls are made available to the calling application.

This exploit takes advantage of how IE handles Multipurpose Internet Mail Extensions (MIME) headers. Specifically, one that it does not recognize as a common type. Generally speaking, a MIME content type is defined within the HTML mail file that is of an uncommon type/subtype combination, IE mistakenly processes the defined binary (which may also be text) by default without prompting the user.

Under most circumstances mail clients such as Outlook and Outlook Express deal with all processes associated with handling e-mail. Sending,

receiving, and displaying are all done by the mail client rather than IE. However, there is one exception. When the e-mail is an HTML file, which is very common, IE is called upon to “render” the content on behalf of the mail client. This is the case because there could be one or more types of data files within the mail, and IE knows what action to take on most of them - the key word being "most". If it is audio, IE calls your audio player to run the binary, if video is specified, your media player is invoked to play the file, and so on. In cases where an actual executable file is involved, the user is usually prompted with whether or not to open the potentially dangerous attachment, save it to disk, or cancel out. But by specifying an uncommon MIME type in the header the browser apparently doesn't know what else to do, so it defaults to rendering the content automatically. At this point the file is opened and run without question, or the user's knowledge.

In the examples that were provided to the public, there was an executable, a batch file, and a Visual Basic script. I would speculate that these are used for demonstration purposes because almost any system will be able to run these file types. But I would also speculate that it could be any type of file as long as IE is duped into running the file, and the operating system can associate the file with an application installed on the local machine.

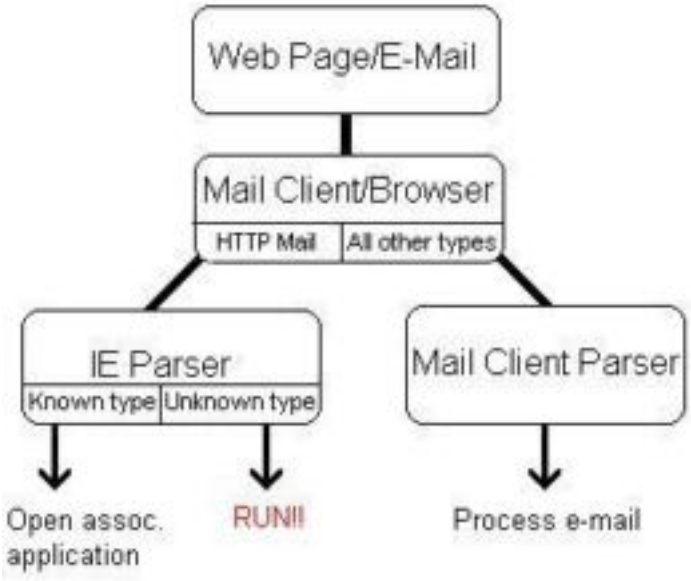
Perhaps one of the most notable things about any of these HTML-based variants is the fact that by default, these attacks subvert just about any firewall being used today. This is because most firewalls are based on packet-filtering technology. Inbound and outbound packets are inspected at the firewall and either accepted, rejected, or dropped depending on the firewall rules. With e-mail and other HTML-based exploits, the firewall sees these packets as being Internet-based and typically allows them to pass. Unless otherwise specified, this is always true for web pages that a user is accessing or e-mail that has been requested from inside the firewall. This being the case, a certain degree of social engineering could be involved in making the exploit successful. Users are required to open an attachment or visit a malicious web page before any damage can actually occur. This leaves it up to the user’s best judgment, which isn’t the optimal situation. Until stateful inspection firewalls, which are able to police the payload within the packets, become commonplace, malicious content can easily enter by masquerading as, or maybe even riding on top of standard web traffic.

The idea behind this exploit goes even deeper than the web browser and it's associated components. Just like Windows assumes that the user wants automation, it also assumes that the user really does know best. We also make a fundamental assumption that all of the shared libraries (DLLs in Windows) are solidly written and work as advertised in any configuration. With all of the possible combinations of modules, plug-ins, and components that you can get from so many individual vendors, this is a dangerous concept to rely upon. In reality, it would be close to impossible to test every possible component in every possible combination and configuration, so we're left trusting that they do work as intended – no more, no less, and without bugs. I doubt that this gives many people the warm/fuzzy feeling. The obvious downside to this assumption, aside from such a state being unattainable (at least for any useful length of time) is that

coders are human and you are guaranteed that there *will* be poor coding, compatibility issues, unforseen holes, and other problems. Some of these problems are bound to be exploitable.

Diagram:

This exploit is very straightforward. All that is required is a vulnerable version of IE with File Downloads at its default setting, a user with enough permissions on the machine to run the exploit, and the user's attention. When IE comes across an EML header embedded within an e-mail or web page, the MIME content type(s) are parsed and the browser decides what to do with the binary part of the file. If an uncommon type is passed to the browser, and the browser is not patched, it will mistakenly execute the attached binary, command, or script automatically. A simple diagram would look like this:



How to use the exploit:

Since this exploit is HTML-based, there are a number of ways that it could be used. As stated in the last section, all that is required is the desire, a target with sufficient privileges to run the exploit, and an un-patched version of IE is on the other end of the connection. Following are several examples of how this exploit could be used in the wild.

Example 1: Our hacker, Alice, who also happens to be the former head of IT at ABC.com, decides that she doesn't agree that she should have been let go for whatever reason. Before she leaves the building for the last time, she creates a false account and/or other backdoor(s) to the system so that she can enter anonymously at a later date, and wipes the logs of such activity. Enough time passes and she decides that it's pay back time for ABC.com. Alice knows that ABC.com pays little attention to the state of the system as long as it is running, so at least one of her backdoors is probably still in place. She logs in, gains root

(easy since she knows the system so well, and has the latest version of Knark), and adds an uncommon MIME type in the URL of the company's main web page that formats the hard drive of anyone unlucky enough to visit the site while the exploit is in place. Drastic. Not subtle. Very scary.

Example 2: Let's take a look at this exploit from a malicious spammers perspective. Our evil spammer (I know, they're all evil) Bob, decides that he's tired of watching his computer send out thousands of e-mails a day without any responses. He's wondering if his spam is being automatically blocked in many places because of the way he's constructing it, or if nobody is bothering to pay attention to the garbage that he peddles. Time to try an experiment. Bob then adds the uncommon MIME type in the header of his spam du jour, with an executable attached that runs a small program which sends him a reply from the machine that the message was opened on... along with the entire address book on that host. What a nifty way to find out how many of the mails made it, and to get direct, active e-mail addresses. Since Outlook and Outlook Express open the mail before you do, the user doesn't even have to read the mail for the executable to run. As long as it is highlighted long enough for IE to take over and dutifully process the strange MIME type that it has encountered, the exploit will run.

Example 3: The previous examples are pretty dramatic, how about something a bit more subtle? Let's say Alice and Bob are working for competing companies, but go to the same underwater basket weaving class once a week. Bob's company happens to be doing really well pushing a product that Alice's company hasn't quite yet perfected. Curious as to how Bob's company solved the problem that has been holding her company back, and knowing that Bob must somehow be involved in the success of the product, she and an evil exec decide that her days as a corporate spy are to begin. Having more loyalty to her company than she does to Bob, she sends Bob an e-mail with a hello and maybe some humor attached. But also embedded in the e-mail is our uncommon MIME header exploit with an executable that will FTP a file from a mysterious location and install it. Bob may or may not think that it's strange to receive e-mail from Alice, but since he knows who she is, he doesn't think much about it. So he sends her a reply, gets a chuckle from the humor, and a sniffer.

The most frightening part of this particular exploit is how easy it is to use and the widespread use of it's medium. There are enough IE users out there that no matter how much the security community yells at the world to patch its systems, anyone with an effective mechanism for propagating a tainted e-mail or web page could get to enough vulnerable systems to do some very serious damage.

Signature of the attack:

Due to the fact that this attack is embedded within HTML content, there is no attack signature attached to the exploit itself. To the operating system the attack would appear as if a user was running a program or issuing a command directly to the system. To the network it looks like standard web traffic. This

means that the best way to catch this type of attack is to provide IE with a new option for handling this type of situation, and to ensure that in the future developers remember to add contingency plans for undefined situations (a.k.a. default!) in the code.

However, the results of such an attack may leave any number of signatures that may be more obvious. Depending on the level of hostility of the attacker, any number of things may or may not be noticed. Attackers could covertly install a trojan, sniffer, or other covert surveillance program. They could mail password files, access control lists, or other sensitive information to themselves, they could render your system instable, or even go so far as to steal the contents of the drive before formatting it to cover their tracks. The list of possible things that could happen as a result of this exploit under prime conditions is limitless.

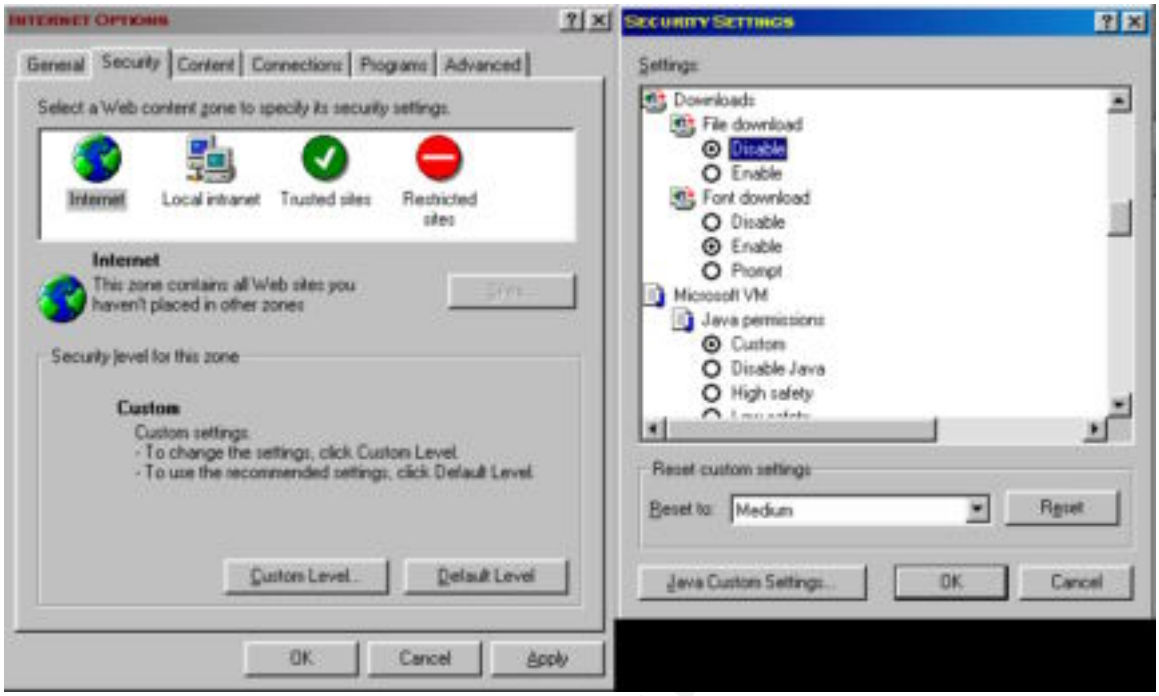
Obvious system activity would likely be noticed rather fast, but if you do actually witness something strange it may be too late to do anything about it. Of course, attacks of this nature would probably not be the method of choice for someone who doesn't want a lot of attention.

More stealthy activity *may* have a chance of getting caught by your local anti-virus software, if such a thing exists on the local system and is current. I emphasize “may” in the last sentence because my anti-virus software, updated twice since this exploit was made public, did not detect anything wrong with the executable version of this attack. You can’t always fault the anti-virus software vendors because you don't know what the reasons are for not including something in an update. There could be too many false positives, or in the case of polymorphic viruses the signatures change and mutate over time, making it extremely difficult to spot. But it does go to show that you have to keep your guard up even when you have the latest and greatest update installed. Once an attack has gotten this far it boils down to the person sitting at the keyboard.

How to protect against it:

This vulnerability is not possible if the user has disabled File Downloads for the security zone that they are currently in. This has to be specified by the user as it is not the default setting.

Here is an example of how to get to the setting. In IE, pull down the Tools menu, select “Internet Options…”, click the Security tab, click the “Custom Level…” button, and scroll down until you see Downloads -> File Downloads. Here is a screenshot:



Disabling file downloads may not be very practical for most people. That being the case, the best way to prevent this exploit is by downloading the patch that Microsoft has issued at:

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

Follow the prompts to download the patch for your system. It will require that you reboot your computer before the changes will take effect.

As in any case involving security, it is absolutely critical to keep your systems patched and up to date. This will help to protect your systems from falling victim to many of the latest exploits. But this will only ensure that your systems are current and as prepared as possible. The weakest link then becomes the users themselves. As we have seen time and time again, there is no software that will always prevent an uneducated user from making a bad decision. Some of the most damaging exploits would not have been an issue if it weren't for vast multitudes of people being so trusting of everything that a computer presents to them. Can your users distinguish between what is typical behavior for your system and what is suspicious? What would they open an e-mail attachment just because it came from a person they know? Would they visit a web site that they found in spam without questioning it? What are their password practices? Are strong passwords enforced? Do users keep their passwords safe from prying eyes? Would they give out their passwords to someone who "sounds official" over the phone? These are they kinds of questions both small businesses and multi-national corporations need to ask themselves. If a user account is compromised or a user is persuaded into installing a trojan, sniffer, or other piece of malicious code on their machines all of the security measures put in place could be subverted.

It would be nice to be able to approach Microsoft with a proposed fix for these types of exploits, but the fact is that until we come up with a standardized secure architecture, we'll be downloading patches, updates, and new versions to keep our systems as secure as possible.

Source Code/Pseudo Code:

The exploit code was not documented at the source, and I have never been a web developer, so I had to play a game of search engine detective in order to come up with exactly where the exploit occurs in the code. This is how I believe it works.

In both of the text examples (there was an executable example as well), the file starts like a standard e-mail:

```
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
```

After the MIME version is defined, the basic content type is defined as "multipart/related". This tells the parser that the attached binary "parts" are a compound object. Compound objects rely upon each other and cannot be processed separately or interchangeably. The second type definition, "multipart/alternative", defines a subtype of the basic type. This tells the parser that the attached binary is made up of parts that are "alternative" versions of the same data, and that the parts are generally ordered with the best choice last for the local environment. In short, we have one content definition stating the attached parts must be processed in order, but that the parts are actually the same information, and try the last part first. So now what? Exactly. This is the point where IE should prompt the user about how to handle the attachment rather than running it.

Once IE is confused enough to run anything, taking advantage is simple. Here is a demonstration of using VBS scripting to create, write to, and close a file:

```
Content-Type: audio/x-wav;
        name="hello.vbs"
Content-Transfer-Encoding: quoted-printable
Content-ID: <THE-CID>

set objFileSystem =3D CreateObject("Scripting.FileSystemObject")
```

```
set objOutputFile =3D objFileSystem.CreateTextFile("C:\deleteme.txt", 1)
objOutputFile.writeline("You can delete this file.")
objOutputFile.close
msgbox "I have created the file : c:\deleteme.txt"
```

The content type of this part is defined as some form of audio file. But the file itself has a VBS extension. The content of "hello.vbs" is after the blank line. In this example, the author creates a text file containing a single line that is called "deleteme.txt", and then pops up a message box to tell the user about the file. Since IE has already been confused by type definitions it mistakenly finds the program associated with VBS files and launches it.

The second example demonstrates the ability to issue system commands:

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

echo OFF
dir C:\
echo YOUR SYSTEM HAS A VULNERABILITY
pause
```

As in the previous example, the Content-Type is defined as an audio file, and the file name is now given a BAT extension, or a DOS batch file. In this case, the author issues a command to list your top-level directory and print a line out alerting you that the exploit worked. Again, IE finds the program associated with the file type and passes the file to that program to run.

In both examples, the Content-Transfer-Encoding is defined as "quoted-printable." This just means that the content is printable ASCII text, rather than some form of binary data. In the executable version the Content-Transfer-Encoding is probably defined differently as the part will be in a non-ASCII binary form.

The Content-ID field actually turned out to be pretty interesting when it came to how it relates to multipart/alternative types, but it doesn't have any bearing on this particular exploit. If you would like to look into this further, see the following section for links to informative sites.

Additional Information:

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

Microsoft Security Bulletin. 29 MAR 2001. URL: http://www.microsoft.com/technet/security/bulletin/MS01-020.asp

Microsoft. BugTraq. 29 Mar 2001. URL:
http://www.securityfocus.com/archive/1/172664

Common Vulnerabilities and Exposures (CVE). URL:
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0154

Borenstein, Nathaniel S. Freed, Ned. Vaudreuil, Gregory M. "MIME (Multipurpose Internet Mail Extensions) Part One:Mechanisms for Specifying and Describing the Format of Internet Message Bodies." *Sep. 1993. URL:*
http://www.nacs.uci.edu/indiv/ehood/MIME/1521/Abstract.html

López de Aguileta, Jesús. "Re: User may be fooled to execute programs browsing with IE5.1" 4 Apr 2001. URL:
http://www.securityfocus.com/archive/1/173733

Cuartango, Juan Carlos Garcia. "MS patch Q292108 opens a vulnerability". 4 Apr 2001. URL:
http://www.securityfocus.com/archive/1/173940

RFC:
Borenstein, Nathaniel S. Freed, Ned. "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies." Sep. 1993. URL:
http://www.faqs.org/rfcs/rfc1521.html

Moore, K. "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text." Sep. 1993. URL:
http://www.faqs.org/rfcs/rfc1522.html

Sirbu, M. "A CONTENT-TYPE HEADER FIELD FOR INTERNET MESSAGES." Mar. 1988. URL:
http://www.faqs.org/rfcs/rfc1049.html

Levinson, E. "The MIME Multipart/Related Content-type." Aug. 1998. URL:
http://www.faqs.org/rfcs/rfc2387.html

Books:
Schneier, Bruce. Secrets & Lies: Digital Security in a Networked World. New York: Wiley Computer Publishing, 2000.

Scambray, Joel. McClure, Stuart. Kurtz, George. Hacking Exposed, Second Edition. Berkeley, New York, St. Louis, San Francisco. 2001.

*Special thanks go to Alice and Bob for always getting the short end of the stick.