ENG SC910 Spring 2002

Name: Jason M. Chu

Project: Email Retraction System

Advisor: Professor Trachtenberg

Abstract

With email as one of the primary mean of communication today, sending outgoing emails to incorrect recipients may be just as common a mistake as dialing an incorrect telephone number. The difference is, for incorrect telephone number dialing, upon the establishment of the connection, you will often be notified immediately about your mistake without having to reveal any personal messages at all. Email, however, is irreversible. If once the messages are send out and then you realized that you have made a mistake, you will not be able to "unsend" them.

The purpose of this project is to design an email retracting system that is capable of correcting such errors.

Applications

The Email Retraction System may be used for retracting emails that are either containing unintended information or are mistakenly sent to the incorrect recipients. This system will be especially useful if the sender need to frequently send email to large groups of mail recipients, in which huge amount of time and effort will be required if any mistakes are made.

The system should be compatible with at least one major existing email platform and the program should be as transparent as possible for the purpose of user friendliness. Installations of the program will also be required.

Introduction

Without having administrative privileges to the recipients' email servers, we will not be able to delete any email from their accounts and therefore we cannot physically "unsend" the email messages. However, we can prevent the recipients from reading the contents of the messages by using encryptions. That is, before sending any outgoing

email, we will encrypt the message body and send out the encrypted email instead. Each encrypted email will be assigned to a unique message identifier and it will be store in a remote server along with its corresponding decryption key.

Upon opening the encrypted message, the program will automatically send requests for the message's decryption key from the server and translate the encrypted message accordingly.

In the case that the emails are mistakenly send out. The sender will only need to change or delete the decryption key from the server. The recipients will not be able to read the email messages unless they obtained the decryption key. Thus we can retract emails message if and only if we change the decryption key before the recipients open our email.

Design & Technique

The first prototype of the Email Retraction System will be designed for Netscape Mail Client, a freeware that comes along with installing the Netscape Navigator. Outgoing emails will be send either in HTML format, or have an .HTML file attached to it. The encrypted message body will be embedded as an object in the HTML file, which means in order to read the message, we will need to define a new MIME type and develop a Netscape plug-in.

In addition, the plug-in will be compatible with most other email clients for Windows, such as Microsoft Outlook and Outlook express, Eudora, Mulbery, or Becky, as long as Netscape Navigator is set to be the default web browser.

The program will be designed with the Netscape Plug-ins SDK and the Microsoft Visual C++ 6.0 compiler will be used.

Plug-in Basics and Introduction

Plug-ins are developed by Netscape after the release of Netscape Navigator 2.0. The purpose of plug-ins is to increase the versatilities of the web browser. With the existence of plug-ins, graphics, media files, as well as various types of documents may be displayed on the otherwise plain HTML file. Different types of data files are assigned to certain MIME types. If an embedded object is

belonged to a known MIME type and its corresponding application program is available, the browser will automatically load the application and open the embedded objects.

Programming

The Netscape Plug-ins SDK may be used in the Visual C++ compiler. However, there are many adjustments to be made if we were to use Visual C++ v6.0 or higher. The latest download for the Netscape Plug-ins SDK was developed in January of 1998 for Netscape Communicator 4.0, and it seems to be more compatible with the Visual C++ compiler version 4.0 or 5.0. Therefore the necessary coding are often written in the C language instead of C++, and none of the Microsoft Foundation Classes (MFC) can be used without changing the format of the SDK. However, external Java program might be included to expand the functions of the plug-in.

The plug-in will be compiled in the format of a dynamic link library (DLL).

File Locations

All the available plug-ins in your computer are located in the “plugin” folder of the directory of the Netscape main program. After the plug-in’s DLL file was compiled, simply copy it to the “plugin” folder and restart the Netscape Navigator.

References

Netscape Plug-ins SDK Documentation:
http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm

Netscape Client Development:
http://developer.netscape.com/viewsource/naru_plugins/naru_plugins.html

Development

To write a Netscape plug-in in WIN32 environment:

1. Download and decompress the Netscape Plug-ins SDK
2. Install compatible compiler (such as Microsoft Visual C++) and open a new project for DLL applications. Project name must begin with NP and set MFC to “Static Link”.
3. Add to project the files that are included in the Netscape Plug-ins SDK. The Most important files are “wintemp.c” and “npwin.cpp”.
4. Determine a new MIME type to be used and the corresponding file extension.
5. Open the resource file (.rc) with a text editor to include the newly defined MIME type.
6. In project settings, use “select Precompiled Headers from Category list” and set to “Automatic Use of Precompiled Headers.
7. Edit the project (mostly in “wintemp.c” and “npwin.cpp”) to include any addition functions for reading the object, or modifications to any existing functions.
8. Compile the project to obtain the DLL file.

Additional Modules

Other than the main program for the plug-in, I’ve also write three other modules.

1. **Sender**

   Originally designed to be a Netscape macro. It allows the user to compose a message, send the identifier and the decryption key to the server, and compose the corresponding HTML file and message file.

   The message file will be an embedded object in the HTML file, in order to send the message, simply attaches both files to an outgoing email.

   Usage:

   Simply fill out the “Message Body” field and click on the “Make HTML” button. The “Email” and “Subject” fields are optional and will

simply provide more information to the mail recipient. Multiple messages may be composed at a time. After finished with writing the mail, simply click on the “Close” button to quit the program. At this time of the development stage, all the created HTML files will be saved to the directory: “c:\910test”.

Future Development:

1. Remote login capabilities
2. Message path assignment
3. Encryption strength options

2. Admin

Admin allows the editing of key database, including the capabilities to alter and delete keys. At this moment, the Admin program must be run on the message server.

Usage:

Simply type in the unique message identifier and click on the “Kill” button. The corresponding email decryption key will be deleted from the key database. Upon finishing, simply click on the “Close” button to quite the program.

Future Development:

1. Remote login capabilities
2. Database “clean-up”
3. Searching capabilities by email and subject

3. Server

Monitor incoming messages and return corresponding replies. The server must be run on the mail and key server – in most case it should be your home computer. It will handle messages from Sender, in which it will update the local key database, and messages from the Netscape plug-in, in which it will search for and return the message decryption key according to the unique message identifier. At the moment, the message key database is stored in an ASCII text file (data.key) located at “c:\910test”.

Difficulties

The most difficult part of this project is that it has only little or no resources and references. Most of the time when you get stuck on one-part, you usually will be stuck forever. There are a few user groups that will give you some pointers, but it far too insufficient. Most of the time spent on the project was on such researching. Also, since the plug-in is developed for Netscape Navigator and will be compiled to be a direct link library, we will not be able to easily use the debugger to trace the progress of the program. The outcomes of the plug-in are purely based on assumptions, experience, and faith. In that case if anything is wrong with the program, the developer will often have to go through the entire project line by line to have figure out what exactly is wrong.

Conclusion

I have learned a great deal about how plug-ins work and how to create them. However, the challenging part is to apply the plug-in with other programs and application, not the programming of the plug-in itself. In order to get everything to work together, we have to make sure that everything is compatible with each other, which is not as easy as it looks.