SECURITY IN COMPUTATIONAL ELECTRONIC MAIL

FOO Schubert, LEONG Peng Chor, SNG Yee Nam
Division of Software Systems
School of Applied Science,
Nanyang Technological University, Singapore
email: assfoo@ntu.edu.sg

Abstract

Computational electronic mail, or active mail, is essentially the embedding of programs or interpretable scripts within traditional electronic mail messages. At the recipient’s end, the mail reading software will distinguish it from normal mail, and executes it accordingly upon confirmation by the recipient. This follow-up article to “Computational electronic mail for library applications” examines the security implications of such a system, proposes and contrast a number of techniques to address security problems when using computational electronic mail. These include mail authentication through cryptography, using an improved computational mail scripting language and interpreter, the existence of an improved operating environment, using the in-built security features and access control of an operating system (OS), such as UNIX, to execute computational mail at a restricted priority and defined areas of accessibility; and containing the potential destructive consequence of computational mail in a localised environment by creating different disk partitions in less secure OS-based (such as DOS) systems to separately execute computational mail and other applications; or different disk partitions to support DOS and UNIX OS with the mail being executed under the UNIX environment.

Keywords: computational electronic mail, active mail, security, access control, safe language, Safe-TCL
1. Introduction

Computational electronic mail has numerous applications in the field of information technology automation which include electronic forms, survey generators, meeting schedulers, document distributors, electronic mail order catalogues, and so on. The first article, *Computational electronic mail for library applications* [1], described the design of such a prototype system, CEmail, to function over the Internet mail environment and demonstrated how such a system can be employed to automate a number of library functions. These include automatic overdue reminders with automatic renewal, survey generator, interest group information dissemination, general information dissemination and seminar scheduler, interlibrary loan servicing and electronic enquiry system.

Computational electronic mail systems exhibit huge potential of usage, coupled with low cost, high productivity and efficiency. However, they have not really taken off in a big way as its ASCII text-only counterpart! The main reason stems from security concerns among users. Recipients are particularly concerned with the potentially destructive consequence of mail execution, some of which take the form of loss of data integrity, loss of information, virus infection, theft of confidential information, and denial of service.

This follow-up article examines the security aspects in computational electronic mail, and proposes a number of possible approaches to minimise or eradicate security loopholes which exist in such systems. It first discusses potential security threats faced by users of computational electronic mail systems and subsequently examines the current approaches used to combat such threats. A number of new techniques and approaches are proposed and contrasted to further improve security in such systems. With improved security, computational mail is likely to gain widespread acceptance and use in future.

2. Security Threats In Computational Electronic Mail Systems

The fact that programs or interpretable scripts in CEmail are executed on the recipient’s computer system gives a malicious sender additional means to launch silent attacks to the recipient’s system which often goes undetected. Depending on the behaviour of such malicious programs, these new forms of attacks can be far more damaging than conventional ones. A number of common security threats exists in using computational electronic mail systems:

- **Mail interception or tapping during transmission.** Like ordinary electronic mail, computational mail is also subjected to mail interception or tapping by hackers as the mail is being transmitted across the network from sender to recipient. In the first case, intercepted mail can be discarded so that it never reaches the recipient. Alternatively, the contents (i.e. the program) are modified or rewritten to include malicious codes, and subsequently passed on the intended recipient. In the second case, mail is tapped along the transmission route. With a high enough frequency of tapping, important information (such as account numbers, passwords, etc.) may be stolen or revealed without being noticed by the user.
• **Mail impersonation.** In this instance, the hacker impersonates the sender and sends the mail to the recipient. The recipient receives the incoming mail and assumes that it is from a trusted source and executes it correspondingly with possible disastrous consequences caused by Trojan horse codes embedded in the mail which is invisible to the recipient.

• **Introduction of computer virus through computational mail execution.** A computer virus infects a host computer to cause damage, and can be transmitted across computers. A computer virus is generally written in machine code which can be copied into the host program when it is activated. The virus spreads to other computers through the exchange of such infected files. The destructive consequences of a computer virus can range from consumption of CPU time by spawning new processes, deletion and corruption of files, denial of services by filling up the hard disk, reformatting of hard disk, and so on. A computer virus is difficult to detect as it may lie dormant for some time before being triggered at some date or by some event.

• **Introduction of Trojan horse and related threats.** The computer version of the Trojan horse performs a separate (usually destructive) function in addition to that originally intended. It fools a recipient in thinking that it is carrying out a legitimate operation while in reality, it may be performing some other malicious activities. In addition, other related Trojan horse threats can be introduced incrementally over time by executing a series of seemingly legitimate operations until the whole Trojan horse program is complete and execution takes place. Trojan horses, like computer viruses, are difficult to detect until such time something out of the ordinary happens to the system.

### 3. Current Approaches To Improve Security In Computational Electronic Mail

A number of current approaches have been used to improve security in computational electronic mail systems. These are basically aimed at combating the security threats identified in the previous section. The threats of mail interception, tapping and mail impersonation coexist in passive electronic mail systems. As these systems have been around for much longer than computational mail, universally accepted solutions to such problems are already in existence. These approaches are summarised and contrasted:

• **Mail Encryption and Verification.** Encryption is defined as the transformation of data into a form unreadable or undecipherable by anyone without a secret decryption key. Current encryption efforts are focused toward allowing secure communication to take place over insecure networks, such as the Internet. Substantial effort has been focused on creating digital signatures for authentication due to commercial reasons. Encryption basically ensures data confidentiality while authentication ensures data integrity. Such an approach can be adopted in computational mail systems to enhance security of data been transmitted and as a means to provide some form of verification to ensure that the sender is who he or she claims to be.
Two kinds of encryption techniques can be used. **Symmetric encryption** uses the same secret key to encrypt and decrypt a message, while **asymmetric encryption** (or public key encryption) use one key (the public key) to encrypt a message and a different key (the private key) to decrypt it, or vice versa. **Pretty Good Privacy** (PGP)[2], which implements the public key encryption using the RSA and IDEA algorithms, is a widely accepted tool to provide secure electronic mail communication. Such an approach is suitable for private computational mail applications between individuals or known sources.

- **Password Authentication.** Authentication is the process to ensure that the computational mail that a recipient receives is from a verifiable source. It is usually implemented using passwords together with symmetric encryption. The password which may or may not be encrypted is first made known to the recipient of the computational mail. This is subsequently sent together with the computational mail. During the authentication process, the recipient enters the password. Authentication is successful if the passwords match. The computational mail is subsequently executed upon confirmation by the recipient. Such a password system is not truly fail-safe. A password can be intercepted while travelling on the network; and a program masquerading as the authentication program and asking for the password are possible ways to capture the password for eventual use.

- **Access Control List.** Access control list (ACL) provides a means for system administrators and individual users the power to set permission on files, directories and hence system resources to other users. In a UNIX-based OS system for example, files can be set with the permission of read(r), write(w) and execute(x), and directories with read(r), access(x), create or remove(w), for users belonging to system, owner(u), group(g) and world(o). While such a mechanism can give rise to better security and contain the damage done to the system, it implies that control of local resources are left into the hands of the user. In this instance, an ignorant user may inadvertently grant access to the computational mail program to more resources than desired, thereby increasing the potential of causing more harm than intended.

- **Auditing.** Auditing is the process of keeping logs for the purpose of identifying the sequence of commands and actions which are taken by a user or computational mail during an on-line session. It records the movements of entities through and within the system and aids the auditor in tracing and recovering from situations, or predicting what has happened to the system. The obvious drawback of this approach is that whatever intended harm would have already taken place, and the log merely gives an indication of what might have happened but is likely to be insufficient to allow a complete recovery to take place.

- **Restriction of Scripting Language or Interpreter.** A computational mail can be constructed using two approaches. The first approach is to compose and compile the mail into an executable form at the sender's end. This is transported in binary form to the recipient's end where it will be executed upon activation by the recipient. The second approach is to use an embeddable and interpretable language (e.g. TCL[3], Safe-TCL[4], UNIX shells[5]) to generate a script file which requires no compilation
at run time. The script file is transmitted, loaded and executed at the recipient's end. The first approach is generally not adopted due to security concerns during execution and the requirement for the recipient to have a compatible computer system in order to be able to execute the mail.

As such, the second approach is preferred and computational mail are generally constructed using an embeddable and interpretable language. In the case of a normal scripting language such as TCL or UNIX shell, the command set comprises entities (such as `open` and `delete`) that can used for malicious and destructive purposes. A possible solution is to parse the computational mail script to ensure the absence of such commands prior of execution. However, there are various ways that such a safeguard can be overcome. Commands can be constructed during runtime. For example, concatenating ‘o’, ‘p’, ‘e’ and ‘n’, or rearranging the word ‘epno’ will and can result in ‘open’ which can be still be executed at runtime and yet not been able to be detected by the parser. Thus, a language with strong expressivity will virtually make it impossible to check all the commands that are eventually executed.

Under such circumstances, an approach is to use a restricted version of the scripting language and to remove all such harmful commands that can be put into malicious use. This has been adopted in Safe-TCL which is essentially a modified form of TCL in which dangerous primitives have been removed and new primitives added. Along with Safe-TCL, a new interpreter is used to execute these scripts under a new environment. In this situation, the mail reading program evaluating the Safe-TCL computational mail will always have within it both the TCL and Safe-TCL interpreters. The TCL is the trusted interpreter and the Safe-TCL is the untrusted interpreter used for executing mail from untrusted sources.

In using this approach, a trade-off exist between security and usage power of computational mail. In a total secure system, dangerous commands are removed to the extent that the host computer acts as a virtual machine without resources for the computational mail to access. The recipient reads the mail and interactively provides input to the mail, if necessary. This mode of operation is still useful for applications such as electronic surveys, meeting schedulers, marketing and advertising products and services to customers, and so on. However, if more advanced features, such as automatic extraction of data from the host computer is to be realised, then the scripting language must be extensible to allow such operations to take place. Such extensions parallel potential security problems, since more extensions (or combinations of certain extensions) will inevitably pose greater security threats. The extreme would be a computational mail program which when executed will behave as if it is owned by the recipient with full access to the host computer.

Despite the various approaches which have been adopted in improving the security of computational mail systems, it should be obvious that when the computational mail is finally allowed to execute, it becomes a trusted application with possibly some form access to the recipient’s computer system so that the various potential security threats becomes a distinct possibility. It is also evident that a trade-off exists between security and usage. It is concluded that these existing approaches do address the security aspects
of computational mail to some extent but not a sufficient level to make it acceptable to all users.

4. Techniques And Approaches To Resolve Security Issues

A number of techniques and approaches are proposed to further improve the security aspects of computational mail. In one of the approaches, it demonstrates that if everything still fails, how the ‘damage’ can be localised and contained within the host computer system.

- **Authentication Through Cryptography.** This follows the current approach in which computational mail is first encrypted using public or private keys. This ensures that the encrypted message does not provide any information that it is carrying even if it is intercepted by hackers. The recipient of the mail can decrypt the mail using the respective private or public key. The advantage to this approach is that the recipient is certain that the message is from the original sender and the contents has not been changed or lost during data transmission from sender to recipient. As a further safeguard, a digital signature in place of password authentication can be attached together with the mail to further confirm the identity of the sender. Such an approach is generally more utilised in interpersonal communication between known correspondences. However, by using a combination of asymmetric and symmetric encryption techniques [6], it is still possible to extend this approach and apply it to general and large group applications. This basically requires an additional process by the recipient to obtain the secret key information prior to decrypting the mail.

- **Improved Computational Mail Scripting Language or Interpreter.** If a better and suitable scripting language or interpreter is used for computational mail, then a significant part of the security problems can be minimised. The language should provide strategic locations where security measures can be added to restrict the power of computational mail. At the same time, flexibility should be allowed to provide more privilege to the computational mail upon the discretion of the user. Although, such languages as Safe-TCL and JAVA exists, these are still at their infancy stage with scope for further improvements.

It may be possible to ensure that these interpreters execute the programs within a safe environment which is specially dedicated to computational mail. In addition, some form of security manager (or master interpreter) to monitor the action of the program at runtime may be included to ensure that no dangerous commands are executed. For instance, the previous ‘open’ command example can easily be detected at runtime prior to execution and the operation disallowed by the security manager.

A set of user-defined new but safe commands may be created and managed by the security manager. For example, a `safe_read` command may be defined to replace the open and read command. The active mail may subsequently use `safe_read` to read the contents of a file on the host computer. When this request is detected, the security manager will be responsible to open, read and subsequently return the
contents of the file to the requester. The requester obtains the desired information without accessing the file system directly but through the security manager.

User defined commands not recognised by the security manager will not be allowed to execute. Thus, commands are confined to those maintained by the security manager who controls fully the execution of such commands. As a last check, all such commands can be optionally communicated to the recipient prior to execution. The recipient must acknowledge and give the go-ahead before the command executes. Although this process hinders automation, it nevertheless keeps the recipient informed of each impeding action of the mail. Furthermore, an audit log of the whole sequence of computational mail execution is kept by the security manager.

All these improvements effectively controls what a computational mail can achieve at the recipient’s host computer and at the same time, ensures that the commands are executed safely to achieve the desired operation.

• **Improved Operating Environment.** The two operating systems (OS) which is likely to be used for computational mail applications are DOS and UNIX. However, these are two distinct OS with very different security characteristics. DOS is an open system which was designed for the personal computer platform. Little or no security features exists since it assumes that the owner will be the sole user with full access to the whole system. In contrast, the UNIX system was designed to support multiple users, time-sharing, and a networked environment. It is much more robust in terms of security. When used in conjunction with computational mail, it offers more protection to the user than compared with DOS. The Access Control List mechanism, when used in the correct manner, can prevent a computational mail from tampering with the file system of the recipient’s machine except for designated directories. It helps to protect other users in using computational mail under the same environment. Thus, UNIX is a preferred OS over DOS in executing computational mail applications.

• **Execution of Computational Mail at Restricted Priority.** If the computational mail can run with a priority lower than an ordinary user then it will not be possible for it to assume the rights of the recipient unless it been explicitly given by the recipient. Thus, it becomes possible to ensure that the mail has no general purpose access to the file system, but only a limited form of access to a subset of the file system. The interpreter is expected to ensure that no tricks (such as ‘..’ or symbolic links in UNIX) can be used to access files outside the defined areas. In addition, it must be recognised that any form of provision of privileges to a computational mail can open up security loopholes.

• **Execution of Computational Mail In A Localised Environment.** This approach uses the rationale that if all security measure fails and that some form of damage will occur in the host computer, then it would be better to localise and contain the damage within some defined environment. This translates into the execution of computational mail in a designated environment which is totally separate and where there is no access to the normal file systems or privileges. In a DOS OS-based system, with built-in protection against low level disk formatting, this can be
achieved by creating separate disk partitions to separately execute computational mail and other DOS-based applications. Alternatively, different disk partitions can be created to support the DOS and UNIX OS with the computational mail being executed under the UNIX environment with the aforementioned security measures. Such a measure may prove cumbersome in practice, but it ensures the other system resources remain safe. In the extreme situation, separate computer systems can be dedicated to do nothing except to execute computational mail.

This localising effect with proper practices reduces the security threats in computational mail. The destructive consequences even if carried out are contained. For instance, destruction of resources such as deleting files are confined to the localised environment; deprivation of resources such as filling up the hard disk, likewise, is confined to the localised environment without affecting the functionality of the host system. Although the program can still tie up the CPU, virtual memory or swap space on host computer by spawning new processes or going into infinite loops, such problems can be handled by means of resource limitations. In order to minimise infection or attacks by computer viruses or Trojan horses, all files written by the computational mail should be deleted immediately. If such files are to be processed, copied to anywhere else within the computer system or transmitted to other users (via email), then some form virus-free check must be implemented.

5. Conclusion

This article has examined the potential security threats faced by users of computational mail systems, reviewed the current approaches used to combat such threats, and proposed a number of new techniques and approaches to further improve security of computational mail. These include authentication through cryptography, the use of an improved computational mail scripting language and interpreter, the existence of an improved operating environment, the execution of computational mail at restricted priority and defined areas of accessibility; and to contain the potential destructive consequence of computational mail in a localised environment. With the adoption of these proposals, computational mail will without doubt, be more robust against security threats. With the potential applications and payoffs of computational mail been made known and the security aspects resolved, computational mail is likely to gain much more widespread acceptance and use in the not too distant future.

References


