Proceedings of the 2003 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 18–20 June 2003

T1B2   1555

A Distributed Multipurpose Mail Guard

Stephen D. Wolthusen

Abstract—

This paper describes a mechanism for incorporating a mail guard mechanism together with automatic, mandatory, and fully transparent digital signatures and encryption for message traffic embedded into the operating system of individual network nodes. By intercepting all inbound and outbound network traffic and analyzing for pertinent information using generalized Büchi automata, the guard mechanism can enforce the application of (centralized) mail security policies without requiring any support from mail clients.

An implementation based on modular modifications to the Microsoft Windows NT/2000/XP family of operating systems and OpenPGP-based messaging is described.

I. Introduction

MOST governmental and corporate security policies encourage or even require the use of cryptographic protection mechanisms for confidentiality, integrity, and non-repudiation.

Moreover, the risks of unprotected message traffic, both within intranets and via the public Internet are well-publicized, and application programs and utilties for secure electronic mail are readily available either integrated into existing applications or as add-ons at negligable costs. Despite this, the actual use of such mechanisms is apparently limited to a narrow user population and even within this group not employed consistently [1], [2].

Even if technical problems such as key exchange and key authentication as well as the problems of user interface design [2], motivation and awareness [1] were addressed, one may assume that a number of other problems would preclude the widespread use of secure electronic mail, but particularly competing requirements. The additional operations required for encrypting or signing a message (and, conversely, validating such signatures and the integrity of the message) are discretionary and may not be performed under time pressure or in the presence of other distracting influences. At the same time not only defense-related activities require control over the type of traffic that is both transmitted and received, even in unclassified environments. While centralized guard mechanisms can provide high assurance services for outgoing traffic and, similarly, virus and malware checking mechanisms can be deployed centrally, this approach requires a highly structured network environment with a well-known topology and no

S. Wolthusen is with the Security Technology Department, Fraunhofer-IGD, Darmstadt, Germany. E-mail: wolt@igd.fhg.de.

transitive network connections. Such network structures are increasingly difficult to retain; it is therefore desirable to have a guard mechanism that does not rely on such boundary conditions.

Based on these observations we conclude that security policies in this area can be effective only if their mandatory enforcement is performed through technical means and describe a mechanism for achieving this objective.

The remainder of this paper is structured as follows: Section II describes the main issues in providing such technical means in a heterogeneous environment dominated by COTS systems and applications, while section III discusses several key aspects of an implementation of secure messaging for such an environment. Section IV then discusses other approaches to mandatory enforcement of messaging traffic security properties; section V finally provides a summary and lessons learned.

II. Mandatory Interception

For the purposes of this discussion, secure messaging is limited to store-and-forward type messaging, typically referred to as electronic mail. While related mechanisms such as "instant messaging" may also be covered by the same or a slightly modified approach as described here, such applications and protocols are typically covered by a different security policy element. However, even for electronic mail, a number of possible transmission paths each utilizing different protocols and encoding standards must all be taken into consideration to provide both seamless and invisible security mechanisms, as will be described in section II-A.

Given that networks, even arguably "secure" networks should be considered untrusted given the limited effectiveness of topologically oriented security mechanisms [3], performing cryptographic operations at central mail servers leaves the message traffic vulnerable (e.g. to a man-in-the-middle attack) between the mail user agent (MUA) and the mail server [4] and, moreover, incurs severe problems in key management since user-oriented key material for digital signatures and encryption is stored at a central location, providing an attractive target.

It is therefore highly desirable to perform all required transformations on the message traffic immediately on a host that is preferably under the physical control of the user and where the amount of sensitive material (e.g. encryption keys) is limited both in quantity and in time.

Moreover, as noted in section I, similar considerations with regard to centralized facilities also apply to guard and other filtering mechanisms in the relative efficacy provided in networks that are unstructured or even ad hoc in nature and which may contain elements accessible by way of wireless interfaces. In this case integrity and confidentiality must be provided as end-to-end services to avoid man-in-the-middle attacks, forged, or otherwise intercepted message traffic. Besides the imperative for providing such mechanisms in the form of end-to-end services, a core concern for the implementation of messaging security is seamless interoperability both with MUA application programs (which must be assumed immutable since these are typically COTS applications) as well as with MTA (Mail Transfer Agents). The latter not only applies to remote MTA systems, but also to local MTA systems since these represent a critical part of an organization's communication infrastructure.

Since inadvertent or deliberate misconfiguration would be problematic even in case of protocols where an explicit interposition of a proxy is supported by the relevant protocols, such an approach (e.g. used by [5]) is limited in its applicability to certain benign protocols and application scenarios. In the general case, interception of messaging traffic must be both mandatory (i.e. all traffic must be interceptable) and, if necessary, transparent. In case of protocol support for intermediate (proxy) hosts, interception can be achieved by forcing traffic to use the transmitting (receiving) host itself as the proxy; for other protocols inline interception and transformation is required.

All such interception should, moreover, occur at the level of the operating system since this provides to some extent protection against manipulation from users or malicious code at the application level. In addition, positioning such mechanisms at the operating system (more precisely, at the network stack level) provides a generalized mechanism for interception, a requirement that is of considerable significance as will be elaborated in the following section.

A. Multiple Pathways

Although Internet electronic mail has become by far the dominant messaging platform even for pure intranet communication, there exist several mechanisms that provide alternative or complementary pathways for message traffic such as the X.400 suite of protocols or proprietary systems such as VMS Mail, Lotus Notes, or Microsoft Exchange which must also be dealt with.

However, even in case of standard Internet mail — which has proven to be remarkably stable over the past two decades — the raw transmission protocol, SMTP [6] has undergone multiple revisions and extensions (e.g. [7], [8]), which must be handled properly. In the case of SMTP and its variants used between transmitting MUAs and MTAs (and between MTAs, although this is not a primary concern here), the mechanisms described in the preceding section can be realized by way of interposing a mandatory proxy mechanism that acts as the first hop of the store-and-forward chain. This restriction can be enforced by ensuring that the SMTP protocol and its variants is not used in any network communication other than by the proxy mechanism itself. This, however, covers only mail sent from a user. For users receiving standard Internet mail there are multiple options with the most common protocols being POP 3 [9] and IMAP 4 [10] (which, again, have been subject to a number of extensions and revisions which must be supported). As with SMTP, the protocols can be recognized and intercepted at the network protocol level in such a way that a mandatory proxy performing transformations such as decryption and signature verification is interposed.

Similar considerations also apply to proprietary mail exchange formats; well-defined protocols can be recognized and thus intercepted by an interposed mandatory proxy. In cases where such information is not available or unreliable, heuristics may help to establish the missing protocol information. However, depending on an organization's security policy, such protocols may also simply be blocked and messaging restricted to approved protocols and application programs. Another possible mechanism for both sending and receiving of electronic mail include Web-based interfaces. This represents an extreme case of the proprietary protocols discussed in the previous paragraph since the actual formatting and hence protocol of the message traffic are not only completely arbitrary within the frameworks provided by the transport (mainly HTTP, but conceivably also FTP or more exotic protocols) and representation protocols (e.g. HTML, XHTML, XML) used as well as the specific representation chosen.

In the general case it will not be possible to reliably intercept arbitrary Web-based messaging services, as e.g. a client-side script program (e.g. JavaScript/ECMAScript) can be trivially used to arbitrarily encode or even encrypt data for transmission in such a way that protocol recognition cannot be performed (this problem is obviously undecidable even if a fixed protocol is assumed prior to encoding). However, for well-defined and benign applications (e.g. where the Web-based messaging application is well known and such traffic restricted to an also well known set of servers), the same observations made with regard to SMTP can be made.

A significant complication compared to the protocols described earlier does arise in the form of optimizations in HTTP 1.1 [11] since this revision of the HTTP protocol allows for the pipelining of multiple requests within the same transmission; this implies that the sequencing of individual protocol data units is not predictable and therefore also that interception must be capable of detecting relevant protocol data units within what is, to a first approximation, an unstructured bidirectional data stream.

B. Encoding Mechanisms

While interception can be based purely on supported protocols or recognized and certain elements (e.g. encryption and decryption) of subsequent processing and transformations can occur without knowledge of the encoding and message transport format (besides requisite information such as sender and recipient information, which are part of the protocols), several important transformations (e.g. filtering for keywords, scanning for malware) cannot proceed without knowledge of the encoding syntax and possible additional transformations applied on the part of the MUA application programs.

Again, standard Internet mail has also evolved a large number of possible encoding formats since the first standard definition [12] mainly related to the transport of multimedia data types. The MIME (Multimedia Mail Extensions) standards [13], [14], [15], [16] provide an extensive selection of encoding and transmission formats as well as means for embedding arbitrary new multimedia types within the processing framework provided by MIME. Typically, however, MUAs support only a subset of this selection [16]. For a MIME message to be subjected to transformation, it must be decoded in a somewhat elaborate process. Each MIME message may contain several parts, which in turn are subject to several processing steps until the desired decoded information is available. Depending on the transport path selected (e.g. [12] supports only 7 Bit ISO 646 transmissions), it may first be necessary to revert a content transfer encoding, which can be different for each part. The MIME standard requires[1] the definition of a content type for each message or part (such as the type `application/postscript` for Adobe PostScript documents). In benign cases this information can be used to infer appropriate processing steps by the guard; however, the reliability of this type information is frequently limited since message types for which MIME types exist are not identified properly or can be misidentified (e.g. by using heuristics based on file names instead of syntactical analysis on the part of the MUA). For transformations by the guard requiring detailed type information, it is therefore frequently necessary to perform a syntactical analysis independent of the MIME content type.

Besides the already mentioned possibility of providing multiple parts within a single message (each potentially of different media types), the MIME standard permits several additional mechanisms which complicate the processing of messages. One such mechanism is the fragmentation of messages (typically used if individual messages exceed a size threshold); in this case the sequencing and path delay problems familiar from packet-based networking occur and a fragment reassembly at the guard level must occur. Another mechanism is the use of message body types. In

[1]The absence of a content type definition is assumed to denote a 7 Bit ISO 646 text

most cases a MIME message body is provided inline within a message. However, it is also possible to specify external message bodies, in which case messages received contain only references to other resource locators from which the MUA or the user can retrieve the catual message (part) body. In this case (e.g. when using FTP external message bodies) the previously mentioned general interception mechanism must be employed to provide the guard mechanisms. General interception mechanisms are also required in systems where messages or message parts are transmitted in the clear as part of the editing process, e.g. in case of a Microsoft Outlook / Exchange environment (attachments of draft messages are transmitted via MAPI RPC [17], rendering purely application-based encryption within the Microsoft Outlook MUA ineffective).

C. Encryption Mechanisms

As noted in section II-A, a variety of transmission and encoding methods complicate the transparent interception of mail traffic for the purposes of providing security services. Of particular interest is a mechanism that is nominally intended to improve the security of the overall information system, namely the use of encrypted channels for sending and especially retrieving mail (e.g. to protect messages received by a user outside a protected network or communicating via exposed wireless interfaces). For IMAP, POP, and ACAP [18] specifies the use of the TLS protocol; this commonly implemented protocol does, however, result in the introduction of a potential vulnerability. By ensuring that the message reception is opaque up to the level of the MUA, security mechanisms such as malware scanners are excluded or at least limited to operating on server systems. Similar considerations also apply to guard mechanisms operating at the sender side.

In case of mails being encrypted at the application level (which may occur in addition to the transport level encryption specified by [18]), the server-side options are even more limited. Server-side systems are either blinded to encrypted data streams terminating at the MUA level of individual systems, or they must centralize message encryption and decryption.

The latter raises a number of issues. First, centralized encryption and guard services leave the message traffic vulnerable while in transit within the logical perimeter the central instance protects. Second, the centralized key management implied in this approach presents an attractive target to adversaries, and, moreover, can in the case of compromise cause a severe disruption of protected communication since it must then be assumed that the entire key material retained at the central site was compromised simultaneously. Finally, centralized key management may not be feasible based on legal or regulatory grounds requiring e.g. the unambiguous identification of a deliberate act by a specific individual as the origin of a digital signature.

As a result, not only the sending but also the reception of encrypted messaging traffic (including the potential superencryption introduced by [18]) must be intercepted mandatorily and processed according to a given security policy prior to the delivery to a MUA.

This requires the interposition of a transparent proxying mechanism that is capable of processing SSL/TLS messages at the operating system level (depending on application requirements the resulting traffic may be forwarded in the clear for performance reasons, causing the application to consider the traffic to be in the clear, or by re-processing the same message again with SSL/TLS). Similarly, providing a transparent interception mechanism for application-level message protection on reception has additional benefits besides providing an access point for processing steps such as malware scanning prior to the MUA. It may provide existing MUAs that do not have integrated facilities for message encryption and integrity protection with such standardized services externally in situations where the use of legacy systems (e.g. proprietary messaging systems) are mandated. The most important such service, however, is the protection of key material and the identification and authentication services for providing access to key material. Key material that is not stored on external media such as smart cards or tokens is susceptible to application programs, other users, or malware reading the key material. The only protection for such keys then is typically key encryption by way of a password or pass phrase, which may be itself susceptible to dictionary attacks or simple brute force if of insufficient strength. Providing only indirect access to such key material mediated by the operating system can reduce the threat in this case.

Moreover, the integration of mediated key management can also limit the vulnerability to another type of attack that can also apply to key material stored on smart cards, reading the pass phrase or PIN to a smart card[2]. This attack can be performed by intercepting the keyboard input of the legitimate MUA (as is commonly accomplished in GUI environments by intercepting message traffic to applications), or simply by using a Trojan horse for this purpose.

By using the trusted path provided by most operating systems for identification and authentication purposes, the mediated key management mechanism can protect against such attacks. However, additional threats of this type remain in the physical interception of keyboard data – whether in the form of easily available intermediate plugs surreptitiously placed by an adversary or in the form of electromagnetic radiation from cables or simply wireless keyboard interfaces.

A beneficial side effect of the transparent proyxing of inbound message traffic is that even if the sender of a message is not equipped with the mechanisms described in this paper, it is nonetheless still possible to provide the transparent verification and decryption services to the user, resulting in a homogeneous user interface for such message traffic. This signaling can be accomplished e.g. by modifying the received message in such a way that it is "wrapped" in a message denoting the status with regard to the protection mechanisms. Even if the receiving MUA is capable of handling such services, however, it is still desirable to externalize such signaling since messages containing active code may generate deceptive output to mislead users.

III. IMPLEMENTATION

While the mechanisms described in this paper apply to most modern operating systems, several aspects of their implementation are of necessity heavily dependent on the characteristics of the operating system into which they are to be embedded.

As a result, a reference implementation based on the Microsoft Windows NT (2000/XP) family of operating systems has been created with additional prototype development performed on the Sun Solaris operating environment; the following describes the implementation on the former platform.

A number of interlocking components are required to obtain the functionality described in section II. Section III-A outlines the mechanisms necessary for forcing the relevant network traffic of a host through the mandatory interception mechanisms, while sections III-B and III-C discuss the processing steps performed on messages themselves. Section III-D covers the isolation of key management from individual users and the trusted path identification and authentication, and section III-E describes the de-blinding mechanism for superencrypted messaging traffic.

A. Network Flow Enforcement

To ensure that all messaging traffic is indeed captured and intercepted, the overall network traffic must be controlled. In the Microsoft Windows NT family of operating systems, this is complicated by the fact that multiple network APIs (WinSock, Named Pipes, Mailslots, Remote Procedure Call, NetBIOS, and Telephony services), each with different control flows, exist. Most of these APIs must, however, bind to the NDIS (Network Driver Interface Specification) layer through one or more intermediate layers. Upper-level APIs such as NetBIOS and Windows Sockets are implemented at the user level and refer to a mechanism called the Transport Driver Interface (TDI), which consists of transports or protocol drivers. The latter must then finally bind to the NDIS layer [19].

By controlling the TDI protocol driver binding mechanism at the NDIS level and either intercepting at that level or disabling undesirable protocols (e.g. NetBIOS) altogether, the protocols available can be suitably restricted.

---

[2]Some systems eliminate this problem by providing a separate PIN pad, but this requires the use of somewhat bulky hardware external to a computer system.

In case of the mechanisms discussed here, the restriction is to the WinSock API and the TCP/IP protocol driver associated with this interface [20], [21]. The WinSock API provides the exposed API DLL (dynamic-link library) which communicates with the SPI (Service Provider Interface) layer, which is controlled by the transport service provider DLL. This, in turn, calls on a number of transport helper DLLs and namespace helper DLLs to perfom its mission. On the other hand, the transport service provider DLL forwards the thus generated calls to the System Support Library DLL that represents the interface to the abovementioned kernel components. Since the Windows NT design is predicated on a file system model and represents sockets as file handles, a translation mechanism is required. This service is performed by an Ancilliary Function Driver (AFD).

The interception of network traffic can be accomplished by inserting a Layered Service Provider (LSP) into this framework, which requires the insertion of the interception layer in the Service Provider Chain Catalog Entries for the relevant protocols (e.g. TCP, UDP). As the manipulation of the requisite data structures must be performed by a thread with administrative privileges, the spurious insertion of additional malicious LSPs is not possible for regular users. Moreover, the proper sequence of chain catalog entries can be verified periodically or on updates to ensure continued operation. The thus inserted LSP can now use access control mechanisms for ensuring that no message protocol traffic is transmitted without prior processing. To this end, unfiltered communication can be blocked by means of an access control list. This mechanism is independent of the filtering callback introduced in Windows 2000, which (besides lacking backward compatibility to earlier operating system revisions) is limited to a single decision per-packet.

As will be described below, it is necessary for the proper analysis and procesing of the messaging traffic to be able to look ahead in the message stream and withhold flows until their characteristics can be established based on additional packets. Any traffic that is passed through must then be analyzed as described in sections III-B and III-C or, for lower threat environments, the analysis can be restricted to the well-known network ports for the messaging protocols. However, in either case the network traffic destined to and from messaging protocol servers is intercepted and only released (and possibly in altered or suppressed form) after processing.

B. Outbound Processing

The LSP mechanism described in the preceding section provides a transport interface that establishes a properly sequenced data stream; it is therefore not necessary to perform network layer tasks such as the (de)fragmentation, flow control, or sequencing of data packets.

For outbound processing there are two main objectives. One is the interception of all relevant messaging protocol data units (PDU) and proper processing of such PDUs to ensure that messaging traffic is handled according to security policy requirements, i.e. encrypted and prepared for in-band transmission. The other objective is the realization of the guard mechanism. This is a subordinate analysis that can occur effectively only after the initial PDU processing has occurred since the messaging traffic may well contain transport-encoded data. Any guard mechanism must therefore be applied to transport-decoded streams (but may also be applied to raw data streams; however, this leads to a significant performance penalty).

Both filtering steps can be achieved by a set of timed general Büchi automata $\mathbf{A} = \{(\mathbf{Z^i}, \mapsto^\mathbf{i}, \mathbf{l^i}, \mathbf{Z_0^i}, \mathcal{F}^\rangle)\}$ where for each i, $(Z^i, \mapsto^i, l^i)$ is a transition system with a fixed alphabet, $Z_0^i \subseteq Z^i$ the set of initial states and $\mathcal{F}^\rangle \subseteq \mathbb{P}(\mathcal{Z}^\rangle)$ the set of sets of accepting states [22].

Each automaton may undergo several transitions prior to establishing either an eccepting state or rejecting the data stream; the outbound stream can only be forwarded if no automaton is still in an intermediate state. Examples for automata include "outer" or transport protocols such as SMTP or the Lotus Notes protocol, and "inner" or encoding protocols such as the MIME protocols (cf. section II-B) or the Microsoft Transport Neutral Encapsulation Format (TNEF) used by Microsoft Outlook/Exchange. A third automata layer may involve the decoding of application-specific protocols such as XML or Adobe PDF to extract the requisite keys for the guard mechanism. Timed automata are required to avoid stalling the network traffic in case of partial matches that do not lead to a rejecting state (e.g. typing PDU elements interactively over a terminal session). Once the relevant innermost protocol of the intercepted traffic has been decoded, the guard mechanism can then (depending on the automata representing the proscribed keywords etc.) perform the actions required by the security policy (e.g. cause the message to be forwarded for manual inspection and generation of an audit event). Provided that the guard mechanism has approved the message, the mandatory encryption mechanism can then check if policy requires messages for the recipient to be signed electronically. If so, it causes keys to be retrieved for this process (see section III-D) and the signature to be applied. Similarly, after checking if encryption is required, key material is obtained and the message encrypted.

The signing and encryption process depends on the encoding used by the encoding protocol; in case of the Internet standard MIME protocol, any message including its attachments etc. can be placed within a nested MIME encoding and therefore processed in its entirety. For the Microsoft TNEF encoding, the preceding steps must be iterated twice since such messages contain a plaintext version of the message and an attachment containing several

other elements, typically including an RTF-formatted text version of the message, Object Linking and Embedding objects (e.g. embedded Microsoft Office documents), and additional proprietary workflow data such as forms or meeting requests. Obviously, once the message has been decoded, a number of additional processing steps can also be included, e.g. the insertion of digital watermarks into multimedia content to provide media-independent annotation information (e.g. origin of the message, identification of sender) [23]. In either case the processing can occur conforming to an established standard for mail message encryption. The implementation described here conforms to the OpenPGP message standard [24], although it is also possible analogous to the decoding process to select the encryption standard based on information and key material available for the recipient.

Only after these three steps can the message be regenerated in the requisite transport protocol and again enqueued into the network traffic. The resulting messages can be read by any MUA capable of receiving the format chosen for digital signatures and encryption (e.g. OpenPGP-compliant), it is not necessary that the recipient also is equipped with the mechanisms described in this paper.

C. Inbound Processing

The processing of inbound network traffic occurs analogous to the steps outlined in the preceding section. Again, for all parts of the traffic that should be processed (which will typically be POP, IMAP, WWW, and potentially also proprietary protocol network ports), a set of timed general Büchi automata for each transport and encoding protocol must be applied to the inbound network data stream.

Accepting states for a transport protocol then leads to the activation of the automata for identifying encoding protocol, which can in turn determine whether a message must be decrypted and digital signatures verified.

If this is the case, the message must – as in the case of outbound processing – be intercepted in its entirety, its format determined and key material as well as public key certificates retrieved if necessary (see section III-D). After decryption and verification, the message then can be reconstituted; this is again dependent on the encoding protocol used (see section III-B), e.g. for MIME messages the outer MIME layer used to encapsulate the signed and/or encrypted message can be replaced with another layer identifying the status of the message.

As a result, any MUA, even one not capable of supporting encrypted messaging can receive such messages and have immediate confirmation of the message status.

It should be noted that this transparent interception and substitution applies not only to mail retrieval protocols such as POP and IMAP (as well as proprietary protocols), but also to WWW-based means for mail retrieval. Since the means for encoding are the same (using MIME for iden-

tifying the encoding format), the activation of Büchi automata for this reception channel will also result in transparent and automatic decryption.

D. Identification, Authentication, and Key Management

By retaining access controls to key material and configuration data such that no process with user privileges has direct access to this material and mediating all accesses to this material, additional protection against malware and unauthorized use is ensured.

Access to the (encrypted) key store is additionally mediated through a pass phrase (possibly in addition to a physical token for multifactor authentication). The identification and authentication as well as authorization for the use of user-specific key material can be realized by inserting an interface into the Microsoft Windows NT login mechanism, which can support chained mechanisms (typically used e.g. for integrating Novell Netware services). The login interface is guaranteed to be unencumbered by possible Trojan horses; Windows NT retains multiple desktops, one of these desktops is reserved only for the login and security subsystem. By activating the Secure Attention Sequence (SAS), the current desktop is deactivated and the security desktop is presented. After the security-relevant operation is completed, the previous desktop is again displayed. Use of this mechanism requires the insertion of a custom Graphical Identification and Authentication (GINA) mechanism; in addition, it must be ensured that ancilliary device drivers required for the support of authentication tokens are loaded sufficiently early in the boot sequence since these must be ready by the time the custom GINA is activated by the `WINLOGON` process.

In the implementation discussed here, keys can be provided either from the local key store or by querying servers for public keys; these can either be PGP Public Key Servers (queried using a simplified version of HTTP, the Horowitz Key Protocol), or via LDAP. In either case since only passive retrieval of signed data is required, no elaborate authentication mechanism is necessary. This ensures that protected mail traffic is possible both in the presence of a directory or key server (or sets thereof) for an organization and beyond, and also in situations where no such infrastructure exists. In case PGP keys are used instead of X.509 certificates signed by a certification authority, appropriate security policy elements must identify the desirable level of key authentication (e.g. by endorsement of other users) before a key is used.

E. Transparent TLS/SSL Proxying

Mainly for inbound processing (in some cases such as mobile users, the same may also be true for outbound processing), the MUA may use a SSL/TLS-secured connection to the MTA (see section II-C). This is problematic for a kernel-based system such as the one described here since

normally the requisite key material is (unlike e.g. in the case of an IPSec-based tunnel) not immediately available to the operating system.

To ensure that the processing described in sections III-B and III-C can occur, the blinding introduced by the application-level encryption can be removed by inserting a transparent, mandatory proxy. This proxy also relies on the interception mechanism described in section III-A and detects the initiation of a SSL/TLS session. By acting as the remote endpoint from the perspective of the MUA application and, if required, also initiating another SSL/TLS session with the MTA, protocol transparency is retained. This also requires that the surrounding data stream be adapted since some protocol variations (e.g. the use of certificates with different or extended attributes) may result in changes such as the size of individual messages.

Moreover, the proxying mechanism must ensure that all certificates used for authentication in the proxy are accepted by the MTA and MUA. This can be accomplished by obtaining an X.509 certificate certified by a CA known to these entities, typically from a third party, or by introducing the organization's CA root key into the respective applications. Unfortunately, however, many applications including MUAs do not adequately verify the content of certificates used in any case. Care must therefore be taken (e.g. also in case of many WWW browsers) to avoid third parties use SSL/TLS proxying for man-in-the-middle attacks. For additional details, please refer to an earlier paper [25].

IV. Related Work

A centralized proxy-based e-mail security mechanism was proposed by Brown and Snow [5], which used external MTAs capable of automatic mail encryption and decryption. Several server-based systems have also been developed commercially. For servers, the ssmail system by Bentley *et al.* provides opportunistic (i.e. whenever key material is available) message traffic encryption between individual MTAs [4].

Guard mechanisms were first developed in the form of the the Advanced Command and Control Architecture or ACCAT Guard by Woodward *et al.* at MITRE and Logicon [26], [27], [28] under Navy sponsorship [29], [30] which provided monitoring and sanitization of bidirectional queries and responses between database systems operating at different security levels with human review and was based on the KSOS system for trusted processing; although a version based on a security kernel enforcing enforces the axioms of the Bell-LaPadula model was created, the requirement for explicit downgrading and the inability to determine the semantics of sanitization limited the enforceability of the *-property. Even unidirectional guard mechanisms must take into consideration that malicious code such as Trojan horses are transmitted from low to high security levels;

Denning [31] describes the Trojan horse problem in an automated security guard.

Deployed guards include the Standard WWMCCS Guard [32], [33] providing the means for DoD organizations to extract Secret and less classified data from the Top Secret WWMCCS, which operates in the system-high mode, and to make that data available automatically to users on Secret command and control systems. Other examples of guard systems include MMS developed by Landwehr, Heitmeyer *et al.* [34], [35], [36], [37], [38] and the Standard Mail Guard whose implementation was derived from the LOCK prototype [39]. The NRL pump by Kang and Moskowitz [40] which was later extended to networked systems [41], [42] represents a guard mechanism explicitly intended for limiting covert channels. Davida *et al.* introduced the systematic use of cryptographic mechanisms for the control of information flow between dedicated units in a multilevel environment [43]. Epstein and Monteith, moreover, proposed the use of probabilistic mechanisms for flow control based on information flow signatures [44], [45]; this enables a probabilistic automated downgrading mechanism.

V. Conclusions

In this paper a mechanism for enforcing security policies for electronic mail messaging traffic, particularly automatic mail encryption and electronic signatures, and guard functionality was described. Unlike earlier approaches particularly of automatic mail security, this is enforced at the operating system level of each node, mitigating threats posed by a changing network environment and activity by malware, particularly Trojan horses manipulating user interfaces and extracting secret key material and access codes to this key material. By transparently filtering all relevant inbound and outbound messaging traffic and analyzing known standard protocols using parallel general Büchi automata to obtain adequate performance, neither users behavior nor particularly application programs need to be modified, and user interaction is limited to identification and authentication as well as authentication, all of which occurs as dictated by the active security policy.

Ongoing and future research involves the seamless integration of the mechanisms described here into a security architecture in which the mechanisms discussed represent one instance of security policy-dictated control over operations within distributed systems.

References

[1] A. Adams and M. A. Sasse, "Users Are Not The Enemy," *Communications of the Association for Computing Machinery*, vol. 42, pp. 40–46, Jan. 1999.

[2] A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," in *Proceedings of the 8th USENIX Security Symposium*, (Washington D.C., USA), pp. 169–184, USENIX, Aug. 1999.

[3] S. Wolthusen, "Layered Multipoint Network Defense and Security Policy Enforcement," in *Proceedings from the Second Annual IEEE SMC Information Assurance Workshop, United*

States Military Academy, (West Point, NY, USA), pp. 100–108, IEEE Press, June 2001.

[4] D. Bentley, G. G. Rose, and T. Whalen, "ssmail: Opportunistic Encryption in sendmail," in *Proceedings of the 13th Conference on Systems Administration (LISA-99)*, (Seattle, WA, USA), pp. 1–8, USENIX, Nov. 1999.

[5] I. Brown and C. R. Snow, "A Proxy Approach to e-Mail Security," *Software — Practice and Experience*, vol. 29, pp. 1049–1060, Dec. 1999.

[6] J. Postel, "RFC 821: Simple Mail Transfer Protocol," Aug. 1982. Obsoletes RFC 788.

[7] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker, "RFC 1869: SMTP Service Extensions," Nov. 1995. Obsoletes RFC 1651.

[8] G. Vaudreuil, "RFC 3030: SMTP Service Extensions for Transmission of Large and Binary MIME Messages," Dec. 2000. Obsoletes RFC 1830.

[9] J. Myers and M. Rose, "RFC 1939: Post Office Protocol — Version 3," May 1996. Obsoletes RFC 1725.

[10] M. Crispin, "RFC 2060: Internet Message Access Protocol — Version 4 Rev 1," Dec. 1996. Obsoletes RFC 1730.

[11] T. Berners-Lee, R. Fielding, H. Frystyk, J. Gettys, and J. Mogul, "RFC 2068: Hypertext Transfer Protocol — HTTP/1.1," Jan. 1997.

[12] D. Crocker, "RFC 822: Standard for the Format of ARPA Internet Text Messages," Aug. 1982. Obsoletes RFC 733.

[13] N. Freed and N. Borenstein, "RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies," Nov. 1996. Obsoletes RFC 1521, RFC 1522, RFC 1590. Updated by RFC 2184, RFC 2231.

[14] N. Freed and N. Borenstein, "RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types," Nov. 1996. Obsoletes RFC 1521, RFC 1522, RFC 1590.

[15] K. Moore, "RFC 2047: MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text," Nov. 1996. Obsoletes RFC 1521, RFC 1522, RFC 1590. Updated by RFC 2184, RFC 2231.

[16] N. Freed and N. Borenstein, "RFC 2049: Multipurpose Internet Mail Extension (MIME) Part Five: Conformance Criteria and Examples," Nov. 1996. Obsoletes RFC 1521, RFC 1522, RFC 1590.

[17] Bundesamt für Sicherheit in der Informationstechnik, "Sichere Verwendung von Verschlüsselungs-Plugins in Outlook," Feb. 2002.

[18] C. Newman, "RFC 2595: Using TLS with IMAP, POP3 and ACAP," June 1999.

[19] D. Solomon and M. Russinovich, *Inside Windows 2000*. Bellevue, WA, USA: Microsoft Press, 3rd ed., 2000.

[20] D. B. Andersen, "Windows Sockets 2 Application Provider Interface," tech. rep., Intel Corp., May 1997. Version 2.2.1.

[21] D. B. Andersen, "Windows Sockets 2 Service Provider Interface," tech. rep., Intel Corp., May 1997. Version 2.2.1.

[22] W. Thomas, "Automata on Infinite Objects," in *Handbook of Theoretical Computer Science* (J. van Leeuwen, ed.), vol. B, ch. 4, pp. 133–192, Cambridge, MA, USA: MIT Press, 1990.

[23] M. Arnold, M. Schmucker, and S. D. Wolthusen, *Techniques and Applications of Digital Watermarking and Content Protection*. The Artech House Computer Security Series, Norwood, MA, USA: Artech House, 2003.

[24] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "RFC 2440: OpenPGP Message Format," Nov. 1998.

[25] E. Rademer and S. Wolthusen, "Transparent Access To Encrypted Data Using Operating System Network Stack Extensions," in *Communications and Multimedia Security Issues of the New Century: Proceedings of the IFIP TC6/TC11 Fifth Joint Working Conference on Communications and Multimedia Security (CMS'01)* (R. Steinmetz, J. Dittman, and M. Steinebach, eds.), (Darmstadt, Germany), pp. 213–226, IFIP, Kluwer Academic Publishers, May 2001.

[26] S. R. Ames, Jr. and D. R. Oestreicher, "Design of a Message Processing System for a Multilevel Secure Environment," in *Proceedings of the National Computer Conference*, vol. 47, (Anaheim, CA, USA), pp. 765–771, AFIPS, AFIPS Press, Nov. 1978.

[27] J. P. L. Woodward, "Applications for Multilevel Secure Operating Systems," in *Proceedings of the National Computer Conference*, vol. 48, (New York, NY, USA), pp. 319–328, AFIPS, AFIPS Press, Nov. 1979.

[28] M. A. Padlipsky, K. J. Biba, and R. B. Neely, "KSOS — Computer Network Applications," in *Proceedings of the National Computer Conference*, vol. 48, (New York, NY, USA), pp. 365–371, AFIPS, AFIPS Press, June 1979.

[29] J. Goodwin, G. Mitchell, and P. S. Tasker, "Concept of Operations for Message Handling at CINCPAC," tech. rep., The MITRE Corporation, Bedford, MA, USA, Oct. 1976. Number MTR-3323.

[30] J. D. Tangney, S. R. Ames, Jr., and E. L. Burke, "Security Evaluation Criteria for MMP Message Service Selection," tech. rep., The MITRE Corporation, Bedford, MA, USA, June 1977. Number MTR-3433.

[31] D. E. Denning, *Cryptography and Data Security*. Reading, MA, USA: Addison-Wesley, 1983.

[32] B. Tretick, "Certification and Accreditation Approach for the WWMCCS Guard," in *Proceedings 16th NIST-NCSC National Computer Security Conference*, (Baltimore, MD, USA), pp. 245–252, Oct. 1993.

[33] B. Tretick, "Operational Requirements for Multilevel Security," in *Proceedings 9th Annual Computer Security Applications Conference (ACSAC'93)*, (Orlando, FL, USA), pp. 30–35, IEEE Press, Dec. 1993.

[34] C. E. Landwehr and C. L. Heitmeyer, "Military Message Systems: Requirements and Security Model," tech. rep., Naval Research Laboratory, Washington D.C., USA, Sept. 1982. NRL Memorandum 4925.

[35] C. E. Landwehr, C. L. Heitmeyer, and J. McLean, "A Security Model for Military Message System," *ACM Transactions on Computer Systems*, vol. 2, pp. 198–222, Aug. 1984.

[36] C. L. Heitmeyer and C. E. Landwehr, "Designing Secure Message Systems: The Military Message Systems (MMS) Project," in *Proceedings of the IFIP TC6.5 Working Conference on Computer-Based Message Services*, (Nottingham, UK), pp. 245–255, IFIP, Kluwer Academic Publishers, May 1984.

[37] C. L. Heitmeyer and M. Cornwell, "Specifications for Three Members of the Military Message System (MMS) Family," tech. rep., Naval Research Laboratory, Washington D.C., USA, Mar. 1982. NRL Memorandum 5654.

[38] M. R. Cornwell and A. P. Moore, "Security Architecture for a Secure Military Message System," tech. rep., Naval Research Laboratory, Washington D.C., USA, Apr. 1989. NRL Memorandum 9187.

[39] R. E. Smith, "Constructing a High Assurance Mail Guard," in *Proceedings 17th NIST-NCSC National Computer Security Conference*, (Gaithersburg, MD, USA), pp. 247–253, Oct. 1994.

[40] M. H. Kang and I. S. Moskowitz, "A Pump for Rapid, Reliable, Secure Communications," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, (Fairfax, VA, USA), pp. 118–129, ACM Press, Nov. 1993.

[41] M. H. Kang, I. S. Moskowitz, and D. C. Lee, "A Network Version of the Pump," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy (SOSP '96)*, (Oakland, CA, USA), pp. 144–154, IEEE Press, May 1996.

[42] M. H. Kang, I. S. Moskowitz, and D. C. Lee, "A Network Pump," *IEEE Transactions on Software Engineering*, vol. 22, pp. 329–338, May 1996. Revised from [41].

[43] G. I. Davida, R. A. DeMillo, and R. J. Lipton, "A System Architecture to Support a Verifiable Secure Multilevel Security System," in *Proceedings of the 1980 IEEE Symposium on Security and Privacy (SOSP '80)*, (Oakland, CA, USA), pp. 137–145, IEEE Press, Apr. 1980.

[44] J. Epstein, "Architecture and Concepts of the ARGuE Guard," in *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*, (Phoenix, AZ, USA), pp. 45–54, IEEE Press, Dec. 1999.

[45] E. Monteith, "Genoa TIE, Advanced Boundary Controller Experiment," in *Proceedings 17th Annual Computer Security Applications Conference (ACSAC'01)*, (New Orleans, LA, USA), pp. 74–82, IEEE Press, Dec. 2001.