Enforcing Messaging Security Policies

Jaromir Likavec
Fraunhofer-IGD
Fraunhoferstr. 5
Darmstadt 64283, Germany
likavec@igd.fhg.de

Stephen D. Wolthusen
Fraunhofer-IGD
Fraunhoferstr. 5
Darmstadt 64283, Germany
wolt@igd.fhg.de

Abstract

A system for enforcing messaging security policies for both store and forward and streaming messaging protocols on COTS operating system platforms is described. Messaging protocols are subjected to interception, transformation, and filtering based on dynamically configurable security policies. Transformations include the automatic policy-based application of cryptographic confidentiality, integrity, and authenticity mechanisms and filtering primarily based on Bayesian analysis. The system provides a low cost, fine granularity compartmentalization mechanism for secure environments as well as for sensitive but unclassified environments using COTS operating systems and application programs without affecting user or application behavior in which the mediation of access to key material and messaging provides protection against malware and insider attacks.

1. Introduction

While guard mechanisms between network segments at different levels of classification have proven adequate for interconnecting larger-scale units, unit cost and administrative overhead generally prohibit the use of network and electronic mail guard mechanisms at finer levels of granularity.

Smaller units, together with the frequent need for high-granularity control over information exchange, particularly in coalition environments and during deployments in network-centric environments, however, make it highly desirable to have such controls in place — ideally at the level of granularity an individual's workstation without requiring large, complex, and expensive messaging infrastructure components.

A second factor for such requirements also exists in the civilian sector, namely that while security policies may well dictate the use of encryption, digital signatures, and even time stamps for certain types of communication (e.g. procurement, sensitive interactions with customers), compliance is not enforced but rather lies in the area of responsibility of individual communicating staff members. As the analysis in section 2 shows, this trust is severely misplaced and would indicate a need to use mandatory (guard) mechanisms as well. Unlike in defense applications, however, the balance between security and other requirements must be observed more closely in such environments, which a mandatory messaging security mechanism must take into account.

2. Baseline Analysis

Existing guidelines recommend and draft guidelines require the use of secure messaging (particularly the use of digital signatures) at the authors' installation even for unclassified messaging traffic.

The guidelines are published internally to the organization flanked by awareness measures.

For email messaging security, two infrastructure mechanisms are provided. For sporadic communications with entities that do not or cannot support public key infrastructures, an OpenPGP infrastructure including key services are provided. Conversely, for more structured communication relations with entities that use public key infrastructures based on open standards (e.g. the ITU-T X.500 series, LDAP, OCSP, and S/MIME), PKI services, including key servers and a local registration authority are provided at the installation. A number of mail user agents are supported on each supported operating system platform for both secure messaging infrastructures, giving each staff member access to secure messaging.

Following an informal survey among senior staff in which low compliance levels were acknowledged, a detailed compliance analysis was performed. To this end, all unclassified (E)SMTP messaging traffic outbound from the installation was intercepted and monitored for the pres-

ence of digital signatures and encryption using the following protocols:

- S/MIME
- (Open)PGP
- PGP/MIME

The analysis was performed over a time of 11 days and intercepted a total of 14356 messages. Of these, 753 were generated using one of the above cryptographic mechanisms, for a total of 5.24%. Figure 1 provides a breakdown of the mechanism types observed versus the total number of messages over time.

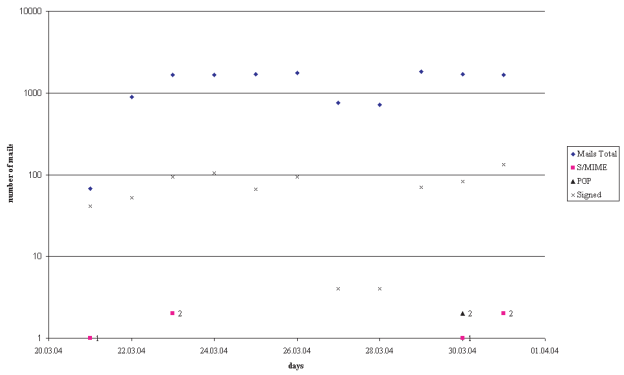

Figure 1. Compliance baseline analysis

3. Usability Considerations

The results outlined above indicate that, despite the presence of extensive technical facilties, actual use of secure messaging facilities is sharply limited.

A qualitative survey of reasons given for noncompliance included the following:

- Lack of suitable key material by recipient (encryption)
- Tendency to forget subtasks not immediately relevant to task at hand (encryption/digital signatures)
- Secure messaging interface is too complex (encryption/digital signatures)
- Additional steps required for secure messaging limit productivity (encryption/digital signatures)

Most of these reasons have been observed in other application areas for cryptographic tools [1, 32].

Given particularly the number of messages using digital signatures in relation to the number of encrypted messages sent, the following main observations can be made:

- Services that can be employed following a one-time configuration without requiring subsequent interactions have a compliance two orders of magnitude larger than those requiring intervention for each use
- Regardless of mail user agent (MUA) employed, concerns about interface complexity were raised.

Based on the above observations and measurements, the following design principles can be formulated for secure messaging:

Non-Bypassability Application-based messaging security mechanisms can be bypassed or disabled intentionally or unintentionally by misconfiguration (including lack of initialization). In addition, application programs may be employed that lack support for required protocols.

To avoid both types of flaws, messaging security must be embedded at a level that is independent of application programs and cannot be bypassed by application programs.

Automation All operations required to enforce applicable security policies must be automated to the largest extent possible, i.e. only require user intervention if necessary for security purposes (e.g. for identification and authentication).

Protection of Key Material and Authentication In software systems typically used for securing unclassified message traffic, key material used for sealing and signing messages as well as access paths to such key material are accessible to processes with user credentials.

Threats resulting from this configuration include unauthorized access to key material and messages by Trojan horses or other processes with access permissions to key material and the unauthorized sending of messages on behalf of a third party.

Access to key material must therefore be subject to mediation and be configurable to require positive identification and authentication for performing operations as dictated by applicable security policies.

The remainder of this paper describes a mail guard architecture and implementation that takes the above design principles into consideration and which achieves the twin goals of improving the assurance of the and the usability messaging process at the same time.

4. Guard Mechanisms

The goals set forth in section 3 (modulated by security policy requirements) can be achieved by embedding interception and filtering mechanisms into the operating system at all applicable layers in the network and application stack.

Depending on the security policy to be enforced, messages or message streams can be subjected to arbitrary transformations (e.g. substituting a message with a message encrypted for the intended recipient or recipients or discarding the message altogether).

While the semantics of the filtering process are theoretically platform-independent, this is not the case for the interception mechanism itself. Moreover, the architecture of the underlying operating system can limit the set of possible transformations for the filtering process (see section 4.3). To reflect this, the following discussion concentrates on a reference implementation based on the Microsoft Windows NT operating system family[1] Section 4.1 summarizes the interception mechanism architecture; sections 4.2 and 4.3 discuss specific requirements for store-and-forward and stream-based messaging protocols, respectively. Section 4.4 describes the preprocessing step required for performing the content-based filtering as well as cryptographic transformations using the example of Internet (SMTP/MIME) mail messaging. Finally, section 4.5 discusses the content filtering processes applied to the message data itself (cryptographic transformations are discussed in [33]).

4.1. Implementation Framework

The basic implementation architecture for providing security functionality in the network protocol stack of the Microsoft Windows NT operating system family has been described earlier [27, 34] and is discussed here in summarized form.

Unlike the other components such as file system handling, the networking mechanisms provided by the Microsoft Windows NT operating system family do not share a common abstraction for all supported types of network communication.

Therefore, in addition to multiple environmental subsystems providing different access mechanisms to network communication subsystems, there exist several conceptually different networking application programming interfaces. With exceptions described in detail in [34], the network architecture of the Microsoft Windows NT family consists of a number of layers, depicted in figure 2.

At the lowest level is the physical device. Access to individual devices is regulated by the hardware abstraction layer (HAL). Network device drivers are generally realized as NDIS (Network Driver Interface Specification) modules consisting of the generic NDIS library and the device--specific NDIS miniport drivers; the library fully encapsulates the miniport drivers.

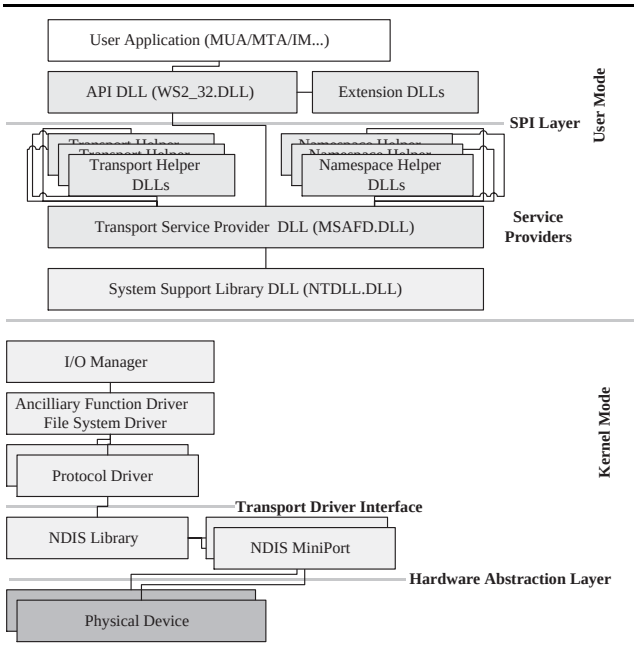

Figure 2. Microsoft Windows NT operating system family network protocol stack

Accessing the NDIS library is the TDI (Transport Driver Interface) mechanism. This itself consists of transports (or protocol drivers), supporting the various transport mechanisms such as NetBEUI and TCP/IP, and TDI clients which provide services for sockets and NetBIOS calls. None of these modules can be called directly from applications since they are protected kernel mode interfaces. It is in this layer that the translation between the file-oriented abstraction presented to upper level protocols and ultimately application programs and the general packet-based I/O architecture of the Windows NT operating system family occurs.

Upper-level APIs such as NetBIOS and Windows Sockets are subsequently implemented at the user level and must use the aforementioned interface layers.

Since multiple access paths for network transmissions exist that can bypass some or all protocol layers above the NDIS layer, it is necessary to perform interception at the NDIS layer itself. However, the NDIS layer is not provided with information on processes originating network traffic and can only observe individual fragments of information flows. To assist in the proper association of network flows with processes and therefore ultimately the subjects of se-

1 including the NT 3.51, 4.0, 2000, XP, and 2003 releases unless otherwise noted.

curity policies, higher protocol layers such as TDI must also be instrumented as supplementary information sources.

The results of this interception mechanism are (both inbound and outbound) either raw network data streams (e.g. IP packets, corresponding to ISO/OSI layer 3/4 data) or payload data streams (corresponding to ISO/OSI layer 5 and above).

4.2. Store and Forward Protocols

The application of arbitrary transformations to messages processed by store and forward protocols is generally straightforward; a suitable heuristic must detect the requisite protocol data units and supplemental information (e.g. the beginning of a protocol transaction on a well-known IP port) and store the full message sent subsequently before forwarding it to the further processing steps discussed below.

However, several complications arise in this case as well. First, since the upper-level abstraction for network traffic is based on a file paradigm, upper-layer protocols can and do keep track of the relative position in the virtual file represented by the network stack. It is therefore necessary to adjust these virtual file positions in translating between the abstraction layers in the network protocol stack.

The second complication is the need for protocol analysis to determine that a store and forward protocol is indeed used. To this end, a set of timed[2] general Büchi automata $\mathbf{A} = \{(Z^i, \mapsto^i, l^i, Z_0^i, \mathcal{F}^i)\}$ 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 $\mathcal{F}^i \subseteq \mathbb{P}(Z^i)$ the set of sets of accepting states [31], is used. The determination of an accepting state (heuristically, since it cannot be ruled out that multiple accepting states exist or that an accepting state is a prefix of another) indicating such a protocol requires that a number of protocol data units has been set or an information exchange has been established. The interception mechanism must admit such flows and therefore retain state information on such prefixes before shunting network flow to the actual filtering mechanism to be able to reconstruct the full exchange of protocol data units that may be required to perform filtering operations.

The special case of HTTP-based messaging has been discussed in an earlier report [27].

4.3. Stream-Based Protocols

For stream-based protocols, transformations must be performed in-line and are therefore typically limited to transformations that do not change the length of each stream element protocol data unit.

Most of the above limitations are dependent on the application protocol in use, necessitating higher level protocol analysis and therefore also reconstruction of protocol elements. This imposes a severe burden for protocols that are both proprietary and experience frequent version changes as is the case for some instant messaging protocols.

More importantly, unlike store and forward protocols, streaming protocols (which include both multimedia protocols that are sensitive to delay and jitter imposed on e.g. audio transmissions and basic network protocols such as the Telnet protocol) are sensitive to the delays imposed by interception and filtering and may either degrade or fail in the presence of excessive delays.

Moreover, such protocols may also use time (e.g. delay between protocol data units) as an implicit element of protocol exchanges and therefore exhibit different semantics when exposed to such excessive delays.

The heuristics for interception and filtering must therefore also ensure that no (protocol-specific) time bounds are exceeded; this is ensured by the use of parallel timed Büchi automata. In case a bound is exceeded without reaching a positive protocol identification or performing a transformation step, the security policy may e.g. dictate to terminate the ongoing data stream in its entirety.

4.4. Preprocessing

Transformations required for content filtering as well as some additional transformations (e.g. encryption) require knowledge of the encoding syntax and possible additional transformations applied on each semantic layer to be analyzed and processed.

In the case of standard Internet mail, a large number of possible encoding formats has emerged since the first standard definition [4] mainly related to the transport of multimedia data types.

The MIME (Multimedia Mail Extensions) standards [9, 10, 24, 11] 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 [11]. 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. [4] 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[3] the definition of a content type for each message or

2 for a rationale for using the timed variant see section 4.3.

3 The absence of a content type definition is assumed to denote a 7 Bit

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 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 actual message (part) body.

Some protocols can also trigger the use of other protocols and communications; this is e.g. also the case for Internet mail in case of MIME external message bodies but is most prominently present in HTTP-based protocols and instant messaging protocols and must therefore be either blocked or also intercepted and transformed in accordance with applicable security policies; for a discussion of several of these issues refer to [33] and [27].

4.5. Analytical Processes

Keyword-based content filtering has been used extensively in guard architectures [16, 28, 29]; other approaches include techniques such as metrics for precision and recall based on feature similarity [23].

To address several threats, however, additional probabilistic techniques are required. One such threat is also commonly found in unsolicited commercial electronic mail, namely spelling errors and homophones (particularly from multilingual character sets that cannot be distinguished visually from one another). Such variations may be inserted both deliberately as well as inadvertently but must be identified in either case[4]. An efficient mechanism for implementing a heuristic to capture such homophones and variations is the use of Bayesian networks [26, 12, 17];

An additional level of protection (partially explored by Monteith *et al.* [23]) is the inverse use of techniques found in information retrieval. Bayesian networks can also be employed here, in this case in the form of Bayesian belief networks (directed acyclical graphs in which vertices represent random variables and edges represent relationships between linked variables where the strength of influences are expressed as conditional (Bayesian) probabilities.

Concepts can now be expresed as vectors of random variables over which various metrics, particularly similarity metrics can be defined [6, 7]. This permits the definition of security policies that heuristically attempt to identify not only sensitive terms and expressions but also the use of related concepts and circumlocutions and take appropriate actions based on this (e.g. notifying administrative or security personnel).

5. Mediation Mechanisms

Threats to the security and integrity of the messaging process on the sending (and to a lesser extent on the receiving) side include malicious users with access to a workstation acting on behalf of the user without his consent or knowledge, and also various types of malware.

Malware can suborn the messaging system in several ways. It may send messages on behalf of the user by directly sending network traffic or by using general messaging interfaces without assuming a legitimate user identity on the host operating system, directly spoofing addresses.

In addition, malware acting within the process context of a user (e.g. a Trojan horse) can use the regular messaging mechanisms, usurping the identity of the user and, moreover, potentially leaving audit trails indicating that a user has indeed performed a given action.

On a GUI-based system, an additional threat comes from processes feeding synthetic GUI actions to simulate user behavior. As in the previously noted case, this also will leave plausible audit trails implicating a legitimate user.

The latter threats are also present if cryptographic mechanisms are used. However, an additional threat exists that renders client-side cryptographic mechanisms largely moot in that malware that has access to the regular graphical user interface (e.g. window message stream) or even the client process memory space can compromise both key material and the authentication data such as passwords used to nominally secure the key material. It is therefore straightforward for malware to both decrypt (or forward encrypted messages together with the required key material) messages and to en-

ISO 646 text

4 An inside threat is always capable of devising an encoding scheme to circumvent such an approach; the ability to detect such changes automatically in all cases would correspond to solving NP-hard problems, e.g. Post's correspondence problem.

crypt and particularly to cryptographically sign messages on behalf of the user.

Particularly in jurisdictions where digital signatures carry evidentiary value, the latter scenarios present a significant threat.

To counter this threat the mail guard architecture discussed here, user entities including processes belonging to a user are not permitted to access key material directly to avoid the leaking of such material to processes acting on behalf of the user or third parties otherwise obtaining control over a process (e.g. accessing an unlocked workstation in the absence of a legitimate user).

To ensure this separation, all cryptographic transformations are performed by the operating system on behalf of the user in accordance with applicable security policies; typical policies include the automatic application of digital signatures to outbound message traffic or mandatory encryption for communication between certain parties or based on the sensitivity of files accessed by the message sending process prior to commencing communication.

To this end, the operating system's trusted path mechanism is used by the implementation to provide a communication channel to the operating system over which the identification, authentication, and authorization (IA^2) is to be transmitted.

In case of the Microsoft Windows NT operating system family, this is achieved by inserting a chained element into the graphical identification and authentication (GINA), and triggering the display of this element whenever a IA^2 datum is required (see figure 3).

The feature[5] implements a switch to a desktop which cannot be accessed for input or output by application programs that is always triggered by entering the system attention sequence[6]. The user can confirm the authenticity of this request by initiating the non-bypassable system attention sequence (SAS), for which the base operating system guarantees that it cannot be intercepted or simulated by application programs. Any impostor application displaying a request similar to that shown in figure 3 is therefore disconnected once the SAS is issued, and only the legitimate request can be displayed. Moreover, the operating system also ensures that no application program can intercept communication with the display once the SAS has been issued. This is of particular interest for input that is e.g. used to unlock key material on behalf of the user as well as for other actions that require the immediate confirmation of a user's presence and expressed will to continue with an operation.

5 A requirement that originated in the TCSEC B2 class but which was nevertheless included in the Windows NT design.

6 However, it should be noted that one must still assume that none of the device driver or other kernel components involved in I/O are tainted or compromised; trusted computing extensions to the basic system platform and hardware proposed by the Trusted Computing Group may in the future provide additional assurance for such circumstances.

Figure 3. Authorization of operations via trusted path

6. Related Work

Guard mechanisms for messaging and subsequently also for other types of data (e.g. database accesses, geospatial intelligence data) 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 [2, 35, 25, 14, 30].

The ACCAT Guard 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.

Other examples of guard systems include MMS developed by Landwehr, Heitmeyer *et al.* [21, 22, 16, 15, 3] and the Standard Mail Guard whose implementation was derived from the LOCK prototype [28, 29]. In addition, a number of products (some also available commercially and intended for civilian use) provide centralized guard as well as encryption functionality pursuant to the U.S. Department of Defense mail guard for high robustness environments Common Criteria (ISO 15408) protection profile, e.g. the XTS-400 Standard Automated Guard Environment by DigitalNet Government Solutions.

The NRL pump by Kang and Moskowitz [18] which was later extended to networked systems [20, 19] 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 [5]. Epstein and Monteith, moreover, proposed the use of probabilistic mechanisms for flow control based on information flow signatures [8, 23]; this enables a probabilistic

automated downgrading mechanism.

For a survey of guard policies and mechanisms as applicable to SECRET and below networks with an emphasis on alliance networks, refer to [13].

7. Conclusions

This paper has, starting from an analysis of operational experience with compliance to messaging security policies for unclassified messaging traffic, identified several contributing factors to strong noncompliance. Based on this analysis, design principles were formulated and an implementation was described that addressed said issues.

The system presented provides the means for enforcing messaging security policies particularly within secured enclaves against malware-induced unauthorized behavior as well as protection against casual security lapses (e.g. leaving a workstation unlocked temporarily, permitting an unauthorized individual to engage in messaging traffic on behalf of another individual). Given the requirement for instrumentation of operating systems of limited trustworthiness, this does not permit the substitution of boundary mail guards (e.g. products conforming to the Common Criteria protection profile for U.S. Department of Defense mail guard for high robustness environments, see section 6), however, and must be considered strictly as a supplementary mechanism for providing low cost, fine granularity compartmentalization of secure environments using COTS operating systems and application programs.

References

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

[2] 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*, volume 47, pages 765–771, Anaheim, CA, USA, Nov. 1978. AFIPS, AFIPS Press.

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

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

[5] 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)*, pages 137–145, Oakland, CA, USA, Apr. 1980. IEEE Press.

[6] B. de Araujo Neto Ribeiro. *Approximate Answers in Intelligent Systems*. PhD thesis, University of California at Los Angeles, Los Angeles, CA, USA, 1995.

[7] B. de Araujo Neto Ribeiro. A Belief Network Model for IR. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, Zurich, Switzerland, Aug. 1996. ACM Press.

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

[9] 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.

[10] 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.

[11] 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.

[12] A. Gelman, J. B. Carlin, and H. S. Stern. *Bayesian Data Analysis*. Texts in Statistical Science. Chapman & Hall, London, UK, 2nd edition, 2003.

[13] T. Gibson. An Architecture for Flexible Multi-Security Domain Networks. In *Proceedings of the 2001 ISOC Network and Distributed System Security Symposium*, San Diego, CA, USA, Feb. 2001. Internet Society.

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

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

[16] 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*, pages 245–255, Nottingham, UK, May 1984. IFIP, Kluwer Academic Publishers.

[17] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Statistics for Engineering and Information Science. Springer-Verlag, Heidelberg, Germany, 2001.

[18] 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*, pages 118–129, Fairfax, VA, USA, Nov. 1993. ACM Press.

[19] M. H. Kang, I. S. Moskowitz, and D. C. Lee. A Network Pump. *IEEE Transactions on Software Engineering*, 22(5):329–338, May 1996. Revised from [20].

[20] 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)*, pages 144–154, Oakland, CA, USA, May 1996. IEEE Press.

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

[22] C. E. Landwehr, C. L. Heitmeyer, and J. McLean. A Security Model for Military Message System. *ACM Transactions on Computer Systems*, 2(3):198–222, Aug. 1984.

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

[24] 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.

[25] M. A. Padlipsky, K. J. Biba, and R. B. Neely. KSOS — Computer Network Applications. In *Proceedings of the National Computer Conference*, volume 48, pages 365–371, New York, NY, USA, June 1979. AFIPS, AFIPS Press.

[26] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann Series In Representation And Reasoning. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1988.

[27] E. Rademer and S. Wolthusen. Transparent Access To Encrypted Data Using Operating System Network Stack Extensions. In R. Steinmetz, J. Dittman, and M. Steinebach, editors, *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)*, pages 213–226, Darmstadt, Germany, May 2001. IFIP, Kluwer Academic Publishers.

[28] R. E. Smith. Constructing a High Assurance Mail Guard. In *Proceedings of the 17th National Computer Security Conference*, pages 247–253, San Diego, CA, USA, Oct. 1994.

[29] R. E. Smith. Cost Profile of a Highly Assured, Secure Generating System. *ACM Transactions on Information and System Security*, 4(1):72–101, Feb. 2001.

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

[31] W. Thomas. Automata on Infinite Objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–192. MIT Press, Cambridge, MA, USA, 1990.

[32] 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*, pages 169–184, Washington D.C., USA, Aug. 1999. USENIX.

[33] S. Wolthusen. A Distributed Multipurpose Mail Guard. In *Proceedings from the Fourth Annual IEEE SMC Information Assurance Workshop, United States Military Academy*, pages 258–265, West Point, NY, USA, June 2003. IEEE Press.

[34] S. D. Wolthusen. Tempering Network Stacks. In *Proceedings of the NATO RTO Symposium on Adaptive Defence in Unclassified Networks*, Toulouse, France, Apr. 2004. NATO Research and Technology Organization.

[35] J. P. L. Woodward. Applications for Multilevel Secure Operating Systems. In *Proceedings of the National Computer Conference*, volume 48, pages 319–328, New York, NY, USA, Nov. 1979. AFIPS, AFIPS Press.