Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 17 May 2015

L. Cailleux
DGA MI
C. Bonatti
IECA

17 November 2014

Securing Header Fields with S/MIME
draft-cailleux-secure-headers-07

Abstract

This document describes how the S/MIME protocol can be extended in order to secure message header fields. This technology provides security services such as data integrity, non-repudiation and confidentiality. This extension is referred to as 'Secure Headers'.

Status of this Memo

This Internet-Draft will expire on 26 January 2015.

Copyright Notice

Table of Contents

1. Introduction

S/MIME [RFC 5751] standard defines a data encapsulation format for the achievement of end to end security services such as integrity, authentication, non-repudiation and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields.

S/MIME provides an alternative solution to secure header fields. "The sending client MAY wrap a full MIME [RFC 2045] message in a message/rfc822 wrapper in order to apply S/MIME security services to header fields". However, the S/MIME solution doesn't provide any guidance regarding what subset of message header fields to secure, procedures for clients to reconcile the "inner" and "outer" headers, or procedures for client interpretation or display of any failures.

Several other security standards supplement S/MIME features, but fail to address the target requirement set of this draft. Such other security standards include DKIM [RFC 6376], STARTTLS [RFC 3207], TLS with IMAP [RFC 2595], and an internet draft referred to as PROTECTED HEADERS. An explanation of what these services accomplish and why they do not solve this problem can be found in subsequent sections.

The goal of this document is to define end to end secure header fields mechanisms compliant with S/MIME standard. This technique is based on the signed attribute fields of a Cryptographic Message Syntax (CMS) [RFC 5652] signature.

2. Terminology and conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

The terms Message User Agent (MUA), Message Submission Agent (MSA) and Message Transfer Agent (MTA) terms are defined in Email architecture document [RFC 5598].

The term Domain Confidentiality Authority (DCA) is defined in the S/MIME Domain Security specification [RFC 3183].

End-to-end Internet Mail exchanges are performed between message originators and recipients.

The term "message header fields" is described in [RFC 5322]. A header field is composed of a name and a value.

Secure Headers technology uses header fields statuses required to provide a confidentiality service toward message headers. The following three terms are used to describe the field statuses:

- Duplicated (the default status). When this status is present or if no status is specified, the signature process embeds the header field value in the digital signature, but the value is will also be present in the message header fields.

- Deleted. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process deletes this field from the message to preserve its confidentiality.

- Modified. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process modifies the value of the header field in the message. This preserves confidentiality and informs a receiver's non-compliant MUA that secure headers are being used. New values for each field might be configured by the sender (i.e., "This header is secured, use a compliant client").

The term "non-repudiation" is used throughout this document in deference to the usage in the S/MIME Message Specification [RFC 5751]. It is recognized that this term carries with it much baggage, and that there is some disagreement as to it's proper meaning and usage. However, in the context of this document the term merely refers to one of a set of possible security services that a conforming implementation might be able to provide. This document specifies no normative requirements for non-repudiation.

3. Context

Over the Internet, email usage has grown and today represents a fundamental service. Meanwhile, continually increasing threat levels are motivating the implementation of security services.

Historically, SMTP [RFC 5321] and IMF [RFC 5322] don't provide, by default, security services. The S/MIME standard [RFC 5751] was published in order to encompass these needs. S/MIME defines a data encapsulation format for the provision of end to end security services such as integrity, authentication, non-repudiation and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields. In order to protect message header fields (for instance, the "Subject", "To", "From" or customized fields), several solutions exist.

S/MIME defines an encapsulation mechanism, chapter 3.1: "The sending client may wrap a full MIME message in a message/rfc822 wrapper in order to apply S/MIME security services to these header fields. It is up to the receiving client to decide how to present this inner header along with the unprotected outer header". However, some use cases are not addressed, especially in the case of message encryption. What happens when header fields are encrypted? How does the receiving client display these header fields? How can a subset of header fields be secured? S/MIME doesn't address these issues.

Some partial header protection is provided by the S/MIME Certificate Handling specification [RFC 5750]. "Receiving agents MUST check that the address in the From or Sender header of a mail message matches an Internet mail address, if present, in the signer's certificate, if mail addresses are present in the certificate". In some cases this may provide assurance of the integrity of the From or Sender header values. However, the RFC 5750 solution only provides a matching mechanism between email addresses, and provides no protection to other header fields.

Other security standards (introduced below) exist such as DKIM, STARTTLS and TLS with IMAP but meet other needs (signing domain, secure channels...).

STARTTLS and TLS with IMAP provide secure channels between components of email system (MUA, MSA, MTA...) but end to end integrity cannot be guaranteed.

DKIM defines a domain-level authentication framework for email. While this permits integrity and origination checks on message header fields and the message body, it does for a domain actor (usually the SMTP service or equivalent) and not for the entity that is sending, and thus signing the message. (Extensions to DKIM might be able to solve this issue by authenticating the sender and making a statement as part of the signed message headers of this fact.) DKIM is also deficient for our purposes as it does not provide a confidentially service.

An internet draft referred to as Protected Headers (PRHDRS) has been proposed. Mechanisms described in this draft are the following. "A digest value is computed over the canonicalized version of some selected header fields. This technique resembles header protection in DKIM. Then the digest value is included in a signed attribute field of a CMS signature". This specification doesn't address all conceivable requirements as noted below. If the protected header field has been altered, the original value cannot be determined by the recipient. In addition, the encryption service cannot provide confidentiality for fields that must remain present in the message header during transport.

This document proposes a technology for securing message header fields. It's referred to as Secure Headers. It is based on S/MIME and CMS standards. It provides security services such as data integrity, confidentiality and non-repudiation of sender. Secure Headers is backward compatible with other S/MIME clients. S/MIME clients who have not implemented Secure Headers technology need merely ignore specific signed attributes fields in a CMS signature (which is the default behavior).

4. Mechanisms to secure message header fields

Secure Headers technology involves the description of a security policy. This policy MUST describe a secure message profile and list the header fields to secure. How this security policy is agreed or communicated is beyond the scope of this document.

Secure headers are based on the signed attributes field as defined in CMS. The details are as follows. The message header fields to be secured are integrated in a structure (secure header structure) which is encapsulated in the signed attributes structure of the SignerInfo object. There is only one value of HeaderFields encoded into a single

SignedAttribute in a signature. See Appendix A for an example. For each header field present in the secure signature, a status can be set. Then, as described in chapter 5.4 of CMS, the message digest calculation process computes a message digest on the content together with the signed attributes. Details of the signature generation process are described in chapter 4.5.1 of this document.

Verification of secure header fields is based on signature verification process described in CMS. At the end of this process, a comparison between the secure header fields and the corresponding message header fields is performed. If they match, the signature is valid. Otherwise, the signature is invalid. Details of the signature verification process are described in chapter 4.5.2 of this document.

Non-conforming S/MIME clients will ignore the signed attribute containing the secure headers structure, and only perform the verification process described in CMS. This guarantees backward compatibility.

Secure headers provide security services such as data integrity, non-repudiation and confidentiality.

For different reasons (e.g., usability, limits of IMAP [RFC 3501]), encryption and decryption processes are performed by a third party. The third party that performs these processes is referred to in Domain Security specification as a "Domain Confidentiality Authority" (DCA). Details of the encryption and decryption processes are described in chapters 4.6.1 and 4.6.2 of this document.

The architecture of Secure Headers is presented below. The MUA performs the signature generation process (C) and signature verification process (F). The DCA performs the message encryption process (D) and message decryption process (E). The encryption and decryption processes are optional.

```
        A Domain                             B Domain
  +-----------------------+            +-----------------------+

  +------+          +------+           +------+          +------+
  | MUA | -------> | DCA | ----------> | DCA |--------> | MUA |
  |  C  |          |  D  |           |  E  |          |  F  |
  +------+          +------+           +------+          +------+
        SignedMsg       EncryptedMsg        SignedMsg
```

Figure 1: Architecture of Secure Headers

4.1. ASN.1 syntax of secure header fields

ASN.1 notation [ASN1-88] of secure header structure is the follow:

```
 SecureHeaderFields ::= SET {
    canonAlgorithm Algorithm,
    secHeaderFields HeaderFields }

 id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::=
    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-9(9) smime(16) id-aa(2) secure-headers (to be
    defined) }

 Algorithm ::= ENUMERATED {
    canonAlgorithmSimple(0),
    canonAlgorithmRelaxed(1) }

 HeaderFields ::= SEQUENCE SIZE (1..max-header-fields) OF
    HeaderField

 max-header-fields INTEGER ::= MAX

 HeaderField ::= SEQUENCE {
    field-Name HeaderFieldName,
    field-Value HeaderFieldValue,
    field-Status HeaderFieldStatus DEFAULT duplicated }
```

```
HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
     -- This description matches with the description of
     -- field name in the chapters 2.2 and 3.6.8 of RFC 5322

HeaderFieldValue ::= UTF8String
     -- This description matches with the description of
     -- field body in the chapter 2.2 of RFC 5322 as
     -- extended by chapter 3.1 of RFC 6532.

HeaderFieldStatus ::= INTEGER {
   duplicated(0), deleted(1), modified(2) }
```

4.2. Secure header fields length and format

This specification requires MUA security capabilities in order to process well formed headers, as specified in IMF. Notice that it includes long header fields and folded header fields.

4.3. Canonization algorithm

During a message transfer through a messaging system, some components might modify headers (i.e., space adding or deletion, lowercase/uppercase rewriting...). This might lead to header fields comparison mismatch. This emphasizes the need of a conversion process in order to transform data to their canonical form. This process is named canonization process.

Two canonization algorithms are considered here, according to DKIM specification [RFC 6376], chapter 3.4. The simple algorithm doesn't allow any modification whereas the relaxed algorithm accepts slight modifications like spaces replacement or line reformatting. Given the scope of this document, canonization mechanisms only involve header fields.

Implementations SHOULD use the relaxed algorithm to promote interoperability with non-conforming SMTP products.

4.4. Header fields statuses

Header fields statuses are necessary to provide a confidentiality service toward message headers. In this specification, the confidentiality of header fields is provided by the DCA. This point is described in chapter 4. The DCA performs the message encryption process and message decryption process and these processes are described in details in the chapters 4.6.1 and 4.6.2. Although header fields statuses are embedded in the signature, the signature processes (generation and verification) ignore them. The header field status defaults to duplicated. If the header field is confidential, the header field status MUST be either deleted or modified.

4.5. Signature Process

4.5.1. Signature Generation Process

During the signature generation process, the sender's MUA MUST embed the SecureHeaderFields structure in the signed attributes, as described in CMS. SecureHeaderFields structure MUST include a canonization algorithm.

The sender's MUA MUST have a list of header fields to secure, statuses and a canonization algorithm, as defined by the security policy.

Header fields (names and values) embedded in signed attributes MUST be the same as the ones included in the initial message.

If different headers share the same name, all instances MUST be included in the SecureHeaderFields structure.

If multiple signatures are used, as explained in CMS and MULTISIGN [RFC 4853] specifications, SecureHeaderFields structure MUST be the same in each SignerInfos object.

If a header field is present and its value is empty, HeaderFieldValue MUST have a zero-length field-value.

Considering secure headers mechanisms, the signature generation process MUST perform the following steps:

1) Select the relevant header fields to secure. This subset of headers is defined according the security policy.

2) Apply the canonization algorithm for each selected header field.

3) Complete the following fields in SecureHeaderFields structure according to the initial message: HeaderFieldName, HeaderFieldValue, HeaderFieldStatus.

4) Complete the algorithm field according to the canonization algorithm configured.

5) Embed the SecureHeaderFields structure in the signed attributes of the SignerInfos object.

6) Compute the signature generation process as described in CMS, chapter 5.5

4.5.2. Signature verification process

During the signature verification process, the receiver's MUA compares header fields embedded in the SecureHeaderFields structure with those present in the message. For this purpose, it uses the canonization algorithm identified in the signed attributes. If a mismatch appears during the comparison process, the receiver's MUA MUST invalidate the signature. The MUA MUST display information on the validity of each header field. It MUST also display the values embedded in the signature.

The receiver's MUA MUST know the list of mandatory header fields in order to verify their presence in the message. If a header field defined in a message is in the secure header list, it MUST be included in the SecureHeaderFields structure. Otherwise, the receiver's MUA MUST warn the user that a non-secure header is present.

Considering secure headers mechanisms, the signature verification process MUST perform the following steps:

1) Execute the signature verification process as described in CMS, chapter 5.6. If the signature appears to be invalid, the process ends. Otherwise, the process continues.

2) Read the type of canonization algorithm specified in SecureHeaderFields structure.

3) For each field present in the signature, find the matching header in the message. If there is no matching header, the verification process MUST warn the user, specifying the missing header name. The signature is tagged as invalid. Note that any headers fields encrypted as per section 4.6 (i.e., status of "deleted" or "modified") have been are already restored by the DCA when the signature verification process is performed by the MUA.

4) Compute the canonization algorithm for each header field value in the message. If the simple algorithm is used, the steps described in DKIM, chapter 3.4.1, are performed. If the relaxed algorithm is used, the steps described in DKIM, chapter 3.4.2, are performed.

5) For each field, compare the value stored in the SecureHeaderFields structure with the value returned by the canonization algorithm. If values don't match, the verification process MUST warn the user. This warning MUST

mention mismatching fields. The signature is tagged as invalid. If all the comparisons succeed, the verification process MUST also notify the user (i.e., using an appropriate icon).

6) Verify that no secure header has been added to the message header, given the initial fields. If an extra header field has been added, the verification process MUST warn the user. This warning MUST mention extra fields. The signature is tagged as invalid. This step is only performed if the sender and the recipient share the same security policy.

7) Verify that every mandatory headers in the security policy and present in the message are also embedded in the SecureHeaderFields structure. If such headers are missing, the verification process MUST warn the user and indicate the names of the missing headers.

The MUA MUST display features for each secure header field (name, value and status) and canonization algorithm used.

4.6. Encryption and Decryption Processes

Encryption and decryption operations are not performed by MUAs. This is mainly justified by limitations of existing email delivery protocols, for example IMAP. The solution developed here relies on concepts explained in Domain Security specification, chapter 4. A fundamental component of the architecture is the Domain Confidentiality Authority (DCA). Its purpose is to encrypt and decrypt messages instead of (respectively) senders and receivers.

4.6.1. Encryption Process

All the computations presented in this chapter MUST be performed only if the following conditions are verified:

- The content to be encrypted MUST consist of a signed message (application/pkcs7-mime with SignedData or multipart/signed) as shown in S/MIME specification, chapter 3.4.

- A SecureHeaderFields structure MUST be included in the signedAttrs field of the SignerInfo object of the signature.

All the mechanisms described below MUST start at the beginning of the encryption process, as explained in CMS. They are performed by the sender's DCA. The following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. Extraction of the field status;

   1.1 If the status is Duplicated, the field is left at its existing value.

   1.2 If the status is Deleted, the header field (name and value) is removed from the message. Mandatory header fields specified in [RFC 5322] MUST be kept.

   1.3 If the status is Modified, the header value is replaced by a new value, as configured in the DCA.

4.6.2. Decryption Process

All the computations presented in this chapter MUST be performed only if the following conditions are verified:

- The decrypted content MUST consist of a signature object or a multipart object, where one part is a detached signature, as shown in S/MIME specification, chapter 3.4.

- A SecureHeaderFields structure MUST be included in the SignerInfo object of the signature.

All the mechanisms described below MUST start at the end of the decryption process, as explained in CMS. They are executed by the receiver's DCA. The following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. If the status is Duplicated, the field is left at its existing value.

2. If the status is Deleted, the DCA MUST write a header field (name and value) in the message. This header MUST be compliant with the information embedded in the signature.

3. If the status is Modified, the DCA MUST rewrite a header field in the message. This header MUST be compliant with the SecureHeaderFields structure.

5. Case of triple wrapping

Secure Headers mechanisms MAY be used with triple wrapping, as described in ESS [RFC 2634]. In this case, a SecureHeaderFields structure MAY be present in the inner signature, in the outer signature, or both. In the last case, the two structure SecureHeaderFields MAY differ. One MAY consider the encapsulation of a header field in the inner signature in order to satisfy confidentiality needs. On the contrary, an outer signature encapsulation MAY help for delivery purpose. Sender's MUA and receiver's MUA must have a security policy for triple wrapping. This security policy MUST be composed of two parts. One part dedicated for the inner signature and one part dedicated for the outer signature.

6. Security Gateways

Some security gateways sign or verify messages that pass through them. Compliant gateways MUST apply the process described in chapter 4.5.

For non-compliant gateways, the presence of SecureHeaderFields structure do not change their behavior.

In some case, gateways MUST generate new signature or insert signerInfos into the signedData block. The format of signatures generated by gateways is outside the scope of this document.

7. Security Considerations

This specification describes an extension of the S/MIME standard. It provides message headers integrity, non-repudiation and confidentiality. The signature and encryption processes are complementary. However, according to the security policy, only the signature mechanism is applicable. In this case, the signature process is implemented between MUAs. The encryption process requires signed messages with Secure Headers extension. If required, the encryption process is implemented by DCAs.

This specification doesn't address end-to-end confidentiality for message header fields. Messages sent and received by MUAs could be transmitted as plaintext. In order to avoid interception, the use of TLS is recommended between MUAs and DCAs (uplink and downlink). Another solution might be the use of S/MIME between MUAs and DCAs in the same domain.

For the header field confidentiality mechanism to be effective all DCAs supporting confidentiality must support SH processing. Otherwise, there is a risk in the case where headers are not obscured upon encryption, or not restored upon decryption process. In the former case confidentiality of the header fields is compromised. In the latter case the integrity of the headers will appear to be compromised.

8. IANA Considerations

IANA must register a suitable Object Identifier (OID) value for the identifier id-aa-secureHeaderFieldsIdentifier. This value will be used to identify an authenticated attribute carried within a CMS [RFC 5652] wrapper. This attribute OID appears in Section 4.1, and again in the reference definition in Appendix A. An appropriate registry arc is suggested in those instances of the draft text.

9. References

9.1. Normative References

[RFC 2045] Borenstein, N., "Multipurpose Internet Mail Extensions Part One", RFC 2045, November 1996.

[RFC 2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[RFC 2634] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[RFC 4853] Housley, R., "Cryptographic Message Syntax (CMS), Multiple Signer Clarification", RFC 4853, April 2007.

[RFC 5322] Resnick, P., "Internet Message Format", RFC 5322, October 2008.

[RFC 5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[RFC 6376]   Crocker, D., Hansen, T., Kucherawy, M., DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011.

[ASN1-88]   CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

9.2. Informative References

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

[RFC 3183]   Dean, T., Ottaway, W., "Domain security services using S/MIME", RFC 3183, October 2001.

[RFC 3207]   Hoffman, P., "SMTP Service Extension for secure SMTP over Transport Layer Security", RFC 3207, February 2002.

[RFC 3501]   Crispin, M., "Internet Message Access Protocol, version 4rev1", RFC 3501, March 2003.

[RFC 5321]   Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[RFC 5598]   Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.

[RFC 5750]   Ramsdell, B., Turner, S., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, January 2010.

[RFC 5751]   Ramsdell, B., Turner, S., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2, Message Specification", RFC 5751, January 2010.

Appendix A. Formal syntax of Secure Header

Note: The ASN.1 module contained herein uses the 1988 version of ASN.1 notation [ASN1-88] for the purposes of alignment with th existing S/MIME specifications. The secure header structure is defined as follows:

```
SMimeSecureHeadersV1
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) smime(16) modules(0) secure-headers-v1(to be
defined) }

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

  id-aa
       FROM SecureMimeMessageV3dot1
            { iso(1) member-body(2) us(840) rsadsi(113549)
            pkcs(1) pkcs-9(9) smime(16) modules(0)
            msg-v3dot1(21) };

-- id-aa is the arc with all new authenticated and
-- unauthenticated attributes produced by the S/MIME
-- Working Group

 id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::= id-aa
    secure-headers (to be defined) }

 SecureHeaderFields ::= SET {
      canonAlgorithm Algorithm,
      secHeaderFields HeaderFields }

 Algorithm ::= ENUMERATED {
      canonAlgorithmSimple(0),
      canonAlgorithmRelaxed(1) }
```

```
HeaderFields ::= SEQUENCE SIZE (1..max-header-fields) OF
   HeaderField

max-header-fields INTEGER ::= MAX

HeaderField ::= SEQUENCE {
     field-Name HeaderFieldName,
     field-Value HeaderFieldValue,
     field-Status HeaderFieldStatus DEFAULT duplicated }

HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
     -- This description matches with the description of
     -- field name in the chapters 2.2 and 3.6.8 of RFC 5322

HeaderFieldValue ::= UTF8String
     -- This description matches with the description of
     -- field body in the chapter 2.2 of RFC 5322 as
     -- extended by chapter 3.1 of RFC 6532.

HeaderFieldStatus ::= INTEGER {
     duplicated(0), deleted(1), modified(2) }

END
```

Appendix B. Secure Header Fields example

In the following example, header fields subject, x-ximf-primary-precedence and x-ximf-correspondance-type are secured and integrated in a SecureHeaders structure. This example should produce a valid signature.

Extract of message header fields

```
From: John Doe <jdoe@example.com>
To: Mary Smith <mary@example.com>
subject: This is a test of Ext.
x-ximf-primary-precedence: priority
x-ximf-correspondance-type: official
```

SecureHeaders structure extracted from signature:

```
2286  150: SEQUENCE {
2289   11:   OBJECT IDENTIFIER '1 2 840 113549 1 9 16 2 80'
2302  134:   SET {
2305  131:     SET {
2308    4:       ENUMERATED 1
2314  123:       SEQUENCE {
2316   40:         SEQUENCE {
2318   25:           VisibleString 'x-ximf-primary-precedence'
2345    8:           UTF8String 'priority'
2355    1:           INTEGER 0
         :           }
2358   41:         SEQUENCE {
2360   26:           VisibleString 'x-ximf-correspondance-type'
2388    8:           UTF8String 'official'
2398    1:           INTEGER 0
         :           }
2401   36:         SEQUENCE {
2403    7:           VisibleString 'subject'
2412   22:           UTF8String 'This is a test of Ext.'
2436    1:           INTEGER 0
         :           }
```

```
           :         }
           :       }
           :     }
           :   }
```

Example is displayed as an output of Peter Gutmann's "dumpasn1" program.

OID used in this example is non-official.

Appendix C. Acknowledgements

The authors would like to thank Jim Schaad, Alexey Melnikov, Damien Roque, Thibault Cassan, William Ottaway, and Sean Turner who kindly provided reviews of the document and/or suggestions for improvement. As always, all errors and omissions are the responsibility of the authors.

Authors' Addresses

Laurent CAILLEUX
DGA MI
BP 7
35998 RENNES CEDEX 9
France
Email: laurent.cailleux@intradef.gouv.fr

Chris Bonatti
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA
Email: bonatti252@ieca.com