MILE
Internet-Draft
Intended status: Standards Track
Expires: August 15, 2014

J. Schaad
Soaring Hawk Consulting

February 11, 2014

Plasma Protected IODEF
draft-schaad-mile-iodef-plasma-00.txt

Abstract

The Incident Object Description Exchange Format (IODEF) defines a XML representation for information about computer security incidents. The driver for the standardization effort for IODEF is the desire to share the information as part of the cybersecurity response. As the security considerations of RFC5070 notes, the data can be sensitive and should only be disclosed to appropriate parties. This document describes how to use the Plasma policy enforcement model to ensure access to the IODEF data follows the appropriate policies in a distributed environment and independent of the transports used to share the information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

It has long been held that ’knowledge is power’ and that getting the right information in a timely manner to decision makers helps them make well informed decisions. In cybersecurity, that information is often spread across many stakeholders. Getting the right information to the operational teams responding to cybersecurity incident helps them reduce risks, deter attacks, mitigate exploits and enhance resilience. The need for effective and timely information sharing has been recognized by policymakers, executives and security professionals alike.

At times, cybersecurity information will be sensitive e.g. because of national security implications or due to potential commercial business impact. Policy will require the information has to be shared on a need-to-know basis which requires definition and enforcement of some criteria to establish a subjects need to know the information. The stakeholders need both confidence in the robustness of the technical controls which implement the policy as well as a means to demonstrate compliance with the policy as prerequisites to entrusting their sensitive data to such a system.

The need for information sharing is a fundamental part of collaborative endeavors. It can take many forms due to the context of the collaboration. The policies governing the information sharing also apply to the information regardless of which tool us used to convey the information. Collaborative efforts have a rich and diverse toolset for exchanging information and cybersecurity collaboration is no exception. It is necessary for any policy enforcement mechanism supporting information exchange such as IODEF be part of a "bigger picture" so that the same policies can be enforced on any cybersecurity information regardless of the tools used to share that information.

1.1. Requirements Terminology

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 [RFC2119].

When the words appear in lower case, their natural language meaning is used.

2. Background

2.1. Cybersecurity Information Sharing

Calls to enhance cybersecurity information sharing have been made regularly over the past two decades. The need for cybersecurity information sharing within private critical infrastructure sectors and with the government has been identified as an important practice to help better secure the increasingly cyber-dependent critical infrastructure. Any policy controls also have to strike a balance between reasonable and robust technical controls and legal enforcement of contractual obligations.

2.1.1. Actors

There are a number of different actors involved in the cybersecurity ecosystem who are each looking for and contributing different information which requires control not only access to the data but also use and onward publication of the information.

Governments are concerned about national economic and security issues.

Enterprises are subject to cybercrime and cyberespionage and need to protect their sensitive information such as customer data, intellectual property and trade secrets.

IT companies who provide products and services to Governments and enterprises are concerned about the security and integrity of their offerings

IT security firms who offer security specific products and services as well as cybersecurity information are concerned about keeping their products and services current.

Researchers track incidents and find vulnerabilities in the products and services from IT companies are looking for new events and trends in the data.

Academia performs security research.

There are several types of cybersecurity information: incidents, situational awareness, best practices, strategic analysis, threat, vulnerability, and mitigation information. These various types of information have different uses, and are often produced and utilized for different purposes by the different actors. This is a similar situation to health care where a health care practitioner and medical statistician would both need access to a particular medical record for totally different purpose, one where the identity of the patient is part of the data set and one where the information forms part of an anonymous data set. Similar consideration exist for cybersecurity information so access control need to be flexible to enables different forms of data use without compromising compliance.

2.1.2. Plasma and the Traffic Light Protocol

The Traffic Light Protocol (TLP) is a means for the originator of data to indicate how widely they want the information shared. It is an advisory notice and relies on the recipients being trusted to understand and obey the rules of the protocol. The originator marks the information with a hierarchical marking to indicate the scope of the onward dissemination of the information. The markings are as follows

RED Most restrictive, Very small community of interest. Admission to the community strictly controlled. Typically named recipients only.

AMBER Limited Disclosure. Moderate sized community of interest within participating organizations. Reasonable need to know admission test to community of interest. Typically named organizations only.

GREEN Moderate Disclose. Large community of interest with participating organizations and their partners. Minimal need to know test for admission to community of interest.

WHITE Public Data

Plasma allow for the implementation of the TLP with more rigor where the incident owner can better control the release of the information. The incident owner can define specific kneed to know criteria it deems appropriate for the incident and for the current TLP making to be communicated to the recipient. As the incident transitions from breaking news to ancient history, it allows the incident owner to relax the policy accordingly without impacting the incident data held across the ecosystem.

2.1.3. Topologies

Cybersecurity information sharing used an asynchronous message paradigm. The Information sharing can follow all the standard topology options

- o Peer to Peer

- o Mesh Topology

- Star Topology

Peer to peer offer the highest control and security but does not scale and is the most fragile. The other topologies improve the scalability and availability but at the cost of security and control. Nodes in the more complex topologies would typically not be under the control of the senders or recipients organizations. They may be trusted to route messages between senders and recipients but do not have a need to know the content of cybersecurity information. The need to leverage services to enable high availability and resilience without the need to also trust such services with sensitive data is paramount. This is similar to email which has similar topologies where users trust services to deliver email and be highly resilient and available while not wanting them to have access to sensitive content.

2.1.4. Requirements for Strong Policy Enforcement on Cybersecurity Information

- For the ecosystem to enable the data sharing while enforcing the policy considerations around the sensitivity and use of the data.

- For the actors, devising new ways to use the data so any policy enforcement mechanism need to be flexible and extensible to adapt to the changes.

- Public and private laws will continue to evolve and adapt so any policy enforcement mechanism needs to be extensible and expressive to ensure fidelity of the policy.

- For implementers, to have a mechanism which abstracts them as much as possible from the details of the policy decisions. To have a clear and concise set of requirements to enable the policy decisions.

to do: more in data use and policy

2.2. PoLicy enhAnced Secure eMAil (Plasma)

Email remains one of the most wildly used tools for collaboration. It has mature and widely deployed standards for security in S/MIME [RFC5751] and PGP [RFC4880] which deliver basic security services (confidentiality, integrity and data origin authentication). S/MIME also has optional Enhanced Security Service [RFC5035] which can deliver policy enforcement on S/MIME messages.

Despite all this, secure email is still the exception as a percentage of the overall email traffic. It is used in communities of interest including cybersecurity, but does not deliver robust policy enforcement on the contents of the message. Plasma was an effort to fundamentally rethink and update email security model to enable it to align with other technologies and enable its broader use and deliver strong policy enforcement on message continents. Though some of the work was specific to S/MIME [I-D.schaad-plasma-cms], it was based on a generic data model [I-D.freeman-plasma-requirements] and has a generic decision request\response protocol [I-D.schaad-plasma-service] which can support other types of data and applications.

Plasma developed a generic data model for policy enforcement on information. One of the objectives of the model is to enable consistent policy enforcement on information for a broad set of users, across a broad set of environments and applications. The Plasma data model, leverages many of the developments in identity and identity attributes. Plasma closely ties the meta-data of the applicable policies to data in order to deliver consistent policy enforcement for mobile data. It users a tamper proof binding so the policy relationship reliably travels with the data. The model relies on attributes about the subject requesting access, their system, the data and their environments as inputs to the policy to deliver Attribute Based Access Control (ABAC). The policy processing is complex so the model does not require the rules to be distributed to clients. The clients request decisions from a Policy Decision and Enforcement Point (PDEP) service who render decisions for the subjects. The PDEP’s in turn, discover the necessary policy from Policy Authoring Points.

2.2.1. Benefits of Policy Enforcement on Cybersecurity Information Sharing

For the Actors, it incentivize the exchange of the very freshest and interesting data, maximizes the way to derive intelligence from the data while managing the risk of unexpected use or abuse of the information.

For the regulators, and lawyers, it supports their policy needs in a smarter, more business friendly way.

For implementers, it simplifies thief products by abstracting a wide variety of issues to be policy decisions.

For the ecosystem, it supports new use cases at scale with reduced compliance costs.

2.2.2. Plasma Policies and Decisions

Polices in Plasma are set of rules which render either a result (permit, deny) or an error (indeterminate, and unknown) based on the supplied attributes of a request. Decisions are logic gates where one or more policies together with their logical relationship are used together to render a policy decision (permit, deny) or an error (indeterminate, and unknown). A single Plasma Token can contain one or more decision logic gates making it possible to render multiple decisions from a single request. The number of decisions within the policy object is hidden by design from the client. Each decision is enforced by a separate encryption key. A separate policy object would only be required if a Plasma server was not trusted to make a decision for all policies e.g. data being aggregated from different communities.

Information may be shared under multiple policies, for examples an organization may have specific cybersecurity information sharing agreements with some organizations, and pre-existing non-disclosure agreements with other organizations and an incident could be shared providing one or other of the policies is met. Equally, information from different organizations can be commingled e.g. where an IODEF document contains incident’s from different organizations, where each organization would be asserting its policies on the incident data. Both scenarios are supported by Plasma. The policy request and evaluation process can be time consuming therefore content creators should minimize the number of policy objects and policy decisions where creating content for publication.

Full details of the Plasma data model can be found in Section 4 of the Requirements for Message Access Control [I-D.freeman-plasma-requirements]

2.3. Plasma and IODEF

XML encryption allows for very granular protection of sensitive data in an XML document. It allows for protection of entire elements, element content and arbitrary data in XML documents. XML encryption can also be nested whereby part of the data being encrypted is already encrypted (Super-Encryption). This allows the content creator of an IODEF documents full control to protect any portion of the document they need. Once the data has been encrypted, Plasma allows the encryption key to be linked to a Plasma Decision via the Plasma Token where one or more policies can be combined to reflect the data governance requirements of the information. Cybersecurity information in the IODEF document requiring different data governance, can be combined in a single document and protected with different keys linked to different decisions.

```
          +-----------------+                +-----------------+
          | Plasma Token    |                | IODEF Incident  |
          +-----------------+                +-----------------+
          | Policy Decision |----------------| Encrypted Data  |
          |   CEK ID ABCD   |                |   CEK ID ABCD   |
          |   CEK 2412      |                |   HASH 1234     |
          +-----------------+                +-----------------+
          | HASH 1234       |
          +-----------------+
```

Figure 1: Single Plasma Policy Decision and Protected Incident

This is the simplest example with a single incident and a single decision. The incident is encrypted with a single CEK. If a recipient passes the policy decision check, the Plasma server would release the CEK enabling the recipient to decrypt the incident.

```
        +-----------------+                +-----------------+
        | Plasma Token    |                | IODEF Incident  |
        +-----------------+                +-----------------+
        | Policy Decision |----------------| Encrypted Data  |
        |   CEK ID ABCD   |       |        |   CEK ID ABCD   |
        |   CEK 2412      |       |        |   HASH 1234     |
        +-----------------+       |        +-----------------+
        | HASH 1234       |       |
        |      5678       |       |        +-----------------+
        +-----------------+       |        | IODEF Incident  |
                                  |        +-----------------+
                                  +--------| Encrypted Data  |
                                           |   CEK ID ABCD   |
                                           |   HASH  5678    |
                                           +-----------------+
```

Figure 2: Single Plasma Policy Decision and Two Protected Incidents with Same Policy Decision

When there are multiple incidents subject to the same decision, they are encrypted using the same CEK. Again, a recipient passing the policy decision check, will receive the CEK which enables them to decrypt both incidents.

```
        +-----------------+                +-----------------+
        | Plasma Token    |                | IODEF Incident  |
        +-----------------+                +-----------------+
        | Policy Decision |----------------| Encrypted Data  |
        |   CEK ID ABCD   |       |        |   CEK ID ABCD   |
        |   CEK 2412      |       |        |   HASH 1234     |
        | Policy Decision |       |        +-----------------+
        +   CEK ID A1B2   |       |
        |   CEK F469      |       |
        +-----------------+       |        +-----------------+
        | HASH 1234       |       |        | IODEF Incident  |
        |      5678       |       |        +-----------------+
        +-----------------+       +--------| Encrypted Data  |
                                           |   CEK ID A1B2   |
                                           |   HASH  5678    |
                                           +-----------------+
```

Figure 3: Single Plasma Policy Decision and Two Protected Incidents with Two Policy Decision

When there are incidents subject to different policy decision, this can still be accommodated within the same token and hence same decision request. Each incident is encrypted with different CEK, one CEK per decision. A recipient receives the CEK for every policy check they pass.

```
        +-----------------+         +-----------------+
        | Plasma Token    |         | IODEF Incident  |
        +-----------------+         +-----------------+
        | Policy Decision |---------| Encrypted Data  |
        |   CEK ID ABCD   |         |   CEK ID ABCD   |
        |   CEK 2412      |         |   HASH 1234     |
        | Policy Decision |         +-----------------+
        +   CEK ID A1B2   |                 |
        |   CEK F469      |                 |
        +-----------------+                 |        +-----------------+
        | HASH 1234       |                 |        | Assessment      |
        |      5678       |                 |        +-----------------+
        +-----------------+                 +--------| Encrypted Data  |
                                                     |   CEK ID A1B2   |
                                                     |   HASH  5678    |
                                                     +-----------------+
```

Figure 4: Single Plasma Policy Decision, a Protected Incidents with child class with a different policy decision

The same approach can be applied when an incident has a child class with a different policy to the parent. The child class is encrypted with different CEK to the parent. A recipient receives the CEK for every policy check they pass.

Question: Do we need to add a policy token consolidation request i.e. if a client finds multiple tokens from the same server, submit to server and ask for them to be merged into one.

2.4. Plasma and Layered Application Design

Today’s applications are built using separate layers which group components which discreet functions together into distinct layers. These layers can be described as follows

- o Presentation Layer: Components responsible for managing users interaction with the application
- o Business Layer: Components responsible for core business logic
- o Data Layer: Components responsible for interacting with data sources to enable the abstraction of the storage mechanism from business layer.

```
            +---------------------+
            |  Users              |
            +---------------------+
                      |
                      |
            +---------------------+
            | Presentation Layer |
            +---------------------+
                      |
                      |
            +---------------------+
            | Business Layer     |
            +---------------------+
                      |
                      |
            +---------------------+
            | Data Layer         |
            +---------------------+
                  |        |
                  |        |
      +--------------+  +--------------+
      | Data Source  |  | Services     |
      +--------------+  +--------------+
```

Figure 5: Layered Application Model

The objective of the layers is to deliver the best maintainability, extensibility and flexibility for the application. Plasma is part of the security service which is a cross layer function which can manifest in all layers. The layers are a logical separation which allows for the different components to be deployed in different physical combinations to respond to different sociability, performance and security considerations without impacting the underlying components.

The Plasma model fully supports the layered application model. Access to data becomes a policy issue i.e. does the policy allow the subject to access the data. For example if the business layer was deployed on a server or local on the users client, it would change the identity of the subject and (and the attributes) of the access request, but providing the subject met the policy requirements, either could be given access to the data.

3. The Plasma Protected IODEF Data Model

Note. Some harmonization work is in progress between this document and [I-D.schaad-plasma-service] so XML schema names and types may change as a result.

The Plasma protected IODEF model supports IODEF documents with multiple Incident’s. If all the incidents have the same security policy, then the same Plasma server(s) can control access to all the Incidents and a single instance of the Plasma Token containing a single content encryption key (CEK) for all incidents can be used. If incidents have different security polices, but the same Plasma server is trusted to perform the access control decision for all the policies, again a single instance of the Plasma Token with multiple CEKs can be used (one for each decision). If the Incidents have different security policies and the same Plasma server is not trusted with all the decisions then multiple Plasma Tokens can be used.

3.1. PlasmaToken Class

The PlasmaToken class contains the Plasma meta-data that allows the Plasma server to enforce policy decisions on the protected IODEF data. This is an XML analog of the ASN.1 encoded Plasma token structure defined in [I-D.schaad-plasma-cms]. The Plasma token contains encrypted content defined in [I-D.schaad-plasma-cms] which is processed by the Plasma server to convey policy requirements and content encryption keys. The token is signed by the Plasma server and the signature has signed elements to enable the receiving client to process the Plasma Token. It has a signed element with one or more URIs that identify the set of Plasma servers which can process the Policy Token. It also has an element containing the hash(s) of the encrypted content associated with the token to establish a binding between the protected IODEF data and the specific Plasma Token.

The PlasmaToken class uses the Class extension mechanism defined in [RFC5070] Section 5.2.

```
      +------------------------+
      | PlasmaToken            |
      +------------------------+
      |                        |<>----------[ EncryptedData ]
      |                        |<>--(1..*)--[ ServerURI ]
      |                        |<>----------[ EncryptedDataHashs ]
      +------------------------+
```

Figure 6: PlasmaToken Class

The aggregate classes in the Plasma Token are as follows:

EncryptedData One. The element defined in [W3C.WD-xmlenc-core1-20101130] that contains the encrypted data used by the Plasma server to process access requests to the protected IODEF data. The encapsulated contents of this element are defined in [I-D.schaad-plasma-cms]

ServerURI One or more. The URI of one or more Plasma servers which can process decisions requests for the Plasma Token. The order of URLs does not indicate any order of priority, it is a matter of local client policy on the order to use. The URL defines both the destination server and the protocol to be used. When the schema for the URL is "plasma", then the protocol which MUST be used is [I-D.schaad-plasma-service].

It is a matter of local policy of the IODEF recipient if it chooses to contact one of the plasma servers identified by the ServerURI based on their trust in the identity of the signer of the Plasma Token.

3.1.1. EncryptedDataHashs Class

Todo, this might get wrapped into the re-factoring.

For privacy reasons, it is highly desirable that the recipient client of an IODEF document can validate that the Plasma Token embedded in a document, is associated with the encrypted data it is attached to prior to contacting the Plasma server. For this reason, in addition to the requirement that a recipient validate the signature of the Plasma server over the token, a new element is defined which contains one or more hashes of the encrypted content(s). These encrypted data hashes constitute a detached signature of the encrypted content.

The EncryptedDataHashs class contains the hash values for the one or more sets of encrypted data.

```
+--------------------------+
| EncryptedDataHashes      |
+--------------------------+
|                          |<>----------[ DigestMethod  ]
|                          |<>--(1..*)--[ DigestValue   ]
+--------------------------+
```

Figure 7: EncryptedDataHashs Class

DigestMethod one. The element defined in [W3C.WD-xmldsig-core2-20100831] that identifies the digest algorithm to be applied to the encrypted protected data e.g. an encrypted incident, associated with the Plasma Token.

DigestValue One or more. The element defined in [W3C.WD-xmldsig-core2-20100831] that contains the encoded value of the digest of the encrypted protected data. If the token has been used to protect multiple elements e.g. multiple incidents, then there will be multiple digest values.

4. Plasma Service Request/Response Messages

This specification uses the [I-D.schaad-plasma-service] specification to process decision requests for IODEF protected data. This specification defines new actions and token types.

4.1. Create IODEF Documents Request

The create document message request is built using the Plasma:PlasmaRequest XML structure defined in [I-D.schaad-plasma-service]. When building the request, follow [I-D.schaad-plasma-service] with the following changes:

- o The client MUST include an action attribute. The document defines the GetXMLPlasmaToken action attribute.

- o A message requesting a XML Plasma token looks like this:

```
   <Plasma:PlasmaRequest>
     <Plasma:Authentication>
       <Plasma:WS_Token>
         Role Token goes here
       </Plasma:WS_Token>
     </Plasma:Authentication>
     <xacml:Request>
       <xacml:Attributes Category="...:action">
         <xacml:Attribute AttributeId="urn:plasma:action-id">
           <xacml:AttributeValue>
             GetXMLPlasmaToken
           </xacml:AttributeValue>
         </xacml:Attribute>
       </xacml:Attributes>
     <xacml:Attributes Category="...:data">
       <xcaml:Attribute AttributeId="urn:plasma:data-id">
         <xacml:AttributeValue>
           <Plasma:GetXMLPlasmaToken>
             <Plasma:Label>
                ... Label Tree for message ...
             </Plasma:Label>
             <Plasma:EncryptedDataHashs>
               ... Hash algorithm and hash(s) of encrypted content ...
             </Plasma:EncryptedDataHashs>
             <Plasma:CEK>
                ... Content Encryption Key ...
             </Plasma:CEK>
           </Plasma:GetXMLPlasmaToken>
         </xacml:AttributeValue>
        </xcaml:Attribute>
       </xacml:Attributes>
     </xacml:Request>
   </Plasma:PlasmaRequest>
```

4.2. Create IODEF Document Response

In response to a create document request, the Plasma server returns a create document response message. The response messages uses the plasma:PlasmaResponse XML structure. When the response message is created, the following should be noted:

- o The xacml:Decisions is always included in the response. If the ’Permit’ value is returned then the Plasma:XMLToken element MUST be present.

- o The PlasmaReturnToken element with a Plasma:XMLToken content is included with a permit response.

An example of a message returning the set of policy information is:

```
<Plasma:PlasmaResponse>
  <xacml:Response>
    <xacml:Result>
      <xacml:Decision>Permit</xacml:Decision>
    </xacml:Result>
  </xacml:Response>
    <Plasma:PlasmaReturnToken xsi:"Plasma:XMLTokenResponseType">
        <Plasma:XMLPlasmaToken>xxx token xxxx</Plasma:XMLPlasmaToken>
  </Plasma:PlasmaReturnToken>
</Plasma:PlasmaResponse>
```

4.3. Read IODEF Document Request

The client sends a request to the Plasma server that is identified in the token. For the XML tokens, the address of the Plasma server to use is located in the ServerURI element of the Plasma Token.

The request uses the plasma:PlasmaRequest XML structure. When building the request, the following should be noted:

- The xacml:Request MUST be present in the first message of the exchange.

- The action used to denote that a XML token should be decrypted is "ParseXMLToken"

- The XML token to be cracked is identified by "XMLToken"

- If the client is using the XML Digital Signature element in this message, then the client MUST include the cryptographic channel binding token (Section 10.1.1) in the set of XACML attributes.

An example of a message returning the set of policy information is:

```
   <plasma:PlasmaRequest>
       <plasma:Authentication>...</plasma:Authentication>
         <xacml:Request>
             <xacml:Attributes Category="...:action">
                 <xacml:Attribute AttributeId="..:action-id">
                      <xacml:AttributeValue>ParsePlasmaToken />
                 </xacml:Attribute>
             </xacml:Attributes>
         <xacml:Attribute Category="...:data">
              <xacml:Attribute AttributeId="..:data:XMLToken">
                <xacml:AttributeValue> XML Token </xacml:AttributeValue>
              </xacml:Attribute>
           </xacml:Attribute>
       </xacml:Request>
   </plasma:PlasmaRequest>
```

4.4. Read IODEF Document Response

In response to a parse token request, the Plasma server returns a decrypted key in the response. The response uses the plasma:Plasma XML structure. When a response message is create the following should be noted:

- o If the Plasma Token contained multiple decisions, a single response can be used for all decisions.

- o For each decision, if the value of xacml:Decision is Permit, then response MUST include an Plasma:XMLKey element.

- o For each decision, if the value of xacml:Decision is not Permit, the plasma:XMLKey MUST be absent.

An example of a message returning the set of policy information is as follows:

```
   <Plasma:PlasmaResponse>
       <xacml:Response>
           <xacml:Result>
                <xacml:Decision>Permit</xacml:Decision>
           </xacml:Result>
       </xacml:Response>
       <Plasma:Key>
            <Plasma:DisplayString>Label Text </Plasma:DisplayString>
           <Plasma:KEK>hex based KEK</Plasma:KEK>
       </Plasma:CMSKey>
   </Plasma:PlasmaResponse>
```

5. Processing Rules for protected IODEF

This is the set of processing steps that either a creator or receiver of protected IODEF needs to follow. The order of the steps is not normative.

5.1. Creating Protected IODEF data

These are the step that the creator of an protected IODEF message needs to do.

1. The creating agent obtains the set of policies under which it can create IODEF data.

2. The creating agent composes the IODEF content.

3. The creating agent determines the set of policies to be applied to the IODEF content.

4. The creating agent selects the content encryption algorithm (with input from the obligations of the policies chosen) and randomly creates the CEK(s).

5. The creating agent encrypts the content with the CEK and computes the encrypted hash value.

6. The creating agent transmits the CEK, the hash of the encrypted content value(s) and the policy label(s) to the PLASMA server.

7. If the creating agents request passes the Plasma server policy check, the Plasma server will return the Plasma Policy meta-data to the creating agent. If the policy validation fails then the creator cannot send the IODEF message under the requested policy label.

8. The creating agent verifies the signature on the Plasma Policy meta-data. If the Signature is current and passes cryptographic processing the sender can add the policy meta-data to the appropriate PolicyData element and sends the IODEF message.

5.2. Receiving Protected IODEF data

These are the steps that the recipient of a protected IODEF message needs to follow. The order of the steps is not normative.

1. The Receiving Agent obtains the message from another IODEF agent.

2. The Receiving Agent recognizes that it is protected IODEF content.

3. The Receiving Agent validates the PolicyData attribute. The following steps need to be taken for validation.

   A. The signature on the PolicyData structure is validated. If the validation fails then processing ends.

   B. The certificate used to validate the signature MUST contain the XXXX value in the EKU extension. The certificate MUST NOT contain the anyPolicy value in the EKU extension. Local policy can dictate that content of the PlasmaURL attribute be used in selecting trust anchors for the signing certificate.

   C. If the PlasmaURL attribute is absent, then processing fails.

   D. The URL value in the PlasmaURL attribute is checked against local policy. If the check fails then processing fails. This check is performed so that information about the user is not given to a random Plasma server. The schema of the URL MUST be one that the client implements. (For example the "plasma" schema associated with RFC XXX [I-D.schaad-plasma-service].) As discussed in Section 4.5 of [I-D.freeman-plasma-requirements], policy can be enforced on the edge of an enterprise, this means that if multiple URLs are present in the Plasma URL attribute they all need to be checked for policy and ability to use before this step fails.

   E. The EncryptedHash attribute value is checked against the encrypted content. If this attribute is absent then processing fails. If the value does not matched the computed value on the encrypted content then processing fails.

4. The recipient agent gathers the necessary identity and attribute statements, usual certificates or SASL statements.

5. The recipient agent establishing a secure connection to the Plasma server and passes in the identity and attribute statements and receives back the CEK or a lock box to allow it to obtain the CEK value.

6. the recipient agent uses the returned CEK to decrypt the protected content and compares the generated Message Authentication Code for the value in Authentication Tag and fail if they don’t match.

6. Examples

The following example is an IODEF document with 3 incidents. The first is a public incident where all data is in the clear. The second incident is a public incident with a private contact. The third incident is private.

```
  <?xml version="1.0" encoding="UTF-8"?>
    <iodef:IODEF-Document lang="en"
        xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:plasma="urn:ietf:params:ns:plasma:1.0">
      <iodef:Incident purpose="reporting" restriction="public">
        <iodef:IncidentID name="CERT-OUR-DOMAIN"
            CERT-OUR-DOMAIN#111-1/>
          <iodef:ReportTime 2014-02-05T10:21:05+00:00/>
          <iodef:Assessment>
            <iodef:Impact severity="high" It go boom />
          </iodef:Assessment>
          <iodef:Contact role="creator" type="organization">
            <iodef:ContactName Trevor Freeman />
            <iodef:Description Lead contact />
          </iodef:Contact>
        </iodef:Incident>
        <iodef:Incident purpose="reporting">
          <iodef:IncidentID
           name="CERT-OUR-DOMAIN">CERT-OUR-DOMAIN#111-2/>
          <iodef:ReportTime>2014-02-06T10:21:00+00:00 />
      <iodef:Assessment>
        <iodef:Impact severity="medium" It go splash />
      </iodef:Assessment>
      <iodef:EncryptedContact>
            <xenc:EncryptionMethod
               Algorithm="http://www.w3.org/2009/xmlenc11#aes128-gcm"/>
           <ds:KeyInfo>
             <ds:KeyName>Plasma#1</ds:KeyName>
           </ds:KeyInfo>
           <xenc:CipherData>
                 <xenc:CipherValue XXXX Encrypted iodef:Contact XXXXX />
           </xenc:CipherData>
         </iodef:EncryptedContact>
      </iodef:Incident>
      <iodef:EncryptedIncident>
            <xenc:EncypteData>
          <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2009/xmlenc11#aes128-gcm">
            <ds:KeyInfo>
              <ds:KeyName Plasma#2 />
                </ds:KeyInfo>
            <xenc:CipherData>
                  <xenc:CipherValue>XXXX Encrypted Incident XXXX />
            </xenc:CipherData>
              </xenc:EncryptionMethod>
        </xenc:EncypteData>
      </iodef:EncryptedIncident>
      <iodef:AdditonalData dtype="xml">
        <xenc:KeyInfo>
          <plasma:PlasmaKey>
            <ds:SignedInfo>
              <ds:CanonicalizationMethod
                 Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
              <ds:SignatureMethod
           Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
              <ds:Reference Id="EncryptedKey">
            <ds:DigestMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                <ds:DigestValue>XXXX Digest XXXX/>
              </ds:Reference>
           </ds:SignedInfo>
           <ds:SignatureValue>XXXX Signature XXXX />
           <ds:KeyInfo>
             <ds:X509Data>Put a certificate here</ds:X509Data>
           </ds:KeyInfo>
           <ds:Object>
             <plasma:LockBox id="EncryptedKey">
               <xenc:CipherText>
             <xenc:CipherValue>xxxxxxxxxx />
           </xenc:CipherText>
           <plasma:EncryptedHashes>
             <ds:DigestMethod Algorithm="#sha1"/>
             <ds:DigestValue>XXXXX#1</ds:DigestValue>
             <ds:DigestValue>XXXXXX#2</ds:DigestValue>
           </plasma:EncryptedHashes>
           <plasma:Server url="plasma:PlasmaServerName.com"/>
             </plasma:LockBox>
               </ds:Object>
             </plasma:PlasmaKey>
          </xenc:KeyInfo>
      </iodef:AdditonalData>
  </iodef:IODEF-Document>
```

7. XML Schema

This schema is the XML analogue of the CMS recipient info structure defined in [I-D.schaad-plasma-cms]. It contains the encrypted data used by the Plasma server. The encrypted data contains the policy decision leaf structures and CEKs. It also has any other attributes necessary for processing the request e.g. resource and audit attributes. The Plasma token also has a number of signed elements necessary for the client to process the token.

7.1. IODEF Document with encrypted classes

When a client wants to validate the XML schema of an IODEF document containing encrypted classes prior to processing the contents, it MUST use a modified schema which allows for the substitution of the encrypted elements.

For example the current IODEF document class is as follows

```
<xs:element name="IODEF-Document">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="iodef:Incident"
                maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string" fixed="1.00"/>
        <xs:attribute name="lang" type="xs:language" use="required"/>
        <xs:attribute name="formatid" type="xs:string"/>
    <xs:complexType>
</xs:element>
```

The modified class schema needs to be as follows:

```
   <xs:element name="IODEF-Document">
       <xs:complexType>
           <xs:sequence>
               <xs:group ref="iodef:IncidentChoice"
                   maxOccurs="unbounded"/>
           </xs:sequence>
           <xs:attribute name="version" type="xs:string" fixed="1.00"/>
           <xs:attribute name="lang" type="xs:language" use="required"/>
           <xs:attribute name="formatid" type="xs:string"/>
       <xs:complexType>
   </xs:element>

   <xs:group name="IncidentChoice">
       <xs:choice>
           <xs:element> ref="iodef:Incident"/>
           <xs:element name="EncryptedIncedent"
                type="xencEncryptedDataType"/>
       </xs:choice>
   </xs:group>
```

The choice between the encrypted and unencrypted class MUST be inserted in every class with a restriction attribute.

7.2. Plasma Token

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:Plasma="PlasmaToken.xsd"
  xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <xs:element name="PlasmaToken" type="XML">
    <xs:complexType>
      <xs:sequence>
        <xenc:KeyInfo>
          <Plasma:PlasmaKey>
            <ds:SignedInfo maxoccurs="unbounded">
              <ds:CanonicalizationMethod />
              <ds:SignatureMethod />
              <ds:Reference id="EncryptedKey">
                <ds:DigestValue />
              </ds:Reference>
              </ds:SignedInfo>
            <ds:SigntureValue />
            <ds:KeyInfo>
              <ds:X509Data />
              <xs:complextype>
                <xs:sequence>
                  <Plasma:LockBox>
                   <xenc:CipherTExt>
                      <xenc:CipherValue />
                    </xenc:CipherText>
                  </Plasma:LockBox>
                  <Plasma:EncryptedDataHashes>
                    <ds:DigestMethod/>
                    <ds:DigestValue maxoccurs="unbounded"/>
                  </Plasma:EncryptedDataHashes>
                  <Plasma:ServerURI maxoccurs="unbounded" />
                 </xs:sequence>
              </xs:complextype>
            </ds:KeyInfo>
          </Plasma:PlasmaKey>
        </xenc:KeyInfo>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8. Mandatory Algorithms

Clients MUST implement the mandatory algorithms defined for XML encryption [W3C.WD-xmlenc-core1-20101130] for the encryption of IODEF document contents. Clients SHOULD use AES128-GCM unless otherwise directed by a policy obligation. Other algorithms may be implemented.

Clients MUST implement SHA-256 and SHA-512 as defined for message digest [W3C.WD-xmlenc-core1-20101130] for computation of the Encrypted Content Hash. Clients SHOULD use SHA-256 unless otherwise directed by a policy obligation. Other algorithms MAY be implemented.

When verifying signatures on the Plasma Token, clients MUST be able to verify the RSA v1.5 signature algorithm with SHA-256 and SHA-512. Clients MUST also be able to verify the EC-DSA signature algorithm with SHA-256 and SHA-512 signature algorithm. Clients MAY be able to verify other signature algorithms.

9. Security Considerations

A malicious Plasma server can generate a Plasma token over any protected content i.e. there is no guarantee that the Plasma server knows the CEK of the protected data or if it is genuine data at all and free from malicious content. For example, it can generate a new Plasma token for some existing protected content with the hashes of the encrypted data. The fact that the signature of the Plasma token validates along with the hashes of the encrypted data is only a integrity check over the data set i.e. if it fails, processing should fail. The fact that the signature and associate data hashes validates MUST NOT be uses as any indication of trustworthiness of the Plasma Server.

10. IANA Considerations

Tbd

11. References

11.1. Normative References

[I-D.schaad-plasma-cms]
Schaad, J., "Plasma Service Cryptographic Message Syntax (CMS) Processing", draft-schaad-plasma-cms-04 (work in progress), March 2013.

[I-D.schaad-plasma-service]
Schaad, J., "Plasma Service Trust Processing", draft-schaad-plasma-service-04 (work in progress), January 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5070] Danyliw, R., Meijer, J., and Y. Demchenko, "The Incident Object Description Exchange Format", RFC 5070, December 2007.

[W3C.WD-xmldsig-core2-20100831]
Eastlake, D., Reagle, J., Solo, D., Yiu, K., Hirsch, F., Roessler, T., and P. Datta, "XML Signature Syntax and Processing Version 2.0", World Wide Web Consortium WD WD-xmldsig-core2-20100831, August 2010, <http://www.w3.org/TR/2010/WD-xmldsig-core2-20100831>.

[W3C.WD-xmlenc-core1-20101130]
Roessler, T., Reagle, J., Hirsch, F., and D. Eastlake, "XML Encryption Syntax and Processing Version 1.1", World Wide Web Consortium LastCall WD-xmlenc-core1-20101130, November 2010, <http://www.w3.org/TR/2010/WD-xmlenc-core1-20101130>.

11.2. Informative References

[I-D.freeman-plasma-requirements]
Freeman, T., Schaad, J., and P. Patterson, "Requirements for Message Access Control", draft-freeman-plasma-requirements-08 (work in progress), October 2013.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

[RFC5035] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, August 2007.

[RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.

Author’s Address

Jim Schaad
Soaring Hawk Consulting

Email: ietf@augustcellars.com