Extended Secure Architecture of HIS: HL7

D. Saini, H. Saini, Wail M Omar & P. Dash

1 Associate Professor, Faculty of Computing & Information Technology, Sohar University, Sultanate of Oman
(Phone: ; Fax: +968-26720102 ; e-mail: dinesh@soharuni.edu.com)

2 Assistant Professor, Department of Information Technology, Orissa Engineering College, Bhubaneswar (India)-752050
(Phone: 0091-9778312596, Fax: +91-674-2542990, e-mail: hemraj1977@yahoo.com)

3 Associate Professor, Faculty of Computing & Information Technology, Sohar University, Sultanate of Oman
(Phone: ; Fax: +968-26720102 ; e-mail: w.omar@soharuni.edu)

4 Lecturer, Department of Information Technology, Orissa Engineering College, Bhubaneswar (India)-752050
(Phone: ; Fax: +968-26720102 ; e-mail: prasannajitdash@yahoo.com)

Abstract

The article deals to explain the object oriented architecture of the HL7 and its security issues. Primarily, the security breaches may occur at the direct interaction point of the HL7 architecture because of its interaction with the users. This direct interaction point is OnDemandDataSource class in the existing object oriented architecture of HL7. To resolve this problem an extended secure architecture of HL7 is provided in the paper. In this two new classes are suggested, one is SettingsManager, which deals to convert the input information in the XML formatted string object and another is DataAccessSettingsManager, which deals to implement all the security measures available in XML document. This XML document can contains different security measures such as Integrity and signatures, Confidentiality, Key Management, Authentication and Authorization, etc. Basic templates to implement these security measures by using XML are also provided for better understanding.

Keywords- HL7, XML security measures, XML document, Integrity and signatures, Confidentiality, Key Management, Authentication and Authorization

1. Introduction

In the current era of Electronic Health Record (EHR) the health sector is also requires an efficient and standardized hospital management information system (HIS) [1,2,3]. There are many HIS, implemented in the field. For interoperability it is required to standardizing the HIS. It can be done by some existing standards [4, 5, 6] such as the Health Level 7 (HL7) Clinical Document Architecture (CDA) [HL7 CDA Release 2.0 2005], CEN EN 13606 EHRcom [CEN prEN 13606-1 2004] and openEHR [OpenEHR]. These standards aim to structure and markup the clinical content for the purpose of exchange. There is also an industrial initiative called Integrating the Healthcare Enterprise (IHE) [IHE] which is specified the Cross-Enterprise Document Sharing (XDS) integration profile [IHE 2005d] for this purpose. The basic idea of IHE XDS is to store healthcare documents in an ebXML registry/repository architecture to facilitate their sharing.

In this paper, we explore and analyzed the existing object oriented architecture of the HL7 [7]. The existing architecture is having the class OnDemandDataSource to generate the clinical information for the concern users. This class is the direct interaction point to the users and hence to be secure. It implements some accessibility rules but these rules are not effectively secured and stored information can be misused by unauthorized personalities. To overcome this problem, the paper suggests adding two more classes in the existing architecture, namely, SettingsManager class and DataAccessSettingsManager class. The SettingsManager class simply generates the XML formatted information to the concerned users and DataAccessSettingsManager provides the rules for security by using XML document. In the later section the templates of XML document for security are given to make the implementation easy and understandable.
2. Existing Generic Architecture of HL7

The object oriented class diagram of existing HL7 is represented by the figure 1. The brief description of the various classes in the diagram is given as under-

ORUManager: ORUManager is responsible for processing the vitals data provided by Vitals Manager. It receives a list of messages from Vitals Manager and sends one vitals data at a time to Message Manager.

OrionMapper: OrionMapper is responsible for formatting the HL7 message based on the HL7 version passed to it.

ADTManager: ADTManager is responsible for processing the ADT events.

HL7MessageManager: HL7MessageManager is responsible for creating a message by adding EventHandlers and Segments associated with it.

MSMQManager: MSMQManager is responsible for managing (adding, retrieving and searching) the messages in MSMQ.

QRYManager: QRYManager receives HIS request through TCPManager. The received HIS message is added to MessageQueue through MSMQManager.

HL7EventHandlerManager: HL7EventHandlerManager is responsible for identifying the specific event and generating the data associated with it.

OnDemandDataSource: OnDemandDataSource provides functionality to access the data source.

HL7MessageParser: MessageParser is responsible for parsing the Query request sent by HIS and determines the query filters.

HL7TCPManager: HL7TCPManager is responsible for communicating with the HIS system.

IEventHandler: IEventHandler provides methods to be used by the event handlers.

ISegment: This interface provides the behaviour to be implemented by the Segments.

In the existing architecture the patients interact with the HIS to get their related health information. This information is provided by OnDemandDataSource to the patient but that information doesn’t incorporate the sufficient security measures. The insecurity makes a vulnerability to the existing architecture of the HL7. To cope of it the extended architecture of the HL7 is being provided in the next section of the paper.
3. Extended Secure Generic Architecture (ESGA) of HL7

A patient can change his/her data without any knowledge of the doctor or nurse or physician in the already existing HL7 architecture. A doctor can even manipulate patient data in ignorance of patient or nurse or physician. These same possibilities could happen even to nurse or physician or biomedical engineer. Hence, the clinical history data related to patient in HIS should undergo any kind of encryption or encoding or digital signature to fulfill the security aspect. This can be achieved by using XML document security measures [8, 9, 10]. To implement the security measures the clinical data must be converted by XML formatted string objects and it is achieved by SettingsManager class (a newly inserted class in the existing HL7 Architecture). Further, the security measures are implemented by another newly inserted class DataAccessSettingsManager. ESGA of HL7 is shown by the figure 2. The description of both the newly introduced classes is as follows:

SettingsManager: SettingsManager class is responsible to generate the XML formatted string objects.

DataAccessSettingsManager: DataAccessSettingsManager is responsible to provide the rules for security which are used for XML document such as encryption, encoding or digital signature.

4. XML Security Templates of ESGA of HL7

The XML Security standards [11, 12] include XML Digital Signature for integrity and signing solutions, XML Encryption for confidentiality, XML Key Management (XKMS) for public key registration, location and validation, Security Assertion Markup Language (SAML) for conveying authentication, authorization and attribute assertions.
XML Security reuses the concepts, algorithms and core technologies of legacy security systems while introducing changes necessary to support extensible integration with XML. This allows interoperability with a wide range of existing infrastructures and across deployments.

This is followed by an overview of the following core XML Security standards:

**Integrity and signatures - XML Digital Signature**

**Confidentiality - XML Encryption**

**Key Management - XML Key Management Specification (XKMS)**


XML that may be processed by common XML tools. They may also explicitly define the structure of the documents they have defined, by creating an XML Schema or Document Type Definition (DTD). This allows documents to be validated.

Example 1 shows a language for managing office medical records, including XML elements like `<PatientRecord>`, `<Name>` and `<Diagnosis>`. It also shows the use of an XML Namespace associated with a lab, to allow a `<lab:Diagnosis>` element that does not conflict with the medical `<Diagnosis>` element.

```
<PatientRecord
   xmlns=http://www.medical.org/
   xmlns:lab=http://www.lab.org/>
   <Name>John Doe</Name>
   <Account>123456</Account>
   <Visit date="10pm March 10, 2002">
     <Diagnosis>Broken second metacarpal</Diagnosis>
     <lab:Diagnosis>
       <lab:encoded Xray image="lab:Xray>
     </lab:Diagnosis>
   </Visit>
</PatientRecord>
```

*Example 1 - Sample XML Document With XML Namespaces*

Security is vital to Hospital Management Information System. These requirements include Authentication, Authorization, Integrity, Signature, Confidentiality & Privacy and Digital Rights Management and are briefly summarized below:

**Authentication - Who is it?** Determine the identity or role of a patient or doctor or nurse or physician attempting to perform some action such as accessing a medical report.

**4.1. Authorization - What can they do?**

Determine whether a doctor or nurse or physician is allowed to perform a requested action, such as viewing a patient record, changing a patient data [13, 14, 15].

**4.2. Integrity - Ensure that information is intact**

Ensure that information is not changed, either due to malicious intent or by accident. This may be information transmitted over a network, such as from a web browser to a web server, information stored in a database or file system, or information passed in a web services message and processed by intermediaries, to give a few examples [13, 14, 15].

**4.3. Signature - Create and verify electronic signatures analogous to handwritten signatures**

Produce or verify an electronic signature intended to be the equivalent of a handwritten signature. Such a signature may be used for different purposes such as approval, confirmation of receipt, acceptance or agreement [13, 14, 15].

**4.4. Confidentiality & Privacy - Make content unreadable by unauthorized parties**

Ensure that content may only be viewed by legitimate parties, even if other access control mechanisms are bypassed. Confidentiality is generally associated with encryption technologies, although other approaches such as steganography (information hiding) might serve a similar purpose [13, 14, 15].

**4.5. Digital Rights Management - Limit use and sharing of content according to license agreements**

Ensure that content is used according to license agreements. Generally access rules are incorporated with the content, and enforcement controls are integrated with the clients needed to use the content. Traditionally, security technologies have required applications to be security or Public Key Infrastructure (PKI) "enabled" [13, 14, 15]. This often involves integrating specialized security code with the application in order to meet security requirements. This created a slow, cumbersome and inflexible customization process. An alternative is to create generic XML tools and generic XML Security and then allow them to be used with a variety of XML applications. This allows generic XML Security filters to be applied to arbitrary content without requiring extensive customization for each application, reducing costs and delay.

The XML Security standards define XML vocabularies for representing security information, using XML technologies, such as XML Schema, for
definition. XML Digital Signature allows XPath expressions to extract portions of XML for processing. XML Security technologies may be used in conjunction with transport security technologies, such as SSL/TLS, as well. Existing algorithms, such as the SHA1 digest algorithm, are also brought into the XML Security standards.

The core XML Security standards are:

1. XML Digital Signature for integrity and signatures,
2. XML Encryption for confidentiality,
3. XML Key Management (XKMS) for key management,
4. Security Assertion Markup Language (SAML) for making authentication and authorization assertions, and XML Digital Signature (XML DigSig)

Digital signatures are useful for two purposes:

1. To provide persistent content integrity, and
2. To create and verify portable electronic signatures

A digital signature associates a digest of the content with the signer of the content using a cryptographic technique. A digest is a digital "fingerprint", a short fixed-length value that is unique to the content and impractical to determine without the content. Using a cryptographic technique with the digest makes it hard for anyone other than the original signer to change the content without detection.

An XML <Signature> element may be handled in different ways, based on the desired application. It may be placed in a document apart from what is signed. This is known as a "detached" signature, and is used when signing non-XML content. When XML content is signed, the <Signature> element may be added to the XML. This is convenient, since signatures may then be bundled within the content and remain embedded with it, making it easy to keep track of them. When placed in an XML document, the <Signature> element may be added to the document being signed under the document element (an "enveloped" signature). In some cases, it is useful to place the content being signed within the <Signature> element, an example being a signature property (an "enveloping" signature).

If a signature is added to the <PatientRecord> as an enveloped signature, for example, the <Signature> element would be a child of the <PatientRecord> as shown in example 2.

When a signature is added to a document as part of the document, it changes the document. To verify the signature, it is necessary to compare the original document without the signature. The XML Digital Signature recommendation defines a mechanism for removing the <Signature> as part of the verification process.

```xml
<PatientRecord
 xmlns="http://www.medical.org/"
>
 <Name>John Doe</Name>
 <Account>123456</Account>
 <Visit date="10pm March 10, 2002">
 <Diagnosis>Broken second metacarpal</Diagnosis>
 </Visit>
 <Signature xmlns='http://www.w3.org/2000/09/xmldsig#'>
 ... 
 </Signature>
</PatientRecord>
```

Another possibility is to create a new XML document with a <Signature> document element and to place the signed element as a child of the <Signature> element. This is usually reserved for information associated with a signature, such as the purpose of the signature, as in example 3-

```xml
<Signature
 xmlns='http://www.xml3.org/2000/09/xmldsig#'>
 <SignedInfo DF </SignedInfo>
 <SignatureValue>120
 </SignatureValue>
 <Object>
 <SignatureProperties>
 <p:Purpose
 xmlns='http://www.myexample.com/schemas'>
 Approval
 </p:Purpose>
 </SignatureProperties>
 </Object>
</Signature>
```

An XML Digital Signature may be stored separately from the signed content (a detached signature) or embedded within the XML content that was signed (enveloped signature).

To continue with the earlier PatientRecord example, suppose that the entire PatientRecord is to be signed by the Doctors chamber, and the signature is to be maintained as part of the PatientRecord. This would
produce the following example 4, showing the layout of an XML Signature-

```xml
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
  <SignatureValue>
    <KeyInfo>
      <X509Data />
    </KeyInfo>
</Signature>
```

XML Security is the interaction of XML Digital Signatures and XML Encryption. Suppose you receive a document with an XML Signature and an `<EncryptedData>` element as in the following example 5 and example 6-

```xml
<PatientRecord xmlns="http://www.medical.org/
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
    <SignatureValue>
      <Reference URI="" />
    </Reference>
</Signature>
```

XML Key Management Specification (XKMS) [16] defines three specifications:

1. XML Key Registration Service Specification (XKRSS).
2. XML Key Information Service Specification (XKISS).
3. Protocol Bindings
   A client might generate a key pair and wish to register it with a trust server, as in the example 7. (Complete examples are provided with the XKMS specification)
A client could request an identity validation (example 9)-

```
<ValidateRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xmlns="http://test.xmltrustcenter.org/2001/01/Validate"
                     Service="http://test.xmltrustcenter.org/2001/01/Validate"
                     RequestID="12345678901234567890"
                     RuleSet="\"urn:ietf:rfc:2433\"
                     KeyFinding="\"\"
                     ResponseID="\"" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
```

The following response indicates that the certificates are valid (example 10)-

```
<ValidateResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xmlns="http://test.xmltrustcenter.org/2001/01/Validate"
                     Service="http://test.xmltrustcenter.org/2001/01/Validate"
                     RequestID="12345678901234567890"
                     RuleSet="\"urn:ietf:rfc:2433\"
                     KeyFinding="\"\"
                     ResponseID="\"" xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
```

3. Conclusion

The text of the paper is of two fold. First, it explains the object oriented existing generic architecture of the HL7. Second, it provides the extended secure architecture of HL7, which resolves the security threats at interaction point of the existing architecture. Finally the paper provides the basic templates of security of patient health record system.

References


