Chapter 13

Securing XML with Role-Based Access Control: Case Study in Health Care

Alberto De la Rosa Algarín
University of Connecticut, USA

Timoteus B. Ziminski
University of Connecticut, USA

Steven A. Demurjian
University of Connecticut, USA

Yaira K. Rivera Sánchez
University of Connecticut, USA

Robert Kuykendall
Texas State University, USA

ABSTRACT

Today's applications are often constructed by bringing together functionality from multiple systems that utilize varied technologies (e.g. application programming interfaces, Web services, cloud computing, data mining) and alternative standards (e.g. XML, RDF, OWL, JSON, etc.) for communication. Most such applications achieve interoperability via the eXtensible Markup Language (XML), the de facto document standard for information exchange in domains such as library repositories, collaborative software development, health informatics, etc. The use of a common data format facilitates exchange and interoperability across heterogeneous systems, but challenges in the aspect of security arise (e.g. sharing policies, ownership, permissions, etc.). In such situations, one key security challenge is to integrate the local security (existing systems) into a global solution for the application being constructed and deployed. In this chapter, the authors present a Role-Based Access Control (RBAC) security framework for XML, which utilizes extensions to the Unified Modeling Language (UML) to generate eXtensible Access Control Markup Language (XACML) policies that target XML schemas and instances for any application, and provides both the separation and reconciliation of local and global security policies across systems. To demonstrate the framework, they provide a case study in health care, using the XML standards Health Level Seven's (HL7) Clinical Document Architecture (CDA) and the Continuity of Care Record (CCR). These standards are utilized for the transportation of private and identifiable information between stakeholders (e.g. a hospital with an electronic health record, a clinic's electronic health record, a pharmacy system, etc.), requiring not only a high level of security but also compliance to legal

DOI: 10.4018/978-1-4666-4514-1.ch013

entities. For this reason, it is not only necessary to secure private information, but for its application to be flexible enough so that updating security policies that affect millions of documents does not incur a large monetary or computational cost; such privacy could similarly involve large banks and credit card companies that have similar information to protect to deter identity theft. The authors demonstrate the security framework with two in-house developed applications: a mobile medication management application and a medication reconciliation application. They also detail future trends that present even more challenges in providing security at global and local levels for platforms such as Microsoft HealthVault, Harvard SMART, Open mHealth, and open electronic health record systems. These platforms utilize XML, equivalent information exchange document standards (e.g., JSON), or semantically augmented structures (e.g., RDF and OWL). Even though the primary use of these platforms is in healthcare, they present a clear picture of how diverse the information exchange process can be. As a result, they represent challenges that are domain independent, thus becoming concrete examples of future trends and issues that require a robust approach towards security.

1. INTRODUCTION

Today's world is dominated by systems with a wide range of technological approaches (e.g. application programming interfaces, Web services, cloud computing, data mining, etc.), where one major objective is to support information sharing and exchange as applications are constructed as meta-systems (systems of systems), with new applications interfacing with multiple technologies, comprised of many interacting components. In such an environment, the one major challenge is to ensure that local security policies (of constituent systems) are satisfied not only when the application accesses a single system, but also when considered from a higher-level perspective. That is, an application's security is the combination of the security that must be attained within each constituent system that is accessed. What happens when security privileges of individual systems are in conflict with one another? How do we reconcile these local security policies? Is it possible to define a global encompassing security process or framework that provides a level of guarantee to the local security policies from an enforcement perspective? As today's applications continue to become more and more complex, interacting with many other systems (or applications) using varied technological paradigms, there will be a need to provide some degree of assurance that security for the application (global) satisfies the sum of the parts (local security of constituent systems). Information exchange has increased exponentially, due to the development of generic data standards (e.g., XML, JSON, RDF, OWL, etc.) and the ease of interconnection across systems, in domains such as biomedical, health informatics, library repositories, collaborative software development, etc. All of these domains present security challenges that, though not unique, have yet to be sufficiently addressed; often neither in the specific format or system (local security), and definitely not across multiple formats and meta-systems (global security).

In this effort to facilitate the intercommunication between heterogeneous systems, the *eXtensible Markup Language (XML)*[1] has become the de facto document standard for information exchange. In health care, which will serve as the case study for this chapter, XML is used for standards such as: the Health Level Seven's (HL7) Clinical Document Architecture (CDA) (Dolin, 2006) that underlies many Health Information Exchange (HIE) approaches; and, the Continuity of Care

Record[2] (CCR), used for storage of administrative, patient demographics, and clinical data. In Health Information Technology (HIT), the clinical document architecture and the continuity of care record come together in systems such as Electronic Health Records (EHR) and Personal Health Records (PHR) (e.g., Microsoft HealthVault[3]). The clinical document architecture is used to support health information exchange among hospitals, clinics, physician practices, laboratories, etc., with the continuity of care record providing the means to model the data that needs to be exchanged. As documents derived from standards such as these are circulated among various systems and made available to particular users with specific needs, we must expand security from each individual system to a focus that is more expansive in controlling the document and its content, particularly for health information exchange. Current approaches to security only do so from the system's perspective, in which the security policies that govern it are the final authority, and no consideration is given to the policies that govern the data repositories or constituent systems. This level of security is inadequate to scenarios such as information exchange in which the data utilized could not be owned by any particular user, but by an external party. Added to this is the rapidly emerging mobile applications domain where, in the case of health care, patients manage personal health information for chronic diseases, and a need to securely access information and authorize its exchange with medical providers via mobile applications, electronic health records, secure emails, or other means is a key concern. A solution that achieves this will require document-level access control of XML schemas to allow XML instances to appear differently to authorized users at specific times based on criteria that include, but are not limited to, a user's role, time and value constraints on data usage, collaboration for sharing data, delegation of authority as privileges are passed among authorized users, etc.

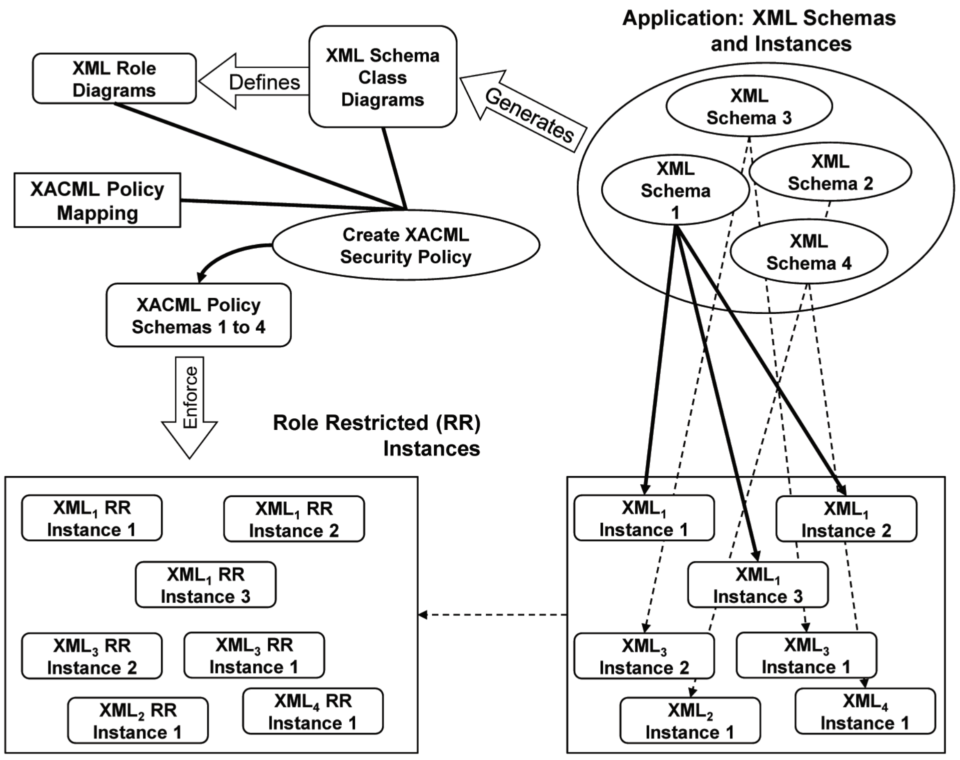
The challenge of attaining customized XML security enforcement necessitates the addressing of legal and adaptability requirements. In health care, the Health Insurance Portability and Accountability Act[4] (HIPAA) provides a set of security guidelines in the usage, transmission, and sharing of Protected Health Information (PHI); in e-commerce, there would be a need to protect Personally Identifiable Information (PII) including names, addresses, accounts, credit card numbers, etc. Protected health information and personally identifiable information must be strictly adhered to in many applications and settings. From an adaptability perspective, XML security policies must be defined at the XML schema level to support the definitions of users grouped in different roles, each with possible different sets of permissions that act on the specific parts of the information (an XML instance), across millions of records (XML instances). For the purposes of this chapter, we focus on the attainment of the National Institute for Standards and Technology's[5] (NIST) standard Role-Based Access Control (RBAC) (Ferraiolo, 1995, 2001) for XML, which would support the definition of security policies at the XML schema level (for example, the continuity of care record document (patient data) at the schema) that can then be used to specify (allow or deny) different permissions on certain portions of an XML instance (for example, a continuity of care record's instance), allowing the same instance to appear differently to specific users (patients and medical providers) acting in a chosen role at different times. To accomplish this, we leverage a secure software engineering process that promotes the consideration of security at an early stage and throughout the process. The usage of an XML schema via a new Unified Modeling Language (UML) schema diagrams requires a security framework for XML that allows the design, implementation, and deployment of enforceable security policies to allow access to XML instances to be precisely controlled by role. The definition of security at the

XML schema level via an external security policy separates the security from the XML instances, which avoids the overhead required when updating security policies that are otherwise embedded in instances and target a large amount of these.

In this chapter, we present our security framework for XML (De la Rosa Algarín, 2012) defined at the schema level and realized at the instance level through the creation and generation of eXtensible Access Control Markup Language (XACML)[6] policies, and demonstrate the work (design, mapping of policy and enforcement) via a case study of an in-house health care scenario composed of a set of health information technology applications. As shown in Figure 1, this generalized framework achieves granular security by taking the XML schemas and instances for any application (right hand side of Figure 1) and using them to define UML[7] diagrams for the respective XML schemas and the associated roles in order to create an enforcement XACML security policy that will be able to generate role-restricted (RR) instances that limit the information in the original instances based on the defined security (left hand side of Figure 1). Our approach provides separation of security concerns to tackle the challenge of changing security policies that can apply to millions of XML instances. In support of this framework, we leverage our prior work on secure software engineering using UML (Pavlich-Mariscal, 2008) and have created new UML diagrams: an XML Schema Class Diagram that captures the structure of the applications XML schemas; and an XML Role Slice Diagram that allows privileges on an XML schema's entities and attributes to be allowed/denied to different users by role at different times, thereby creating a role-restricted instance that is customized for that user. We note that Figure 1 is referring to any XML schemas, instances, and security definitions regardless of domain. In this chapter, after briefly reviewing our security framework for XML schemas and documents, we apply it to a case study of health care, consisting of the continuity of care record standard, utilized as the information exchange document, coupled with two in-house developed applications: the Personal Health Assistant (PHA), which consists of two mobile applications that support the exchange of information stored in a personal health record (Microsoft HealthVault) between patients and providers; and, SMARTSync (Ziminski, 2012), a medication reconciliation application, built as a meta-system utilizing Microsoft HealthVault and the Harvard SMART Platform[8], that generates a list of potential overmedication, adverse interactions, and adverse reactions for the patient and provider. All of these applications require that the XML that is delivered be restricted by role (a filtering of the content of the instance) in order to insure that only the authorized information is provided to the user. Our in-house mobile and Web apps both share the same server, with the Web app accessing another server, using well accepted Web standards (XML, RDF, JSON) for information modeling and exchange; thus our work is applicable to any such architecture. The use of these in-house applications supports our ability to apply and experiment with our XML security solutions with actual working systems that, while health care based, are just mobile/Web apps and servers.

The remainder of this chapter has five sections. In Section 2, background is provided on the NIST RBAC standard, XML, and the continuity of care record for the reader's benefit and understanding of the examples used throughout the chapter. In Section 3, we briefly describe our existing security framework for XML (De la Rosa Algarín, 2012) with a review of the new UML diagrams (XML Schema Class Diagram and XML Role Slice Diagram in Figure 1) for XML schemas and security definition, the generation of enforcement XACML policy schemas from these new diagrams, and relevant related work. Section 4 presents our case study in health care using two in-house developed applications, Personal Health Assistant (PHA) and SMARTSync, by detailing: the overall architecture and associated technologies (An-

Figure 1. Security framework and enforcement process for XML



droid[9], JSON[10], Microsoft HealthVault, and Harvard's SMART Platform), the Personal Health Assistant and SMARTSync applications, and the attainment of security for these applications using our security framework from Section 3. Note that while we utilize the health care domain as the case study for our demonstration, this security framework can be applied in any domain where the document structure to be secured is XML and an XML schema that validates the instances is available. Following our case study of the health care domain, Section 5 presents future trends by detailing a large scale view of health information technology systems and applications, with an emphasis on the interplay of health information exchange among the various systems; and the role of security at global and local levels across such a complex architecture. As part of this discussion, accessible health information technology platforms are explored, including: Microsoft HealthVault personal health record; Open *m*Health[11], which promotes mobile health via an open architecture; and, the Harvard SMART platform for substitutable medical applications that promote reuse, are explored. The wide range of open electronic health records, the myriad of XML standards, and the way that applications like Personal Health Assistant and SMARTSync interact to gather data effectively are also explored. These platforms, their role in the health information exchange process, and the large amount of data formats, standards and usage of data present concrete examples of research problems not unique to the health care domain, but present in domains that utilize information exchange, meta-systems or traditional system interoperability, as part of their daily workflow, requiring the intercommunication of information stored in different repositories with different formats and security policies. The end result is the recognition of a greater need for a comprehensive approach to security operating under information exchange. Towards this end, in Section 5, we also include a number of recommendations for the health care discipline and health information technology for improvements towards a more cohesive and shared future that promotes patient's health via electronic means. These recommendations, though directed to the health care domain as part of the case study, are presented in a general way so that the underlying, common application construction and interoperations issues are evident, and the proposed recommendation can be likewise achieved in this setting. We finish the chapter by offering concluding remarks in Section 6.

2. BACKGROUND

In support of this chapter, we provide background in three key areas: the National Institute for Standards and Technology (NIST) Role-Based Access Control (RBAC) model (Ferraiolo, 1995, 2001) which is intended to allow a user to be assigned permissions (read, write, etc.) to access objects (or portions of objects) based on his/her responsibilities as defined by a role; the eXtensible Markup Language (XML) a well-established standard for data representation that facilitates ease of exchange among users and systems; and, the health standard Continuity of Care Record (CCR) the represents data on patients (demographic, medications, allergies, medical history, etc.) using XML. Collectively, all three of these background areas establish the concepts and terms that are utilized throughout the paper. Health care is also an easy-to-understand domain, since most readers have experience with the stakeholders (medical providers) and their venues (offices, clinics, hospital, labs, etc.).

Role-based access control has long been utilized in the industry to represent permissions to an application based on a user's responsibilities. A role (e.g., Physician or Nurse) represents a category of permissions against objects (e.g., the way the role can access the data in a patient medical record), and by assigning permissions to roles, we can authorize users to roles against specific objects (e.g., Dr. Smith with Physician role can access objects of Patient Jones). When a role needs to change, we can change its permissions without impacting its authorization. The NIST RBAC (Ferraiolo, 1995, 2001) model organizes roles into different levels. First, $RBAC_0$ defines permissions on a role and authorizes a role to a user. Second, $RBAC_1$ allows for role hierarchies where permissions defined at the parent role can pass down to the child roles, e.g., the Nurse is a parent role with Staff_RN a role for taking care of patients, Discharge_RN a role for handling patient's upon leaving a hospital, Education_RN would teach patients about managing their chronic disease, etc. Third, $RBAC_2$ supports constraints, such as separation of duty and mutual exclusion, e.g., the roles Staff_RN and Physician are not allowed to assigned to the same individual (user); this prevents a user assigned a Staff_RN being assigned a Physician role in the future. Finally, from an authorization perspective, a user can be assigned multiple roles, but is only allowed to play a single role at any given time, which corresponds to the concept of sessions in $RBAC_3$, which provide the enforcement of permissions on specific objects authorized to a user playing a role at runtime. For example, Dr. Smith may have a Primary_MD role when treating patients in his practice while have an Attending_MD role when treating patients at a hospital.

XML is intended as a unifying means for data in terms of its representation to allow for it to be collected, transmitted, displayed, and exchanged among users and systems with ease. XML is a modeling language with the ability to define an *XML Schema* for the structure of the data being modeled (akin to a class in a UML diagram) which can then be instantiated to create *XML Instances* that are also referred to as XML documents. Collectively, a given application (like an electronic medical record) can have a set of XML schemas that describe the application and all of its instances. In this context, each XML schema serves as both the blueprint and validation agent for instances seeking to comply and be used for information representation and exchange. XML schemas support the definition of information to be hierarchically structured and tagged, and the tags themselves can be exploited to capture and represent the semantics of the information. The main modeling capability of XML schemas is the XML Schema Definition and associated XML Schema language. As an example, an XML schema can be composed of multiple xs:simpleType, xs:sequence, xs:element, etc, and these can be combined and nested in any way to form a more encompassing xs:complexType, a characteristic shared with classes in UML.

A continuity of care record document includes both protected health information and personally identifiable information such as demographics, social security number, insurance policy details, and health related information (such as medications, procedures, psychological notes, etc.). The continuity of care record schema defines all of the structure and interdependencies of information, but in practice, not all of the information at the schema level is available to all users based neither on role, nor at the instance level available to be written by some users based on role. For example, a Secretary role at a private practice performing financial operations might only need to see the patient's demographics and insurance policy details (personally identifiable information), whereas the Primary_MD role may to access the entire patient's information, but not the social security number. Select protected health information, such as psychiatric notes may not be available to the Primary_MD role, but be more constrained. Thus, when given information modeled using XML schemas (like the continuity of care record) and the associated instances (data for actual patients), the intent of the work presented in this chapter is to allow for the continuity of care record instances to be authorized to a user by role which will allow the instances appear differently at particular times and will also limit if the user (by the permissions of the role) will be able to read and/or write the authorized portions of an instance.

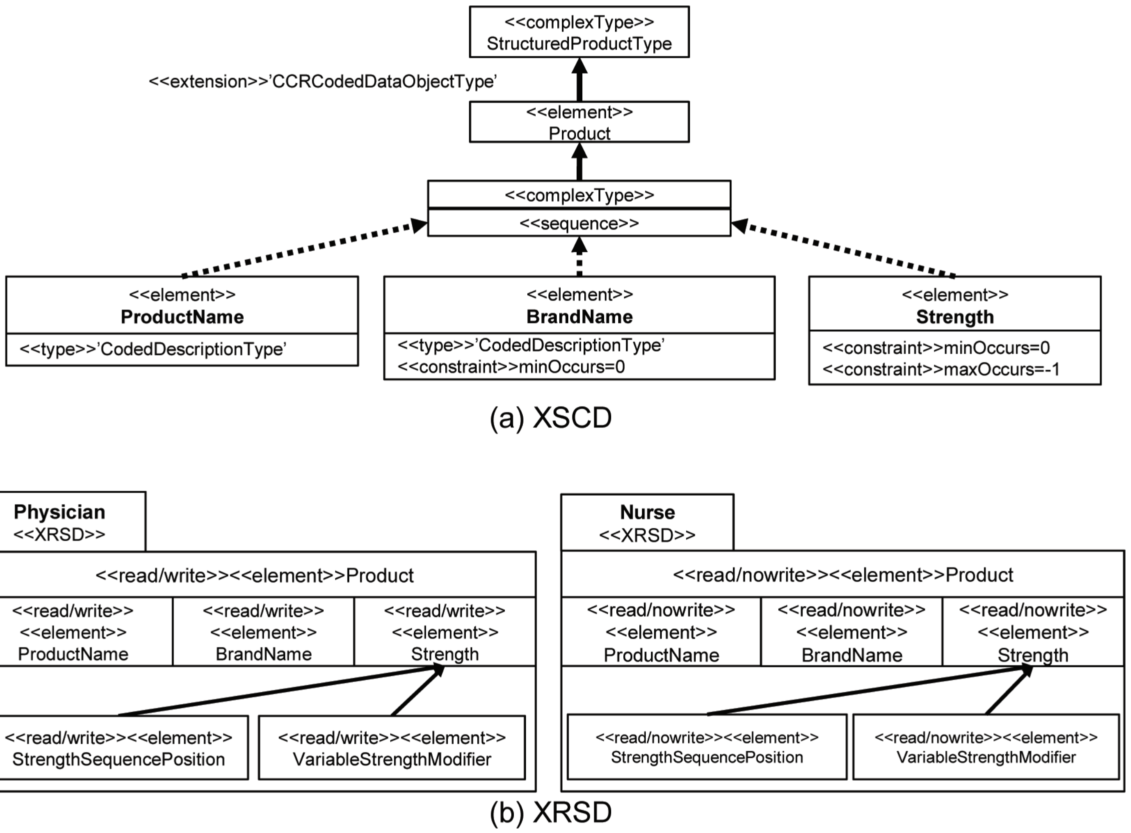
3. SECURITY FRAMEWORK FOR XML

Our security framework for XML schemas and instances (see Figure 1 again) separates the security policies from the schema by utilizing extended UML diagrams and a mapping algorithm that places the XACML policies at the same layer of the UML diagrams. These two diagrams, the XML Schema Class Diagram (XSCD) and the XML Role Slice Diagram (XRSD), are XML representative artifacts in the UML model, as we detail in Section 3.1, to address XML security from a software engineering perspective. Tackling the problem this way allows for the change of policies affecting large numbers of XML instances without the inherent cost of updating each instance. With our framework, designers can follow both a secure software engineering approach (Pavlich-Mariscal, 2008), and a secure information engineering approach for a more complete and secure solution. As a result, from the XSCD and XRSD artifacts, we generate a XACML policy that can enforce the defined security at the schema level, as we present in Section 3.2. To complete the discussion, Section 3.3 reviews related research. Note that we again stress that the security approach that is being demonstrated focuses on XML schemas and instances, and the generation of XACML policies; health care is simply an explanation vehicle.

3.1. XML Schema Class and Role Slice Diagrams in UML

UML provides multiple diagrams to visually model applications, but there is a lack of integrating security. Our prior work has defined new UML security diagrams for supporting RBAC (Pavlich-Mariscal, 2008) via the UML meta-model. Using this as a basis, we have extended this work to define two new UML artifacts (De la Rosa Algarín, 2012): the XML Schema Class Diagram (XSCD) in Figure 2a that contains architecture, structure characteristics, and constraints of an XML schema; and, the XML Role Slice Diagram (XRSD) in Figure 2b which has the ability to add permissions to the various elements of the XSCD, i.e. read/write, read/nowrite, noread/write, noread/nowrite. The set of all XML schemas for a given application are converted into a corresponding set of XSCDs. As a result, we provide secure software engineering to the XML design process where the creation of an XML schema is placed into the UML context alongside other diagrams. XSCD, in Figure 2a, presents the way that the XSCD for the continuity of care record's schema xs:complexType 'StructuredProductType' would be represented in an UML-like XSCD diagram. The XSCD allows for customized access control policies to be generated for the respective concepts of the XML schema. The XRSD in Figure 2b is capable of applying access control policies or permissions on the attributes of the XSCD based on role, thereby achieving fine-grained control. Permissions on XML documents are read, no read, write, and no write permissions with respective stereotypes, <<read/write>>, <<read/nowrite>>, <<noread/write>>, and <<noread/nowrite>>. Figure 2b defines Physician and Nurse XRSDs with permissions against the XSCD in Figure 2a. Note that in Figure 2b, the continuity of care record's complexType 'StructuredProductType' element Product allows a Physician role all of the

Figure 2. XSCD of a continuity of care record schema segment (a) and XRSD of the XSCD in a health care scenario (b)


<<complexType>>
StructuredProductType
<<extension>>'CCRCodedDataObjectType'
<<element>>
Product
<<complexType>>
<<sequence>>
<<element>>
ProductName
<<type>>'CodedDescriptionType'
<<element>>
BrandName
<<type>>'CodedDescriptionType'
<<constraint>>minOccurs=0
<<element>>
Strength
<<constraint>>minOccurs=0
<<constraint>>maxOccurs=-1
(a) XSCD
Physician
<<XRSD>>
<<read/write>><<element>>Product
<<read/write>> <<element>> ProductName
<<read/write>> <<element>> BrandName
<<read/write>> <<element>> Strength
<<read/write>><<element>> StrengthSequencePosition
<<read/write>><<element>> VariableStrengthModifier
Nurse
<<XRSD>>
<<read/nowrite>><<element>>Product
<<read/nowrite>> <<element>> ProductName
<<read/nowrite>> <<element>> BrandName
<<read/nowrite>> <<element>> Strength
<<read/nowrite>><<element>> StrengthSequencePosition
<<read/nowrite>><<element>> VariableStrengthModifier
(b) XRSD

information on a drug and be able to create new instances following the continuity of care record schema, with the Nurse role limited to read the drug details and cannot create new records. Note that the XSCD (Figure 2a) and the XRSD (Figure 2b) do not cover the whole continuity of care record schema representation due to space limitations.

3.2. Generating XACML Policies from XSCD and XRSD

As given in Figure 2b, XRSDs act as the blueprint of the access-control policy for reading and writing permissions for a specific element or component of an XML schema for any given role, and are used to represent the portions of the application's XSCD (Figure 2a) that are to be allowed (or denied) access at an instance level to create role restricted instances (Figure 1), which can then be used to generate an XACML policy using the XACML Policy Mapping process in Figure 3. The architecture has a number of components: Policy Enforcement Point (PEP) allows a request to be made on a resource (a user playing a Physician role to access an continuity of care record instance); Policy Decision Point (PDP), which evaluates the request and provides a response according to the policies in place (evaluate if a Physician role can access (read and/or write) a portion of a continuity of care record schema); the Policy Administration Point (PAP) is utilized to write and manage policies (a realization of the XRSD against the continuity of care record schema and its associated instances); and, the Policy Information Point (PIP) to arbitrate very fine grained security issues (control access to psychiatric data). To map the XRSD in Figure 2b into an XACML policy, we utilize an XACML PolicySet to make the authorization decision via a set of rules in order to allow for access control decisions that may contain multiple Policies, and each Policy contains the access control rules. Note that multiple XACML Polices may be generated, resulting in a PolicySet for a specific set of XML schemas that comprise a given application. Our prior work (De la Rosa Algarín, 2012) has all of the details for this mapping process to generate XACML policies; and while we omit this discussion due to length considerations, in Figure 4 we present the generated XACML policy for the Physician XRCD in Figure 2b.

Briefly, we explain the generated XACML. First, the Policy's PolicyId attribute value is the Physician XRSD is concatenated to 'AccessControlPolicy'; the Rule's RuleId attribute value is the Physician XRSD value concatenated to the XRSD's higher order element (in Figure 4 it would be Product as defined in the XSCD in Figure 2b) and concatenated to 'ProductRule'; the Rule's Description value is the Physician XRSD is concatenated to 'Access Control Policy Rule'; and, the XACML Policy and Rules target and match the role (*Subject*, e.g., *Physician* in Figure 2b and 4), the schema elements (*Resources*, e.g., ProductName, BrandName and Strength in Figure 2a, 2b and 4), and the permissions (*Actions*, e.g., read and write in Figure 2b and 4). Second the

Figure 3. XACML mapping from XRSD's and enforcement architecture

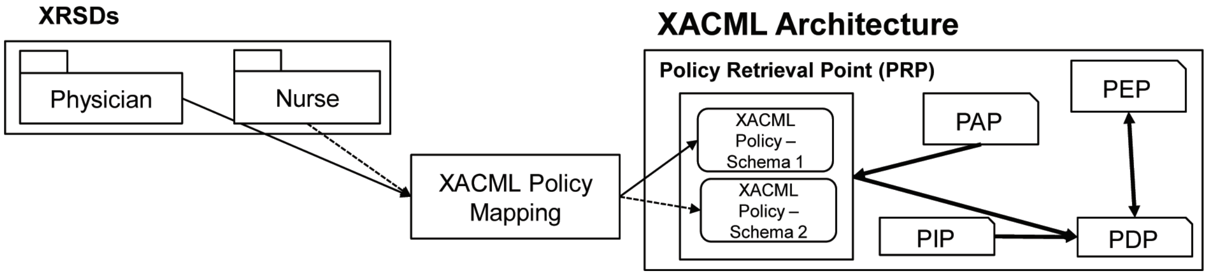


Figure 4. Mapped XACML policy from physician XRSD

```
<?xml version="1.0" encoding="UTF-8"?>
   <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os”
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance”
     xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
    http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
    xmlns:md="http:www.med.example.com/schemas/record.xsd“
    PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:PhysicianAccessControlPolicy”
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
    <Target/>
       <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:PhysicianProductRule“ Effect=“Permit">
       <Description>Physician Access Control Policy Rule</Description>
       <Target>
          <Subjects>
             <Subject>
             <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                  Physician
             </AttributeValue>
             <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role“
               DataType="http://www.w3.org/2001/XMLSchema#string"/>
             </SubjectMatch>
             </Subject>
          </Subjects>
          <Resources>
             <Resource>
             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    ccr:schema:product:productname
                </AttributeValue>
                <ResourceAttributeDesignator
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
             </ResourceMatch>
             </Resource>
             <Resource>
             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    ccr:schema:product:brandname
                </AttributeValue>
                <ResourceAttributeDesignator
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
             </ResourceMatch>
             </Resource>
             <Resource>
             <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    ccr:schema:product:strength
                </AttributeValue>
                <ResourceAttributeDesignator
                  AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
                  DataType="http://www.w3.org/2001/XMLSchema#string"/>
             </ResourceMatch>
             </Resource>
          </Resources>
          <Actions>
             <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                     read
                   </AttributeValue>
                   <ActionAttributeDesignator
                           AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-read"
                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </ActionMatch>
             </Action>
             <Action>
                <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                   <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                     write
                   </AttributeValue>
                   <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-write"
                     DataType="http://www.w3.org/2001/XMLSchema#string"/>
                </ActionMatch>
             </Action>
          </Actions>
       </Target>
       </Rule>
   </Policy>
```

XACML *Subject* Physician is identified as an attribute. Third, the resources are identified; namely, the *AttributeValue's* value is the Physician XRSD's element names from the XSCD (e.g., ProductName, BrandName and Strength in Figure 2a, 2b and 4). Finally, the XACML *Actions* as operations and values (read and write in Figure 2b and 4) are defined. The end result in Figure 4 is an XACML policy that when applied to a continuity of care record instance for the Physician role will generated a role restricted XML instance that limits the visibility and usage of the continuity of care record instance for a particular patient.

3.3. Related Work

In this section, we present related work in a number of areas. First in, XML security frameworks, one effort on enterprise resource planning consists of an integrated packaged software that serves as a single solution for database and communication utilizing XML (Chandrakumar, 2012) by focusing on the XML Signature specification (Ardagna, 2007), and another effort (Ammari, 2010) presents an architecture capable of handling the receiving of XML messages from heterogeneous systems. Second, in embedded XML security, the work of (Damiani, 2000) presents an access control system that embeds the definition and enforcement of the security policies in the structure of the XML documents in order to provide customizable security using document type definitions (outmoded XML) that incurs high overhead since security changes impact all instances, while the work of (Damiani, 2008.) details a model that combines the embedding of policies and rewriting of access queries to provide security to XML datasets.

Third, in XML and access control, one effort (Bertino, 2002; Bertino, 2004) presents Author-X, a Java-based system for discretionary access control in XML documents (using document type definitions) that provides customizable protection to the documents with positive and negative authorizations. A second effort (Leonardi, 2010) considers the scenario of a federated access control model, in which the data provider and policy enforcement are handled by different organizations, while a third effort (Kuper, 2005) presents a model consisting of access control policies over a document type definition with XPath expressions in order to achieve XML security. Last, the work of (Müldner, 2009) uses an approach of supporting RBAC to handle the special case of role proliferation, which is an administrative issue that happens in RBAC when roles are changed, added, and evolve over time, making security of an organization difficult to manage. Finally, in encryption-based XML security, the XML Security Working Group[12] (SWG) works on three different security aspects: XML signatures, XML encryption, and XML Security Maintenance, a second effort (Bertino, 2002) encrypts different sections of an XML document with different encryption keys which are distributed to the specific users based on the access control policies in place, and a third effort (Rahaman, 2008) presents a distributed access control model for collaborative environments where XML documents are used.

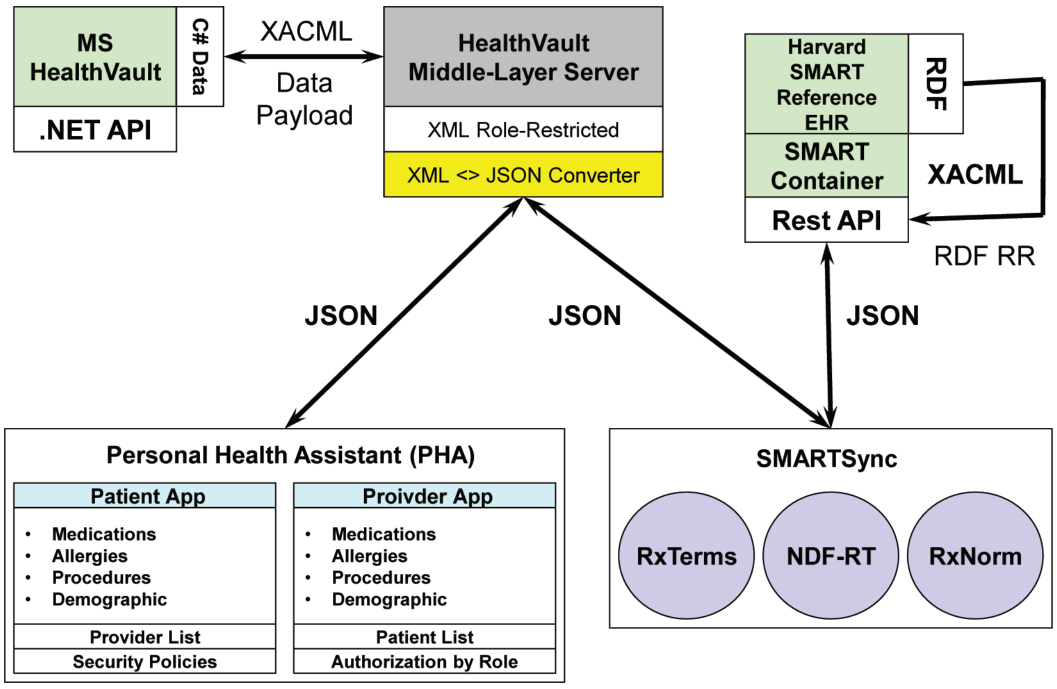
4. CASE STUDY OF HEALTHCARE APPLICATIONS

In this section, we present a case study of attaining security in XML for two in-house developed health applications, demonstrating the generation and enforcement of XACML policies on XML instances based on an a subset of the continuity of care record schema. The first, a mobile health application, the Personal Health Assistant (PHA), consists of two perspectives for medication management. One perspective allows a patient to keep track of their medications, nutritional supplements, allergies, etc., and also authorize that protected health information (continuity of care record information), which is stored in Microsoft HealthVault, to his/her specific medical providers at different times. The second perspective allows a provider to select and view the authorized protected health information on a patient-by-patient basis as determined by his/her assigned role. The second application, SMARTSync for medication reconciliation (Ziminski, 2012), takes patient medications from HealthVault and the Harvard SMART Platform Reference Electronic Health Record and from this information is able to generate a summary list of medications/supplements added by patients (in HealthVault) with those prescribed by a patient's medical provider. The intent is to generate a color-coded list of potential overmedication, adverse interactions, and adverse reactions for the patient and provider. Both applications have been coded by undergraduate, masters, and doctoral students as part of research related to biomedical and health informatics and its security and interoperability issues. The remainder of this section begins the case study by presenting the overall architecture of Personal Health Assistant and SMARTSync in Section 4.1. Then, Personal Health Assistant and SMARTSync are described in Sections 4.2 and 4.3, respectively, with a focus on their functional capabilities and user interfaces. Finally, in Section 4.4, we explore the way that XACML based security is achieved in the Personal Health Assistant application (where the documents to secure are XML instances) and in SMARTSync (where information is represented in RDF/XML and JSON-DL that must then be converted to XML in order to allow the information to be appropriately secured). Note, from a generalized perspective, we have a mobile app (PHA) and Web-based app that both interact with a server (MSHV) using JSON with XML conversion occurring to retrieve data entered by the end-user (patient), with the Web app (SMARTSync) also interacting with another external system (SMART EHR) which is effectively a database controlled by a third party (physician's office with patient data). If you reread the prior sentence without the parenthetical remarks, you have a mobile app and Web app interacting with one server, and the Web app interacting with another, all with information flowing with standard formats (XML, JSON, RDF); clearly the architecture is generalizable in this way to many other business and industrial domains.

4.1. Overall Architecture

The overall architecture of the two healthcare applications is given in Figure 5, where the bottom of the figure indicates Personal Health Assistant and SMARTSync. Microsoft HealthVault acts as the data source (server) for both applications, and stores information in a proprietary format which to be exported via a .NET API which can then be used to generate a continuity of care record compliant document in XML. The HealthVault Middle-Layer Server (center of Figure 5) acts as the contained solution of policy access, information, decision, and enforcement points (see right hand side of Figure 3). The XACML policies created and stored in the account of each respective user limits access to HealthVault through the HealthVault Middle-Layer Server, which handles the requests (where data is sent as JSON) of both applications. To store the relations between the authorized list of providers and their respective patients, the Middle-Layer Server uses MySQL[13]. JSON is utilized for the communication of the two applications and the Middle-Layer Server, allowing us to insure a uniform communication with any application (not only with Personal Health Assistant) that can be created for users. The communication between the Personal Health Assistant (patient version) and the Middle-Layer Server is done with unmodified JSON objects, while the communication between the Personal Health Assistant - (provider) version and SMARTSync and the Middle-Layer Server is a combination of unmodified (for the initial request of patients) and filtered (for the resulting data allowed by the policies enforced) JSON. From HealthVault, XML role restricted instances are generated. Requests done by the provider application determine the format of the data. If a provider is requesting

Figure 5. Medication management and reconciliation applications



information in the patient's continuity of care record document, then data from HealthVault is exported as a continuity of care record schema compliant XML document with policy enforcement performed, whereas any input from the provider to HealthVault is first received as a JSON payload, converted to an XML document based on the continuity of care record schema, enforced with policies (Section 4.3), and once authorized, translated to HealthVault objects for write back. A similar process occurs on the SMARTSync side to merge and save the data from HealthVault and SMART Reference Electronic Health Record (another server) back into HealthVault.

4.2. Personal Health Assistant (PHA)

Personal Health Assistant (PHA) is an in-house developed mobile (not publicly available), test-bed Android application for medication management that allows: patients to view and update their personal health record stored in their HealthVault account and authorize medical providers to access certain portion of protected health information; and, for providers to obtain the permitted information from their respective patients that they have been authorized to view. The patient version of Personal Health Assistant allows users to perform a set of actions regarding their health information. Users can view and edit their medication list, allergies, observations of daily living, and set security policies for read/write permissions on their medical providers by role per the discussion in Section 3. Security settings can be set at a fine granular level, and each provider gets view/update authorizations to the different information components available in Personal Health Assistant. The provider version of Personal Health Assistant allows the users (health professionals or medical providers) to view and edit the medical information of their patients as long as they are permitted to do so as dictated by the security set by the user (patient).

4.3. SMARTSync Application

SMARTSync is an in-house developed (not publicly available), Web-based test-bed medication reconciliation application used to create and preserve a patient's medication list through transfers among locations of care, preventing immediate interactions, and avoiding dosage errors in situations where brand and generic drugs are received or multi-component drugs are used (Barnsteiner, 2005; Poon, 2006). Significant risks include (Huang, 2004): *overmedication* when a provider prescribes a new medication (or one from the same class) or when an interacting medication is prescribed; *adverse interactions*, the result of conflicts between medications, which can change effect strength or serum concentration; and *adverse reactions*, allergic/other effects, experienced by patients which can result in a patient being wrongly labeled as allergic to a medication, unnecessarily excluding it as a treatment option in the future. To accomplish this, we gather data form HealthVault and SMART Reference Electronic Health Record as shown in Figure 5. Any medical data source (e.g., an electronic medical record, a personal health record, etc.) can be turned into a SMART container by exposing the SMART REST API, the SMART Connect API, and the related RDF/XML based data model[14].

In the SMART framework, applications are grouped on the SMART dashboard, which offers authentication and a set of basic services based on RDF/SPARQL for accessing the underlying medical data source in the SMART container. SMARTSync is also operated through this user interface component. In addition, SMARTSync communicates with HealthVault and takes advantage of the RxNorm, RxTerms, and the National Drug File – Reference Terminology[15] nomenclature/terminologies for semantic navigation of clinical drugs. The graphical user interface for SMARTSync is designed provide the alert information to the user in a quick and easily recognizable fashion, geared towards simplicity in order to serve a wide range of patients and to be easily portable to mobile devices. The main application screen is currently divided in two tabs, visualizing the personal health record (HealthVault) and the SMART Reference Electronic Health Record. Patients can switch between the tabs to see the list of medications stored in each record. The *Reconcile Medications* and the *Find Medication Interactions* buttons perform on-demand reconciliation and interaction searches. In the HealthVault tab, the user is presented with the reconciled list of medications. If any of the entries interact, the severity of interaction is indicated by a yellow (significant interaction) or red (critical interaction) background. Entries for which no interactions are found are displayed with a neutral background color. There are up to three buttons located next to each of the medications, over the counters, and natural supplements on either tab: *View Interactions*, *Details*, and *Remove*. Since a patient cannot modify the information located in the provider's EMR, the only button visible in this tab is *Details*. *View Interactions* presents the user with a listing of cross-interactions between the specified medication (over the counter/natural supplement) and any other reconciled entry. *Details* presents information of the medication ingredients, generic names, and the dates when the user started and stopped taking the medication. *Remove*, only available in the personal health record tab, allows the user to permanently delete the medication from their personal health record.

4.4. Achieving Security in Personal Health Assistant and SMARTSync

Securing the protected health information in Personal Health Assistant and SMARTSync is accomplished by utilizing the new UML-like XSCD and XRSD diagrams that define the security (see Section 3.1) in order to generate the XACML security policies (see Section 3.2). While

Personal Health Assistant strictly uses HealthVault to store and retrieve information, SMARTSync (by the nature and objective of the SMART Platform) is capable of obtaining information from heterogeneous data sources that do not share the same XML standard. These two cases present the diversity of formats and standards (sometimes equivalent, often non-equivalent) on which not only the health care domain operates, and must be considered in order to effectively secure information that is being exchanged in different formats among a range of health information technology systems. This approach of using XML to exchange and share information is occurring using a mobile app, a Web app, and multiple servers; this is a very typical model for any application domain. In the remainder of this section, we describe the way that the XACML policy is enforced when handling reading and writing requests on XML instances whose schema has been secured in Personal Health Assistant, as well as the realization of the security framework in SMARTSync which requires additional steps to deal with additional data formats.

Providing security on the continuity of care record utilized by Personal Health Assistant is achieved by the enforcement in the HealthVault Middle-Layer server (see Figure 5). The read and write operations to be enforced are initialized by the provider perspective of Personal Health Assistant, handled by the HealthVault Middle-Layer server, and realized in the generated XACML (see Figure 4 in Section 3.2). When a request is initiated from a provider to read the protected health information of a patient, the Middle-Layer Server retrieves the patient's information exported as a continuity of care record along with the targeting XACML policy. After this step, enforcement is performed and those elements with read permissions denied for the provider are filtered out and deleted from the continuity of care record using the XACML policy (Figure 4). Once this has occurred, the filtered instance of the patient's continuity of care record is then converted into an equivalent JSON object for Personal Health Assistant utilization; JSON is utilized to provide a common abstraction layer in data model for any other developed application that wishes to utilize HealthVault data. Consider an example scenario where a user with a role of Nurse is requesting information on a patient's personal health record. The permission of read for the Nurse role has been allowed for medications and allergies, and denied for medical procedures. The permission of write has been disallowed for all data elements. When a nurse utilizes the provider's Personal Health Assistant, s/he selects the patient named Jane Doe. As explained, the Middle-Layer Server retrieves the Jane Doe continuity of care record along with the XACML policy, and enforces security by filtering the continuity of care record as directed by the XACML policy. The filtered continuity of care record is then converted into a JSON object so that the Personal Health Assistant application can present the information to the user.

The steps to enforce security on writing operations done by a provider are similar. Starting with a write-back request with the JSON payload of new information, the Middle-Layer Server utilizes the XACML (see Figure 4) to evaluate which elements the provider is allowed to update. Only these elements are then updated in the continuity of care record, which goes through a validation process with the continuity of care record schema (for consistency in structure and integrity), and then written back to HealthVault in their respective objects. If the user requesting a write operation has a role with a permission that allows it to occur, the continuity of care record instance is updated with the sent data, and validated with the continuity of care record schema before the write-back to HealthVault. If validation against the schema is successful, then the write-back occurs, and the update performed by the provider is saved in the patient's HealthVault record. If the requester has a role that is not allowed to perform writing operations on the desired element, the Middle-Layer Server drops the request. Our approach provides a means for updating XML documents (in this case continuity of care record instances) that is controlled via an XACML security policy with the assistance of the Middle-Layer Server.

While HealthVault provides the information in continuity of care record, the SMART Platform's data model is capable of providing information in RDF/XML[16], N-TRIPLES[17], TURTLE[18] and JSON-LD[19]. RDF[20], which is a semantically augmented extension to XML, shares similar design, structure and hierarchical characteristics. The RDF/XML format provides XML syntax for RDF. This syntax is defined with respect to the XML namespaces, information set, and base. By using N-TRIPLES, the formal grammar for RDF/XML is annotated from the RDF graph. N-TRIPLES is an RDF graph-serializing format that enables the precise recording of the RDF graph mapping to machine-readable form. TURTLE allows the writing of RDF graphs in textual form, consisting of directives and triple-generating statements. Finally, JSON-LD is a linked data format utilized to provide context to data. Based on JSON, JSON-LD is capable of augmenting RESTful[21] services into providing data to the semantic-Web (Lanthaler, 2012). To secure the information obtained from the SMART Platform that is utilized by SMARTSync, we make use of the JSON-LD format. While an RDF/XML instance is at its core an XML instance annotated with RDF, it lacks a unique serialization from which an XML schema can be abstracted. That is, multiple XML schemas exist that validate against the different RDF/XML serializations. This presents a scalability problem in our approach, as we only consider a unique and valid XML schema to secure. The use of JSON-LD provides a unique JSON representation from which an equivalent XML instance can be generated using a variety of tools that are available for this purpose. To properly apply our security framework to JSON-LD, we first apply an XML transformation to the JSON-LD instance. Since JSON-LD is extended JSON, any JSON to XML transformation tool will do the conversion and create an equivalent XML document, from which an XML schema can then be generated. To demonstrate, the Figure 6a has JSON-LD for the medication AMITRIPTYLINE (for depression), while the right hand side has the resulting XML instance. Since the generated XML instance only has one serialization, the one obtained from the transformation operation, abstracting a unique XML schema that can validate is possible using an XML schema generator or tool, e.g., Microsoft's Visual Studio[22], Stylus Studio[23], Eclipse's Oxygen XML Plugin[24], Trang[25], etc. This XML Schema abstracted from AMITRIPTYLINE instance is shown in Figure 7a.

To complete the process, we again leverage the XSCD and XRSD's from Section 3.1 to generate XACML Policies (using the process in Section 3.2). In Figure 7, the XML Schema for the medication (Figure 7a) is then enforced using an XACML policy (Figure 7b). The XACML Policy only changes, with respect to the continuity of care record targeting policy, in the resources and their references. Note that we utilize the same color-coding scheme from Section 3.2 to illustrate the different aspects of the XACML with respect to the shading for policy, and blue and red lettering for read and write, respectively. While the SMART Platform does not currently support writing data back to the data sources, we still provide the mechanism to enforce security on write operations. That is, the Action elements in the XACML policy are still defined for read and write operations (and evaluated to Deny or Permit based on the credentials deduced from the XRSD). The SMARTSync example clearly illustrates that it is possible for our XACML security framework to work in many different settings, as long as there are tools available to allow the data translation to occur and the appropriate XML schema to be generated.

Figure 6. SMART JSON-LD for medication (a) and transformed XML instance (b)

```
{
 "@context": "…/contexts/smart_context.jsonld",
 "@graph": [
  {
   "@type": "Medication",
   "belongsTo": {
    "@id": "http://sandbox-api.smartplatforms.org/records/2169591"
   },
   "drugName": {
    "@type": "CodedValue",
    "code": {
     "@id": "http://purl.bioontology.org/ontology/RXNORM/856845"
    },
    "dcterms__title": "AMITRIPTYLINE HCL 50 MG TAB"
   },
   "endDate": "2007-08-14",
   "frequency": {
    "@type": "ValueAndUnit",
    "unit": "/d",
    "value": "2"
   },
   "instructions": "Take two tablets twice daily as needed for pain",
   "quantity": {
    "@type": "ValueAndUnit",
    "unit": "{tablet}",
    "value": "2"
   },
   "startDate": "2007-03-14"
  },
  {
   "@id": "http://purl.bioontology.org/ontology/RXNORM/856845",
   "@type": [
    "spcode__RxNorm_Semantic",
    "Code"
   ],
   "dcterms__identifier": "856845",
   "dcterms__title": "AMITRIPTYLINE HCL 50 MG TAB",
   "system": "http://purl.bioontology.org/ontology/RXNORM/"
  }
 ]
}
```

(a) JSON-LD

```
<?xml version="1.0" encoding="UTF-8" ?>
        <@context>
                …/contexts/smart_context.jsonld
        </@context>
        <@graph>
                <@type>Medication</@type>
                <belongsTo>
                        <@id>http://sandbox-api.smartplatforms.org/records/2169591</@id>
                </belongsTo>
                <drugName>
                        <@type>CodedValue</@type>
                        <code>
                                <@id>http://purl.bioontology.org/ontology/RXNORM/856845</@id>
                        </code>
                        <dcterms__title>AMITRIPTYLINE HCL 50 MG TAB</dcterms__title>
                </drugName>
                <endDate>2007-08-14</endDate>
                <frequency>
                        <@type>ValueAndUnit</@type>
                        <unit>/d</unit>
                        <value>2</value>
                </frequency>
                <instructions>Take two tablets twice daily as needed for pain</instructions>
                <quantity>
                        <@type>ValueAndUnit</@type>
                        <unit>{tablet}</unit>
                        <value>2</value>
                </quantity>
                <startDate>2007-03-14</startDate>
        </@graph>
        <@graph>
                <@id>http://purl.bioontology.org/ontology/RXNORM/856845</@id>
                <@type>spcode__RxNorm_Semantic</@type>
                <@type>Code</@type>
                <dcterms__identifier>856845</dcterms__identifier>
                <dcterms__title>AMITRIPTYLINE HCL 50 MG TAB</dcterms__title>
                <system>http://purl.bioontology.org/ontology/RXNORM/</system>
        </@graph>
```

(b) Transformed XML

Figure 7. Segment of XML schema (a) and a segment of the targeting XACML policy (b)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="graph">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="type"/>
        <xs:element ref="belongsTo"/>
        <xs:element ref="drugName"/>
        <xs:element ref="endDate"/>
        <xs:element ref="frequency"/>
        <xs:element ref="instructions"/>
        <xs:element ref="quantity"/>
        <xs:element ref="startDate"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="belongsTo" type="id"/>
  <xs:element name="drugName">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="type"/>
        <xs:element ref="code"/>
        <xs:element ref="dctermstitle"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="code" type="id"/>
  <xs:element name="dctermstitle" type="xs:string"/>
  <xs:element name="endDate" type="xs:NMTOKEN"/>
  <xs:element name="frequency">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="type"/>
        <xs:element ref="unit"/>
        <xs:element ref="value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="instructions" type="xs:string"/>
  <xs:element name="quantity">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="type"/>
        <xs:element ref="unit"/>
        <xs:element ref="value"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="startDate" type="xs:NMTOKEN"/>
  <xs:element name="type" type="xs:NCName"/>
  <xs:complexType name="id">
```

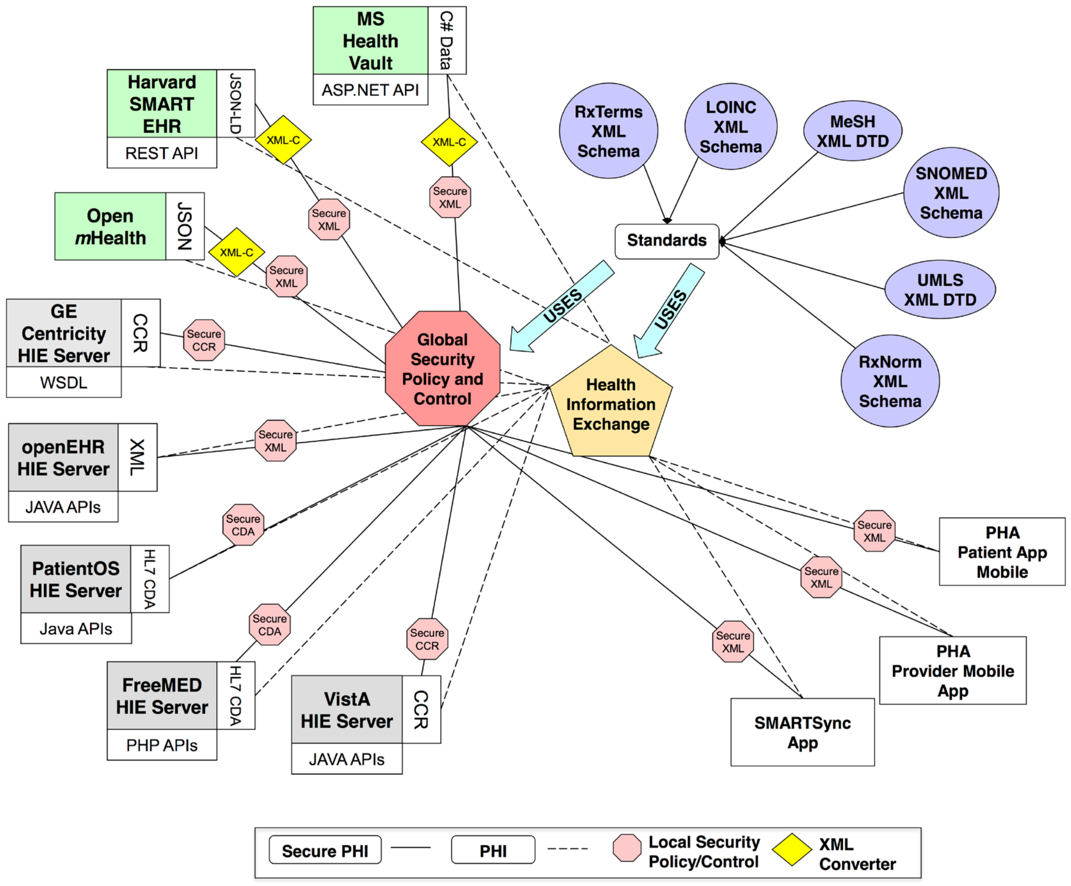
```
<?xml version="1.0" encoding="UTF-8"?>
  <Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
  xmlns:md="http:www.med.example.com/schemas/record.xsd"
  PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:PhysicianAccessControlPolicy"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Target/>
    <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:PhysicianProductRule"
           Effect="Permit">
    <Description>Physician Access Control Policy Rule</Description>
    <Target>
      <Subjects>
        <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Physician
        </AttributeValue>
        <SubjectAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
        </Subject>
      </Subjects>
      <Resources>
        <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            generated:schema:graph:type
          </AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
        </Resource>
        <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            generated:schema:graph:belongsTo
          </AttributeValue>
          <ResourceAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
        </Resource>
        <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            generated:schema:graph:drugName
          </AttributeValue>
          <ResourceAttributeDesignator
```

5. FUTURE TRENDS AND RESEARCH DIRECTIONS

Future trends and research directions in security are related to taking a high-level view of the information data exchange process, in general, and its application to the healthcare domain, in particular. Our focus in this section, with respect to health care as shown in Figure 8, considers all of the different health information technology systems and relevant standards that are utilized in the care and treatment of patients, with an emphasis on the interplay of health information exchange among various health information technology systems with security at global and local levels across such a complex architecture. Content wise, the lower left of Figure 8 contains open electronic health record systems (openEHR[26], PatientOS[27], VistA[28]) that all share an ability to export patient data in XML formats (XML, continuity of care record, and clinical document architecture); this is in contrast to commercial electronic health records (GE Centricity) which often have proprietary formats that hinder health information exchange. The upper left of Figure 8 contains various emerging platforms: Open *m*Health to promote mobile health via an open architecture, the Harvard SMART platform for substitutable medical applications that promote reuse), and Microsoft HealthVault, a personal health record. Open *m*Health and SMART have JSON and JSON-LD, respectively, to model patient data, which must be converted (XML-C diamond) before it can be secured. HealthVault has .NET classes that can export to XML instances. The bottom right contains the medical applications that must be securely managed, Personal Health Assistant and SMARTSync, as reviewed in Section 4. The upper right contains the various standards and services involving medications (RxTerms[29] and RxNorm[30]), medical codes (SNOMED[31]), medical nomenclature (UMLS[32] and MeSH[33]), and laboratory codes (LOINC[34]) that are used by all of the systems and applications. Again, we note that the architecture shown in Figure 8 has many servers (left side), access to standards (upper right), and end user applications (lower right); this can logically be mapped to another domain that has a similar architecture.

The two complementary aspects to allow all of the interactions to occur across the diagram is overlaid in Figure 8 via: health information exchange (pentagon) that uses dotted lines to indicate the need to share information among health information technology systems and applications; and, the Global Security Policy and Control (octagon) that provides a centralized location from which secure interactions can locally occur within the framework. The end result and major challenge, represented in Figure 8 for the health

Figure 8. The interplay of health information exchange and security



care domain, is the recognition of a greater need for a comprehensive approach to security at global and local levels operating within an environment that is driven to share data through health information exchange. This need for a global approach towards security, addressed at the document-level, is not unique to healthcare, as other domains (such as e-commerce, etc.) also make use of data found in distributed repositories, each with their own local and global security policies to enforce. Note that Figure 8 does not contain all of the possible scenarios and possibilities that can arise from the interplay of information exchange in healthcare or another domain with a similar architecture. First, other security threats (intrusion detection, name server attacks, etc.) can take place. These vulnerabilities and the effects they have in the information exchange process must be protected proactively. We have realized that, in current security approaches, these types of attacks are found retroactively (e.g., when audits of systems are performed). The impact of these attacks can create a disparity between system trust and sharing policies (local and global security). For example, as shown in the lower left of Figure 8, electronic health record systems that share information typically do so via the use of a Health Information Exchange (HIE) server that contains data from the production counterpart offloaded at regular intervals. In the event of a security compromise, only the information deemed sharable would be breached. This layer of defense allows the systems to only share the data they seem comfortable with in case of a security threat such as intrusion or server attacks.

Second, another major component in health information exchange is the data analytics and mining across the entire interoperating systems to allow clinicians and clinical researchers to query the data in support of analyzing health data towards improving medical care. Institutions and individuals who are placed under this component typically use data dispersed throughout repositories under a set of terms of conditions and agreement of fair usage. The case of such data analytics and mining presents a challenge in terms of security since the users of the data are not usually the owners, and therefore must abide by the security policies set in each individual system. In the case of health care, privacy of protected health information and personal identifiable information is controlled by an array of policy enforcement or computational methods, some which prove sufficient for a set of cases, but not all. For example, there exist generic data publishing methods (e.g., k-anonymity and (α, k)-anonymity) that are not useful for cases with demographic data and the specific needs of electronic health records. An example of this is that, if age is not to be disclosed, removing the age from the shared dataset might not be enough since health codes (e.g., International Classification of Diseases – 9[35] codes) can reveal a person's age. Towards the goal of providing proper security while maintaining data usefulness, several machine learning techniques have been developed not only as mechanisms of verification for privacy enforcement, but also as mechanisms for feedback on the quality and completeness of the desired security policies.

Third, a scenario, which is not typically discussed, is what happens in the event where information providers decide to enter and/or leave the information exchange architecture. In these situations, global and local security is impacted in not only the constituent systems of the information exchange process, but also in those meta-systems already deployed that make use of resources found on information providers that left; in this case, both outdated resources and information from new systems can skew data analysis. Such events require the constant need of updating the security policies on those systems that share and utilize share data. Currently, when such an event happens, each information provider still part of the architecture must scramble to update their sharing and security policies to the new architecture's components.

In Section 4, we reviewed security for XML as attained with Personal Health Assistant and SMARTSync. Our intent in this section is to provide a look at future trends in regards to the attainment of security for XML data exchange in health care from two perspectives. First, in Section 5.1, we explore the accessible technology platforms in the health care domain, more specifically we focus on Open *m*Health, Harvard SMART, and Microsoft HealthVault, that are all intended for widespread use by different users and in different contexts, to provide a means for various stakeholders (patients, clinicians, medical researchers, medical vendors, health information technology companies, etc.) to more easily interact with one another in different ways. These are concrete examples of the variety of platforms that can be found in an information exchange process, and while we focus on the healthcare counterpart as part of our case study, the content can be generalized to other domains with respect to the different platforms available to them. Next, in Section 5.2, we examine the open and free data repositories in the form of open electronic health records, namely, openEHR, PatientOS, and VistA. These repositories are intended to promote an environment of open access and sharing via a community approach, as viable alternatives to commercial products that are difficult to install and maintain, particularly for medical providers that lack information technology staff. In the same manner as Section 5.1, these electronic health records serve as concrete examples of the diversity and distributed nature of repositories in the information exchange process found across domains. Collectively, both perspectives demonstrate the significant level of complexity needed in regards to information exchange, and the diverse scenarios under which a security policy and control approach with both global and local components must effectively operate. Finally, Section 5.3 takes a concerted look at the information exchange and security issues from the health care perspective, their underlying computational issues, and makes recommendations that stakeholders in any domain would need to pursue in order to utilize information exchange process to insure that information can be successfully shared, exchanged, and secured. Again, many applications will have accessible platforms (open), proprietary systems (commercial), require the use of standards (in XML, RDF, OWL), and interact with Web-based and mobile apps via different protocols (JSON, SOAP, Web services, etc.); our work is generalizable.

5.1. Accessible Technology Platforms

In terms of accessible health information technology platforms, we focus on Microsoft HealthVault, Open *m*Health, and Havard SMART, all of which offer different capabilities to specific stakeholders for particular purposes. HealthVault, launched in 2007, is a personal health record intended to allow patients to manage their own medical information including demographic data, personal data (height, weight), medications, allergies, etc. The larger scale intent is to provide a means for this to be stored securely (protected health information and HIPAA compliant) while simultaneously facilitating interactions with health care providers, medical device companies, health information technology applications, etc. For example, there are a wide range of applications and devices[36] for managing medical conditions such as diabetes meters, blood pressure monitors, etc. that can connect to HealthVault. Also, major pharmacies (CVS and Walgreens) allow patients to be able to link their prescription records into HealthVault. Such a capability would greatly improve our Personal Health Assistant Patient and Provider applications, and the accuracy of medication reconciliation in SMARTSync. Other HealthVault partners include the American Cancer Society, American Diabetes Association, and the American Heart Association. The reach of HealthVault from a patient centered personal health record to one that can reach out into application, devices, and the medical provider community at large is an important future trend.

Harvard University's SMART (Sustainable Medical Applications, Reusable Technologies) is one of the projects funded by The Office of the National Coordinator for Health Information Technology through the Strategic Health IT Advanced Research Projects[37] (SHARP) program. The goal of the SMART project is to provide a uniform, well-defined, reusable infrastructure for applications to interact with medical-record data. The creators of SMART motivate their work with the argument that environments which are constantly changing and evolving such as the health care system have the inherit need for information technology infrastructures that are of general purpose nature rather than monolithic and pre-designed (Mandl, 2009). SMART has a diverse range of partners, including Microsoft, CVS/Caremark, Athena Health, the Massachusetts Department of Public Health, etc.; all share the objective of improving the delivery of health care to patients via technology. However, SMART's focus differs from HealthVault since it is emphasizing the promotion of the platform for research endeavors with the recent launch of a SMART app at Boston Children's hospital. The future trend to promote the exchange and sharing of information to positively impact patient care is a laudable objective.

Lastly, the Open *m*Health organization has defined an open source architecture with mix and match components for mobile health applications to be constructed from reusable units; the intent is to use the architecture to make data an information more meaningful to patients and clinicians via personal evidence that is provided by patients and analyzing by clinicians. To support this, Open *m*Health provides *Data Processing Units (DPU)* and *Data Visualization Units (DVU)* (Estrin, 2012). A data processing unit is a Web service that defines a set of data inputs and outputs (via an application programming interface) and provides an underlying algorithmic capability to extract, infer, and analyze a data set from one or more sources. For example, a data processing unit may take as input a set of glucose readings and insulin dosages with date time stamps, along with patient time (weight, age, etc.) and be able to output an analysis to determine the trends in terms of diabetes care (e.g., low glucose levels too often, etc.). A data visualization unit provides the means via a returned browser component to display information from a data processing unit in a form that is conducive to the recipient (patient, provider, researcher, etc.). Open *m*Health, via its data processing and visualization units, will provide access to a myriad of information including sensors from mobile devices, sensor data collected from the cloud (via glucose meters that update values to cloud repository), data from public sources (Food and Drug Administration DailyMed[38], nutrition), data from electronic health records, etc. One of their major initiatives is the Post Traumatic Stress Disorder collaboration with the Department of Veteran Affairs, having a mobile post-traumatic stress disorder Coach to help veterans cope with post-traumatic stress disorder. This future trend has the features of HealthVault (putting care under a patient's control) with a simultaneous eye to providers and researchers interested in treating patients or analyzing data sets.

These three platforms present the diversity of purpose and orientation (user-oriented for HealthVault, developer oriented for Open *m*Health, developer and user-oriented for SMART) in platforms that are constituents of the information exchange process, yet must act in harmony in order to provide the best and/or intended functionality and results for their end users. They also demonstrate the differences in the use of technologies and standards that can be found across platforms, from completely proprietary products (HealthVault) to community driven efforts (SMART) to semi-private/collaborative approaches (Open *m*Health). These differences result in different policies for usage and data sharing, and in turn result in conflicts of which policy should be enforced completely, partially, or not at all.

5.2. Open and Free Data Repositories

The adoption of electronic health records has increased in the previous years because of the benefits they provide and enacted laws, such as the Affordable Care Act of 2010[39]. The shift from paperwork to electronic data has pushed vendors to seize the opportunity and create their own respective versions of electronic health records for consumer adoption. We focus on those alternatives that are open source and free to implement: openEHR, PatientOS and VistA. openEHR[40] serves as a framework and standard to describe the administration and storage of patient data for electronic clinic history. This electronic clinic history acts as the repository of patient information, and is independent of the technology utilized for its access. The openEHR specifications are maintained by the openEHR foundation. These specifications, which arise from research done throughout a decade, include information and data models for the electronic clinic history, demographic information, clinical procedures, etc., and are implemented to provide a base for health information exchange. With the ongoing binding of Systematized Nomenclature of Medicine – Clinical Terms to openEHR, as well as the addition of a virtual electronic health record for the user interface, openEHR serves as an open source, complete solution framework for users and institutions to implement their own EHRs capable of complying with information exchange standards and data storage. This important future trend sets the standard for all electronic health records to target.

The objective of PatientOS[41] is to make the user's (medical provider) workflow a rapid one. To achieve this, PatientOS provides detailed observations on which values should be default in which forms, an almost never-changing user interface (as the tool is updated), automated functions and using the least amounts of clicks necessary to complete a task. PatientOS permits the user workflow to be customized based on the user, his/her role, and the institution in which the user works. While other software solutions are plagued by a scarce maintenance and update schedule, PatientOS has been designed to be scalable and easily maintained. These two factors reduce the bar on updating the electronic health record to a newer version by assuring users that the backend does not change. For developers, PatientOS offers a Java application programming interface that permits the development of plugins, customized forms, and user interface themes. By partnering with businesses such as MResult Healthcare, new alternatives to reducing the cost of implementation of electronic health records are possible. As a future trend, PatientOS targets the ease of both deploying and integrating health information technology into medical practices for usage by providers, a vital problem that hinders adoption of electronic health records.

Lastly, VistA[42] (Veterans Health Information Systems and Technology Architecture) is an open source information system built around an electronic health record. Developed by the United States Department of Veterans Affairs (VA), VistA consists of hundreds of clinical, financial, infrastructure and patient-Web functions. Clinical functions range from Admission Discharge Transfer, clinical procedures, pharmaceutical, laboratories, and mental health. Financial and administrative functions include an automated information collection system, incident reporting, fugitive felon program, and others. The infrastructure functions cover the maintenance of the backend, as well as communication standards, e.g., capacity management tools, Health Level 7 (messaging), broker, an SQL interface, and a network health exchange. The last sets of functions, the patient-Web functions, provide clinical information decision support, health record keeping, and a personal finance system. With the Veterans Health Administration utilizing VistA, the electronic health record is considered the largest medical system in the United States, spanning the largest health information exchange system and covering over 8 million patients. This widespread use also translates to physician usage: 60% of United States trained physicians rotate through the Veterans Health Administration, making VistA the most familiar and widely used electronic health record in the country. The widespread use of VistA has spanned different iterations, including Austronaut VistA[43], WorldVistA[44], OpenVistA[45], and vxVistA[46]. As a future trend, VistA is often cited as the "trend setting" and model for electronic health records and their adoption across the world.

VistA, openEHR and PatientOS demonstrate the way that data formats utilized for export and import functionalities, as well as internal operations, can differ between solutions intended for the same domain. These differences in data structures and formats utilized by repositories is not unique to the health care domain, and demonstrate a need to present an encompassing approach that handles all of the possibilities in term of document standards and secure information exchange. Added to this are the proprietary solutions that are widespread (for example, in the health care domain GE Centricity[47]) that could only support one of the many available standards due to development and economic reasons.

5.3. Recommendations for Success

The computing profession operates in the world where standards are the norm rather than the exception. After a tumultuous period in the late 1980s/early 1990s, where there were many different incompatible versions of C++, the profession has striven to an emphasis and strong reliance on the standards definition and approval process. In the early years of UML, every vendor's tool was incompatible; today, the UML via OMG has a standard of not just the language but the structure of the diagrams, allowing XMI to be exported for importing into any other UML tool. The same is true for database systems, which easily allow the porting of relational schemas and databases of all sizes via XML and XMI, allowing data to easily move from MySQL to Oracle or to SQL Server. In the process, the standards community in computing has provided the tools, and all of the vendors have accepted the responsibility to allow for designs, information, and data of all formats to be easily and effectively exchanged.

One troubling trend in widespread information exchange, especially in health care, is the lack of such a commitment by vendors and a promotion of data exchange. Clearly, as shown in Figure 8 for health care, there are many standards that have been adopted and are in use, ranging from JSON to RDF to XML to the continuity of care record and clinical document architecture standards; but those standards and the modeling of data have not been unified. Further, when one attempts to share information across commercial vendors (for example, between the electronic health records such as GE Centricity, AllScripts[48], etc.), it ends up requiring n^2 custom mappings of data between n different systems, as vendors are more concerned with proprietary protection of information as opposed to facilitating sharing; one major issue is competition as hospitals in a region see sharing information leading to the potential of losing patients. In health care this is clearly evident, as vendors do not want to provide the ability to export to a common XML format for medical data, since then it would allow the medical provider to potentially change vendors (akin to changing from Oracle to SQL Server in order to save licensing costs).

Another troubling trend is the inability to access information in repositories (for example, electronic health records) in a manner that would be able to present information that cuts across multiple instances of common information (patient's record); patient's visit multiple providers, labs, health facilities, etc., and if each has their own repository, the ability to get a complete collection of all of the information has not been achievable to date. Following the theme of health care, consider than an electronic health record is set up to manage individual patients and their medical records electronically including medications, allergies, immunizations, etc. However, suppose a medication is recalled, as Vioxx[49] was back in 2004, then every medical provider would need to contact his/her patients to switch their medication. But, electronic health records don't provide the ability to easily do this. In fact, for it to occur, one would need to have enough technical expertise to understand and access the underlying relational schema and database to write an ad-hoc query. There is a further hindrance in regards to supporting analysis of a medical provider's practice from a data and treatment perspective. For example, if you are a medical provider who wants to check on all of the diabetes patients in your practice taking a specific medication and determine commonalities related to other diseases (say congestive heart failure) or conditions (obesity), there is no way for you to make such an investigation without sophisticated ad-hoc queries.

Another issue to consider is the legal implications involved in data sharing and exchange, especially data that is confidential or otherwise protected by legal statues. For example, the Ethical, legal and social implications (ELSI) of human genomics are tied to the Genetic Information Nondiscrimination Act (GINA) of 2008[50] and the Health Insurance Portability and Accountability Act of 1996 (HIPAA). GINA protects a patient's genetic information against discrimination in health insurance and employment; this includes: genetic test of patient, his/her family members, fetus of individual or family member, family medical history, and request/receipt of genetic services that may including clinical research trials. HIPAA's Privacy Rule insures that protected health information is securely maintained by entries with patients retaining rights to access that information while still allowing entities to disclose the information under certain situations. HIPAA's Security Rule defines the "series of administrative, physical, and technical safeguards for covered entities to use to assure the confidentiality, integrity, and availability of electronic protected health information". For ELSI, protection of information must be reconciled across HIPAA and GINA to securely deliver the appropriate combination of clinical, genomic, and phenotypic information to all of the involved stakeholders (medical, researchers, clinical providers, support personnel, insurers, and patients).

Based on these issues and their underlying computational, political and acceptance limitations, we propose three major recommendations to facilitate information exchange in domains clouded by different standards, platforms, purposes and orientations. *First*, we note that there is a need for a *formal and unifying standards process* for data to allow the easy exporting and importing of data across information technology systems. This does not only mean to achieve a common format, but also one that uses the most up-to-date technologies. For example, in the health care scenario (see Figure 8), there are not just XML schemas used in current standards, but outdated and outmoded standards such as the document type definitions (XML DTD). One unifying standard or a set of standards with the most up-to-date features is a must to successfully achieve information exchange. We know this works well, since the UML community with its standard allows the easy exchange of UML designs between different tools and the database community provides export in XML to allow ease of porting a database across vendor products.

Second, open source and commercial data repositories need to be more conducive towards *cross platform access*, providing the potential for more effective tools and applications that improve the functionality, analysis or usage of information to improve end-user experience. In the case of health care, this involves open and commercial electronic health record vendors to provide *cross patient access* in order to provide better tools for medical providers so patient care and treatment improves. The computing community has been a leader in this regard, and these successful approaches need to be applied.

Third, involving the combination of legal statutes on domains that utilize and exchange data that is protected (for example, in health care with GINA and HIPAA for ELSI), will require careful consideration of the *policy, cost, usage, and exchange of information in a secure manner* across a wider range of data (for example, clinical, genomic, and phenotypic) than has normally been required. HIPAA gets involved in many non-medical settings, as does the need to manage personal identifiable information for many domains.

6. CONCLUSION

This chapter has presented a security framework for XML (see Section 1 and Figure 1) that is intended to allow security to be defined at the XML schema level to be enforced on XML instances using NIST Role-Based Access Control (RBAC) in Section 2. In the process, the XML instances delivered to users (by role) are customized to insure that a user is only able to see and/or modify what has been authorized, effectively yielding role restricted XML instances. To achieve the security, we leverage XACML to define policies against XML schemas that can then be enforced on all XML instances; our approach separates the security privileges from both the XML schema and their instances, allowing changes as security policies evolve to have no impact on schemas and existing instances. Our approach has been demonstrated on a healthcare domain case study with an emphasis on medical or clinical patient data as represented by the continuity of care record standard (see Section 2 and Figure 2), and we have briefly reviewed our earlier work (De la Rosa Algarín, 2012) that has created UML diagrams: an XML Schema Class Diagram (XSCD) to graphically represent an XML schema, and XML role slice diagram (XRSD) to define roles and their privileges (read, write, etc.) against XML schema elements (see Section 3.1). Using this as a basis, we described the generation of an XACML security policy for enforcement purposes, as detailed in Section 3.2, and illustrated in Figure 6. Collectively, the work in Section 3 was applied to in Section 4 to two applications: the Personal Health Assistant (PHA) in Section 4.2 hooked to Microsoft HealthVault for a patient to track of medications, nutritional supplements, allergies, etc., and also authorize that protected health information to his/her specific medical providers who can use their own app to select and view the authorized protected health information on a patient-by-patient basis as determined by his/he assigned role; and SMARTSync, an application for medication reconciliation in Section 4.3 hooked to Microsoft HealthVault and Harvard SMART's Reference Electronic Health Record to gather information from multiple sources. For each application, we demonstrated the generation of XACML policies in different ways from an eventual XML representation (Section 4.4). While we demonstrated the work in a health care domain, our architecture is simply servers, mobile, and Web apps, interacting with one another using many different mans (XML, JSON, SOAP, Web services, etc.). The XML security approach that we have presented is intended to target the wide array of apps on the World Wide Web.

Our future trends in Section 5 has taken a larger scale view of the security process for XML in regards to secure data sharing and exchange; while the work in this chapter (Section 3) is applicable to any domain, Section 5 focuses on the health care domain and the unique challenges of security at global and local levels that must interact with health information exchange across a myriad of health information technology systems and applications that have differing data formats that must be reconciled. The emerging trends in Sections 5.1 and 5.2 include accessible information technology platforms: the HealthVault system that allow patients to manage their health information with increasing linkages to applications, medical devices, pharmaceutical records, etc.; the Harvard SMART platform promoting a reusable infrastructure for health care data sharing and exchange, with diverse partners and roll out of research tools at a major medical center; the Open *m*Health platform trending to both satisfy patient, provider, and research requirements through an innovative and open means to collect and visualize information; and, a set of open and free data repositories, more specifically the electronic health records openEHR, PatientOS, and VistA, all striving to promote open access and sharing via a community approach, as viable alternatives to commercial products that are difficult to install and maintain. These platforms and systems served as concrete examples to pinpoint the major issues currently present to attain information exchange, and in Section 5.3 we presented a number of recommendations, some controversial, chastising the adoption of standards and free exchange of information by commercial information technology vendors, and hindering the attempt to utilize and exchange information in information technology systems for more effective usability, such as patient care in health care. Many of these recommendations are accepted practice in computing and information technology, and there needs to be a migration of these successes by applying those approaches to health care and other domains.

REFERENCES

Ammari, F., & Lu, J. (2010). Advanced XML security: Framework for building secure XML management system (SXMS). In *Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations (ITNG)* (pp. 120-125). ITNG.

Ardagna, C., Damiani, E., Capitani di Vimercati, S., & Samarati, P. (2007). XML security. In *Proceedings of Security, Privacy, and Trust in Modern Data Management* (pp. 71-86). Springer. doi:10.1007/978-3-540-69861-6_6.

Barnsteiner, J. (2005). Medication reconciliation: Transfer of medication information across settings-keeping it free from error. *The American Journal of Nursing*, *105*(3), 31. doi:10.1097/00000446-200503001-00007 PMID:15802996.

Bertino, E., Carminati, B., & Ferrari, E. (2004). Access control for XML documents and data. *Information Security Technical Report*, (9): 19–34. doi:10.1016/S1363-4127(04)00029-9.

Bertino, E., Castano, S., Ferrari, E., & Mesiti, M. (2002). Protection and administration of XML data sources. *Data & Knowledge Engineering*, (43): 237–260. doi:10.1016/S0169-023X(02)00127-1.

Bertino, E., & Ferrari, E. (2002). Secure and selective dissemination of XML documents. *ACM Transactions on Information and System Security*, (5): 290–331. doi:10.1145/545186.545190.

Chandrakumar, T., & Parthasarathy, S. (2012). Enhancing data security in ERP projects using XML. *International Journal of Enterprise Information Systems*, (8): 51–65. doi:10.4018/jeis.2012010104.

Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., & Samarati, P. (2000). Design and implementation of an access control processor for XML documents. *Computer Networks*, *33*(1), 59–75. doi:10.1016/S1389-1286(00)00053-0.

Damiani, E., Fansi, M., Gabillon, A., & Marrara, S. (2008). A general approach to securely querying XML. *Computer Standards & Interfaces*, *30*(6), 379–389. doi:10.1016/j.csi.2008.03.006.

De la Rosa Algarín, A., Demurjian, S. A., Berhe, S., & Pavlich-Mariscal, J. (2012). A security framework for XML schemas and documents for healthcare. In *Proceedings of 2012 International Workshop on Biomedical and Health Informatics (BHI 2012)*, (pp. 782-789). BHI.

Dolin, R. H., Alschuler, L., Boyer, S., Beebe, C., Behlen, F. M., Biron, P. V., & Shvo, A. S. (2006). HL7 clinical document architecture, release 2. *Journal of the American Medical Informatics Association*, *13*(1), 30–39. doi:10.1197/jamia.M1888 PMID:16221939.

Estrin, D., & Sim, I. (2010). Open mHealth architecture: An engine for health care innovation. *Science*, *330*(6005), 759–760. doi:10.1126/science.1196187 PMID:21051617.

Ferraiolo, D., Cugini, J., & Kuhn, D. R. (1995). Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference* (pp. 241-248). CSAC.

Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., & Chandramouli, R. (2001). Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, *4*(3), 224–274. doi:10.1145/501978.501980.

Huang, S., & Lesko, L. (2004). Drug-drug, drug-dietary supplement, and drug-citrus fruit and other food interactions: What have we learned? *Journal of Clinical Pharmacology*, *44*(6), 559. doi:10.1177/0091270004265367 PMID:15145962.

Kuper, G., Massacci, F., & Rassadko, N. (2005). Generalized XML security views. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies* (pp. 77-84). ACM.

Lanthaler, M., & Gütl, C. (2012). On using JSON-LD to create evolvable RESTful services. In *Proceedings of the 3rd International Workshop on RESTful Design (WS-REST 2012) at WWW2012* (pp. 25-32). Lyon, France: ACM Press.

Leonardi, E., Bhowmick, S., & Iwaihara, M. (2010). Efficient database-driven evaluation of security clearance for federated access control of dynamic XML documents. In *Database Systems for Advanced Applications* (pp. 299–306). Academic Press. doi:10.1007/978-3-642-12026-8_24.

Mandl, K., & Kohane, I. (2009). No small change for the health information economy. *The New England Journal of Medicine*, *360*(13), 1278–1281. doi:10.1056/NEJMp0900411 PMID:19321867.

Müldner, T., Leighton, G., & Miziołek, J. (2009). Parameterized role-based access control policies for XML documents. Information Security Journal: A Global Perspective, Taylor & Francis, (18), 282-296.

Pavlich-Mariscal, J., Demurjian, S., & Michel, L. (2008). A framework of composable access control definition, enforcement and assurance. In *Proceedings of the SCCC'08. International Conference of the IEEE* (pp. 13–22). IEEE.

Poon, E., Blumenfeld, B., & Hamann, C. et al. (2006). Design and implementation of an application and associated services to support interdisciplinary medication reconciliation efforts at an integrated healthcare delivery network. *Journal of the American Medical Informatics Association*, *13*(6), 581–592. doi:10.1197/jamia.M2142 PMID:17114640.

Rahaman, M., Roudier, Y., & Schaad, A. (2008). Distributed access control for XML document centric collaborations. In *Proceedings of the EDOC'08. 12th International IEEE* (pp. 267-276). IEEE.

Ziminski, T. B., De la Rosa Algarín, A., Saripalle, R., Demurjian, S. A., & Jackson, E. (2012). SMARTSync: Towards patient-driven medication reconciliation using the SMART framework. In *Proceedings of 2012 International Workshop on Biomedical and Health Informatics (BHI 2012)*, (pp. 806-813). BHI.

ADDITIONAL READING

Baumer, D., Earp, J., & Payton, F. (2000). Privacy of medical records: IT implications of HIPAA. *ACM SIGCAS Computers and Society*, *30*(4), 40–47. doi:10.1145/572260.572261.

Berhe, S., Demurjian, S., & Agresta, T. (2009). Emerging trends in health care delivery: Towards collaborative security for NIST RBAC. In *Proceedings of Research Directions in Data and Applications Security LNCS* (*Vol. 5645*, pp. 283–290). Berlin: Springer. doi:10.1007/978-3-642-03007-9_19.

Berhe, S., Demurjian, S., Gokhale, S., Pavlich-Mariscal, J., & Saripalle, R. (2011). Leveraging UML for security engineering and enforcement in a collaboration on duty and adaptive workflow model that extends NIST RBAC. [LNCS]. *Proceedings of Research Directions in Data and Applications Security*, *6818*, 293–300.

Berhe, S., Demurjian, S., Saripalle, R., Agresta, T., Liu, J., & Cusano, A. … Gedarovich, J. (2010). Secure, obligated and coordinated collaboration in health care for the patient-centered medical home. In *Proceedings of the AMIA Annual Symposium* (p. 36). AMIA.

Bernauer, M., Kappel, G., & Kramler, G. (2003). *Representing XML schema in UML - An UML profile for XML schema (Tech. Rep.)*. Citeseer.

Bernauer, M., Kappel, G., & Kramler, G. (2004). Representing XML schema in UML – A comparison of approaches. In *Proceedings of Web Engineering* (pp. 767–769). Web Engineering. doi:10.1007/978-3-540-27834-4_54.

Blechner, M., Sariapalle, R., & Demurjian, S. (2012). A proposed star schema and extraction process to enhance the collection of contextual and semantic information for clinical research data warehouses. In *Proceedings of 2012 International Workshop on Biomedical and Health Informatics (BHI 2012)*. BHI.

Demurjian, S., Ren, H., Berhe, S., Devineni, M., Vegad, S., & Polineni, K. (2010). Improving the information security of collaborative web portals via fine-grained role-based access control. In Murugesan, S. (Ed.), *Handbook of Research on Web 2.0, 3.0 and X.0: Technologies, Business and Social Applications* (pp. 430–448). Hershey, PA: IGI Global.

Demurjian, S., Saripalle, R., & Berhe, S. (2009). An integrated ontology framework for health information exchange. In *Proceedings of 21st International Conference on Software Engineering and Knowledge Engineering (SEKE09)* (pp. 575–580). SEKE.

Doan, T., Demurjian, S., Ting, T. C., & Phillips, C. (2004). RBAC/MAC security for UML. *Research Directions in Data and Applications Security*, *144*, 189–204. doi:10.1007/1-4020-8128-6_13.

Klyne, G., Carroll, J. J., & McBride, B. (2004). *Resource description framework (RDF), concepts and abstract syntax*. W3C Recommendation, 10.

Mandl, K., Mandel, J., & Murphy, S. et al. (2012). The SMART platform: Early experience enabling substitutable applications for electronic health records. *Journal of the American Medical Informatics Association*, *19*(4), 597. doi:10.1136/amiajnl-2011-000622 PMID:22427539.

Montelius, E., Astrand, B., Hovstadius, B., & Petersson, G. (2008). Individuals appreciate having their medication record on the web: A survey of attitudes to a national pharmacy register. *Journal of Medical Internet Research*, *10*(4). doi:10.2196/jmir.1022 PMID:19000978.

Pavlich-Mariscal, J., Demurjian, S., & Michel, L. (2010). A framework for security assurance of access control enforcement code. *Computer & Security Journal*, *29*(7), 770–784. doi:10.1016/j.cose.2010.03.004.

Pavlich-Mariscal, J., Demurjian, S., & Michel, L. (n.d.). A framework of composable security features: Preserving separation of security concerns from models to code. *Computer & Security Journal, 29*(3), 350-379.

Pavlich-Mariscal, J., Doan, T., Michel, L., Demurjian, S., & Ting, T. C. (2005). *Role slices: A notation for RBAC permission assignment and enforcement. Research Directions in Data and Applications Security (LNCS)* (*Vol. 3654*, pp. 40–53). Berlin: Springer.

Phillips, C., Demurjian, S., & Bessette, K. (2005). A service-based approach for RBAC and MAC security. In Stojanovic, Z., & Dahanayake, A. (Eds.), *Service-Oriented Software System Engineering: Challenges and Practices* (pp. 317–339). Hershey, PA: IGI Global.

Routledge, N., Bird, L., & Goodchild, A. (2002). UML and XML schema. *Australian Computer Science Communications*, *24*(2), 157–166.

Saripalle, R. Knath, Demrjian, S., & Berhe, S. (2011). Towards a software design process for ontologies. In *Proceedings of 2011 International Conference on Software and Intelligent Information (ICSII 2011)*. ICSII.

Skogan, D. (1999). UML as a schema language for XML based data interchange. In *Proceedings of the 2nd International Conference on the Unified Modeling Language (UML'99)*. UML.

KEY TERMS AND DEFINITIONS

Continuity of Care Record (CCR): A document standard for health information typically used for Personal Health Records (PHR) with the intended purpose of information exchange. It provides a universal structure to the patient's information that can be utilized by different personal health records, applications and systems.

Electronic Health Record (EHR): An electronic version of the patient's medical record. An electronic health record contains all related health information, from medications to procedures, and is managed by the institution in which it is stored (e.g. hospital, private practice, clinic, etc).

eXtensible Access Control Markup Language (XACML): A security policy language designed from XML. Its specifications allow for a uniform policy language that can be enforced in heterogeneous systems. XACML policies can be enforced at a systems level, software level, or information level, depending on the policies' targets and rules.

eXtensible Markup Language (XML): A structured language utilized for information exchange, standards and information validation via the use of schemas. Its extensibility allows developers and experts to design and implement common standards for the use across systems and domains.

Health Information Exchange (HIE): The ability to share information among health information technology systems by linking information for the same patient across multiple repositories to provide a complete health care view.

Health Information Technology (HIT): Information technologies (e.g. mobile applications, computer programs, decision support systems, etc.) whose use is intended for the healthcare domain.

Meta-System: A system or platform built from many constituent systems that makes use of functionality or data distributed among its components or external data repositories.

Personal Health Record (PHR): An electronic version of a medical record that is managed by the patient. PHRs typically provide the means to manage medication lists, allergies, procedures, emergency contacts, and other clinical data.

Personal Identifiable Information (PII): Information that contains attributes and values that can help determine a person's identity.

Protected Health Information (PHI): Clinical and other health related information regarding a patient that is protected under laws, or must be protected as dictated by laws.

Role-Based Access Control (RBAC): An access control model where permissions are assigned directly to roles, which are assigned to users.

Role-Restricted (RR): A filtered version of a document. Its filtering depends on the role of the user requesting the information, and the security policies in place.

Substitutable Medical Apps, Reusable Technologies (SMART): A platform that permits the development and reusability of applications by targeting a common abstraction layer, removing the need to target specific data repositories.

XML Role-Slice Diagram (XRSD): The XRSD is a diagram containing the role's credentials and the elements of the XML schema on which these credentials act.

XML Schema Class Diagram (XSCD): The XSCD is an UML artifact that serves as an equivalent representation of an XML schema in a UML diagram.

ENDNOTES

1 eXtensible Markup Language, http://www.w3.org.com/XML/

2 Continuity of Care Record (CCR), http://www.astm.org/Standards/E2369.htm

3 Microsoft HealthVault, http://www.microsoft.com/en-us/healthvault/

4 Health Insurance Portability and Accountability Act of 1996 (HIPAA), http://www.hhs.gov/ocr/privacy/

5 National Institute of Standards and Technology, http://www.nist.gov/index.html

6 OASIS XACML, https://www.oasis-open.org/committees/xacml/

7 Unified Modeling Language, http://www.uml.org/

8 SMART Platforms, http://www.smartplatforms.org/

9 Android, http://www.android.com/

10 JavaScript Object Notation, http://www.json.org/

11 Open *m*Health, http://openmhealth.org/

12 XML Security Working Group (SWG), http://www.w3.org/2008/xmlsec/

13 MySQL, http://www.mysql.com/

14 SMART RDF/XML Data Model, http://dev.smartplatforms.org/reference/data_model/

15 NDF-RT, http://www.pbm.va.gov/NationalFormulary.aspx

16 RDF/XML, http://www.w3.org/TR/REC-rdf-syntax/

17 N-TRIPLES, http://www.w3.org/2001/sw/RDFCore/ntriples/

18 Terse RDF Triple Language (TURTLE), http://www.w3.org/TeamSubmission/turtle/

19 JSON for Linking Data (JSON-LD), http://json-ld.org/

20 Resource Description Framework (RDF), http://www.w3.org/RDF/

21 RESTful Services, http://www.ibm.com/developerworks/Webservices/library/ws-restful/

22 MS VisualStudio XML Schema Definition Tool, http://msdn.microsoft.com/en-us/library/x6c1kb0s(v=vs.80).aspx

23 StylusStudio, http://www.stylusstudio.com/

24 Eclipse's Oxygen XML Plugin, http://oxygenxml.com/eclipse_plugin.html

25 Trang, http://www.thaiopensource.com/relaxng/trang.html

26 openEHR, http://www.openehr.org/home.html

27 PatientOS, http://www.patientos.org/

28 VistA, http://worldvista.org/AboutVistA

29 RxTerms, http://wwwcf.nlm.nih.gov/umlslicense/rxtermApp/rxTerm.cfm

30 RxNorm, http://www.nlm.nih.gov/research/umls/rxnorm/

31 SNOMED, http://www.ihtsdo.org/snomed-ct/

32 UMLS, http://www.nlm.nih.gov/research/umls/
33 MeSH, http://www.ncbi.nlm.nih.gov/mesh
34 LOINC, http://loinc.org/
35 ICD-9 codes, http://www.icd9data.com/
36 HealthVault App Directory, https://account.healthvault.com/Directory
37 Strategic Health IT Advanced Research Projects, http://goo.gl/62K5d
38 FDA DailyMed, http://dailymed.nlm.nih.gov/dailymed/about.cfm
39 Affordable Care Act of 2010, http://www.healthcare.gov/law/full/
40 openEHR, http://www.openehr.org/home.html
41 PatientOS, http://www.patientos.org/index.html
42 VistA, http://worldvista.org/AboutVistA
43 Austronaut VistA, http://astronautvista.com/
44 WorldVistA, http://www.worldvista.org/
45 OpenVistA, http://www.medsphere.com/solutions/openvista-for-the-enterprise
46 vxVista, https://www.vxvista.org/display/vx4h/Welcome
47 GE Centricity, http://goo.gl/LCvU7
48 AllScripts, http://www.allscripts.com/
49 Vioxx, http://www.merck.com/newsroom/vioxx/
50 Genetic Information Nondiscrimination Act of 2008, http://www.genome.gov/24519851/