A Policy-Based Dialogue System for Physical Access Control

Mohammad Ababneh
Department of Computer Science
George Mason University
Fairfax, VA, USA
mababneh@gmu.edu

Duminda Wijesekera
Department of Computer Science
George Mason University
Fairfax, VA, USA
dwijesek@gmu.edu

James Bret Michael
Department of Computer Science
Naval Postgraduate School
Arlington, VA, USA
bmichael@nps.edu

***Abstract*—We prototype a policy-based dialog system for providing physical access control to secured facilities and smart buildings. In our prototype system, physical access control policies are specified using the eXtensible Access Control Markup Language. Based on the policy and the user's presence information, our dialog system automatically produces a series of questions and answers that, if correctly answered, permit the requester to enter the secure facility or smart building. The novelty of this work is the system's ability to generate questions appropriate to the physical location, time of day, and the requester's attributes.**

***Index Terms*—Dialogue, Question Answering, Voice Recognition, VXML, XACML, Access Control Policy, Security**

I. OVERVIEW

We developed a prototype policy-based system for guarding entry to physical facilities, such as smart buildings. The system interacts with potential entrants using a spoken dialogue. Our physical access control system uses the OASIS consortium's eXtensible Access Control Markup Language (XACML) standard [1] and the W3C's Voice eXtensible Markup Language (VXML) [2] for specifying the dialogue.

In order to generate dialogues from physical access control policies specified in XACML, we generate so-called "VXML Voice forms" from XACML policy rules. In this paper we describe our initial implementation of the prototype. Given that emerging mobile applications use interactive voice commands such as Apple's Siri, Google's Andriod S-Voice, and Microsoft Windows Speech Recognition, we envision that new applications would emerge for interactive voice-based access to resources.

The dialogues we generate are in the form of a question and an (acceptable) answer. In our prototype, questions are generated using a grammar for words or phrases—belonging to a restricted vocabulary—that are taken from an XACML rule's subject-attribute values [3][4]. Answers should conform to a grammar that is linked to the rule's data type, and acceptable answers are those that provide the values that match the values specified in the XACML policy. The class of grammars may come from and be checked against a database or other sources of subject attributes.

For each question, user input is collected and stored in a variable. These variables are used to generate an XACML request that is passed to an XACML policy decision point (PDP). The PDP is responsible for the decision of either granting or denying access. If access is granted, the system sends a control-system message to an actuator that unlocks the entry door to the facility.

We succeeded in generating a question for each rule in the policy. Our existing implementation uses a small number of policy rules and their conversion. We are currently working on scaling this implementation by addressing the run-time and automatic conversion of a large number of rules, in addition to developing the capability to dynamically generate the grammars using .grxml files [3].

Due to advancements in mobile applications and the emergence of voice user interfaces (VUI) as well as their being provided as a service in the cloud, new access control mechanisms are needed. When completed, our current architecture and implementation will serve as a testbed for further research and development.

A. Potential Applications

The following are potential applications for our system:

- *Mobile computing:* The adaptation of the voice technology and VUI in mobile computing (e.g., Apple's Siri, Google's Android S-Voice, Microsoft Windows Speech Recognition,

Research in Motion's BlackBerry) introduces challenges in using the technology in order to accomplish more sophisticated tasks such as access control to either resources and services locally on the devices or remotely at data centers via cloud services. We envision this system to be useful for providing ubiquitous policy-based automated physical access control from mobile devices, where hands-free usage is valued or required.

- *Military and first-responder applications:* It is important to provide hands-free user interfaces to military personnel, especially those participating in combat operations, as well as first responders (e.g., firefighters, police, paramedics). These personnel have to access support facilities (e.g., gates at a forward operating bases or police sub-stations) and equipment (e.g., fire trucks) (see [5]).
- *Information systems:* In general this access control can be used to provide appropriate access to any information system using a VUI.
- *Electronic Commerce and Business:* This new access control approach can be used in performing transactions in e-commerce applications accessed via mobile computing devices.
- *Physical Access Control:* Access control to critical facilities can be automated and access can be granted or denied based on an enterprise's policy. Policy languages such as XACML are standardized to unify policies across enterprises and reduce administrative load. This is all hands-free and it only depends on answers to questions that represent the subject's attributes. A sample dialogue for access control is illustrated in Table 1.

1	U	Hello
	S	Welcome, Please Say who you are?
2	U	I am User01 (Alice)
	S	Please say your Password or enter it using the key pad
3	U	Pass01
	S	Ok, I got that. Now tell me Why do you need to access the building (Role)
4	U	Professor
	S	What is your office number
5	U	4429
	S	In which floor is the dean's meeting room
6	U	5th
	S	What is the time on the clock to your right
7	U	7:30
	S	Are you sure
8	U	Yes
	S	Is it raining outside?
9	U	Yes
	S	Is it cold or warm?
10	U	Cold
	S	Is the temperature below 40 degrees?
11	U	No
	S	Can you estimate the range that the temperature might be in?
12	U	50 to 60
	S	Ok, granted access

U: User, S: System

TABLE 1. A SAMPLE DIALOGUE FOR ACCESS CONTROL

The original intention of access control policy languages such as XACML was to deal with systems and access control enforcement points, but not humans. In our approach we use XACML to drive human interaction through dialogues with the system. This new approach has implications on the way dialogues are generated and access control decisions are taken. As mentioned above, many potential applications might take advantage of this approach. To be applicable, many aspects of human voice interactions should be studied—more research on this but it is outside of the scope of the work we report on here.

One important aspect of such an approach is the scale of implementation, especially in case of physical control. The user of such a system will likely not use the system if the system takes a long time to make an access decision for each individual of a large number of humans waiting in crowds such as at a sporting event (e.g., a World Cup football match). The processing time of an access request initiated by a human entity is going to be different than the time initiated by an automated process or application. This is another area of research we have left to future work.

II. BACKGROUND

In this section we discuss the most relevant standards and technologies to our research.

A. VoiceXML

VoiceXML (VXML) is the Voice Markup Language developed and standardized by the W3C's Voice Browser Working Group. It is intended for creating audio dialogues that feature synthesized speech, digitized audio, recognition of spoken and Dual Tone Multi-Frequency (DTMF) key inputs, recording of spoken input, telephony, and mixed initiative conversations. VXML is similar to HTML in the textual arena in providing an interface between a user and the Web, using a voice interface. Its purpose is to bring the advantages of web-based development and content delivery to interactive voice response (IVR) applications. All Web technologies are

still relevant in any voice interface, such as services, markup languages, linking, URIs, caching, standards, accessibility, and cross-browser [2].

The most important terms in VXML are:

- *Form:* Forms define an interaction that collects values for a set of form-item variables. Each field may specify a grammar that defines the allowable inputs for that field. If a form-level grammar is present, it can be used to fill several fields from one utterance
- *Block:* An item is a component of a form that presents information by synthesizing a phrase of text into speech to the user.
- *Field:* An item is a component of a form that gathers input from the user by synthesizing a textual phrase into speech for the user. The user must provide a value for the field before proceeding to the next element in the form.
- *Menu:* A menu presents the user with a choice of options and then transitions to another dialog based on that choice

B. XACML

XACML is an OASIS standard XML-based language for specifying access control policies [1]. In a typical XACML usage scenario, a subject that seeks access to a resource submits a query through an entity called a Policy Enforcement Point (PEP). The PEP, responsible for controlling access to the resource, forms a request in the XACML request language and sends it to the PDP. The PDP in turn evaluates the request and sends back one of the following responses: accept, reject, error, or unable to evaluate, with the PEP allowing or denying access to the requester accordingly, as shown in Figure 1.

Figure 1 contains the following entities:

- *Policy Set:* A set of policies.
- *Policy:* A set of rules, an identifier for the rule-combining algorithm, and (optionally) a set of obligations. May be a component of a policy set.
- *Rule:* A target, an effect, and a condition. A component of a policy.
- *Combination Algorithm:* The procedure for combining the decision and obligations from multiple policies.
- *Subject:* An actor whose attributes may be referenced by a predicate.
- *Target:* The set of decision requests, identified by definitions for resource, subject, and action that a rule, policy, or policy set is intended to evaluate.
- *Resource:* Data, service or system component.
- *Attribute:* All entities are identified using attributes.

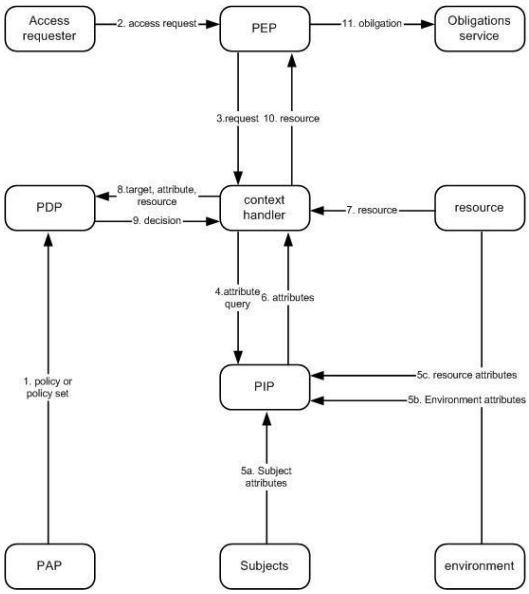

Fig. 1. XACML's Data-Flow Diagram

- *Predicate:* An evaluable statement about attributes.
- *PEP:* Governing entity of a resource.
- *PDP:* The entity that evaluates access requests.
- *PIP:* The entity that fetches attribute values for the PDP..
- *PAP:* The entity that retains polices.
- *Context Handler:* The entity that converts decision requests to XACML requests.

Some of the currently available implementations of the XACML specification are for example OpenXACML, enterprise-java-xacml from Google code, HERAS XACML, JBoss XACML, Sun Microsystem's XACML, and WSO2 Identity Server, which is used in this implementation.

III. IMPLEMENTATION PLATFORMS

In this section we introduce briefly the implementation platforms of the standards and technologies used to implement our prototype.

A. Voxeo for Speech Recognition

Voxeo Prophecy is a standards-based platform for speech, IVR, and Software Implemented Phone (SIP) applications for Voice over Internet Protocol (VoIP) applications [6]. Some of the capabilities integrated into the platform are: automatic speech recognition, speech synthesis, and visual programming. Prophecy provides libraries to create and deploy IVR or VoIP applications, including VXML and Call Control (CCXML) browsers with speech recognition and synthesis engines, and a

built-in SIP soft-phone. Prophecy supports ASP, CGI, C#, Java, PERL, PHP, Python, and Ruby and has a built-in web server that supports PHP and Java applications. It complies with VXML and CCXML standards.

B. XACML Implementation – WSO2

WSO2 Identity Server [7] provides security and identity management of enterprise web applications, services, and APIs. WSO2 full implementation supports identity management, single sign-on, Role-based Access Control (RBAC), fine-grained access control, LDAP, OpenID, SAML, Kerberos, OAuth, WS-Trust, and the XACML 2.0/3.0.

C. Programming and development

In addition to the above two major platforms, programming languages such as Java, Java Server Pages (JSP), and Java Script (JS) were used to accomplish the integration and interoperability work between these platforms and thus enabled us to develop our prototype.

IV. OUR APPROACH

The core of our work transforms an access control policy into a voice platform supported voice language. We use XACML and W3C VXML for these two purposes.

In order to generate a dialogue between a user and an access control system to make it possible for the system to make a decision to whether grant or deny access, the access control policy is transformed into VXML. The rules in each policy are read and transformed into VoiceXML blocks and forms. The entire policy is parsed using a DOM parser and then each rule element is converted to a VXML block for the user interface translating text to speech (TTS), posing the question to the user, and then waiting for the user's response through voice recognition. The details of how TTS and voice recognition technologies are outside the scope of our current work; we are implementing these services through an integrated Voice Application Development Environment introduced in Section III.A. Figure 2 depicts the high-level architecture for our prototype system.

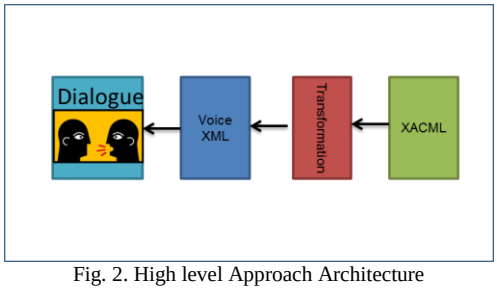

Fig. 2. High level Approach Architecture

A more detailed illustration of this policy voice (VXML) is shown in Figure 3.

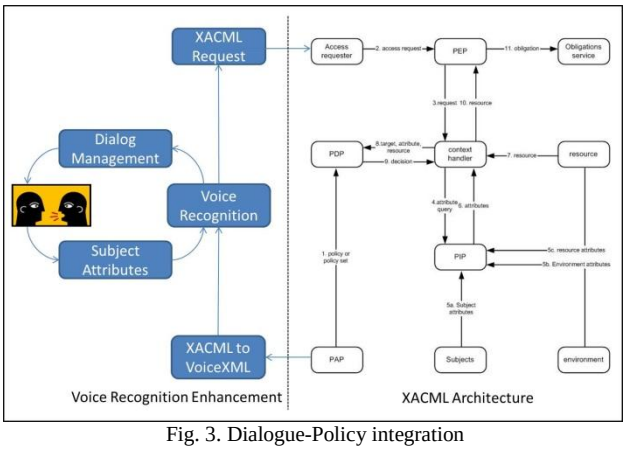

Fig. 3. Dialogue-Policy integration

A. A Working Scenario

We used the Voxeo Prophecy IVR platform (see www.voxeo.com/products/voicexml-ivr-platform.jsp)—including the webserver, designer, SIP, and application manager—to develop our application for voice recognition. We use a scenario to illustrate how the application works.

Our scenario starts with a XACML policy file, with the rule shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy RuleCombiningAlgId="identifier:rule-
combining-algorithm:deny-overrides"
PolicyId="urn:oasis:names:tc:example:SimplePolic
y1"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0
:policy:schema:cd:04 http://docs.oasis-
open.org/xacml/access_control-xacml-2.0-policy-
schema-cd:04.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schem
a:cd:04">
<Description> Med Example Corp access control
policy
</Description>
<Target/>
<Rule Effect="Permit"
RuleId="urn:oasis:names:tc:xacml:2.0:example:Sim
pleRule1">
<Description> Any subject with an e-mail name in
the med.example.com domain can perform any
action on any resource.
</Description>
<Target>
<Subjects>
<Subject>
<SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:r
fc822Name-match">
<AttributeValue
DataType="urn:oasis:names:tc:xacml:1.0:data-
```

```
type:rfc822Name"> @med.example.com
</AttributeValue>
<SubjectAttributeDesignator
DataType="urn:oasis:names:tc:xacml:1.0:data-
type:rfc822Name"
AttributeId="urn:oasis:names:tc:xacml:1.0:subjec
t:subject-id"/>
</SubjectMatch>
</Subject>
</Subjects>
</Target>
</Rule>
</Policy>
```

Fig. 4. A sample XACML rule

Using JSP, we load the XACML file into a Document Object Manager (DOM) object. We read the rules inside the XACML document and link each rule to a VXML block/form. The JSP script extracts the attribute's value from the DOM's rule element and passes it to the Voxeo designer application, which converts it to VXML. Figure 5 shows an example of a VXML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/
  XMLSchema-instance"
 xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0">
  <form>
  <field name="e-mail">
     <prompt>What is your e-mail address?
     </prompt>
     <grammar src="email.grxml"
            type="application/srgs+xml"/>
  </field>
  <block>
<submit next="http://www.example.com/user.asp"/>
  </block>
 </form>
</vxml>
```

Fig. 5. A sample VXML

In order to create the question, a phrase is inserted before the attribute value in the form of "What is?" or "Is your?" followed by the attribute name extracted from the RuleID of the DOM's rule element. Our current implementation supports the yes/no answers to "Is your?" type of questions. We are in the process of enlarging the question formation syntax to support other question formats.

In this way a question will be generated for every rule in the policy file. The human user then needs to answer the questions posed by the system.

The next step is to collect attribute "VoiceXML variable" values generated throughout the dialogue and use them to generate an XACML request. Figure 6 shows an example of a request.

```
<Request>
<Subject>
<Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subjec
t:subject-id"
DataType="urn:oasis:names:tc:xacml:1.0:data-
type:rfc822Name">
<AttributeValue>mababneh@@med.example.com
</AttributeValue>
</Attribute>
<Attribute AttributeId="group"
DataType=http://www.w3.org/2001/XMLSchema#string
Issuer="admin@gmu.edu">
<AttributeValue>Developers</AttributeValue>
</Attribute>
</Subject>
<Resource>
<Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resour
ce:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#anyUR
I">
<AttributeValue>http://server.example.com/code/d
ocs/developer-guide.html</AttributeValue>
</Attribute>
</Resource>
<Action>
<Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action
:action-id"
DataType="http://www.w3.org/2001/XMLSchema#strin
g">
<AttributeValue>read</AttributeValue>
</Attribute>
</Action>
</Request>
```

Fig. 6. A sample XACML request

The XACML request will be sent to the XACML implantation of choice. In our case, we have chosen WSO2 Identity Server version 3.2. It has a distinguished, modern, and high-performance XACML implementation with a service-oriented implementation option. The Voxeo development environment supports both HTTP requests and web services. Our choice was to enforce the policy by accepting the XACML request through a web service interface with the WSO2 XACML PEP. The PDP will take a decision based on the attribute values collected from the dialogue and matches of values of subjects and resources in the XACML implementation (see Figure 1). The grant or deny decision will then be ready to be returned back to the application. In our current case, it should be translated to a physical access decision as to whether to open a door.

The XACML's access decision is based on the output of the policy and rule combining algorithms. Following the standard, the rule and policy should evaluate to true in order to grant access; otherwise it would produce "indeterminate" or "not applicable." In case there are multiple applicable rules and policies, the final access decision is the result of the logical combination of these algorithms.

Our work, in its final state, will illustrate the use of XACML to control access to resources through building dialogues with human users. There are efforts proposing access control systems through XACML interfacing with other data models. In the published literature, a majority of this integration effort is with Web Services [8]. Most of this harmonizing work relied on the use of the de facto Simple Object Access Protocol (SOAP) [9] messages in the Web Services architecture to extract security-related attributes and use them in XACML for the purpose of access control [10] [11].

Security Assertion Markup Language (SAML) profile for XACML is heavily relied on when there is a need to use additional subject's attributes that are administered by other authorities to evaluate access control requests [12]. SAML and other message exchange protocols can be the means through which XACML can interface with other data models. Our work is different by trying to let XACML reach the human user directly and initiating a dialogue with him, manage the dialogue, and then decide whether to allow access.

V. NEXT STEPS

Our next steps in this work would be:

- Finishing the XACML implementation
- Being able to generate requests and responses and execute them
- Determining the best way to use grammars
- Looking into the best way to generating VoiceXML from XACML: there are options to evaluate such as DOM and XSLT

Some of the follow-on research items we are pursuing are:

- Integrating presence information with the dialog access control system. It is critical for an automated system to authenticate to the system the speaker or requester of access to avoid certain attacks, such as by verifying the physical presence and human nature of the speaker.
- Making the dialog as short as possible. In a spoken dialog or question-answer system, it is different than filling a form using a keyboard and a mouse. It can take more time to say the voice block (form) than filling it by hand or via a keyboard. In some cases, we might need to make a quick access decision through dialog which might require thinking of ways to reduce the time required to collect attributes and make a correct XACML decision. Maybe asking only the most difficult or important questions according to their weight can reduce the number of questions without affecting the accuracy. An analysis similar to the one in [13] implementing item response theory [14] [15] might be helpful to this work.
- Some policy sets might have a large number of policies and rules with different combination algorithms. It would be interesting to see how this can affect our spoken policy-based access control (question-answer) system.
- After being able to generate dialogues from policies, it would be interesting to see if we can generate rules from dialogues.
- In any dialogue, it is important to guarantee the privacy of what is spoken. In order to answer a question the user has to say things that might be considered to be sensitive (e.g., a unique government identity card number, such as a social security number) and the user might be reluctant to answer the question in public, which might affect the attributes collected in order to build the request and thus the decision might not be correct.

VI. CONCLUSION

In this work, we have presented a novel approach to generate a dialogue for the purpose of physical access control from a standard access control policy language. This policy language driven interaction with the user or authorization requester is generated at runtime and is implemented in a standard language.

ACKNOWLEDGMENT

This material is based upon work supported by the US Air Force Office of Scientific Research under grant FA9550-09-1-0421.

REFERENCES

[1] OASIS, eXtensible Access Control Markup Language (XACML), available at URL: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, last accessed Aug. 6, 2012.

[2] World Wide Web Consortium, Voice Extensible Markup Language (VoiceXML)(VXML), available at URL: http://www.w3.org/Voice/, last accessed Aug. 6, 2012

[3] World Wide Web Consortium, Speech Grammar Recognition Specification, white paper available at URL: http://www.w3.org/TR/speech-grammar/, last accessed Aug. 6, 2012.

[4] J. E. Hopcroft and J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[5] Thomas Massie and Duminda Wijesekera, "TVIS: Tactical Voice Interaction Services for Dismounted Urban Operations," in Proceeding of Military Communications Conference, IEEE, pp. 258-264, San Diego, Calif., Nov. 17-19, 2008.
[6] URL: http://www.Voxeo.com, last accessed Aug. 6, 2012.
[7] URL: http://www.wso2.com, last accessed Aug. 6, 2012.
[8] OASIS, Web Services, available at URL: https://www.oasis-open.org/standards
[9] W3C, Simple Object Access Protocol (SOAP), available at URL: http://www.w3.org/TR/soap
[10] Chongshan Ran and Guili Guo, "Security XACML Access Control Model Based On SOAP Encapsulate," International Conference on Computer Science and Service System, IEEE, pp. 2543-2546, Nanjing, China, 27-29 June 2011.
[11]Ardagna, C.A., De Capitani di Vimercati, S., Paraboschi, S., Pedrini, E., Samarati, P., Verdicchio, M. , "Expressive and Deployable Access Control in Open Web Service Applications," IEEE Transactions on Services Computing, Volume 4, Issue 2, pp. 96-109, April-June 2011.
[12] OASIS, Security Assertion Markup Language (SAML), SAML 2.0 profile of XACML v2.0, available at URL: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf
[13] Ahmed A. L. Faresi and Duminda Wijesekera, "Preemptive Mechanism to Prevent Health Data Privacy Leakage," in Proceedings of International Conference on Management of Emergent Digital EcoSystems, ACM, pp. 17-24, San Francisco, Calif., Nov. 21-23, 2011,
[14] F. B. Baker, The basics of item response theory, ERIC Clearinghouse on Assessment and Evaluation, 2001.
[15] F. B. Baker, Item response theory: Parameter estimation techniques, vol. 176, CRC, 2004.

AUTHOR BIOGRAPHIES

MOHAMMAD ABABNEH is a doctoral student in the Information Technology program at George Mason University.

DUMINDA WIJESEKERA is an Associate Professor of Computer Science at George Mason University, where he serves as Program Director of the Doctor of Philosophy program in Information Security and Assurance. He is a member of the university's Center for Security Information Systems.

JAMES BRET MICHAEL is a Professor of Computer Science and Electrical and Computer Engineering at the Naval Postgraduate School, Arlington, Virginia.