PREVENTING MALICIOUS PORTLETS FROM COMMUNICATING AND INTERCEPTING IN COLLABORATION PORTALS

Oliver Gmelch and Günther Pernul
Department of Information Systems, University of Regensburg, Regensburg, Germany

Keywords: Inter-portlet communication, Portal servers, Security policies.

Abstract: In a "networked enterprise", distributed teams of partner organizations, humans, computer applications, autonomous robots, and devices are interlinked to collaborate with each other in order to achieve higher productivity and to perform joint projects or produce joint products that would have been impossible to develop without the contributions of multiple collaborators. Within a collaboration, security aspects are of critical importance. This is in particular true for loosely coupled collaborations in which individual members of one alliance are working with each other within a certain project only, but may be competitors in other market fields at the same time. Going beyond the current state of the art in portal-based collaboration platforms, this paper presents an approach to prevent unintended information disclosure by malicious portlet instances. The solution is built on open standards (JSR 286 and XACML) and may be incorporated in collaboration-wide enterprise portals in order to regulate information flow during inter-portlet communication.

1 INTRODUCTION

Organizations today are confronted with big challenges such as globalization, shorter innovation cycles or increased competition. One possible remedy against this situation is forming virtual partnerships in order to create flexible and agile business networks between a number of partner organizations having complementary competencies. As diagnosed by recent surveys performed for instance by AT&T (AT&T Corp., 2008), a significant increase in the number of business alliances can be expected in the near future.

One of the key challenges in virtual enterprises is the backing by dedicated information and communication technology (ICT). Identified as a crucial characeristic of virtual partnerships, ICT systems are expected to provide support for different success factors such as integration on process level across company boundaries or information integration, implying provisioning the right information to the responsible user at the proper point in time (Katzy, 1998).

One way of achieving this information integration is via the introduction of specifically-tailored Enterprise Portals, allowing for integration of different applications (even from different alliance partners) in one common user interface comprising a number of specifically-crafted portlets (Shilakes and Tylman, 1998). Using communication facilities between these portlets, the enterprise portal is empowered to apply context changes in one application to other applications involved in a specific setting. While the applications run independently in the different sites, inter-portlet communication is necessary at the portal level. However, security aspects imposed by the communication of independent applications at varying security level must not be neglected. For instance, unrestricted communication may result in unintended information disclosure, thus posing great risks on confidential information processed in such settings and resulting in great loss of acceptance of involved applications or even a loss of trust in an alliance member or alliances as a whole.

The research project "SPIKE" (Secure Process-oriented Integrative Service Infrastructure for Networked Enterprises)[1], funded within the 7th framework programme of the European Union, is working on a portal solution providing tool support for collaborations. On the portal level, the SPIKE project also deals with the need for dynamic evaluation of security policies of individual applications based on the current context of the user. Another major concern is the potential disclosure of information through malicious portlets.

The goal of this paper is to provide a novel mech-

[1] http://www.spike-project.eu

Gmelch O. and Pernul G. (2010).
PREVENTING MALICIOUS PORTLETS FROM COMMUNICATING AND INTERCEPTING IN COLLABORATION PORTALS.
In *Proceedings of the International Conference on Security and Cryptography*, pages 177-182
DOI: 10.5220/0002990301770182

anism for secure communication among individual portlets in order to prevent unintended information disclosure by malicious portlets. A malicious portlet can be characterized as hosting an application which is only trusted to a limited extent, but whose applications' functionality cannot be abstained from. In this context, an approach is presented which targets at a standards-compliant integration of elements of the XACML architecture in accordance with the Java portlet specifications. Based on this integration, XACML policies are used to evaluate the distribution of specific events in a portal user session to other portlets deployed on the same page. Using these policies, it will be possible to exclude malicious portlets from receiving any but the bare minimum of required information about other surrounding portlets used in the same portal session simultaneously, whilst preserving functionality of all other portlet applications. At the same time, the solution will allow for flexible changes in portlet communication policies without the need to redeploy the portlets in use.

This paper is structured as follows: Following this introduction on inter-portlet communication and corresponding security requirements, section 2 presents relevant work related to the topic of both portal systems and inter-portlet communication. Section 3 then provides further information about security policies, which are then further elaborated upon and practically implemented into the setting of SPIKE in section 4. Section 5 presents future work of the authors and concludes this paper.

2 RELATED WORK AND BUILDING BLOCKS

Collaborative software as introduced in section 1 in most cases is built around a portal system. A rather technical definition of the term portal is given in the Java Portlet Specification JSR 168. According to this specification, "a portal is a web-based application that – commonly – provides personalization, single-sign-on, content aggregation from different sources and hosts on the presentation layer of information systems. Aggregation is the action of integrating content from different sources within a webpage." In the context of Java-based portal servers as employed by SPIKE, it was identified a major drawback of the Java portlet specification JSR 168 that individual portlets running in one portal could not communicate with each other (Yang and Allan, 2006), whilst security considerations only were ascribed a subordinated role. Thus, work has been carried out to enable inter-portlet communication in an extension to JSR 168, focusing on communication between individual portlets, what is known as inter-portlet communication. Noteworthy in this context are the works of Beeson and Wright (Beeson and Wright, 2005), focusing on the reusability aspects of portlets in Grid-based environments, including a portable inter-portlet communication mechanism. Moreno et al. also follow the paradigm of modelling inter-portlet communication in a provider-independent manner (Moreno et al., 2005). Likewise, Priebe et al. introduce the concept of a communication bus, enabling portlets to publish their current user context, which can be picked up by other portlets and used to show related information (Priebe and Pernul, 2003). Song et al. focus on the usage of HTML fragments in order to achieve portlet interoperability (Song et al., 2007).

With the advent of JSR 286 extending JSR 168, a standardized solution has been made available "to send and retrieve events and perform state changes or send further events as a result of processing an event" (Hepper, 2008). After the occurence of a portlet event, the event is communicated to the portlet event broker provided by the portal server, looking up all registered event types and associated portlets. It is the portlet event broker that finally distributes portlet events to any associated portlet capable of processing it. Security requirements, however, in the form that inter-portlet communication is subject to security considerations, play only a subordinated role. Another closely related drawback to the inter-portlet communication solution provided by JSR 286 is its lack of flexibility. During deployment of a portlet, its communication policy is defined and taken into account by the portal server afterwards. This way, a malicious portlet, acting as a trojan horse, can subscribe to all events occuring in the portal session, leading to the possibility of unintended information disclosure of potentially confidential information, for instance about customer or accounting information active in another portlet instance running within the same portal context. Likewise, the approach imposed by JSR 286 does not consider any potentially available context information – i.e., about the active collaboration or the current workflow. This all leads to the necessity of security policies defining the allowed communication behaviour of a portlet in a specific context.

The area of security policies is widely dominated by the XACML standard, whose current version 2.0 has been published in 2005 by OASIS (Moses et al., 2005) with efforts towards release 3.0 ongoing since. Based on an abstract model as defined by IETF in (Yavatkar et al., 2000) and (Westerinen et al., 2001) and ISO/IEC in (ISO/IEC, 1996), XACML provides a standardized method regarding access control for re-

sources and describes how rules or request/response messages are to be defined in order to enable dynamic authorization based on either a set of provided attributes or available context information. Following a common description of entities and their attributes, XACML allows for fine-granular access control going beyond a static *permit* or *deny* result, taking into account runtime aspects of distributed systems.

One great advantage regarding an implementation of secure inter-portlet communication backed by XACML is its distributed approach, which enables separation between policy enforcement and policy definition. This is of special importance when used in distributed environments like collaboration platforms spanning various sources of software from individual collaboration members. Hence, it is possible to have the definition of a baseline policy per portlet application provided by the application itself, which can then be accepted by the portlet administrator or further limited regarding the application's intended communication behaviour and trust level. Likewise, the evaluation of aforementioned portlet communication policies can be performed during runtime, introducing a greater level of flexibility in comparison to the approach currently followed by JSR 286-compliant implementations as outlined in section 2.

Recent activities in the area of policies in combination with portal systems have mainly focused on the aspects of achieving a uniform login into portal systems. A significant number of portal solution vendors is already offering support for XACML-based login into their systems, for instance in the Liferay portal server[2]. Yin et al. have extended the idea of portal authorization to the employment of web services (Yin et al., 2007). The idea of federated authentication and authorization for portal systems in Grid environments based on the PERMIS framework was first presented by Chadwick et al. (Chadwick et al., 2005) and further elaborated by Vullings et al. (Vullings et al., 2007).

The OpenPortal project strives to provide an open-source portal server, originally derived from the Sun Java System Portal Server. As a feature enhancement to the standard implementation in JSR 286, the portlet container of OpenPortal server in version 2.0 introduced a portlet policy, governing access to events, container events and public render parameters (Sun Microsystems, Inc., 2008). However, the feature remains vaguely documented and so far has not created further impact onto other projects whilst providing only limited extensions to the original specification in JSR 286.

In this paper, we will employ the XACML standard in order to provide a mechanism for definition and evaluation of security policies in collaboration settings, imposing restrictions on inter-portlet communication. Based on the portlet standards JSR 168 and JSR 286, a possible solution will be shown.

[2]http://www.liferay.com

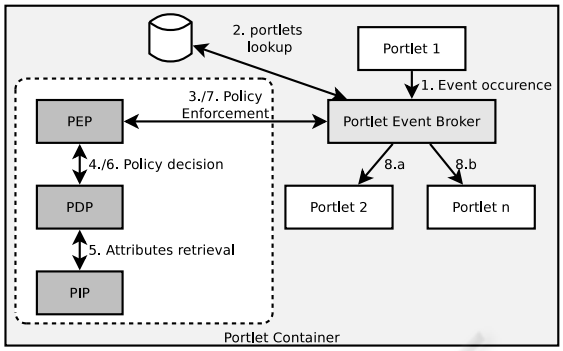

Figure 1: XACML architecture for secure inter-portlet communication.

3 INTER-PORTLET COMMUNICATION IN PORTAL SYSTEMS

Since the advent of standardized communication mechanisms provided by JSR 286-compliant portal servers, sending and retrieving events between individual portlets has been made possible. In this model, the portlets follow a loosely coupled paradigm with the portal server acting as a message broker in between different portlets, distributing the events according to the specifications during deployment of the individual portlets.

In the portlet configuration file *portlet.xml*, communication channels via distinct message types as provided by individual portlets are defined. Listing 1 shows an excerpt of the configuration of a maps portlet which is intended to publish information about the current location in the map via two portlet events, *SetMarkerEvent* and *MapBoundsEvent*, represented via two result types containing information about map search results.

As illustrated in listing 1, severe security limitations occur because the assignment does not take into account dynamic changes in the context of the portlets nor does it respect changes due to different users employing the same applications with differing access rights. Instead, upon appearance of an event as originating from one source portlet, the brokering mechanism of the portal server determines the recipients the event is to be delivered to, based on the static assignments during deployment time.

```
<supported-publishing-event>
  <qname xmlns:x="http://www.spike-
      project.eu">
    x:demo.SetMarkerEvent</qname>
</supported-publishing-event>
<supported-processing-event>
  <qname xmlns:x="http://www.spike-
      project.eu">
    x:demo..</qname>
</supported-processing-event>(...)
<event-definition>
  <qname xmlns:x="http://www.spike-
      project.eu">
    x:demo.MapBoundsEvent</qname>
  <value-type>SPIKEipcDemo.MapBounds</
      value-type>
</event-definition>
<event-definition>
  <qname xmlns:x="http://www.spike-
      project.eu">
    x:demo.SetMarkerEvent</qname>
  <value-type>SPIKEipcDemo.
      ResultsWrapper</value-type>
</event-definition>
```

Listing 1: Portlet configuration.

Also, listing 1 shows the definition of two event types, *x:demo.MapBoundsEvent* and *x:demo.SetMarkerEvent*. The tag *supported-publishing-event* defines which events can be published by the corresponding portlet, in this case *x.demo.SetMarkerEvent*. Contrary, the tag *supported-processing-event* defines which events a specific portlet subscribes to. In this case, the portlet configuration uses a wildcard mechanism, implying the subscription of all events labeled *x:demo.**. Going one step further, this way, all possible events could be subscribed to by a malicious actor, leading to a potential loss of confidentiality.

Given a possible subscription of all available inter-portlet events using the wildcard mechanism as laid out above, a malicious portlet could try to steal sensitive information transferred via the inter-portlet communication mechanism by sniffing it and further transferring it to an attacker's IT systems. Thus, the attacker in turn could gain internal information about a collaboration, associated companies, and processed data. This knowledge could afterwards be used to launch an attack, for instance via social engineering. Likewise, an attacker could remain entirely passive, collecting as much information as possible.

4 XACML POLICIES FOR SECURE INTER-PORTLET COMMUNICATION

To overcome the shortcomings outlined in the previous section, the XACML architecture is introduced to the portal event mechanisms as imposed by the Java specification JSR 286, focusing on extensibility and flexibility when implementing security policies for portlet communication. Starting from the inter-portlet communication as imposed by the JSR 286 standard, the authors have introduced elements of the XACML architecture into the inter-portlet communication mechanisms of the OpenPortal project[3], which is also used by the Liferay portal server.

As can be seen from figure 1 which gives a simplified overview over the integration of the portal server's event distribution with an excerpt of the basic elements of the XACML architecture, the authors have introduced the concept of a policy enforcement point (PEP) into the distribution of inter-portlet events. After the occurence of a portlet event (step 1), all portlets in question of receiving the current event as defined in the global portlet deployment policy are examined (step 2). For every portlet identified, the portlet event broker in turn requests a decision to allow or disallow communication of one event between current source and identified destination portlet. Extending the existing portlet event broker of the portal server, it is the duty of the PEP to enforce the policy of a specific source portlet as defined by the portal administrator (step 3). The PEP in turn requests a decision from the Policy Decision Point (PDP), considering source and destination portlet, including further information about the user request, i.e. the user's portal context, date and time information, and the type of event to be issued (step 4). For the final decision, the PDP queries for further attributes from the Policy Information Point (PIP, step 5). After successful attribute retrieval by the PIP, the policy decision is issued by PDP and returned back to PEP afterwards, finally taken into account by the portlet event broker (steps 6 and 7). Ultimately, the portlet event is delivered to the portlet treated by the current iteration of the global list of portlets as initially identified in step 2 (steps 8.a and 8.b).

With the introduction of policy evaluation during runtime allowing for context-dependent decisions, it is yet to be defined what constitutes the context in a portal session. First and foremost, context information should contain information about the user itself, i.e. about tasks and roles assigned to that user

[3]https://portal.dev.java.net

```
<Policy PolicyId="SPIKEipcDemoPolicy" (...)
  RuleCombiningAlgId="identifier:rule-combining-algorithm:
    permit-deny-overrides">
  <Target>(...)
    <Resources><ResourceMatch
          MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <ResourceAttributeDesignator
           AttributeId="urn:spike:names:spike:1.0:portal:portlet:portletId"
           DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
          SPIKEipcDemo</AttributeValue>
      </ResourceMatch></Resources>  (...)
  </Target>
  <Rule/>  (...)
</Policy>
```

Listing 2: Excerpt of sample portlet communication policy.

in accordance to the alliance the user is currently working in. Additionally, the user context should contain company- and alliance-related information as well. This way, it is possible to define individual application-specific portlet communication rights per user and per alliance. Individual alliances may have different security policies, thus treating one portlet application in different ways. Likewise, this opens the opportunity to limit the execution of specific applications to a predefined workflow. Furthermore, an application's communication behaviour may be limited according to date and time information. For instance, communication of a portlet may only be granted in special time frames, i.e. an application is allowed to send and/or retrieve events only during the regular operating hours of a company, hence limiting the attack vector. Apart from these two examples laid out, more scenarios of potential usage can be thought of by the use of corresponding policies.

In the implementation, a number of important attributes for usage within portlet communication policy evaluation has been identified and collected in a common SPIKE vocabulary, defining available attributes for policy decisions. Extending the example of the *SPIKEipcDemo* portlet, listing 2 shows excerpts of a policy definition in charge of regulating the portlet's communication behaviour, beginning with the specification of a rule combining algorithm as introduced in the following sections. Furthermore, the definition of a resource target is shown in listing 2, consisting of the definition of a *ResourceMatch* function, performing a string comparison. The *ResourceAttributeDesignator* induces comparison of the attribute *portletId* as provided by the SPIKE vocabulary with each evaluated portlet's id parameter, which is stated in the individual policies generated. Following the definition of *Target*, individual rules can be specified, matching different criteria as mentioned above, for instance date and time information, using the *Rule* tag, left empty in this example. As potentially conflicting interests of service user, platform provider, and the provider of a service have to be balanced, the matter of combining these individual policies becomes crucial. With the XACML standard providing different combination algorithms with regard to evaluation results of multiple policies, the *ordered-deny-overrides* algorithm has proven to be the most valuable. As security concerns of users of collaboration platforms are to be rated higher than the interests of providers of both platform and its associated services, the user should be able to deny all communication between individual applications inside a portal. Using the *ordered-deny-overrides* algorithm, policies extracted from the user's preferences can be put first into the *PolicySet* generated automatically, meaning that the user's policy can block individual event types, whilst allowing desired communication behaviour as suggested and described via a service provider's service policy.

5 CONCLUSIONS AND FUTURE WORK

This paper has shown the necessity for a secure inter-portlet communication mechanism. Based on a standard JSR 286-compliant implementation, it was shown that securing communication between individual portlets can be achieved easily whilst increasing flexibility at the same time. Compatibility with existing solutions being a major goal of the implementation, the approach presented in this paper focuses on a minimal-invasive solution with regard to the inter-

portlet communication mechanism as defined in the Java specification request (JSR) 286.

Future work of the authors is going to be performed in a variety of fields, based on the work presented in this paper. First and foremost, the authors are going to continue the work of thorough investigation of portal security aspects, predominantly in the context of virtual enterprises and alliance-wide application integration. A main cornerstone of further research is the combination of different policies stemming from individual stakeholders, i.e. service providers and consumers. Besides, the question of runtime performance aspects of the encompassed solution is to be investigated in more detail. Even though inter-portlet communication can only seldomly be characterised as a time-critical functionality to the overall system, it is interesting to perform deeper analysis on the scalability aspects of portlet communication policies.

ACKNOWLEDGEMENTS

The research leading to these results is receiving funding from the European Community's Seventh Framework Programme under grant agreement no. 217098 and from the European Regional Development Funds (ERDF). The content of this publication is the sole responsibility of the authors and in no way represents the view of the European Commission or its services.

REFERENCES

AT&T Corp. (2008). Collaboration across borders. http://www.corp.att.com/emea/docs/s5_collaboration_eng.pdf, retrieved 2010-02-26.

Beeson, B. and Wright, A. (2005). Developing reusable portals for scripted scientific codes. In *Proceedings of the First International Conference on e-Science and Grid Computing*, pages 502–507. IEEE Computer Society.

Chadwick, D., Otenko, S., and Welch, V. (2005). Using SAML to link the GLOBUS toolkit to the PERMIS authorisation infrastructure. In *Proceedings of 8th Annual IFIP TC-6 TC-11 Conference on Communications and Multimedia Security*, pages 251–261. Springer.

Hepper, S. (2008). *JSR 286: Java Portlet Specification Version 2.0*. Java Community Process.

ISO/IEC (1996). ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework. Technical report, ISO/IEC, New York, NY, USA.

Katzy, B. R. (1998). Design and implementation of virtual organizations. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences (HICSS)*, volume 4, pages 142–151, Los Alamitos, CA, USA. IEEE Computer Society.

Moreno, N., Romero, J. R., and Vallecillo, A. (2005). Incorporating cooperative portlets in web application development. In *Proceedings of the 1st Workshop on Model-Driven Web Engineering (MDWE 2005)*, pages 70–79.

Moses, T. et al. (2005). eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard.

Priebe, T. and Pernul, G. (2003). Towards integrative enterprise knowledge portals. In *CIKM '03: Proceedings of the twelfth international conference on Information and Knowledge management*, pages 216–223, New York, NY, USA. ACM Press.

Shilakes, C. C. and Tylman, J. (1998). Enterprise information portals. *Merril Lynch*.

Song, J., Wei, J., and Wan, S. (2007). An HTML fragments based approach for portlet interoperability. In *Distributed Applications and Interoperable Systems*, volume 4531/2007, pages 195–209. Springer Berlin / Heidelberg.

Sun Microsystems, Inc. (2008). *Sun Java System Portal Server 7.2 Developer's Guide*. Sun Microsystems, Inc., http://dlc.sun.com/pdf/820-2057/820-2057.pdf, retrieved 2010-02-22.

Vullings, E., Dalziel, J., and Buchhorn, M. (2007). Secure Federated Authentication and Authorisation to GRID Portal Applications using SAML and XACML. In *Journal of Research and Practice in Information Technology*, volume 39, pages 101–114. Australian Computer Society Inc.

Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. (2001). RFC3198: Terminology for Policy-Based Management. Technical report, IETF.

Yang, X. and Allan, R. (2006). Web-based Virtual Research Environments (VRE): support collaboration in e-Science. In *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, pages 184–187. IEEE Computer Society Washington, DC, USA.

Yavatkar, R., Pendarakis, D., and Guerin, R. (2000). RFC2753: A Framework for Policy-based Admission Control. Technical report, IETF.

Yin, H., Zhou, J., Wu, H., and Yu, L. (2007). A SAML/XACML Based Access Control between Portal and Web Services. In *Proceedings of the The First International Symposium on Data, Privacy, and E-Commerce*, pages 356–360. IEEE Computer Society Washington, DC, USA.