Towards Policy-based System on Privacy

ASSADARAT KHURAT, JOERG ABENDROTH
Corporate Technology
Siemens AG
Otto-Hahn-Ring 6, 81739 Munich
GERMANY

Abstract: - Policies have been widely used to govern the action of a system. With the increasing availability of sensitive data on-line, privacy becomes one of the most important aspects to be considered in e.g. health, financial and genetic information areas. However, when deployed in real systems, many different types of system failures are likely to occur due to software malfunction, human error, contradiction of internal policies with the legislation, etc. This can result in a privacy breach. In this paper we introduce a *Privacy Policy Fail-Safe* device which is based on the *Fail-Safe* concept to mitigate the effect of system failures to be in the safe state.

Key-Words: - Privacy Policy, Fail-safe, Policy-based System

1 Introduction

With the improvement of information and communication systems technology, the availability of on-line data has hugely increased. Not only can everyone easily access available public information, but also sensitive data that only authorized entities are allowed to access.

In the security area, policies have been employed to control the action of a system. With sensitive data, privacy is one of the important aspects to be considered in health, financial, and genetic information area, etc. Both the European Union by the European Union Parliament and the United States by the Department of Human Services have encouraged and specified privacy law in [1] and [2] respectively.

For a policy-based system in practice, there can be many means causing privacy failures such as policy contradiction, human error, software malfunction, etc. It is common that the policies become quite numerous and complicated. It is thus hard to manage such a huge set of policies in a consistent way. This can easily result in contradiction of policies [4] which wrong decisions are more probable to occur such as the PDP allowing an unauthorized client to obtain the requested data. In addition, when there is human interaction involved to accomplish the service, man-made error [7] is highly probable to happen arising from such things as misunderstanding, incaution and unawareness. With software flaws, the potential for a system malfunction is raised e.g. when the restricted content of data is revealed or the response message is sent to the wrong destination. Moreover, nowadays many means of communication have emerged such as e-mail, SMS, MMS, etc. The request and response messages can be transmitted in the different channel types which introduce in increased potential for system errors that the message is sent in the wrong channel.

These problems may cause conflict against the privacy law. Therefore, proper reaction of a system against failure is also very important which the *Fail-Safe* concept seems to be appropriate. The concept of Fail-Safe is not new. It has been introduced in several applications such as Integrated Circuits [3], railway applications [6], distributed computing [5], airplane navigation systems, etc. Its general idea is to limit the consequences of system or device failures so that failures occur in a harmless or in the least harmless way.

In this paper we are interested in the privacy issues of outgoing data from a policy-based system. Our goal is to improve the security of the existing policy-based systems by introducing a device called *Privacy Policy Fail-Safe (PPFS)*. The purpose of this device is to be able to handle the effect of system failures causing privacy breaches into a safe state. This new device applies the Fail-Safe concept to safeguard the outgoing sensitive data by using simple prohibition policies. The PPFS performs a simple add-on device to ensure the privacy policy in critical cases and also enhances privacy by introducing a sticky policy appended with the response message.

The remainder of this paper is organized as follows: Section 2 discusses related work. In Section 3 we describe existing policy-based system architectures and introduce a

novel concept of the PPFS device and how it is integrated into the system. Section 4 shows an example of an electronic health record service system combined with our device. Some advantages of our proposal in comparison to existing systems are presented in Section 5. Finally, Section 6 gives a summary and discusses further work.

2 Related Work

An Email Fail-Safe service [10] was proposed. It employs Policy Enforcement to provide customizable rules. Emails are filtered for detecting, managing and enforcing according to the policies of the customers at the system inbound and outbound. However, it does not provide checking of content attributes of the data.

There is a proposal [9] that proposes an extended access control model for XML to capture better the user's consent and need-to-know principles in medical record due to the privacy law. However this work can apply only with the content of the data.

3 Proposed Scheme

The architecture of a policy-based service system will be explained in the first subsection. After that the PPFS will be described.

3.1 Policy-based Service System Architecture

In a policy-based system, policy is used to designate the procedure or process of an organization. An important part of the system is called the policy enforcing engine. Two main entities of the policy enforcing engine are the Policy Enforcement Point (PEP) [8] and the Policy Decision Point (PDP) [8]. The PEP provides all relevant attributes to the PDP in order for the PDP to make a decision whether or not the requested action is allowed. Fig. 1 shows the architecture of a policy-based system. We assume that the Encryption/Decryption layer and the Authentication Authorization Support (AAS) are highly trustable entities.

To establish secure communication, secure channel techniques such as IPSec and TLS/SSL or WS-Security may be deployed. The Encryption/Decryption layer is an entity supporting this secure communication.

The Authentication Authorization Support e.g., identity provider (IdP), is an entity which helps in checking whether the client is the one who he claims to be.

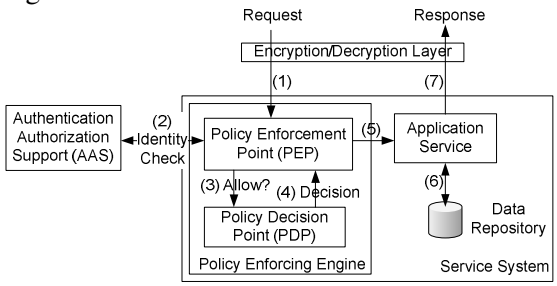

Figure 1: Policy-based System Architecture

When a client wants to obtain some data in a service system, it sends a request message to the service. The following actions then take place.

1. The identity and privilege of the requester are verified at the policy enforcing engine.
2. The PEP checks the client's identity by querying the Authentication Authorization Support (AAS).
3. Once the request is authenticated, the PEP asks the PDP to determine the client's permission. This can be done by extracting necessary attributes from the request and sending them to the PDP.
4. The PDP makes a decision by matching the obtained attributes with the appropriate policies. The PDP's decision is sent back to the PEP.
5. The PEP enforces the policy by sending the message to the Application Service.
6. If the PDP's decision is "permit", the Application Service retrieves the requested data from the Data Repository.
7. Finally the Application Service creates a response with data attached and sends it to the client.

3.2 Policy-based System with Privacy Policy Fail-Safe

In this work we are interested in the case where a failure in this procedure occurs. Such a failure may result in a security breach, e.g. when the output data is sent to the wrong recipient. Possible causes of failure might include policy contradiction, human error, software malfunction, etc. To mitigate against the effect of system errors causing security breaches, we specify three main criteria for outgoing message verification:

1. **Permit recipient**: This criterion checks whether the recipient has the privilege to obtain the data attached in the response message. Therefore the information is only sent to the clients who have the right to receive it.

2. **Channel information**: This criterion guarantees that the response message is sent through the correct secure channel corresponding with the recipient. For this purpose, the channel information is verified before sending out the response.
3. **Data content**: Since the identifiable data is very sensitive information, this criteria assures that this information complies with the privacy law.

With these three criteria, we can be sure that even if policy contradiction, software malfunction and human error, etc. occur, the outgoing message is at least sent to a permitted client via the right secure channel with the data content that conforms to the privacy law.

Our proposal for a modified and enhanced policy-based system is shown in Fig. 2. This new system includes a *Privacy Policy Fail-Safe* device (PPFS) and a *Channel Information* entity.

For the PPFS, we have applied the concept of Fail-Safe whereby the outbound message is verified according to the PPFS privacy rules and appropriate actions are taken such that any prohibited situation will never happen. The PPFS is also based on policies, but these are much simpler and easier to manage compared to the ones employed in the policy enforcing engine. It does not check every case. Instead the verification occurs only for unacceptable situations, which results in simpler policies. When any conflict is detected, the PPFS creates an error report and sends it to the administrator.

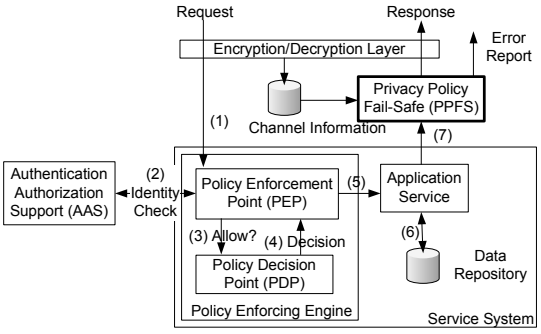

Figure 2: Policy-based System with PPFS

We have also introduced the channel information that is used to avoid errors in sending out the data to the wrong channel. Such a problem is likely to happen when the requester wants to receive data through different communication channels. The channel information repository stores channel related information and in particular the other end-point of the channel. These will be checked by the PPFS when there is an outgoing message with data contained coming out from the application service. The channel related information is collected from the Encryption/Decryption Layer.

These two entities will be described in detail in the following subsections. Note that we consider only the case of an outgoing message with privacy related data.

3.2.1 Privacy Policy Fail-Safe Architecture

Fig. 3 shows the components inside the PPFS. It consists of four entities which are *Fail-Safe Policy Enforcement Point* (Fail-Safe PEP), Fail-*Safe Policy Decision Point* (Fail-Safe PDP), *Fail-Safe Enforcement Point* and *Fail-Safe Policy Access Point* (Fail-Safe PAP).

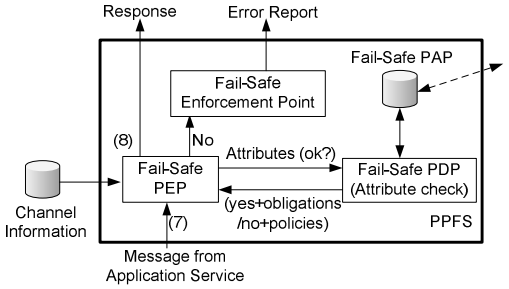

Figure 3: Privacy Policy Fail-Safe Architecture

The *Fail-Safe PEP* handles the verification of the response from the Application Service. When there is data contained in the response message, it extracts the necessary attribute values and delivers them to the Fail-Safe PDP. After receiving the result, it either sends the related attributes together with the policies which do not comply with each other to the Fail-Safe Enforcement Point or forwards the response message with optional obligations to the client (8).

The *Fail-Safe PDP* decides the result either "*yes*" or "*no*" using attributes from the Fail-Safe PEP and policies from the Fail-Safe PAP.

The *Fail-Safe PAP* is a repository containing all policies such as legislations which cannot be overwritten and privacy related obligations (sticky policy) which can be partly defined by the authorized users as shown in dashed line in Fig.3.

The *Fail-Safe Enforcement Point* creates an error report which will be sent to the administrator when it receives information from the Fail-Safe PEP.

Fig. 4 shows the flowchart of the PPFS. Once there is a response message from the Application Service, the Fail-Safe PEP checks the message whether there is data attached. If there is no data, it forwards the response message to the client. We do not consider messages without data since they have no severe effect on the

privacy issue. If there is data contained, the Fail-Safe PEP extracts the necessary attributes and sends them to the Fail-Safe PDP for verification. The Fail-Safe PDP queries appropriate policies from the Fail-Safe PAP to use for its decision. If the decision from the Fail-Safe PDP is "*yes*" for the recipient permission, channel information and restricted data content check sequentially, the privacy related obligations may be attached with the response message before sending it out to the client. If the answer is "*no*", meaning that either the recipient has no right to obtain the data, channel information mismatches or identifiable data content is not protected, the Fail-Safe PEP safeguards the data by not transferring the response message to the client. Instead it sends the related attributes together with the policies which do not conform to each other to the Fail-Safe Enforcement Point. The Fail-Safe Enforcement Point uses this information to create an error report which is sent to the administrator.

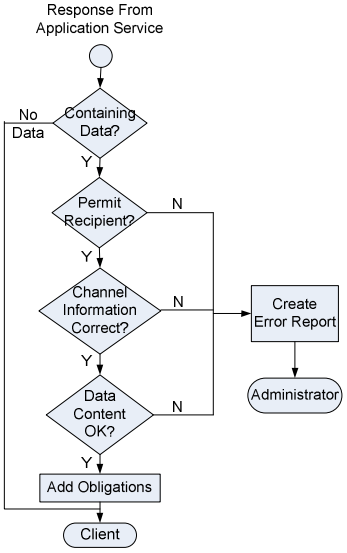

Figure 4: PPFS flowchart

3.2.2 Channel Information

Channel information is a logical (or virtual) set of data that contains records about the different communication channels that can be used to contact external communication partners (end-points) securely. A channel may be as abstract as a public key, belonging to a known communication partner or as concrete as an existing IPsec security association or TLS connection. Each logical channel record is indexed by a unique channel index (*ch_idx*), and corresponds to an authenticated identity (*end-point*) and contains further information about the channel type, the addressing information of that channel (IP-address, Port, etc.) depending on the type of channel. The Channel information may be seen as a dictionary, that may be used either to decide which channel to use to contact a given communication partner, to decide who has sent a given message or to decide to whom a given channel will deliver information.

The *channel index* field identifies the channel that the end-user employs to contact the service. In an implementation, it may depend on the communication means such as TLS, IPSec and WS-Security. For these cases, their channel indexes can be session ID, security parameter index and public key respectively. The *end-point* indicates the username (could be an authenticated pseudonym) of the client. The *Channel type* field shows the channel type and/or address of which the message is transmitted. An example of channel information is shown in Table 1 below:

Table 1: Channel Information

Channel Index	End-point	Channel type
Public Key	Bob	MMS:0179-1468302
SessionID	Bob	TLS:Socket
Public Key	Alice	Email:Alice@yy.com

From this table, Bob established two connections i.e. wireless using his private key to secure the data and TLS to contact the service while Alice established a connection i.e. Email using her private key to secure the content.

The channel information is updated each time that a new logical channel is being constructed, updated or deleted. This happens when a new key agreement is run (like IKE or TLS-handshake) or when keys are being enrolled with the server.

3.2.3 Privacy related Obligations

To provide more privacy, the PPFS introduces sticky policies appended to the response message, which are to be understood as obligations by the recipient. This sticky policy can be categorized in several levels according to the data. Because the obligations are provided by the client, it should have default obligations for high sensitive data so that we can avoid the case of a naïve client who is not aware of it.

3.2.4 Error Report

Since the Fail-Safe PEP examines the message step-by-step i.e., permit recipient, channel information and data content, it is thus able to show in a certain degree where an error has occurred which helps in fault analyzing and error correction. We have specified the structure of the error

report in XML schema. Only the report for channel information error is presented below due to the limitation of space. Each report denotes a policy and attributes which are not compliant with each other.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="policy" type="xs:string"/>
   <xs:element name="end_point" type="xs:string"/>
   <xs:element name="channel_index"
type="xs:string"/>
   <xs:element name="channel_type"
type="xs:string"/>
   <xs:element
name="time_stamp"type="xs:dateTime""/>
       …
   <xs:element name="permit-
recipient">…</xs:element>
   <xs:element name="channel_information">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="policy"/>
            <xs:element ref="channel_index"/>
            <xs:element ref="end_point"/>
            <xs:element ref="channel_type"/>
            <xs:element ref="time_stamp"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="data-content">…</xs:element>
   <xs:element name="error">
      <xs:complexType>
         <xs:choice>
            <xs:element ref="permit-recipient"/>
            <xs:element ref="channel-
information"/>
            <xs:element ref="data-content"/>
         </xs:choice>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

4 Example

Below is an example of an Electronic Health Record (EHR) service in which the PPFS is integrated. The EHR service is a service that provides medical records of clients who are registered with it. The medical record is divided into several categorizes which the clients can choose to conceal part of the data. After registration, each client obtains a username and a token which is used to identify himself during the authentication process. In this scenario, Bob requests his medical record via EHR service website using TLS and wants to receive the data on his PDA (MMS). Since sending data via MMS is not secure, the EHR service uses Bob's public key to encrypt the data before sending. However due to some software failure, the wrong public key is selected during the service process. For the EHR service without PPFS, this error cannot be realized. The result is that the owner of the private key corresponding to this public key can read the data. With the add-on PPFS, this problem is detected and the response message is prevented as shown in Fig. 5. Additionally, an error report is created and is sent to the administrator.

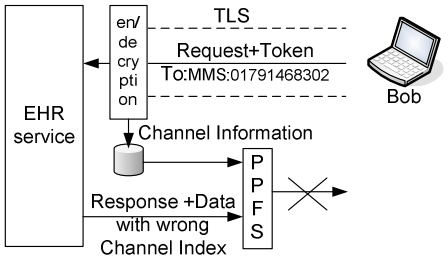

Figure 5: A scenario of EHR service with PPFS

When the response message is sent from the Application Service, the following checking sequences take place:

1. The PPFS verifies the message with its policies which can be described into four steps. Note that for the other cases which are not specified in the policies, we assume the answer to be "yes". Each step is performed in sequence by checking with policies as follows:

- First, policy for permit recipient check.
  $permission(recipient, data\_category) = false$
  $\rightarrow answer = no$
- Second, policy for channel information check.
  $end\text{-}point(ch\_idx(response)) \neq recipient$
  $\rightarrow answer = no$
- Third, policy for data content check.
  $item(data) \in identifiable\_item\_list$
  $\rightarrow answer = no$
- Finally, which requirement will be attached in the response message is governed by obligation policies. The first policy is for data with concealed item while the other is for data without concealed item.
  $item(data) \in concealed\_item\_list(data)$
  $\rightarrow attach(obg\_level1)$
  $item(data) \notin concealed\_item\_list(data)$
  $\rightarrow attach(obg\_level2)$

2. When the wrong secure response channel is detected, an error report is created. This report denotes that the response message corresponding to recipient Bob contains wrong channel information after checking with a policy stating in the report. This assists in narrowing down the

target domain for the administrator to find and correct the error.

```
<error>
   <channel_information>
      <policy>…</policy>
      <channel_index>public_key</channel_index>
      <end_point>Bob</end_point>
      <channel_type>MMS:01791468302</channel_type>
      <time_stamp>2006-08-14T17:34:00</time_stamp>
   </channel_information>
</error>
```

5 Advantages of PPFS

The PPFS has a number of advantages over the existing policy-based system. These advantages are simpler policies, permit recipient checking to assure the permission of the recipient to obtain particular data, channel checking to avoid sending to the wrong channel, effective error report mechanism to ease error correction, additional obligations to obtain more privacy, and added security due to extra checking of the response message.

The policies at PPFS are much simpler than at the Policy Enforcing Engine in several aspects:

- **No identity proving** – We have not specified the PPFS to perform identity proving, since it concerns only the outgoing message.
- **Less sophisticated constraints** – not checking all attributes and not applying to all cases. Since the PPFS policies are specified to govern for forbidden cases, only some related constraints, some important attributes and critical situation are concerned.
- **Not administrated**. – Compared with the policies from the policy enforcing engine changed from time to time, the PPFS policies are not altered except for some part of the sticky policies which is trivial.

6 Conclusions

Due to possible causes of system failures which can result in a privacy breach, we propose the *Privacy Policy Fail-Safe* device to handle these problems in safe state. Our device aims to verify the outgoing message containing sensitive data and to make the system robust against critical privacy breaches in the system. We assure that the critical prohibition case will never happen by checking three criteria. We have also presented an example of the EHR service with PPFS integrated.

Our main contribution is the proposal of a novel system architecture for a policy-based system. The main advantages of our proposed scheme are the added security, the tolerance to policy failures, channel checking, recipient permission check, the additional error report mechanism and also the fact that our solution can be used as a simple "add-on" feature in the design of secure systems.

The medical application is a promising area which meets our interest. After implementing the PPFS, we then plan to integrate it with the electronic health record service system and specify its policies for this service.

References:

[1] The European Parliament and the Council of the European Union, Directive on privacy and electronic communications, *Official Journal of the European Communities*, 2002.

[2] Department of Health and Human Services, Standards for Privacy of Individually Identifiable Health Information; Final Rule, *Federal Register*, Vol. 67, No. 157, 2002.

[3] M. Lubaszewski and B. Courtois, A Reliable Fail-Safe System, *IEEE Transactions on Computers*, Vol. 47, No. 2, 1998.

[4] T. Jaeger, X. Zhang and A. Edwards, Policy Management Using Access Control Spaces, *ACM Transactions on Information and System Security*, Vol. 6, No. 3, 2003, pp. 327-364.

[5] J. H. Abawajy, Fault-Tolerant Scheduling Policy for Grid Computing Systems, *In Proceedings of the 18th International Parallel and Distributed Processing Symposium*, IEEE Computer Society, 2004.

[6] D. Essamé, J. Arlat, D. Powell, Available Fail-Safe Systems, *In Proceedings of the 6th IEEE Computer Society Workshop on Future Trends*, 1997.

[7] J. Reason, *Human Error,* Cambridge University Press, Cambridge, 1990.

[8] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, Terminology for Policy-Based Management, RFC 3198, 2001.

[9] B. Finance, S. Medjdoub and P. Pucheral, Privacy of Medical Records: from Law Principles to Practice, *In Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, 2005.

[10] Mail2World Inc., Email FailSafe Policy Enforcement Service, Email FailSafe Service, http://www.mail2world.net/net/services/EFPES.asp.