Refined Method for Choosing a Security Solution for Mobile Application – Web Service Interactions

Silvia Trif, Adrian Visoiu
University of Economic Studies
Department of Economic Informatics and Cybernetics
Bucharest
ROMANI
silviatrif@gmail.com, adrian.visoiu@csie.ase.ro

Abstract: This paper aims to present a refined algorithm for choosing the appropriate security implementation for mobile applications connecting to web services. Common security scenarios are presented. Each scenario has several characteristics associated. The correlations between these security scenarios characteristics are computed and selected only the characteristics that are less correlated. The proposed algorithm inventories the available scenarios, inventories the requirements and selects the security scenario that match. A case study applies the proposed method to a situation where a WP7 application and a WCF service are interacting. Security scenarios for WCF services and their characteristics are described. Application security requirements are established. A security solution is proposed based on the results of the proposed method.

Key-Words: mobile applications, WCF, security characteristics, web service, security scenarios, refinement

1. Introduction

Mobile applications need to have implemented security solutions. There are various types of security models and choosing one of these models implies taking into account their characteristics. The particular case of applications that address remote functionality over the network, establishing dialogues with web services is discussed.

In the process of choosing the proper security model, an important role is played by the configuration of client and service security.

The configuration depends on the location of the client or service, if it is in intranet then transport security is less risky but otherwise security providers must be taken into account such as the transport security HTTPS or Windows Integrated.

2. Problem Formulation

The design phase of a mobile application connecting to a web service must take into account several aspects.

The type of web service defines the type of interaction. If the web service already exists, then the application must adapt to interact with it. Consider the fact that there are various types of web services, for example REST or SOAP. The effort to adapt the application is given by the existence of tools and API for connecting to such web services on the platform of development. Otherwise, if the web service is developed in line with the application, then the effort is reduces as design tries to match the capabilities of both the application platform and web service platform in order to obtain an optimum result.

The application platform comes with many restrictions in terms of performance, usability, API. Regarding security, each mobile platform has its restrictions, for example, on Windows Phone there are many limitations on installing client certificates. Regarding API, each provider offers specific implementations and support, for example in Windows Phone there is integrated support for WCF services and legacy web services, but no special support for RESTful web services.

The problem discussed refers to the case where a mobile application and a web service are considered. There is a need for secure communication between the application and the web service. The mobile platform offers a series of
characteristics that permit several security implementations. Also, the web service type is able to offer a series of security scenarios. These two parties must be taken into account, along with the description of the desired security characteristics. A solution for implementing security must be found to satisfy all the established requirements.

3. Problem Solution

3.1. Method for choosing an appropriate security solution

Choosing a security solution requires following steps:

1. Identify the web service implementation: web services are hosted on various platforms; web services use various protocols for data exchange; this identifies what types of protocols are compatible with the application that is developed.

2. Identify security features that the web service platform provides; check if encryption is available and what are the ways of achieving it: at transport level, at message level; check what are the limitations for certificate usage; identify how security is addressed for: authentication, authorization, integrity, confidentiality; these identify the security scenarios able to be implemented; the list of scenarios is denoted by $S_1$, $S_2$, $S_n$, where $n$ is the number of available security scenarios.

3. Establish the security characteristics derived from the available scenarios that are relevant for the application type that is developed; the list of characteristics is denoted by $C_1$, $C_2$, $C_m$, where $m$ is the number of characteristics.

4. Define and assess the complexity of implementing each scenario; the effort of implementation must be estimated; the effort is usually correlated to relevant complexity measures; the complexities for each scenario are denoted by $Q_1$, $Q_2$, ..., $Q_n$.

5. The security scenarios and their associated characteristics are aggregated as shown by table 1, where $V_{ij}$ denotes the value of characteristic $C_j$ for scenario $S_i$.

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$V_{11}$</td>
<td>$V_{n1}$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$V_{12}$</td>
<td>$V_{n2}$</td>
</tr>
<tr>
<td>$C_m$</td>
<td>$V_{1m}$</td>
<td>$V_{nm}$</td>
</tr>
<tr>
<td>$Q_1$</td>
<td>$Q_2$</td>
<td>$Q_n$</td>
</tr>
</tbody>
</table>

6. The optimum scenario $S_k$ and the list of desired values for characteristics is defined as needed by the design of the mobile application, denoted by $V_{k1}$, $V_{k2}$, $V_{km}$.

7. The table is searched for a match scenarios $S_f$ for which $V_{f1}=V_{k1}$ and $V_{f2}=V_{k2}$ and ... and $V_{fm}=V_{km}$.

8. If there is no match scenario found then the chosen desired security characteristics are too restrictive to be implemented; an application security redesign is needed to match one of the available options.

9. If only one scenario is found then it is a valid solution that may be further implemented.

10. If more than one scenario matches the requirements then further analysis is required to see the impact of each scenario on the implementation; in this discussion, more characteristics may be added to limit the number of eligible scenarios.

11. Also, if more scenarios are eligible, $S_{t1}$, $S_{t2}$, $S_{tg}$, then the security criterion is satisfied; further the effort criterion must be minimized; it is chosen the scenario $S_f$ with the minimum complexity $Q_f=\min \{Q_{t1}, Q_{t2}, ...Q_{tg}\}$.

In the case of no found scenario, meaning that there are desired security characteristics that are not feasible directly, meaning API is not available, effort may also be directed to implement missing features or implement custom
solutions that cover the requirements. An example of custom security implementation is presented in [3] where a message level security using shared keys is implemented using unsecured transport and basic unsecured message exchange. The security measures are implemented over the unsecured layer and this approach open ways for interoperability being easily implemented on other mobile platforms.

Between the identified characteristics can be identified correlations and is useful to operate with a small, refined list of characteristics, formed only by the relevant characteristics. Using a refined list of characteristics gave the following advantages:

- the sets of data are obtained more easier, because the data need to comply less characteristics, the data are more homogenous;
- the result is obtained easier and faster.

In table 2 is presented a generic refined matrix of the security characteristics $C_i$ followed, obtained using the correlation indicator between the characteristic $C_i$ and $C_j$:

$$\rho(C_i, C_j) = \text{correl}(C_i, C_j).$$

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>...</th>
<th>$C_i$</th>
<th>...</th>
<th>$C_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>$\rho(C_1, C_1)$</td>
<td>$\rho(C_1, C_2)$</td>
<td>...</td>
<td>$\rho(C_1, C_i)$</td>
<td>...</td>
<td>$\rho(C_1, C_m)$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$\rho(C_2, C_1)$</td>
<td>$\rho(C_2, C_2)$</td>
<td>...</td>
<td>$\rho(C_2, C_i)$</td>
<td>...</td>
<td>$\rho(C_2, C_m)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$C_i$</td>
<td>$\rho(C_i, C_1)$</td>
<td>$\rho(C_i, C_2)$</td>
<td>...</td>
<td>$\rho(C_i, C_i)$</td>
<td>...</td>
<td>$\rho(C_i, C_m)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$C_m$</td>
<td>$\rho(C_m, C_1)$</td>
<td>$\rho(C_m, C_2)$</td>
<td>...</td>
<td>$\rho(C_m, C_i)$</td>
<td>...</td>
<td>$\rho(C_m, C_m)$</td>
</tr>
</tbody>
</table>

When the correlation coefficient between two characteristics $C_i$ and $C_j$ $\rho(C_i, C_j)$ is bigger than a limit value, $\rho_0$, the elimination of a characteristic is decided. The algorithm of eliminating a characteristic is computing the medium correlation of every characteristic and is kept the one with the smallest medium correlation. Like this the kept characteristic is more representative.

The medium correlation for a characteristics $C_i$ is computed using the following formula:

$$\rho_{\text{med}}=\frac{\sum_{j=1}^{m} \text{correl}(C_i, C_j)}{n},$$

and for characteristic $C_j$ is computed using the following formula:

$$\rho_{\text{med}}=\frac{\sum_{i=1}^{m} \text{correl}(C_j, C_i)}{n}.$$  

Using the correlation coefficient is determined the degree in which two characteristics vary (depends) one on each other. For computing the correlation coefficient, is used the following formula:

$$\rho_{XY} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y}.$$  

where:

- $\mu_X$ and $\mu_Y$ represents the mean of the data sets $X$ and $Y$.
- $\sigma_X$ and $\sigma_Y$ represents the standard deviation of the data set $X$ and $Y$.

The characteristic that has the smallest value for the indicator medium correlations, meaning that is less correlated with the other characteristics is chosen to be part of the relevant list of characteristics.

### 3.2. Case study

The proposed algorithm is tested on a case where a WP7 mobile application and a WCF web service are considered. WP7 applications are presented extensively in [6]. Windows Communication Foundation, WCF, is an API in the .Net Framework for building service-oriented applications [2]. This API is used to send messages asynchronous between various endpoints of the service. There are various scenarios depending on the messages that are exchanged by
the endpoints of a WCF service: sending streams of binary data or data sent as XML.

WCF is characterized by the following features [2]:

- **Service Orientation**: allows the development of service-oriented application; any client can connect to any service as long as the essential contracts are met;
- **Interoperability**: the WCF web service can be accessed by any type of application and is independent of the operating system of the machine where it was developed;
- **Multiple Message Patterns**: messages are exchanged asynchronously; WCF uses multithreading processes;
- **Service Metadata**: WCF allows publishing service metadata using various formats: WSDL, XML Schema; the metadata is used to automatically generate and configure clients for accessing the WCF services; the metadata can be published over HTTP and HTTPS or using the Web Service Metadata Exchange standard;
- **Data Contracts**: WCF allows the creation of classes that represent data entities; the web service generates automatically the metadata that allows clients to use the designed data types;
- **Security**: messages are encrypted and the users are required to authenticate themselves before receiving messages;
- **Multiple transports and encodings**: the messages can be sent over any of the protocols (TCP, HTTP, HTTPS);
- **Durable Messages**: the messages are saved into a database that allows the resuming of the messages if a problem occurs;
- **Transactions**: WCF uses the following transactions models: WS-AtomicTransactions, the API in the System.Transactions namespace and Microsoft Distributed Transaction Coordinator;
- **AJAX and REST Support**: WCF uses XML various formats and non-XML formats, JSON;
- **Extensibility**: the service endpoints can be customized according to the user specifications; there can be extended the bindings, the security, the metadata system, encoding and serializations.

It is necessary to identify the type of service. The WCF service in its default approach uses SOAP to exchange messages with the client as presented in [5]. The WCF service is hosted on an IIS web server. In this deployment, there are several security options available. WCF security considerations are presented in [7]. In the process of developing software products exists the following security scenarios [1]:

- **S1 - Internet unsecure Client and Service**: the public and unsecured WCF client and service has the property that the messages transported between the client and service are not signed or encrypted and communications peers are not authenticated; the endpoints have no security.
- **S2 - Intranet unsecure Client and Service**: the WCF service provides information on a secure private network; the network is secure inherently or the security is provided by a layer below the WCF infrastructure; the message sent between the client and the service is not encrypted or signed and the communication peers are authenticated at the transport layer;
- **S3 - Transport Security with an Anonymous Client**: the scenario uses the transport security (HTTPS) to ensure confidentiality and integrity; the server is authenticated with a Secure Sockets Layer (SSL) certificate, and the clients must trust the server's certificate; the client is anonymous; the message sent between the client and the server is encrypted and signed using HTTPS service; the service is authenticated using a SSL Certificate;
- **S4 - Transport Security with Basic Authentication:** the custom authentication is used; the server need a valid X.509 certificate that is used for SSL; the client must trust the server’s certificate; the messages send between the client and the service are encrypted and signed using HTTPS; the service is authenticated using a SSL layer;

- **S5 - Transport Security with Windows Authentication:** the client and service are secured by Windows authentication; the messages are secured at the transport level by Windows security;

- **S6 - Transport Security with Certificate Authentication:** a SSL certificate that is issued by a certificated authority is used to secure service and the client is authenticated at transport with a X509 certificate; both the client and the service have their own certificated used to validate the client/server identity; the messages between the service and the client are encrypted and signed using HTTPS; the service is authenticated using a SSL certificate; the client is authenticated using a X509 certificate;

- **S7 - Message Security with an Anonymous Client:** the message exchanged between the service and the client is encrypted and signed with SOAP security based on established security context; the service is authenticated and a secure security context is established;

- **S8 - Message Security with a User Name Client:** the service is authenticated with an X.509 certificate and the client is authenticated using an user name and password; the messages exchanged between the client and the service are encrypted and signed using SOAP message security;

- **S9 - Message Security with Certificate Client:** both the client and the service are authenticated with certificates; the messages exchanged between the client and the service are encrypted and signed using SOAP message security;

- **S10 - Message Security with a Windows Client:** the client and service are authenticated using Windows credential; the messages exchanged between the client and the service are encrypted and signed using SOAP message security;

- **S11 - Message Security with a Windows Client without Credential Negotiation:** the service and the client are in the same domain or trusted domains; the messages exchanged between the client and the service are encrypted and signed using SOAP message security;

- **S12 - Message Security with Mutual Certificates:** the client and the service are authenticated using certificates; the server certificate can be distributed with the application; the messages exchanged between the client and the service are encrypted and signed using SOAP message security based on service and client certificates.

For these scenarios are presented the following security characteristics:

- **integrity:** the data should not be altered;

- **authentication:** the property of validating the client/service;

- **confidentiality:** a third party cannot decrypt the exchanged message between the server and the client;

- **interoperability:** the property of being compatible with other systems;

- **binding:** the property that allows clients to comply with the designed data types;
transport; the transport protocol used
security mode; the scope of security, message or transport.

In table 3 are presented the characteristics of the presented scenarios.

Table 3. Security scenarios and their characteristics

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>18</td>
</tr>
<tr>
<td>S2</td>
<td>19</td>
</tr>
<tr>
<td>S3</td>
<td>20</td>
</tr>
<tr>
<td>S4</td>
<td>19</td>
</tr>
<tr>
<td>S5</td>
<td>21</td>
</tr>
<tr>
<td>S6</td>
<td>29</td>
</tr>
<tr>
<td>S7</td>
<td>27</td>
</tr>
<tr>
<td>S8</td>
<td>27</td>
</tr>
<tr>
<td>S9</td>
<td>32</td>
</tr>
<tr>
<td>S10</td>
<td>20</td>
</tr>
</tbody>
</table>

For each scenario complexity is assessed. It is considered that the implementation of a security scenario requires not only coding by many additional steps such as configuration. This takes much of the development time. For example, if using certificate authentication for both the server and the client then the steps are:
- certificate creation for server;
- signing server certificate;
- exporting server's public key;
- importing server's public key into the client’s trust store;
- creating client certificate;
- signing client certificate;
- export client public key;
- import client public key into server’s trust store.

If only server authentication is used, then only half of the steps are necessary. All these steps map to configuration files. We consider that the configuration reflects the effort of setting up the secure connection. For WCF, XML files are used to set up necessary parameters. The complexity of XML for each configuration is assessed using an available metric.

In [4] there are presented several metrics regarding XML file complexity. The number of tags is a simple measure that is used to assess the volume of the XML configuration file. The complexity of each considered security scenario for a sample application is presented in table 4.

Table 4. Scenario complexity

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>18</td>
</tr>
<tr>
<td>S2</td>
<td>19</td>
</tr>
<tr>
<td>S3</td>
<td>20</td>
</tr>
<tr>
<td>S4</td>
<td>19</td>
</tr>
<tr>
<td>S5</td>
<td>21</td>
</tr>
<tr>
<td>S6</td>
<td>29</td>
</tr>
<tr>
<td>S7</td>
<td>27</td>
</tr>
<tr>
<td>S8</td>
<td>27</td>
</tr>
<tr>
<td>S9</td>
<td>32</td>
</tr>
<tr>
<td>S10</td>
<td>20</td>
</tr>
</tbody>
</table>
It is observed that scenarios offering more security features require more configuration effort in order to implement them.

Regarding the application, it is desired to obtain an implementation that:
- security is scoped at transport level
- authentication is done both for server and for client
- server authenticates itself with a certificate
- client authenticates itself with username and password
- transport is over HTTPS
- there is a special need for interoperability as other mobile applications developed under other platforms may become available and need to use the same web service.

The optimum scenario $S_k$ contains the desired values $V_{k1} = \text{"Transport"}$, $V_{k2} = \text{"HTTPS"}$, $V_{k3} = \text{"Yes"}$, $V_{k4} = \text{"Yes"}$, $V_{k5} = \text{"Username/password"}$, $V_{k6}$, $V_{k7}$ and $V_{k8}$ are not specified but are implicit from the previous selection.

The search identifies a single scenario to be taken into account by application designers, which is $S_4$. The implementation will follow the guidelines of the identified scenario.

If fewer characteristics are specified then more scenarios are eligible. Take into account the case where $V_{k5}$ is not specified. In this case the eligible scenarios are $S_3$, $S_4$ and $S_6$. Considering their configuration complexity in ascending order then $S_3$ is easier to implement and configure.

Between the characteristics identified can be computed various quality measurement. The degree of correlation between these characteristics are computed, for example, it can be easier seen a correlation between the binding and interoperability. It can be easily seen that the type of interoperability is correlated with the type of binding. Interoperability characteristic has two values:
- Yes: is interoperable with another platforms
- WCF Only: is only interoperable with WCF services:

The binding characteristic has the following values:
- NetTcpBinding
- Others bindings types: BasicHttpBinding and WsHttpBinding.

For these characteristics, the correlation between them is computed to maintain only the relevant one. The computed correlations for the considered application are presented in table 5.

<table>
<thead>
<tr>
<th>Security mode</th>
<th>Transport</th>
<th>Interoperability</th>
<th>Authentication</th>
<th>Integrity</th>
<th>Confidentiality</th>
<th>Binding</th>
<th>$\rho_{med}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>C2</td>
<td>C3</td>
<td>C4</td>
<td>C5</td>
<td>C6</td>
<td>C7</td>
<td>C8</td>
</tr>
<tr>
<td>C1</td>
<td>1</td>
<td>0.7</td>
<td>0.4</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>C2</td>
<td>0.7</td>
<td>1</td>
<td>0.9</td>
<td>0.77</td>
<td>0.66</td>
<td>0.88</td>
<td>0.7</td>
</tr>
<tr>
<td>C3</td>
<td>0.4</td>
<td>0.9</td>
<td>1</td>
<td>0.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>C4</td>
<td>0.8</td>
<td>0.77</td>
<td>0.5</td>
<td>1</td>
<td>0.8</td>
<td>0.56</td>
<td>0.55</td>
</tr>
<tr>
<td>C5</td>
<td>0.8</td>
<td>0.66</td>
<td>0.4</td>
<td>0.8</td>
<td>1</td>
<td>0.44</td>
<td>0.65</td>
</tr>
<tr>
<td>C6</td>
<td>0.5</td>
<td>0.88</td>
<td>0.5</td>
<td>0.56</td>
<td>0.44</td>
<td>1</td>
<td>0.44</td>
</tr>
<tr>
<td>C7</td>
<td>0.7</td>
<td>0.7</td>
<td>0.3</td>
<td>0.55</td>
<td>0.65</td>
<td>0.44</td>
<td>1</td>
</tr>
<tr>
<td>C8</td>
<td>0.7</td>
<td>0.9</td>
<td>0.3</td>
<td>0.7</td>
<td>0.66</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>$\rho_{med}$</td>
<td>0.7</td>
<td>0.81375</td>
<td>0.5375</td>
<td>0.71</td>
<td>0.67625</td>
<td>0.6275</td>
<td>0.655</td>
</tr>
</tbody>
</table>
It is observed that the interoperability is less correlated with the others characteristics, having the medium correlation indicator value 0.53 which is less than the correlation indicator value for the binding characteristic: 0.73 the interoperability characteristic will be used as a relevant characteristic in the process of choosing the appropriate security scenario. The limit of eliminating a characteristic, $p_0$ is considered to be 0.7. Because the correlation level of the binding characteristic is bigger than the considered limit, it will be eliminating, being obtained the following refined list of characteristics, described in table 6, to be taked into account in the process of implementing of the application.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Security mode</td>
</tr>
<tr>
<td>C2</td>
<td>Transport</td>
</tr>
<tr>
<td>C3</td>
<td>Interoperability</td>
</tr>
<tr>
<td>C4</td>
<td>Authentication at server level</td>
</tr>
<tr>
<td>C5</td>
<td>Authentication at client level</td>
</tr>
<tr>
<td>C6</td>
<td>Integrity</td>
</tr>
<tr>
<td>C7</td>
<td>Confidentiality</td>
</tr>
</tbody>
</table>

Table 6. List of refined characteristics

With the new list of relevant characteristic and the desired implementation wanted to be obtain, the optimum scenario that comply all these remains: $S_4: V_{k1} = "Transport", V_{k2} = "HTTPS", V_{k3} = "Yes", V_{k4} = "Yes", V_{k5} = "Username/password."

4. Conclusion

The proposed method forces the designer to perform a systematization of available security scenarios on the development platform. Security is addressed via known characteristics. Each security scenario covers the chosen characteristics to a certain degree. The security requirements for the application to be developed are specified using the same characteristics. A match between a security scenario and requirements shows a way of implementing a security solution. Each solution has an associated complexity correlated with the effort of implementation.

Using techniques of refinement, it is easier to obtain the needed optimum security scenario. Are gained the next advantages:

- data is obtained easier;
- results are obtained faster, because the scenarios have commune and relevant characteristics, the irrelevant characteristics being eliminated.

If more scenarios are eligible, the designers must take into account such indicators of complexity. Also this approach makes the designers assess the impact of each security solution on the final product.

Acknowledgements

This work was co-financed from the European Social Fund through Sectoral Operational Programme Human Resources Development 2007-2013, project number POSDRU/107/1.5/S/77213 „Ph.D. for a career in interdisciplinary economic research at the European standards”.

References