Security Aspects in Virtual and Remote Laboratories: Implementations in the Virtual Electro Lab project

Silviu Leahu, Sorin Moraru, Adrian Pelcz, Liviu Perniu

Faculty of Electrical Engineering and Computer Science,
"Transilvania" University of Brasov, Politehnicii street, Brasov, Romania

Abstract. When it comes about accessing an application remotely, that is from another computer than that where the application resides, the security should always be of concern. Because the connection between the two computers is done through a physical space, on which we may not have any control, whether that connection is through wires or wireless, we must be on guard of any possible identity and information stealing. This paper deals with the security aspect encountered in any remotely communication between two or more computers, connected among them through a physical or wireless connection. It explains the theoretical base and the implementation of these aspects in "Virtual Electro Lab".

1 INTRODUCTION

When it comes about accessing an application remotely, that is from another computer than that where the application resides, the security should always be of concern. Because the connection between the two computers is done through a physical space, on which we may not have any control, whether that connection is through wires or wireless, we must be on guard of any possible identity and information stealing. Thus, the matter of secure communication boils down to authentication, proving someone is who he claims to be; integrity, the exchanged data between the computers is unmodified on its course; and privacy, the exchanged data between the computers cannot be seen by any third party[1].

2 AUTHENTICATION

Any communication is realized between two or more subjects. Each of them must identify itself to the others, in brief or in details, before being able to participate to that communication. After that, a continuously switch between emitter and receptor roles takes place among the subjects.

In our case we have a communication between the client computer which makes requests and two servers which respond: the LMS server and the VEL server.

Leahu S., Moraru S., Pelcz A. and Perniu L. (2004).
Security Aspects in Virtual and Remote Laboratories: Implementations in the Virtual Electro Lab project.
In *Proceedings of the First International Workshop on e-Learning and Virtual and Remote Laboratories*, pages 137-143
DOI: 10.5220/0001152001370143

2.1 The client authentication

In order to maintain a communication with more than one client, the application uses a session mechanism, meaning that it allocates resources (variables, threads, memory etc.) for each one of them, and an identification which distinguishes one client from the other. Every time a client makes a request, it has to reveal its own session ID, letting the server know its identity and the stage of the communication.

In order to initiate a communication, the client must send his username and password to the LMS server. The server will verify the existence of this user into the database and depending of the result, will allow the client to connect to the features of the application or not. In case the login succeeds, the server allocates new resources for this new client and sends the ID of this newly created session back to the client. As mentioned, the client will use this session ID every single time it makes a request to the server, identifying itself among all others connected users.

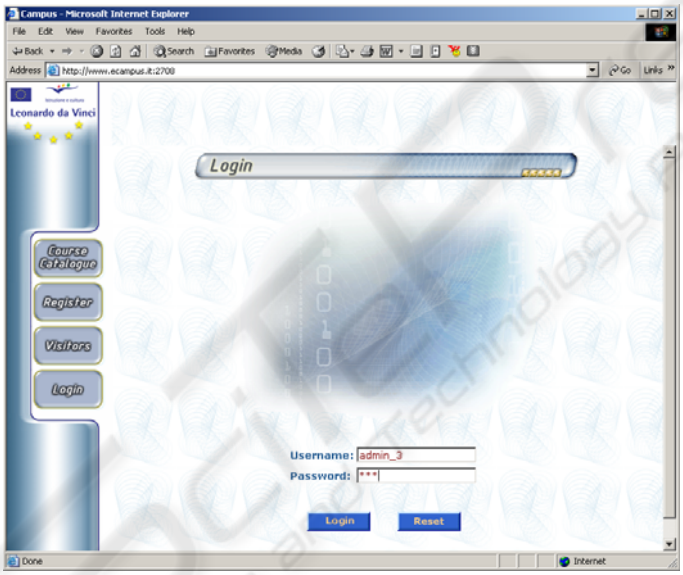

Fig. 1. Login page for the LMS application.

The use of the pair username / password has another reason too. It allows the access of the client to different features of the application, depending of his role: administrator, teacher, student etc. (fig. 1).

The VEL server has not a login mechanism of its own, but it will use the login feature of the LMS server. Thus, if a client accesses the VEL server without passing through the checking made by LMS through username/password pair, it will be rejected. The VEL server knows if a client has successfully passed the login, by looking at the session ID provided by the client along with the request.

2.2 The server authentication

Not only must the client identify itself, but also the server. Because the application of VEL is very complex, a part of it is executed on the client machine. For this reason, a Java Applet has been chosen as solution. Within it several computing operations are performed on the client part. In this way, we can reduce the network traffic, because instead of sending multiple requests to the server for different manipulation of data, we can receive the data in the Java Applet client and then all the wanted actions regarding that set of data can be done locally, without any new request to the server.

But this facility offered by the applet raises another issue. The Java Applet, being capable to execute commands on the client machine can executed unwanted commands in order to harm the client's computer.

Because of this potential problem, an applet is not allow to perform freely, but instead it is subject to a so-called “sandbox”, that is a delimited set of permitted actions, anything else outside this sandbox not being available for execution. Among the actions prohibited are accessing client's files, opening channels of communications to other computer than that where the Applet originated etc.

However, if an application really needs to access some resources on the client machine, such as the file system, the printer etc., there is a mechanism through which this can be accomplished. This mechanism is called “Applet signing”. By signing it, an applet requires at its initialization the acceptance from the user to execute actions outside the sandbox[2].

Therefore, when one loads into a browser a Java Applet, he must know who the server is in order to make a clear judgment about if he trusts it or not to let it run commands on his computer. That brings us to the server authentication matter. For this very reason a server must be able to identify itself in order to make the client trust that it is who it claims it is and allowing the clients deciding if they trust it or not. In this way, a third party server cannot claim to be someone else for deceiving the users and running malicious code on their machines.

2.2.1 Certificates

The first thing in authentication is the creation of a certificate that describes the entity of the claimer. It is made of the main details of the entity: name of person, company name, department name, location, country. This would be the equivalent of an ID of a person in the real world.

Now, this certificate must be authenticated by some authority in the domain, the same way, in the real world, an ID or a driving license is issued by an authority. In case of digital signing, there are some companies like DeutschePost, Thawte, Verisgn or Ecquifax each one known as Certificate Authority. These companies are entitled to validate those certificates by priorly verifying the credentials of the claimer. Then, it applies its own signature over the entity's certificate, thus validating it.

2.2.2 Packing the Java Applet

The next thing after having the signed certificate, is packaging all the components of the Applet in order to signing it. This is done with the JAR tool (fig. 2).

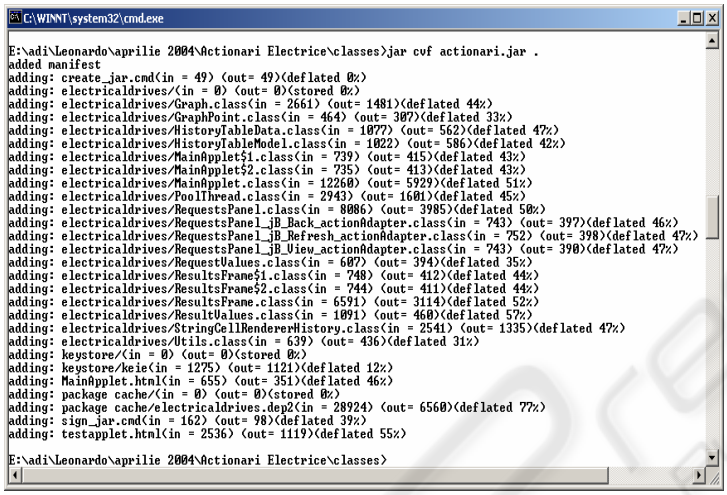

Fig. 2. Java Archive creation (JAR).

2.2.3 Signing the Applet

The third step represents the actual signing of the Applet, now packed into a Java ARchive. The tool used for the operation is *signtool*, provided with every Java Development Kit.

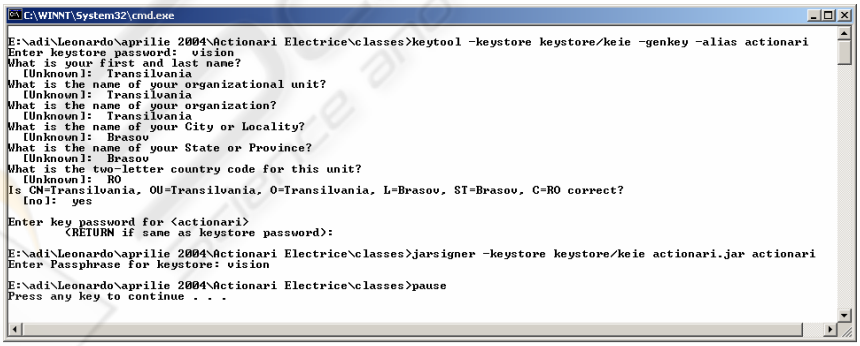

Fig. 3. Certificate creation and Java Applet signing.

We have built a script that creates a certificate (the zone marked) and that signs the JAR file (marked with green) (fig. 3).

When a user loads a web page that contains a signed applet, a warning is presented to the user (fig. 4).

The user can see the details about the certificate by clicking “More Details”.

Fig. 4. Warning of loading a signed applet, which requires additional privileges for the execution of commands outside the sandbox.

3 DATA INTEGRITY

In every communication, especially when we put a lot of trust on the data received and use it as base for our decision-making process, the integrity of data is very important. This means that the data we send and received must not be altered in any way during transmission over the network and arrives exactly as it was sent.

This desideratum is achieved with algorithms that takes a content and calculates a so-called “digital fingerprint”. This mark uniquely identifies a content of bytes, whether that content is pure text, an image, a ZIP file etc. If someone tries to change even only one bit from this content, the digital fingerprint will significantly differ from the previous one. Thus, any message can be verified regarding its original state by calculating its digital mark upon its arrival and then comparing it with the original one. If they are different, that means that the message was modified along the network it passed.

Among the most used and secure algorithms for generating the message digest are MD5 and SHA1.

4 COMMUNICATION PRIVACY

We have seen how to authenticate an entity present into a communication process, letting the others participants knowing its identity and, also, how we can assure that

our sent or received messages have not been in any manner modified along the way to their destinations.

But none of those things does address the issue of privacy. If anyone is able to “listen” the communication can have access to all the data sent back and forward, even if he doesn't altered the message, or sending his own messages. It can be a “passive” participant.

That is why we need another mechanism involved and this is the encryption.

The encryption represents the process through which a content can be transformed into a collection of codes which doesn't represents anything intelligible. The transformation is based again on algorithms and it is reversible – from this collection of codes we can retrieve the original message.

There are two ways of achieving an encryption, using the single-key approach and public/private keys approach.

4.1 The single-key or the symmetric encryption

The single-key approach is based on one key with which the content will be transformed. Then this encrypted message is transmitted over a network and decrypted with the same key by the intended recipient.

No one can get the original message unless he possesses the key used for encryption. That is why the sending of the key from one recipient to other must be done in a very secure way. The unauthorized access to it, compromises the secure of the whole communication.

4.2 Public/private keys pair or the asymmetric encryption

Its name of asymmetric encryption is due to the fact that one key is used for encrypting the message and another for decryption. The key for encryption is called the public one and is generally available in the Internet. With this key, anyone can encrypt a message to the intended recipient, of whom the public key belongs, and only that recipient can decrypt the message by using its private key, which must be very securely stored. The disclosure of this private key brings to compromising the communication to this recipient, because anyone who possesses it can now gain access at the data beneath, primarily intended solely for him.

5 CONCLUSIONS

The security aspects of applications become more and more a crucial issue, due to the growing instabilities and attacks in the networks today. The solution to this matter resides, as always has, in the trade-off between the features and the security. If we want to have more and more facilities for the user, usually that means a downgrade to the tidiness of security, and vice versa.

We can't have an application made only with one of these two ingredients. An application without any security is hard to respond to the necessity of the user, due to this very lack of trust in the data processed. On the other hand, a strictly controlled application, without any concern over the facilities for the user, has no meaning because of the lack of addressing the specific functional needs.

REFERENCES

1. http://ecommerce.mit.edu/class/private/crypto_summary.html
2. http://java.sun.com/developer/onlineTraining/Security/Fundamentals/Security.html
3. Horstmann, C., Cornell, G., Core Java 2 (vol 1. and 2), Sun Microsystems Press, 2000
4. http://java.sun.com/security
5. http://www.openssl.org