Biometrics for Global Web Authentication: an Open Source Java/J2EE-Based Approach

Ruchir Choudhry
ruchirchoudhry@cint.co.in;

Abstract.
J2EE based Web applications have largely spread over our multiple networks/VLAN over the last couple of years. Thus it becomes essential for the applications often need authentication across the multiple networks. Taking this point into consideration we have been working on integrating biometric verification capabilities using open source single sign on techniques. Going with this thought that the source code can be shared and can be used for multiple purposes and can enrich the knowledge of the entire biometric society we chosen a widely accepted Java-based open-source system for web authentication called Central Authentication Service (CAS) using the Java Authentication and Authorization Services (JASS).
The main idea behind this integration was to take advantage of the infrastructure provided by CAS to offer single sign-on web authentication, while improving security beyond basic mechanisms based on login and password, by adding biometrics. Thus, we make possible that any application prepared to use CAS for authenticating its users can also use our biometric extension for this purpose, supporting any Bio API-compliant biometric software or devices.

Introduction:
Biometrics are difficult to fabricate, which makes them much harder to share or steal than traditional authentication mechanisms such as passwords, tokens, certificates or smartcards, which have potential vulnerabilities due to credentials shared, forgotten, stolen or used without the consent of the owner. For robust security, the recommended approach is often to combine two or more authentication methods — a process called multi-factor authentication. For example, a highly secure installation could use three-factor authentication based on password (what the user knows), smartcard (what the user possesses), and fingerprints (who the user is).
Incorporating biometric sign-on in an application enforces a highly secure user authentication by comparing a registered biometric sample, also referred to as a biometric template, against a newly acquired biometric sample during the application login process. If the match score between the newly acquired sample and the registered template exceeds a given threshold, the authentication is successful and assures the application Provider about the identity who was really accessing application.

1 Introduction

During the last years web-based applications as mailers, forums, agendas and other specific applications have largely spread over our networks. Typically it is needed to perform user authentication when accessing to some of these web applications or services. In a normal web browsing session, the user needs to access to different applications (webmail, e-learning tools . . .) and for each one he must provide credentials in order to be allowed to use each service. This is a tedious task; a more user-friendly approach would be authenticating only once in a browsing session in order to access multiple applications. This is the basic principle of all single sign-on solutions.

Otherwise, classical techniques for electronic person authentication have several drawbacks in terms of performing reliable and user-friendly identity recognition; this occurs particularly with remote operations, where hacker attacks add to forgotten, shared, lost or stolen passwords or cards. Automatic identity verification, based on distinctive anatomical features (e.g., face, voice, fingerprint, iris, etc.) and behavioral characteristics (e.g., online/offline signature, keystroke dynamics, etc), is becoming an increasingly reliable standalone solution and attracting a great deal of attention as far as remotely-based applications are concerned [1].

Taking this aspect into considerations, we have been working on integrating biometric verification capabilities into a classical single sign-on solution for web authentication. For this purpose, we have chosen a widely accepted Java-based open-source authentication system known as Central Authentication Service (CAS) [2]. This system was originally developed at Yale University and later placed under the auspices of the Java Architectures Special Interest Group (JA-SIG). Nowadays, it has an extensive community of adopters. In fact, this open source system has quickly become the most popular single sign-on solution for universities, especially on U.S.A.

The main idea behind the integration of biometric verification functionality within the Central Authentication Service was to take advantage of the infrastructure provided by CAS to offer single sign-on web authentication, while improving security beyond basic mechanisms based on login and password, by adding biometrics. Thus, we make possible that any application prepared to use CAS for authenticating his users can also use our biometric system for this purpose, supporting any BioAPI-compliant biometric software or device in order to authenticate users. The open-source e-learning platforms Moodle, ILIAS, or Claroline are well known examples of web applications that are yet capable of relying the authentication task on CAS. We had used Moodle and ILIAS to demonstrate the usability of our biometric extension of CAS within a common web application.

The remainder of this paper is organized as follows. Section 2 is devoted to the description of the concept of single sign-on web authentication, and the open-source Central Authentication Service architecture. Section 3 presents the results of integrating biometric verification functionality within the Central Authentication
Service, in order to provide single sign-on web authentication based on any BioAPI-compliant biometric software or devices. Finally, Section 4 describes our conclusions and future research lines.
The final subsection presents the overall system, described both from a structural and functional point of view.

2 Single Sign-On with Central Authentication Service

Single sign-on is a session/user authentication process that allows a user to provide his credentials once in order to access multiple applications. The single sign-on authenticates the user to access all the applications he has been authorized to access. It eliminates future authentication requests when the user switches applications during that particular session.
Web single sign-on works strictly with applications accessed with a web browser. The request to access a web resource is intercepted either by a component in the web server, or by the application itself. Unauthenticated users are diverted to an authentication service and returned only after a successful authentication.
The JA-SIG Central Authentication Service (CAS) is an open-source single sign-on service, originally developed by Yale University. It allows web applicaBiometrics
for Web Authentication: an Open Source Java-Based Approach 3
tions the ability to defer all authentications to a trusted central server or servers. It is made up of Java servlets, and runs over any (JSP spec 1.2 compliant) servlet engine, offering a web-based authentication service. Its strong points are security, proxying features, flexibility, reliability, and its numerous client libraries freely available, including clients for Java, .Net, PHP, Perl, Apache, uPortal, Liferay and others.
Because of these advantages, CAS is used by many American Universities, with LDAP or Kerberos-based authentication. Moreover, it can be directly plugged into uPortal, chosen by the ESUP-Portail consortium, on the way to become a standard for open source portals [3]. This makes us confident in it permanence.

Fig. 1 shows a Central Authentication Service protocol with single sign-on.

The steps in the authentication protocol are as follows:
1. The actor requests a web resource protected by a Central Authentication Service. Access Manager's policy agent running in the J2EE server intercepts the request and verifies the user's SSO token, if any exists.
2. The user is authenticated by the Central Authentication Server. As a result, he obtains credentials and is forwarded to the web resource.
3. At the second attempt requesting the protected web resource, the browser Automatically sends the user credentials, it again checks the token.
4. If the token's authentication level is insufficient (or none exists) the Access Manager calls the biometric authentication service (Biometric Login Module) requesting authentication, which redirects the user to a login page prompting the user to provide username and terminal ID
5. The biometric authentication service verifies that the provided user and terminal information matches the data stored in the BiObex repository
6. The success or failure is determined

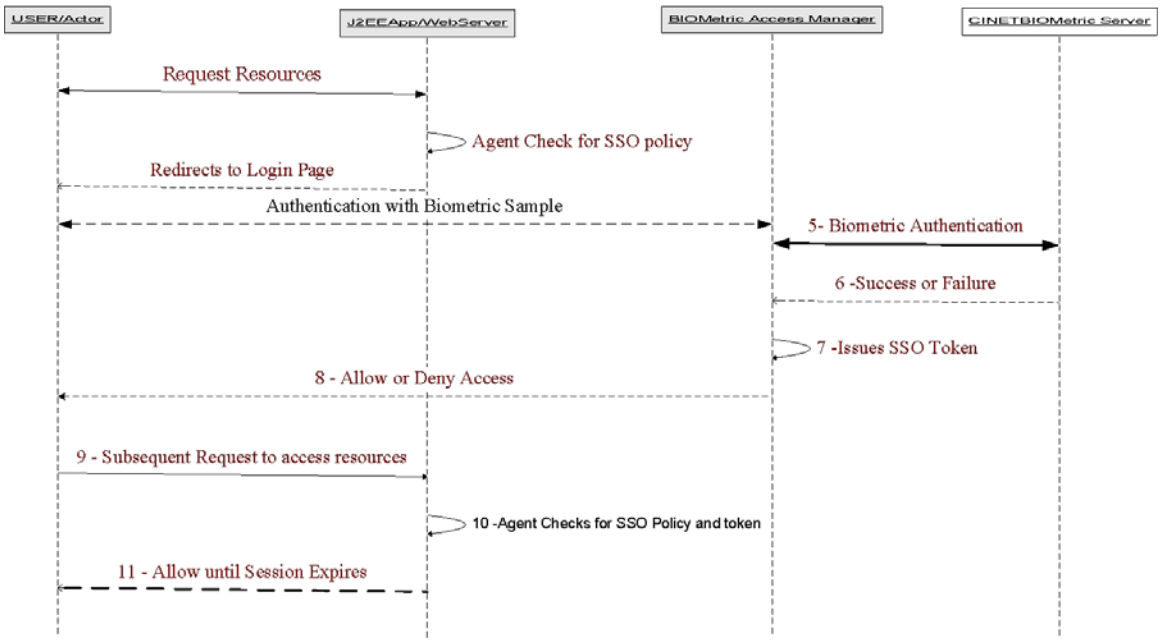

Fig. 1. Single sign-on sequence architecture

7. On success the sso token is issues
8. If denied the actor gets a failure message on the browser, in case of success
9. The actor is redirected to the subsequent menu or to the application or the access of resource is allowed.

10. The agent keeps on checking the request for sso token, based on this check it keeps giving the access to the resources
11. The actor is allowed until the session along with the token is valid.

3 Architecture for Biometric Web Authentication

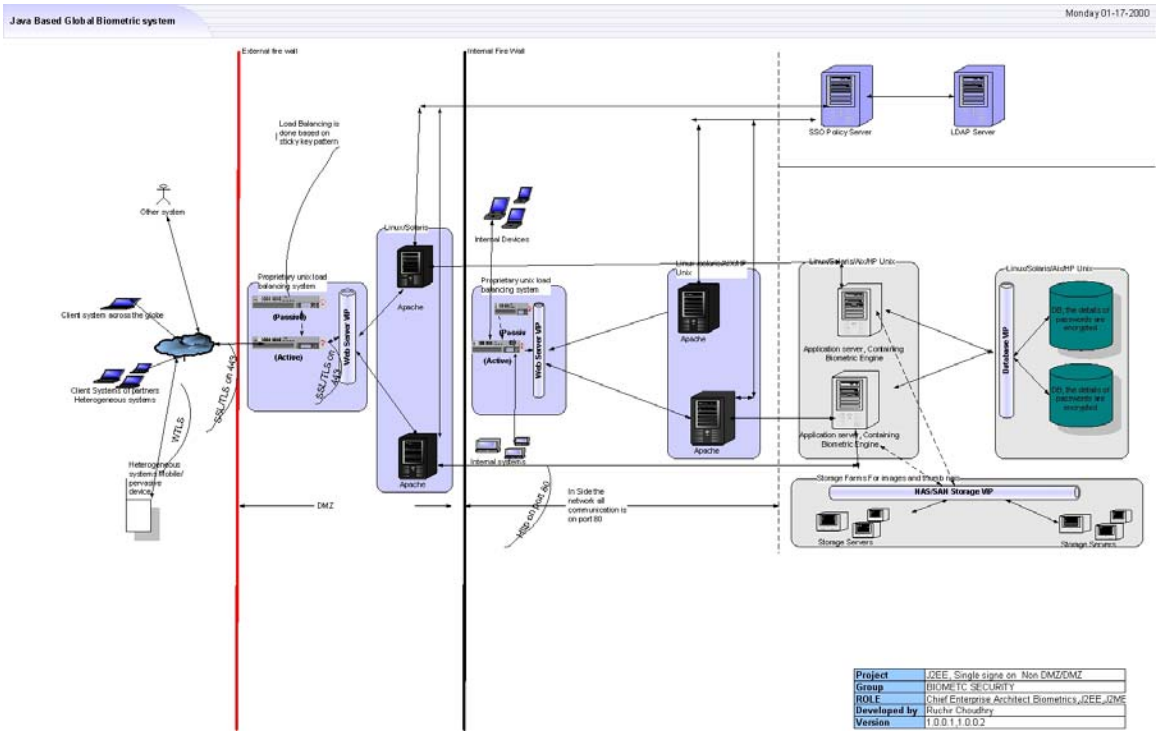

Fig. 2 Is an Architecture which expands the same concept to the enterprise/global level using the an extension of the classical Central Authentication Service protocol, JAAS(Java authentication and authorizations service)for web single sign-on, adapted to include biometric verification. The steps in the authentication protocol are as follow:

1. Initial request can come from any heterogeneous system over port 443 SSL(1024 bit encryption).
2. The call is then redirects from the load balancer to the apache servers
3. The apache server contains the web agent which redirect the call to the sso policy server
4. The sso server based on the sso ID generates a token and sends the call back to the web server
5. In case of failure the call is sent back to the client/actor
6. In case of success the call is passed back to the application server containing the biometric module
7. The access details are passed to the db to get the encryption key
8. The call is sent back to the Application sever and then the call is made to get the respective image
9. In case of success the resources are granted else failure is sent to the client

2. With the biometric module the authentication process is broken down as follows:
(a) The user launches the biometric client.
(b) Server-side biometric verification is performed.
(c) The result of the biometric verification is stored in a server database.
(d) The user request credentials to the Central Authentication Server.
(e) In order to provide valid credentials to the user, the Central Authentication Server checks the result of the corresponding biometric verification.
3. Request.
4. Validation.
The main guidelines for the development of a Java-based biometric system to be integrated within the Central Authentication Service for web single signon were focused on security, interoperability and usability issues [4]. For this purpose, some widely accepted standards in the field of biometrics were adopted.

Summarizing:
– Security: In order to comply with the Core Security Requirements of the ANSI X9.84 standard for Biometric Information Management [5], SSL(Secure socket Layer Security),WTLS(Wireless transport layer Security) connections are used, and local disk writing of user samples is avoided, for instance.
Biometrics for Web Authentication: an Open Source Java-Based Approach 5
– Interoperability: A great deal of attention has been paid to the design of a client-server architecture capable of controlling any kind of biometric software or device compliant with the standard BioAPI and frvt [6] [7].With this goal, an open source Java Native Interface wrapper for the BioAPI framework on Linux/Unix has been used [8]. To integrate into our system, this Java wrapper has been extended to include Windows support and access to low-level BioAPI primitives [9].
– Usability: The user interacts with the system through a user-friendly graphical user interface. This interaction is driven by an easily configurable dialogue. Thus, verification tasks are modeled as human-machine dialogues Specified by an XML document which describes the sample acquisition process and the biometric verification mode.
Fig. 2 depicts a Architecture diagram of the biometric authentication system itself, Detailing the functionality corresponding to step 2 presented on Fig. 2. Starting from a client verification or enrolment request, the successive actions and Functionalities are explained as follows (see diagram numbering):
Fig. 2. Building blocks and functionality description of the biometric authentication System
1. The biometric client application obtains, from the server, an XML document that specifies the human-machine dialogue with the enrolment or verification process description.
2. The client application interprets the protocol contained in the XML dialogue, prompts the corresponding information to the user, acquires the biometric sample, and performs an enrolment or verification.

3. Each time a biometric sample is required, the sample is captured from the corresponding BioAPI-compliant module. For this purpose, the client application calls the BioAPI_Capture primitive using the Java Native Interface wrapper for the BioAPI framework. BioAPI-compliant modules are also called Biometric Service Providers or BSPs.
4. The result of the acquisition process is sent to the server bound to an enrolment or verification request.
5. The enrolment or verification process is executed in the server as a sequence of BioAPI calls.
6. The verification results or enrolment templates are stored in the server database.
7. The database with the biometric verification results will be available to finally authenticate users for the web through the Centralized Authentication Service (CAS).

4 Conclusion and Future Work

We have successfully integrated biometric verification functionality in a truly global environment, within a widely accepted open source solution for single sign-on web authentication called Central Authentication Service (CAS) in conjunction with JAAS Framework . Thus, any application prepared to use CAS and JAAS for authenticating its users can also use our biometric extension for this purpose.
The overall system provides single sign-on web authentication beyond basic mechanisms based on login and password, by adding biometrics. Concretely, our biometric extension of CAS supports any BioAPI-compliant biometric software or devices in order to authenticate users.
As a result, any BioAPI-compliant kind of biometric verification could be used in order to get single-sign-on web authentication.
In order to demonstrate the usability of our biometric extension of CAS, we have tested successfully the overall system with different web, client server based, wireless based applications that allows the use of CAS to authenticate users Current version of the presented system for biometric authentication is available on http://cint.us/Default.aspx
We are continuously working to improve the system and to integrate it with the pervasive /mobile computing platforms and to make it more easy to use without compermising the core security functionality.
Acknowledgments. This project has been a part of core product of Biometrics.
MEC under the project PRESA TEC2005-07212 and the European NoE BioSecure.

References

1. Jain (A.), Bolle (R.), Pankanti (S.): Introduction to Biometrics. In Biometrics. Personal Identification in Networked Society. Kluwer Academic Publishers, 2000.
Biometrics for Web Authentication: an Open Source Java-Based Approach 7
2. JA-SIG (Java Architectures Special Interest Group) Central Authentication Service (CAS): http://www.ja-sig.org/products/cas/
3. Aubry P., Mathieu V., Marchal J., ESUP-Portail: open source Single Sign-On with CAS (Central Authentication Service) Proceedings of EUNIS04 - IT Innovation in a Changing World, Bled (Slovenia)Fl´orez, O.W.: An Open Framework For Distributed Biometric Authentication In A

Web Environment, Annals of Telecommunications. Vol. 62, No. 1-2. Special issue on multimodal biometrics
5. http://www.biometrics.gov/Standards/Default.aspx
6. BioAPI Consortium: http://www.bioapi.org
7. http://www.frvt.org/DLs/FERET7.pdf
8. JBioAPI, A library of tools for accessing BioAPI-compliant biometric service providers in Java: http://code.google.com/p/jbioapi/
9. http://developers.sun.com/identity/reference/techart/bioauthentication.html Java wrapper for the BioAPI framework. Submitted to Computer Standards & Interfaces.
10. Biometrics for Web Authentication: http://sourceforge.net/projects/biowebauth/
11. http://www.biometrics.dod.mil/CurrentInitiatives/architecture.aspx