Using Kerberos protocol for Single Sign-On in Identity Management Systems

Ivan Milenković, Olja Latinović, Dejan Simić

Faculty of Organizational Sciences, University of Belgrade, Belgrade, Republic of Serbia

ivan.milenkovic@fon.bg.ac.rs, olja.l@apeiron-uni.eu; dejan.simic@fon.bg.ac.rs

Critical review

DOI: **10.7251/JIT1301027M** UDC: **004.738.5:005.41**

Abstract: Today, identity management systems are widely used in different types of organizations, from academic and government institutions to large enterprises. An important feature of identity management systems is the Single Sign-On functionality. Single Sign-On allows users to authenticate once, and freely use all services and resources available to them afterwards. In this paper, we present the usage of Kerberos in identity management systems. An overview of Kerberos protocol, state of the art of identity management systems and different generic architectures for identity management is given in the paper. Also, we present a Single Sign-On identity management architecture proposal based on Kerberos protocol, and discuss its properties. Special attention was given to authentication, authorization and auditing.

Keywords: identity management, authentication, Kerberos, Single Sign-On.

Introduction

In modern corporate environment, identity management systems are of an utter importance. Today, many companies have hundreds or thousands of employees. With the advent of distributed systems, each user has accounts for several different applications, which are accessed remotely over the network [6]. These applications vary from webmail to inventory management, and may use various authentication methods. As the number of applications and users rises, the risk of attacks such as identity theft or identity disclosure also increases. Therefore, an appropriate set of policies, methods and rules must be applied [3].

Most applications require separate authentication, and do not provide means for centralized management. For example, let us consider a following scenario. To use several applications, user has to memorize separate password for each of the applications. This situation has several consequences. As first, the more passwords users have to memorize, the greater is the likelihood of using insecure passwords, which are easy for attackers to guess. Even strong password policies can be compromised by use of birthdays, personal id numbers or personal name derivatives. As second, the more passwords user has to memorize, the greater is the likelihood of forgetting some of them. For large enterprises, helpdesk expenses can raise to a significant amount. Moreover, users waste their time because they need to separately authenticate to each application.

In order to solve this problem, an identity management system should be used. Identity management system is responsible for different activities - identification, authentication, authorization, user provisioning and auditing [7]. Identification is the process of claiming user identity, while the process of verifying user identity is labeled as authentication. Some authors, like [13], refer to identification as a part of authentication process. User authentication methods can be divided in three categories - password based authentication, use of digital certificates or tokens and biometric based authentication. Password based authentication methods use something that user knows, tokens are based on something that user poses, while biometric authentication uses person's physiological or behavioral characteristic - something that user is. Authorization is the process of asserting user rights to access resources or services. User provisioning is correlated with authorization, as it manages user authorization privileges to be consistent with user role in the enterprise. The last, but not the least important activity is user auditing, the activity of tracking and logging system events.

Another advantage of using identity management system is the Single Sign-On (SSO) functionality. Single Sign-On allows users to authenticate only once, and freely use all services and resources available to them afterwards. In that way, users have to manage only one set of authentication credentials in order to access SSO-enabled services and resources [12]. Single Sign-On authentication has several benefits, such as increased security and usability. Problematical scenario described before in this section can be solved by the use of SSO.

A possible implementation of Single Sign-On functionality can be achieved by the use of Kerberos protocol. This paper will present a SSO architecture based on the Kerberos protocol. In the next section problem statement is given, and main reasons for the use of Single Sign-On are given. In section 3 a description of the Kerberos protocol is given. Section 4 describes state of the art of both of brand-name and open source identity management systems. Section 5 describes generic Single Sign-On architectures for identity management. In section 6, an architecture proposal is given. The last section summarizes our conclusions and gives recommendations for future work.

Problem definition

In this paper the problem of Single Sign-On, primarily for corporate environment, is considered. There are several reasons for using the Single Sign-On functionality. Possible benefits gained by the use of Single Sign-On can be substantial.

As the number of applications per system user increases, it gets more difficult to manage authentication credentials. In the case of password based authentication, users have to memorize password they use for each application. This is a possible threat for system overall security, as some of the users may choose passwords that are easy to guess [4]. Another downside of this approach is the increased number of help desk calls because of forgotten passwords. It is estimated that an average help desk call costs about US$25 [10].

Each time system user has to authenticate to a service, the login process consumes some time. Although this fact may seem of a minor significance, if there is a large number of authentication requests, total time consumed may sum up to a significant amount. Unsuccessful authentication attempts because of improperly entered credentials can additionally extend this waste of time and productivity. User experience also suffers from a large number of required login attempts. If the authentication and authorization process is more convenient, then users will be more inclined to use available applications and services more frequently.

Single Sign-On is also tightly connected with user provisioning and authorization. The use of the SSO functionality allows easier maintenance of user accounts and privileges. However, when implementing Single Sign-On functionality, it is necessary to take special care of security of authentication credentials. If these credentials are compromised, potential imposter gains access to a wide range of system applications and services. Also, mutual authentication needs to be implemented [2], in order to prevent spoofing attacks. In following sections of this paper, an SSO architecture proposal based on the Kerberos protocol will be presented. The main goal of the proposed architecture will be to address these issues.

Kerberos protocol

Kerberos was developed at MIT under the Project Athena and became the most widely deployed system for authentication and authorization in modern computer networks. The first three versions were used internally at MIT. The primary designers of Kerberos version 4 are Steve Miller and Clifford Neuman [11]. They published that version in the late 1980. Version 5 was designed by John Kohl and Clifford Neuman in 1993 and finally MIT made an implementation of Kerberos freely available. Kerberos uses symmetric-key cryptography (system where both the client and the server share a common key that is used to encrypt and decrypt network communication) to authenticate users to network services [8].

Kerberos authentication protocol offers the possibility of reliable authentication over an open network. It provides a mechanism for authentication - and mutual authentication - between a client and a server, or between two different servers. Kerberos protocol uses specially formatted data packets, which are known as tickets where they pass through the network instead of passwords. Kerberos messages are encrypted with encryption keys to ensure that no one can tamper with the client's ticket or with other data in a Kerberos message.

Kerberos authentication process is conducted like this: The client sends a request to the authentication server (AS) for "credentials" for the server. The result is a coded key for the client. Credentials consist of a "ticket" for the server and a temporary encryption key ("session key"). The client transmits the ticket to the server. The session key is used to authenticate the client and may optionally be used to authenticate the server.

If the client machine can decrypt the ticket encrypted in user password, then it is considered that the user is authenticated. If a target service is able to decrypt an encrypted ticket using its own secret key, the service may presume that the user who presented the ticket is authentic. The main benefit of such protocol architecture is that no system or party in the Kerberos exchange has access to sufficient information to impersonate any other system or party. There is no password passing through the open network.

Authorization model is based on the principle that every service knows the user, so that each one can maintain its own authorization information. In fact, the Kerberos Authentication System can be expanded by information and algorithms that can be used for authorization.

Protocol implementation requires that one or more authentication servers run on physically secure hosts. The authentication servers maintain a database of principals and their secret keys. Code libraries provide encryption and implement the Kerberos protocol. Typical network application calls the Kerberos library directly or through the Generic Security Services Application Programming Interface.

Figure 1 is a description of the Kerberos protocol, which looks like this:

1. The client must first contact an authentication server (AS) to receive a ticket and an encryption key.
2. Client receives a ticket granting ticket (TGT) and an encryption key. The encryption key, called the session key, is used to unlock communication between the client and the server and thereby authenticate that communication.
3. Client requests a service ticket from the Kerberos server. Service ticket includes ticket-granting ticket obtained from the previous message and an authenticator generated by the client and encrypted with the session key. When the client wants to access a particular service, it sends the ticket to a ticket-granting server (TGS).
4. The TGS gives the client a ticket that securely identifies the client to the requested service
5. The client presents the ticket to the network service it is trying to access and is granted access to the resource as many times as desired until the ticket expires.
6. Access is authorized and gives to the client prove it really is the server the client is expecting. This packet is not always requested. The client requests the server for it only when mutual authentication is necessary.

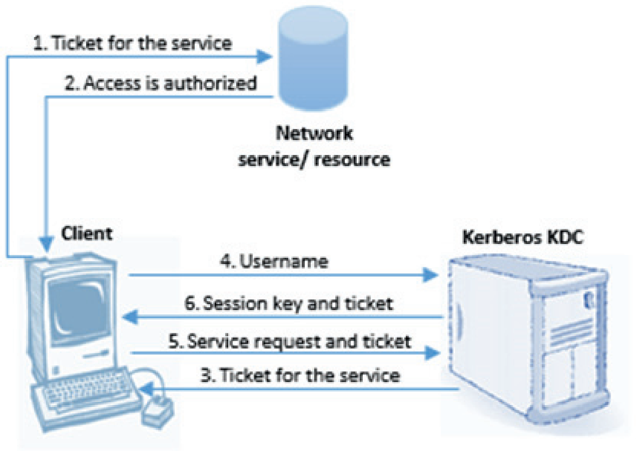
1. Ticket for the service
2. Access is authorized
Network service/ resource
Client
Kerberos KDC
4. Username
6. Session key and ticket
5. Service request and ticket
3. Ticket for the service

Figure 1 Kerberos protocol

Ticket-granting ticket (TGT) is the ticket for the full ticket-granting service. TGT presents credential in the form of an authenticator message and a ticket. It is a small amount of encrypted data that is issued by a server in the Kerberos authentication model to begin the authentication process.

Identity Management - state of the art

Identity management has grown up over the years. Critical issue around the world is how to achieve effective identity management. Most identity management systems are designed for a narrowly defined set of goals. There are many different vendors of proprietary identity management systems and open-source solutions, and their solutions offer various benefits. Today's available technology is good, but there is still room for improvements, in order to achieve better privacy protection and security.

Identity management solutions from Microsoft offer: Efficient and secure delivery of e-services, seamless user experience across boundaries, simplified management, application development efficiency and Single Sign-On (SSO) experience across borders, platforms, and various authentication methods. Microsoft account enables users to log into many websites using one account.

Hewlett Packard provides a single, unified technology platform which offers the movement of biometric information from the device to the database, from the edge to the enterprise. HP's identity management solutions incorporate identity lifecycle management, federation services, directory management and access management, with reporting and auditing services. HP IceWall SSO is Web Single Sign-On software of Hewlett Packard Company.

IBM's Identity management solutions allow operative management of the entire identity lifecycle: assessing, planning, implementing, auditing, monitoring and maintaining identities and access privileges. Among SSO solutions provided by IBM, the most relevant representative is the IBM Tivoli identity management [1]. It provides the ability to combine Single Sign-On, strong authentication and audit tracking without change of the existing infrastructure.

OpenIAM is an open source identity management solution. There are two types of products: Identity Manager (Password Management, Provisioning, Audit and Compliance, Self-Service, Delegated Admin) and Access Manager (RBAC, XACML, Federation and SSO, Web Access Control, SOA Security).

Yale University created an authentication system which is called Central Authentication Service (CAS). Since 2004, CAS is a member of Java Architectures Special Interest Group. Formerly called "Yale CAS", CAS is now also known as "Jasig CAS". When a client wants to authenticate, the application redirects to the CAS, which confirms the authenticity, and checks your username and password in the database (such as Kerberos).

In 2000, MACE working group started Shibboleth project which solved problems in sharing resources between organizations with often wildly different authentication and authorization infrastructures. This project created an architecture and open-source implementation for Identity management and federated identity-based authentication and authorization infrastructure based on Security Assertion Markup Language (SAML).

Another open-source project is FreeIPA by Red Hat which combines Linux (Fedora), 389 Directory Server, MIT Kerberos for authentication and Single Sign-On, NTP, DNS, Dogtag (Certificate System). It consists of a web interface and command-line administration tools. FreeIPA can be used for managing DNS domains, defining password policies and for integration with NIS domains and netgroups.

Generic identity management architectures for Single Sign-On

Single Sign-on functionality largely depends on system architecture. For example federated identity management has some additional requirements, when compared to centralized identity management. In this section, the main elements of identity management systems are described by using block diagrams. However, before we can describe different architectures, it is necessary to define basic components present in every generic architecture.

System user is a consumer of services provided by the system. User must own at least one identity in order to use available services within a context defined by owned identity. To confirm the claimed identity, system user communicates with identity provider. Identity provider is responsible for accepting or denying users identity, but it is also strongly tied with service provider. This way identity provider confirms or propagates identity information to service provider. Depending on the information received from identity provider, service provider allows or rejects usage of requested services.

A traditional approach to identity management is the Centralized identity management system architecture [5]. In this case a centralized identity provider and all system services are provided by a single service provider. This architecture is shown in Figure 2, and process steps are numbered. In Step 1, system user identifies himself to identity provider. After a successful identification of the system user, the system needs to authenticate the user. Identity provider is responsible for both identification, and authentication. After completion of authentication, user receives a token from identity provider (Step 2), which is passed to the service provider in Step 3. Token is valid for a certain amount of time, and users do not need to authenticate each time they send request to the service provider. Therefore, SSO requirements have been met.

Token is used by the service provider to verify user credentials and claims. This is done in Steps 4 and 5, where identity provider and service provider communicate in order to validate information carried by the token. After a successful validation, system user is eligible to use desired services (Step 6).

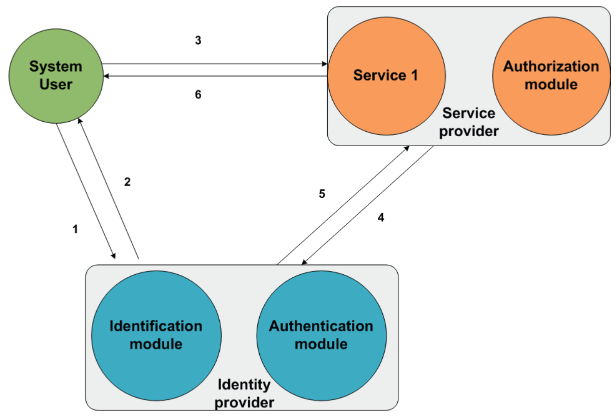

Figure 2 - Centralized identity management system architecture

In this architecture, service provider is responsible for authentication. It is important to notice that all identity management architectures described in this paper follow previously described steps.

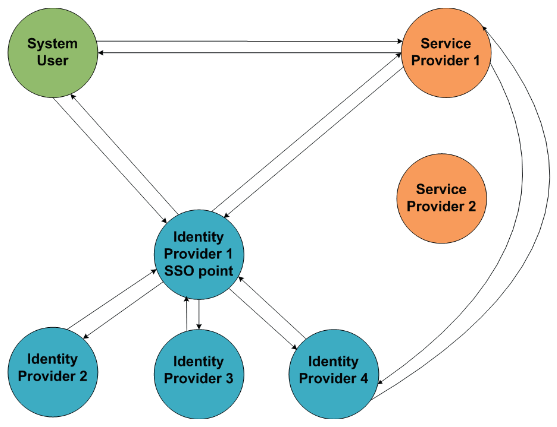

Figure 3 - Centralized SSO architecture with independent identity providers

In an alternative scenario, there is a centralized identity provider which represents a Single Sign-On point. Beneath it, on the lower level, there are numerous service specific identity providers. System user identifies and authenticates with centralized identity provider, while authorization process is delegated to service specific identity provider. It is important to highlight that centralized identity provider does not care about authorization, nor its data. After initial sign on, user only needs to authorize with service specific identity provider in order to get access to a desired service.

Next identity management architecture exploits the fact that multiple identity providers could share information they have. By sharing the information and agreeing to work together, identity providers form a "federation", thus allowing system users to identify with any identity provider belonging to the federation. Because of that, this architecture is called Federated identity management architecture [14]. The main advantage of the Federated identity management architecture is that it enables system to work even if the service provider and identification provider are not in the same organization. The Federated identity management architecture is shown in Figure 4.

Kerberos authentication - Architecture proposal

Among described architectures, Kerberos is most suitable for the use in centralized SSO architectures

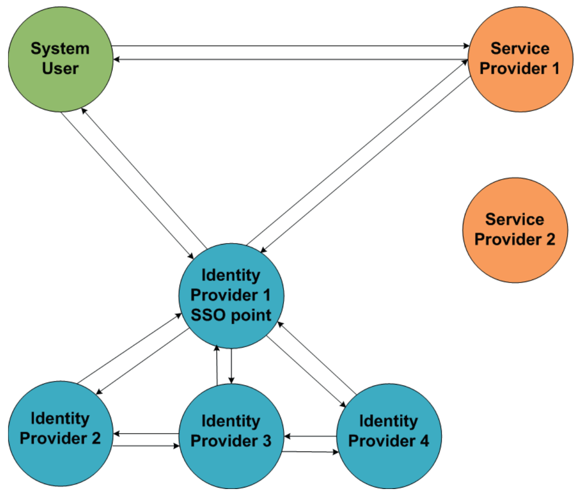

Figure 4 - Federated identity management architecture

with independent identity providers responsible for authentication. We propose the usage of Kerberos as identification and authentication module, as there are several benefits from such choice.

As first, system user authenticates with the Kerberos authentication server and receives ticket granting ticket and session key. When authenticating to service providers, system user requests service tickets from Kerberos ticket granting server. System user uses service tickets to authenticate with service providers. Independent identity providers are used for authorization. Different solutions could be used for storing authorization data, for example an LDAP directory. For example, Free IPA uses 389 Directory

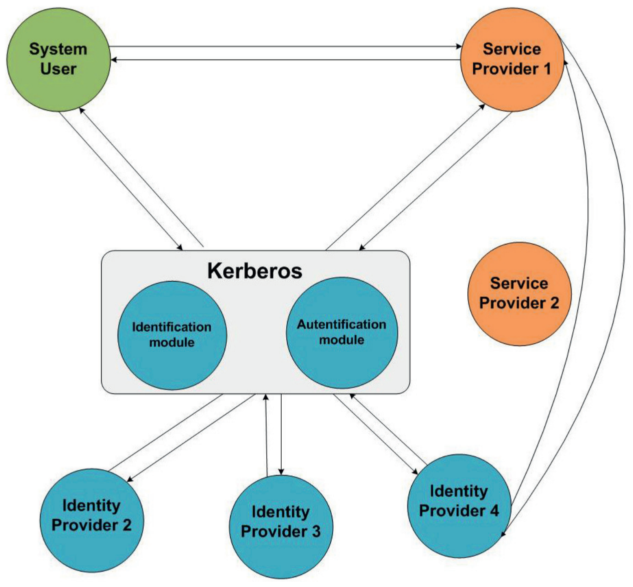

Figure 5 - Centralized SSO - Kerberos authentication

service for this purpose. Various extensions of this architecture are possible, such as the use of Security Assertion Markup Language, an XML-based open standard data format for exchanging authentication and authorization data between parties.

Architecture allows use of different authentication methods. PKI (Public Key Infrastructure) can be used for initial, or cross realm authentication [15]. Use of PKI infrastructure allows easier administration of Kerberos key distribution center, revocation of certificates, and various other benefits. Kerberos provides mutual authentication, which makes phishing and man in the middle attacks difficult and not likely to succeed. Ticket system provides safe communication through open network, as system users do not have to send unencrypted passwords.

Kerberos offers flexibility with forwardable, renewable and proxiable tickets [9]. Identity theft risks can be lowered by the use of renewable tickets. At each renewal, Kerberos KDC can check if the ticket was compromised. Proxiable tickets allow services to do task on behalf of system users, while forwardable tickets allow complete use of client's identity.

Kerberos does not provide authorization functions or auditing functions. Therefore, it is necessary to use other tools and applications for this task. In a workstation environment, one of the Kerberos weaknesses is a Spoofing Login where intruder very simple can replace the login command with a version that records users' passwords before employing them in the Kerberos dialog. Also, each network service with a different hostname will need its own set of Kerberos keys. This complicates virtual hosting and clusters.

Conclusions

In this paper, various important aspects of the Single Sign-On functionality are revised. Special attention is given to the Kerberos protocol and its use in identity management systems. Several generic Single Sign-On architectures are presented, and an architecture proposal based on the Kerberos protocol is described.

Centralized Single Sign-On architecture based on the Kerberos protocol offers various benefits. Secure mutual authentication, various authentication methods, real-world interoperability, possible integration with PKI are just some among many. Kerberos has been widely used by both academic and enterprise organizations for many different tasks. Therefore, Kerberos use is not limited only to centralized identity management. Federated SSO architecture could benefit from the use of Kerberos as well. Cross realm authentication allows user of one Kerberos realm (administrative domain) to access services from other realms. This functionality is based on sharing keys, so it has some limitations, although these restrictions can be stretched by using transitive trust relationships. Another approach for federation is based on the use of PKI.

Primary objective of this paper was not a detailed analysis of Kerberos integration with other solutions for authorization and auditing. Such analysis should be done in future research. Future work could also include a comparison of Kerberos with possible authentication alternatives, such as Distributed Computing Environment (DCE). Moreover, challenges concerned with the use of biometric authentication in Kerberos are an important problem, and should be investigated.

Acknowledgment
This work is a part of the project Multimodal biometry in identity management, funded by Ministry of Education and Science of Serbia, contract number TR-32013.

Authorship statement
Author(s) confirms that the above named article is an original work, did not previously published or is currently under consideration for any other publication.

Conflicts of interest
We declare that we have no conflicts of interest.

References

[1] Buecker, A. et al. (2008). Integrated Identity and Access Management Architectural Patterns, Retrieved from http://www.redbooks.ibm.com/redpapers/pdfs/redp4423.pdf (Accessed: May 2013).

[2] Dhamija, R. & Dusseault, L. (2008). The Seven Flaws of Identity Management: Usability and Security Challenges," IEEE Security and Privacy, vol. 6, March/April, pp. 24-29.

[3] Elliot, J. et al. (2011). Managing multiple electronic identities, retrieved from www.enisa.europe.eu (Accessed: May 2013).

[4] Haley, K. (2010). Symantec Security Response Password Survey, Retrieved from http://www.symantec.com/connect/blogs/password-survey-results (Accessed: May 2013).

[5] Hermans, J. & Valkenburg, P. (2009). "European Identity and Access Management Survey", KPMG and Everett, 2009, Retrieved from http://www.everett.it (Accessed: May 2013).

[6] Milenković, I. et al. (2012). Identity management in cloud computing, ITEO 2012, Proceeding of ITEO 2012, ISBN 978-99955-49-94-7, pp 81-87.

[7] Milenković, I. et al. (2012). Architectures of comprehensive identity and access management, EIIC 2012 (Electronic International Interdisciplinary Conference), Proceedings of the EIIC 2012, ISSN:1338-7871, ISBN 978-80-554-0551-3.

[8] MIT Kerberos Consortium, (2008). The Role of Kerberos in Modern Information Systems, Retrieved from http://www.kerberos.org/software/rolekerberos.pdf (Accessed: May 2013).

[9] Neuman, C. et al. (2005). The Kerberos network authentication service (V5). RFC 4120, retrieved from http://tools.ietf.org/html/rfc4120, (Accessed: May 2013).

[10] Oracle, Implementing Enterprise Single Sign-On in an Identity Management System, Retrieved from http://www.oracle.com/us/products/middleware/identity-management/wp-esso-idm-207215.pdf (Accessed: May 2013).

[11] Pachghare, V.K. (2009). Cryptography And Information Security, New Delhi, pp. 184-185.

[12] Pashalidis, A. & Mitchell, C.J. (2003). A taxonomy of single sign-on systems. In Information Security and Privacy (pp. 249-264). Springer Berlin Heidelberg.

[13] Prasad, G. & Rajbhandari, U. (2011). Identity Management on a Shoestring, Retrieved from http://www.infoq.com/minibooks/Identity-Management-Shoestring (Accessed: May 2013).

[14] Shim, S. et al. (2005). Federated Identity Management," IEEE Computer, vol. 38, 2005, pp. 120-122.

[15] Zhu, L. & Tung, B. (2006). Public key cryptography for initial authentication in Kerberos (PKINIT), Retrieved from http://www.ietf.org/rfc/rfc4556.txt (Accessed: May 2013).

Submitted: May 27, 2013.
Accepted: June 19, 2013.