An Authentication Middleware for Squid Proxy-Cache: a Single Sign-on Approach

John Lenon Cardoso Gardenghi
Universidade São Francisco, USF
Itatiba, Brazil
Email: john.gardenghi@live.saofrancisco.edu.br

Marcelo Augusto Gonçalves Bardi
Universidade São Francisco, USF
Itatiba, Brazil
Email: marcelo.bardi@saofrancisco.edu.br

***Abstract*—The identification process in network environments consists of knowing who is using a resource in order to aggregate security, integrity and control to it. However, the diversification of resources in such environments implies a set of credentials for the same user as well as the requirement of authentication as many times as the resources have to be accessed. In this context, Squid is an interesting tool to work because it allows the use of a third-part authentication engine, which is referred as helper. By this way, the objective of this work is to develop a new scheme of secure and centralized authentication for Squid proxy-cache, with strong support in single sign-on (SSO) strategy, optimizing safety and usability in the use of this resource. This scheme is based on a distributed application that eliminates the direct interaction between Squid identification engine and client that requests access, centralizing the authentication and identification process in a part that we call Middleware Server. It was observed a tradeoff between security and usability in the employment of SSO strategy, particularly because the usage of a central user database is the ideal scenario.**

***Keywords*-authentication; single sign-on; Squid Proxy-Cache;**

I. Introduction

Authentication is a very important issue in computer environments, because it introduces security and identification to such environments, making it possible to do access control and accounting. In this context, authentication is a crucial factor, since an user-based method is the most common strategy to implement access control. Additionally, the management of multiple user-names and passwords is not only an annoying aspect of the current Internet, it is also one of the most serious security weakness [1].

Squid is a proxy-cache (or web-cache) system widely used by network administrators to provide Internet access control and optimize Internet bandwidth utilization. Squid authentication works as follows: when the client (the user) wants to access the Internet, Squid requests its credentials, which is supplied by the user input and is sent by Hypertext Transfer Protocol (HTTP). Squid currently supports many authentication methods, grouped into four classes: basic, digest, NT LAN Manager (NTLM) and negotiate [5]. As authentication sources, we also have many options, such as file, database, Lightweight Directory Access Protocol (LDAP), Server Message Block (SMB), Network Information Service (NIS) and Pluggable Authentication Modules (PAM). However, NTLM and Negotiate are strategies that work only with Microsoft Active Directory platforms, for this reason, the main methods used are basic and digest.

The basic method supports many authentication sources, including the aforementioned ones. The disadvantage of this method is that user-name and password are sent over HTTP without any encryption. The digest method, on the other hand, supports only files and LDAP, and the credentials are sent over the HTTP protocol with encryption. But these methods have two problems that play the main role of this work: the credentials are sent over the HTTP protocol (on every HTTP request from the client) and it requires user interaction to supply them each time the web browser is opened (or some application that requires internet access).

Actually, these two issues represent problems due to some factors. Firstly, the ideal authentication source for Squid Proxy-Cache should be an unique user database that is used for other purposes, thus only one credential would be used by a single client to access the resources needed. From this point of view, the credentials transmission through the HTTP protocol may represent a security issue, since if the credential is discovered an inappropriate access to other resources can be granted. The main solution presently adopted in this case is to create a different, separate user database for Squid. This is not a good solution because it generates a set of credentials for a same user. So, from the administration point of view, Single Sign-on (SSO) would be a potential solution. Using SSO, a user authenticates itself only once and is automatically authenticated for other resources he/she needs access. Secondly, this is a usability problem, because the user has to supply its credentials every time he/she needs to access Internet. This is a problem for the user and the administrator, since many softwares do not support the input of user and password to access the Internet.

In this context, it is presented a new authentication scheme for Squid Proxy-Cache based on any user source with LDAP support, by developing a middleware application in order to answer to identification and authentication requests from Squid instead of the client answer by itself. This middleware

is a distributed application that can be easily deployed in a network environment, based on a client-server approach, with strong support on SSO concepts, that will be addressed in the next section.

II. Theoretical properties

It is important to present some theoretical properties to guarantee the agreement on the terminology used on this paper.

A. Authentication terminology

We follow De Clercq [2] and adopt the following infrastructure-related terms:

- *Authentication servers* are the machines that provide authentication functions, refers to the physical part.
- *Authentication authorities* are the entities responsible to perform authentication functions, refers to the logical trust-concept part.
- *Authentication infrastructure* is the set of authentication servers and authorities, which provide authentication services.

Access control is the mechanism used to protect network resources. It includes three processes [6]:

1) *Authentication*: often referred as identification, that consists in providing an identity to the client that is requesting access, and authentication itself, the process of verification and validation of the user identity.
2) *Authorization*: grant access to some resource based on the identity of the requesting user.
3) *Accounting*: provides auditing of the users actions and requests, also called auditing.

The user is identified in a database by an unique identity, also referred as User ID, and proves its authenticity by providing an evidence related to its identity, also referred as Password. This pair is called credential. It is important to mention that many other types of credentials than the pair user/password exist, such as smart cards and biometric identification. But in this work we consider only user/password way of identification. The process in which the user supply its credentials and the authentication infrastructure offers the authentication service based on this input is called *logon process*.

B. Single Sign-On

Recalling from the introduction, SSO is the mechanism that provides an unique way of authentication for the client, that has access granted to any resource that is needed. Pashalidis and Mitchell [3] present two main type of SSO systems: *pseudo-SSO* and *true SSO* systems. Our middleware consists in a pseudo-SSO system, with some modifications, since we are not concerned about storing the users credentials in this system, but this only works as an "authentication helper" to the Squid Proxy-Cache.

SSO is an interesting strategy to be adopted in a network environment, because it is a good trade-off between security and usability. SSO comes in favour of network administrators in the sense that an unique user database is used for many applications. In this context, SSO may represent a security issue, in two ways: first, from the availability point of view, if this database becomes unavailable, the access to many services would be injured and, finally, from the information security itself, since if someone discovers the user password, this one will have inappropriate access to many resources. But internet access is a feature that is closely related to just one user identity. It is not desirable that user has one credential just to access internet. Furthermore, internet access authentication is supposed to be something transparent to the user, thus an application that favors usability is desired.

III. Methodology

In this work, we use resources provided by Squid Proxy-Cache to purpose a third-part application, that we refer as middleware, to offer an alternative authentication service based on SSO proposal and improve the security and usability in opposite of the actual authentication scenario in Squid.

As mentioned earlier, when a client requests access, Squid asks for its credentials, which is received by HTTP protocol based on the input of the requesting user (the userID and password pair). Squid does not authenticate users by itself, it sends the received credentials to a program called "authentication helper". This program validates the user identity and returns a message to Squid. Our main point is to change this way of authentication so that the communication with the client is minimized or made in a different manner, by developing a new authentication helper. This proposal consists in two approaches:

1) If the user is already authenticated in an authentication infrastructure, the SSO approach is used so that a SSO client is deployed and used by the middleware to retrieve the current authenticated user credential.
2) Otherwise, a secure interface is offered such that the client authenticates itself into the middleware. The middleware is responsible to do the logon process and verify the user authenticity against a pre-configured authentication authority with LDAP support, validating the given user credentials.

On the next section, we present the main results of this proposal.

IV. Results

To achieve the aimed authentication engine, we divide our proposal in four parts:

A) The middleware server;
B) A SSO client;
C) A secure web interface for user logon;

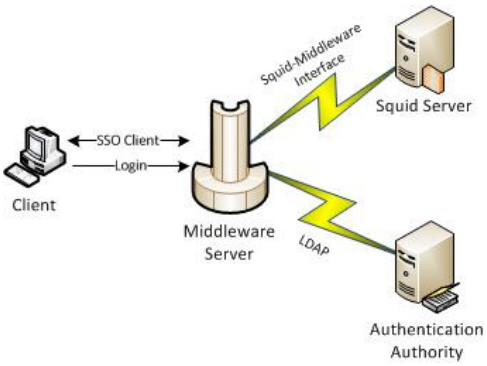


Figure 1. Main Communication Scheme

D) A communication interface between the middleware server and Squid.

We present these parts in the following sections, concluding the ideas putting it all together and describing some computational properties dealt during the development.

A. The Middleware Server

When a client requests access to internet, Squid essentially identify this client by its IP address. Based on this information, middleware server will try to retrieve current logged in user on this IP address. If no user is found, middleware server throws an error message to Squid. Due to this strategy, the direct communication between the Squid server and the client is eliminated, and the middleware server does the work.

This task is done in two ways. First, middleware server tries to contact the SSO client in this address, which is expected to return the user logged in or the absence of one. Second, it searches a cache that contains the users that logged into the middleware server through the secure web interface.

Therefore, the middleware server is the kernel of this engine and uses the other three parts to communicate with the client and the Squid server. This scheme is illustrated in Figure 1.

B. The SSO Client

The SSO client is responsible for answering requests from the middleware server about the logged in user on a workstation. It is a customizable part, middleware server is not concerned about which application runs as SSO client, neither the language that it was written. The main role of the SSO client is to send to the middleware server a string containing the current logged in user or a string informing that no user is logged in the workstation.

It is important to highlight that the SSO client is not concerned about the logon process, neither with credentials, the client is supposed to be or not to be properly logged into an authentication authority already.

In this work, a case study was done with Novell networks. It is a favorable environment because the client uses an application called Novell Client to log in. Novell Client assures the authenticity of the logged in user. As Novell provides a couple of libraries to communicate with this application, it is possible to recover this user. This is the main idea of SSO client. In this context, a C application was made in order to retrieve the current logged in userID and send it to the middleware server.

C. The Secure Logon Interface

This is a simple web interface, written in PHP, whereby the user logs in with its credentials (the user/password pair). The application then performs the logon process against the LDAP user database and, finally, insert the user in the middleware cache.

The middleware cache is a folder that contains the files created by this interface, whose content is the user ID, its time of login and its IP address. The middleware server processes this directory and stores the information of these files in the memory. The user will have to authenticate again after the time of login expires. Notice that no passwords are stored in these files, since the user identity is already validated.

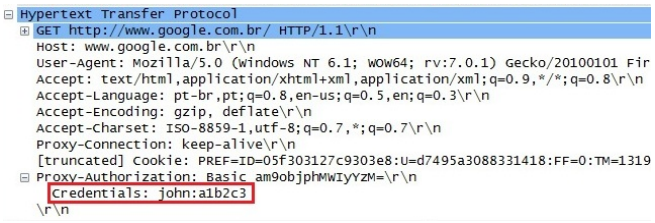
D. The Squid-Middleware Interface

This is a Perl script that receives the requests from Squid server and sends them to the middleware server, returning the answer to the Squid server. Actually, this acts as an “authentication helper” for Squid proxy-cache (see Wessels [7, Chapter 12]).

This is also a customizable part. This can be adjusted to attend to security policies requirements. The essential is that this script shall communicate Squid server and middleware server. In the current implementation, if a user-name is retrieved from a workstation, this script also verify the participation of this user in a group, to improve the access control. In this context, Squid is concerned about groups, and not users themselves. With this, the script also make LDAP queries to the authentication authority.

This middleware was developed to run under Unix technology, but since it has been written in Java, it also can run on Windows platforms. The Squid-Middleware Perl interface searches the middleware server by its IP address and change information through network sockets, so the system can be easily distributed in separate machines.

Also many Linux resources were used to accomplish the operation of this engine. For the secure web console, HTTPS was used, and some special configurations in Apache web server. An external ACL configuration was done in Squid to use the Perl interface as authentication helper, and also the use of “deny_info” pages in order to redirect the access denied information to the secure logon interface.



(a) On a basic authentication scheme.

(b) Using middleware server.

Figure 2. Wireshark HTTP captures on both scenarios. Notice that the Proxy-Authorization header field is not present in the middleware scheme anymore, since Squid is not communicating with the client directly.

Additionally, the middleware has one configuration file in order to define some parameter, namely logs directory, cache directory and logon time to live. But two configurations are noteworthy: these are the client timeout and the use of SSO. The first one refers to the maximum time that middleware server tries to communicate with SSO client. If this time expires, middleware server concludes that no SSO client is running on the target machine. The second parameter enable to the administrator the choice of using or not the SSO client, in this case, the source of authenticated users will be the middleware cache only.

E. Results and Discussion

The running application complied with the main ideas of this project. Both problems presented in the Introduction of this article were solved: with this scheme running, the user does not need to supply its credentials every time that Squid requires them and these are not sent through HTTP protocol anymore, as illustrated in Figure 2. It was also observed that Novell eDirectory integration allowed that all changes can be introduced only by the administrator, a difference that may contribute to greater security and control over compliance with institutional policies [4].

From the security point of view, the present proposal mitigates many vulnerabilities from the current Squid authentication scenario. Credentials are not stored in the middleware server neither transmitted through HTTP protocol. User identification proceeds in two ways: by the SSO client, the engine that retrieves the userID logged in a client (no user interaction required), or through the secure authentication interface, where the user provides its credentials only once, valid for a certain period of time. By the use of this strategy, we avoid passwords being transmitted through the network, as Squid does in basic and digest methods.

Another interesting problem was solved. Some applications need access to Internet but does not support the supply of credentials by the user. It is required by a common implementation of authenticated Squid. So, in these cases, the network administrator need to add many exceptions to Squid rules in order to grant direct access to such application. It works, but in direct access auditing is not done. In our proposal, transparent authorization is done, so any application that needs access to Internet will have it granted if the user is already authenticated to middleware server, and this access will be properly audited.

V. Conclusion

An interesting tradeoff between usability and security was noticed in SSO implementation, as discussed in Section II-B, particularly in this case, in which a central user database is the ideal scenario. The development of a distributed application was also worth, since the middleware server, the Squid server and the authentication authority can be running in different physical machines without problems.

Acknowledgment

We are indebted to two anonymous referees whose comments helped us to improve the present paper.

References

[1] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, “Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps”. In: *6th ACM Workshop on Formal Methods in Security Engineering*, 2008.

[2] J. De Clercq, “Single Sign-On Architectures”. In: *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002. Vol. 2437, pp. 40-58.

[3] A. Pashalidis and C. J. Mitchell, “A taxonomy of single sign-on systems”. In: *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003. Vol. 2727, pp. 249-264.

[4] L.M. Prevedello, K.P. Andriole, R. Roobian, and R. Khorasani, “Integration of the Medical Imaging Resource Center into a Teaching Hospital Network to Allow Single Sign-on Access”. In: *Informatics in Radiology*. Radiographics, 2009. Vol. 29, pp. 973-979.

[5] K. Saini, *Squid Proxy Server 3.1*. Birmingham: Packt Publishing, 2011.

[6] D. Todorov, *Mechanics of User Identification and Authorization*: Fundamentals of Identity Management. Florida: Auerbach, 2007.

[7] D. Wessels, *Squid: The Definitive Guide*. Sebastopol: O’Relly Media, 2004.