International Journal of Electronic Commerce Studies
Vol.5, No.1, pp. 81-90, 2014
doi: 10.7903/ijecs.1040

AUTHENTICATION AND LOOKUP FOR NETWORK SERVICES

Daniel J. Buehrer
National Chung Cheng University
168 University Rd., Min-Hsiung Township, Chiayi County, Taiwan, R.O.C.
dan@cs.ccu.edu.tw

Tzu-Yang Wang
National Chung Cheng University
168 University Rd., Min-Hsiung Township, Chiayi County, Taiwan, R.O.C.
wty@cs.ccu.edu.tw

ABSTRACT

Sharing on networks is common nowadays. There are many sites, and users typically must register for an account on each site. Sometimes, sites or services can communicate or share data with each other or cooperate to perform some functions together. Such intercommunication between sites uses a shared network. However, some sites may not be trusted, and the user's own data, especially passwords, might be exposed or fraudulent. Authentication is needed in order to both identify users and to hide user information via some authorization policies. In this paper, we describe a method for authentication via sessions. This authentication procedure is able to provide authentication of proxies and also allow concealed passwords. It is a little like OpenID[1] for websites, which prevents hacks and attacks from malicious servers and allows ordinary network connections. Moreover, it also allows proxy-proving, which permits only registered servers to be agents of a requesting user to request data from other servers.

Keywords: Sessional Authentication, Sharing Network, Sign-Up

1. INTRODUCTION

On the Web, there are many sites that provide services for users, and many of them need the user to login to his account. It is a challenge to remember the account name and password for each site. Moreover, the site

which we are attempting to use might be malicious and try to steal our personal information. Some identity management schemes have been proposed to solve the problem of remembering accounts and passwords on websites, such as OpenID. However, they are weak against phishing and hacking attacks. Moreover, their utilization focuses on a decentralized identity for each website. They cannot deal with the case of one site requesting authorized data from another site for a given user.

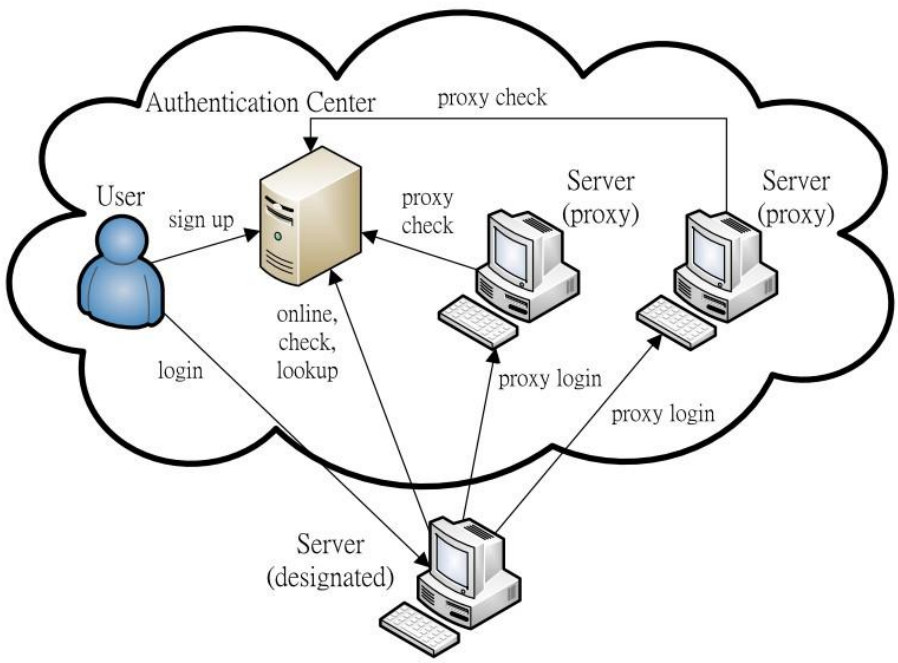

Figure 1. Three-tiered architecture

As well as serving users directly, a server can often provide data or services for other servers on a shared network. We propose a session-based authentication scheme, a little like OpenID, to address such circumstances. It uses a session and sign-up authentication that allows users to hide their passwords in a login server, permitting servers to request authorized data for the login user and preventing unauthorized servers from performing proxy requests without this user's approval. In this architecture, a server that is requested by another server performs a proxy-proving process to ensure that the requesting server has been granted permission by the requesting user. Our proxy security depends on a sign-up and session key mechanism that is supported by the authentication center (AC). It not only protects servers and clients against attacks of fake clients, servers, or proxies but also prevents password exposure. Under such session-based authentication, data and

information for authorization can be shared on the network. For instance, semantic webs[2] can share secure ontologies since servers can identify users and their groups.

The rest of this paper is organized as follows. Section 2 outlines the communication among client users and the AC and between servers for cross-querying. Section 3 describes the responsibility of the AC and the service it provides. The protocol of a client login is explained in detail in Section 4. Finally, Section 5 explains how this approach prevents hacks.

2. ARTICHECTURE

Before explaining the architecture, we first assume that the Internet is stable and that all computers have their own IP address which cannot be hijacked. Connections between users and the authentication center are secure. IPv4 exhaustion should not be a problem, since IPv6[3] will be popular in the future. IPv6 provides an incredible number of IP addresses. Every computer can have its own IP. Although IPv6 tends to use transparent end-to-end connections to get rid of Network Address Translation (NAT), our sign-up mechanism takes the IP and the port of clients as factors in constructing a session key. An IP and port pair can locate a user even within a private network. Moreover, although hijacking is not in our domain of discussion, replay and fake-id attacks in IPv6 can be avoided in ways similar to those for IPv4.

Our architecture is shown in Figure 1. We adopt a sign-up mechanism[4], for which the first step is to login to an AC server to obtain a session key. This action provides the user's ID (denoted by C) and the user's intention of where to login. Then, C receives a session key (denoted by K) if C's registration in AC succeeds. Next, C takes the session key to login to the server which C designated in the sign-up step. The designated server (denoted by M) performs a session check by asking the AC when it detects a login attempt. M may choose to reject the login from C if the session key check does not pass. The choice of M is adjustable by the security policy. That is, M can be a fully public server, so it may not do any identity check. After passing the session check, C gets permission to login to the designated server.

In addition, the AC can look up servers. However, a server may not be popular even though it has been made fully public. The reason is that some servers might not inform the AC of their availability. A server which is public but not currently registered as active can only be visible to users who already know its location (i.e. its IP and port number). Of course, this could

be separated as an independent service which does not make use of the AC, such as WSDL[5].

Communications of a user login and a proxy are shown in Figure 1, which also presents the communication for sharing. For security, we use a proxy design based on a session key. During a query from one server to another, a query needing a proxy query is sent from user C. Meanwhile, a designated server M which still holds the session key of C first invokes a proxy login to the server (denoted by P) which is to be queried. Then, P asks AC whether the K from M is verified for proxy purposes. P grants M as C's agent for a proxy connection if the check passes. P is actually a server just like M, so it can set itself as a fully public server if it wishes. If so, the proxy check is not necessary.

Unless it does not allow proxies, every server has two login services: one for users and another for proxies. If proxy logins are not allowed, there will be no sharing of data with other servers. When a proxy server notifies the AC of online activation, it can be looked up by other servers. Then, it can be safely queried under secure control. The detailed communication of sign-up and proxies are described in Section 4.

3. AUTHENTICATION CENTER

There always needs to be a mechanism to control authentication, even on shared networks. The Authentication Center (AC) is recommended as a network-global service, even though every host may set up one server of its own, like a DNS service. Decentralized identity management can be constructed, but service servers have to make a list of trusted identity providers for security. Services provided by the AC include registration, sign-up, online activation, name lookup, authentication checks, and proxy authentication checks. A global service can help to prevent problems of naming and inconsistency. In other words, a server can set its own AC, and every client user must register with the identity providers, which are trusted by servers for needed proxies. Some servers may trust some identity providers but not others. However, users still have to register with many identity providers in order to use their servers, and some servers may be phishing ones. Therefore, a global AC would be cheaper, easier, and more secure.

There are two registration services in the AC. A user has to register an ID for connecting to servers and using their services. Only a registered user can further register a server name, representing a server for connecting and querying, including proxies. Registered server owners can contact the AC for notification of online activation with respect to this server's location. The

location of a server contains at least its service IP and ports, and this information is recorded dynamically in the AC. Thus, the AC can maintain a list of servers for black lists and white lists. Other client users and servers can ask the AC for its location or load condition. Name-lookup for online servers and services makes it easy for clients to find all servers on the Web.

4. LOGIN PROTOCOL

4.1 Client Login

We described client logins in Section 2. This section explains the client login protocol in detail. Before a client user C logs onto a server, a sign-up must be sent to the AC with C, its password, and the server to which it intends to log on. C wants to log onto a designated server (denoted as M) in a few minutes. The AC then a session key back with the specified session if C's identity is verified. The session key (denoted by K_C) represents a dependency of <C, M, IP_C, P_C, TO> where IP_C and P_C are the IP address and the port of C, respectively, and TO is the time-out limit. The following actions, such as login and query, on M by C have to use the K_C for authentication. Note that the session key may be invalid if no time-out update is sent to the AC before the time-out.

Once C receives K_C, it can start a login connection to M. When M gets a login request with (C, K_C), a login session check process is invoked (see step 4 in Figure 2). M first asks AC about the verification of (ID_C, K_C, incoming IP, incoming Port) where ID_C is the ID given by the client. Note that the incoming IP and port are obtained from the network packet and are not given by the client to avoid imposter hacks. AC then checks the consistency of (ID_C, K_C, incoming IP, incoming Port) with IP_M (the IP of designated server). K_C is valid only if the existence, consistency, and time limit are all satisfied. K_C must exist. The packet <ID_C, IP_M, incoming IP, incoming Port > related to K_C must match <C, M, IP_C, P_C>. IP_M could be checked with M, since M gave its information in its activation step. Lastly, K_C should not have timed out. A session key is always invalid if it is timed out, no matter whether or not other conditions are satisfied. A granted login will be sent back to C as long as the session check is verified. Then, C can start to query M with the permissions specified in the database.

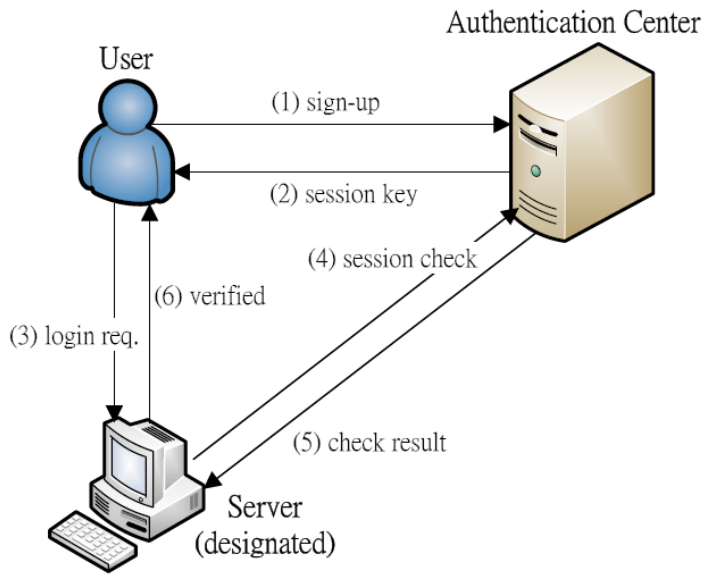

Figure 2. Protocol of client login

4.2 Proxy Login

The proxy process can both verify user session keys and prevent servers from fake proxy hacking. Figure 3 shows the granted proxy query in the direct proxy contact part and the forbidden proxy query in the indirect proxy contact part. The forbidden proxy query communication has a dashed line. A client C makes a query on a designated server M which results in some proxy queries. The figure presents a single proxy query. Before the proxy query, a proxy login is required. Its protocol is similar to the client login described in the previous subsection. The difference is that the proxy login service (denoted by P) receives (ID_C, K_C) from M and sends <ID_C, K_C, IP_M, P_M> to AC for verification. IP_M and P_M must be obtained from the network packet of the proxy login requester. The requester is assumed to be M. If P grants proxy login to M, the proxy session check is passed. M then starts a proxy query for C after receiving permission.

Proxy queries only allow direct proxies. Indirect proxies like the dashed line in the Figure 3 will be rejected. The defense against indirect proxies is described in the next section. Although this protocol prevents proxy hacking, once again servers have the option of making themselves fully public, in which case they do not need to communicate with the AC for proxy checking.

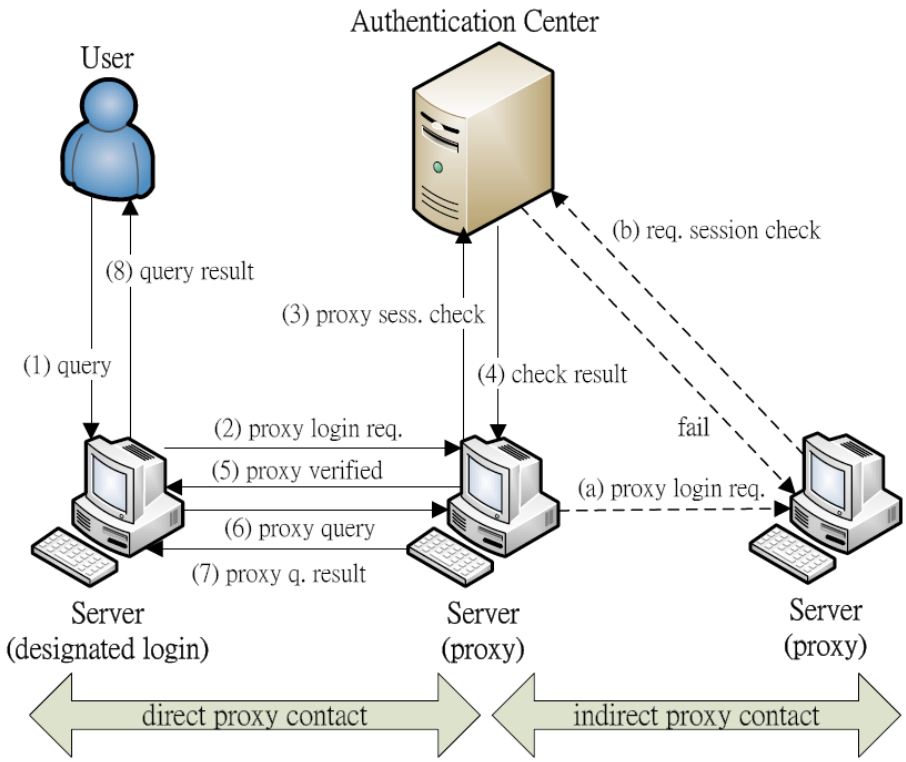

Figure 3. Protocol of a proxy query

5. DEFENDING AGAINST ATTACKS

We have described the authentication protocol. Here, we show how it defends against some hacking or phishing attacks.

A replay attack is very common: what the victim sends is repeated, so that the server thinks that the attacker is the client. This attack becomes useless, because the session key for the login server holds the dependency of K_C and $<ID_C, M, IP_C, P_C>$. The server will find that the client does not match the key returned from the AC according to the client login protocol. Moreover, the password of the client user is always kept safe during the session, because the password is only used during the connection from the client user to the AC. Even if the session key is stolen, the user is not affected because of the consistency and time limit of the key. The designated server can figure out that the key does not match the attacking client's key. The only danger for a stolen password is the connection between the client and the AC. Therefore, an encrypted connection, such as TLS/SSL[6, 7] and HTTPS, is strongly recommended.

Another possible attack is a fake server, which phishes clients to get them to login in order to obtain the session key and perform unauthorized proxy querying. Because of the consistency, the key cannot be used to login to other servers. The user's password is also kept safe. Besides fake users and fake servers, there may be a fake proxy for steps (a) and (b) shown in Figure 3. Anyone obtaining (K_C, ID_C, IP_M, IP_C, P_C) may try to make a fake proxy query to another server. That action is regarded as becoming an indirect proxy. A victimized proxy server only gets (C, K_C) from the attacker. IP_C and P_C are never used in the proxy, and IP_M and P_M are not used in the assigned way. The victimized server catches the IP and port from the network packet and then asks the AC for verification of <ID_C, K_C, attacker's IP, attacker's port>. Finally, the attacker is rejected due to the failed verification of its IP address and port number.

The sign-up mechanism plays a very crucial role in these defenses. User passwords are only employed when registering and signing up for a session or for online activation of services on the AC. The session key takes the place of a password and is safely exposable. The client and proxy login are consequently secured by distinguishing different IPs and session keys. Thus we can focus on the secure connection between clients and the AC rather than the connection between the client and server.

6. DISCUSSION

OpenID is popular with web applications for decentralized identity on authentication. Although it successfully addresses the risk of user passwords being leaked from service providers and provides a decentralized identity approach[8], it has some limitations. It is mainly used on Web services and has weak security in some situations. For instance, the redirection step may redirect users to a phishing identity provider. One proposed approach[9] strengthens the security by combining OpenID with the One-Time Password (OTP) and mobile phones. It efficiently balances the weakness of OpenID for users logging onto the identity provider by using extra device support. However, this obviously has some limitations, too. The users have to own smartphones to execute an app (i.e. an application) for challenge numbers, and the identity providers must pay an additional cost for the use of the app. Moreover, there is the risk that the algorithm and parameters of OTP may be exposed, or there may be a phishing relay party (service provider). Also, it is used only on Web services.

In our protocol, users have to actively contact the authentication center to sign up for a session, rather than passively being redirected to it. This difference prevents phishing redirection attacks. Our protocol is also

different from the point of OTP, which enhances security for the login step between users and authentication centers. Furthermore, we strengthen the verification of service providers. Our proposal has no conflicts with OTP, so the two approaches can work together to raise the security level.

Another popular protocol, OAuth[10], raises authorization issues. It is often used with OpenID, on which it is based. Our proxy contact protocol may look like an authorization issue, but it addresses authentication when the users request resources from other service providers. In this case, the designated server must be identified. This protocol can verify authenticity. Authorization is still needed for the proxy server to grant the designated services and resources to the requesting user. In other words, OAuth also can be combined with our authentication mechanism.

7. CONCLUSION

We propose a sessional authentication security mechanism for shared network services. A sign-up mechanism protects communication from attacks by fake users, servers, and proxies. It efficiently prevents a user's password from being stolen by phishing servers. After a user performs a sign-up action to the AC, the session key obtained from AC exists, is consistent, and has a time limit. The session key can then substitute for the password so that there is no worry of password exposure. The session key is used to check consistency. Compared to OpenID as a means of providing identity on websites, our authentication allows a user's session to be transmitted to any designated server. It is not limited to websites; any service can use this methodology. Designated servers are permitted to query to other servers using the protocol of the proposed architecture via proxy-querying. The protocol is also guarded by a sign-up mechanism that achieves proxy-proving. The AC contains services such as registration, sign-up, online activation, name lookup, authentication checks and proxy authentication checks. These services achieve hidden passwords, decentralized identity, and sessional proxy queries and make the sharing of network resources more convenient and more secure. We use such a sharing mechanism to share classes, relations, and their instances securely in a semantic web.

8. REFERENCES

[1] D. Recordon, and D. Reed, OpenID 2.0: A platform for user-centric identity management. In M. Winslett and A. Goto (Eds.), *Proceedings of the second ACM workshop on Digital identity management* (p11-16). New York, NY, USA: ACM Press. 2006.

http://dx.doi.org/10.1145/1179529.1179532.

[2] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web. *Scientific American*, 284(5), p34-43, 2001.

[3] S. Deering, and R. Hinden, *Internet protocol, version 6 (IPv6), IETF*. Retrieved on January 18, 2013, from http://tools.ietf.org/html/rfc2460.

[4] W.-F. Huang, Authentication system for Cadabia Database. *A thesis submitted to Institute of Computer Science and Information Engineering, National Chung Cheng University, Taiwan*, 2006.

[5] Web Services Description Language, Retrieved on January 18, 2013, from http://www.w3.org/TR/wsdl20-adjuncts.

[6] A. Freier, P. Karlton, and P. Kocher, *The secure sockets layer (SSL) protocol version 3.0, IETF*. Retrieved on January 18, 2013, from http://tools.ietf.org/html/rfc6101.

[7] T. Dierks, and E. Rescorla, *The transport layer security (TLS) protocol version 1.2, IETF.* Retrieved on January 18, 2013, from http://tools.ietf.org/html/rfc5246.

[8] E. Maler, and D. Reed, The venn of identity: Options and issues in federated identity management. *IEEE Security and Privacy*, 6(2), p16-23, 2008. http://dx.doi.org/10.1109/MSP.2008.50.

[9] H. Wang, C. Fan, S. Yang, J. Zou, and X. Zhang, A new secure OpenID authentication mechanism using one-time password (OTP). *Paper Presented at the 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Wuhan, September 23-25, 2011. http://dx.doi.org/10.1109/wicom.2011.6040525.

[10] E. Hammer-Lahav, *The OAuth 1.0 protocol, IETF*. Retrieved on January 18, 2013, from http://tools.ietf.org/html/rfc5849.