SCRAM in LDAP
Better Password-based Authentication

Kurt Zeilenga
kurt.zeilenga@isode.com

Isode Limited

Abstract

The Salted Challenge Response Authentication Mechanism (SCRAM) is a password-based authentication mechanism for use in application protocols, such as LDAP, via the Simple Authentication and Security Layer (SASL) framework. SCRAM offers a number of improvements over older password-based mechanisms, including channel bindings for use with TLS. This paper provides a summary of the history of password-based authentication in LDAP to show how mechanisms have evolved to meet the ever evolving threat model; summarizes the current situation; and discusses how the SCRAM mechanism, specifically SCRAM-SHA-1-PLUS, is the better mechanism for addressing the current threats.

Introduction

Password-based authentication is heavily used to authenticate users to Internet services from web services to various messaging services to directory services. Different password-based authentication mechanisms have been introduced over the years, each intended to address different attacks. This paper discusses the evolution of mechanisms and attacks against them.

The Salted Challenge Response Authentication Mechanism (SCRAM) is a recently introduced password-based authentication mechanism for use in application protocols. The SCRAM-SHA-1-PLUS authentication mechanism offers significant improvements over older password-based mechanisms (and well as its SCRAM-SHA-1 sibling mechanism).

This paper offers a brief history of password-based mechanisms used in LDAP and attacks against these mechanisms. The intent of this paper is to encourage developers to implement the SCRAM-SHA-1-PLUS and to generally encourage the use of this mechanism for password-based authentication.

Simple Authentication with DN and Password

LDAP has since its introduction supported Simple Authentication with DN and Password. The client provides, in the clear, an authentication identity and a password. The server historically compares this value against the clear text password it stores (generally in userPassword). There are few problems with this mechanism:

- As the authentication identity and password is sent in the clear, the mechanism is subject to eavesdropping.
- Passwords were stored in clear text. This is problematic as it anyone who has access or gains access to the stored value can impersonate the user.
- Lastly, the mechanism does not provide any server authentication.

LDAPv2 over SSL

LDAP over Secure Socket Layers (SSL) was primarily introduced to protect directory authentication and directory data exchange from eavesdropping. LDAPv2 over SSL was never formally specified.

While SSL provided for client verification of server's identity through use of Public Key facilities, this verification was not widely performed in early days of LDAPv2 over SSL. Many LDAPv2 clients did not implement authentication of the server.

The general use of LDAP over SSL subject to a wide range of active attacks. For instance, an attacker could to impersonate the server or cause the client to downgrade to unprotected LDAP.

SSL is the predecessor to the Transport Layer Security (TLS).

SASL

The Simple Authentication and Security Layer (SASL) was introduced in RFC 2222 in hope to unify authentication mechanisms used in Internet message protocols and other application-level protocols. LOGIN, and subsequently PLAIN, were introduced. Both provide the authentication identifier and password in the clear. (PLAIN also provides for authorization identifier for identity assumption purposes.)

As LDAP already had SImple authentication with DN and password, it had little use for PLAIN. Even today, PLAIN is not generally used in LDAP but is heavily used in other application level protocols.

CRAM-MD5

CRAM-MD5 stands for Challenge/Response Authentication Mechanism - MD5. It's a password-based authentication mechanism that utilizes cryptographic hash algorithm to protect the password during the authentication exchange against eavesdropping. CRAM-MD5 however does not provide for client authentication of the server, nor does it provide data integrity/confidential protections.

While it was, and still is, heavily used in internet messaging protocols, it is not generally used in LDAP. This is likely due to the use of LDAP over SSL to protect not only passwords during authentication exchanges, but to protect the entire application data stream.

DIGEST-MD5

DIGEST-MD5 is a enhanced challenge-response authentication mechanism designed to be compatible with the HTTP Digest authentication mechanism. It was designed to have a number of improvements over CRAM-MD5, including mutual authentication and a security layer providing data integrity and confidential protections. DIGEST-MD5 also provided for hashed password storage.

However, mutual authentication facilities were not generally well implemented and generally not used. Additionally, the security layer facilities were also not well implemented and generally not used, and were prone to downgrade attacks. The stored hashed password was a password equivalent.

Hence, in practice, the mechanism only has little to no benefit over CRAM-MD5.

Hashed Password Storage

Hashed password storage was introduced to LDAP by RFC 2307 as an experimental extension to the userPassword. This extension has become the de-facto standard for hashed password storage. The hash schemes used initially were Unix crypt(3) and MD5. Subsequently, salted hash schemes were introduced, as well as hashes utilizing other hash algorithms, such as SHA-1 and SHA-2.

The use of hashed password storage requires use of mechanisms which are compatible with the hash algorithm used. Mechanisms which provide the actual password to the server, such as LDAP Simple authentication with DN and Password, are compatible with any hash algorithm.

Mechanisms which exchange values derived from the actual password are only compatible with a limited set of hash storage algorithms, often defining a mechanism-

specific algorithm to be used. Where multiple such mechanism are desired, clear text storage is often required.

Because of this, deployments which choose hashed password storage are generally limited to one stronger mechanism in addition to clear text mechanisms.

Formalization of SASL and TLS use in LDAP

Use of SASL and TLS in LDAP was standardized in RFC 2829 and RFC 2830. DIGEST-MD5 became the mandatory-to-implement authentication mechanism. The StartTLS operation became the standard mechanism for starting TLS, the successor of SSL.

Implementors and users were strongly encouraged to use TLS to protect Simple Bind with password, providing both client authentication of the server and password confidentiality.

LDAPv3 Revised

LDAPv3 was revised by RFC 4510. In this revision, the mandatory-to-implement authentication mechanism was changed to LDAP Simple Bind with DN and Password protected by TLS initiated with Start TLS. This was due to interoperability problems between DIGEST-MD5, lack of complete implementation of DIGEST-MD5, and a number of certain security concerns with DIGEST-MD5 even if were to be properly implemented.

The Current Situation

Simple Bind with DN + Password over TLS (startTLS or ldaps://) is ubiquitous. Though many clients do well implement TLS server authentication, this authentication is often disabled or not well used by the user. While some users may deal well with server authentication warnings presented by their client, most users simply "click through" the warnings. The server has no assurance that the user and its client well performed server authentication.

Use of hashed password storage is common (often required), commonly newer RFC 2307 schemes such as Salted SHA1 or even Salted SHA2.

This leads to the following problems:

- Users often send actual passwords without authenticating the server.

- User has to trust server with its actual password.
- Server can impersonate user elsewhere.
- Server has to assume client properly implements server authentication and it’s properly used.

While it common that TLS be used with server certificates, the "click through" problem makes this use effectively little better than anonymous TLS, especially from the server's perspective.

Enter SCRAM

SCRAM is short for Salted Challenge Response Authentication Mechanism. It was introduced by RFC 5802 as both a SASL and GSS-API mechanism. The authentication mechanism utilizes a cryptographic hash algorithm to provide mutual authentication. That is, the mechanism proves to the server that the user knows a secret derived from the user's password and proves to the client that the server knows a secret derived from the user's password. Actual passwords are not sent.

SCRAM is designed to allow a mechanism-specific password hash to be stored. The value is constructed such that the value cannot be used to impersonate the user.

The mechanism itself does not provide a security layer. It's intended to be used with TLS.

SCRAM-SHA-1

The basic SCRAM mechanism, SCRAM-SHA-1, TLS use is not independent of the mechanism. Hence it's use will suffer from the "click through" problems we've seen past mechanisms use of TLS, which opens LDAP sessions to hijack attack.

SCRAM-SHA-1-PLUS

The "plus" SCRAM mechanism, SCRAM-SHA-1-PLUS, add "channel bindings" to TLS.

Channel bindings to TLS prove that the TLS end-points are the same as the SCRAM end-points. That is, channel bindings prove to the client that the server authenticated by SCRAM is in control of the TLS server end-point and proves to the server that the client authenticated by SCRAM is in control of the TLS client end-point.

This effectively addresses the "click through" problem.

Unfortunately, until SCRAM-SHA-1-PLUS is ubiquitous, clients will need to continue supporting Simple Bind with DN and Password over TLS. There is significant risk of downgrade attack from SCRAM-SHA-1-PLUS to a lessor mechanism prone to hijack attack.

Conclusions

SCRAM, specifically SCRAM-SHA-1-PLUS, is a better password-based mechanism than the pre-existing mechanisms used in LDAP and commonly in other Internet application-level protocols.

The author strongly encourages developers to fully implement SCRAM-SHA-1-PLUS. Client developers should take steps to mitigate downgrade attacks. In particular, client developers should avoid changing the mechanism used without warning user that the previously available mechanism is no longer available.

The author also encourages deployers of directory services and other application services to make use of SCRAM-SHA-1-PLUS for password authentication and deprecate use of older mechanisms, including clear text mechanisms, whenever possible.

Additional Reading:

- "SCRAM: A New Protocol for Password Authentication", Isode Limited, <http://www.isode.com/whitepapers/scram.html>
- "GNU Network Security Labyrinth", Simon Josefsson, <http://josefsson.org/talks/gnu-network-security-labyrinth.pdf>.
- "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", Chris Newman, et. al., RFC 5802.