AFT Working Group
INTERNET DRAFT
Category: Standards Track
Title: draft-ietf-aft-socks-eap-00.txt
Date: March 1998

Pat Calhoun
Sun Microsystems, Inc.
Jeff Haag
3Com Corporation
Glen Zorn
Microsoft Corporation

EAP Authentication for SOCKS Version 5

Status of this Memo

Abstract

This document defines how EAP [5] (Extensible Authentication Protocol) can be used with the SOCKS V5 protocol. EAP was defined to allow any authentication mechanism to be used without any modification to the authentication protocol (i.e. CHAP, OTP).

Table of Contents

1.0 Introduction

EAP defines a set of messages which are used for the authentication protocol negotiation as well as a set of messages which are used to transport the authentication protocol payloads from a client to a server.

The RADIUS Working Group currently has a proposal [6] to encapsulate EAP messages within a RADIUS message to allow a RADIUS server to authenticate EAP messages. This powerful combination of EAP and RADIUS allows a dial-up NAS supporting this extension to seamlessly support any EAP defined authentication mechanism using a single server.

Today's firewalls typically support one or more proprietary token card vendor's authentication protocol. This is done by integrating the vendor's client software within the firewall to ensure that it can communicate with the token card vendor's authentication server.

This document proposes a mechanism which permits EAP to be used within the SOCKS V5 protocol for authentication firewall traversal. This powerful combination of SOCKS and EAP would allow a firewall to take advantage of all of the authentication extensions defined within EAP as well as to make use of the existing authentication servers.

The SOCKS protocol between a client and a firewall is over the TCP protocol. However, the protocol between the firewall and the RADIUS Server is UDP based. Care must be taken to ensure that the proper RADIUS retransmission scheme is used by the firewall.

1.1 Definitions

In this document, several words are used to signify the requirements of the specification. These words are often capitalized.

MUST This word, or the adjective "required", means that the definition is an absolute requirement of the specification.

MUST NOT This phrase means that the definition is an absolute prohibition of the specification.

SHOULD This word, or the adjective "recommended", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications must be understood and carefully weighed before choosing a different course.

MAY This word, or the adjective "optional", means that this item is one of an allowed set of alternatives. An implementation which does not include this option MUST be prepared to interoperate with another implementation which does include the option.

1.2 Terminology

This document uses the following term:

displayable message

This is interpreted to be a human readable string of characters, and MUST NOT affect operation of the protocol. The message encoding MUST follow the UTF-8 transformation format [5].

2.0 Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is a general protocol for user authentication which supports multiple authentication mechanisms. EAP does not select a specific authentication mechanism at SOCKS Authentication Method negotiation but rather postpones this until the method-dependent sub-negotiation. This allows the SOCKS server to request more information before determining the specific authentication mechanism. This also permits the use of a "back-end" server which actually implements the various mechanisms while the SOCKS server merely passes through the authentication exchanges.

After the SOCKS Authentication Method has been negotiated, the SOCKS

server or RADIUS Server sends one or more Requests to authenticate the

SOCKS Client. The Request has a TYPE field to indicate what is being requested. Examples of Request types include Identity, MD5-Challenge, One-Time Password, Generic Token Cards, etc [5]. The MD5-Challenge type corresponds closely to the CHAP authentication protocol. Typically, the SOCKS or RADIUS Server will send an initial Identity Request followed by one or more Requests for authentication information.

Additional EAP documents also propose other authentication types (i.e. TLS, RSA, etc) [3][4].

The SOCKS client sends a Response packet in reply to each Request. As with the Request packet, the Response packet contains a TYPE field which corresponds to the TYPE field of the Request.

The SOCKS or RADIUS Server ends the authentication phase with a Success or Failure packet.

Advantages

The EAP Protocol allows a firewall to support all of the defined EAP authentication extensions without having to pre-negotiate a specific one during the SOCKS V5 authentication method phase. This allows the authentication server to determine the protocol based on the user's identity.

SOCKS V5 Servers (e.g. firewall) do not need to support all EAP authentication extensions as it can act as a pass-through agent for a "back-end" server. The device only needs to look for the success/failure code to terminate the authentication phase/TCP session.

Disadvantages

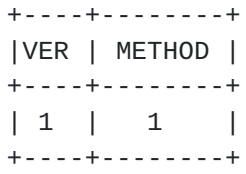
EAP does require the addition of a new authentication type to SOCKS and thus SOCKS implementations will need to be modified to use it.

2.1 SOCKS EAP Negotiation

Once the TCP session is established between a client and a SOCKS server, the client sends a version identifier/method selection message in the following format:

VER	NMETHODS	METHODS
1	1	1 to 255

The server selects from one of the methods given in METHODS, and sends a METHOD selection message:



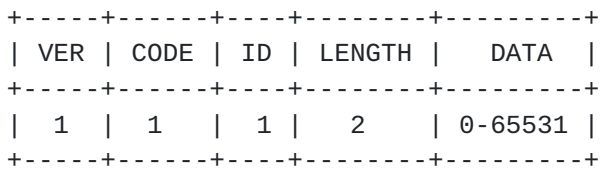
VER	METHOD
1	1

The VER field is set to X'05' for this version of the protocol. The NMETHODS field contains the number of method identifier octets that appear in the METHODS field. The following value is used in the METHODS field to indicate that EAP is to be used:

- (TDB) Extensible Authentication Protocol

2.2 EAP Packet Format

Exactly one EAP packet is transmitted at any time. A summary of the EAP packet format is shown below. The fields are transmitted from left to right.



VER	CODE	ID	LENGTH	DATA
1	1	1	2	0-65531

VER contains the current version of the subnegotiation, which is X'01'. CODE contains the EAP packet type as shown below. The ID field is one octet and aids in matching responses with requests. The LENGTH field is two octets and indicates the length of the EAP packet including the CODE, ID, LENGTH and DATA fields. The DATA field consists of one or more octets whose format is determined by the CODE field.

Code Definitions

The following EAP message types apply to all messages:

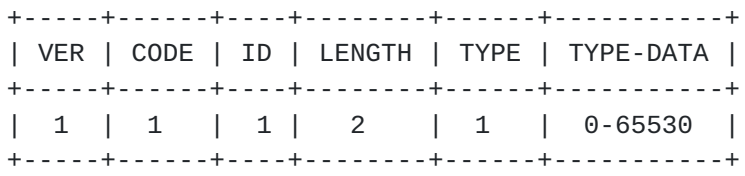
ATT	Label	Meaning
X'01'	EAP-Request	Request from Server to Client
X'02'	EAP-Response	Response from Client to Server
X'03'	EAP-Success	Server responds with Success
X'04'	EAP-Failure	Server responds with Failure

2.2.1 Request and Response

The Request packet is sent by the SOCKS/RADIUS Server to the SOCKS Client. Each Request has a TYPE field which serves to indicate what is being requested. The SOCKS/RADIUS Server MUST transmit an EAP packet with the CODE field set to X'01' (EAP-Request). The contents of the DATA field is dependent on the Request type. The peer MUST send a Response packet in reply to a Request packet. Responses MUST only be sent in reply to a received Request. The ID field of the Response MUST match that of the Request.

SOCKS/RADIUS Server Implementation Note: Because the authentication process will often involve user input, some care must be taken when deciding upon authentication timeouts.

A summary of the Request and Response packet format is shown below. The fields are transmitted from left to right.



VER	CODE	ID	LENGTH	TYPE	TYPE-DATA
1	1	1	2	1	0-65530

VER contains the current version of the subnegotiation, which is X'01'. CODE contains contains either X'01' (EAP-Request) or X'02' (EAP-Response). The ID field is one octet and aids in matching responses with requests. The LENGTH field is two octets and indicates the length of the EAP packet including the CODE, ID, LENGTH, TYPE and TYPE-DATA fields.

The TYPE field is one octet and indicates the type of Request or Response. Only one type MUST be specified per EAP Request or Response. Normally, the TYPE field of the Response will be the same as the TYPE field of the Request. However, there is also a Nak Response type for indicating that a Request type is unacceptable to the peer. When sending a Nak in response to a Request, the peer MAY indicate an alternative desired authentication type which it supports. An initial specification of types follows in a later section of this document.

The TYPE-DATA field varies with the type of Request and the associated Response.

2.2.2 Success and Failure

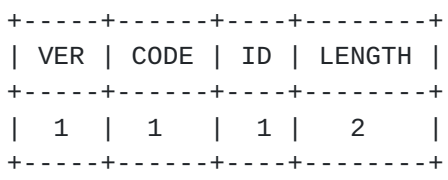
The Success packet is sent by the SOCKS/RADIUS Server to the SOCKS Client to acknowledge successful authentication. The SOCKS/RADIUS

Server MUST transmit an EAP packet with the Code field set to X'03'

(Success).

If the SOCKS/RADIUS Server cannot authenticate the SOCKS Client (unacceptable Responses to one or more Requests) then the implementation MUST transmit an EAP packet with the Code field set to X'04' (Failure). A SOCKS/RADIUS Server MAY wish to issue multiple Requests before sending a Failure response in order to allow for human typing mistakes.

A summary of the Success and Failure packet format is shown below. The fields are transmitted from left to right.



VER contains the current version of the subnegotiation, which is X'01'. CODE contains contains either X'03' (EAP-Success) or X'04' (EAP-Failure). The ID field is one octet and aids in matching responses with requests. The ID field MUST match the ID field of the Response packet that it is sent in response to. The LENGTH field is two octets and indicates the length of the EAP packet including the CODE, ID and LENGTH fields.

3.0 Initial EAP Request/Response Types

This section will provide an overview of the initial set of EAP Types as defined in [5] in order to provide the reader with a basic understanding of the EAP messages. More types are defined in follow-on documents.

The TYPE field is one octet and identifies the structure of an EAP Request or Response packet. The first 3 types are considered special types. The remaining types define authentication exchanges.

The following basic EAP types are defined in [5]:

X'01'	Identity
X'02'	Notification
X'03'	Nak (Response only)
X'04'	MD5-Challenge
X'05'	One-Time Password (OTP) (RFC 1938)
X'06'	Generic Token Card

The following EAP type MUST be supported by all SOCKS/EAP implementations and is specified in [x]:

TDB HMAC-MD5-Challenge

3.1 Identity

The Identity type is used to query the identity of the peer. Generally, the authenticator will issue this as the initial Request. An optional displayable message MAY be included to prompt the peer in the case where there is expectation of interaction with a user. A Response MUST be sent to this Request with a type of X'01' (Identity).

Implementation Note: The SOCKS Client MAY obtain the Identity via user input. It is suggested that the SOCKS/RADIUS Server retry the Identity Request in the case of an invalid Identity or authentication failure to allow for potential typos on the part of the user.

It is suggested that the Identity Request be retried a minimum of 3 times before terminating the authentication phase with a Failure reply. The Notification Request MAY be used to indicate an invalid authentication attempt prior to transmitting a new Identity Request (optionally, the failure MAY be indicated within the message of the new Identity Request itself).

3.2 Notification

The Notification type is optionally used to convey a displayable message from the SOCKS/RADIUS Server to the SOCKS Client. The client SHOULD display this message to the user or log it if it cannot be displayed. It is intended to provide an acknowledged notification of some imperative nature. Examples include a password with an expiration time that is about to expire, an OTP sequence integer which is nearing 0, an authentication failure warning, etc. In most circumstances, notification should not be required.

3.3 Nak

The Nak type is valid only in Response messages. It is sent in reply to a Request where the desired authentication Type is unacceptable. Authentication types are numbered 4 and above. The Response contains the authentication type desired by the peer.

3.4 MD5-Challenge

The MD5-Challenge type is analogous to the PPP CHAP protocol [1] (with MD5 as the specified algorithm). The PPP Challenge Handshake Authentication Protocol RFC [1] should be referred to for further

implementation specifics. The Request contains a "challenge" message

to the peer. A Response MUST be sent in reply to the Request. The Response MAY be either of type X'04' (MD5-Challenge) or type X'03' (Nak). The Nak reply indicates the peer's desired authentication mechanism type. All EAP implementations MUST support the MD5-Challenge mechanism.

3.5 One-Time Password (OTP)

The One-Time Password system is defined in "A One-Time Password System" [2]. The Request contains a displayable message containing an OTP challenge. A Response MUST be sent in reply to the Request. The Response MUST be of Type X'05' (OTP) or Type X'03' (Nak). The Nak reply indicates the peer's desired authentication mechanism Type.

3.6 Generic Token Card

The Generic Token Card type is defined for use with various Token Card implementations which require user input. The Request contains an ASCII text message and the Reply contains the Token Card information necessary for authentication. Typically, this would be information read by a user from the Token card device and entered as ASCII text.

3.7 HMAC-MD5-Challenge

The HMAC-MD5-Challenge MUST be supported by all SOCKS implementation and MUST be the default authentication mechanism. The protocol is defined in xxx [x]. A Response MUST be sent in reply to the Request. The Response MAY be either of type TDB (HMAC-MD5-Challenge) or type X'03' (Nak). The Nak reply indicates the peer's desired authentication mechanism type.

4.0 Security Considerations

Security issues are the primary topic of this RFC.

The interaction of the authentication protocols within SOCKS are highly implementation dependent.

There is no requirement that authentication be full duplex or that the same protocol be used in both directions. It is perfectly acceptable for different protocols to be used in each direction. This will, of course, depend on the specific protocols negotiated.

In practice, within or associated with each firewall, it is not anticipated that a particular named user would be authenticated by multiple methods. This would make the user vulnerable to attacks

which negotiate the least secure method from among a set (such as OTP

rather than RSA). Instead, for each named user there should be an indication of exactly one method used to authenticate that user name. If a user needs to make use of different authentication methods under different circumstances, then distinct identities SHOULD be employed, each of which identifies exactly one authentication method.

5.0 Acknowledgements

Much of this text was taken from [5] and the authors wish to recognize L.J. Blunk and J.R. Vollbrecht for their original draft.

6.0 References

[1] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[2] N. Haller, and C. Metz, "A One-Time Password System", RFC 1938, May 1996.

[3] Bernard Aboba, Dan Simon, "PPP EAP TLS Authentication Protocol", Work in Progress, October 1997.

[4] William Whelan, "PPP EAP RSA Public Key Authentication Protocol", Work in Progress, "February 1997.

[5] L.J. Blunk, J. R. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", Work in Progress, December 1997.

[6] P. Calhoun, A. Rubens, B. Aboba, "Extensible Authentication Protocol Support in RADIUS", Work in Progress, May 1997.

[7] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS Protocol V5", RFC 1928, April 1996.

7.0 Authors' Addresses

Questions about this memo can be directed to:

Pat R. Calhoun
Technology Development
Sun Microsystems, Inc.
15 Network Circle
Menlo Park, California, 94025
USA

Phone: +1 847 548-9587

Fax: +1 650 786-6445

E-mail: pcalhoun@toast.net

Jeff Haag
3Com Corporation
1800 W. Central Ave
Mount Prospect, Il 60031

Phone: +1 919 676-9971
E-Mail: jhaag@usr.com

Glen Zorn
Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052

Phone: +1 425 703-1559
E-Mail: glennz@microsoft.com