Workgroup:
Operations and Management Area Working Group
Updates: RFC8907 (if approved)
Published: 2 June 2022
Intended Status: Informational
Expires: 4 December 2022
Authors: T. Dahm D. Gash A. Ota J. Heasley
Cisco Systems, Inc. NTT

TACACS+ Security and SSH Public Keys

Abstract

The TACACS+ Protocol [RFC8907] provides device administration for routers, network access servers and other networked computing devices via one or more centralized servers. This document, a companion to the TACACS+ protocol [RFC8907], adds new packet formats to improve security and function and support for SSH [RFC4716] public keys.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP14] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The TACACS+ Protocol [RFC8907] provides device administration for routers, network access servers and other networked computing devices via one or more centralized servers. The protocol provides authentication, authorization and accounting services for TACACS+ clients.

Authors of [RFC8907] and network operators have expressed interest in deprecating a few antiquated protocol features, addressing a limitation of authentication policy, and adding support for SSH public keys. These are addressed herein.

To improve security and functionality of applying policy to the authentication process in a TACACS+ Session, new authentication packet formats are introduced that are uniform to authorization and accounting. For SSH authentication using public keys, highly desired by the operator community, this document introduces a method to support sending public keys to a TACACS+ client, allowing centralized management.

2. Technical Definitions

The Technical Definitions section of the TACACS+ Protocol [RFC8907] is fully applicable here and will not be repeated, though may be augmented. The following terms are also used in this document.

2.1. AVP

An Attribute-Value Pair or AVP is another name a TACACS+ argument as defined in [RFC8907] Sections 6.1 and 8.

2.2. Empty Value

An empty or zero-length value of an AVP as defined in [RFC8907] Sections 8.1.

2.3. Peer

This refers to a TACACS+ Server or Client.

3. TACACS+ Extended Authentication Packet Types

Versions 1 and 2 of the TACACS+ Protocol, as defined in [RFC8907], specify the TACACS+ Authentication Packets for START, REPLY and CONTINUE which support the credential validation use case but does not accommodate any further augmentation which may be used to give context to the request.

Further, advanced use cases (such as SSH key distribution) would otherwise rely on embedding structured information into the single data fields, thus obfuscating the content of the protocol, can instead augment the arguments.

To support these use cases, and allow clients to add environment information to the request, the Extended Authentication Packets brings the Authentication phase of the protocol inline with the Authorization and Accounting Phase by incorporating extensible argument s.

The server should expect Extended Authentication Packet Bodies if the minor version in the Packet Header is: 0x2

3.1. The Extended Authentication START Packet Body

```
 1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8
+----------------+----------------+----------------+----------------+
|    action      |    priv_lvl    |  authen_type   | authen_service |
+----------------+----------------+----------------+----------------+
|    user_len    |    port_len    |  rem_addr_len  |    data_len    |
+----------------+----------------+----------------+----------------+
|    arg_cnt                                                        |
+----------------+----------------+----------------+----------------+
|    arg_1_len                                                      |
+----------------+----------------+----------------+----------------+
|      ...                                                          |
+----------------+----------------+----------------+----------------+
|    arg_N_len                                                      |
+----------------+----------------+----------------+----------------+
|    user ...
+----------------+----------------+----------------+----------------+
|    port ...
+----------------+----------------+----------------+----------------+
|    rem_addr ...
+----------------+----------------+----------------+----------------+
|    data...
+----------------+----------------+----------------+----------------+
|    arg_1 ...
+----------------+----------------+----------------+----------------+
|    arg_2 ...
+----------------+----------------+----------------+----------------+
|    ...
+----------------+----------------+----------------+----------------+
|    arg_N ...
+----------------+----------------+----------------+----------------+
```

Figure 1

The action, priv_level, authen_type, authen_service, user_len, port_len, rem_addr_len, data_len, user, port, rem_addr and data fields are used exactly as defined in the Authentication START Packet Body in [RFC8907].

The following fields contain the arguments that may be used to extend the authentication process. These are common to the Extended Authentication START, Extended Authentication REPLY, and Extended Authentication CONTINUE packet bodies; these fields represent the sole update from the previous START, REPLY and CONTINUE packet bodies.

The new fields are as follows:

arg_cnt

This represents the number of arguments in the packet.

arg_1_len ... arg_N_len, arg_1 ... arg_N

Each argument is encoded in the packet as a single arg field (arg_1 ... arg_N) with a corresponding length field that indicates the length of each argument in bytes.

The arguments are argument-value pairs. The argument and the value are in a single string and are separated by either a "=" (0X3D) or a "*" (0X2A). The equals sign indicates a mandatory argument. The asterisk indicates an optional one. For the rules regarding optional and mandatory arguments, refer to [RFC8907]

Multiple arguments with the same name are permitted within a packet, a common example is cmd-arg. The handling of repeated arguments is specific to the semantics of the argument and so are documented with that argument. Order is significant when processing arguments.

The addition of arguments to the authentication packets is intended to permit the flexibility for the TACACS+ authentication phase that has been available previously for authorization and accounting. These fields are intended to be used as needed in deployment, they are used in this document in the enhancements for SSH (Section 4):

origin_client

Contains the IP-Address of the originating TACACS+ client. This is text encoded in line with the rest of the TACACS+ protocol, and may be IPv4 or IPv6. This argument is optional and can be included in all TACACS+ protocol START packets, Authentication, Authorization, and Accounting. IPv4 addresses are specified as quad-octet numeric values separated by dots ('.'). IPv6 address text representation is defined in [RFC5952].

3.2. The Extension Authentication REPLY Packet Body

The TACACS+ server sends only one type of extended authentication packet to the client.

```
 1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8
+----------------+----------------+----------------+----------------+
|     status     |      flags     |        server_msg_len           |
+----------------+----------------+----------------+----------------+
|           data_len              |
+----------------+----------------+----------------+----------------+
|    arg_cnt                                                        |
+----------------+----------------+----------------+----------------+
|    arg_1_len                                                      |
+----------------+----------------+----------------+----------------+
|      ...                                                          |
+----------------+----------------+----------------+----------------+
|    arg_N_len                                                      |
+----------------+----------------+----------------+----------------+
|    data ...
+----------------+----------------+----------------+----------------+
|    server_msg ...
+----------------+----------------+----------------+----------------+
|    arg_1 ...
+----------------+----------------+----------------+----------------+
|    arg_2 ...
+----------------+----------------+----------------+----------------+
|    ...
+----------------+----------------+----------------+----------------+
|    arg_N ...
+----------------+----------------+----------------+----------------+
```

Figure 2

The status, flags, server_msg_len, data_len, server_msg, and data fields are used exactly as defined in the Authentication REPLY Packet Body in [RFC8907]].

The new arg_cnt, arg_1 ... arg_N, and arg_1_len .... arg_N_len fields are used as defined in The Extended Authentication START Packet Body (Section 3.1).

3.3. The Extended Authentication CONTINUE Packet Body

This packet is sent from the client to the server following the receipt of an Extended REPLY packet.

```
 1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8  1 2 3 4 5 6 7 8
+----------------+----------------+----------------+----------------+
|             status              |           user_msg len          |
+----------------+----------------+----------------+----------------+
|            data_len             |
+----------------+----------------+----------------+----------------+
|    arg_cnt                                                        |
+----------------+----------------+----------------+----------------+
|    arg_1_len                                                      |
+----------------+----------------+----------------+----------------+
|      ...                                                          |
+----------------+----------------+----------------+----------------+
|    arg_N_len                                                      |
+----------------+----------------+----------------+----------------+
|  user_msg ...
+----------------+----------------+----------------+----------------+
|    data ...
+----------------+----------------+----------------+----------------+
|    arg_1 ...
+----------------+----------------+----------------+----------------+
|    arg_2 ...
+----------------+----------------+----------------+----------------+
|    ...
+----------------+----------------+----------------+----------------+
|    arg_N ...
+----------------+----------------+----------------+----------------+
```

Figure 3

The user_msg len, data_len, user_msg, and data fields are used exactly as defined in the Authentication REPLY Packet Body in [RFC8907]. However, the status field replaces the flags field and has the following enumeration:

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_NONE := 00

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_PASS := 01

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_FAIL := 02

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_FRAGMENT := 03

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_ERROR := 04

*TAC_PLUS_AUTHEN_CONTINUE_STATUS_ABORT := 05

TAC_PLUS_AUTHEN_CONTINUE_STATUS_NONE or TAC_PLUS_AUTHEN_CONTINUE_STATUS_ABORT MUST be used when the Extended Authentication Packets are used for the continuation of authentication flows documented in [RFC8907].

The client may prematurely terminate a session by setting the TAC_PLUS_AUTHEN_CONTINUE_STATUS_ABORT or TAC_PLUS_AUTHEN_CONTINUE_STATUS_ERROR status in the CONTINUE message. The remainder are detailed in SSH (Section 4).

The new arg_cnt, arg_1 ... arg_N, and arg_1_len .... arg_N_len fields are used as defined in The Extended Authentication START Packet Body (Section 3.1).

4. SSH

Most network equipment now support SSH [RFC4251] for Command Line Interface (CLI) and NETCONF [RFC6242]. Operators SHOULD use SSH public keys for authentication. Some devices support public keys in native configuration, but there is desire to centrally manage keys and SSH subsystem authorization.

4.1. New Enumerated TACACS+ Protocol Values and well-known AVPs

The following new enumerated TACACS+ protocol values and well-known AVPs are needed to support SSH in the subsequent sections. These new values augment those in [RFC8907] Sections 5.1 - 5.3, 6.1, and 8.2 as follows:

TAC_PLUS_AUTHEN_TYPE_SSHPK := 0x07
Extended Authentication START Packet authen_type for SSH pubkeys.

TAC_PLUS_AUTHEN_STATUS_GETSSHPKTYPE := 0x22
Extended Authentication REPLY Packet status to solicit SSH pubkey type.

TAC_PLUS_AUTHEN_STATUS_SSHPK := 0x23
Extended Authentication REPLY Packet status to provide SSH pubkeys.

TAC_PLUS_REPLY_FLAG_FRAGMENT := 0x02
Extended Authentication REPLY Packet flag indicating the REPLY is incomplete.

TAC_PLUS_AUTHEN_CONTINUE_STATUS_PASS := 0x01
Extended Authentication CONTINUE Packet flag indicating authentication success.

TAC_PLUS_AUTHEN_CONTINUE_STATUS_FAIL := 0x08
Extended Authentication CONTINUE Packet flag indicating authentication failure.

TAC_PLUS_AUTHEN_CONTINUE_STATUS_FRAGMENT := 0x03
Extended Authentication CONTINUE Packet flag requesting the next REPLY packet of an incomplete REPLY.

TAC_PLUS_AUTHEN_CONTINUE_STATUS_ERROR := 0x04
Extended Authentication CONTINUE Packet flag indicating authentication error.

AVP ssh_pubkey_type (String)
Attribute to carry SSH public key type names.

AVP ssh_pubkey (String)
Attribute to carry SSH public keys.

TAC_PLUS_AUTHEN_METH_SSHPK := 0x21
Authorization REQUEST Packet authen_method for SSH pubkey authentication.

AVP ssh_subsystem (String)
Attribute to carry SSH subsystem name for authorization

4.2. SSH Public Key Support

To support central management of SSH public keys via TACACS+, the Authentication sequence of [RFC8907] Section 5.4 is extended using Extended Authentication Packet (Section 3) sequences to deliver SSH public keys to devices for local verification.

Besides new header values and flags and AVPs for Extended Authentication Packets, the SSH public key authentication process differs from other TACACS+ authentication types in that there may be more Authentication Reply and Authentication Continue Packets pairs than previously.

The process follows:

1. The client begins an authentication session with an Extended Authentication START Packet. The START packet MUST include a non-zero-length username and the server MUST send an Authentication REPLY Packet with status TAC_PLUS_AUTHEN_STATUS_ERROR, if the client fails to do so.

   The client MAY include one or more instances of the ssh_pubkey_type AVP, indicating the SSH public key types that it wants. The set of permissible values for this AVP are the SSH public algorithm names defined in the IANA SSH Protocol Parameters Registry [SSHALGS], which are case-sensitive as specified and otherwise constrained by [RFC4250] Section 4.6.1. Multiple values MUST be separated by a comma, therefore multiple ssh_pubkey_type AVPs MUST include commas for separation when the Peer concatenates them and the Peer MUST be prepared to ignore a leading or trailing comma in the concatenated value. The server MUST NOT reply with status TAC_PLUS_AUTHEN_STATUS_ERROR if it receives an algorithm name that it does not recognize. If the client marks a ssh_pubkey_type AVP as mandatory, the server MUST reply with at least one key of that type for the given user or reply with status TAC_PLUS_AUTHEN_STATUS_SSHNOPK with the relevant ssh_pubkey_type AVP.

The client MAY send an Empty Value for the algorithm name to request all types available for the given user.

The process ends and the client MUST start a new authentication session if it receives status SSHNOPK or ERROR.

2. If a ssh_pubkey_type AVP was not provided in the START packet, the server replies with the status code TAC_PLUS_AUTHEN_STATUS_GETSSHPKTYPE. The client MUST send a CONTINUE packet with one or more ssh_pubkey_type AVPs, else the server sends a REPLY packet with status TAC_PLUS_AUTHEN_STATUS_ERROR.

3. If the server has none of the requested ssh_pubkey_type(s) or any of the mandatory ssh_pubkey_types for the user or no pubkeys at all, the server MUST send a REPLY packet with status TAC_PLUS_AUTHEN_STATUS_SSHNOPK with the ssh_pubkey_type AVP(s) that it received.

   The process ends and the client MUST start a new authentication session if it receives status SSHNOPK or ERROR.

4. The server sends REPLY packets with status TAC_PLUS_AUTHEN_STATUS_SSHPK and includes one or more ssh_pubkey optional AVPs, each containing one or more keys. The ssh_pubkey AVPs are formatted according to the rules of SSH Public Key File Format [RFC4716]. As such, the client MUST be prepared to accept keys with Key File Markers. To address concatenation of multiple ssh_pubkey AVPs or multiple keys in a single AVP, the server MUST terminate each key file End Marker with a Line Termination sequence as specified in RFC4716 Section 3.1.

   Since it is possible to have more ssh_pubkey AVPs than fit in a REPLY packet, the server SHOULD set the REPLY packet flag TAC_PLUS_REPLY_FLAG_FRAGMENT if two or more packets are required, indicating that the client SHOULD request the remainder.

   An AVP SHALL NOT span multiple fragments; each must be contained entirely in the fragment in which it begins.

5. If the TAC_PLUS_REPLY_FLAG_FRAGMENT flag is set, the client MAY reply with the same CONTINUE packet as before with the

TAC_PLUS_AUTHEN_CONTINUE_STATUS_FRAGMENT flag set. The server replies with the next REPLY fragment as before, clearing the TAC_PLUS_REPLY_FLAG_FRAGMENT flag of the last REPLY fragment. This repeats until the last REPLY fragment is received, the client aborts the authentication process, or an error occurs. The client MUST NOT set TAC_PLUS_AUTHEN_CONTINUE_STATUS_FRAGMENT if the REPLY packet did not have the TAC_PLUS_REPLY_FLAG_FRAGMENT flag set and the server MUST reply with TAC_PLUS_AUTHEN_STATUS_ERROR if it does so.

6. Once the client has all of the pubkeys, it performs the ssh pubkey authentication with its ssh client. The client MUST then reply to the server with the status of that authentication by sending a CONTINUE packet with one of the following new or existing CONTINUE flags: TAC_PLUS_CONTINUE_FLAG_ABORT, TAC_PLUS_AUTHEN_CONTINUE_STATUS_PASS, TAC_PLUS_AUTHEN_CONTINUE_STATUS_FAIL, or TAC_PLUS_AUTHEN_CONTINUE_STATUS_ERROR.

7. The client MUST give the server the final consent, by waiting for a REPLY packet with one of the status: TAC_PLUS_AUTHEN_STATUS_PASS, TAC_PLUS_AUTHEN_STATUS_FAIL, or TAC_PLUS_AUTHEN_STATUS_ERROR, thus ending the authentication session.

4.3. SSH Authorization and Accounting

To support central management of SSH and SSH subsystem authorization and accounting via TACACS+, this document adds a new authen_method to RFC8907 Section 6.1 Authorization REQUEST [RFC8907] and a well-known AVP to Section 8.2 Authorization Arguments [RFC8907].

The new authen_method TAC_PLUS_AUTHEN_METH_SSHPUBKEY indicates that the user was authenticated with a SSH public key.

The well-known ssh_subsystem AVP defines the SSH subsystem for which the authorization is requested and MUST be present any time the authorization is for a SSH connection.

The set of permissible values for this AVP are the SSH Subsystem Names defined in the IANA SSH Connection Protocol Subsystem Names Registry [SSHSUBSYS], which are case-sensitive as specified and otherwise constrained by [RFC4250] Section 4.6.1. The client MAY send an Empty Value for the subsystem name to indicate no subsystem, also known as a shell or CLI. The server MUST NOT reply with status TAC_PLUS_AUTHOR_STATUS_ERROR if it receives a subsystem name whose syntax is valid but whose value is not recognized. Subsystems might need additional data for authorization or accounting that will be particular to that subsystem and are therefore out of scope for this document.

These new authen_methods and AVPs apply equally to accounting.

5. Protocol Deprecations

This section deprecates features from the TACACS+ Protocol.

MS-CHAPv1: has been replaced by MS-CHAPv2 in most deployments, the intent of this deprecation is to complete the transition. MD4 is still required to support MS-CHAPv2 so cannot be deprecated at this point It should be noted that the use of MD4 is purely to allow compatible MS-CHAPv2 operation and not for security; the TLS transport is intended to provide that function.

TAC_PLUS_AUTHEN_SENDAUTH: the sendauth mechanism can not be supported, as it permits the leak of sensitive information.

6. Security Considerations

6.1. SSH Public Key Caching

A Client MUST NOT cache SSH public keys received from a Server for future SSH client authentication. Doing so would deny the Server the opportunity to deny authentication for other reasons than key validity or to revoke a key. The Server has no method to revoke a key, except by not offering the key in future authentication sessions.

7. Acknowledgments

The author(s) would like to thank Russ Housley, Steven M. Bellovin, Stephen Farrell, Alan DeKok, Warren Kumari, and Tom Petch for their support, insightful review, and/or comments. [RFC5425] was also used as a basis for the approach to TLS.

8. Normative References

[BCP14] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017. <https://www.rfc-editor.org/bcp/bcp14.txt>

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, DOI 10.17487/RFC4250, January 2006, <https://www.rfc-editor.org/info/rfc4250>.

[RFC4716]
Galbraith, J. and R. Thayer, "The Secure Shell (SSH) Public Key File Format", RFC 4716, DOI 10.17487/RFC4716, November 2006, <https://www.rfc-editor.org/info/rfc4716>.

[RFC5425] Miao, F., Ed., Ma, Y., Ed., and J. Salowey, Ed., "Transport Layer Security (TLS) Transport Mapping for Syslog", RFC 5425, DOI 10.17487/RFC5425, March 2009, <https://www.rfc-editor.org/info/rfc5425>.

[RFC8907] Dahm, T., Ota, A., Medway Gash, D.C., Carrel, D., and L. Grant, "The Terminal Access Controller Access-Control System Plus (TACACS+) Protocol", RFC 8907, DOI 10.17487/RFC8907, September 2020, <https://www.rfc-editor.org/info/rfc8907>.

[SSHALGS] IANA, "Public Key Algorithm Names", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-19>.

[SSHSUBSYS] IANA, "SSH Protocol Subsystem Names", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-15>.

9. Informative References

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <https://www.rfc-editor.org/info/rfc4251>.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

Authors' Addresses

Thorsten Dahm

Email: thorsten.dahm@gmail.com

Douglas Gash
Cisco Systems, Inc.

Email: dcmgash@cisco.com

Andrej Ota

Email: andrej@ota.si

John Heasley
NTT

Email: heas@shrubbery.net