INTERNET-DRAFT
<draft-ietf-krb-wg-kerberos-sam-03.txt>
Updates: RFC 1510
July 15, 2004

Ken Hornstein
Naval Research Laboratory
Ken Renard
WareOnEarth
Clifford Newman
ISI
Glen Zorn
Cisco Systems

Integrating Single-use Authentication Mechanisms with Kerberos

0. Status Of this Memo

1. Abstract

This document defines extensions to the Kerberos protocol specifi- cation [RFC1510] which provide a method by which a variety of single-use authentication mechanisms may be supported within the protocol. The method defined specifies a standard fashion in which the preauthentication data and error data fields in Kerberos messages may be used to support single-use authentication mechanisms.

2. Terminology

To simplify the following discussion, we will define those terms which may be unfamiliar to the audience or specific to the discussion itself.

Single-use Preauthentication Data (SPD): Data sent in the padata-value field of a Kerberos V5 message proving that knowledge of

certain unique information is held by a principal. This information may or may not be identical to the single-use authentication data input to the client. For example, in the case of S/Key, the principal might input a one-time password (in any of several forms); the knowledge of this one-time password is taken to indicate knowledge of the principal's secret passphrase. Similarly, the SPD may or may not contain the provided single-use authentication data. For instance, if a given single-use authentication mechanism includes a token which generates an encryption key for a supported cryptosystem, that key could be used to encrypt portions of the SPD before transmission. As long as the verification process of the mechanism was capable of independently generating the same key, the successful decryption of the SPD would provide assurance that the originator of the message was in possession of the token, as well as whatever information the token required to generate the encryption key.

Single-use Authentication Mechanism (SAM): A system for generating and verifying authentication data which is usable only once.

Single-use Authentication Data (SAD): SAM-specific data provided by a principal as input to client software to be used in the creation of SPD.

3. Motivation and Scope

Several single-use authentication mechanisms are currently in widespread use, including hardware-based schemes from vendors such as Enigma Logic, CRYPTOCard, and Security Dynamics and software-based methods like S/Key [RFC1760]. The hardware-based schemes typically require that the authenticating user carry a small, credit-card-sized electronic device (called a token) which is used to generate unique authentication data. Some tokens require the user to enter data into the device. This input may take the form of a Personal Identification Number (PIN), a server-generated challenge string or both. Other tokens do not use a challenge-response technique, instead spontaneously generating new and unique authentication data every few seconds. These tokens are usually time-synchronized with a server. The use of one-time passwords and token cards as an authentication mechanism has steadily increased over the past few years; in addition, the Internet Architecture Board has encouraged the use of SAMs to improve Internet security [RFC1636].

The widespread acceptance of Kerberos within the Internet community has produced considerable demand for the integration of SAM technology with the authentication protocol. Several currently available implementations of Kerberos include support for some types of

token cards, but the implementations are either not interoperable, or would require the release of source code (not always an option) to make them interoperate. This memo attempts to remedy that problem by specifying a method in which SAM data may be securely transported in Kerberos V5 messages in a standard, extensible fashion. This document does not, however, attempt to precisely specify

either the generation or verification of SAM data, since this is likely to be SAM-specific; nor does it dictate the conditions under which SAM data must be included in Kerberos messages, since we consider this to be a matter of local policy.

A primary reason for using a SAM with Kerberos is to reduce the threat from common attacks on Kerberos passwords (poorly chosen passwords, password guessing, etc). If passwords are used in combination with SAM authentication data, users must still adhere to sensible password policies and safe practices regarding the selection, secrecy, and maintenance of their passwords. Depending on the specific mechanism used, the purpose of the SAD is to augment (or sometimes replace) the use of a password as a secret key.

4. Generic Approach - Two Models

As outlined above, there are essentially two types of single-use authentication mechanisms: challenge/response and time-based. In order to support challenge/response mechanisms, the Kerberos Key Distribution Center (KDC) must communicate the appropriate challenge string to the user, via the client software. Furthermore, some challenge/response mechanisms require tight synchronization between all instances of the KDC and the client. One example is S/Key and its variants. If the KDC and client do not perform the same number of message digest iterations, the protocol will fail; worse, it might be possible for an eavesdropping attacker to capture a valid S/Key passcode and replay it to a KDC replica which had an outdated iteration number. In the time-based case, no challenge is required. This naturally gives rise to two modes of client behavior, described below.

4.1 Challenge/Response Model

The client begins with an initial KRB_AS_REQ message to the KDC, possibly using existing preauthentication methods (PA-ENC-TIMESTAMP (encrypted timestamp), PA-OSF-DCE (DCE), etc.). Depending on whether preauthentication is used, the user may or may not be prompted at this time for a Kerberos password. If (for example) encrypted timestamp preauthentication is used, then the user will be prompted; on the other hand, if no preauthentication is in use the prompt for the password may be deferred (possibly forever). Note that the use of preauthentication here may allow an offline guessing attack against the Kerberos password separate from the SPD. However, if the use of a SAM is required, then the password by itself is not sufficient for authentication. (Specify character strings as UTF-8)

The KDC will determine in an implementation- and policy-dependent

fashion if the client is required to utilize a single-use authentication mechanism. For example, the implementation may use IP address screening to require principals authenticating from outside a firewall to use a SAM, while principals on the inside need not. If SAM usage is required, then the KDC will respond with a KRB_ERROR message, with the error-code field set to

KDC_ERR_PREAUTH_REQUIRED and the e-data field containing the ASN.1 structure that is a sequence of PA-DATA fields.

If the type of one of the PA-DATA fields is PA-SAM-REDIRECT, the client should re-execute the authentication protocol from the beginning, directing messages to another of the KDCs for the realm. This is done to allow some methods to require that a single KDC be used for SAM authentication when tight synchronization is needed between all replicas and the KDC database propagation code does not provide such synchronization. The corresponding padata-value will contain an encoded sequence of host addresses [RFC1510], from which the client must choose the KDC to be contacted next. The PA-SAM-REDIRECT is defined as:

```
PA-SAM-REDIRECT ::=    HostAddresses
```

Client implementations SHOULD check the addresses in the PA-SAM-REDIRECT and verify that they are a subset of the KDC addresses that they have been configured for that realm.

If none of the PA-DATA fields have a value of PA-SAM-REDIRECT, then if one of the PA-DATA fields has the type PA-SAM-CHALLENGE-2, the exchange will continue as described in section 5, below.

Note that some Kerberos implementations support an older preauthentication mechanism with the padata types PA-SAM-CHALLENGE and PA-SAM-RESPONSE. That protocol is depreciated and not defined here.

4.2 Time-based Model

For mechanisms where no challenge is required, the user (or the client software being utilized) may or may not know a priori whether SAM usage is required. If it does not know, then the initial exchange may proceed as above. If it is known that a use of a single-use authentication mechanism is required then the first exchange can be skipped and the authentication will continue as follows.

5. SAM Preauthentication

An optional SAM-CHALLENGE-2 may be sent from the KDC to the client and the client will send a SAM-RESPONSE-2 as pre-authentication data in the KRB-AS-REQ. The details of the messages follow.

5.1 SAM-CHALLENGE-2

Prior to performing preauthentication using a single-use authentication mechanism, the client must know whether a challenge is required (if the client doesn't have this information prior to its sending the first KRB_AS_REQ message, it will be informed of the requirement by the KDC, as described in section 4.1). The client

does NOT need to know the specific type of SAM in use. If a challenge is required the client will be sent the challenge by the KDC. This means that a client supporting SAMs will be able to work with new methods without modification. The challenge, as well as all other prompts mentioned herein, can be internationalized by the KDC on a per-principal basis.

If a KRB_ERROR message is received from the KDC indicating that SAM usage is required, that message will include in its e-data field a PA-DATA structure that encodes information about the SAM to be used. This includes whether a challenge is required, and if so, the challenge itself; and informational data about the type of SAM that is in use, and how to prompt for the SAD. The SAM type is informational only and does not affect the behavior of the client. The prompt is also informational and may be presented to the user by the client, or it may be safely ignored.

The ASN.1 definition for the SAM challenge is:

```
PA-SAM-CHALLENGE-2 ::= SEQUENCE {
    sam-body[0]                 PA-SAM-CHALLENGE-2-BODY,
    sam-cksum[1]                SEQUENCE (1..MAX) OF Checksum,
    ...
}

PA-SAM-CHALLENGE-2-BODY ::= SEQUENCE {
    sam-type[0]                 INTEGER (0..4294967295),
    sam-flags[1]                SAMFlags,
    sam-type-name[2]            GeneralString OPTIONAL,
    sam-track-id[3]             GeneralString OPTIONAL,
                                  -- Key usage of 26
    sam-challenge-label[4]      GeneralString OPTIONAL,
    sam-challenge[5]            GeneralString OPTIONAL,
    sam-response-prompt[6]      GeneralString OPTIONAL,
    sam-pk-for-sad[7]           OCTET STRING OPTIONAL,
    sam-nonce[8]                INTEGER (0..4294967295),
    sam-etype[9]                INTEGER (0..4294967295),
    ...
}

SAMFlags ::= BIT STRING (SIZE (32..MAX))
    -- use-sad-as-key(0)
    -- send-encrypted-sad(1)
    -- must-pk-encrypt-sad(2)
```

5.1.1 SAM-TYPE and SAM-TYPE-NAME Fields

The sam-type field is informational only, but it must be specified

and sam-type values must be registered with the IANA.

Initially defined values of the sam-type codes are:

```
    PA_SAM_TYPE_ENIGMA     1   -- Enigma Logic
    PA_SAM_TYPE_DIGI_PATH  2   -- Digital Pathways
```

```
PA_SAM_TYPE_SKEY_K0    3   -- S/key where  KDC has key 0
PA_SAM_TYPE_SKEY       4   -- Traditional S/Key
PA_SAM_TYPE_SECURID    5   -- Security Dynamics
PA_SAM_TYPE_CRYPTOCARD 6   -- CRYPTOCard
```

PA_SAM_TYPE_SECURID, PA_SAM_TYPE_DIGI_PATH, PA_SAM_TYPE_ENIGMA, and PA_SAM_TYPE_CRYPTOCARD represent popular token cards. PA_SAM_TYPE_SKEY is the traditional S/Key protocol, in which the SAD verifier does not have knowledge of the principal's S/Key secret. PA_SAM_TYPE_SKEY_K0 is a variant of S/Key that uses the same SAD and PC software or hardware device, but where the zeroth key (the S/Key secret) is actually stored on, and can be used by, the SAD verifier to independently generate the correct authentication data.

Note that using PA_SAM_TYPE_SKEY_K0 gives up one advantage of S/Key, viz., that the information required to generate the SAD need not be stored on the host; but since the SAD verifier (which may be the KDC) is assumed to be more secure than other hosts on the network, it may be acceptable to give up this advantage in some situations. The advantage of using this S/Key variant is that the security of the network protocol is strengthened since the SAD need not be sent from the client to the KDC. Thus, the SAD can be used as part of the key used to encrypt the encrypted parts of both the SPD and the KRB_AS_REP message, rather than being sent protected by the principal's Kerberos secret key which may have been previously exposed to an attacker (see section 6, below). In any case, there is a definite advantage to being interoperable with the S/Key algorithm.

Due to the volatility of, and rapid developments in, the area of single-use authentication mechanisms (both software-only and hardware supported), any subsequently defined sam-type codes will be maintained by the IANA.

The optional sam-type-name field is a UTF-8 character string for informational use only. It may be used by the client to display a short description of the type of single-use authentication mechanism to be used.

5.1.2 SAM-FLAGS Field

The sam-flags field indicates whether the SAD is known by the KDC (in which case it can be used as part of the encryption key for the ensuing KRB_AS_REP message), or if it must be provided to the KDC in a recoverable manner. If it is known to the KDC, use-sad-as-key indicates that the SAD alone will be used to generate the encryption key for the forthcoming KRB_AS_REQ and KRB_AS_REP messages,

and that the user will not need to also enter a password. We recommend that this option only be used if the SAD will be used to generate adequate keying material (sufficient length, secrecy, randomness) for the cryptographic algorithm used. If the single-use authentication data is not known (and cannot be generated or discovered) by the KDC, then send-encrypted-sad flag will be set,

indicating that the SAD must be sent to the KDC encrypted under the principal's secret key. If neither use-sad-as-key nor send-encrypted-sad are set, the client may assume that the KDC knows the SAD, but the Kerberos password should be used along with the passcode in the derivation of the encryption key (see below). No more than one of the send-encrypted-sad and use-sad-as-key flags should be in a SAM-CHALLENGE-2.

The must-pk-encrypt-sad flag is reserved for future use. If this flag is set and a client does not support the must-pk-encrypt-sad option (to be defined in a separate document), the client will not be able to complete the authentication and must notify the user.

5.1.3 SAM-CHECKSUM Field

The sam-cksum field contains a sequence of at least one cryptographic checksum of encoding of the PA-SAM-CHALLENGE-2 sequence. If the send-encrypted-sad flag is set, the key to be used for this checksum is the client's long-term secret. If the use-sad-as-key flag is set, then the SAD alone will be used as the key. If neither flag is set, then the key used for this checksum is derived from the SAD and the user's password (see section 5.2).

The checksum algorithm to be used for this is the mandatory checksum associated with the encryption algorithm specified in the sam-etype field, with a key usage of 25.

In some cases there may be more than one valid SAD; some preauthentication mechanisms may have a range of valid responses. In that case, the KDC may elect to return multiple checksums, one for each possible SAD response. The number of possible responses of course depends on the mechanism and site policy. In the case where multiple checksums are returned, the client MUST try each checksum in turn until one of the checksums is verified successfully. Note that in the non-send-encrypted-sad case the checksum cannot be verified until the user enters in the SAD, but if no checksum can be verified, the client MUST not send a response but instead return an error to the user.

The sam-cksum field is generated by calculating the specified checksum over the DER-encoded SAM-CHALLENGE-2-BODY sequence.

If no checksum is included, or is of the wrong type, or none are found which are correct, the client MUST abort the dialogue with the KDC and issue, respectively, KRB5_SAM_NO_CHECKSUM, KRB5_SAM_BAD_CHECKSUM_TYPE, or KRB5_SAM_BAD_CHECKSUM error messages.

5.1.4 SAM-TRACK-ID Field

The optional sam-track-id field may be returned by the KDC in the KRB_ERROR message. If present, the client MUST copy this field into the corresponding field of the SAM response sent in the subsequent KRB_AS_REQ message. This field may be used by the KDC to

match challenges and responses. It might be a suitably encoded integer, or even be encrypted data with the KDC state encoded so that the KDC doesn't have to maintain the state internally. Note that when a KDC supplies a sam-track-id, it MUST link the sam-track-id with the sam-nonce field to prevent spoofing of the sam-track-id field.

The key usage type 26 is reserved for use to encrypt the sam-track-id data. The key used to encrypt the sam-track-id is mechanism-dependent.

5.1.5 SAM-CHALLENGE-LABEL Field

The sam-challenge-label field is informational and optional. If it is included, is will be an UTF-8 encoded character. If present, a client may choose to precede the presentation of the challenge with this string. For example, if the challenge is 135773 and the string in the sam-challenge-label field is "Enter the following number on your card", the client may choose to display to the user:

Enter the following number on your card: 135773

If no challenge label was presented, or if the client chooses to ignore it, the client might display instead:

Challenge from authentication server: 135773

Internationalization is supported by allowing customization of the challenge label and other strings on a per-principal basis. Note that this character string should be encoded using UTF-8.

5.1.6 SAM-CHALLENGE Field

The optional sam-challenge field contains a string that will be needed by the user to generate a suitable response. If the sam-challenge field is left out, it indicates that the SAM in use does not require a challenge, and that the authorized user should be able to produce the correct SAD without one. If the sam-challenge field is present, it is the data that is used by the SAD generator to create the SAD to be used in the production of the SPD to be included in the response.

5.1.7 SAM-RESPONSE-PROMPT Field

The sam-response-prompt field is informational and optional. If present, a client may choose to precede the prompt for the response with the specified string.

Passcode:

5.1.8 SAM-PK-FOR-SAD Field

sam-pk-for-sad is an optional field. It is included in the interest of future extensability of the protocol to the use of

public-key cryptography.

5.1.9 SAM-NONCE Field

The sam-nonce is a KDC-supplied nonce and should conform to the specification of the nonce field in a KRB_KDC_REQ message [RFC1510].

Challenge/Response mechanisms MUST link the nonce field with the sam-track-id (if one is included) to prevent replay of the sam-track-id field.

5.1.10 SAM-ETYPE Field

The sam-etype field contains the encryption type to be used by the client for all encrypted fields in the PA-SAM-RESPONSE-2 message. The KDC should pick an appropriate encryption algorithm based on the encryption algorithms listed in the client's initial KRB_AS_REQ.

5.2 Obtaining SAM Authentication Data

If the client is performing SAM preauthentication in the initial message, without receipt of a PA-SAM-CHALLENGE-2 (i.e. without waiting for the KRB_ERROR message), and the SAM in use does not require a challenge, the client will prompt for the SAD in an application-specific manner.

Once the user has been prompted for and entered the SAD (and possibly the Kerberos password), the client will derive a key to be used to encrypt the preauthentication data for a KRB_AS_REQ message. This key will be determined as follows:

> By default, the key is derived from the password and the SAD by running each through the string_to_key function [RFC1510] separately; i.e., K1 = string_to_key(password) and K2 = string_to_key(SAD). When the keys are both DES or 3DES, keys K1 and K2 will be combined using the algorithm described in Appendix B, ``DES/3DES Key Combination Algorithm''. For all other encryption algorithms, the algorithm described in Appendix A, ``Key Combination Algorithm'' shall be used. Note that this algorithm is not commutative; an implementation MUST insure that K1 is the key corresponding to the user's long-term password, and K2 is the output from the SAD. In either case, the salt used by the string_to_key algorithm for the SAD shall be the same salt as used for the user's password.
>
> If the send-encrypted-sad flag is set, the key will be

derived by running the Kerberos password though the string_to_key function in the normal fashion.

If the use-sad-as-key flag is set and the integrity of the PA-SAM-CHALLENGE-2 PADATA field can be verified using the

sam-cksum field, then the SAD is run through the string_to_key function and the result is used as the encryption key for the request. WARNING: the use of single-use authentication data in this manner is NOT recommended unless the range of the SAD is large enough to make an exhaustive off-line search impractical and the risks involved in the use of SAD alone are fully considered. Also, note that without the availability to the KDC of a relatively static, unique secret key shared with the user, the only mechanisms that can be used to protect the integrity of the PA-SAM-CHALLENGE-2 PADATA field are based on either public key cryptography or the KDC's a priori knowledge of the SAD itself. In the latter case, the client must obtain the SAD from the user and use it to verify the integrity of the challenge before the new KRB_AS_REQ message is sent.

The sam-pk-for-sad field is reserved for future use. If this field is not empty and the client does not support the use of public-key encryption for SAD (to be defined in a separate document), the client will not be able to complete the authentication and must notify the user.

5.3 SAM-RESPONSE PA-DATA

The client will then send another KRB_AS_REQ message to the KDC, but with a padata field with padata-type equal to PA-SAM-RESPONSE-2 and padata-value defined as follows:

```
PA-SAM-RESPONSE-2 ::= SEQUENCE {
    sam-type[0]                 INTEGER (0..4294967295),
    sam-flags[1]                SAMFlags,
    sam-track-id[2]             GeneralString OPTIONAL,
    sam-enc-nonce-or-sad[3]     EncryptedData,
                                   -- PA-ENC-SAM-RESPONSE-ENC
                                   -- Key usage of 27
    sam-nonce[4]                INTEGER (0..4294967295),
    ...
}

PA-ENC-SAM-RESPONSE-ENC ::= SEQUENCE {
     sam-nonce[0]               INTEGER (0..4294967295),
     sam-sad[1]                 GeneralString OPTIONAL,
     ...
}
```

The source of the data included in the PA-SAM-RESPONSE-2 structure depends upon whether or not a KRB_ERROR message was received by the client from the KDC.

5.3.1 SAM-TYPE, SAM-FLAGS, and SAM-NONCE Fields

If an error reply was received, the sam-type, sam-flags, and sam-nonce fields will contain copies of the same fields from the error message.

If no error reply was received (i.e., the client knows that a single-use authentication mechanism is to be used), the sam-type field must be set to a value chosen from the list of registered sam-type codes.

The value of the sam-flags field may vary depending upon the type of SAM in use, but in all cases the must-pk-encrypt-sad flag must be zero. If the send-encrypted-sad flag is set, the sam-sad field must contain the entered single-use authentication data (see Section 5.3.3).

5.3.2 SAM-TRACK-ID Field

Note that if there is no sam-track-id in the request, it MUST be omitted in the response. Otherwise, the sam-track-id data MUST be copied from the SAM-CHALLENGE-2 to the SAM-RESPONSE-2.

5.3.3 SAM-ENC-NONCE-OR-SAD

The sam-enc-nonce-or-sad field represends the results of the preauthentication process. It contains the encrypted SAD or a SAD-encrypted nonce. The PA-ENC-SAM-RESPONSE-ENC message is encrypted with the SAD, password + SAD, or password (based on the sam-flags) with key usage 27. The fields of the PA-ENC-SAM-REPONSE-ENC message are populated as follows:

The sam-nonce contains the nonce from the SAM-CHALLENGE-2. This is the same as the unencrypted sam-nonce described in section 5.2.2.

The sam-sad field contains the SAD if send-encrypted-sad is set in the sam-flags. Otherwise, it is omitted.

5.4 Verification of the SAM-RESPONSE-2

Upon receipt the KDC validates this PADATA in much the same way that it validates the PA-ENC-TS preauthentication method except that it uses the SAD (if available, and possibly in conjunction with saved state information or portions of the preauthentication data) to determine the correct key(s) required to verify the encrypted data. Note that if the KDC uses the sam-track-id field to encode its state, the SAM-verification routine is responsible for including information in that field to detect modification or replay by an attacker.

5.5 KRB5-AS-REP

The rest of the processing of the request proceeds normally, except that instead of being encrypted in the user's secret key, the KRB_AS_REP message is encrypted in the key obtained above. Note,

however, that some single-use authentication mechanisms may require further KRB_AS_REQ/KRB_ERROR exchanges to complete authentication; for example, in order to allow the server to resynchronize with the drifting clock on a time-based token card. In these cases the KDC may respond with another KRB_ERROR message containing a different

sam-type value, along with appropriate prompts and/or challenges. This sequence of exchanges will continue until authentication either succeeds or fails.

6. Requirements for Single-use Authentication Mechanisms

Single-Use Authentication Mechanisms vary in their capabilities. To aid implementers, we summarize here how various types of SAMs would operate using this protocool.

If a SAM system can provide a SAD or a sequence of valid SADs to the KDC, then the implementation SHOULD NOT set the send-encrypted-sad flag. This SAM system should provide the SAD to the KDC, which will combine it with the user's long-term key (password) to generate the key used to generate the checksum placed in the sam-cksum field in the PA-SAM-CHALLENGE-2 message. This combined key will also be used by the KDC to verify PA-SAM-RESPONSE-2 message by using it to decrypt the sam-enc-nonce-or-sad field and as the key to encrypt the KRB-AS-REP. If a SAM system returns a range of valid responses, each response can be used to generate a valid checksum which can be placed in the sam-cksum sequence.

If a SAM system can generate enough entropy, it can set the use-sad-as-key field to use the SAD solely as keying material, but it should be noted that most SAM systems that require the user to enter in a response do not have enough entropy to replace the user's long-term key. The most likely consumer of use-sad-as-key is a hardware token which communicates a key directly with Kerberos client software. With or without the use of use-sad-as-key, this is the preferred method as it protects against offline dictionary attacks against the user's password.

If a SAM system cannot provide a SAD or a sequence of SADs to the KDC, then the send-encrypted-sad flag must be set. In this case, the SAD will be encrypted using the user's long-term key in the PA-SAM-RESPONSE-2 message. It should be noted that this is a weaker solution, as it does not protect the user's password against offline dictionary attacks, and any additional entropy provided by the SAM system cannot be used.

7. Security considerations

Single-use authentication mechanisms requiring the use of the send-encrypted-sad option are discouraged as their use on the network is less secure than the case where a combination of the users password and SAD is used as the encryption key. In particular, when the send-encrypted-sad option is used, an attacker who observes the response and is in possession of the users' secret key

(which doesn't change from login to login) can use the key to decrypt the response and obtain the single-use authentication data. This is dependent on the SAM technology used.

If the KDC sets the must-pk-encrypt-sad flag of the sam-flags field but the client software being used does not support public-key

cryptography, it is possible that legitimate users may be denied service.

An attacker in possession of the users encryption key (again, which doesn't change from login to login) might be able to generate/modify a SAM challenge and attach the appropriate checksum. This affects the security of both the send-encrypted-sad option and the must-pk-encrypt-sad option.

8. Expiration

This Internet-Draft expires on January 19, 2004.

9. References

[RFC1510]
The Kerberos Network Authentication System; Kohl and Neuman; September 1993.

[RFC1760]
The S/Key One-Time Password System; Haller; February 1995

[RFC1636]
Report of IAB Workshop on Security in the Internet Architecture; Braden, Clark, Crocker and Huitema; June 1994

[KCRYPTO]
Encryption and Checksum Specifications for Kerberos 5; Raeburn; May 2002

10. Authors' Addresses

Ken Hornstein
Naval Research Laboratory
4555 Overlook Avenue
Washington, DC 20375

Phone: 202-404-4765
EMail: kenh@cmf.nrl.navy.mil

Ken Renard
WareOnEarth
6849 Old Dominion Dr, Suite 365
Annandale, VA 22003

Phone: 703-622-3469
EMail: kdrenard@wareonearth.com

B. Clifford Neuman
USC/Information Sciences Institute
4676 Admiralty Way #1001
Marina del Rey, CA 90292-6695

Phone: 310-822-1511
EMail: bcn@isi.edu

Glen Zorn
Cisco Systems
500 108th Ave NE
Suite 500
Bellevue, WA 98004

Phone: 425-344-8113
EMail: gwz@cisco.com

Appendix A - Key combination algorithm

Definitions:

prf - Pseudo-random function that outputs an octet string based on an input key and a input octet string (defined in [KCRYPTO])

^ - Exclusive-OR operation

random-to-key - Generates an encryption key from random input (defined in [KCRYPTO])

Given two input keys, K1 and K2, where K1 is derived from the user's long-term password, and K2 is derived from the SAD, output key (K3) is derived as follows:

Two sequence of octets, R1 and R2, shall be produced for each key K1 and K2. R1 and R2 will be generated by iterating over calls to prf() until enough bits are generated as needed by the random-to-key function for the encryption type specified for K3.

The octet-string parameter to the prf() function shall be the ASCII string "CombineA" for K1, and "CombineB" for K2. These have the following byte values:

```
{ 0x43 0x6f 0x6d 0x62 0x69 0x6e 0x65 0x41 }
{ 0x43 0x6f 0x6d 0x62 0x69 0x6e 0x65 0x42 }
```

Furthermore, on each iteration both octet-strings will have appended to them the iteration count in the form of an ASCII, base 10, numeral. The iteration count shall start at zero. The format of the iteration count is equivalant to the C language "%d" format to the printf() function call. Pseudo code implementing this follows:

```
count = 0;
while ( bits < required_bits) {
    sprintf(A1, "CombineA%d", count);
    sprintf(A2, "CombineB%d", count);
    R1 += prf(K1, A1);
    R2 += prf(K2, A2);
    count++;
}
```

When R1 and R2 have been generated, they are truncated if the they are longer than the length required by random-to-key. The key is then generated as follows:

```
    K3 = random-to-key(R1 ^ R2)
```

Appendix B - DES/3DES Key combination algorithm

Definitions:

DR - generate "random" data from an encryption key (defined in [KCRYPTO])

n-fold - "stretches" or "shrinks" a sequence bits to a specified size (defined in [KCRYPTO])

random-to-key - Generates an encryption key from random input (defined in [KCRYPTO])

DK - Derive-Key, defined in [KCRYPTO])

CombineConstant - The ASCII encoding of the string "combine", which is defined as the following byte string:

{ 0x63 0x6f 0x6d 0x62 0x69 0x6e 0x65 }

Note: | means "concatenate"

Given two input keys, K1 and K2, the Combine-Key function is as follows:

R1 = DR(K1, n-fold(K2)) R2 = DR(K2, n-fold(K1))

rnd = n-fold(R1 | R2)

tkey = random-to-key(rnd)

Combine-Key(K1, K2) = DK(tkey, CombineConstant)