Open Software Foundation **J. Wray (DEC)**
Request For Comments: 68.0 **OSF Security SIG**
November 1994

PUBLIC-KEY LOGIN FOR DCE 1.2

1. INTRODUCTION AND CONTEXT

This document contains a discussion of the proposed public-key based login protocol for DCE 1.2. The major goal of the work is to allow log-in without requiring that user secret data be known to the DCE security server, and to provide a means for secure storage of user private-keys for users without smart-cards.

Section 2 of this document describes the two operations that are required in order for a public-key user to log-in to DCE. Section 3 describes a protocol that will perform these two operations, and that does not require the log-in system to have made a previous determination of whether the user is public-key capable. Section 4 describes situations where the two public-key login operations may be required individually, and describes protocols for performing the operations in isolation.

The goal of this effort is to allow users to choose to use public-key cryptography instead of symmetric cryptography to prove their identity to the system. The immediate benefit of such a change is that, in the event of a KDC compromise public-key users will suffer less exposure than normal users. If the KDC is compromised, all user secret keys will be revealed to the intruder, which means that they become worthless as a proof of identity, and therefore the cell administrator must re-issue passwords to all such users before they can be allowed to log-in top the cell. Under the design described in this RFC, public-key users prove their identity by knowledge of a private-key that is never known to the KDC, and therefore a KDC compromise cannot reveal these keys. For public-key users who do not have smart-cards for secure storage of their private-keys, the registry will provide a storage service; although the storage key used for a given user is not known to the registry, a registry compromise will reveal the user's encrypted private-key, enabling a dictionary attack on this key. Therefore, in the event of a registry compromise, public-key users whose private-keys are maintained by the registry should also change their key and password; however (depending on cell policy) they may be permitted to log-in using their current passwords/keys to do so.

2. LOG-IN

Today's DCE uses standard Kerberos V5 mechanisms to perform log-in. A user's DES key is derived from her password, and is used both to prove that the user knows the password, and to protect the returned Kerberos credentials. In DCE 1.2, a public-key principal may still have a DES-key stored in their account entry in the reistry; however this key will be only be used to verify that the user knows her password — no data will ever be protected under it.

This section describes the two basic operations that a public-key principal must perform in order to log-in: retrieval of private-key and acquisition of Kerberos credentials.

2.1. Kerberos Credential Acquisition

In concept, the proposed DCE 1.2 public-key login mechanism is a simple variant of the current DES-based Kerberos log-in. In both mechanisms, a user requests a TGT from the security server, and in response the server creates the TGT and corresponding session-key, encrypts the session-key under a long-term key known to the user, and returns it to the user. The format of the message that contains the TGT and session-key is defined as a `KRB_AS_REP` within the Kerberos protocol.

The user then decrypts the session-key and stores the session-key and the TGT in a credential cache for later use. Successful decryption of the session-key is required in order to be able to use the TGT, and only a user who knows the long-term key may decrypt the session-key.

The major difference between the DCE 1.0/1.1 DES-based log-in protocol and the proposed DCE 1.2 RSA-based protocol is that in the RSA case, the user's session-key is returned encrypted under the user's public-key, which will be stored in the registry as part of the user's account information. Since the `KRB_AS_REP` message includes information as to which crypto algorithm is used to protect the session-key, and since this choice is independent of the algorithm used to encrypt the ticket, the existing `KRB_AS_REP` message may be used to return encrypted credentials for both RSA and DES users.

The use of the RSA algorithm in this way means that the security server needs no knowledge of any confidential data pertaining to the user. While this property improves the ease of recovery from a compromised security server, it introduces one weakness over the DES case — a successful decryption of a public-key-protected session-key cannot be taken as assurance that the message originated at the security server (since anyone who knows the user's public-key, which by definition is public, could have created the encrypted session-key component of the message). However, a successful context certification will verify that the credentials can be trusted, by showing that the session-key is valid for the accompanying TGT, which itself is valid for communicating with a trusted server.

While context certification will typically be performed by system login procedures, there will be cases where existing client programs will create credentials and use them without previously certifying them, basing their trust in the credential on a successful password-based decryption of the session-key (in `sec_login_validate_identity()`). Applications using public-key credentials must take care when using this routine not to assign unwarranted trust to a context on the basis that might be based on falsified credentials. This strongly argues that in DCE 1.2, the `sec_login_certify_identity()` routine be callable without requiring privileges, and that non-privileged applications should be encouraged to certify any contexts they create.

As an alternative to encouraging the use of `sec_login_certify_identity()`, the encryption algorithm used by the security server to encrypt the session-key could be extended to include an RSA-based signature of the encrypted-portion, calculated using the security-server's private-key. This would require that the security-server be given an RSA key-pair, and that the public-key be know to all systems in the cell. This alternative has the advantage that it could remove the requirement for a node to protect secret information, at least if the node were to limit itself to supporting public-key principals. This property would be important in a diskless or public-access environment (DCE security today does not support such environments).

Indeed, such an RSA-based signature could be included in the algorithm used to protect the session-key for even DES-based users. This would allow context certification to be performed by a system without requiring the system to maintain secret data. While this is a useful property, today's DCE requires that systems be capable of acting as principals in several places, and a removal of this requirement is not a part of this proposal. However, further investigation towards lifting this requirement should be carried out as part of the long-term public-key architectural effort.

A minor variant of the above protocol would have the login-system generate a random DES session-key, and include this key within its credential request message, encrypted under the KDC's public-key. The KDC's response (TGT and associated session-key encrytped under the user's public-key) would be super-encrypted under this random key, thus proving to the login system that the response originated at the real KDC. This variant of the protocol is favored, since in the common case of combined login and private-key retrieval (section 3 below), a suitable DES session-key is already available as a result of the existing pre-authentication protocol ([RFC 26.0]).

2.2. Private Key Acquisition

In the preceeding discussion, it was assumed that the login-node has access to the user principal's private-key (in order to decrypt the Kerberos session-key). User private-keys may be stored in smart-cards, but will be more commonly retrieved at log-in time from a network server, the login-agent, co-located with the DCE security server. The login-agent concept and protocol are described in Internet RFC 1507 [RFC 1507]. This section provides a summary of the login-agent's design, and outlines proposed deviations from [RFC 1507].

User private-keys, encrypted under a per-user DES storage key `H1`, are stored within the login-agent. Associated with each such record will be a 64-bit value (`H2`), derived from the user's password, which will be used at log-in to prove knowledge of the password. Algorithms by which hashes `H1` and `H2` are calculated from the password is described in [RFC 1507]; however, for backwards-compatibility with today's secret-key login, an alternate algorithm will be used for calculation of `H2`.

The private-key retrieval process described in [RFC 1507] is as follows. The login-node sends a time-stamped request, encrypted under the public-key of the login-agent, containing the name of the principal whose key is being requested, a random DES key `S1`, and the hash-value `H2`.

The login-agent will decrypt the request, verify the time-stamp and check that the hash matches its stored value for the user principal record. If so, the login-agent will retrieve the user's encrypted private-key, encrypt it under DES key `S1`, and return it to the login-node.

The login-node decrypts the private-key using first key `S1` followed by key `H1`. Since `S1` is chosen randomly for each request, the response is bound to a specific request.

The details of the protocol described above will differ in the DCE implementation, primarily to support a combined login/key-retrieval that does not require the login-system to previously determine whether the user is public-key capable.

3. COMBINED PRIVATE-KEY RETRIEVAL AND LOG-IN PROTOCOL

Since the login-agent and the DCE security server will be co-located, there is the potential to reduce the number of messages required when a login-node first retrieves a user's private-key and then requests Kerberos credentials. The DCE 1.1 pre-authentication protocol provides a means whereby the two transactions may be combined into a single request-response pair.

The Kerberos protocol provides a means whereby additional (non-Kerberos) data may be provided by a login-node along with a Kerberos encrypted-credential request, for the purpose of allowing the Kerberos KDC to gain some assurance that the entity requesting encrypted credentials is actually entitled to them. Several alternate pre-authentication schemes may be in use simultaneously, as the pre-authentication data field contains a field indentifying the particular scheme used.

DCE 1.1 uses this general mechanism to implement two such pre-authentication schemes:

(a) `PADATA−ENC−TIMESTAMPS`

In this scheme, a timestamp is encrypted under the user principal's key and sent with the TGT request as proof that the login-node has access to the user's password. This defeats an attack where an intruder simply requests encrypted credentials for a user, and then mounts a dictionary attack on the encrypted credential. Such an attack is possible because the credential is encrypted under a DES key derived from a human-chosen password.

While more secure than a request with no pre-authentication data, this scheme is still vulnerable to an intruder who records a valid login exchange. Since both the request and the response are encrypted under the password-derived user key, a dictionary attack is still available.

(b) `PADATA−ENC−THIRD−PARTY`

This scheme relies upon the fact that DCE systems have keys of their own, and can act as Kerberos clients. The pre-authentication data contains the login-system's TGT, a random key `S2` encrypted under the session-key from the TGT, and a timestamp encrypted first under the user's key and then again under `S2`. On receipt of a `PADATA−ENC−THIRD−PARTY` request, the KDC decrypts the TGT to find the session-key, uses that to decrypt `S2`, and uses `S2` and the claimed principal's key to decrypt the timestamp. If the timestamp is valid, the KDC knows that the login-system has access to the user password, and constructs a TGT and session-key protected under `S2`.

In this scheme, all traffic between client and KDC is protected by machine-generated (as opposed to password-derived) keys, and will therefore not be amenable to dictionary attacks.

The combined private-key retrieval and login protocol will use the `PADATA−ENC−THIRD−PARTY` format. The login-agent's hash `H2` (described above) will be calculated using the same algorithm as is used in today's DCE for converting a password into a DES key. Therefore, the receipt of a DCE 1.1-format `PADATA−ENC−THIRD−PARTY` pre-authentication field attached to a request for a public-key user login will provide proof to the login-agent that the system requesting the login does have access to the user's password. This will also mean that `H2` need not be stored in an additional registry field; the current ''account key'' field will be interpreted as a ''password hash'' for public-key users, and never used as a DES key. An algorithm for deriving the storage key `H1` is still to be determined; one option is to use the algorithm specified in [RFC 1507].

Whether full [RFC 1507] compatibility should be provided in addition to the DCE-specific variant is for further discussion.

Since the `PADATA-ENC-THIRD-PARTY` protocol generates a session-key `S2`, which is known only to the login-system and the KDC, and since the KDC's response to a combined retrieval and login request will be super-encrypted under this DES key, the login-system can be sure that it has obtained its credentials from a genuine DCE security server, thus removing the context certification requirement discussed above in section 2.1.

4. INDIVIDUAL KEY-RETRIEVAL AND CREDENTIAL ACQUISITION PROTOCOLS

While the combined protocol described above is expected to be the dominant login mechanism for public-key users, the two functions of Kerberos credential acquisition and private-key retrieval will be available via independent network requests. Kerberos credential acquisition without private-key retrieval will be needed by users who use smart-cards to store their RSA keys, and private-key retrieval without Kerberos acquisition may be needed for users who have multiple key-pairs (possibly for use with different applications). This section describes the protocols used to perform these operations in isolation.

4.1. Kerberos Credential Acquisition Protocol

The assumption is that the user already has access to her private-key, and wishes to acquire Kerberos credentials. The user will issue a standard Kerberos TGT request, but using a new pre-authentication field with a type of `PADATA-ENC-PUBLIC-KEY`.

The simplest form of this protocol would simply have the presence of a `PADATA-ENC-PUBLIC-KEY` field indicate to the registry that the user already has her private-key, and that it should return Kerberos credentials, protecting the Kerberos session-key under the user's public-key. Since this protocol sends no secret data over the network encrypted in anything other than truly random machine-generated keys, additional pre-authentication may not be strictly necessary. However, portions of DCE 1.2 (e.g., ''DCE who'') make the use of a true pre-authentication protocol valuable, so the following variant will be used instead.

In the first form, the pre-authentication field will contain the user's name and a timestamp, signed with the user's private-key. The registry will verify the signature (since it knows the user's public-key), decrypt and verify the timestamp, and (assuming a successful signature verification) will create a Kerberos TGT/session-key and return these to the user system, protecting the session-key under the user's public-key, in a standard Kerberos `KRB_AS_REP` message (using DCE RPC as a transport in the usual way). This form of the protocol will require that the encryption used to protect the session-key include a signature using the registry's public-key, as described in section 2.1 above.

A minor variation on the above protocol would perform all public-key operations on the initial message, and saves one public-key operation. In this version of the protocol, the pre-authentication field contains the user's name, a random DES key S3, and a timestamp. The timestamp and DES key are encrypted under the public-key of the registry, and the entire message is signed with the user's private-key. The returned Kerberos credentials will be protected under S3.

4.2. Private Key Retrieval Protocol

There are two possible protocol options available for private-key retrieval. Both require the login-system to generate a DES key and pass it to the login-agent, which uses this key to protect the encrypted private-key in the reply. The difference is in how the key is protected on its journey to the login-agent, and the means by which the login-system proves knowledge of the user's password.

The first protocol is a simple variant of the [RFC 1507] protocol. This protocol assumes that the login-agent is issued a key-pair, the public component of which is known to all systems within the cell. This public-key is used to protect the message from login-system to login-agent, which contains the hash `H2`, a timestamp, and the DES key to be used for the response. The request also names the principal whose key is to be retrieved. The response consists of a message containing the encrypted private-key, super-encrypted under the DES key from the first message.

The only difference between this option and the [RFC 1507] protocol is that `H2` is calculated using the DCE algorithm, as described above in section 3.0.

The second option is to use the machine's key to protect the key in transit, as in the `PADATA-ENC-THIRD-PARTY` pre-authentication protocol.

REFERENCES

[RFC 1507] C. Kaufman, ''Distributed Authentication Security Service'', Internet RFC 1507.

[RFC 26.0] J. Pato, ''Using Pre-authentication to Avoid Password-Guessing Attacks'', DCE-RFC 26.0.

AUTHOR'S ADDRESS

John Wray
Digital Equipment Corporation
550 King Street, LKG2-2/Z7
Littleton, MA 01460
USA

Internet email: wray@tuxedo.enet.dec.com
Telephone: +1-508-486-5210