Internet Engineering Task Force — **Eddie Kohler**
INTERNET-DRAFT — **UCLA**
draft-kohler-dccp-mobility-02.ps — **25 June 2006**
Expires: 25 December 2006

Generalized Connections in the Datagram Congestion Control Protocol

Status of this Memo

Abstract

This document describes a mechanism by which the set of addresses bound to an application-level Datagram Congestion Control Protocol (DCCP) connection [RFC4340] can change over that connection's lifetime. The essential abstraction is the Generalized Connection, which combines multiple distinct transport-level DCCP connections into a single application-level entity.

Table of Contents

1. Introduction

The Datagram Congestion Control Protocol (DCCP) as currently specified [RFC4340] does not support address rebinding. Each DCCP connection is associated with exactly two network-level addresses over its lifetime, one per endpoint. However, there are good arguments for supporting address rebinding, such as mobility and multihoming, at the transport layer, not least required interactions with congestion control. DCCP is an unreliable protocol; its application-level semantics allow packet reordering, loss, and duplication. This allows DCCP to address the address rebinding problem in a particularly simple way. Multiple transport-level DCCP connections, with different sequence number spaces, congestion control state, and so forth, are simply aggregated in the relevant endpoints into a single application-level entity, called a Generalized Connection. Little coordination is required among a generalized connection’s components.

2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

All multi-byte numerical quantities in DCCP, such as port numbers, Sequence Numbers, and arguments to options, are transmitted in network byte order (most significant byte first).

Random numbers in DCCP Generalized Connections are used for their security properties and SHOULD be chosen according to the guidelines in [RFC4086].

3. Requirements

The generalized connection mechanism was designed to fulfill the following requirements and non-requirements.

- An endpoint does not need to announce a new address before moving to that address.

  RATIONALE: Mobile hosts may not know a new address until a move completes; and by that time, the old address may not be usable. Some multihomed hosts can know each of their addresses, but announcing addresses before using them does not prevent all attacks; see, for example, the "address squatting" attacks in [ANC04].

- Move requests must be safe against hijacking. Even attackers that can snoop on part or all of data traffic must not be able to move a connection. However, move requests need not be safe against man-in-the-middle attackers with control over which packets get delivered (such as routers).

  RATIONALE: Moving a connection is in some ways the worst possible attack: An attacker takes over a user’s identity, without the user becoming aware of the attack or being able to stop the attack. We must prevent this. However, an endpoint with full control over the path could carry out this kind of attack even without mobility support. Therefore, we choose to allow a DCCP mobility mechanism to be vulnerable to attackers that can snoop a packet sent by the mobile host, then prevent that snooped packet from being delivered to the stationary host.

- Mobility must not create new, large opportunities for denial-of-service attacks.
- Endpoints must be able to move freely between different NAT domains using the mobility mechanism.

- Simultaneous moves need not be supported.
- Cryptography is allowed.

It is difficult, perhaps impossible, to fulfill both the NAT traversal and hijacking prevention requirements. Natural mechanisms for preventing hijacking, such as cryptographically signing the packet's network headers, fail in the presence of NATs, which change those headers. NATs essentially hijack connections by definition; we want to allow that, but prevent malicious hijacking. The solution below represents one attempt.

4. Protocol

4.1. Overview

A generalized connection groups one or more transport connections, called component connections, into a single application-level entity. To move addresses, a host attaches a new component connection, then detaches the old component connection. To implement multihoming, a host maintains multiple component connections with different endpoint addresses.

The multiple component connections that make up a generalized connection are treated independently at the transport level. They maintain independent sequence number spaces and congestion control state, for example. The application, however, sees only one socket, which corresponds to the generalized connection. Data received on any component connection is sent to that socket, and data sent via the socket may be transmitted over any component connection.

The first connection handshake in a generalized connection must register the intention to set up a generalized connection. The generalized connection's identifier is then agreed upon by the two endpoints (assuming they both support generalized connections). Thereafter, new component connections specify the intended generalized connection in their handshakes. A public-key cryptographic protocol prevents connection hijacking by passive attackers. However, attackers who can prevent packets from being delivered, or alter packets in flight, can hijack the connection, as is also possible in the absence of this extension.

```
(1) Connection initiation with preparation for generalized
    connections:
    A  -->  DCCP-Request with Initiate Gencon   -->  B
    A  <--  DCCP-Response with Approve Gencon   <--  B

(2) Adding a new component to the generalized connection:
    A’ -->  DCCP-Request with Attach Gencon     -->  B
    A’ <--  DCCP-Response with Challenge Gencon <--  B
    A’ -->  DCCP-Ack with Confirm Gencon        -->  B

(3) Removing a component from the generalized connection:
    A  -->  DCCP-Sync with Detach Gencon        -->  B
    A  <--  DCCP-SyncAck                        <--  B

(4) Marking a component as preferred for data transfer:
    A  -->  any packet with Prefer Gencon       -->  B
```

4.2. Gencon Option

The Gencon option, for Generalized Connection, is used to implement the subprotocols that create and update generalized connections. These subprotocols may contain messages that exceed the 253-byte option length limit. Therefore, the payloads from all of a packet’s Gencon options are concatenated to form a single Gencon message.

Gencon messages follow a common format, as follows.

```
+--------+------ ... ------+------ ... ------+-------- ...
| GCType |    Gencon ID    |  Component ID   | Payload
+--------+------ ... ------+------ ... ------+-------- ...
              (8 bytes)         (4 bytes)
```

Gencon Type (GCType): 8 bits

Defines the Gencon message type. Seven types are currently defined, as follows:

GCType	Meaning	Payload
0	Initiate	Public Key
1	Approve	Public Key
2	Attach	Optional Nonce
3	Challenge	Nonce and Optional Token
4	Confirm	Token
5	Detach	Token
6	Prefer	-
7-255	Reserved	

Table 1: Gencon Types

Gencon ID: 64 bits (8 bytes)

The Gencon ID uniquely identifies the generalized connection at both endpoints, and is used to identify new component connections for an existing generalized connection. To ensure uniqueness at both endpoints, the Gencon ID is defined in two parts: the client defines the upper 32 bits in its Initiate Gencon message, which is sent on the first DCCP-Request, and the server adds the lower 32 bits in its Approve Gencon message, which is sent on the corresponding DCCP-Response. Neither the upper nor the lower 32 bits of the Gencon ID may equal zero.

Component ID: 32 bits (4 bytes)
The Component ID identifies a component connection within a generalized connection. It MUST NOT equal zero.

4.3. Initiate Gencon Message

The client sends an Initiate Gencon message on the DCCP-Request packet that initiates a new generalized connection. This message contains half of the Gencon ID that will define the generalized connection, and the client's public key that will be used to validate later Gencon messages. The server's DCCP-Response packet will include an Approve Gencon message to complete the handshake.

The Initiate Gencon message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000000|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=0    (8 bytes)       (4 bytes)

    +--------+--------+--------+--------+------ ... ------+
    |   Crypto Suite  |     Length      |       Key       | ...
    +--------+--------+--------+--------+------ ... ------+
                                          (Length-4 bytes)
```

Gencon ID
The client specifies the upper 32 bits (4 bytes) of the generalized connection's Gencon ID in its Initiate Gencon message. These bits MUST NOT equal zero, while the lower 32 bits MUST equal zero. They also MUST NOT equal the upper 32 bits of the Gencon ID of any other generalized connection currently active at the server. Furthermore, they SHOULD be chosen so as to minimize duplication: that is, no recently-active generalized connection from this endpoint should have had the same upper 32 bits.

Component ID
This field is chosen by the client to identify this component connection. It MUST NOT equal zero, and its most significant bit MUST equal zero; thus, values 1-2147483647 are acceptable. One is a perfectly reasonable choice for this first component connection.

Crypto Suite: 16 bits
Defines the public-key cryptographic protocol to be used by other Gencon messages to validate future component connections. The Crypto Suite defines how certain other fields, such as Key, are interpreted.

Length: 16 bits
Equals the byte length of the following Key, plus four.

Key: variable length

The client's public key in the specified Crypto Suite. The receiving endpoint uses this public key only for this generalized connection, although an endpoint MAY reuse a public key on multiple generalized connections. There is no way to change this key on an existing generalized connection; if a client suspects that a generalized connection's key has been compromised, the client should shut down all corresponding generalized connections. The format of this field depends on the value of Crypto Suite.

Remainder of message

The Initiate message MAY contain more than one Crypto Suite/Length/Key triple, allowing the receiving endpoint to select a Crypto Suite it understands.

The Crypto Suite/Key pair resembles a DCCP feature. We do not use the existing DCCP feature negotiation mechanism for several reasons: (1) Crypto Suite/Key negotiation can take place only on connection initiation. (2) Keys in some Crypto Suites may exceed the 255-byte feature length limitation.

4.4. Approve Gencon Message

On receiving a DCCP-Request containing an acceptable Initiate Gencon message, the server responds with a DCCP-Response packet containing an Approve Gencon message. This serves three purposes: it acknowledges the creation of a generalized connection, it completes the specification of the Gencon ID, and it defines the server's public key.

The Approve message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000001|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=1    (8 bytes)       (4 bytes)

    +--------+--------+--------+--------
    |   Crypto Suite  |      Key ...
    +--------+--------+--------+--------
```

Gencon ID

The upper 32 bits of this field MUST equal the value specified by the client in its Initiate Gencon message. The lower 32 bits are newly provided by the server, and MUST NOT equal zero. The result -- a 64-bit number, neither of whose halves equals zero -- is the connection's complete Gencon ID. The server MUST choose these lower 32 bits so that no other currently-active connection with the same endpoint has the same Gencon ID. Furthermore, it SHOULD choose these bits so as to minimize duplication, ensuring that new connections receive Gencon IDs that have not been seen before.

Component ID

MUST equal the value specified by the client in its Initiate message.

Crypto Suite

MUST equal one of the Crypto Suites specified by the client in its Initiate message.

Key The server’s public key in the specified Crypto Suite. See the comments above on the client’s public key. No Length field is necessary as this Gencon message will contain exactly one Key.

4.5. Attach Gencon Message

To add a component connection to a generalized connection, one of the endpoint hosts opens a new DCCP connection to the other in the conventional way -- that is, using a DCCP-Request packet. This DCCP-Request packet contains an Attach Gencon message with the generalized connection’s Gencon ID. This initial message is unverified; a protocol consisting of a Challenge Gencon message, sent on the DCCP-Response, and a Confirm Gencon message, sent on the DCCP-Ack, verifies that the mobile endpoint host’s private key approves of the move.

Note that the "client" of the new component connection need not be the same endpoint as the "client" of the original component connection. The original server is the endpoint that received the original DCCP-Request containing an Initiate Gencon message; to add a new component connection, it sends a DCCP-Request Gencon message to the original client, and is thus the client for the new component connection. Alternatively, the server could convince the client to open a new connection using application messages of some kind.

The Attach message has the following format:

```
+--------+----- ... -----+----- ... -----+----- ... -----+
|00000010|   Gencon ID   | Component ID  |    C-Nonce    |
+--------+----- ... -----+----- ... -----+----- ... -----+
 GCType=2    (8 bytes)       (4 bytes)       (8 bytes)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of this new component connection. This value is chosen by the component client; it MUST NOT have been used for any previous successful component connection, and MUST NOT equal zero. The value of the most significant bit is set based on whether the component client (the endpoint sending this DCCP-Request) is the original client (the endpoint that sent the first DCCP-Request in the generalized connection), according to this table:

Component Client	Component ID MSB	Component ID Range
Original Client	0	1-2147483647
Original Server	1	2147483648-4294967295

This restriction ensures that the endpoints will not pick the same Component ID if they try to attach new component connections simultaneously.

C-Nonce: 64 bits
Nonce fields are used as challenges to verify that the other protocol endpoint knows the expected private key. The special value zero indicates the lack of a nonce. Nonce values MUST be chosen apparently randomly, and MUST NOT be reused on the same generalized connection ID. (That is, given an endpoint and a Gencon ID, the multiset of Nonce values contained in Gencon messages with that endpoint and

Gencon ID must contain no duplicates, except possibly for zero.) Thus, attackers should not be able to predict the next nonce an endpoint will use. Endpoints can obtain apparently-random Nonce values either with sources of true randomness (as long as values are not reused), through encryption operations (for example, using a block cipher to encrypt a value that increases by one per Nonce), or possibly in other ways. If an endpoint uses encryption, it MUST include a random salt generated specifically for the generalized connection, ensuring that attackers cannot predict the next Nonce even given the value of the current Nonce. Random numbers

The component client MAY include an 8-byte Nonce, called the C-Nonce, in the Attach Gencon message. This expresses a desire to verify that the component server is valid, i.e. knows the expected private key.

4.6. Challenge Gencon Message

The DCCP-Response packet sent in response to a new component connection -- that is, to a DCCP-Request packet containing an Attach Gencon message -- MUST contain a Challenge Gencon message. This message is effectively an Init Cookie; the component server MUST NOT accept the new component connection until the Challenge is correctly confirmed with a Confirm Gencon message.

The Challenge Gencon message has the following format:

```
+--------+----- ... -----+----- ... -----+
|00000011|   Gencon ID   | Component ID  | (continued)
+--------+----- ... -----+----- ... -----+
 GCType=3    (8 bytes)       (4 bytes)

    +----- ... -----+--------+--------
    |    S-Nonce    |      Token ...
    +----- ... -----+--------+--------
        (8 bytes)       (optional)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of the new component connection; MUST equal the Component ID from the corresponding Attach Gencon message.

S-Nonce
This Nonce is used to verify that the component client knows the relevant private key. It follows the restrictions described above, and MUST NOT be zero.

Token: variable length
If the Attach Gencon message contained a nonzero C-Nonce, then the component server MUST include a signed Token in its Challenge Gencon message. This Token will prove to the component client that the component server knows the relevant private key. The format for the Token depends on the specified Crypto Suite. Most Crypto Suites will construct Tokens using the following general procedure.

To create a Token in response to a particular Nonce, an endpoint constructs the following 40-byte Generic Token Message structure:

```
Bytes +--------+--------+--------+--------+
  0   | GCType |00000000| Sequence Number .   3
      +--------+--------+--------+--------+
  4   .     Sequence Number (low bits)    |   7
      +--------+--------+--------+--------+
  8   |00000000|00000000|   Ack. Number   .  11
      +--------+--------+--------+--------+
 12   . Acknowledgement Number (low bits) |  15
      +--------+--------+--------+--------+
 16   |       Gencon ID (high bits)       .  19
      +--------+--------+--------+--------+
 20   .       Gencon ID (low bits)        |  23
      +--------+--------+--------+--------+
 24   |           Component ID            |  27
      +--------+--------+--------+--------+
 28   |           Component ID            |  31
      +--------+--------+--------+--------+
 32   |         Nonce (high bits)         .  35
      +--------+--------+--------+--------+
 36   .         Nonce (low bits)          |  39
      +--------+--------+--------+--------+
```

GCType is the Gencon Type of the Gencon message containing the Token. Sequence Number is the Sequence Number of the packet that will contain this message. Acknowledgement Number is the Acknowledgement Number of the packet that will contain this message, which must equal the Sequence Number of the packet that contained the Nonce. Note that the Component ID is included twice.

This structure is then signed using the local endpoint’s private key. The signature is the Token. The other endpoint can decrypt the Token using the corresponding public key; if the signature matches, then some party that knows the relevant private key approved of those contents.

The Nonce in a Challenge Gencon message’s Token shall equal the C-Nonce included in the corresponding Attach Gencon message.

4.7. Confirm Gencon Message

After receiving the DCCP-Request containing a Challenge Gencon message, the component client MUST send a DCCP-Ack packet containing a Confirm Gencon message in response. This message contains a token that the component server will use to verify the client’s identity.

The Confirm message has the following format:

```
+--------+----- ... -----+----- ... -----+--------+--------
|00000100|   Gencon ID   | Component ID  |      Token ...
+--------+----- ... -----+----- ... -----+--------+--------
 GCType=4    (8 bytes)       (4 bytes)       (variable)
```

Gencon ID
    The Gencon ID of the generalized connection.

Component ID
The Component ID of the new component connection; MUST equal the Component ID from the corresponding Attach and Challenge Gencon messages.

Token
A Token created in response to the S-Nonce from the Challenge Gencon message.

4.8. Detach Gencon Message

A component connection may be closed in the usual way, via DCCP-CloseReq, DCCP-Close, and DCCP-Reset packets. Sometimes, however, an endpoint loses control of a component connection before getting a chance to close it; this may particularly happen in mobile hosts. The Detach Gencon message allows an endpoint to close a different component connection by Component ID. The receiver treats the named component connection(s) as if they had received a sequence-valid DCCP-Reset message, with Reset Reason 1, "Closed". Detach Gencon messages MUST be sent only on DCCP-Sync, DCCP-CloseReq, DCCP-Close, and DCCP-Reset packets. Since a packet can contain at most one Gencon message, one component connection can be detached per packet (unless the Component ID is set to zero, in which case all component connections are detached except the current component connection).

The Detach message has the following format:

```
+--------+----- ... -----+----- ... -----+--------+--------
|00000101|   Gencon ID   | Component ID  |      Token ...
+--------+----- ... -----+----- ... -----+--------+--------
 GCType=5    (8 bytes)       (4 bytes)       (variable)
```

Gencon ID
The Gencon ID of the generalized connection.

Component ID
The Component ID of the component connection that should be closed. This MUST NOT equal the Component ID of the component connection on which the message appears. The special value of 0 indicates that all component connections of the given generalized connection should be shut down EXCEPT for the component connection on which the message appears.

Token
A Token, as above. The Nonce field used to construct the Token equals zero.

4.9. Prefer Gencon Message

The Prefer Gencon message asks the receiving endpoint to send its data over the named component connection. This allows an endpoint to manage how it receives data in case that, for example, different simultaneously-active component connections have different costs or varying reliability. This information is treated as a hint; the receiving endpoint need not actually change how it sends data.

The Prefer message has the following format:

```
+--------+----- ... -----+----- ... -----+--------+--------
|00000110|   Gencon ID   | Component ID  |      Token ...
+--------+----- ... -----+----- ... -----+--------+--------
 GCType=6    (8 bytes)       (4 bytes)       (variable)
```

Gencon ID

The Gencon ID of the generalized connection.

Component ID

The Component ID of the component connection that the receiving endpoint should use to send data.

Token

A Token, as above. The Nonce field used to construct the Token equals zero.

4.10. Gencon Reset Code

DCCP Reset Code 12 is allocated for resets relating to Generalized Connections. The Data 1 field corresponds to the Gencon Type of the Gencon message that caused the reset. The Data 2 & 3 fields default to zero, or are set as described below.

4.11. Unexpected Options

Endpoints MUST handle invalid, unexpected, and otherwise malformed Gencon options in the following way.

- A Gencon message is Mandatory when any of its component Gencon options is marked Mandatory. If a Mandatory Gencon message is "ignored" according to the following list, then the receiving endpoint MUST reset the connection using Reset Code 6, "Mandatory Failure", as described in [RFC4340], Section 5.8.2.
- Any Gencon message that does not meet basic formatting requirements, such as message length, MUST be ignored.
- Any Gencon message with unrecognized Gencon Type MUST be ignored.
- An Initiate Gencon message received on any packet other than a DCCP-Request MUST be ignored.
- An Initiate Gencon message whose Gencon ID and/or Component ID do not meet the specified requirements MUST be ignored.
- An Initiate Gencon message whose Crypto Suites are not understood, whose Length values are invalid (less than four or too large for the Gencon message) or inappropriate for the corresponding Crypto Suite, or whose corresponding Key does not meet the requirements of the selected Crypto Suite, MUST result in connection reset. The receiver will reset the connection with Reset Code 12. The Data 2 & 3 fields of the DCCP-Reset packet are set to the problematic Crypto Suite.
- An Approve Gencon message received on any packet other than a DCCP-Response MUST be ignored.
- An Approve Gencon message received on a DCCP-Response, where the corresponding DCCP-Request did not contain an Initiate Gencon message, MUST be ignored.
- An Approve Gencon message whose Gencon ID and/or Component ID do not meet the specified requirements, or whose Crypto Suite does not equal some Crypto Suite from

its client's Initiate message, or whose Key does not meet the requirements of the Crypto Suite, MUST be ignored.

- An Attach Gencon message received on any packet other than a DCCP-Request MUST be ignored.
- An Attach Gencon message whose Gencon ID does not correspond to a current connection, or whose Component ID is zero, or whose Component ID was used by a previous successful component connection on this generalized connection, MUST be ignored. (A component connection is "successful" once a suitable Confirm Gencon message has been received.)
- A Challenge Gencon message received on any packet other than a DCCP-Response MUST be ignored.
- A Challenge Gencon message received on a DCCP-Response, where the corresponding DCCP-Request did not contain a Challenge Gencon message, MUST be ignored.
- A Challenge Gencon message whose Gencon ID and/or Component ID do not correspond to the Attach message's, or whose Nonce is zero, MUST be ignored.
- A Challenge Gencon message sent in response to an Attach message with a Nonce, whose Token is missing or invalid, MUST be ignored.
- A Confirm Gencon message received on any packet other than a packet whose Acknowledgement Number equals that of a DCCP-Response packet that contained a Challenge message, MUST be ignored.
- A Confirm Gencon message whose Gencon ID and/or Component ID do not correspond to the Attach message's, or whose Token is missing or invalid, MUST be ignored.
- A Detach Gencon message received on any packet other than a DCCP-Sync, DCCP-CloseReq, DCCP-Close, or DCCP-Reset packet MUST be ignored.
- A Detach Gencon message whose Gencon ID does not correspond to the expected Gencon ID, or whose nonzero Component ID does not equal that of a currently active component connection, MUST be ignored.
- A Detach Gencon message whose Component ID equals that of the component connection on which the message appears MUST be ignored.
- A Detach Gencon message whose Token is missing or invalid MUST be ignored.
- A Prefer Gencon message whose Gencon ID does not correspond to the expected Gencon ID, or whose Component ID does not equal that of a currently active component connection, MUST be ignored.
- A Prefer Gencon message whose Token is missing or invalid MUST be ignored.

5. Using Generalized Connections

A client that expects to use address rebinding, or that wants to support servers that use address rebinding, SHOULD send an Initiate Gencon message on its DCCP-Request. If the server responds with a valid Approve Gencon message, then the connection is Gencon-enabled; and as a consequence, the application's data stream may be associated with more than one active connection. This section describes how that association is managed.

A generalized connection lasts as long as it has at least one associated component connection. When a generalized connection's last component connection is closed (moves to TIMEWAIT or CLOSED state), either through an explicit termination handshake or because of a timeout, the application-level connection MUST also be closed. APIs that report a reason for connection closure SHOULD use the reason associated with the last-closed component connection; if more than one connection closes at the same time, the choice is arbitrary.

When a generalized connection has multiple components, the endpoint must decide which connection to use to send data. Unless there is some external basis for choice, such as cost, the endpoint SHOULD send its data on the last connection for which it received a valid Prefer message.

An endpoint SHOULD NOT use generalized connections simply to improve its throughput with parallel connections. There SHOULD be a substantive difference between the component connections, such as different network access technologies or failure dependencies. An endpoint that suspects that its partner is "cheating" with generalized connections MAY reset one or more component connections with Reset Code 11, "Aggression Penalty".

When multiple component connections are sending data, that data MUST be enqueued for the application in the order it is received. There is no way, other than simple delay, to enforce ordering among data received on different component connections. This is intentional; application-level sequence numbers are easily layered on top of DCCP, and lack of sequencing may discourage aggressive use of parallel component connections.

6. Crypto Suites

One Crypto Suite is currently recognized for use with Generalized Connections. Crypto Suite 1, RSA-SHA512, implies the RSASSA-PKCS1-v1_5 signature scheme described in [RFC3447], Section 9.2.1, using the SHA-512 hash function. Key values are represented in ASN.1 using the RSA public key syntax described in [RFC3447], Appendix A.1.1. Only the encoded RSAPublicKey sequence is transmitted, not the rsaEncryption OID. Valid Keys MUST have modulus "n" greater than or equal to 2^1536, and publicExponent "e" greater than or equal to 65537. Token values are signatures output from the RSASSA-PKCS1-V1_5-SIGN function defined in [RFC3447], Section 8.2.1, where the EMSA-PKCS1-v1_5-ENCODE function uses the SHA-512 hash function and the SHA-512 DigestInfo value. The message signed to create the Token equals the Generic Token Message construct described in Section 4.6.

7. Security Considerations

The base DCCP protocol is intended to protect against some classes of attacks, and explicitly declares itself vulnerable to other classes of attacks. Specifically,

> Attackers cannot hijack a DCCP connection (close the connection unexpectedly, or cause attacker data to be accepted by an endpoint as if it came from the sender) unless they can guess valid sequence numbers. Thus, as long as endpoints choose initial sequence numbers well, a DCCP attacker must snoop on data packets to get any reasonable probability of success. Sequence number validity checks provide this guarantee. See Section 18 of [RFC4340].

The address rebinding support described in this document preserves this security model for existing connection features. Generalized connections, however, enlarge the possible semantics of DCCP interactions. This section describes the security consequences of the Gencon mechanism, as applied to those new semantics.

A "full hijacking" attack is defined as an attack where, using mobility and multihoming support, an attacker transparently adds itself as an endpoint to a generalized connection. Any attacker that resides on the path, and can control the delivery of messages, thus faking the ownership of an endpoint's IP address, can execute a full hijacking attack; this is true in unextended DCCP, and multihoming and mobility support does not change this fact. DCCP multihoming and mobility support aims to provide the following security guarantee for other attackers: An attacker cannot successfully execute a full hijacking attack unless it (1) can snoop the channel, and (2) knows an endpoint's private key.

Assume that the attacker does not know an endpoint's private key. Such an attacker cannot generate correct Tokens, and in particular, it cannot generate the Gencon Confirm Token required to complete a connection handshake. Thus, the only way for it to execute a full hijacking attack is by replaying a previous Gencon Confirm message. That message must have the same Gencon ID as the connection to be hijacked. The current connection must not have successfully used the replayed message's Component ID. The attacker must use the same Sequence Number as in the replayed message, and convince the other endpoint to use the same Sequence Number for its packets (ensuring that the Acknowledgement Number in the replayed message is correct). All of this is difficult, but not impossible. However, the attacker must also arrange for the other endpoint to use the same Nonce as in the previously replayed message, and the protocol explicitly forbids this.

8. IANA Considerations

IANA is requested to reserve the value 45 for the Gencon option from the DCCP Option Types registry, and to create a new registry for DCCP Gencon Types, populated initially with the values in Table 1.

9. Thanks

Thanks to Pasi Sarolahti for comments that inspired revisions to the specification.

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol", RFC 4340, March 2006.

Informative References

[ANC04] Tuomas Aura, Pekka Nikander, and Gonzalo Camarillo. Effects of Mobility and Multihoming on Transport-Protocol Security. 2004 IEEE Symposium Security and Privacy.

Authors' Addresses

Eddie Kohler <kohler@cs.ucla.edu>
4531C Boelter Hall
UCLA Computer Science Department
Los Angeles, CA 90095
USA

Full Copyright Statement

Intellectual Property