Internet Engineering Task Force
Internet Draft
draft-byerly-sip-radius-00.txt
October, 2000
Expires: March, 2001

Bryan J. Byerly
David Williams
Cisco Systems

SIP Authentication using CHAP-Password

Status of this Memo

Abstract

This document describes a proposed extension to SIP. This document proposes using an alternative SIP authentication mechanism for use in Proxy-Authorization or Authorization headers in order to support SIP client Authentication using backend RADIUS servers.

The introduction of this extension would allow a SIP proxy (or called SIP client) to authenticate a SIP client using a backend RADIUS server.

1 Introduction

Some ISPs currently use RADIUS servers to authenticate (and implicitly authorize) dialup users for PPP service. It would be advantageous to allow the re-use of this same RADIUS infrastructure for SIP client authentication.

Although the currently defined mechanisms for SIP client authentication [section 3.2.2.2 of RFC 2617] and RADIUS authentication using a User-Password or CHAP-Password [sections 5.2, 5.3 of RFC 2138] both use MD5, they run MD5 across differently formatted messages. There are two approaches to solving this problem. One is to extend RADIUS to support HTTP-Digest; the other is to extend the SIP list of authentication schemes to support a CHAP-Password. This document proposes extending the SIP list of authentication schemes to support a CHAP-Password.

2 Definitions

The definitions of several terms used in this document follow:

nonce

A nonce is a octet string that is uniquely generated each time a request is made. It is recommended that a nonce be constructed to exhibit global and temporal uniqueness.

The SIP specification [SIP] calls this a "nonce-value" (Section 3.2.1 of [DIG]).

The RADIUS specification calls this a (random) challenge. (Section 2.2 of [RAD])
In RADIUS, the nonce can be placed in the Request Authenticator (Section 4.2 of [RADIUS]) or in the CHAP-Challenge attribute. (Section 5.40 of [RADIUS])

The CHAP Response in the CHAP-Password and the nonce-value in the HTTP-Digest use a 16-octet nonce.

sequence number

A sequence number is a monotonically increasing integer.
Sequence numbers allow detection of replays.

The "nonce-count" of the HTTP-Digest is a 32-bit sequence number
(formatted as 8 hex digit characters)

The "Chap-ID" in the CHAP-Password is a 1 octet sequence number.

shared secret

A secret shared between two entities.

In this document, we assume that:

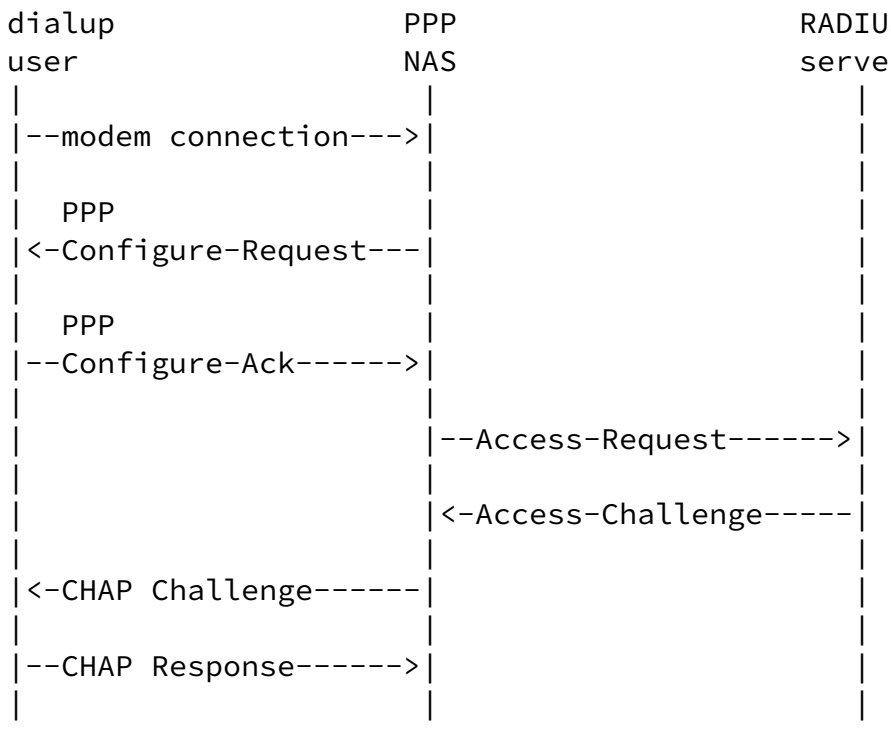
- A user shares a secret (i.e. password) with
  the RADIUS server.

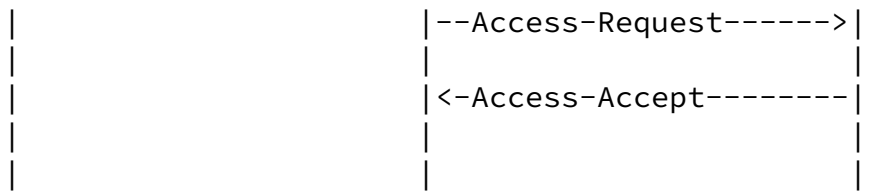
- The PPP NAS (or SIP proxy) also shares a secret
  with the RADIUS server.

3 Analogous Model - PPP

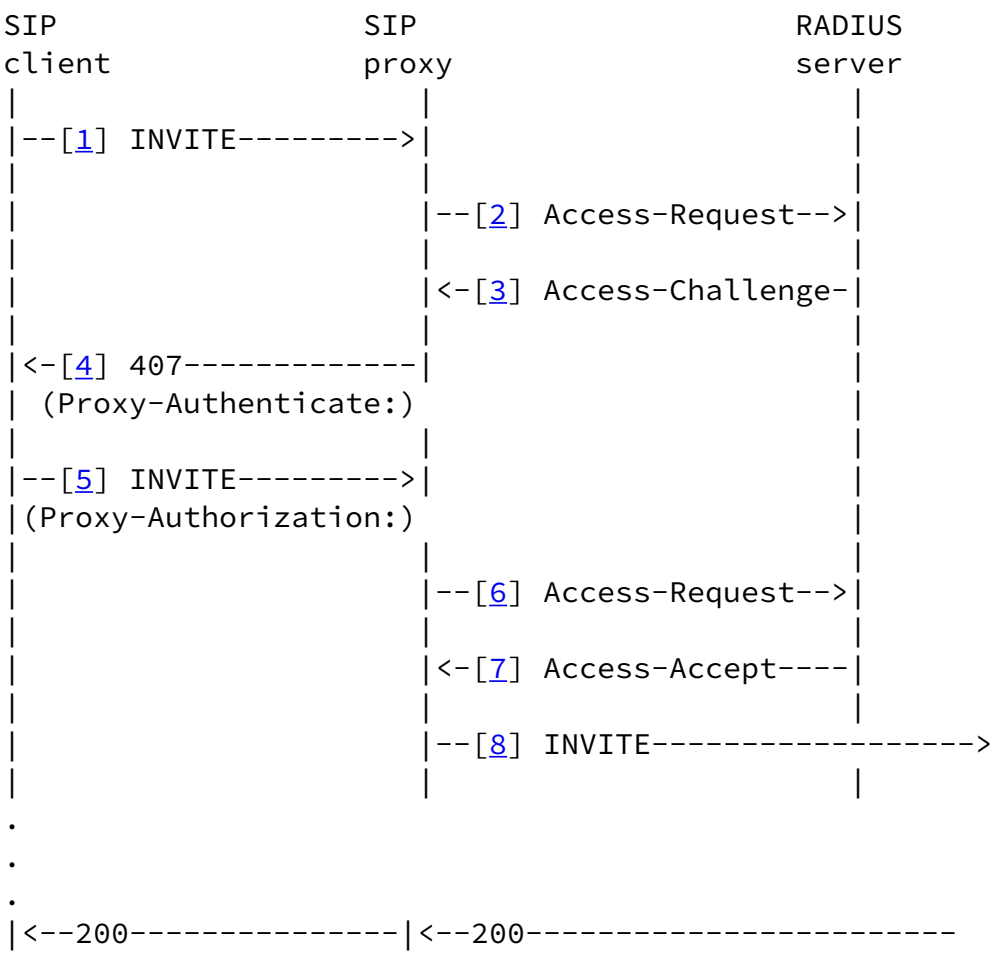
When a SIP proxy is used for user authentication
an analogy can be drawn from PPP message flows.

3.1 Message flow for PPP user authentication with Radius backend:





3.2 Message flow for SIP user authentication with Radius backend:



When a PPP client authentication failure occurs, in some cases the PPP NAS implementation terminates the link. However, other PPP NAS implementations may choose to allow the client to continue, but with a filtered list of services. A PPP NAS may allow traffic which lets the user update his credentials (such as email to the sysadmin).

Similarly, a SIP proxy server may wish to allow the user to place calls to the ISP's home office (to obtain updated credentials). A SIP proxy server may also wish to allow 911 calls to complete.

4 Current PAP/CHAP/SIP/HTTP Authentication mechanisms

The following sections briefly describe the current mechanisms used for user authentication in PPP (PAP/CHAP) and HTTP/SIP (Basic and Digest).

4.1 PAP authentication mechanism

Why not use RADIUS User-Password?

The RADIUS User-Password attribute is calculated as:

User-Password = md5hash(NAS-secret, nonce) XOR user-password

If a PPP user sends his password in cleartext (eg. using PAP), then the PPP server can calculate the User-Password attribute of the Access-Request to authenticate the user.

It is undesirable for a SIP user to send his password in cleartext.

If the user does NOT send his password in cleartext, the User-Password cannot be calculated by either the PPP client (because he doesn't know the NAS-secret) or the NAS (because he doesn't know the user-password).

4.2 CHAP authentication mechanism

PPP defines usage of the CHAP-Password (as an alternative to User-Password) to avoid cleartext transmission of the users's password.

When a CHAP-Password is used a cleartext sequence number, cleartext nonce, and the following MD5 hash are transmitted by the client:

md5hash(seqnum, user-password, nonce)

The nonce can be generated by the client or the server. If the nonce is generated by the client, the server may choose to accept it or may challenge the client with a new nonce.

4.3 SIP/HTTP authentication mechanisms:

SIP and HTTP define two basic authentication mechanisms. HTTP-Basic and HTTP-Digest. Usage of HTTP-Basic involves sending the the user's password in cleartext, and is thus undesirable.

The currently defined mechanisms for SIP client authentication using HTTP-Digest are taken from section 3.2.2.2 of RFC 2617 and the hash constructions are repeated here for clarity:

If the directive's value is "MD5" or is unspecified, then A1 is:

```
A1 = unq(username-value) ":" unq(realm-value) ":" passwd
```

If the directive's value is "MD5-sess", then A1 is

calculated only once - on the first request by the client following receipt of a WWW-Authenticate challenge from the server. It uses the server nonce from the challenge, and the first client nonce value to construct A1 as follows:

```
A1 = H( unq(username-value) ":" unq(realm-value)
        ":" passwd
        ":" unq(nonce-value) ":" unq(cnonce-value))
```

This creates a 'session key' for the authentication of subsequent requests and responses which is different for each "authentication session", thus limiting the amount of material hashed with any one key. [RFC 2617]

5 Interaction/Mapping problems

There are two problems:

5.1 CHAP-Password construction problem:

If a SIP proxy receives an HTTP-Digest from a SIP client (without CHAP-Password support), the SIP proxy is unable to construct a CHAP-Password. This is because the SIP proxy doesn't have access to the client's password.

The SIP proxy only has access to a hash of the client's password, and (as dicussed above) this hash is computed across a message whose format is different than the RADIUS server expects.

Nor can the SIP proxy simply forward the hash calculated in the HTTP-Digest:

5.2 Message mapping problem:

Since the message format over which a hash is computed is different for the CHAP-Password than the message format used for the HTTP-Digest "MD5" or "MD5-sess" algorithms, a RADIUS server could not verify a proxied HTTP-Digest (which uses either the "MD5" or "MD5-sess" algorithms). The RADIUS server would discard such a HTTP-Digest formulated hash as invalid.

Here is the proposed solution to these problems:

6 SIP Authentication using CHAP-Password

To solve these problems, we specify an additional mechanism for SIP authentication which uses a CHAP-Password. CHAP-Password can either be used for endpoint-to-endpoint authentication (when used in WWW-Authenticate and Authorization) or for endpoint-to-proxy authentication (when used in Proxy-Authenticate and Proxy-Authorization).

6.1 The WWW-Authenticate Response Header

When a CHAP-Password is used for SIP authentication, the WWW-Authenticate Response Header (3.2.2 of RFC 2617) is defined as:

```
WWW-Authenticate = "WWW-Authenticate" ":" "CHAP-Password" chap-challenge
chap-challenge   = * (";" chap-params )
chap-params      = chap-username | chap-algorithm | chap-id | nonce
chap-algorithm   = "algorithm" "=" ( "MD5" | token )
chap-username    = quoted-string
chap-id          = "id" "=" + ( digit )
chap-nonce       = "nonce" "=" nonce-value
chap-nonce-value = <"> 32LHEX <">
LHEX             = "0" | "1" | "2" | "3" |
                   "4" | "5" | "6" | "7" |
                   "8" | "9" | "a" | "b" |
                   "c" | "d" | "e" | "f"
```

chap-algorithm: A string indicating the authentication method to be used.

chap-username: A string containing the user name.

chap-id: The CHAP Identifier is a one octet sequence number.

nonce: A string of 32 hex digits. The contents of the nonce are implementation dependent. The quality of the implementation depends on a good choice.

Example:

```
WWW-Authenticate: CHAP-Password ;username="byerly" ;algorithm="MD5"
  ;id=0 ;nonce="10131973aaa511bb05261975aaa505fb"
```

The chap-username is copied from the User-Name attribute of the Access-Challenge message received from the RADIUS server.

The chap-id is copied from the (1-octet) Identifier field of the Access-Challenge message received from the RADIUS server.

The chap-nonce-value is copied from the Access-Challenge message from the RADIUS server (from the CHAP-Challenge attribute if present, otherwise from the Request Authenticator).

6.2 The Authorization Request Header

When challenged, the SIP client is expected to retry the request, passing an Authorization header line, which is defined as follows:

```
Authorization = "Authorization" ":" "CHAP-Password" chap-response-line
chap-response-line   =  * (";" chap-response-params )
chap-response-params = chap-username | chap-id | nonce | chap-response
chap-response        = "response" "=" chap-response-value
chap-response-value  = <"> 32LHEX <">
```

chap-response-value: A string of 32 hex digits computed as defined in Section 4.1 of RFC1994, which proves that the user knows a password.

Example:

```
Authorization: CHAP-Password ;username="byerly" ;id=0
  ;nonce="10131973aaa511bb05261975aaa505fb"
  ;response="f53a66e43c12a383aa65219ec873ce35"
```

The client MUST increment the CSeq header before resubmitting the request.

A server MAY be configured not to generate nonces only if replay attacks are not a concern.

The Response Value (chap-response-value) of the CHAP-Password is computed per Section 4.1 of RFC 1994 [CHAP]. The 16-octet Response Value of the CHAP-Password should be formatted as 32 hex digits and placed in the "chap-response-value" of the Authorization request.

The chap-id should be placed in the (1-octet) Identifier field of the Access-Request message to the RADIUS server. The chap-id should also be placed in the (1-octet) CHAP Identifier field of the CHAP-Password attribute of the Access-Request message to the RADIUS server. (See sections 3, 5.3, [RAD])

The nonce-value SHOULD be placed in the Request Authenticator of the Access-Request message to the RADIUS server.
(See section 3, [RAD])
Alternatively, the nonce-value MAY be placed in a CHAP-Challenge attribute in the Access-Request message to the RADIUS server.
(See section 5.3, [RAD])

The chap-response-value should be placed in the 16-octet String field of the CHAP-Password attribute in the Access-Request message to the RADIUS server. (See section 5.3, [RAD])

7 Proxy-Authenticate and Proxy-Authorization

The CHAP-Password authentication scheme may also be used for authenticating users to proxies.

8 Security Considerations

Security issues are the primary topic of this RFC.

The security issues for this document are the same as those in the Security Considerations sections of RFC 1994 [CHAP] and RFC 2617 [DIG].

9 Further Examples

Only the relevant headers have been included in the following examples.

9.1 User Authentication using backend RADIUS server - With Server Challenge.

[1] SIP Client to SIP proxy server:

```
INVITE sip:+19195551212@cisco.com SIP/2.0
From: sip:+19195551234@domain.com
To: sip:+19195551212@cisco.com
Call-ID: 12345600@cisco.com
CSeq: 1 INVITE
Proxy-Authorization: CHAP-Password
  ;username="byerly"
  ;algorithm="MD5"
  ;id=0
  ;nonce="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
  ;response="bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
Content-Type: application/sdp
```

NOTE: The Proxy-Authorization header sent in the first message may have been cached from a previous exchange with the SIP proxy. If the SIP client does not place a Proxy-Authorization header in the INVITE, the RADIUS server will (transitting through SIP proxy) challenge him with a new nonce.

[2] SIP proxy server to RADIUS server:

```
Code = 1        (Access-Request)
ID = 0
Length = 71
Request Authenticator = {16 octet random number also used as
                         CHAP challenge
                         (aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa)}
```



```
Attributes:
    User-Name = "byerly"
    CHAP-Password = {1 octet CHAP ID (00) followed by 16 octet
```

```
                            CHAP response
                            (bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb)}
            NAS-IP-Address = 192.168.1.16
            NAS-Port = 5
```

[3] RADIUS server to SIP proxy server:

```
     Code = 11       (Access-Challenge}
     ID = 0          (same as in Access-Request)
     Length = 68
     Attributes:
         Reply-Message = "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
         State = {Magic Cookie to be returned along with user's
                  response; in this example 8 octets of data}
```

[4] SIP proxy server to client:

```
     SIP/2.0 407 Proxy Authentication required
     From: sip:+19195551234@domain.com
     To: sip:+19195551212@cisco.com
     Call-ID: 12345600@cisco.com
     CSeq: 1 INVITE
     Proxy-Authenticate: CHAP-Password
       ;username="byerly"
       ;algorithm="MD5"
       ;id=0
       ;nonce="ccccccccccccccccccccccccccccccc"
     State: {Magic Cookie from Access-Challenge packet, unchanged}
            (formatted as hex digits)
```

NOTE: In this instance, the RADIUS server chooses to re-challenge the SIP client with a new nonce.

[5] SIP Client to SIP proxy server:

```
     INVITE sip:+19195551212@cisco.com SIP/2.0
     From: sip:+19195551234@domain.com
     To: sip:+19195551212@cisco.com
     Call-ID: 12345600@cisco.com
     CSeq: 2 INVITE
     Content-Type: application/sdp
     Proxy-Authorization: CHAP-Password
       ;username="byerly"
       ;algorithm="MD5"
       ;id=0
       ;nonce="ccccccccccccccccccccccccccccccc"
       ;response="ddddddddddddddddddddddddddddddd"
     State: {Magic Cookie from Access-Challenge packet, unchanged}
            (formatted as hex digits)
```

[6] SIP proxy server to RADIUS server:

```
Code = 1          (Access-Request)
ID = 1            (Note that this changes)
Length = 71
Request Authenticator = {NEW 16 octet CHAP challenge
                         ()}
Attributes:
    User-Name = "byerly"
    CHAP-Password = {1 octet CHAP ID followed by 16 octet
                     CHAP response
                     (ddddddddddddddddddddddddddddddddd)}
    NAS-IP-Address = 192.168.1.16
    NAS-Port = 5
    State = {Magic Cookie from Access-Challenge packet,
             unchanged}
```

[7] RADIUS server to SIP proxy server:

```
Code = 2          (Access-Accept)
ID = 1            (same as in Access-Request)
Length = 30
```

[8] SIP proxy server to next hop UAS:

```
INVITE sip:+19195551212@cisco.com SIP/2.0
From: sip:+19195551234@domain.com
To: sip:+19195551212@cisco.com
Call-ID: 12345600@cisco.com
CSeq: 2 INVITE
Content-Type: application/sdp
```

9.2 User Authentication using backend RADIUS server - Without Server Challenge.

[a] SIP Client to SIP proxy server:

```
INVITE sip:+19195551212@cisco.com SIP/2.0
From: sip:+19195551234@domain.com
To: sip:+19195551212@cisco.com
Call-ID: 12345601@cisco.com
CSeq: 3 INVITE
Proxy-Authorization: CHAP-Password
  ;username="byerly"
  ;algorithm="MD5"
  ;id=0
  ;nonce="eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee"
```

```
        ;response="ffffffffffffffffffffffffffffffff"
      Content-Type: application/sdp
```


[b] SIP proxy server to RADIUS server:

```
     Code = 1        (Access-Request)
     ID = 0
     Length = 71
     Request Authenticator = {16 octet random number also used as
                              CHAP challenge
                              (eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee)}
     Attributes:
         User-Name = "byerly"
         CHAP-Password = {1 octet CHAP ID (00) followed by 16 octet
                          CHAP response
                          (ffffffffffffffffffffffffffffffff)}
         NAS-IP-Address = 192.168.1.16
         NAS-Port = 5
```

[c] RADIUS server to SIP proxy server:

```
     Code = 2        (Access-Accept)
     ID = 1          (same as in Access-Request)
     Length = 56
```

[d] SIP proxy server to next hop UAS:

```
     INVITE sip:+19195551212@cisco.com SIP/2.0
     From: sip:+19195551234@domain.com
     To: sip:+19195551212@cisco.com
     Call-ID: 12345601@cisco.com
     CSeq: 3 INVITE
     Content-Type: application/sdp
```

There are two cases when a SIP client could pre-send a Proxy-Authorization that the RADIUS server might accept:

1) The RADIUS server originally generated the nonce when challenging the SIP client on a previous call. The SIP client is reusing the previously sucessful Authorization for a new call.
2) The SIP client originally generated the nonce. The parsed format of the nonce is known to both the SIP client and the RADIUS server. The nonce contains a timestamp which the RADIUS server can extract and use to limit the replay window. Since a RADIUS server silently discards

invalid/unauthorized requests, this scheme is not subject to the form of the man-in-the-middle attack where Mallory sends a bogus request to the server and uses the response to make the client believe she is a legitimate server.

To dos:
1) Fix the RADIUS Lengths to be correct
2) Calculate real MD5 hashes

Outstanding issues:
1) Do we/how do we support the RADIUS State: attribute? What are the implications for collision with (DCS-)State: object?
2) Do we reuse the RADIUS NAS-Port and NAS-Port-Type attributes to allow the RADIUS server to have media port control over calls? (eg. using UDP port numbers for RTP streams)

10 Acknowledgements

We would like to thank Roger Levesque, David Oran, Mike Thomas, David Daiker, Shail Bhatnagar, and Denise Caballero-McCann for discussions on the need for and improvements to this draft. We would also like to thank Tyrone Floryanzia for his insights on H.323 gateway/gatekeeper call authorization using RADIUS.

11 References

[SIP] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, SIP: Session Initiation Protocol", RFC 2543, March 1999.

[RAD] C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote Authentication Dial in User Service (RADIUS)," RFC 2138, April 1997.

[DIG] Franks, J, et al. "HTTP Authentication: Basic and Digest Access Authentication," RFC 2617, June 1999.

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

[PAP] Lloyd, B, W. Simpson. "PPP Authentication Protocols", RFC 1334, October 1992.

[PPP] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD

51, RFC 1661, DayDreamer, July 1994.

[MD5] Rivest, R., and S. Dusse, "The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, Inc., RFC 1321, April 1992.

[REQ] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," RFC-2119, March 1997.

Authors' Addresses

Bryan J. Byerly
Cisco Systems
7025 Kit Creek Road
P.O. Box 14987
Research Triangle Park, NC 27709
USA
Email: byerly@cisco.com

David Williams
Cisco Systems
7025 Kit Creek Road
P.O. Box 14987
Research Triangle Park, NC 27709
USA
Email: dwilli@cisco.com