

JUNOS RADIUS Authentication

Stephen Gill
E-mail: gillsr@cymru.com
Revision: 1.1, 11/19/2001

Contents

Credits	2
Introduction.....	2
RADIUS Protocol.....	2
Architecture	3
Interoperability	5
JUNOS Configuration	5
Funk Steel-Belted Radius Configuration.....	8
Sniffer Trace and Decode.....	10
Conclusion.....	13
References	14

Credits

- Funk Tech Support – provided the Juniper dictionary and helpful troubleshooting hints
- Juniper TAC – opened PR [19548] to correct the “service-type authenticate-only” attribute in JUNOS RADIUS request packets.

Introduction

This paper provides a detailed account on how to configure RADIUS authentication and authorization on a Juniper router (client) in conjunction with Funk’s Steel-Belted Radius (server). It also provides a basic review of the RADIUS protocol before presenting specific details on architecture, implementation, and packet level troubleshooting.

The steps required to properly configure a well defined authentication architecture are not inherently obvious and configuring server and client authentication can seem like a daunting task. This paper demonstrates just how easy it is to understand and configure.

RADIUS Protocol

Before delving into the configuration of both client and server, a review of the RADIUS protocol is necessary to set the stage for what we are trying to accomplish. Remote Authentication Dial-In User Service, or RADIUS, is a standard used for centralizing network authentication of remote access users. RADIUS was originally developed by Livingston Enterprises and has been subsequently documented in RFCs 2865 [1] and 2866 [2].

RADIUS is a client-server authentication and authorization access protocol used to authenticate users attempting to connect to a network device. The network device (in our case, a Juniper router) functions as a client, passing

user information to one or more RADIUS servers. User access is either granted or denied to the device based on the response received from one or more RADIUS servers.

RADIUS includes three components: an authentication server, client protocols, and an accounting server. The RADIUS authentication server is installed on a central computer on the network. The client protocols run on remote access devices such as remote access routers and firewalls. These RADIUS clients send UDP authentication requests, typically over port 1812, with MD5 encrypted passwords to the RADIUS authentication server and act on responses sent back by the server. RADIUS accounting captures statistics about sessions that are established to the network, and typically operates over UDP port 1813.

Architecture

Several server implementations exist that fully support the RADIUS protocol and its standards. However, many RADIUS clients implement their own vendor attributes into the protocol and it is difficult to find a product that will interoperate with all of them. Vendor attributes are simply extensions to the RADIUS protocol whereby third parties can add additional functionality and meaning specific to their products.

Funk wisely incorporates support for several vendor products and their attributes into Steel-Belted Radius, and packages it into an easy to use, flexible, and highly scalable server. Moreover, Steel-Belted Radius is in widespread use in many ISPs as it is targeted at market segments dominated by remote access operators, carriers, and Network Service Providers. For this reason, the author has chosen to devote attention to interoperability between this Steel-Belted Radius and Juniper routers. More information about Steel-Belted Radius and other solutions offered by Funk can be found at: <http://www.funk.com/>. The reader may wish to explore other well known and widely used solutions such as FreeRADIUS available free of charge at: <http://www.freeradius.org/>.

Regardless of RADIUS server platform, the topology described herein remains unchanged. Authentication traffic is best suited traversing a secured network. Confidential information such as user data, and accounting are kept private, and certain design flaws in the RADIUS protocol are avoided. Numerous attacks have been uncovered for RADIUS authentication traffic that traverses an unsecured network, including issues with user password protection, response authenticators, and a suboptimal encryption cipher [4]. Among the two primary AAA protocols in use to date, RADIUS and TACACS+, the latter is generally considered more secure because the entire contents of the packet are encrypted. TACACS+ is not without its own flaws though, such as accounting vulnerabilities and session-id limitations which can lead to compromised encryption [5]. Fortunately, a

newer protocol named “Diameter Base Protocol” is currently in development by the IETF to provide a more secure AAA framework. [6]

Figure 1 presents a simple visual overview of the network topology assumed in this document along with the steps involved in establishing a RADIUS authenticated connection with a Juniper Router.

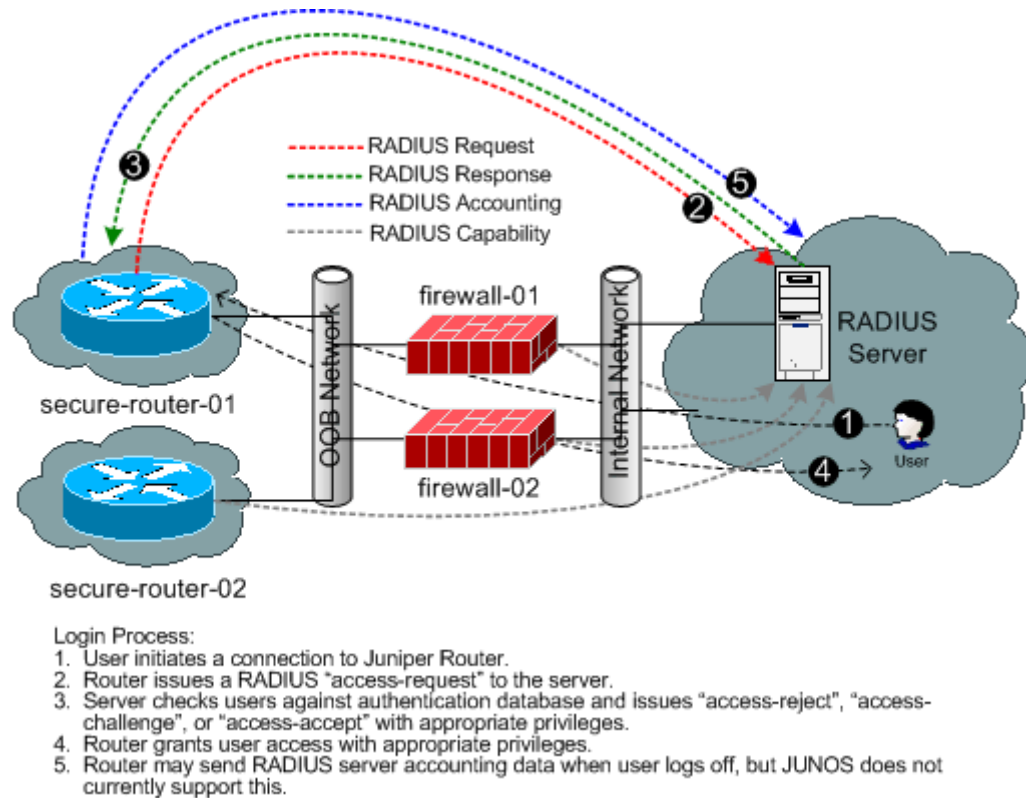


Figure 1 - RADIUS Network Topology

The authentication process begins in step one (1), when a user initiates a connection to the Juniper Router. If the flow is not permitted by filters on the firewall or router, then it should be silently discarded and the user will not be allowed to connect. If the flow is permitted by the filters, then the router will in turn issue an “access-request” to the RADIUS server in step two (2). If the RADIUS server is not reachable within a specified number of retries and time interval, then the router should check its local authentication database as a fallback mechanism. If the RADIUS server is reachable, then it will check its authentication database and issue either an “reject”, “challenge”, or “accept” message along with any attributes and values it has been configured to return. In response to an “accept” message, the router will grant the user access combining returned RADIUS attributes with its local authorization information. A reject message will cause the router to query its own authentication database if configured to do so. Finally, when the user

terminates the connection in step five (5), the router may send session accounting data back to the RADIUS server for reporting.

Currently JUNOS does not support RADIUS accounting. However, network administrators that require user accounting information may wish to deploy syslog as a viable alternative by monitoring “auth.info” entries. Through the use of careful scripting, syslog user accounting messages could potentially be summarized in RADIUS accounting packets and forwarded to the RADIUS server for record-keeping.

Interoperability

The configurations included in this document have been fully tested with JUNOS 5.0R2.4 and Funk Steel-Belted Radius EE v. 2.27. JUNOS 5.0R2.4 RADIUS authentication was also been tested against Funk Steel Belted Radius v. 3.0 with limited success. Though users can be authenticated, authorization between these two platforms does not work properly because in version 3.0 Funk updated the way they handle “access-request” packets with “service-type authenticate-only” attributes. In the interest of being fully RFC compliant, packets that are received with this attribute should not send back authorization information. Details surrounding the use and purpose of the “service-type authenticate-only” attribute can be found in RFC 2138 [1], section 5.6. For more information on the attribute sent from the Juniper router in an authentication request, refer to Sniffer Trace and Decode.

Since JUNOS sends the “service-type authenticate-only” attribute in all authentication requests without providing a mechanism for changing it, authorization information is not returned from Steel-Belted Radius 3.0. This keeps the vendor attributes of “Juniper-Local-User-Name”, “Juniper-Allow-Commands”, and “Juniper-Deny-Commands”, from being returned at all. These attributes are used on the server to control the local user template as well as what commands a particular user is allowed to enter on the router. A problem record was opened with Juniper’s TAC on this issue, and it has been formally documented under PR [19548]. No information was yet available as of the time of this writing on how soon this bug would be fixed. A different attribute will likely be required such as “service-type login” (or none at all) in the correction of this bug. In the meantime, platforms, such as Steel-Belted Radius v. 2.27, that ignore this attribute in an “access-request” packet should interoperate adequately.

JUNOS Configuration

The steps required to configure RADIUS authentication on the Juniper side are quite minimal. Configuration steps include defining the address of the RADIUS server, selecting a shared secret, and determining system authentication order. Each shared secret should be unique per RADIUS client in order to increase device security and minimize exposure. It is also

generally recommended to leave system password authentication enabled as a backup in case the RADIUS server is not reachable. Other parameters that may be modified are the radius-server timeout, and retry interval. The following configuration excerpt summarizes these steps:

```
[edit system]
/* Use local password authentication if RADIUS fails */
authentication-order [ radius password ];
/* Enable RADIUS authentication. */
radius-server {
  10.0.0.100 {
    /* Shared secret between client and server */
    secret "<UNIQUE-PASSWORD>"; # SECRET-DATA
    /* Wait 5 seconds until timeout */
    timeout 5;
  }
}
```

Next, the Juniper router requires one or more local user templates from which to map the RADIUS users. By default, users that are authenticated through RADIUS are mapped to a local user template named “remote”. However, multiple local user templates can be configured such that a “remote” user template is not required. User templates serve as a basis for common user settings such as idle-timeouts, class permissions, and authorization levels.

Attributes that are configured for a particular username on the RADIUS server are encoded and returned in an “access-accept” packet. On the Juniper side, a username may be mapped to one of several templates (other than “remote”) based on the “Juniper-Local-User-Name” attribute that has been returned. These user templates are then joined with their corresponding classes to provide the basic permission structure and timeout settings. Figure 2 displays the process flow of the attributes that are received from the server, which in turn are mapped to users and classes on the client.

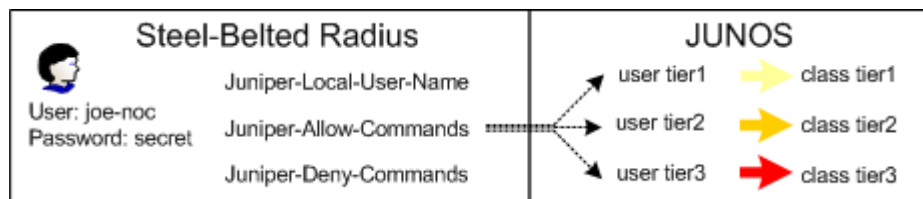


Figure 2 - RADIUS AAA Username Architecture

Below we have defined three login classes on which to base our RADIUS user accounts. Each class maintains a specific set of user privileges that allows for a different set of priorities based on functional requirements. Tier1 operators have been assigned read-only privileges; tier2 operators have received a controlled subset of read-write permissions; tier3 administrators have been assigned unlimited read-write access. Though

each architecture and implementation will vary, the three major classes provided here should serve as an excellent starting point for privilege differentiation.

```
[edit system login]
/* Provides basic read-only privileges */
class tier1 {
    idle-timeout 15;
    permissions [ configure interface network routing snmp system
                 trace view firewall ];
}
/* Provides a controlled subset of read-write privileges */
class tier2 {
    idle-timeout 15;
    permissions [ admin clear configure interface interface-
                 control network reset routing routing-control
                 snmp snmp-control system system-control trace
                 trace-control view maintenance firewall
                 firewall-control secret rollback ];
}
/* Provides unlimited access */
class tier3 {
    idle-timeout 15;
    permissions all;
}
```

Additional command authorization granularity is possible through the use of the “allow-commands” and “deny-commands” statements configured at the “class” level. However, we will use the authorization capability of the RADIUS server to return these attributes as opposed to implementing them on the router. This reduces the amount of configuration required on the router, and allows for a more centralized, and manageable system of user authorization.

Once the appropriate classes and authority levels have been configured, usernames must be mapped to their corresponding classes. Local router usernames serve as templates by which remote users are authorized through the RADIUS server. Specific usernames do not require configuration on the router. Instead, templates are established into which users will be transposed. Additionally, passwords should not be configured locally unless local authentication is desired such as in a backup scenario. In this case, a fallback password has been configured locally for user “tier3” which has full access privileges. Only this user will be allowed to log in locally if connectivity to the RADIUS server is lost. Following is the router configuration for the local user templates:

```
/* RADIUS template tier1 user */
user tier1 {
    uid 2001;
    class tier1;
}
/* RADIUS template tier2 user */
```

```

user tier2 {
    uid 2002;
    class tier2;
}
/* RADIUS template tier3 user */
user tier3 {
    uid 2003;
    class tier3;
    /* Backdoor password in case RADIUS server is unreachable */
    authentication {
        encrypted-password "<PASSWORD>"; # SECRET-DATA
    }
}

```

It is important to note the password for user “tier3” will still be checked even if an “access-reject” message is returned from the RADIUS server for a user mapped to this template. This means that user “tier3” will always be able to log in locally on the router, despite any outside RADIUS server configuration. This password should be maintained and updated in the same manner as all the user accounts on the RADIUS server.

Funk Steel-Belted Radius Configuration

The default Steel-Belted Radius installation requires a few modifications before it is capable of returning vendor-specific Juniper attributes in an “access-accept” packet. First, one must place the custom dictionary below into a file named “radius/service/Juniper.dct”.

```

... snip snip ...
#####
# Juniper.dct - Juniper Dictionary File
#
# This dictionary contains Juniper Vendor Specific Attributes
#
# (See README.DCT for more details on the format of this file)
#####
# Use the Radius specification attributes
#
@radius.dct

#
# Define Juniper Vendor Specific Attributes
#
MACRO Juniper-VSA(t,s) 26 [vid=2636 type1=%t% len1=+2 data=%s%]

#
# Description of Juniper Attributes
#

ATTRIBUTE Juniper-Local-User-Name          Juniper-VSA(1, string) r
ATTRIBUTE Juniper-Allow-Commands           Juniper-VSA(2, string) r
ATTRIBUTE Juniper-Deny-Commands            Juniper-VSA(3, string) r

#####
# Juniper.dct - Juniper Dictionary File
#####
... snip snip ...

```

Next, the file named “radius/service/vendor.ini” should be edited to include the following text:

```
... snip snip ...
vendor-product      = Juniper
dictionary          = Juniper
ignore-ports       = no
port-number-usage  = per-port-type
help-id            = 2008
... snip snip ...
```

Finally, the file “radius/service/dictiona.dcm” should be edited to include the following line:

```
@juniper.dct
```

Once the above procedures have been followed, the RADIUS service should be restarted to recognize the recent changes. At this point, a new RAS client model will be available named “Juniper” in the Steel-Belted Radius admin GUI. Additionally, the Juniper vendor-specific attributes will be available in the “Return List Attributes” list under a particular user.

Configuration of the Steel-Belted Radius server admin GUI is fairly straightforward and requires very few steps. The following information is necessary to fully define a Juniper RAS Client:

```
RAS Clients
Client Name: M10
IP Address: [Juniper IP]
Make/Model: Juniper
Authentication Secret: [Unique Shared Secret]
```

Once the RAS Client has been defined, user information should be entered including the password, and all the attributes that should be returned to the client. This is where usernames can be mapped to specific templates, and where particular commands can be allowed or disallowed on a per-user basis. Below we have defined only two users, both of which are mapped to the “tier1” template on the Juniper side. A generic account, “tier1”, and user account, “joe-noc”, both map to the same template.

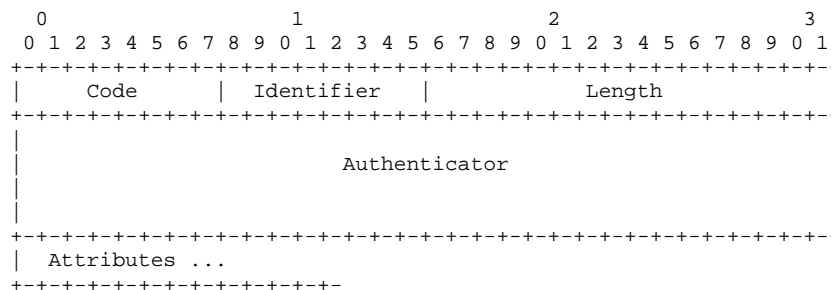
```
Users
User Name: joe-noc
Password: [User Password]
Return List Attributes:
    Juniper-Local-User-Name      tier1
    Juniper-Allow-Commands      [regular expression]
    Juniper-Deny-Commands       [regular expression]

User Name: tier1
Password: [User Password]
Return List Attributes:
    Juniper-Local-User-Name      tier1
    Juniper-Allow-Commands      [regular expression]
    Juniper-Deny-Commands       [regular expression]
```

For specific guidelines for configuring the authorization regular expression, please refer to the JUNOS Getting Started manual. The expression “^set | ^show” would allow all commands that begin with “set” or “show” despite permissions that have been configured on the Juniper router. Attributes entered in the “allow” or “deny” commands take precedence over the rights assigned to the template on the router by default for increased authorization granularity.

Sniffer Trace and Decode

The attributes that are returned from a RADIUS server show up in clear text after the packet header. According to RFC 2865 [1], a RADIUS packet should look like the following:



The code field is one byte in size and defines the type of the RADIUS packet. Packets received with invalid codes are silently discarded. Following is a list of code numbers and their associated packet types:

Code	Type
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge
12	Status-Server (experimental)
13	Status-Client (experimental)
255	Reserved

After the code field comes the identifier byte - a random number that is used to associate an authentication request with a reply. The length field defines the size of the RADIUS packet, followed by the sixteen (16) byte Authenticator, or shared secret. Finally, a list of attributes required for the particular type of service along with any other optional properties are included after the shared secret.

Based on a sniffer trace taken from a RADIUS authentication session between a Juniper router and Steel-Belted Radius, we can see that packets of code 1 (access-request) and 2 (access-accept) have been used. Based on the packet codes, we can determine that the RADIUS server has

accepted the client's authentication request. Following is a brief packet trace of the authentication request and subsequent response:

NO.	SRC	DST	PROTOCOL	DESCRIPTION
1	10.0.0.50:1047	10.0.0.100:1812	UDP	RADIUS Request
2	10.0.0.100:1812	10.0.0.50:1047	UDP	RADIUS Response

As stated earlier, the RADIUS protocol uses UDP port 1812 for authentication. More detailed information about the trace above can be determined by investigating the contents of each of the packets. Funk provides a unique way of examining the packet contents through the use of detailed logging. By editing the "radius/service/Radius.ini" and increasing the logging level from 0 to 2, the contents of all RADIUS packets both inbound and outbound will be provided in the log files.

```
[Configuration]
LogLevel          = 2
TraceLevel       = 2
```

Both packets from the trace above were included in the server log file after increasing the logging level to 2. Packet contents were subsequently dissected manually, byte by byte, to walk the reader through exactly what happens during an authentication request. The contents of both packets and corresponding byte descriptions are included below.

```
[NO. 1]-----
Packet Data
000: 01c80038 29b231fd 27f13467 2f76d3cf |...8).1.'.4g/v..|
010: 0dce9e7e 01077469 65723102 125b1226 |...~..tier1...[.&|
020: e9d0d684 87fe870e d4e12ebe 8720056d |..... .m|
030: 31300606 00000008 |10.....|

Packet Decode:
X-byte Value Description
01 01 access request
02 c8 unique identifier (0xc8 = 200)
03-04 0038 length (0x38 = 56)
05-20 29b231fd 27f13467 2f76d3cf [shared secret]
0dce9e7e
21 01 attribute 01: User-Name
22 07 attribute length
23-27 7469 657231 tier1
28 02 attribute 02: User-Password
29 12 attribute length: 0x12 = 18
30-45 5b1226 e9d0d684 87fe870e [user password]
d4e12ebe 87
46 20 attribute 20: Nas-Identifier
47 05 attribute length
48-50 6d 3130 m10
51 06 attribute 06: Service-Type
52 06 attribute length
53-56 00000008 Authenticate-only
```

Note that the client, or Juniper router in this case, has sent a "service-type authenticate-only" attribute. This attribute should not be sent from the Juniper router by default, as it tells the server it does not need to send back authorization information. Section 5.6 of RFC 2865 [1] describes the "service-type" attribute in more detail. Fortunately Steel-Belted Radius 2.27 ignores this statement and still sends back the vendor attributes as witnessed below.

```
[NO. 2]-----
Packet Data:
000: 02c80056 3f58f0be ba685407 e5c76b23 |...V?X...hT...k#|
010: 05330018 191b5342 522d434c 20444e3d |.3...SBR-CL DN=|
020: 22544945 52312220 41543d22 3022001a |"TIER1" AT="0"...|
030: 0e00000a 4c010874 69657231 001a0c00 |...L..tier1...|
040: 000a4c03 06736574 001a0d00 000a4c02 |..L..set.....L.|
050: 0773686f 7700 |.show.
```

```
Packet Decode:
X-byte Value Description
01 02 access response
02 c8 unique identifier (0xc8 = 200)
03-04 0056 length (0x56 = 86)
05-20 3f58f0be ba685407 e5c76b23 [shared secret]
05330018
21 19 attribute 19: class
22 1b attribute length 0x1b = 27
23-47 5342 522d434c 20444e3d SBR-CL DN="TIER1" AT="0"
22544945 52312220 41543d22
302200
48 1a attribute: Vendor-Specific
49 0e attribute length: 0x0e = 14
50-53 00000a 4c vendor-id: 2636
54 01 vendor-attribute: Juniper-Local-User-Name
55 08 attribute length
56-61 74 69657231 00 tier1
62 1a attribute: Vendor-Specific
63 0c attribute length 0x0c = 12
64-67 00000a4c vendor-id: 2636
68 03 vendor-attribute: Juniper-Deny-Commands
69 06 attribute length
70-73 736574 00 set
74 1a attribute: Vendor-Specific
75 0d attribute length 0x0d = 13
76-79 00000a4c vendor-id: 2636
80 02 vendor-attribute: Juniper-Allow-Commands
81 07 attribute length
82-86 73686f 7700 show
```

Unfortunately “traceoptions” are not available to troubleshoot RADIUS connectivity issues on a Juniper router. However, another method of gathering the same packet level information is to perform a sniffer trace using the “monitor traffic” command on the router itself. The syntax required to trace router RADIUS traffic over the fxp0 interface is marked in **bold** below.

Step 1: Trace the packets on the router.

```
tier3@m10> monitor traffic matching "port 1812||1813" no-domain-names no-resolve
print-ascii size 200 count 5 extensive
Listening on fxp0
```

Step 2: Establish a connection to the router.

```
telnet m10

m10 (ttypl)

Login: tier1
Password: password
Last login: Wed Nov 7 16:14:30 from 10.0.0.100

--- JUNOS 5.0R2.4 built 2001-09-25 02:34:13 UTC
```

```
tier1@m10>
```

Step 3: Hit ^C when finished, and interpret the output on the router.

```
16:16:10.954545 Out 10.0.0.50.1046 > 10.0.0.100.1812: udp 56 (ttl 64, id 25680)
0x0000 4500 0054 6450 0000 4011 01b4 0a00 0032 E..TdP..@.....2
0x0010 0a00 0064 0416 0714 0040 dc23 0165 0038 ...d.....@#.e.8
0x0020 d6c9 e616 010a 5d05 d7f0 ccaa 1236 f7ed .....].....6..
0x0030 0107 7469 6572 3102 12a3 9e36 e919 6c21 ..tierl....6..l!
0x0040 d75a a5c3 8745 1f14 c620 056d 3130 0606 .Z...E.....m10..
0x0050 0000 0008 ....
16:16:10.957610 In 10.0.0.100.1812 > 10.0.0.50.1046: udp 86 (ttl 128, id 35059)
0x0000 4500 0072 88f3 0000 8011 9cf2 0a00 0064 E..r.....d
0x0010 0a00 0032 0714 0416 005e a6cf 0265 0056 ...2.....^...e.V
0x0020 cf35 d43b 9061 380f 2605 2620 dea2 5e19 .5.;a8.&.&...^
0x0030 191b 5342 522d 434c 2044 4e3d 2254 4945 ..SBR-CL.DN="TIE
0x0040 5231 2220 4154 3d22 3022 001a 0e00 000a R1".AT="0".....
0x0050 4c01 0874 6965 7231 001a 0c00 000a 4c03 L..tierl.....L.
0x0060 0673 6574 001a 0d00 000a 4c02 0773 686f .set.....L..sho
0x0070 7700 w.
^C
132 packets received by filter
0 packets dropped by kernel
```

Note that data above differs slightly from the contents gathered from the Steel-Belted Radius log files because the sniffer trace includes the entire packet starting at the Ethernet layer.

Conclusion

From packet sniffing, to architecture review, this paper presented all of the information required to configure and troubleshoot RADIUS authentication and authorization between a Juniper router and Funk Steel-Belted Radius. By following the steps outlined in this paper, the reader should be able to fully implement and troubleshoot a Juniper and Funk RADIUS solution with ease.

References

- [1] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
<http://www.ietf.org/rfc/rfc2865.txt>

- [2] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
<http://www.ietf.org/rfc/rfc2866.txt>

- [3] Rigney, C., Willats, W. and P. Calhoun, "RADIUS Extensions", RFC 2869, June 2000.
<http://www.ietf.org/rfc/rfc2869.txt>

- [4] Hill, Joshua. "An Analysis of the RADIUS Authentication Protocol", November 2001.
<http://www.untruth.org/~josh/security/radius/radius-auth.html>

- [5] Designer, Solar. "An Analysis of the TACACS+ Protocol and its Implementations", July 2000.
http://www.openwall.com/advisories/OW-001-tac_plus.txt

- [6] Calhoun, P., Akhtar, H., Arkko, J., Guttman, E., Rubens, A. and G. Zorn. "Diameter Base Protocol", July 2001.
<http://www.ietf.org/internet-drafts/draft-ietf-aaa-diameter-07.txt>