INTRODUCTION

RADIUS, TACACS, and DIAMETER are classified as authentication, authorization, and accounting (AAA) servers. The Internet Engineering Task Force (IETF) chartered an AAA Working Group in 1998 to develop the authentication, authorization, and accounting requirements for network access. The goal was to produce a base protocol that supported a number of different network access models, including traditional dial-in network access servers (NAS), Mobile-IP, and roaming operations (ROAMOPS). The group was to build upon the work of existing access providers like Livingston Enterprises.

Livingston Enterprises originally developed RADIUS (Remote Authentication Dial-in User Service) for their line of network access servers (NAS) to assist timeshare and Internet service providers with billing information consolidation and connection configuration. Livingston based RADIUS on the IETF distributed security model and actively promoted it through the IETF Network Access Server Requirements Working Group in the early 1990s. The client/server design was created to be open and extensible so it could be easily adapted to work with other third-party products. At this writing, RADIUS, TACACS, and DIAMETER are classified as authentication, authorization, and accounting (AAA) servers. The Internet Engineering Task Force (IETF) chartered an AAA Working Group in 1998 to develop the authentication, authorization, and accounting requirements for network access. The goal was to produce a base protocol that supported a number of different network access models, including traditional dial-in network access servers (NAS), Mobile-IP, and roaming operations (ROAMOPS). The group was to build upon the work of existing access providers like Livingston Enterprises.

Livingston Enterprises originally developed RADIUS (Remote Authentication Dial-in User Service) for their line of network access servers (NAS) to assist timeshare and Internet service providers with billing information consolidation and connection configuration. Livingston based RADIUS on the IETF distributed security model and actively promoted it through the IETF Network Access Server Requirements Working Group in the early 1990s. The client/server design was created to be open and extensible so it could be easily adapted to work with other third-party products. At this writing, RADIUS, TACACS, and DIAMETER are classified as authentication, authorization, and accounting (AAA) servers. The Internet Engineering Task Force (IETF) chartered an AAA Working Group in 1998 to develop the authentication, authorization, and accounting requirements for network access. The goal was to produce a base protocol that supported a number of different network access models, including traditional dial-in network access servers (NAS), Mobile-IP, and roaming operations (ROAMOPS). The group was to build upon the work of existing access providers like Livingston Enterprises.

Got the telecommuter, mobile workforce, VPN, multi-platform, dial-in user authentication blues? Need a centralized method for controlling and auditing external accesses to your network? Then RADIUS, TACACS, or DIAMETER may be just what you have been looking for. Flexible, inexpensive, and easy to implement, these centralized authentication protocols improve remote access security and reduce the time and effort required to manage Remote Access Server (RAS) clients.
US version 2 was a proposed IETF standard managed by the RADIUS Working Group.

The origin of the Terminal Access Controller Access Control System (TACACS) daemon used in the early days of ARPANET is unknown. Cisco Systems adopted the protocol to support AAA services on its products in the early 1990s. Cisco extended the protocol to enhance security and support additional types of authentication requests and response codes. They named the new protocol TACACS+. The current version of the TACACS specification is a proposed IETF Standard (RFC 1492) managed by the Network Working Group. It was developed with the assistance of Cisco Systems.

Pat Calhoun (Sun Laboratories) and Allan Rubens (Ascend Communications) proposed the DIAMETER AAA framework as a draft standard to the IETF in 1998. The name DIAMETER is not an acronym but rather a play on the RADIUS name. DIAMETER was designed from the ground up to support roaming applications and to overcoming the extension limitations of the RADIUS and TACACS protocols. It provides the base protocols required to support any number of AAA extensions, including NAS, Mobile-IP, host, application, and Web-based requirements. At this writing, DIAMETER consisted of eight IETF draft proposals, authored by twelve different contributors from Sun, Microsoft, Cisco, Nortel, and others. Pat Calhoun continues to coordinate the DIAMETER effort.

**AAA 101: KEY FEATURES OF AN AAA SERVICE**

The key features of a centralized AAA service include (1) a distributed (client/server) security model, (2) authenticated transactions, (3) flexible authentication mechanisms, and (4) an extensible protocol. Distributed security separates the authentication process from the communications process, making it possible to consolidate user authentication information into a single centralized database. The network access devices (i.e., a NAS) are the clients. They pass user information to an AAA server and act upon the response(s) the server returns. The servers receive user connection requests, authenticate the user, and return to the client NAS the configuration information required to deliver services to the user. The returned information may include transport and protocol parameters, additional authentication requirements (i.e., callback, SecureID), authorization directives (i.e., services allowed, filters to apply), and accounting requirements (Exhibit 1).

Transmissions between the client and server are authenticated to ensure the integrity of the transactions. Sensitive information (e.g., passwords) is encrypted using a shared secret key to ensure confidentiality and prevent passwords and other authentication information from being monitored or captured during transmission. This is particularly important when the data travels across public carrier (e.g., WAN) links.
AAA servers can support a variety of authentication mechanisms. This flexibility is a key AAA feature. User access can be authenticated using PAP (Password Authentication Protocol), CHAP (Challenge Handshake Authentication Protocol), the standard UNIX login process, or the server can act as a proxy and forward the authentication to other mechanisms like a Microsoft domain controller, a Novell NDS server, or a SecureID ACE server. Some AAA server implementations use additional mechanisms like calling number identification (caller ID) and callback to further secure connections.

Because technology changes so rapidly, AAA servers are designed with extensible protocols. RADIUS, DIAMETER, and TACACS use variable-length attribute values designed to support any number of new parameters without disturbing existing implementations of the protocol. DIAMETER's framework approach provides additional extensibility by standardizing a transport mechanism (framework) that can support any number of customized AAA modules.

From a management perspective, AAA servers provide some significant advantages:

- reduced user setup and maintenance times because users are maintained on a single host
- fewer configuration errors because formats are similar across multiple access devices
- less security administrator training requirements because there is only one system syntax to learn
- better auditing because all login and authentication requests come through a single system
- reduced help desk calls because the user interface is consistent across all access methods

**EXHIBIT 1 — Key Features of a Centralized AAA Service**
• quicker proliferation of access information because information only needs to be replicated to a limited number of AAA servers
• enhanced security support through the use of additional authentication mechanisms (i.e., SecureID)
• extensible design makes it easy to add new devices without disturbing existing configurations

RADIUS: REMOTE AUTHENTICATION DIAL-IN USER SERVICE
RADIUS is by far the most popular AAA service in use today. Its popularity can be attributed to Livingston's decision to open the distribution of the RADIUS source code. Users were quick to port the service across multiple platforms and add customized features, many of which Livingston incorporated as standard features in later releases. Today, versions of the RADIUS server are available for every major operating system from both freeware and commercial sources, and the RADIUS client comes standard on NAS products from every major vendor.

A basic RADIUS server implementation references two configuration files. The client configuration file contains the address of the client and the shared secret used to authenticate transactions. The user file contains the user identification and authentication information (e.g., user ID and password) as well as connection and authorization parameters. Parameters are passed between the client and server using a simple five-field format encapsulated into a single UDP packet. The brevity of the format and the efficiency of the UDP protocol (no connection overhead) allow the server to handle large volumes of requests efficiently. However, the format and protocol also have a downside. They do not lend themselves well to some of today’s diverse access requirements (i.e., ROAMOPS), and retransmissions are a problem in heavy load or failed node scenarios.

Putting the AA in RADIUS: Authentications and Authorizations
RADIUS has eight standard transaction types: access-request, access-accept, access-reject, accounting-request, accounting-response, access-challenge, status-server, and status-client. Authentication is accomplished by decrypting a NAS access-request packet, authenticating the NAS source, and validating the access-request parameters against the user file. The server then returns one of three authentication responses: access-accept, access-reject, or access-challenge. The latter is a request for additional authentication information such as a one-time password from a token or a callback identifier.

Authorization is not a separate function in the RADIUS protocol but simply part of an authentication reply. When a RADIUS server validates an access request, it returns to the NAS client all the connection attributes specified in the user file. These usually include the data link (i.e., PPP, SLIP) and network (i.e., TCP/IP, IPX) specifications, but may also include
vendor-specific authorization parameters. One such mechanism automatically initiates a Telnet or rlogin session to a specified host. Other methods include forcing the port to a specific IP address with limited connectivity, or applying a routing filter to the access port.

The Third A: Well, Sometimes Anyway!
Accounting is a separate function in RADIUS and not all clients implement it. If the NAS client is configured to use RADIUS accounting, it will generate an Accounting-Start packet once the user has been authenticated, and an Accounting-Stop packet when the user disconnects. The Accounting-Start packet describes the type of service the NAS is delivering, the port being used, and user being serviced. The Accounting-Stop packet duplicates the Start packet information and adds session information such as elapsed time, bytes inputs and outputs, disconnect reason, etc.

Forward Thinking and Other Gee-whiz Capabilities
A RADIUS server can act as a proxy for client requests, forwarding them to servers in other authentication domains. Forwarding can be based on a number of criteria, including a named or number domain. This is particularly useful when a single modem pool is shared across departments or organizations. Entities are not required to share authentication data; each can maintain their own RADIUS server and service proxied requests from the server at the modem pool. RADIUS can proxy both authentication and accounting requests. The relationship between proxies can be distributed (one-to-many) or hierarchical (many-to-one), and requests can be forwarded multiple times. For example, in Exhibit 2, it is perfectly permissible for the “master” server to forward a request to the user’s regional server for processing.

Most RADIUS clients have the ability to query a secondary RADIUS server for redundancy purposes, although this is not required. The advantage is continued access when the primary server is offline. The disadvantage is the increase in administration required to synchronize data between the servers.

Most RADIUS servers have a built-in database connectivity component. This allows accounting records to be written directly into a database for billing and reporting purposes. This is preferable to processing a flat text accounting “detail” file. Some server implementations also include database access for authentication purposes. Novell’s implementation queries NDS, NT versions query the PDC, and several vendors are working on LDAP connectivity.

It Does Not Get Any Easier than This. Or Does It?
When implementing RADIUS, it is important to remember that the source code is both open and extensible. The way each AAA, proxy, and data-
base function is implemented varies considerably from vendor to vendor. When planning a RADIUS implementation, it is best to define one’s functional requirements first and then choose NAS components and server software that support them. Here are a few factors to consider:

- **What accesses need to be authenticated?** External accesses via modem pools and VPN servers are essential, but internal accesses to critical systems and security control devices (i.e., routers, firewalls) should also be considered.

- **What protocols need to be supported?** RADIUS can return configuration information at the data link, network, and transport levels. Vendor documentation as well as the RADIUS RFCs and standard dictionary file are good sources of information for evaluating these parameters.

- **What services are required?** Some RADIUS implementations require support for services like Telnet, rlogin, and third-party authentication (i.e., SecureID), which often require additional components and expertise to implement.

- **Is proxy or redundancy required?** When NAS devices are shared across management or security domains, proxy servers are usually required and it is necessary to determine the proxy relationships in advance. Redundancy for system reliability and accessibility is also an important consideration because not all clients implement this feature.
Other considerations might include:

- authorization, accounting, and database access requirements
- interfaces to authentication information in NDS, X.500, or PDC databases
- the RADIUS capabilities of existing clients
- support for third-party Mobile-IP providers like iPass
- secure connection support (i.e., L2TP, PPTP)

Client setup for RADIUS is straightforward. The client must be configured with the IP address of the server(s), the shared secret (encryption key), and the IP port numbers of the authentication and accounting services (the defaults are 1645 and 1646, respectively). Additional settings may be required by the vendor.

The RADIUS server setup consists of the server software installation and three configuration files:

1. The dictionary file is composed of a series of Attribute/Value pairs the server uses to parse requests and generate responses. The standard dictionary file supplied with most server software contains the attributes and values found in the RADIUS RFCs. One may need to add vendor-specific attributes, depending upon one’s NAS selection. If any modifications are made, double-check that none of the attribute Names or Values are duplicated.

2. The client file is a flat text file containing the information the server requires to authenticate RADIUS clients. The format is the client name or IP address, followed by the shared secret. If names are used, the server must be configured for name resolution (i.e., DNS). Requirements for the length and format of the shared secret vary, but most UNIX implementations are eight characters or less. There is no limitation on the number of clients a server can support.

3. The user file is also a flat text file. It stores authentication and authorization information for all RADIUS users. To be authenticated, a user must have a profile consisting of three parts: the username, a list of authentication _check items_, and a list of _reply items_. A typical entry would look like the one displayed in Exhibit 3. The first line contains the user's name and a list of check items separated by commas. In this example, John is restricted to using one NAS device (the one at 10.100.1.1). The remaining lines contain reply items. Reply items are separated by commas at the end of each line. String values are put in quotes. The final line in this example contains an authorization parameter that applies a packet filter to this user's access.

The check and reply items contained in the user file are as diverse as the implementations, but a couple of conventions are fairly common.
Username prefixes are commonly used for proxy requests. For example, usernames with the prefix CS/ would be forwarded to the computer science RADIUS server for authentication. Username suffixes are commonly used to designate different access types. For example, a user name with a %vpn suffix would indicate that this access was via a virtual private network (VPN). This makes it possible for a single RADIUS server to authenticate users for multiple NAS devices or provide different reply values for different types of accesses on the same NAS.

The DEFAULT user parameter is commonly used to pass authentication to another process. If the username is not found in the user file, the DEFAULT user parameters are used to transfer the validation to another mechanism. On UNIX, this is typically the /etc/passwd file. On NT, it can be the local user database or a domain controller. Using secondary authentication mechanisms has the advantage of expanding the check items RADIUS can use. For example, UNIX and NT groups can be checked as well as account activation and date and time restriction.
Implementations that use a common NAS type or one server for each NAS type have fairly uncomplicated user files, but user file contents can quickly become quite convoluted when NAS devices and access methods are mixed. This not only adds complexity to the management of the server, but also requires more sophistication on the part of users.

Stumbling Blocks, Complexities, and Other RADIUS Limitations
RADIUS works well for remote access authentication but is not suitable for host or application authentication. Web servers may be the first exception. Adding a RADIUS client to a Web server provides a secure method for authenticating users across open networks. RADIUS provides only basic accounting facilities with no facilities for monitoring nailed-up circuits or system events. User-based rather than device-based connection parameters are another major limitation of RADIUS. When a single RADIUS server manages several different types of NAS devices, user administration is considerably more complex. Standard RADIUS authentication does not provide facilities for checking a user’s group membership, restricting access by date or time of day, or expiring a user’s account on a given date. To provide these capabilities, the RADIUS server must be associated with a secondary authentication service.

Overall, RADIUS is an efficient, flexible, and well-supported AAA service that works best when associated with a secondary authentication service like NDS or NT where additional account restrictions can be applied. The adoption of RADIUS version 2 as an IETF standard will certainly ensure its continued success and importance as a good general-purpose authentication, authorization, and accounting service.

TACACS: TERMINAL ACCESS CONTROLLER ACCESS CONTROL SYSTEM
What is commonly referred to today as TACACS actually represents two evolutions of the protocol. The original TACACS, developed in the early ARPANet days, had very limited functionality and used the UDP transport. In the early 1990s, the protocol was extended to include additional functionality and the transport changed to TCP. To maintain backward compatibility, the original functions were included as subsets of the extended functions. The new protocol was dubbed XTACACS (Extended TACACS). Virtually all current TACACS daemons are based on the extended protocol as described in RFC1492.

Cisco Systems adopted TACACS for its AAA architecture and further enhanced the product by separating the authentication, authorization, and accounting functions and adding encryption to all NAS-server transmissions. Cisco also improved the extensibility of TACACS by permitting arbitrary length and content parameters for authentication exchanges. Cisco called their version TACACS+, but in reality, TACACS+ bares no
resemblance to the original TACACS and packet formats are not backward compatible. Some server implementations support both formats for compatibility purposes. The remainder of this section is based on TACACS+ because it is the proposed IETF standard.

TACACS+ servers use a single configuration file to control server options, define users and attribute/value (AV) pairs, and control authentication and authorization actions. The options section specifies the settings of the service’s operation parameters, the shared secret key, and the accounting file name. The remainder of the file is a series of user and group definitions used to control authentication and authorization actions. The format is “user = username” or “group = groupname,” followed by one or more AV pairs inside curly brackets.

The client initiates a TCP session and passes a series of AV pairs to the server using a standard header format followed by a variable length parameter field. The header contains the service request type (authentication, authorization, or accounting) and is sent in the clear. The entire parameter field is encrypted for confidentiality. TACACS’ variable parameter field provides for extensibility and site-specific customization, while the TCP protocol ensures reliable delivery. However, the format and protocol also increase communications overhead, which can impact the server’s performance under heavy load.

A 1: TACACS Authentication

TACACS authentication has three packet types: Start, Continue, and Reply. The client begins the authentication with a Start packet that describes the type of authentication to be performed. For simple authentication types like PAP, the packet may also contain the user ID and password. The server responds with a Reply. Additional information, if required, is passed with client Continue and server Reply packets. Transactions include login (by privilege level) and password change using various authentication protocols (i.e., CHAP, PAP, PPP, etc.). Like RADIUS, a successful TACACS authentication returns attribute-value (AV) pairs for connection configuration. These can include authorization parameters or they can be fetched separately.

A 2: TACACS Authorization

Authorization functions in TACACS consist of Request and Response AV pairs used to:

- permit or deny certain commands, addresses, services or protocols
- set user privilege level
- invoke input and output packet filters
- set access control lists (ACLs)
• invoke callback actions
• assign a specific network address

Functions can be returned as part of an authentication transaction or an authorization-specific request.

**A 3: TACACS Accounting**

TACACS accounting functions use a format similar to authorization functions. Accounting functions include Start, Stop, More, and Watchdog. The Watchdog function is used to validate TCP sessions when data is not sent for extended periods of time. In addition to the standard accounting data supported by RADIUS, TACACS has an event logging capability that can record system level changes in access rights or privilege. The reason for the event as well as the traffic totals associated with it can also be logged.

**Take Another Look (and Other Cool Capabilities)**

TACACS authentication and authorization processes are considerably enhanced by two special capabilities: recursive lookup and callout. Recursive lookup allows connection, authentication, and authorization information to be spread across multiple entries. AV pairs are first looked up in the user entry. Unresolved pairs are then looked up in the group entry (if the user is a member of a group) and finally assigned the default value (if one is specified). TACACS+ permits groups to be embedded in other groups, so recursive lookups can be configured to encompass any number of connection requirements. TACACS+ also supports a callout capability that permits the execution of user-supplied programs. Callout can be used to dynamically alter the authentication and authorization processes to accommodate any number of requirements; a considerably more versatile approach than RADIUS’ static configurations. Callout can be used to interface TACACS+ with third-party authentication mechanisms (i.e., Kerberos and SecureID), pull parameters from a directory or database, or write audit and accounting records.

TACACS, like RADIUS, can be configured to use redundant servers and because TACACS uses a reliable transport (TCP), it also has the ability to detect failed nodes. Unlike RADIUS, TACACS cannot be configured to proxy NAS requests, which limits its usefulness in large-scale and cross-domain applications.

**Cisco, Cisco, Cisco: Implementing TACACS**

There are a number of TACACS server implementations available, including two freeware versions for UNIX, a Netware port, and two commercial versions for NT, but the client implementations are Cisco, Cisco, Cisco. Cisco freely distributes the TACACS and TACACS+ source code, so fea-
Features and functionality vary considerably from one implementation to another. CiscoSecure is generally considered the most robust of the commercial implementations and even supports RADIUS functions. Once again, be sure to define functional requirements before selecting NAS components and server software. If your shop is Cisco-centric, TACACS is going to work well; if not, one might want to consider a server product with both RADIUS and TACACS capabilities.

Client setup for TACACS on Cisco devices requires an understanding of Cisco's AAA implementation. The AAA function must be enabled for any of the TACACS configuration commands to work. The client must be configured with the IP address of the server(s) and the shared secret encryption key. A typical configuration would look like this:

```bash
aaa new-model
tacacs-server key <your key here>
tacacs-server host <your primary TACACS server IP address here>
tacacs-server host <your secondary TACACS server IP address here>
```

followed by port-specific configurations. Different versions of Cisco IOS support different TACACS settings. Other NAS vendors support a limited subset of TACACS+ commands.

TACACS server setup consists of the server software installation and editing the options, authentication, and authorization entries in the configuration files. Comments may be placed anywhere in the file using a pound sign (#) to start the line. In the following example, Jane represents a dial-in support contractor, Bill a user with multiple access methods, and Dick an IT staff member with special NAS access.

```bash
# The default authentication method will use the local UNIX
# password file, default authorization will be permitted for
# users without explicit entries and accounting records will be
# written to the /var/adm/tacacs file.
default authentication = file /etc/passwd
default authorization = permit
accounting file = /var/adm/tacacs

# Contractors, vendors, etc.
user = jane {
    name = "Jane Smith"
global = cleartext "Jane'sPassword"
expires = "May 10 2000"
service=ppp
protocol=ip {
    addr=10.200.10.64
    inaccl=101
    outaccl=102
}
}
```

# Employees with “special” requirements

user = bill {
    name="Bill Jones"
    arap = cleartext "Apple_ARAP_Password"
    pap = cleartext "PC_PAP_Password"
    default service = permit
}

user = dick {
    name="Dick Brown"
    member = itstaff
    default service = permit
    # Permit Dick to access the exec command using connection access list 4
    service = exec {
        acl = 4
    }
    # Permit Dick to use the telnet command to everywhere but 10.101.10.1
    cmd = telnet {
        deny 10\101\10\1
        permit .*
    }
}

# Standard Employees use these entries

user = DEFAULT {
    service = ppp {
        # Disconnect if idle for 5 minutes
        idletime = 5
        # Set maximum connect time to one hour
        timeout = 60
    }
    protocol = ip {
        addr-pool=hqnas
    }
}

# Group Entries

group = itstaff {
    # Staff uses a special password file
    login = file /etc/itstaff_passwds
}

Jane’s entry sets her password to “Jane’sPassword” for all authentication types, requires her to use PPP, forces her to a known IP, and applies
both inbound and outbound extended IP access control lists (a.k.a. IP filters). It also contains an account expiration date so the account can be easily enabled and disabled. Bill's entry establishes different passwords for Apple and PAP logins, and assigns his connection the default service parameters. Dick's entry grants him access to the NAS executive commands, including Telnet, but restricts their use by applying a standard IP access control list and an explicit deny to the host at 10.101.10.1. Bill and Dick's entries also demonstrate TACACS' recursive lookup feature. The server first looks at user entry for a password, then checks for a group entry. Bill is not a member of any group, so the default authentication method is applied. Dick, however, is a member of "itstaff," so the server validates the group name and looks for a password in the group entry. It finds the login entry and authenticates Dick using the /etc/itstaff_passwds file. The default user entry contains AV pairs specifying the use of PPP with an idle timeout of five minutes and a maximum session time of one hour.

In this example, the UNIX /etc/password and /etc/group files are used for authentication, but the use of other mechanisms is possible. Novell implementations use NDS, NT versions use the domain controller, and CiscoSecure support LDAP and several SQL compatible databases.

Proxyless, Problems, and Pitfalls: TACACS limitations
The principle limitation of TACACS+ may well be its lack of use. While TACACS+ is a versatile and robust protocol, it has few server implementations and even fewer NAS implementations. Outside of Cisco, this author was unable to find any custom extensions to the protocol or any vendor-specific AV pairs. Additionally, TACACS' scalability and performance are an issue. Unlike RADIUS' single-packet UDP design, TACACS uses multiple queries over TCP to establish connections, thus incurring overhead that can severely impact performance. TACACS+ servers have no ability to proxy requests so they cannot be configured in a hierarchy to support authentication across multiple domains. CiscoSecure scalability relies on regional servers and database replication to scale across multiple domains. While viable, the approach assumes a single management domain, which may not always be the case.

Overall, TACACS+ is a reliable and highly extensible protocol with existing support for Cisco's implementation of NAS-based VPNs. Its "outcalls" capability provides a fairly straightforward way to customize the AAA functions and add support for third-party products. Although TACACS+ supports more authentication parameters than RADIUS, it still works best when associated with a secondary authentication service like NDS or an NT domain. The adoption of TACACS+ as an IETF standard and its easy extensibility should improve its adoption by other NAS manufactures. Until then, TACACS+ remains a solid AAA solution for Cisco-centric environments.
DIAMETER: TWICE RADIUS?

DIAMETER is a highly extensible AAA framework capable of supporting any number of authentication, authorization, or accounting schemes and connection types. The protocol is divided into two distinct parts: the Base Protocol and the Extensions. The DIAMETER Base Protocol defines the message format, transport, error reporting, and security services used by all DIAMETER extensions. DIAMETER Extensions are modules designed to conduct specific types of authentication, authorization, or accounting transactions (i.e., NAS, Mobile-IP, ROAMOPS, and EAP). The current IETF draft contains definitions for NAS requests, Mobile-IP, secure proxy, strong security, and accounting, but any number of other extensions are possible.

DIAMETER is built upon the RADIUS protocol but has been augmented to overcome inherent RADIUS limitations. Although the two protocols do not share a common data unit (PDU), there are sufficient similarities to make the migration from RADIUS to DIAMETER easier. DIAMETER, like RADIUS, uses a UDP transport but in a peer-to-peer rather than client/server configuration. This allows servers to initiate requests and handle transmission errors locally. DIAMETER uses reliable transport extensions to reduce retransmissions, improve failed node detection, and reduce node congestion. These enhancements reduce latency and significantly improve server performance in high-density NAS and hierarchical proxy configurations. Additional improvements include:

- full support for roaming
- cross-domain, broker-based authentication
- full support for the Extensible Authentication Protocol (EAP)
- vendor-defined attributes-value pairs (AVPs) and commands
- enhanced security functionality with replay attack protections and confidentiality for individual AVPs

There Is Nothing Like a Good Foundation

The DIAMETER Base Protocol consists of a fixed-length (96 byte) header and two or more attribute-value pairs (AVPs). The header contains the message type, option flags, version number, and message length, followed by three transport reliability parameters (see Exhibit 4).
AVPs are the key to DIAMETER’s extensibility. They carry all DIAMETER commands, connection parameters, and authentication, authorization, accounting, and security data. AVPs consist of a fixed-length header and a variable-length data field. A single DIAMETER message can carry any number of AVPs, up to the maximum UDP packet size of 8192 bytes. Two AVPs in each DIAMETER message are mandatory. They contain the message Command Code and the sender’s IP address or host name. The message type or the Extension in use defines the remaining AVPs. DIAMETER reserves the first header byte and the first 256 AVPs for RADIUS backward compatibility.

A is for the Way You Authenticate Me

The specifics of a DIAMETER authentication transaction are governed by the Extension in use, but they all follow a similar pattern. The client (i.e., a NAS) issues an authentication request to the server containing the AA-Request Command, a session-ID, and the client’s address and host name followed by the user’s name and password and a state value.

The session-ID uniquely identifies this connection and overcomes the problem in RADIUS with duplicate connection identifiers in high-density installations. Each connection has its own unique session with the server. The session is maintained for the duration of the connection and all transactions related to the connection use the same session-ID. The state AVP is used to track the state of multiple transaction authentication schemes such as CHAP or SecureID.

The server validates the user’s credentials and returns an AA-Answer packet containing either a Failed-AVP or the accompanying Result-Code AVP or the authorized AVPs for the service being provided (i.e., PPP parameters, IP parameters, routing parameters, etc.). If the server is not the HOME server for this user, it will forward (proxy) the request.

Proxy on Steroids!

DIAMETER supports multiple proxy configurations, including the two RADIUS models and two additional Broker models. In the hierarchical model, the DIAMETER server forwards the request directly to the user’s HOME server using a session-based connection. This approach provides several advantages over the standard RADIUS implementation. Because the proxy connection is managed separately from the client connection, failed node and packet retransmissions are handled more efficiently and the hop can be secured with enhanced security like IPSec. Under RADIUS the first server in the authentication chain must know the CHAP shared secret, but DIAMETER’s proxy scheme permits the authentication to take place at the HOME server. As robust as DIAMETER’s hierarchical model is, it still is not suitable for many roaming applications.
DIAMETER uses a Broker proxy server to support roaming across multiple management domains. Brokers are employed to reduce the amount of configuration information that needs to be shared between ISPs within a roaming consortium. The Broker provides a simple message routing function. In DIAMETER, two routing functions are provided: either the Broker forwards the message to the HOME server or provides the keys and certificates required for the proxy server to communicate directly with the HOME server (see Exhibit 5).

A Two Brute: DIAMETER Authorization
Authorization transactions can be combined with authentication requests or conducted separately. The specifics of the transaction are governed by the Extension in use but follow the same pattern and use the same commands as authentications. Authorization requests must take place over an existing session; they cannot be used to initiate sessions but they can be forwarded using a DIAMETER proxy.

Accounting for Everything
DIAMETER significantly improves upon the accounting capabilities of RADIUS and TACACS+ by adding event monitoring, periodic reporting, real-time record transfer, and support for the ROAMOPS Accounting Data Interchange Format (ADIF). DIAMETER accounting is authorization-server directed. Instructions regarding how the client is to generate accounting records is passed to the client as part of the authorization process. Additionally, DIAMETER accounting servers can force a client to send current accounting data. This is particularly useful for connection troubleshooting or to capture accounting data when an accounting server experiences a crash. Client writes and server polls are fully supported by both DIAMETER proxy models.

For efficiency, records are normally batch transferred but for applications like ROAMOPS where credit limit checks or fraud detection are required, records can be generated in real-time. DIAMETER improves upon standard connect and disconnect accounting with a periodic reporting

<table>
<thead>
<tr>
<th>EXHIBIT 5 — A Typical Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
</tr>
<tr>
<td>john</td>
</tr>
<tr>
<td>Service-Type = Framed-User,</td>
</tr>
<tr>
<td>Framed-Protocol = PPP,</td>
</tr>
<tr>
<td>Framed-IP-Address = 10.200.10.1,</td>
</tr>
<tr>
<td>Framed-IP-Netmask = 255.255.255.0,</td>
</tr>
<tr>
<td>Filter-Id = “firewall”</td>
</tr>
</tbody>
</table>
capability that is particularly useful for monitoring usage on nailed-up circuits. DIAMETER also has an event accounting capability like TACACS+ that is useful for recording service-related events like failed nodes and server reboots.

**Security, Standards, and Other Sexy Stuff**

Support for strong security is a standard part of the DIAMETER Base Protocol. Many applications, like ROAMOPS and Mobile-IP, require sensitive connection information to be transferred across multiple domains. Hop-by-hop security is inadequate for these applications because data is subject to exposure at each interim hop. DIAMETER’s Strong Proxy Extension overcomes the problem by encrypting sensitive data in S/MIME objects and encapsulating them in standard AVPs.

Got the telecommuter, mobile workforce, VPN, multi-platform, dial-in user authentication blues? One does not need to! AAA server solutions like RADIUS, TACACS, and DIAMETER can chase those blues away. With a little careful planning and a few hours of configuration, one can increase security, reduce administration time, and consolidate one’s remote access venues into a single, centralized, flexible, and scalable solution. That should put a smile on one’s face.

**Additional Reading**


Bill Stackpole, CISSP, is a senior security consultant with Olympic Resource Management in Poulsbo, Washington.