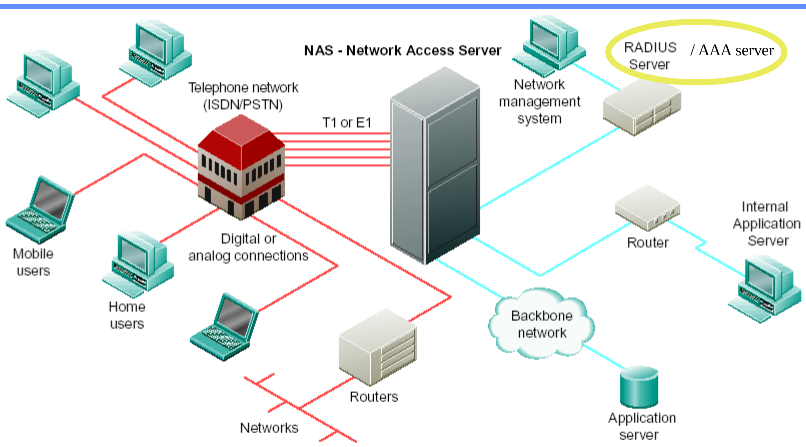
AAA – Authentication, Authorization, and Accounting

Ibéria Medeiros

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa

1

Motivation



- In large scale settings, where client authentication has to be performed at various places, it is more practical to use backend authentication servers
- Eventually, it also made sense to include authorization and accounting operations

3

Authentication

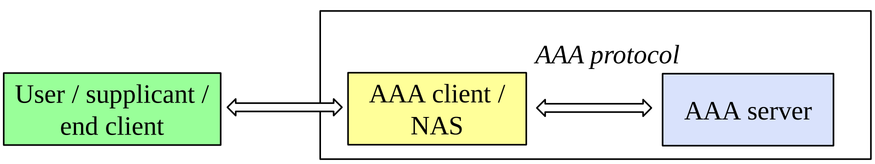
- We interested in two forms of authentication
  - identity verification
  - information authenticity and integrity

- Client authentication – user vs. device authentication, are they the same?
- Message authentication – usually together with data integrity
- Mutual vs. unilateral authentication

Models for Message Authentication

- Two-Party Authentication Model
  - the two parties communicate directly and authenticate each other

- Three-Party Authentication Model
  - employed in large networks with many users
  - allows most authentication operations to be moved from low-cost point-of-presence devices to centralized/specialized servers
  - AAA protocols carry out the authentication credentials



NAS – Network Access Server

Authorization

- Authorization determines if a particular privilege can be granted to the presenter of a certain *credential*
  - the privilege many times is the right to access a resource or service
  - the presenter is either a user or a device
- After authentication, why do we need authorization?
  - unnecessary in several scenarios if all users have the same access rights
  - necessary if certain users have access to differentiated services **or** if there is a interval of time associated with the use of a resource
  - sometimes only authorization is important, and not so much the authentication

Accounting

- Accounting is concerned with collection of information on resource consumption at all or at specific parts of the network
- Some of the applications supported by accounting
  - *auditing*: act of verifying an invoice or the conformance to usage policy, security guidelines, etc
  - *cost allocation*: understand the cost structure associated with some task
  - *trend analysis*: forecast future usage for capacity planning
- Each application is processed by a different logic management entity
- Accounting protocols are used to carry out the collected data to the applications
  - these protocols might have different requirements in terms of reliability and security depending on the specific application

EXAMPLE STANDARDIZED AUTHENTICATION PROTOCOLS

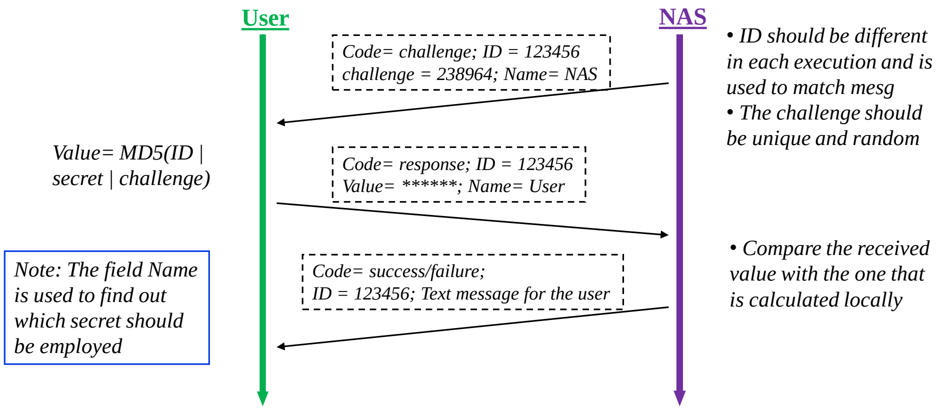
Some (Not So) Legacy …

- PAP – *Password Authentication Protocol*
  – it is the basic protocol where the password is sent in the clear

Description	1 byte	1 byte	2 bytes	1 byte	Variable	1 byte	Variable
Authentication-request	Code = 1	ID	Length	Username length	Username	Password length	Password
Authentication-ack	Code = 2	ID	Length	Message length	Username		
Authentication-nak	Code = 3	ID	Length	Message length	Username		

CHAP - *Challenge Handshake Authentication Protocol*

- CHAP is a challenge-based protocol, where the user is expected to be able to perform a calculation based on its password, on a random value transmitted by the NAS (Network Access Server)



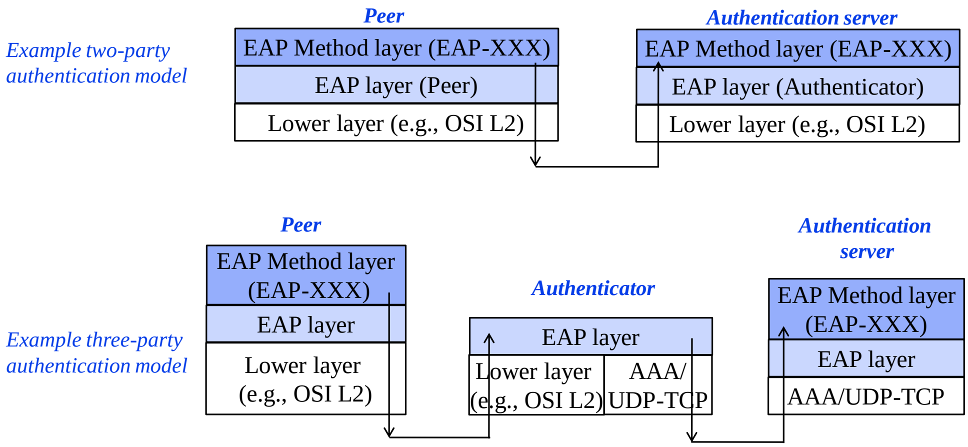
EAP – *Extensible Authentication Protocol*

- EAP was created to
  - authentication framework that can support various methods of authentication
  - extensible to accommodate future authentication mechanisms
  - peers first carry out a negotiation phase followed by the authentication

  Potential problems :
  1. downgrade attacks on the authentication method;
  2. performance penalty due to multiple layers of negotiations

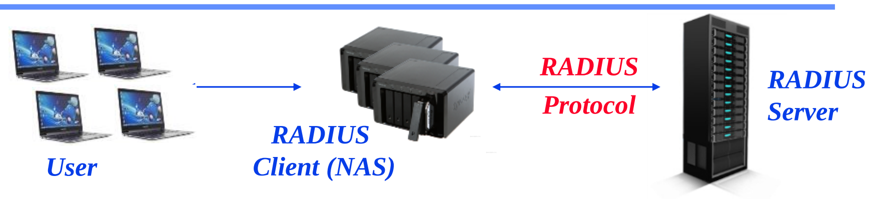
Main Building Blocks of EAP

- EAP has support for two-party and three-party authentication models (EAP-XXX corresponds to some XXX authentication method built on top of EAP)



RADIUS : REMOTE ACCESS DIAL-IN USER SERVICE

Main Characteristics

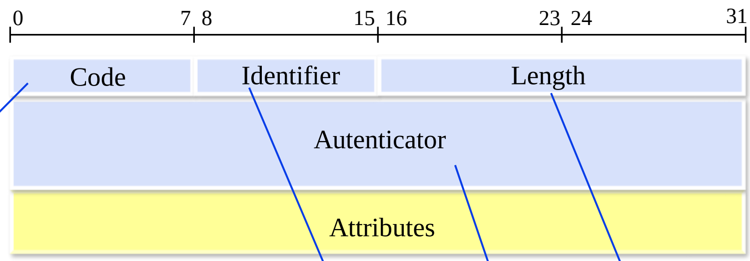


- RADIUS is a client-server mechanism where
  - the **client** offers some service to the **user**
  - the client passes user information to the **server** in the form of requests (e.g., authentication) and returns the responses from the server
  - the server is responsible for authenticating the users (supports several methods like PAP or CHAP) and for providing configuration information
  - the **RADIUS protocol** is used between the client and the server, and it protects the interactions using a **shared secret**
  - after the connection is established, the client collects the resource usage data and reports it back to the server

RADIUS Messages

- ***Access request*** : C → S, contains the user data
- ***Access challenge*** : S → C, is used to question the user or request some sort of negotiation
- ***Access accept*** : S → C, successful completion of a request
- ***Access reject*** : S → C, indicate the rejection of a request
- ***Accounting request*** : C → accounting S, passing accounting information about the service provided to the user
- ***Accounting response*** : accounting S → C, acknowledges the reception of the accounting data
- ***Status-Server and Status-Client*** : experimental

Message Format



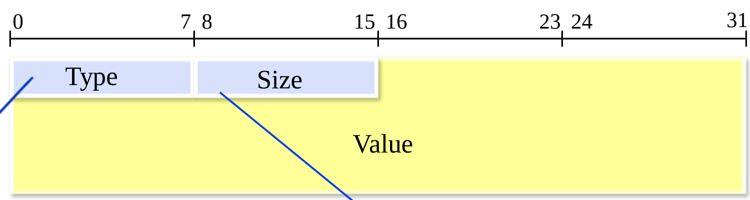
Code	Packet type
1	***access-request***
2	***access-accept***
3	***access-reject***
4	accounting-request
5	accounting-response
11	***access-challenge***
12	status-server (experimental)
13	status-client (experimental)
255	Reserved

To match requests and responses, and to detect duplicates

Number of bytes in the packet (includes code, identifier, ...)

Packet authentication (16 bytes)

Attributes



ID	Example attributes
1	User-Name
2	User-Password
3	CHAP-Password
4	NAS-IP-Address
5	NAS-Port
32	NAS-Identifier
60	CHAP-Challenge
61	NAS-Port-Type
62	Port-Limit

Number of bytes in the attribute (includes the type, size and value)

Attributes
- *support the extensability of RADIUS*
- *more than one attribute can be sent in a mesg*
- *there are vendor specific attributes*

The Authenticator Field

- When used in a ***access-request***, it contains a random number that should not be repeated during the lifetime of the secret shared between the client and server (except if the message is being retransmitted)
- To "improve" security
  - the server should check the IP address of the sender (it should come from the correct NAS)
  - if necessary, a method based on the authenticator can be used to hide the passwords (see next slides)
  - more recently, people have started employing a "Message Authenticator" attribute to protect the messages

    authent_value = MD5(code | identifier| length | ***authenticator*** *| attributes | c_s_secret)*
  - for the specific Accounting Requests messages, the following value should be placed in authenticator field

    authenticator = MD5(code | identifier| length | ***16 zero bytes*** *| attributes | c_s_secret)*

The Authenticator Field (cont.)

- When used in a response like ***access-accept/reject/challenge***, it serves to authenticate the decision/data transmitted by the server. The value is calculated using:

  authenticator_response = MD5(code | identifier| length |
  authenticator_request *| attributes | c_s_secret)*

  Note:
  - » the *identifier* in the response is equal to the request
  - » the *authenticator_request* is the random value placed in the authenticator field of the request

Protecting Attribute Data

- In RADIUS, most confidential data is transmitted in the attributes
  - with PAP authentication, the authenticator can be utilized to hide the password transmitted by the user (placed in the User-Password attribute)

    $User\text{-}Password\_value = MD5(c\_s\_secret \mid authenticator) \oplus u\_password$

  Notes on potential problems:
  - security is highly dependent on the capabilities of the client (e.g., NAS) to generate good random authenticators
  - since *c_s_secret* protects passwords of all users, there is the attack
    - » the attacker sends his password and listens to the request message from the client
    - » the attacker obtains *MD5(c_s_secret | authenticator)*
    - » the client repeats the authenticator and the attacker listens to the network, then he can obtain other people passwords
  - more recently, a *salt* value has been added (*MD5(c_s_secret | authenticator | salt))* which is transmitted in the clear

Mode of Operation with PAP

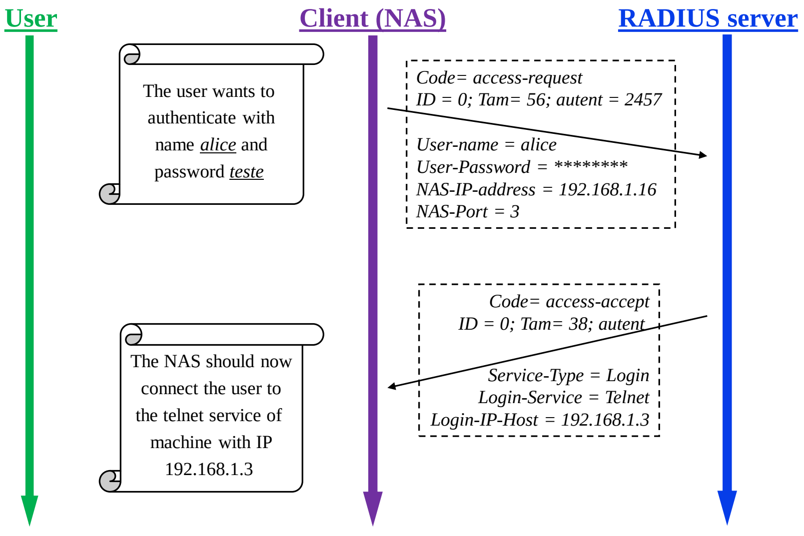
1. The client gets the user information for authentication for instance by
   - presenting a login request where the user has to fill in the username and password
   - exchanging a few messages (e.g., during the initialization of the protocol) where the information is provided
2. The client selects the RADIUS server to carry out the authentication
   - creates a *access-request* message containing various attributes, among them, the username, the password, the client identifier (NAS IP), and the port identifier (NAS port) where the connection arrived
   - since messages are sent with UDP (to port 1812), if no response arrives within a pre-determined interval, the request is re-transmitted to the same server or to an alternative one
3. When the server receives the request
   - it will validate the request fields and obtain from the database the secret shared with the client (the request is otherwise discarded)

Mode of Operation with PAP (cont)

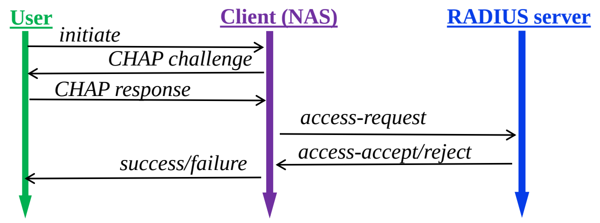
- it goes to the database to obtain information about the user, which includes the corresponding password and other configurantion data

4. The server returns one of the following answers to the client
   - *access-reject* if the request was invalid; the message might contain some text explaining the reason for the problem, which would be later shown to the client
   - *access-accept* if the request is correct; this message contains all configuration information, such as, the type of service that should be provided (SLIP, PPP, Login) and the specific values (IP address, subnet mask, MTU)

Example: Telnet to a Host

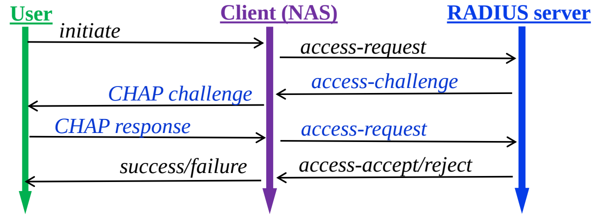


Mode of Operation with CHAP



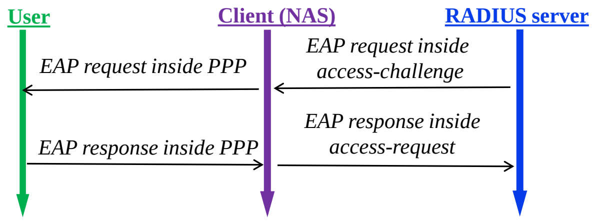
- the client sends to the user the challenge (16 bytes)
- the user returns the CHAP response including the CHAP ID and username
- the client sends to the server a *access-request* where it places in the User-Name attribute the CHAP username and in the CHAP-password attribute the CHAP ID and the value of the CHAP response; the challenge can be placed in the authenticator field <u>or</u> in the CHAP-Challenge attribute
- the server obtains the user password from the database and performs the necessary calculations to verify the correctness of the CHAP response; then, it returns an *access-accept/reject* to the client

Mode of Operation with CHAP (alternative)



- similar to the previous one, but now the server generates the challenge

Mode of Operation with EAP



- The EAP-RADIUS framework allows the messages from EAP authentication methods to be provided in the RADIUS attributes and transmitted in the RADIUS messages
  - EAP-Message attribute : encapsulates one fragment of the EAP message, including the request ID, length and EAP-type fields
  - Message Authenticator attribute : ensures the integrity of the message

Proxy-Servers: Roaming and Mobility

- RADIUS protocol provides some support for roaming through ***proxy-servers***, where the authentication of the client is not done directly in the final server
  - the client interacts with a local server A
  - server A redirects the request to the next server in-line by acting as a client and by making a request
  - as the request goes through the servers, they can modify some attributes to ensure local policies, or they can simply stop forwarding the request
  - eventually the request reaches the final server which produces a response; the response goes through the inverse path to reach the client
- Operators need to establish relationships (i.e., contracts) to support roaming
  - *roaming path* : a path needs to be created from the local server until the end server, possibly going through several other proxy-servers
  - *central proxy* : instead of creating a path, operators can create a central proxy to route requests

Potential (Security) Problems with RADIUS

- Manually configured shared secrets
  - no support for automated secret establishment
  - no methods for secret refresh
- Lookup of the shared secret at the server
  - uses the IP address in the IP header of the packet
  - causes problems if client does not have static IP address
- Proxy chaining of request
  - the client shares a secret with the first (proxy) server and not the one that gives the response (we have a transitive security)
- Protection of data
  - only a subset of the RADIUS header is protected in terms of confidentiality
  - IP and UDP headers are not protected (easy for spoofing attacks)

Bibliography

- *AAA and Network Security for Mobile Access: Radius, Diameter, EAP, PKI and IP Mobility* : Chapters 6 and 7 (optional chapters 1, 2)
- *Cryptography and Network Security, Sixth Edition*, Stallings : pages 516-525 (EAP)