Authentication in ACS 5.8.1

Authentication verifies user information to confirm the user's identity. Traditional authentication uses a name and a fixed password. More secure methods use cryptographic techniques, such as those used inside the Challenge Authentication Handshake Protocol (CHAP), OTP, and advanced EAP-based protocols. ACS supports a variety of these authentication methods.

A fundamental implicit relationship exists between authentication and authorization. The more authorization privileges granted to a user, the stronger the authentication should be. ACS supports this relationship by providing various methods of authentication.

Authentication Considerations

Username and password is the most popular, simplest, and least-expensive method of authentication. The disadvantage is that this information can be told to someone else, guessed, or captured. Simple unencrypted username and password is not considered a strong authentication mechanism but can be sufficient for low authorization or privilege levels such as Internet access.

You should use encryption to reduce the risk of password capture on the network. Client and server access-control protocols such as TACACS+ and RADIUS encrypt passwords to prevent them from being captured within a network.

However, TACACS+ and RADIUS operate only between the AAA client and ACS. Before this point in the authentication process, unauthorized persons can obtain clear-text passwords; for example, in the following setups:

- The communication between an end-user client dialing up over a phone line
- An Integrated Services Digital Network (ISDN) line terminating at a network-access server
- Over a TELNET session between an end-user client and the hosting device

Authentication and User Databases

ACS supports a variety of user databases. It supports the ACS internal database and several external user databases, including:

- Windows Active Directory
- LDAP
- RSA SecurID Servers
- RADIUS Identity Servers

This appendix describes the following:

- RADIUS-based authentication that does not include EAP:
  - PAP, page C-2
  - CHAP, page C-32
  - MSCHAPv1
  - EAP-MSCHAPv2, page C-30
- EAP family of protocols transported over RADIUS, which can be further classified as:
  - Simple EAP protocols that do not use certificates:

  EAP-MD5—For more information, see EAP-MD5, page C-5.

  LEAP—For more information, see LEAP, page C-32.

  - EAP protocols that involve a TLS-handshake and in which the client uses the ACS server certificate to perform server authentication:

  PEAP, using one of the following inner methods: PEAP/EAP-MSCHAPv2 and PEAP/EAP-GTC—For more information, see PEAPv0/1, page C-14.

  EAP-FAST, using one of the following inner methods: EAP-FAST/EAP-MSCHAPv2 and EAP-FAST/EAP-GTC—For more information, see EAP-FAST, page C-19.

  - EAP protocols that are fully certificate-based, in which the TLS handshake uses certificates for both server and client authentication:

  EAP-TLS—For more information, see EAP-TLS, page C-5.

  PEAP with inner method EAP-TLS, see PEAPv0/1, page C-14.
- Certificate Attributes, page C-32
- Machine Authentication, page C-35
- Authentication Protocol and Identity Store Compatibility, page C-36

For a list of known supplicant issues, see *Release Notes for Cisco Secure Access Control System 5.8.1*.

PAP

The Password Authentication Protocol (PAP) provides a simple method for a user to establish its identity by using a two-way handshake. The PAP password is encrypted with the shared secret and is the least sophisticated authentication protocol.

ACS checks the ID-Password pair against the external database, Identity Store, until ACS acknowledges the authentication or terminates the connection.

PAP is not a strong authentication method since it offers little protection from repeated trial-and-error attacks.

Note The RADIUS with PAP authentication flow includes logging of passed and failed attempts.

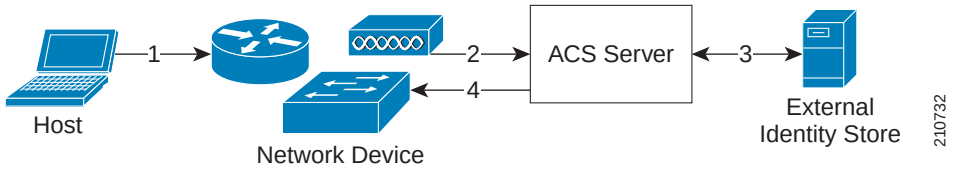
RADIUS PAP Authentication

You can use different levels of security concurrently with ACS for different requirements. PAP applies a two-way handshaking procedure. If authentication succeeds, ACS returns an acknowledgment; otherwise, ACS terminates the connection or gives the originator another chance.

The originator is in total control of the frequency and timing of the attempts. Therefore, any server that can use a stronger authentication method will offer to negotiate that method prior to PAP. RFC 1334 defines PAP.

Figure C-1 illustrates RADIUS with PAP authentication.

Figure C-1 RADIUS with PAP Authentication Use Case



1	A host connects to the network. Any communication protocol may be used depending on the host.	**3**	ACS uses an external identity store to validate the user's credentials.
2	The network device sends a RADIUS access request to ACS.	**4**	The RADIUS response (Access-Accept or Access-Reject) is sent to the network device that will apply the decision.

EAP

Extensible Authentication Protocol (EAP) is an authentication framework for wireless networks and point-to-point connections. EAP supports multiple authentication methods, and provides common functions and rules for negotiation of the desired authentication method:

- Server authentication request
- Client authentication response
- Server success authentication result
- Server failure authentication result
- Silent discard of client packets if they do not meet integrity and security conditions
- Rules for server-initiated EAP method negotiation
- Message sequencing, and tracking responses to requests
- Retransmit

EAP is a lock-step protocol; after the initial request, ACS cannot send a new request before receiving a valid response from the client.

In ACS 5.8.1, EAP is encapsulated in the RADIUS protocol. Incoming and outgoing EAP messages are stored in a RADIUS EAP-Message attribute (79). A single RADIUS packet can contain multiple EAP-Message attributes when the size of a particular EAP message is greater than the maximum RADIUS attribute data size (253 bytes).

The RADIUS State attribute (24) stores the current EAP session reference information, and ACS stores the actual EAP session data.

The EAP standard is described in:

- RFC 3748—Extensible Authentication Protocol (EAP).
- RFC 3579—RADIUS Support For Extensible Authentication Protocol (EAP).

In the EAP process:

Step 1 The network device sends an EAP Request to a host when the host connects to the network.

Step 2 The host sends an EAP Response to the network device; the network device embeds the EAP packet that it received from the host into a RADIUS request and sends it to ACS, which is acting as the EAP server.

Step 3 ACS negotiates the EAP method for authentication. The client can acknowledge the EAP method that the EAP server suggests or, it can respond with a negative acknowledgment (NAK) and suggest a list of alternative EAP methods. The server and client must reach agreement about the EAP method to use to instantiate authentication.

Table C-1 lists the EAP codes for each type of EAP message.

Table C-1 EAP Codes

EAP message type	EAP code
Accept-request	1
Response	2
Success	3
Failure	4

Table C-2 describes the EAP methods that ACS 5.8.1 supports.

Table C-2 Supported EAP methods

EAP Method	Description
EAP-MD5	Message Digest 5 Protocol. For more information see EAP-MD5, page C-5.
LEAP	Lightweight Extensible Authentication Protocol.
PEAPv0v1	Protected Extensible Authentication Protocol version 0 and version 1. For more information see PEAPv0/1, page C-14.
EAP-FAST	EAP Flexible Authentication via Secured Tunnel (EAP-FAST) protocol. For more information see EAP-FAST, page C-19.
EAP-MSCHAPv2	Microsoft Challenge Handshake Authentication Protocol version 2. For more information see EAP-MSCHAPv2, page C-30.

Table C-2 Supported EAP methods (continued)

EAP Method	Description
EAP-GTC	EAP Generic Token Card.
EAP-TLS	Extensible Authentication Protocol-Transport Layer Security. For more information, see Exporting Credentials, page C-11.

ACS supports full EAP infrastructure, including EAP type negotiation, message sequencing and message retransmission. All protocols support fragmentation of big messages.

In ACS 5.8.1, you configure EAP methods for authentication as part of access service configuration. For more information about access services, see ACS 5.x Policy Model, page 3-1

EAP-MD5

This section contains the following topics:

Overview of EAP-MD5

EAP Message Digest 5-(EAP-MD5) provides one-way client authentication. The server sends the client a random challenge. The client proves its identity by hashing the challenge and its password with MD5. EAP-MD5 is vulnerable to dictionary attacks when it is used over an open medium.

This is because hackers are able to see the challenge and response. Since no server authentication occurs, it is also vulnerable to falsification.

Related Topics

- Host Lookup, page 4-12
- Overview of Agentless Network Access, page 4-12

EAP- MD5 Flow in ACS 5.8.1

ACS supports EAP-MD5 authentication against the ACS internal identity store. Host Lookup is also supported when using the EAP-MD5 protocol. See Host Lookup, page 4-12.

Related Topics

- Authentication Protocol and Identity Store Compatibility, page C-36
- Overview of Agentless Network Access, page 4-12

EAP-TLS

This section contains the following topics:

Overview of EAP-TLS

EAP-TLS is one of the methods in the EAP authentication framework, and is based on the 802.1x and EAP architecture. Components involved in the 802.1x and EAP authentication process are the:

- Host—The end entity, or end user's machine.
- AAA client—The network access point.
- Authentication server—ACS.

The EAP-TLS standard is described in:

- RFC 2716—PPP EAP-TLS Authentication Protocol
- RFC 3079—Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)

This section contains the following topics:

The host must support EAP-TLS authentication. The access point must support the EAP authentication process in the 802.1x environment (the access point is not aware of the EAP authentication protocol type).

Related Topics

- Configuring CA Certificates, page 8-95
- Certificate-Based Network Access, page 4-8
- ACS and Cisco Security Group Access, page 4-22
- EAP-TLS Flow in ACS 5.8.1, page C-13

User Certificate Authentication

EAP-TLS is a mutual authentication method for certificate-based authentication; the client and server authenticate each other by using digital certificates. Certificates must meet specific requirements on the server and client for successful authentication. EAP and TLS are Internet Engineering Task Force (IETF) RFC standards.

The EAP protocol carries initial authentication information, specifically the encapsulation of EAP over LANs (EAPOL) as established by IEEE 802.1x. TLS uses certificates for user authentication and dynamic ephemeral session key generation.

After the peer is authenticated and a session is created, the information is cached on ACS for a certain amount of time. The session can be re-established by using the EAP-TLS session state and the session ticket resume, without an additional certificate exchange.

ACS 5.8.1 maintains the server certificate and private key in files on the ACS server, which it uses during EAP-TLS processing. You can choose the certificate authorities (CAs) that can be trusted to sign on client certificates.

EAP-TLS authentication involves two elements of trust:

- The EAP-TLS negotiation establishes end-user trust by validating, through RSA signature verifications, that the user possesses a keypair that a certificate signs.

  This process verifies that the end user is the legitimate keyholder for a given digital certificate and the corresponding user identification in the certificate. However, trusting that a user possesses a certificate only provides a username-keypair binding.

- Using a third-party signature, usually from a CA, that verifies the information in a certificate. This third-party binding is similar to the real-world equivalent of the stamp on a passport.

  You trust the passport because you trust the preparation and identity-checking that the particular country's passport office made when creating that passport. You trust digital certificates by installing the root certificate CA signature.

Some situations do not require this second element of trust that is provided by installing the root certificate CA signature. When such external validation of certificate legitimacy is not required, you can use the ACS self-signed certificate capability.

Depending on the end-user client involved, the CA certificate for the CA that issued the ACS server certificate is likely to be required in local storage for trusted root CAs on the end-user client computer. For more information, see Adding a Certificate Authority, page 8-95.

EAP-TLS-compliant AAA clients include:

- Cisco 802.1x-enabled switch platforms (such as the Catalyst 6500 product line)
- Cisco Aironet Wireless solutions

To accomplish secure Cisco Aironet connectivity, EAP-TLS generates a dynamic, per-user, per-connection, unique session key.

ACS 5.8.1 now supports certificate name constraint extension. It accepts client certificates whose issuers contain the name constraint extension. It checks the client certificates for CA and sub-CA certificates. This extension defines a name space for all subject names in the subsequent certificates in a certificate path. It applies to both the subject distinguished name and the subject alternative name. These restrictions are applicable only when the specified name form is present in the client certificate. The ACS authentication fails if the client certificate is excluded or not permitted by the namespace.

Related Topics

- Configuring CA Certificates, page 8-95
- Certificate-Based Network Access, page 4-8

PKI Authentication

EAP-TLS uses public key infrastructures (PKI) concepts:

- A host requires a valid certificate to authenticate to the LAN network.
- The AAA server requires a server certificate to validate its identity to the clients.
- The certificate-authority-server infrastructure issues certificates to the AAA server(s) and the clients.

An SSL/TLS tunnel authentication is conducted by both peers and is initiated by the client. In ACS, the tunnel can be either authenticated by:

- both peers
- either one
- neither client or host

A tunnel that is constructed without an authentication is considered an anonymous tunnel, and is usually constructed by the Diffie-Hellman key exchange protocol. ACS supports the SSL/TLS session resume feature for TLS. ACS maintains the tunnel keys and cipher used to establish the tunnel communication in the cache for each session. Fetching an old session is based on the session ID which is unique for each client.

You can configure the timeout for each session in the cache, for each protocol individually. The lifetime of a session is measured from the beginning of the conversation and is determined when the TLS session is created.

ACS supports establishment of a tunnel from a commonly shared key known to the client and the server for the EAP-FAST protocol. The key that is securely agreed upon between the two peers is used to derive a shared tunnel TLS-master-key that is used to open a tunnel. This mechanism involves a shorter TLS negotiation.

An anonymous Diffie-Hellman tunnel relates to the establishment of a completely anonymous tunnel between a client and a server for cases where none of the peers authenticates itself. ACS runtime supports anonymous Diffie-Hellman tunnels for EAP-FAST with a predefined prime and a predefined generator of two. There is no server authentication conducted within anonymous Diffie-Hellman tunnel cipher-suites.

An authenticated Diffie-Hellman tunnel is similar to an anonymous Diffie-Hellman tunnel. The additional factor of the authenticated Diffie-Hellman tunnel is that peer authentication is conducted through an RSA certificate. ACS supports Authenticated-Diffie-Hellman tunnels for EAP-FAST where the server authenticates by using its own certificate.

Additional client authentications are conducted within the tunnel by using other protocols, such as EAP-MSCHAPv2 or EAP-GTC for the inner EAP method.

Related Topics

- Configuring Local Server Certificates, page 18-16
- Configuring CA Certificates, page 8-95
- Configuring Certificate Authentication Profiles, page 8-99

PKI Credentials

This section contains the following topics:

PKI Usage

ACS supports using certificates for various PKI use cases. The main use case is the EAP-TLS protocol, where the PKI is used to authenticate not only the server, but also the client (PEAP and EAP-FAST also make use of certificates for server authentication, but do not perform client authentication). Other protocols which use the PKI credentials are LDAPS, HTTPS Management protocol, SSH, and SFTP.

For TLS related EAP protocols, a single local certificate is used to authenticate the server for all the TLS related EAP protocols. You can pick the certificate to use from any of the certificates containing a private-key in the Local Certificate store.

For other protocols, such as HTTPS, SFTP, and SSH, and for the message-bus ActiveMQ authentication, a single certificate should be configured to authenticate ACS. You can pick the certificate to use from any of the certificates containing a private-key in the Local Certificate store. You can configure the same local certificate for the TLS-related EAP protocols and for HTTPS Management protocol.

For HTTPS, SFTP, SSH and ActiveMQ, an auto-generated self-signed certificates can be used as the means for server authentication.

If ACS deployment is to be operated in FIPS mode, you must ensure that all local and certificate store certificates are FIPS-compliant. This means that each certificate must have a minimum key size of 2048 bytes, and use SHA-1 or SHA-256 encryption.

Fixed Management Certificates

ACS generates and uses self-signed certificates to identify various management protocols such as the Web browser, HTTPS, ActiveMQ SSH, and SFTP.

Self-signed certificates are generated when ACS is installed and are maintained locally in files outside of the ACS database. You cannot modify or export these certificates. You can, however, assign imported certificates to management interfaces.

Importing Trust Certificates

ACS supports PEM or DER formatted X509 certificate files. You can add a trust certificate to the trust certificate store. ACS verifies that an imported certificate complies with the X509 format and does not perform any hierarchical certificate signature verification. ACS also supports the Microsoft proprietary private key format.

You can mark the acquired certificate for immediate trust for TLS related EAP protocols as the EAP CTL. The trust certificate store does not allow for duplicate trust certificates. These are the rules for rejecting certificates:

- Two certificates cannot have the same subject.
- Two certificates cannot have the same issuer and the same serial-number.

Acquiring Local Certificates

This topic describes the methods for ACS to acquire PKI credentials, and the ways that you can sets the public or private keys pairs to each ACS server in the ACS domain.

An X509 certificate contains the credentials which include the public key, and a PKCS#12 [?10.1] that holds the private key protected with a password that goes with it.

The ACS domain may have more than a single ACS server; each domain should have its own set of PKI key pairs to identify itself through the appropriate interfaces.

Some interfaces may require that the certificate that identifies ACS, contain the IP or FQDN of the ACS server, in its Common Name (CN) for better binding of the certificate to the IP of the server, for example, the HTTPS ACS server certificate which is used for the Web interface.

For other interfaces, it may be possible to use a common certificate that can be shared between the servers, however, Cisco does not recommend that you use a common certificate. Each ACS PKI credentials may be obtained either from a self-signed certificate or a certificate signed by a common certificate authority (CA).

For protocols that require the ACS identification, clients should be deployed with at least the lowest common certificate that dominates all the ACS servers certificates that are used to identify each ACS.

You can pick the PKI policy to be used in your organization and configure the PKI credentials for the ACS domain.

The configured certificate with its private-key should not be used outside the ACS machine

Related Topics

- Importing the ACS Server Certificate, page C-10
- Initial Self-Signed Certificate Generation, page C-10
- Certificate Generation, page C-11

Importing the ACS Server Certificate

When you manually import and ACS server certificate you must supply the certificate file, the private key file, and the private key password used to decrypt the PKCS#12 private key. The certificate along with its private-key and private-key-password, is added to the Local Certificate store. For non-encrypted private-keys, the user supplied password may be ignored.

ACS supports PEM or DER formatted X509 certificate files. ACS verifies that an imported certificate complies with a the X509 format and does not perform any hierarchical certificate signature verification.

When importing a certificate, you can configure the certificate for protocol that require an ACS server certificate, such as TLS related EAP protocols and HTTPS Management protocol.

Note Only EAP and HTTPS Management protocols can be configured in ACS 5.8.1 for certificate-based authentication.

The input password and private-key, which are cryptographically sensitive information, are passed over the HTTPS channel. Using HTTPS with a non-authenticated server, for example, a self-signed certificate for HTTPS server authentication, is not a secure method of passing this sensitive information.

Note If ACS is set to operate in FIPS mode, the certificate key size must be 2048 bits or greater in size and use either SHA-1 or SHA-256 hash algorithm.

Related Topics

- Importing Trust Certificates, page C-9
- Initial Self-Signed Certificate Generation, page C-10
- Certificate Generation, page C-11

Initial Self-Signed Certificate Generation

An automatically generated, self-signed certificate is placed in the Local Certificate store for each ACS server. This certificate is used to identify ACS for TLS-related EAP protocols and for HTTPS Management protocols.

The self-signed certificate is generated with the CN equal to the machine's hostname, as required for HTTPS certificates, and is generated when ACS is installed.

Certificate Generation

You can generate ACS server certificates through the Web interface. The output of this process is a certificate or a certificate request and it's corresponding private-key and password. The generated private-key is structured as PKCS#12 encrypted, by using a relatively strong automatically generated password based on at least 128 bit of randomness.

You can select any of these generated private-key lengths: 512, 1024, 2048 or 4096 bit. The certificate digest algorithm used by the ACS is SHA1 and SHA2 256-bit.

Note You should install Windows XP SP3 to use SHA2 256-bit certificates as management certificates.

There are two types of certificate generation:

- Self-signing certificate generation—ACS supports generation of an X.509 certificate and a PKCS#12 private key. The passphrase used to encrypt the private key in the PKCS#12 automatically generates stronger passwords, and the private key is hidden in the local certificate store.

  You can select the newly generated certificate for immediate use for HTTPS Management protocol, for TLS-related EAP protocols, or both.

- Certificate request generation—ACS supports generation of a PKCS#10 certificate request with a PKCS#12 private key. The request is downloaded through the Web interface and should be formatted with PEM representation with a REQ extension.

  The passphrase used to encrypt the private key in the PKCS#12 automatically generates stronger passwords, and the private-key is hidden in the ACS database. You can download the request file to be signed offline by the RA.

  After the RA signs the request, you can install the returned signed certificate on ACS and bind the certificate with its corresponding private key. The binding of certificate and its private key is automatic.

  After binding the signed certificate with the private key, you can mark this certificate for immediate use for HTTPS Management protocol, for TLS-related EAP protocols, or both.

Related Topics

- Configuring CA Certificates, page 8-95
- Configuring Certificate Authentication Profiles, page 8-99
- EAP-TLS Flow in ACS 5.8.1, page C-13

Exporting Credentials

You can export a general trust certificates, an ACS server certificate with or without private keys, and previously generated certificates requests from the certificate stores. You cannot export the request for a private-key. You can download certificates file with a *.CER* extension. The file format is not changed from the format that is imported into ACS.

You can download the public certificate as a regular certificate with *.CER* extension for ACS server certificates, that also contain a private key. The file format is retained.

You can export a public request to re-issue a certificate request to an RA, for certificate-requests. The request is downloaded with an REQ extension and is formatted identically to the format that it was generated by.

Only administrators with the highest administrator privileges can export the certificate private key and its password. A warning about the security implications of such an action is conveyed twice, to approve the export operation.

After this double check, the private-key files can be downloaded as a *.PVK* extension, and the private-key password can be downloaded as a *.PWD* extension. The private-key file format is retained.

Credentials Distribution

All certificates are kept in the ACS database which is distributed and shared between all ACS nodes. The ACS server certificates are associated and designated for a specific node, which uses that specific certificate.

Public certificates are distributed along with the private keys and the protected private key passwords by using the ACS distributed mechanism. ACS implements a method of protection to prevent a private-key to be used by other servers other than the one to which the private-key is designated to. This protection mechanism applies only to encrypted private-keys.

The PKI policy for private keys is that private keys are not supposed to be usable by other entities which are not associated with the ACS server to which they are designated to. ACS supports cryptographic protection of the private-keys to prevent possible use outside of the ACS server machine to which they are designated to.

Hardware Replacement and Certificates

When hardware fails, a new node is used for replacing a malfunctioning node. The malfunctioning node's certificates are removed from the distributed database of the primary server, and the new node's certificates are then being passed to the primary to be associated with the newly replaced node.

This process of certificate changing is conducted as part of the hardware replacement process when the new node registered to the domain, The certificate distribution is based on the server’s IP address.

Securing the Cryptographic Sensitive Material

There are several types of PKI-related keys that are stored in the ACS database. These keys have different cryptographic storage requirements that must comply to SEC-RCV-CRED-2 which is part of the Cisco security baseline. These requirements include:

- Public keys that usually reside in a certificate may be stored plain open as they are used to pass on the clear text to clients and contain only public keys.
- Private keys must be stored encrypted as PKCS#12 by using a relatively strong password.
- The password for the PKCS#12 private-keys must be stored in the ACS database. Since the ACS database is encrypted, this does not pose a serious security concern. ACS 5.8.1 distributes the entire database between all the ACS servers.

  ACS encrypts the private-key passwords by using a password that exists only for the machine, thus preventing possible use of the private-keys by other machines. The private-key password key is maintained in */opt/CSCOacs/config/prikeypwd.key* on the ACS file-system.

Other certificate repositories such as the tomcat key-store should have the same properties as the ACS database. Private-keys are encrypted by a password that is kept secured in the database.

Private Keys and Passwords Backup

The entire ACS database is distributed and backed-up on the primary ACS along with all the certificates, private-keys and the encrypted private-key-passwords. The private-key-password-key of the primary server is also backed up with the primary's backup.

Other secondary ACS private-key-password-keys are not backed-up. Backups are encrypted and also can pass relatively secured in and out of the ACS servers. The private keys in backups are protected by the PKCS#12 and the backup file encryption. The passwords that are used to open the PKCS#12 private-keys are protected with the backup encryption.

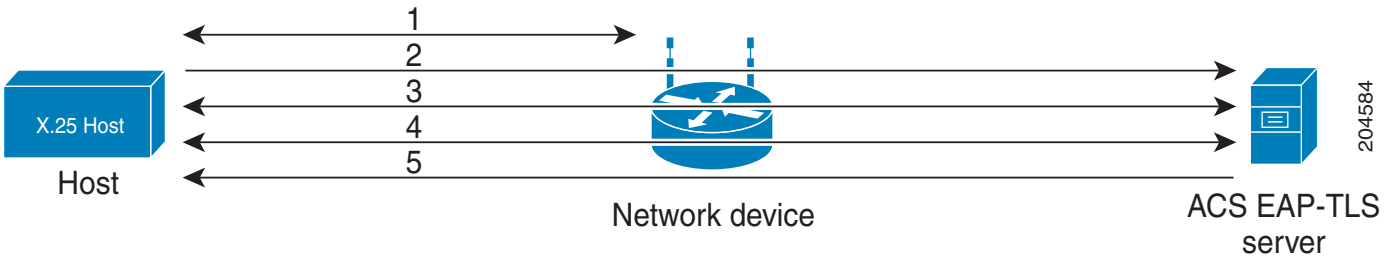
EAP-TLS Flow in ACS 5.8.1

An EAP-TLS server exchanges data with a client by using packets based on the EAP Request and response packets; the packets are extended by specific EAP-TLS data. ACS acts as the EAP-TLS server and uses the Open Secure Sockets Layer (OpenSSL/CiscoSSL) library to process the TLS conversation. The ACS EAP-TLS server produces 128-bit MPPE send and receive keys that are used for encrypted communication between the client and server.

The ACS EAP-TLS server sends MPPE keys to the client in vendor-specific RADIUS attribute (26) by using vendor code Microsoft (311), and attributes MS-MPPE-Send-Key (16) and MS-MPPE-Recv-Key (17).

Figure C-2 shows the EAP-TLS processing flow between the host, network device, and ACS EAP-TLS server when the stateless session resume option is not used.

Figure C-2 EAP-TLS Flow



1	A host connects to the network. The network device sends an EAP Request to the host.	**2**	The host sends an EAP Response to the network device; the network device embeds the EAP packet that it received from the host into a RADIUS Access-Request and sends it to ACS.
3	ACS negotiates the EAP method for authentication. The server and client must reach agreement to use EAP-TLS (EAP Request method 13) during EAP method negotiation to instantiate EAP-TLS authentication.	**4**	The client (host) and server (ACS) exchange certificates; this exchange involves several messages. EAP-TLS authentication is successful after the client and server have authenticated each other, and each side is aware that the other side has authenticated them.
5	ACS returns an EAP Success (or EAP Failure) message to the host and returns a RADIUS Access-Accept (or RADIUS Access-Reject) that includes session keys to the network device.		

Note All communication between the host and ACS goes through the network device.

EAP-TLS authentication fails if the:

- Server fails to verify the client's certificate, and rejects EAP-TLS authentication.
- Client fails to verify the server's certificate, and rejects EAP-TLS authentication.

  Certificate validation fails if the:

  – Certificate has expired.

  – Server or client cannot find the certificate issuer.

  – Signature check failed.
- The client dropped cases resulting in malformed EAP packets.

EAP-TLS also supports the Session Resume feature. ACS supports the EAP-TLS session resume feature for fast reauthentication of a user who has already passed full EAP-TLS authentication. If the EAP-TLS configuration includes a session timeout period, ACS caches each TLS session for the duration of the timeout period.

When a user reconnects within the configured EAP-TLS session timeout period, ACS resumes the EAP-TLS session and reauthenticates the user with TLS handshake only, without a certificate check.

ACS 5.8.1 supports EAP-TLS session resumption without session state to be stored at the server. It also supports session ticket extension as described in RFC 5077. The ACS server creates a ticket and sends it to an EAP-TLS client. The client presents the ticket to ACS to resume a session.

The Stateless session resumption is supported in the distributed deployment, so that a session ticket issued by one node is accepted by another node.

The entire ticket is authenticated over its fields using a MAC with a 128-bit authentication key. The fields are encrypted using AES-CBC with a 128-bit encryption key and IV that are found in the ticket. The ACS administrator configures a limited lifetime for the session ticket.

Related Topics

- Types of PACs, page C-23
- User Certificate Authentication, page C-6

PEAPv0/1

This section contains the following topics:

ACS 5.8.1 supports these PEAP supplicants:

- Microsoft Built-In Clients 802.1x XP (PEAPv0 only)
- Microsoft Built-In Clients 802.1x Vista (PEAPv0 only)
- Microsoft Built-In Clients 802.1x Windows 7
- CSSC v.4.0
- CSSC v.5

- Cisco AC 3.x
- Funk Odyssey Access Client 4.0.2 and 5.x
- Intel Supplicant 12.4.x

Overview of PEAP

PEAP is a client-server security architecture that you use to encrypt EAP transactions, thereby protecting the contents of EAP authentications. PEAP uses server-side public key certificates to authenticate the server.

It then creates an encrypted SSL/TLS tunnel between the client and the authentication server. The ensuing exchange of authentication information to authenticate the client is then encrypted and user credentials are safe from eavesdropping.

PEAP is similar to EAP-TLS but uses a different client authentication method. PEAP provides authentication, by using server certificates, a TLS tunnel and client authentication through that encrypted tunnel. Unlike EAP-TLS, PEAP requires the client to use another EAP type, like EAP-MSCHAPv2.

PEAP authentications always involve two phases:

- In phase1, the end-user client authenticates ACS. This action requires a server certificate and authenticates ACS to the end-user client, ensuring that the user or machine credentials sent in phase two are sent to a AAA server that has a certificate issued by a trusted CA. The first phase uses a TLS handshake to establish an SSL tunnel between the end-user client and the AAA server.

Note Depending on the end-user client involved, the CA certificate for the CA that issued the ACS server certificate is likely to be required in local storage for trusted root CAs on the end-user client computer.

- In the second phase, ACS authenticates the user or machine credentials by using an EAP authentication protocol. The SSL tunnel that was created in phase1 protects the EAP authentication.

  The inner-method authentication type that is negotiated during phase 2 can be either EAP-MSCHAPv2, EAP-GTC or EAP-TLS. The combination of the outer PEAP method with a specific inner EAP method is denoted using brackets (); for example, PEAP(EAP-MSCHAPv2) or PEAP(EAP-GTC).

  An improvement in security that PEAP offers is identity protection. This improvement is the potential for protecting the username in all PEAP transactions. After phase one of PEAP, all data is encrypted, including username information that is usually sent in clear text.

  The Microsoft PEAPv0 client does not provide identity protection; the Microsoft PEAPv0 client sends the username in clear text in phase one of PEAP authentication.

In ACS 5.8.1, PEAP is encapsulated in RADIUS protocol. Inner-method EAP messages are encapsulated in an EAP-TLV method. ACS also supports cryptobinding TLV extension in MS PEAP. In ACS 5.8.1, you have an option to deliberately enable PEAPv0 only for the legacy clients.

Supported PEAP Features

This section contains the following topics:

Server Authenticated and Unauthenticated Tunnel Establishment Modes

Tunnel establishment helps prevent an attacker from injecting packets between the client and the network access server (NAS) or, to allow negotiation of a less secure EAP method. The encrypted TLS channel also helps prevent denial of service attacks against the ACS.

A client EAP message is always carried in the RADIUS Access-Request message, and the server EAP message is always carried in the RADIUS Access-Challenge message. The EAP Success message is always carried in RADIUS Access-Accept message.

The EAP Failure message is always carried in the RADIUS Access-Reject message. The client's PEAP message may cause the RADIUS client's message to drop unless the policy component is configured otherwise.

Fast Reconnect

When a session resumes, another method of decreasing the authentication time is to skip the inner method, also known as fast reconnect. After a tunnel is built, the authentication flow goes directly to exchange authentication information with a Result TLV Success (v0)/tunneled EAP Success message for successful authentication and an EAP Failure message in case of unsuccessful authentication.

You can configure ACS to enable the fast reconnect option. After successful authentication, the client is able to perform a fast reconnect during a certain timeframe. PEAP fast reconnect reduces the delay in the time between an authentication request by a client and the response by ACS.

Fast reconnect also allows wireless clients to move between access points without repeated requests for authentication, which reduces resource requirements for the client and the server.

The user identity and the protocol used for user authentication (inner method) should be cached along with the TLS session to allow fast reconnect.

Session Resume

ACS supports a session resume feature for PEAP-authenticated user sessions. When this feature is enabled, ACS caches the TLS session that is created during phase one of PEAP authentication, provided that the user successfully authenticates in phase two of PEAP.

If a user needs to reconnect and the original PEAP session has not timed out, ACS uses the cached TLS session, resulting in faster PEAP performance and a lessened AAA server load.

ACS stores the session in the cache after a successful full authentication. A client may try to resume the same session during a specific timeframe. A server certificate is not presented and the tunnel is built by using the session information from the OpenSSL/CiscoSSL session cache. The authentication flow then goes directly to the inner method.

If a client attempts to perform session resume but the timeout elapsed, ACS reverts to the full authentication flow.

You can configure the session resume and timeout values.

Protected Exchange of Arbitrary Parameters

TLV tuples provide a way to exchange arbitrary information between the peer and ACS within a secure channel.

Cryptobinding TLV Extension

The cryptobinding TLV extension in MS PEAP authentication is used to ensure that both the EAP peer (client) and the EAP server (ACS) are participating in the inner and outer EAP authentications of the PEAP authentication.

This cryptobinding process takes place as a two-way handshake between the PEAP server and PEAP peer. It consists of two messages, which include the cryptobinding request that is sent from a PEAP server to the PEAP peer and the cryptobinding response that is sent back from the PEAP peer to the PEAP server. This feature is implemented in ACS as primary for the MS Win 7 supplicant.

The TLV contains a compound MAC that is calculated using the following: PRF based on HMAC-SHA1-160 with TLV body as input data, a key derived from the PEAP tunnel key, and the inner method as session key. ACS verifies that the cryptobinding response TLV is received from the client. If the compound MAC is not equal to the expected data, then ACS fails the conversation. Cryptobinding is available for all inner methods. Cryptobinding is restricted to PEAPv0, because there are differences in protected termination flow. Cryptobinding is also applicable for PEAP session resume and fast reconnect. Some supplicants may not support cryptobinding TLV. If you send a cryptobinding TLV to a supplicant that does not support cryptobinding, then the supplicant does not provide a proper cryptobinding response. This improper response is considered to be an error on ACS and is accompanied with a PEAP_CRYPTOBINDING_FAILED message.

PEAP Flow in ACS 5.8.1

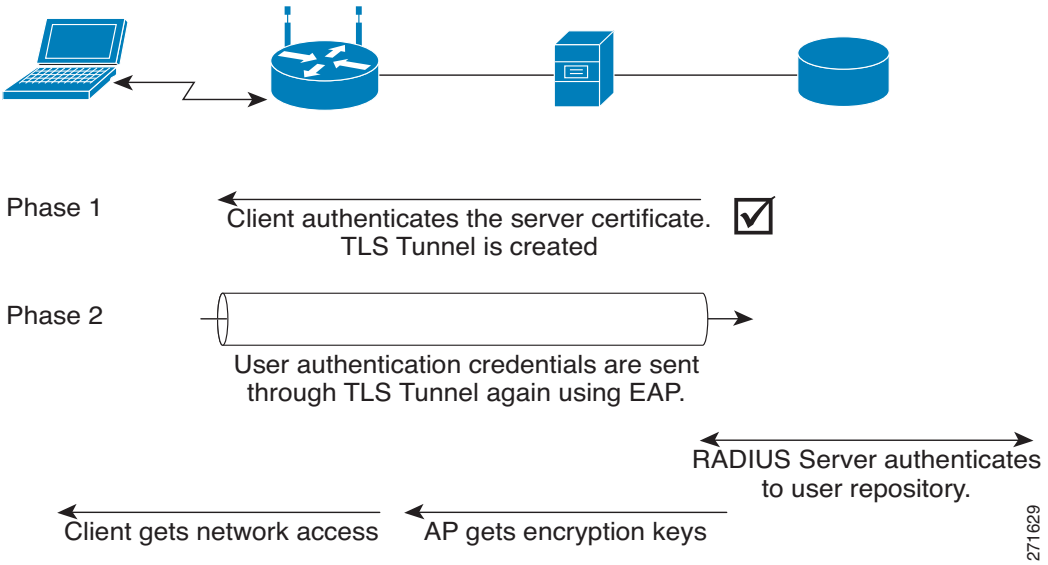
The PEAP protocol allows authentication between ACS and the peer by using the PKI-based secure tunnel establishment and the EAP-MSCHAPv2 protocol as the inner method inside the tunnel. The local certificate can be validated by the peer (server-authenticated mode) or not validated (server-unauthenticated mode).

This section contains:

- Creating the TLS Tunnel, page C-18
- Authenticating with MSCHAPv2, page C-18

Figure C-3 shows the PEAP processing flow between the host, access point, network device, and ACS EAP-TLS server.

Figure C-3 PEAP Processing Flow



Creating the TLS Tunnel

The following describes the process for creating the TLS tunnel:

1	After creating a logical link, the wireless AP sends an EAP-Request/Identity message to the wireless client.	**2**	The wireless client responds with an EAP-Response/Identity message that contains the identity (user or computer name) of the wireless client.
3	The wireless AP sends the EAP-Response/Identity message to ACS. From this point on, the logical communication occurs between ACS and the wireless client by using the wireless AP as a pass-through device.	**4**	ACS sends an EAP-Request/Start PEAP message to the wireless client.
5	The wireless client and ACS exchange a series of TLS messages through which the cipher suite for the TLS channel is negotiated. In ACS 5.8.1, the client certificate is not used in PEAP.	**6**	At the end of the PEAP negotiation, ACS has authenticated itself to the wireless client. Both nodes have determined mutual encryption and signing keys (by using public key cryptography, not passwords) for the TLS channel.

Authenticating with MSCHAPv2

After the TLS tunnel is created, follow these steps to authenticate the wireless client credentials with MSCHAPv2:

1	ACS sends an EAP-Request/Identity message.	**2**	The wireless client responds with an EAP-Response/Identity message that contains the identity (user or computer name) of the wireless client.
3	ACS sends an EAP-Request/EAP-MSCHAPv2 challenge message that contains a challenge string.	**4**	The wireless client responds with an EAP-Response/EAP-MSCHAPv2 Response message that contains the response to the ACS challenge string and a challenge string for ACS.
5	ACS sends an EAP-Request/EAP-MSCHAPv2 success message, which indicates that the wireless client response was correct and contains the response to the wireless client challenge string.	**6**	The wireless client responds with an EAP-Response/EAP-MSCHAPv2 acknowledgment message, indicating that the ACS response was correct.
7	ACS sends an EAP-Success message.		

At the end of this mutual authentication exchange, the wireless client has provided proof of knowledge of the correct password (the response to the ACS challenge string), and ACS has provided proof of knowledge of the correct password (the response to the wireless client challenge string). The entire exchange is encrypted through the TLS channel created in PEAP.

Related Topics

- Authentication Protocol and Identity Store Compatibility, page C-36
- Configuring PEAP Settings, page 18-3

EAP-FAST

This section contains the following topics:

Overview of EAP-FAST

The EAP Flexible Authentication via Secured Tunnel (EAP-FAST) protocol is a new, publicly accessible IEEE 802.1x EAP type that Cisco developed to support customers that cannot enforce a strong password policy and want to deploy an 802.1x EAP type that does not require digital certificates.

EAP-FAST supports a variety of user and password database types, password change and expiration, and is flexible, easy to deploy, and easy to manage. For more information about EAP-FAST and comparison with other EAP types, see:

http://www.cisco.com/en/US/products/hw/wireless/ps430/products_qanda_item09186a00802030dc.shtml.

EAP-FAST is a client-server security architecture that encrypts EAP transactions with a TLS tunnel. While similar to PEAP in this respect, it differs significantly in that EAP-FAST tunnel establishment is based on strong secrets that are unique to users.

These secrets are called Protected Access Credentials (PACs), which ACS generates by using a master key known only to ACS. Because handshakes based on shared secrets are intrinsically faster than handshakes based on PKI, EAP-FAST is the fastest of the advanced EAP protocols (including EAP-TLS and PEAP) that establish a TLS connection to encrypt the traffic between the supplicant and ACS. No certificate management is required to implement EAP-FAST.

EAP-FAST occurs in three phases:

- Phase zero—Unique to EAP-FAST, phase zero is a tunnel-secured means of providing an EAP-FAST end-user client with a PAC for the user requesting network access. (See Automatic In-Band PAC Provisioning, page C-24.)

  Providing a PAC to the end-user client is the sole purpose of phase zero. The tunnel is established based on an anonymous Diffie-Hellman key exchange for Anonymous In-band provisioning. Authenticated In-band provisioning uses other cipher suites.

  If EAP-MSCHAPv2 or EAP-GTC authentication succeeds, ACS provides the user with a PAC. To determine which databases support EAP-FAST phase zero, see Authentication Protocol and Identity Store Compatibility, page C-36.

Note Phase zero is optional and PACs can be manually provided to end-user clients. (See Manual PAC Provisioning, page C-24.)

  The Allow Anonymous In-Band PAC provisioning option provides an end-user client with a PAC by using EAP-FAST phase zero. If this check box is checked, ACS establishes a secured connection with the end-user client for the purpose of providing the client with a new PAC.

  This option allows an anonymous TLS handshake between the end-user client and ACS (EAP-MSCHAPv2 and EAP-GTC are used as inner methods.)

  The Allow Authenticated In-Band PAC provisioning option provisions an end-user client with a PAC by using EAP-FAST phase zero with TLS server-side authentication. This option requires that you install a server certificate.

  In general, phase zero of EAP-FAST does not authorize network access. However, if you choose the Accept Client on Authenticated Provisioning option, ACS sends a RADIUS Access-Accept (containing an EAP Success) at the end of a successful phase zero PAC provisioning, and the client is not forced to reauthenticate again.

  This option can be enabled only when the Allow Authenticated In-Band PAC Provisioning option is also enabled.

- Phase one—In phase one, ACS and the end-user client establish a TLS tunnel based on the PAC that the end-user client presents. This phase requires that the end-user client has been provided a PAC for the user who is attempting to gain network access and that the PAC is not expired. The means by which PAC provisioning has occurred is irrelevant; you can use automatic or manual provisioning.
- Phase two—In phase two, ACS authenticates the user's credentials from within the protected TLS tunnel that was constructed in phase one, using EAP-MSCHAPv2 or EAP-GTC as the inner EAP method. To determine which databases support EAP-FAST phase two, see Authentication Protocol and Identity Store Compatibility, page C-36.

Phase one and phase two are subsequent parts of the same EAP-FAST conversation.

EAP-FAST can protect the username in all EAP-FAST transactions. ACS does not perform user authentication based on a username that is presented in phase one, however, whether the username is protected during phase one depends on the end-user client.

If the end-user client does not send the real username in phase one, the username is protected. After phase one of EAP-FAST, all data is encrypted, including username information that is usually sent in clear text.

ACS supports password aging with EAP-FAST for users who are authenticated by Windows user databases. Password aging can work with phase zero or phase two of EAP-FAST. If password aging requires a user to change passwords during phase zero, the new password would be effective in phase two.

EAP-FAST Benefits

EAP-FAST provides the following benefits over other authentication protocols:

- Mutual Authentication—The EAP server must be able to verify the identity and authenticity of the peer and the peer must be able to verify the authenticity of the EAP server.
- Immunity to passive dictionary attacks—Many authentication protocols require a password to be explicitly provided, either as clear text or hashed, by the peer to the EAP server.
- Immunity to man-in-the-middle (MitM) attacks—In establishing a mutually authenticated protected tunnel, the protocol must prevent adversaries from successfully interjecting information into the conversation between the peer and the EAP server.
- Flexibility to enable support for many different password authentication interfaces such as MSCHAPv2 and GTC, and others—EAP-FAST is an extensible framework that allows support of multiple internal protocols by the same server.
- Efficiency—When using wireless media, peers are limited in computational and power resources. EAP-FAST enables the network access communication to be computationally lightweight.
- Minimization of the authentication server's per user authentication state requirements—With large deployments, it is typical to have many servers acting as the authentication servers for many peers.

  It is better for a peer to use the same shared secret to secure a tunnel much the same way it uses the username and password to gain access to the network. EAP-FAST facilitates the use of a single strong shared secret by the peer while enabling servers to minimize the per-user and device state it must cache and manage.

EAP-FAST in ACS 5.8.1

ACS supports in-band provisioning of the peer with a shared secret credential (PAC) based on PKI or ADHP (phase 0). Authentication of the peer and allowing the peer access to the network is implemented in phase 1 and phase 2.

ACS 5.8.1 supports EAP-FAST versions 1 and 1a.

This section contains the following topics:

About Master-Keys

EAP-FAST master-keys are strong secrets that ACS automatically generates and of which only ACS is aware. Master-keys are never sent to an end-user client. EAP-FAST requires master-keys for two purposes:

- **PAC generation**—ACS generates PACs by using the active master-key. For details about PACs, see About PACs, page C-22.
- **EAP-FAST phase one**—ACS determines whether the PAC that the end-user client presents was generated by one of the master-keys it is aware of.

To increase the security of EAP-FAST, ACS changes the master-key that it uses to generate PACs. ACS uses Master Key Generation Period values that you define to determine when it generates a new master-key and the age of all master-keys.

An active master-key is the master-key used by ACS to generate PACs. The Master Key Generation Period setting determines the duration that a master-key remains active. At any time, only one master-key is active. For more information about how TTL values determine whether PAC refreshing or provisioning is required, see Master Key Generation and PAC TTLs, page C-26.

About PACs

PACs are strong shared secrets that enable ACS and an EAP-FAST end-user client to authenticate each other and establish a TLS tunnel for use in EAP-FAST phase two. ACS generates PACs by using the active master-key and a username.

PAC comprises:

- **PAC-Key**—Shared secret bound to a client (and client device) and server identity.
- **PAC Opaque**—Opaque field that the client caches and passes to the server. The server recovers the PAC-Key and the client identity to mutually authenticate with the client.
- **PAC-Info**—At a minimum, includes the Authority ID to enable the client to cache different PACs. Optionally, it includes other information such as the PACs expiration time.

An EAP-FAST end-user client stores PACs for each user accessing the network with the client. Additionally, a AAA server that supports EAP-FAST has a unique Authority ID. An end-user client associates a user's PACs with the Authority ID of the AAA server that generated them. PACs remove the need for PKI (digital certificates).

During EAP-FAST phase one, the end-user client presents the PAC that it has for the current user and Authority ID that ACS sends at the beginning of the EAP-FAST transaction. The means of providing PACs to end-user clients, known as PAC provisioning, are discussed in Automatic In-Band PAC Provisioning, page C-24 and Manual PAC Provisioning, page C-24.

Modifying the master key generation values does not affect already created PACs. Any modifications you make to the master key generation values specify the period when the next master keys are generated.

Provisioning Modes

ACS supports out-of-band and in-band provisioning modes. The in-band provisioning mode operates inside a TLS tunnel raised by Anonymous DH or Authenticated DH or RSA algorithm for key agreement.

To minimize the risk of exposing the user's credentials, a clear text password should not be used outside of the protected tunnel. Therefore, EAP-MSCHAPv2 or EAP-GTC are used to authenticate the user's credentials within the protected tunnel. The information contained in the PAC is also available for further authentication sessions after the inner EAP method has completed.

EAP-FAST has been enhanced to support an authenticated tunnel (by using the server certificate) inside which PAC provisioning occurs. The new cipher suites that are enhancements to EAP-FAST, and specifically the server certificate, are used.

At the end of a provisioning session that uses an authenticated tunnel, network access can be granted because the server and user have authenticated each other.

ACS supports the following EAP methods inside the tunnel for provisioning:

- EAP-MSCHAPv2
- EAP-GTC

By default, when you use EAP-MSCHAP inner methods, ACS allows authentication attempts up to the specified value you configured on the Service page inside the TLS tunnel if the initial authentication attempt fails. After the fourth failed authentication attempt inside the SSL tunnel, ACS terminates the EAP conversation, resulting in a RADIUS Access-Reject.

ACS supports issuing an out-of-band PAC file that allows you to generate a PAC that can be downloaded to ACS.

Types of PACs

ACS supports the following types of PACs:

- Tunnel v1 and v1a
- SGA
- Machine
- Authorization

ACS provisions supplicants with a PAC that contains a shared secret that is used in building a TLS tunnel between the supplicant and ACS. ACS provisions supplicants with PACs that have a wider contextual use.

The following types of PACs are provisioned to ACS, as per server policies:

- **Tunnel/Machine PAC**—Contains user or machine information, but no policy information.
- **User Authorization PAC**—Contains policy elements (for example, inner method used for user authentication). You can use the User Authorization PACs to allow a stateless server session to resume, as described in Session Resume, page C-16.

The various means by which an end-user client can receive PACs are:

- **PAC provisioning**—Required when an end-user client has no PAC. For more information about how master-key and PAC states determine whether PAC provisioning is required, see Master Key Generation and PAC TTLs, page C-26.

  The two supported means of PAC provisioning are:

  - **Automatic In-Band PAC Provisioning**—Sends a PAC by using a secure network connection. For more information, see Automatic In-Band PAC Provisioning, page C-24.
  - **Manual provisioning**—Requires that you use ACS to generate a PAC file for the user, copy the PAC file to the computer that is running the end-user client, and import the PAC file into the end-user client. For more information, see Manual PAC Provisioning, page C-24.

- **PAC refresh**—Occurs based on the value you specify in the Proactive PAC Update When field. For more information about how master-key and PAC states determine whether a PAC is refreshed, see Master Key Generation and PAC TTLs, page C-26.

PACs have the following two states, which the PAC TTL setting determines:

- **Active**—A PAC younger than the PAC TTL is considered active and can be used to complete EAP-FAST phase one.
- **Expired**—A PAC that is older than the PAC TTL is considered expired.At the end of EAP-FAST phase two, ACS generates a new PAC for the user and provides it to the end-user client.

Automatic In-Band PAC Provisioning

Automatic In-Band PAC Provisioning, which is the same as EAP-FAST phase zero, sends a new PAC to an end-user client over a secured network connection. Automatic In-Band PAC Provisioning requires no intervention of the network user or an ACS administrator, provided that you configure ACS and the end-user client to support Automatic In-Band PAC Provisioning.

Note Given that ACS associates each user with a single identity store, the use of Automatic In-Band PAC Provisioning requires that EAP-FAST users be authenticated with an identity store that is compatible with EAP-FAST phase zero. For the databases with which ACS can support EAP-FAST phase zero and phase two, see Authentication Protocol and Identity Store Compatibility, page C-36.

In general, phase zero of EAP-FAST does not authorize network access. In this general case, after the client has successfully performed phase zero PAC provisioning, the client must send a new EAP-FAST request in order to begin a new round of phase one tunnel establishment, followed by phase two authentication.

However, if you choose the Accept Client on Authenticated Provisioning option, ACS sends a RADIUS Access-Accept (that contains an EAP Success) at the end of a successful phase zero PAC provisioning, and the client is not forced to reauthenticate again. This option can be enabled only when the Allow Authenticated In-Band PAC Provisioning option is also enabled.

Because transmission of PACs in phase zero is secured by MSCHAPv2 authentication, when MSCHAPv2 is vulnerable to dictionary attacks, we recommend that you limit use of Automatic In-Band PAC Provisioning to initial deployment of EAP-FAST.

After a large EAP-FAST deployment, PAC provisioning should be done manually to ensure the highest security for PACs. For more information about manual PAC provisioning, see Manual PAC Provisioning, page C-24.

To control whether ACS performs Automatic In-Band PAC Provisioning, use the options on the Global System Options pages in the System Administration drawer. For more information, see EAP-FAST, page C-19.

Manual PAC Provisioning

Manual PAC provisioning requires an ACS administrator to generate PAC files, which must then be distributed to the applicable network users. Users must configure end-user clients with their PAC files.

You can use manual PAC provisioning to control who can use EAP-FAST to access your network. If you disable Automatic In-Band PAC Provisioning, any EAP-FAST user who is not provisioned with a PAC will not be able to access the network.

If your ACS deployment includes network segmentation, wherein a separate ACS controls access to each network segment, manual PAC provisioning enables you to grant EAP-FAST access on a per-segment basis.

For example, if your company uses EAP-FAST for wireless access in its Chicago and Boston offices and the Cisco Aironet Access Points at each of these two offices are configured to use different ACSs, you can determine, on a per-employee basis, whether Boston employees visiting the Chicago office can have wireless access.

While the administrative overhead of manual PAC provisioning is much greater than that of automatic in-band PAC provisioning, it does not risk sending the PAC over the network. Although manually provisioning the PACs requires a lot of effort early on, in configuring many end-user clients during the initial deployment, this type of provisioning is the most secure means for distributing PACs.

We recommend that, after a large EAP-FAST deployment, you manually perform PAC provisioning to ensure the highest security for PACs.

You can generate PAC files for specific usernames. You can also generate a PAC for a machine and provision the PAC manually to the client.

The following parameters are required to create a PAC:

- Specifying whether it is a user or machine PAC.
- Identity stored in Internal Identity Store ID field.
- PAC Time to Live (TTL).
- PAC encryption on or off, and password for encryption.

The PAC could be encrypted with the specified password by using the RC4 or AES algorithm. The detailed decryption algorithm must be provided to the client to allow decryption of the manually received PAC data.

ACS-Supported Features for PACs

ACS 5.8.1 support these features for PACs.

Machine PAC Authentication

Machine PAC-based authentication allows the machine to gain restricted network access before user authentication.

Proactive PAC Update

ACS proactively provides a new PAC to the client after successful authentication when a configured percentage of the PAC TTL remains. The tunnel PAC update is initiated by the server after the first successful authentication that is performed before the PAC expiration.

The proactive PAC update time is configured for the ACS server in the Allowed Protocols Page. This mechanism allows the client to be always updated with a valid PAC.

Note There is no proactive PAC update for Machine and Authorization PACs.

Accept Peer on Authenticated Provisioning

The peer may be authenticated during the provisioning phase.

PAC-Less Authentication

With PAC-less EAP-FAST Authentication, you can run EAP-FAST on ACS without issuing or accepting any tunnel or machine-generated PAC. The secure tunnel may be established by using a certificate rather than a PAC. Some PACs may be long-lived and not updated, which may cause authentication and security problems.

When PAC-less EAP-FAST is enabled, requests for PACs are ignored. Authentication begins with EAP-FAST phase zero and all subsequent requests for PACs are ignored. The flow moves on to EAP-FAST phase two. ACS responds with a Success-TLV message, without a PAC.

If a client attempts to establish a tunnel with a PAC, ACS responds with a PAC Invalid message. The tunnel establishment does not occur, and an Access-Reject is sent. The host or supplicant can reattempt to connect.

Anonymous phase zero, also known as ADHP is not supported for PAC-less authentication since the protocol does not support rolling over to phase two. PAC-less EAP-Fast supports configuration and does not require a client certificate.

Table C-3 displays the different types of PACs and the authentication and authorization methods you can use them for.

Table C-3 PAC Rules Summary

PAC Type	Tunnel v1/v1a/SGA	Machine	Authorization
Provide PAC on request on provisioning	Yes	Yes	Provide PAC on request on provisioning.
Provide PAC on request on authentication	Yes	Yes	Only if the PAC was not used in this authentication.
Proactive update	Yes	No	No
When PAC is expired	Reject, try to fall on TLS fall back, provide a new PAC after successful authentication only (tunnel PAC).	Reject, try to fall on TLS fall back, provide a new PAC after successful authentication only (machine PAC).	Reject and provide a new PAC after successful authentication only (authorization PAC).
Support ACS 3.x/4.x PACs	For Tunnel PAC v1/v1a only	Yes	No

Related Topics

- About PACs, page C-22
- Provisioning Modes, page C-22
- Types of PACs, page C-23
- Master Key Generation and PAC TTLs, page C-26

Master Key Generation and PAC TTLs

The values for master key generation and PAC TTLs determine their states, as described in About Master-Keys, page C-22 and Types of PACs, page C-23. Master key and PAC states determine whether someone requesting network access with EAP-FAST requires PAC provisioning or PAC refreshing.

Related Topics

- About PACs, page C-22

- Provisioning Modes, page C-22
- Types of PACs, page C-23
- ACS-Supported Features for PACs, page C-25

EAP-FAST for Allow TLS Renegotiation

You may be prompted to enter a password twice when you use an anonymous PAC provisioning schema. When you enter the password the first time, ACS provisions the PAC and sends an access-reject to the client. The client is then prompted to re-enter the password so that they will be able to authenticate and be granted access to the network.

ACS checks for a TLS client handshake record. If it finds the TLS client handshake record, ACS will initiate a TLS renegotiation at the end of EAP-Fast phase zero, instead of rejecting the user's request for access.

You should use this option with a Vista client when the host is using anonymous PAC provisioning. Vista client do not save the user password in the cache, so you are allowed to enter the password once. When this option is enabled, ACS initiates the TLS renegotiation request to the client at the end of EAP-FAST phase zero, instead of rejecting the access attempt after PAC provisioning.

EAP-FAST Flow in ACS 5.8.1.

Note You must configure the end-user clients to support EAP-FAST. This procedure is specific to configuring ACS only.

Before You Begin

The steps in this procedure are a suggested order only. Enabling EAP-FAST at your site may require recursion of these steps or performing these steps in a different order.

For example, in this procedure, determining how you want to support PAC provisioning comes after configuring a user database to support EAP-FAST; however, choosing Automatic In-Band PAC Provisioning places different limits on user database support.

To enable ACS to perform EAP-FAST authentication:

Step 1 Configure an identity store that supports EAP-FAST authentication.

To determine which identity stores support EAP-FAST authentication, see Authentication Protocol and Identity Store Compatibility, page C-36. For information about configuring identity stores, see Managing Users and Identity Stores, page 8-1

Step 2 Determine master key generation and PAC TTL values.

For information about how master key generation and PAC TTL values determine whether PAC provisioning or PAC refreshing is required, see Master Key Generation and PAC TTLs, page C-26.

Step 3 Determine whether you want to use automatic or manual PAC provisioning.

For more information about the two means of PAC provisioning, see Automatic In-Band PAC Provisioning, page C-24, and Manual PAC Provisioning, page C-24.

We recommend that you limit the use of Automatic In-Band PAC Provisioning to initial deployments of EAP-FAST, before you use manual PAC provisioning for adding small numbers of new end-user clients to your network and replacing PACs based on expired master keys.

Step 4 Using the decisions during Step 2Determine master key generation and PAC TTL values., page C-27 and Step 3Determine whether you want to use automatic or manual PAC provisioning., page C-27, enable EAP-FAST in the Global Systems Options drawer. See EAP-FAST, page C-19 for more information.

ACS is ready to perform EAP-FAST authentication.

Note Inner-identity will not be logged when: `the workstation not allowed` error appears, the SSL Handshake fails, EAP-PAC is provisioned, and ACS receives an invalid PAC.

Related Topics

- Managing Internal Identity Stores, page 8-4
- Managing External Identity Stores, page 8-29

EAP-FAST PAC Management

The EAP-FAST master-key in ACS is used to encrypt or decrypt, sign and authenticate the PACs and PAC-Opaque's that are used by EAP-FAST to store server opaque data by a supplicant. EAP-FAST requires a distributed mechanism by which each server in the ACS domain is able to pack and unpack PACs securely, including those which were packed on a different server.

The EAP-FAST master-key must have a common secret that is known to all servers in the ACS domain. The master-key is periodically refreshed and keys are replaced securely and synchronized by all ACS servers. The keys are generated of high entropy to comply with strong cryptographic standards such as FIPS-140.

In previous versions of ACS, the master-key was distributed by the ACS distribution mechanism and was replaced from time to time to improve the security of those keys. ACS 5.8.1 introduces a new scheme that provides simplicity, correctness, robustness, and security for master -key distribution.

The ACS EAP-FAST new distribution scheme contains a secure way of distributing the common seed-key, from which each ACS server can deterministically derive the same set of master-keys. Each PAC contains the information that the master-key was derived from, and each server can securely reconstruct the master-key that encrypted and signed the PAC.

This scheme improves the security by reducing the amount of cryptographic sensitive material that is transmitted.

This section contains the following topics:

- Key Distribution Algorithm, page C-28
- EAP-FAST PAC-Opaque Packing and Unpacking, page C-29
- Revocation Method, page C-29
- PAC Migration from ACS 4.x, page C-29

Key Distribution Algorithm

The common seed-key is a relatively large and a completely random buffer that is generated by the primary ACS server. The seed-key is generated only once during installation, or it can be manually regenerated by an administrator. The seed-key should rarely be replaced, because if you change seed-key, of all the previous master-keys and PACs would automatically be deactivated.

The seed-key is generated by using a FIPS approved RNG generator that exists in the runtime cryptographic module (CryptoLib). The ACS primary server management determines when to generate the seed-key, and communicates with the ACS runtime to request a new seed-key to be generated.

The size of the seed-key may vary and should consist of at least 64 bytes (512 bit). A larger seed might have some performance implication as each master-key derivation is dependent on it subsequently.

At any given time, a single seed-key should be used by each ACS server and the primary ACS server should ensure to distribute the latest seed-key to all the servers. Old seed-keys must discarded.

The seed-key contains critical cryptographic sensitive information. Disclosing the seed-key information would expose the entire EAP-FAST PAC mechanism to a large set of possible identity vulnerabilities.

Because of that, the mechanism which transports the seed-key between the primary and the secondary ACS servers must be fully secured. Further security measures must be taken with respect to storing the seed-key in the data-base. The seed-key should be protected with the strongest means of security.

EAP-FAST PAC-Opaque Packing and Unpacking

When the server generates a new PAC, it must derive the master-key to be used. When the server accepts a new PAC the same algorithm should be used for deriving the master-key with some additional verification used to prevent possible attacks on the master-key scheme. The derivation calculation may be skipped if the master-key was already placed in the cache in the past.

Revocation Method

You can revoke all PACs and all Master-Keys. For this type of extensive revocation, all you need to do is to revoke the seed-key and replace it by a new one.

Having only a single seed-key to be used in the system facilitates implementation.

PAC Migration from ACS 4.x

Although the configuration can be migrated from 4.x, the PACs themselves, as being stored only in supplicants, may still be issued from versions as far back as ACS 3.x. ACS 5.8.1 accepts PACs of all types according to migrated master-keys from versions 4.x and onwards, and re-issues a new 5.0 PAC, similar to the proactive PAC update for EAP-FAST 5.0.

When ACS 5.8.1, accepts a PAC from either ACS 3.x or 4.x, it decrypts and authenticates the PAC according to the 4.x master-key that was migrated from ACS 4.x configuration. The decryption and handling of this type of PAC is similar to the way the ACS 4.x PAC was handled.

The migration process involves converting the following data-items:

- EAP-FAST A-ID of ACS (Authority ID). The parameter replaces the deployment's A-ID of ACS 5.8.1.
- A list of retired ACS 4.x master-keys. The list is taken from the ACS 4.x configuration and placed in a new table in ACS 5.8.1. Each migrated master-key is associated with its expected time of expiration. The table is migrated along with the master-key identifier (index) and the PAC's-cipher assigned to each key.

EAP Authentication with RADIUS Key Wrap

You can configure ACS to use PEAP, EAP-FAST and EAP-TLS authentication with RADIUS Key Wrap. ACS can then authenticate RADIUS messages and distribute the session key to the network access server (NAS). The EAP session key is encrypted by using Advanced Encryption Standard (AES), and the RADIUS message is authenticated by using HMAC-SHA-1.

Because RADIUS is used to transport EAP messages (in the EAP-Message attribute), securely authenticating RADIUS messages ensures securely authenticated EAP message exchanges. You can use RADIUS Key Wrap when PEAP, EAP-FAST and EAP-TLS authentication is enabled as an external authentication method. Key Wrap is not supported for EAP-TLS as an inner method (for example, for EAP-FAST or PEAP).

RADIUS Key Wrap support in ACS uses three new AVPs for the cisco-av-pair RADIUS Vendor-Specific-Attribute (VSA); the TLV value of Cisco VSA is [26/9/1]):

- Random-Nonce—Generated by the NAS, it adds randomness to the key data encryption and authentication, and links requests and response packets to prevent replay attacks.
- Key—Used for session key distribution.
- Message-Authenticator-Code—Ensures the authenticity of the RADIUS message, including the EAP-Message and Key attributes.

While using RADIUS Key Wrap, ACS enforces the use of these three RADIUS Key Wrap AVPs for message exchanges and key delivery. ACS will reject all RADIUS (EAP) requests that contain both RADIUS Key Wrap AVPs and the standard RADIUS Message-Authenticator attribute.

To use RADIUS Key Wrap in PEAP, EAP-FAST and EAP-TLS authentications, you must enable the EAP authentication with RADIUS KeyWrap in the Network Devices and AAA Clients page or Default Network Device page.

You must also define two shared secret keys for each AAA Client. Each key must be unique and be distinct from the RADIUS shared key. RADIUS Key Wrap does not support proxy functionality, and should not be used with a proxy configuration.

EAP-MSCHAPv2

Microsoft Challenge Handshake Authentication Protocol (MSCHAP v2) provides two-way authentication, also known as mutual authentication. The remote access client receives verification that the remote access server that it is dialing in to has access to the user's password.

This section contains the following topics:

Overview of EAP-MSCHAPv2

Some of the specific members of the EAP family of authentication protocols, specifically EAP-FAST and PEAP, support the notion of an “EAP inner method.” This means that another EAP-based protocol performs additional authentication within the context of the first protocol, which is known as the “EAP outer method.”

One of the inner methods supported by the EAP-FAST and PEAP outer methods is EAP-MSCHAPv2, which is an adaptation of the MSCHAPv2 protocol that complies with the general framework established by EAP.

Using EAP-MSCHAPv2 as the inner EAP method facilitates the reuse of Microsoft directory technology (such as Windows Active Directory), with the associated database of user credentials for wireless authentication in the following contexts:

- MSCHAPv2 for User Authentication, page C-31
- MSCHAPv2 for Change Password, page C-31
- Windows Machine Authentication Against AD, page C-31

MSCHAPv2 for User Authentication

ACS supports the EAP-MSCHAPv2 authentication protocol as the inner method of EAP-FAST and PEAP. The protocol is an encapsulation of MSCHAPv2 into the EAP framework. Mutual authentication occurs against the configured credential database.

The client does not send its password, but a cryptographic function of the password. Using EAP-MSCHAPv2 as the inner method of tunneling protocols, increases protection of secured communication. Every protocol message is encrypted inside the tunnel and server, and client challenges are not generated randomly but, derived from outer method cryptographic material.

EAP-MSCHAPv2 is supported for AD and the ACS internal identity store.

MSCHAPv2 for Change Password

When you use EAP-MSCHAPv2 (as an EAP inner method) to authenticate a user whose password has expired, ACS sends a specific EAP-MSCHAPv2 failure notification to the client. The client can prompt the user for new password and then provide it to ACS inside the same conversation.

The new password is encrypted with the help of the old one. When a user password is changed successfully, the new user password is stored in the credential database.

EAP-MSCHAPv2 change password is supported for AD and ACS internal identity store.

Windows Machine Authentication Against AD

EAP-MSCHAPv2 can be used for machine authentication. EAP-MSCHAPv2 Windows machine authentication is the same as user authentication. The difference is that you must use the Active Directory of a Windows domain, since a machine password can be generated automatically on the machine and the AD, as a function of time and other parameters. The password generated cannot be stored in other types of credential databases.

EAP- MSCHAPv2 Flow in ACS 5.8.1

Components involved in the 802.1x and MSCHAPv2 authentication process are the:

- Host—The end entity, or end user's machine.
- AAA client—The network access point.
- Authentication server—ACS.

The MSCHAPv2 protocol is described in RFC 2759.

Related Topic

- Authentication Protocol and Identity Store Compatibility, page C-36

CHAP

CHAP uses a challenge-response mechanism with one-way encryption on the response. CHAP enables ACS to negotiate downward from the most secure to the least secure encryption mechanism, and it protects passwords that are transmitted in the process. CHAP passwords are reusable.

If you are using the ACS internal database for authentication, you can use PAP or CHAP. CHAP does not work with the Windows user database. Compared to RADIUS PAP, CHAP allows a higher level of security for encrypting passwords when communicating from an end-user client to the AAA client.

LEAP

ACS currently uses LEAP only for Cisco Aironet wireless networking. If you do not enable this option, Cisco Aironet end-user clients who are configured to perform LEAP authentication cannot access the network. If all Cisco Aironet end-user clients use a different authentication protocol, such as EAP-TLS, we recommend that you disable this option.

Note If users who access your network by using a AAA client that is defined in the Network Configuration section as a RADIUS (Cisco Aironet) device, then you must enable LEAP, EAP-TLS, or both; otherwise, Cisco Aironet users cannot authenticate.

Certificate Attributes

ACS parses the following client certificate's attributes:

- Certificate serial-number (in binary format)
- Encoded certificate (in binary DER format)
- Subject's CN attribute
- Subject's O attribute (Organization)
- Subject's OU attribute (Organization Unit)
- Subject's L attribute (Location)
- Subject's C attribute (Country)
- Subject's ST attribute (State Province)
- Subject's E attribute (eMail)
- Subject's SN attribute (Subject Serial Number)
- Issuer I attribute
- SAN (Subject Alternative Name)

You can define a policy to set the principle username to use in the TLS conversation, as an attribute that is taken from the received certificate.

The attributes that can be used as the principle username are:

- Subject CN
- Subject Serial-Number (SN)
- SAN
- Subject
- SAN—Email
- SAN—DNS
- SAN—otherName

If the certificate does not contain the configured attribute, authentication fails.

Note ACS 5.8.1 supports short hard-coded attributes and certificate attribute verification for the only the EAP-TLS protocol.

Certificate Binary Comparison

You can perform binary comparison against a certificate that ACS receives from an external identity store and determine the identity store's parameters that will be used for the comparison.

Note In ACS 5.8.1, AD and LDAP are the only external identity stores that hold certificates.

ACS uses the configured principle username to query for the user's certificate and then perform binary comparison between the certificate received from external identity store and the one received from the client. The comparison is performed on a DER certificate format.

Rules Relating to Textual Attributes

ACS collects client certificate textual attributes and places them in the ACS context dictionary. ACS can apply any rule based policy on these attributes as with any rule attributes in ACS.

The attribute that can be used for rule verification are:

- Subject's CN attribute
- Subject's O attribute (Organization)
- Subject's OU attribute (Organization Unit)
- Subject's L attribute (Location)
- Subject's C attribute (Country)
- Subject's ST attribute (State Province)
- Subject's E attribute (eMail)
- Subject's SN attribute (Subject Serial Number)
- Issuer I attribute
- SAN (Subject Alternative Name)
- Subject

- SAN—Email
- SAN—DNS
- SAN—otherName

Certificate Revocation

Every client certificate that ACS receives is verified with a Certificate Revocation List (CRL) according to a policy that is defined.

The CRL mechanism verifies whether or not you can still rely on a client certificate. This is done by checking the serial number of the certificate, and that of each member of the corresponding certificate chain, against a list of certificates that are known to have been revoked.

Possible reasons for revocation of a certificate include suspicion that the associated private key has been compromised or the realization that the certificate was issued improperly. If either of these conditions exist, the certificate is rejected.

ACS supports a static-CRL that contains a list of URLs used to acquire the CRL files that are configured in ACS database.

Note ACS does not support delta CRLs in certificate revocation validation.

You can configure a set of URLs used for CRL update for each trusted CA certificate,. By default, when adding a CA certificate, ACS automatically sets all the URLs stored in the certificate *crlDistributionPoint* as the initial static CRL for that CA. In most cases, the *crlDistributionPoint* is used to point to the CRL location used to revoke the CA certificate, but you can edit the URL to point to the CRL file issued by this CA. You can only configure a single HTTP based URL for each CA.

You can configure the parameters for each CA, which will apply to all the URLs that are configured to the CA. ACS supports two download modes, one for periodic download, and the other for downloading the next CRL update just before the previous is about to expire.

- For the periodic download, you can define the download periods.
- For automatic downloading, you define the amount of time before the CRL file expires, should ACS download it. The CRL expiration time is taken from the *CRL nextUpdate* field.

For both modes, if the download somehow fails, you can define the amount of time that ACS will wait before trying to redownload the CRL file.

ACS verifies that the downloaded CRL file is signed correctly by any one of the CAs in the trust store, for each downloaded CRL file and whether they are trusted. ACS uses the CRL file only if the signature verification passes. The verified CRL file replaces the previous CRL file issued by the same CA.

Note CRL files are not kept persistent, and should be re-downloaded when you restart ACS.

The configuration of URLs and their association to CA's is distributed to the entire ACS domain. The downloaded CRLs are not distributed and are autonomously populated in parallel in each ACS server.

Machine Authentication

ACS supports the authentication of computers that are running the Microsoft Windows operating systems that support EAP computer authentication. Machine authentication, also called computer authentication, allows networks services only for computers known to Active Directory.

This feature is especially useful for wireless networks, where unauthorized users outside the physical premises of your workplace can access your wireless access points.

When machine authentication is enabled, there are three different types of authentications. When starting a computer, the authentications occur in this order:

- **Machine authentication**—ACS authenticates the computer prior to user authentication. ACS checks the credentials that the computer provides against the Windows identity store.

  If you use Active Directory and the matching computer account in AD has the same credentials, the computer gains access to Windows domain services.

- **User domain authentication**—If machine authentication succeeded, the Windows domain authenticates the user. If machine authentication failed, the computer does not have access to Windows domain services and the user credentials are authenticated by using cached credentials that the local operating system retains.

  In this case, the user can log in to only the local system. When a user is authenticated by cached credentials, instead of the domain, the computer does not enforce domain policies, such as running login scripts that the domain dictates.

Note If a computer fails machine authentication and the user has not successfully logged in to the domain by using the computer since the most recent user password change, the cached credentials on the computer will not match the new password. Instead, the cached credentials will match an older password of the user, provided that the user once successfully logged in to the domain from this computer.

- **User network authentication**—ACS authenticates the user, allowing the user to have network connectivity. If the user exists, the identity store that is specified is used to authenticate the user.

  While the identity store is not required to be the Windows identity store, most Microsoft clients can be configured to automatically perform network authentication by using the same credentials used for user domain authentication. This method allows for a single sign-on.

Note Microsoft PEAP clients may also initiate machine authentication whenever a user logs off. This feature prepares the network connection for the next user login. Microsoft PEAP clients may also initiate machine authentication when a user shuts down or restarts the computer rather than just logging off.

ACS supports EAP-TLS, EAP-FAST, PEAP (EAP-MSCHAPv2), and PEAP (EAP-GTC) for machine authentication. You can enable each separately on the Active Directory: General Page, which allows a mix of computers that authenticate with EAP-TLS, EAP-FAST, or PEAP (EAP-MSCHAPv2).

Microsoft operating systems that perform machine authentication might limit the user authentication protocol to the same protocol that is used for machine authentication.

Related Topics

- Microsoft AD, page 8-52
- Managing External Identity Stores, page 8-29

Authentication Protocol and Identity Store Compatibility

ACS supports various authentication protocols to authenticate against the supported identity stores.

Table C-4 specifies non-EAP authentication protocol support.

Table C-4 Non-EAP Authentication Protocol and User Database Compatibility

Identity Store	ASCII/PAP	MSCHAPv1/MSCHAPv2	CHAP
ACS	Yes	Yes	Yes
Windows AD	Yes	Yes	No
LDAP	Yes	No	No
RSA Identity Store	Yes	No	No
RADIUS Identity Store	Yes	No	No

Table C-5 specifies EAP authentication protocol support.

Table C-5 EAP Authentication Protocol and User Database Compatibility

Identity Store	EAP-MD5	EAP-TLS [1]	PEAP-TLS [2]	PEAP EAP-MSCHAPv2	EAP-FAST MSCHAPv2	PEAP-GTC	EAP-FAST-GTC
ACS	Yes	Yes[3]	Yes	Yes	Yes	Yes	Yes
Windows AD	No	Yes	Yes	Yes	Yes	Yes	Yes
LDAP	No	Yes	Yes	No	No	Yes	Yes
RSA Identity Store	No	No	No	No	No	Yes	Yes
RADIUS Identity Store	No	No	No	No	No	Yes	Yes

1. In EAP-TLS authentication, the user is authenticated by cryptographic validation of the certificate. Additionally, ACS 5.8.1 optionally allows a binary comparison of the user's certificate sent by the end-user client against the certificate located in the user's record in the LDAP identity store.
2. In PEAP-TLS authentication, the user is authenticated by cryptographic validation of the certificate. Additionally, ACS 5.8.1 optionally allows a binary comparison of the user's certificate sent by the end-user client against the certificate located in the user's record in the LDAP identity store.
3. ACS Identity Store cannot store the certificates.