

LAN Security Guide

Version 1.0



Compiled by

Andrew Riddell

LAN Security Guide

First Edition

Andrew Riddell

About Allied Telesis, Inc.

Allied Telesis is a world class leader in delivering IP/Ethernet network solutions to the global marketplace. We create innovative, standards-based IP networks that seamlessly connect users with their voice, video and data services. We are an international company headquartered in Japan with major divisions in Europe, Asia and North and South America. Our partners include the world's largest distributors, integrators, solution providers and resellers to assure you receive immediate local service and support. Allied Telesis has been designing, manufacturing and selling networking products for over 20 years. Our philosophy of producing products of the highest quality, at affordable prices, has resulted in Allied Telesis products being deployed in networks of all types and sizes across the world. Our proven track record of providing solid technology, excellent support and full feature products has allowed Allied Telesis to become the worldwide de-facto standard in many areas of technology. With a portfolio of products that can provide end-to-end networking for both Service Provider, Enterprise and SMB customers, Allied Telesis is the natural choice for many world class organizations.

Please visit us online at alliedtelesis.com

Copyright © 2012 Allied Telesis Inc.

All rights reserved. This documentation is subject to change without notice. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's internal use without the written permission of Allied Telesis, Inc.

Trademarks

Allied Telesis, AlliedWare Plus, EPSRing, SwitchBlade, and VCStack are trademarks or registered trademarks in the United States and elsewhere of Allied Telesis, Inc. Adobe, Acrobat, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Additional brands, names, and products mentioned herein may be trademarks of their respective companies.

Warning and Disclaimer

The information in this guide is provided on an "as is" basis. The author and the publishers shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this guide.

Co-authors

Andrew Darrell
Cameron Drewett
Wayne Sangster

Technical Writer

Carin van Bolderen

Illustrators

Angela Maxwell
Carin van Bolderen

Editors

Carin van Bolderen
Rebecca Officer

Review Team

Andrew Riddell
Rebecca Officer
Miranda de Gouw
Mikhail Ginodman

Cover Designer

Angela Maxwell

Produced in New Zealand
First Printing: November 2011
Document Number:
C613-04005-00-REV C

ISBN: 978-0-473-19020-0

About this Guide

One of the guiding principles of good network design is security. Network owners need to protect the data that is stored within their network. Moreover, organizations have become as dependent on the assurance of network availability as they are on the assurance of electricity or water supply. A network must be secure against data theft and against denial of service.

This Guide focuses on security within a LAN—protection against 'the enemy within'. Protection against external attack and data theft occurring outside of the LAN, such as Firewalling and Virtual Private Networking, are beyond the scope of this Guide.

The material covered is a mixture of practical advice on secure configuration of network devices, and explanations of the protocols and technologies that are employed in LAN security.

The three prime aspects of LAN security are:

1. Ensuring networking equipment is able to block attacks aimed at compromising or disabling it.
2. Ensuring networking equipment is able to combat attacks that exploit weaknesses in LAN protocols. These are attacks that are not aimed at compromising the networking devices themselves, but rather are attacks that disrupt data flow in the network or use tricks to give users access to sectors of the network that they should not be able to access.
3. Preventing unauthorized users from getting any access to the network.

The subject matter of each of the chapters in this Guide relate to one or more of these activities.

First we consider how to configure switches in a way that enables them to protect themselves against attacks directed at them. This is a combination of secure configuration of the services that run on the switch, and the use of ACLs to block undesirable traffic. We then discuss how to combat well-known LAN-based attacks such as Root bridge spoofing, MAC-flooding, DoS, and VLAN hopping attacks.

This is followed by several chapters that relate to various aspects of user authentication. The first is a discussion of the configuration framework, called AAA, that is used to unify all the configuration of user authentication and user activity monitoring within a switch. This is followed by an explanation of a key user authentication and data protection technology—Private/Public keys and X.509 certificates. This use of this technology has become widespread, not just in LAN security, but in secure online transactions, and the distribution of software. This chapter provides an easy to follow explanation of what Private/Public keys are and how they work hand-in-hand with X.509 certificates to provide a very robust framework of trust and security.

The next two chapters describe the two most popular protocols that underlie the user authentication and user activity monitoring process—RADIUS and TACACS+. Each of the protocols is explained in some detail.

We then discuss the three LAN user authentication mechanisms used in AlliedWare Plus™ : 802.1x, Web authentication, and MAC authentication. The discussion of MAC authentication, in fact, is subsumed into a chapter that presents the concept of tri-authentication - combining all three authentication mechanisms to provide a comprehensive solution that can enforce authentication of all network-attached devices. The examination of LAN user authentication takes in related features like dynamic VLAN allocation, Guest VLAN, auth-fail VLAN, and re-authentication timers.

The final chapter focuses on DHCP snooping, which is a powerful defence against address spoofing and DHCP-based attacks. DHCP snooping blocks unauthorized IP traffic from untrusted ports, and prevents it from entering the trusted network. We explain how DHCP snooping focuses on security tasks, unlike DHCP relay which is an integral part of the DHCP lease distribution process.

Contents

About this Guide	iii
Chapter 1 Securing the Switch	1
Introduction	1
Secure Configuration of Management Services	2
SSH	2
Secure Web management	4
SNMPv3	5
Telnet	6
Interoperation with management stations	6
Configuration Script for Securing Management Services	7
Configuration to Aid Network Monitoring	8
Logging	8
Secure configuration of NTP	8
Configuration Script for Aiding Secure Network Monitoring	10
Protecting the CPU	11
Control plane bandwidth control	11
Using hardware ACLs to protect the CPU from IPv4 attacks	11
Using hardware ACLs to protect the CPU from IPv6 attacks	13
Using Hardware ACLs to Block Packets from Invalid Source IP Addresses	14
Configuration Script for ACLs	15
Secure Configuration of Routing Protocols	16
OSPF	16
RIP	16
VRRP	17
Configuration Script for Securing Routing Protocols	18
Chapter 2 Blocking Network Attacks	19
Introduction	19
Mitigating common network attacks	20
Secure Configuration of Spanning Tree Protocol	24
Protecting against Root Bridge spoofing attacks	24
Protecting Against MAC Flooding Attacks	26
MAC flooding attack	26
Protecting Against Denial of Service Attacks in the LAN	27
DoS attack prevention	27
Good Practice for Switch Security	28
Configuration Script for Blocking Network Attacks	30
Chapter 3 AAA	31

Introduction	31
AAA Commands	32
Server Groups	32
Method Lists	33
Configuring method lists	33
Accounting method lists	35
Applying login method lists	35
Processing Authentication Requests	36
Checking multiple authentication servers	36
Generic authentication features	37
Critical ports	37
Quiet period	38
Roaming authentication	39
Roaming authentication in wireless and wired environments	40
Chapter 4 X.509 Certificates	43
Introduction	43
Aspects of Security	44
Encryption	44
Asymmetric encryption	45
Validation	47
Certificate signing	49
Tamper prevention	51
Example – 802.1x Authentication with X.509 Certificates	52
Creating the server and user certificates using AlliedWare Plus local RADIUS server ..	52
Configuring ports as authenticator ports	54
Installing certificates on the supplicant workstation	54
Setting up the PC's NIC card as an 802.1x supplicant	62
Attaching the PC NIC to the switch	63
Using X.509 Certificates to Authenticate a VPN Tunnel	64
Solution overview	64
Summary of trust relationship that the certificates create	65
Secure session key exchange	66
Chapter 5 RADIUS	67
Introduction	67
RADIUS Overview	68
Original use of RADIUS - PPP dialup	68
RADIUS terminology and components	69
RADIUS packet exchange	70
TLV format	73
Security aspects of the RADIUS protocol	73
Extensions in later RFCs	77

RADIUS proxy	77
RADIUS accounting	78
RADIUS accounting on AlliedWare Plus	80
Chapter 6 TACACS PLUS	81
Introduction	81
Evolution from TACACS to TACACS+	81
Overview of TACACS+	82
Protocol description	82
Authentication	84
User Access-Request sessions	86
Password change sessions	86
Authentication credential request sessions	87
Authorization	87
Accounting	88
AlliedWare Plus Implementation of TACACS+	89
AlliedWare Plus TACACS+ configuration example	90
Chapter 7 Web-Authentication	93
Introduction	93
What is Web-authentication?	94
Web-authentication Basics	94
Configuring Web-authentication	96
Choosing the Web-authentication server address	96
Starting a Web-authentication Session	98
Understanding the Web-authentication Features	100
Support for protocols underlying Web-authentication	100
Secure Authentication	105
Copying a certificate onto the switch	105
Ping-poll Monitoring of Supplicant Presence	107
Checking the IP addresses of the supplicants	108
Ping-poll and promiscuous mode	108
Managing Traffic of Unauthenticated Supplicants	109
No Guest VLAN or Auth-fail VLAN	109
Guest VLAN but no Auth-fail VLAN	110
Auth-fail VLAN, but no Guest VLAN	111
Auth-fail VLAN, and Guest VLAN	111
Monitoring the Operation of Web-authentication	112
Chapter 8 802.1x	113
Introduction	113
The role of 802.1x in networks	114

What is 802.1x?	114
802.1x System Components	114
802.1x component protocols	115
Basic steps in an 802.1x conversation	116
Example message sequence	116
Some important concepts	117
Understanding EAP	117
Evolution of the protocol	117
Communication with the authentication server	119
Some new terminology	121
Levels of Increasing Security in 802.1x	122
Protocol layering	125
Table of authentication types	127
Configuring 802.1x	128
Multi-supplicant modes	131
Timers	133
802.1x VLAN Assignment	134
Dynamic VLAN assignment	134
VLAN assignment application	137
Dynamic VLAN assignment with multiple supplicants	139
Using a guest VLAN	142
Verify the operation of 802.1x	143
Troubleshooting	144
Simple Definitions for 802.1x Port Authentication	146
Chapter 9 Tri-authentication	147
Introduction	147
Why is MAC-based Authentication Required?	148
How does MAC-based authentication work?	148
Configuring MAC authentication	149
A superior alternative to legacy MAC-based VLANs	150
Tri-authentication operation	150
Multiple supplicants on tri-authenticating ports	151
Chapter 10 DHCP Snooping	153
Introduction	153
What is DHCP snooping?	154
Filtering packets based on information gathered by DHCP snooping	161
Basic DHCP snooping configuration in AlliedWare Plus	163
Monitoring using SNMP	164

Chapter I | Securing the Switch

Introduction

Increasingly, we see the deployment of switched networks in the Enterprise and the use of switches in other environments—alongside motorways, around cities, within factories etc. Security on these switches is as important as on servers and end user computer equipment.

A breach in security on a core switch can bring a large network to a complete standstill. When installing networking equipment into an environment that exposes it to unauthorized access attempts and malicious attacks, it is important that it is configured in a way that blocks attacks.

This section gives best practice guidelines for configuring switches for maximum protection against attacks, and maximum network stability. It includes descriptions of how to:

- securely configure management services that must be available, and disable ones that are not required.
- use hardware filters to block undesirable traffic.
- ensure that networks protocols are secure.

List of terms

SNMPv3

Simple Network Management Protocol. Version 3 primarily added security and enhancements to SNMP

AAA

AAA is the collective title for the three related functions of Authentication, Authorization and Accounting.

Control Plane Bandwidth Control

This controls the rate that the data can access the CPU of a switch, to prevent the operation of the software being affected by malicious or inadvertent floods of data.

Secure Configuration of Management Services

Secure switch management requires the use of encrypted management protocols. AlliedWare Plus™ provides the following methods for secure remote device management.

- Secure Shell (SSH) on [page 2](#)
- Secure Web management on [page 4](#)
- Simple Network Management Protocol v3 (SNMPv3) on [page 5](#)
- Secure configuration of NTP on [page 8](#)

Let us look at how each of these methods of remote access can be configured securely. To see the commands in this section in script format, see "[Configuration Script for Securing Management Services](#)" on [page 7](#).

SSH

Using SSH, it is possible to have encrypted access to the switch's command line interface.

1. Configure the SSH service.

```
awplus(config)# crypto key generate hostkey rsa
awplus(config)# service ssh
```

To check that the service is running, use the following command: **show ssh server**

```
awplus#show ssh server

Secure Shell Server Configuration
-----
SSH Server                : Enabled
Protocol                  : IPv4,IPv6
Port                      : 22
Version                   : 2,1
Services                  : scp, sftp
User Authentication       : publickey, password
Resolve Hosts             : Disabled
Session Timeout           : 0 (Off)
Login Timeout             : 60 seconds
Maximum Startups          : 10
Debug                    : NONE
```

Authenticating users connecting to SSH sessions

Users connecting to the switch by SSH can be authenticated in one of these ways:

- checking their credentials in the local user database on the switch.
- sending their credentials to a RADIUS or TACACS+ server to be checked.
- using the RADIUS or TACACS+ server if available, or the local user database if the RADIUS or TACACS+ server does not respond.

By default, SSH users are checked against the **local** user database on the switch.

2. Enable SSH connections to be authenticated by the local user database.

Add a user to the local user database:

```
awplus(config) # username <name> password <password>
```

Creating a user in the local user database does not automatically enable that user to log into the switch by SSH. They need to be explicitly allowed SSH access:

```
awplus(config) # ssh server allow-users <username>
```

3. Enable SSH connections to be authenticated by RADIUS or TACACS.

Configure the switch with the address (and optionally UDP port) of a RADIUS server:

```
awplus(config) # radius-server host <ip-address> auth-port 1812
key <secret-key>
```

Configure an AAA authentication method list for login that uses the RADIUS server.

There are two options:

- Set the switch to use only the RADIUS server to authenticate all attempts to log into the switch's CLI:

```
awplus(config) # aaa authentication login default group radius
```

If the RADIUS server does not respond, then the login attempt fails.

- Configure the switch to query the RADIUS server first, then use the local user database as a backup option if the RADIUS server does not respond:

```
awplus(config) # aaa authentication login default group radius
local
```

If the RADIUS server rejects the login because it is not configured with a matching user, the login fails. If the RADIUS server is not available or does not recognise the switch as a RADIUS client, it will not respond, and the switch will check the local user database. If you use the local user database as a backup, you must also add the user to the local user database, as described above.

Similarly, the switch can be configured with the details of TACACS+ servers:

```
awplus(config) # tacacs-server host {<host-name>|<ip-address>}
[key [8<sup>1</sup>]<key-string>]
```

and an AAA authentication login definition can specify the TACACS-server group as the authentication method:

```
awplus(config) # aaa authentication login default group tacacs+
```

1. [8] Specifies that you are entering a password as a string that has already been encrypted instead of entering a plain text password. The running config displays the new password as an encrypted string even if password encryption is turned off.

Note that there is just one default AAA authentication method list that covers all logins to the switch's CLI. As described in the section ["Applying login method lists" on page 35](#), it is possible to create another named login method list for Telnet and SSH that is different to that used for console login. It is not possible for SSH to have a different method list from Telnet.

Secure Web management

Web access to a switch running AlliedWare Plus is achieved as follows:

1. Initially, the web browser connects to the switch by HTTP.
2. Via this HTTP connection, the web browser downloads a Java applet.
3. The rest of the management session from the web browser to the switch is controlled by the Java applet. The Java applet uses a combination of SNMPv3 and remote CLI sessions to exchange information with the switch.

After the initial download of the Java applet, there is very little HTTP exchanged between the web browser and the switch. In particular, no user authentication, switch monitoring or configuration information is exchanged by HTTP. All that information is exchanged by SNMPv3 or remote CLI sessions. As a result, it is not important to encrypt the HTTP connection. However, AlliedWare Plus does have the option of connecting to the GUI via HTTPS.

Note: The SSL certificate used for HTTPS is normally self signed, and will result in a warning message from a standard web browser.

The SNMPv3 user via which the Java applet communicates with the switch is forced to use both authentication and encryption. There is no option to use a user that implements the **noauth** or **auth** levels of security; the user **must** implement the **priv** security level. The SNMPv3 aspects of the web management session are inherently secure. To ensure that the remote CLI connection involved in the web management session is secure, you must enable SSH on the switch. If SSH is not enabled, the Java applet uses Telnet but as soon as the SSH service is enabled on the switch, the Java applet will stop using Telnet and use SSH instead.

Configure the switch for secure web browser management.

1. Create a GUI user.
(With software version 5.4.1-2.6 and higher, this step is not necessary.)

```
awplus(config)# username <gui-user-name> privilege 15 guiuser  
password <password>
```
2. Disable Telnet.

```
awplus(config)# no service telnet
```
3. The GUI uses SSHv1, so a host key for SSHv1 is required.

```
awplus(config)# crypto key generate hostkey rsa1
```
4. The SSH service also requires a host key for SSHv2.

```
awplus(config)# crypto key generate hostkey rsa
```

5. Register the GUI user as an SSH client.

```
awplus(config)# ssh server allow-users <gui-user-name>
```

6. Enable the SSH service.

```
awplus(config)# service ssh
```

SNMPv3

The SNMPv3 protocol provides the opportunity to configure SNMP in a much more secure way than was possible with previous versions of SNMP. In particular, with SNMPv3, you can:

- encrypt the SNMP messages being sent across the network.
- check that SNMP messages are not tampered with during transit across the network.
- set up restricted views, that is, limited sets of MIB variables that can be accessed by particular users. These users need to enter a password to get access to their view.

Configuring SNMPv3

The following steps show a typical SNMPv3 configuration:

Note: By default, the switch does **not** respond to SNMP messages, so if you do not wish to use SNMP, there is no need to enter any commands to disable it on the switch.

1. Configure SNMPv3.

In this example the user is named `secure-user`, the group is named `secure`, and the authentication algorithm is SHA (Secure Hash Algorithm):

```
awplus(config)# snmp-server host 172.28.76.128 version 3 priv
secure-user

awplus(config)# snmp-server group secure priv

awplus(config)# snmp-server user secure-user secure auth sha
<hash-password> priv des <encrypt-password>
```

Note that in the **snmp-server group** and **snmp-server host** commands, the security level can have three possible settings:

1. **noauth**—performs no authentication and no encryption
2. **auth**—performs authentication, but no encryption
3. **priv**—performs authentication and encryption, which provides the best security

2. Configure notifications (traps).

```
awplus(config)# snmp-server enable trap auth lldp loopprot mstp
nsm rmon vcs vrrp
```


Some notifications must also be enabled within their protocols. For example, Link Layer Discovery Protocol (LLDP) notifications are enabled using the commands **lldp notifications** and/or **lldp med-notifications** as well as the command above.

3. Configure a restricted view for an SNMPv3 group.

By default (and using the commands above), an SNMPv3 user is able to receive all SNMP Object IDs (OID) that are supported by the switch, and to set all the settable OIDs supported by the switch. However, you can restrict the range of OIDs that a user can receive and/or set by using an SNMPv3 view.

For example, a view can be created that includes all the OIDs in the tree from 1.3.6.1.2.1, but specifically excludes the sub-tree below 1.3.6.1.2.1.4:

```
awplus(config)# snmp-server view v1 1.3.6.1.2.1 included
awplus(config)# snmp-server view v1 1.3.6.1.2.1.4 excluded
```

The view is applied to a group. To restrict all users in a group to only be able to get the OIDs allowed by the view, configure the view as a **read view** for the group:

```
awplus(config)# snmp-server group secure priv read v1
```

Similarly, to restrict all the users in a group to only be able to set the OIDs allowed by the view, configure the view as a **write view** on the group:

```
awplus(config)# snmp-server group secure priv write v1
```

Telnet

Telnet access to the switch is enabled by default. We recommend that you disable Telnet, and instead use SSH for command line remote access.

```
awplus(config)# no service telnet
```

Interoperation with management stations

While the configuration described above is aimed at very secure switch management, it may not always be possible to impose such a high level of security. In particular, the requirement to interoperate with certain management tools that do not support the most secure protocols may require a different configuration. For example, using a management tool that uses Telnet rather than SSH, you must **remove** the command **no service telnet** from your configuration. Similarly, using a management tool that requires SNMPv1 or SNMPv2c communication, rather than SNMPv3, then you must configure SNMP communities.

Configure SNMP communities for SNMPv1 or SNMPv2c.

```
awplus(config)# snmp-server community <name1> ro
awplus(config)# snmp-server community <name2> rw
```

To restrict access, configure ACLs as described in ["Using hardware ACLs to protect the CPU from IPv4 attacks" on page 11](#).

Configuration Script for Securing Management Services

This section provides all the commands from the previous sections, with brief comments included, in a format that can be copied and modified to create parts of a configuration script file. To use these commands in a configuration script, copy the relevant lines, comment out (with an "!") or delete lines you do not need, and replace values with ones appropriate to your network.

```
!
! SNMP
!
! Configure SNMPv3.
snmp-server host 172.28.76.128 version 3 priv secure-user
snmp-server group secure priv
snmp-server user secure-user secure auth sha <hash-password> priv des
<encrypt-password>
!
! Configure notifications (traps).
snmp-server enable trap auth lldp loopprot mstp nsm rmon vcs vrrp
!
! Configure a restricted view for an SNMPv3 group.
snmp-server view v1 1.3.6.1.2.1 included
snmp-server view v1 1.3.6.1.2.1.4 excluded
snmp-server group secure priv read v1
snmp-server group secure priv write v1
!
! Telnet
! Disable Telnet.
no service telnet
!
! Secure Shell (SSH)
! Configure the SSH service.
crypto key generate hostkey rsa
crypto key generate hostkey rsa1
service ssh
! Enable SSH connections to be authenticated by the local user database.
username <name> password <password>
ssh server allow-users <username>
! Enable SSH connections to be authenticated by RADIUS.
radius-server host <ip-address> auth-port 1812 key <secret-key>
aaa authentication login default group radius local
!
! Secure configuration of web access
! Configure the switch for secure web-browser management.
username <gui-user-name> privilege 15 guiuser password <password>
! GUI-User not required with software version 5.4.1-2.6 and higher.
```

Configuration to Aid Network Monitoring

Logging

It is highly advisable to log all activity on the switch to a syslog server, as this will provide a detailed audit trail in the event of a suspected security breach, or other problem. A suitable configuration can be seen above on [page 7](#).

Secure configuration of NTP

NTP, like any service on the switch, is vulnerable to attack. The simplest attack is a Denial of Service attack—flooding the NTP module with requests or updates. A more sophisticated attack involves spoofing a time source, thereby providing an incorrect time to the switch when the switch is acting as an NTP client.

AlliedWare Plus provides two mechanisms for guarding against NTP attack: NTP filtering and NTP authentication.

NTP filtering

Filtering makes use of access lists to specify the IP addresses with which the switch's NTP process will interact. A number of different types of access list can be applied to the NTP module, to control the different types of relationship that can occur in NTP. Access lists can include the following types:

peer

Time requests and NTP control queries will be accepted from devices whose addresses pass this access list. The switch's NTP process is able to synchronize itself to a device whose address passes this access list.

query-only

NTP control queries are accepted from devices whose addresses pass this access list.

serve

Time requests and NTP control queries will be accepted from devices whose addresses pass this access list. The switch's NTP process is **not** able to synchronize itself to a device whose address passes this access list.

serve-only

Only time requests are accepted from devices whose addresses pass this access list. The access lists are applied using the command:

```
awplus(config)# ntp access-group [peer|query-only|serve|
serve-only] [<1-99>|<1300-1999>]
```

where <1-99> or <1300-1999> is the ID of a standard IP access list that has been created using the command:

```
awplus(config)# access-list {<1-99>|<1300-1999>} {deny|permit}
<source>
```

NTP authentication

The purpose of NTP authentication is to enable the client to authenticate the server, and not vice versa. NTP authentication specifically deals with malicious users attempting to spoof a valid NTP server. The authentication is performed by using an MD5 key. The server and the client must be both configured to perform authentication and to use the same MD5 key.

Configuring authentication on the NTP client

On the client, there are four commands required to configure authentication:

1. Enable authentication for NTP

```
awplus(config)# ntp authenticate
```

2. Create one or more MD5 keys that can be used for NTP authentication

```
awplus(config)# ntp authentication-key <keynumber> md5 <key>
```

where the *<keynumber>* is an ID number that will be used in other commands to refer to this key.

The *<key>* is just a string, for example: AB123434, and is limited to the maximum command line length, which is 464 characters without spaces (less the length of the preceding parameters).

3. Create a list of which of these keys is currently trusted (i.e. can currently be used for authentication).

```
awplus(config)# ntp trusted-key <keynumber>
```

where the *<keynumber>* is the ID number of an MD5 key that has been created for NTP authentication.

4. When defining the NTP server that the switch wishes to receive time updates from, specify the key that will be used to authentication the session to this server.

```
awplus(config)# ntp server <serveraddress> key <keynumber>
```

where the *<keynumber>* is the ID number of an MD5 key that has been created for NTP authentication, and is in the list of trusted keys.

The server must also be configured to use this key.

Configuring authentication on the NTP server

When configuring the switch as an NTP server, the configuration required to enable authentication on the server is:

1. Enable authentication for NTP.

```
awplus(config)# ntp authenticate
```

2. Create an MD5 key that can be used for NTP authentication.

```
awplus(config)# ntp authentication-key <keynumber> md5 <key>
```

where the *<keynumber>* is an ID number that will be used in other commands to refer to this key. The *<key>* is just a string.

3. Designate this key as currently trusted (i.e. can currently be used for authentication).

```
awplus(config)# ntp trusted-key <keynumber>
```

where the `<keynumber>` is the ID number of the MD5 key.

All clients that wish to carry out authenticated sessions with this server must specify this key as the key they will use for sessions to this server.

Note: Authentication is initiated by the client. If the server is configured for authentication but the client is not, then the server will still accept the client's session and serve time updates to the client. But if the client is configured for authentication and the server is not, then the session will never become established.

Configuring authentication for NTP peers

It is also possible for a pair of NTP peers to authenticate each other. In this case, the configuration is the same as in the NTP client case, except that the final command is replaced by one that defines a peer relationship

```
awplus(config)# ntp peer <peeraddress> key <keynumber>
```

Of course, both peers must use the same key in the session with each other.

Filtering and authentication can be used together to provide a secure configuration. Filtering limits the addresses from which NTP messages will be accepted, and authentication enables a client to determine that a server is who it says it is.

Configuration Script for Aiding Secure Network Monitoring

```
! Logging
! Configure logging to a syslog server.
log host <syslog-server-IP-address>
log host <syslog-server-IP-address> level informational
!
! NTP
! Configure secure NTP synchronisation.
ntp authenticate
ntp authentication-key 23 md5 secretKey
ntp trusted-key 23
ntp server <server-IP-address> key 23
!
```


Protecting the CPU

Control plane bandwidth control

To ensure that the CPU processing capability will never be oversubscribed by the data arriving from the switching fabric, a strict limit can be imposed on the rate at which data is transmitted from the Fabric-to-CPU channel. This works because network management and control traffic, whilst vital, is not high in volume. If high volumes of data are coming up to the switch's CPU, most of this data will not be valid control plane packets.

For instance, they could be packets generated by deliberate DoS (Denial of Service) attacks, or sustained high levels of broadcasts caused by a loop or a faulty device on the network. Limiting the rate of data transfer to the CPU will not penalise normal control plane communications, but will combat the effect of DoS attacks and storms.

By default, the AlliedWare Plus operating system sets the maximum rate to 60Mbps. This rate is ample to accommodate the requirements for control plane traffic in any normal network environment, and yet is low enough to prevent oversubscription of CPU-based processes. We recommend that you avoid altering the limit unless there is no other option. However, if you need to alter this limit, use the command:

```
awplus(config)# platform control-plane-prioritization rate  
                <rate-limit>
```

If the control plane data rate requirements in the network exceed 60Mbps, then you may need to consider which elements in the network design require such a high rate of control plane traffic, and consider other design options.

Using hardware ACLs to protect the CPU from IPv4 attacks

Malicious attacks sometimes try to overload the CPU with traffic destined for the switch. Getting the CPU to run at its maximum capacity (100%) causes problems processing network control traffic, which is critical to keep a network functioning well.

Control Plane bandwidth control reduces the effect of such attacks. The effect can be further reduced by filtering out traffic that does not need to be sent to the CPU.

Using hardware Access Control Lists (ACLs) to block traffic destined for the switch's own IP address can protect the CPU. The following example shows how to configure ACLs to match on traffic destined for the switch's IP address. The ACLs allow SNMP, SSH, and Web traffic to the switch's management IP address, but block all other traffic to the management IP address 172.28.78.23.

Configuring ACLs to match on traffic destined for the switch's IP address

1. Create ACLs (filters) to allow access for the management protocols and deny other traffic to the management IP address.

1. Permit SNMP traffic from the management subnet to the management address via UDP destination port 161:

```
awplus(config)# access-list hardware protect-management
awplus(config-ip-hw-acl)# 10 permit udp 172.28.0.0/16
172.28.78.23/32 eq 161
```

2. Permit HTTP traffic from the management subnet to the management address via TCP destination port 80:

```
awplus(config-ip-hw-acl)# 20 permit tcp 172.28.0.0/16
172.28.78.23/32 eq 80
```

3. Permit SSH traffic from the management subnet to the management address via TCP destination port 22:

```
awplus(config-ip-hw-acl)# 30 permit tcp 172.28.0.0/16
172.28.78.23/32 eq 22
```

4. If you wish to allow network managers to ping the switch, then permit ICMP traffic from the management subnet to the management address:

```
awplus(config-ip-hw-acl)# 40 permit icmp 172.28.0.0/16
172.28.78.23/32
```

5. Create an ACL to block all other traffic destined to the management IP address:

```
awplus(config-ip-hw-acl)# 50 deny ip any 172.28.78.23/32
```

2. Add ACLs to ports to allow management access to the management IP address.

If there are some ports via which you wish to allow management access to the switch, then configure the hardware ACL on those ports:

```
awplus(config)# interface port1.0.1-1.0.10
awplus(config-if)# switchport mode access
awplus(config-if)# access-group protect-management
```

3. Add ACL to ports to block all traffic to the management IP address.

If there are some ports via which you wish to block management access to the switch, then configure a blocking ACL on those ports:

```
awplus(config)# access-list hardware block-management
awplus(config-ip-hw-acl)# 10 deny ip any 172.28.78.23/32
awplus(config-ip-hw-acl)# interface port1.0.11-1.0.24
awplus(config-if)# switchport mode access
awplus(config-if)# access-group block-management
```

Using hardware ACLs to protect the CPU from IPv6 attacks

When SSH and SNMP are enabled on the switch, they are automatically accessible by both IPv4 and IPv6. If you wish to allow SSH and SNMP management of the switch by IPv6, but block all other IPv6 access to the management IPv6 address of the switch, you could proceed as follows.

1. Create IPv6 ACLs to permit SNMP and SSH to the management IPv6 address.

```
awplus(config)# ipv6 access-list snmpv6
awplus(config-ipv6-acl)# permit udp any 3ffe:89:90::1/128 eq
161 vlan 1
awplus(config)# ipv6 access-list sshv6
awplus(config-ipv6-acl)# permit tcp any 3ffe:89:90::1/128 eq 22
vlan 1
```

2. Create an IPv6 ACL to block all other IPv6 traffic to the management IPv6 address.

```
awplus(config)# ipv6 access-list other
awplus(config-ipv6-acl)# deny ip any 3ffe:89:90::1/128
```

3. Apply these three IPv6 ACLs as global IPv6 traffic filters.

```
awplus(config-ipv6-acl)# exit
awplus(config)# ipv6 traffic-filter snmpv6
awplus(config)# ipv6 traffic-filter sshv6
awplus(config)# ipv6 traffic-filter other
```

Using Hardware ACLs to Block Packets from Invalid Source IP Addresses

There is no good reason to accept packets from invalid source IPs, and there is a good chance of such packets being part of an attack. It is good practice to filter out such packets.

1. Create ACLs to block IP packets from invalid source addresses.

```
awplus(config)# access-list 3200 deny ip 127.0.0.0/8 any
awplus(config)# access-list 3201 deny ip 0.0.0.0/32 any
awplus(config)# access-list 3202 deny ip 224.0.0.0/3 any
```

2. Apply these ACLs globally, so these packets will be blocked on all ports.

```
awplus(config)# access-group 3200
awplus(config)# access-group 3201
awplus(config)# access-group 3202
```

Note: It is not necessary to explicitly block packets from multicast source MAC addresses, as these are blocked by default.

Configuration Script for ACLs

```

! Using hardware ACLs to protect the CPU from undesirable traffic
!
! Create ACLs (filters) to allow access for the management protocols
! and deny other traffic to the management IP address.
access-list hardware protect-management
10 permit udp 172.28.0.0/16 172.28.78.23/32 eq 161
20 permit tcp 172.28.0.0/16 172.28.78.23/32 eq 80
30 permit tcp 172.28.0.0/16 172.28.78.23/32 eq 22
40 permit icmp 172.28.0.0/16 172.28.78.23/32
50 permit udp 172.28.0.0/16 eq 172.28.78.23/32
60 deny ip any 172.28.78.23/32
!
! Add ACLs to ports to allow management access to the management IP
! address.
interface port1.0.1-1.0.10
    switchport mode access
    ip access-group protect-management
!
! Add ACL to ports to block all traffic to the management IP address.
access-list hardware block-management
10 deny ip any 172.28.78.23/32
interface port1.0.11-1.0.24
    switchport mode access
    ip access-group block-management
!
! Using hardware ACLs to protect IPv6 management
!
! Create IPv6 ACLs to permit SNMP and SSH to the management IPv6
! address.
! ipv6 access-list snmpv6
! permit udp any 3ffe:89:90::1/128 eq 161 vlan 1
ipv6 access-list sshv6
! permit tcp any 3ffe:89:90::1/128 eq 22 vlan 1
!
! Create an IPv6 ACL to block all other IPv6 traffic to the management
! IPv6 address.
ipv6 access-list other
! deny ip any 3ffe:89:90::1/128
!
! Apply these IPv6 ACLs as global IPv6 traffic filters.
ipv6 traffic-filter snmpv6
ipv6 traffic-filter sshv6
ipv6 traffic-filter other
!
! Block packets from invalid source IP addresses
!
! Create ACLs to block IP packets from invalid source addresses.
access-list 3200 deny ip 127.0.0.0/8 any
access-list 3201 deny ip 0.0.0.0/32 any
access-list 3202 deny ip 224.0.0.0/3 any
ip access-group 3200
ip access-group 3201
ip access-group 3202

```


Secure Configuration of Routing Protocols

You can configure authentication to secure the Layer 3 routing protocols OSPF, RIP, and VRRP. To see the commands in this section in script format, see "[Configuration Script for Securing Routing Protocols](#)" on page 18.

OSPF

Configure OSPF with MD5 authentication.

1. Initiate an OSPF routing process.

```
awplus(config)# router ospf 1
```

2. Configure the switch to use OSPF on the interfaces that fall within specified subnets.

```
awplus(config-router)# network 172.28.0.0 0.0.255.255 area 0
```

```
awplus(config-router)# network 221.189.62.0 0.255.255.255 area  
4.3.3.1
```

3. Configure the areas to use MD5 authentication.

```
awplus(config-router)# area 0 authentication message-digest
```

```
awplus(config-router)# area 4.3.3.1 authentication  
message-digest
```

4. Leave OSPF configuration mode and then enter interface configuration mode to configure a message digest key per VLAN interface.

```
awplus(config-router)# exit
```

```
awplus(config)# interface vlan1
```

```
awplus(config-if)# ip ospf message-digest-key 1 md5  
<key-string>
```

RIP

Configure authenticated RIP.

1. Configure the switch to send RIP out the interfaces with IP addresses within specified IP subnets.

```
awplus(config)# router rip
```

```
awplus(config-router)# network 172.28.0.0/16
```

```
awplus(config-router)# network 221.189.62.0/24
```

2. Create a key chain. In this example, the name of the key chain is **rip-key-chain**. A key chain consists of a set of keys. Associated with each key are:

- a send-lifetime—the period during which the key will be sent

- an accept-lifetime—the period during which the key is a valid match.

The accept-lifetime of one key can overlap the send-lifetime of another key, to allow for slight non-synchronisation between the times at which neighbouring switches cut over from sending one key to sending another one.

```
awplus(config)# key chain rip-key-chain
awplus(config-keychain)# key 1
awplus(config-keychain-key)# key-string piano8trip
awplus(config-keychain-key)# accept-lifetime 08:30:00 Jan 09
2012 21:00:00 Feb 10 2012
awplus(config-keychain-key)# send-lifetime 09:00:00 Jan 09
2012 20:00:00 Feb 10 2012
awplus(config-keychain)# key 2
awplus(config-keychain-key)# key-string hop78run
awplus(config-keychain-key)# accept-lifetime 08:30:00 Feb 09
2012 21:00:00 Mar 10 2012
awplus(config-keychain-key)# send-lifetime 20:30:00 Feb 09
2012 20:00:00 May 10 2012
```

3. Configure particular interfaces to use MD5 authentication of RIP, and specify the key chain to use in this authentication.

```
awplus(config)# interface vlan1
awplus(config-if)# ip rip authentication mode md5
awplus(config-if)# ip rip authentication key-chain
rip-key-chain
```

VRRP

Configure authenticated VRRP.

1. Create a VRRP instance.

```
awplus(config)# router vrrp 1 vlan1
awplus(config-router)# virtual-ip 172.28.78.23 master
awplus(config-router)# enable
```

2. Specify authentication for the interface over which VRRP is configured.

```
awplus(config)# interface vlan1
awplus(config-if)# ip vrrp authentication mode text
awplus(config-if)# ip vrrp authentication string dog2jump
```

At time of writing, AlliedWare Plus supports only clear-text authentication VRRP.

Configuration Script for Securing Routing Protocols

This configuration is described in ["Secure Configuration of Routing Protocols"](#) on page 16.

```
!  
! OSPF  
! Configure OSPF with MD5 authentication.  
router ospf 1  
  network 172.28.0.0 0.0.255.255 area 0  
  network 221.189.62.0 0.255.255.255 area 4.3.3.1  
  area 0 authentication message-digest  
  area 4.3.3.1 authentication message-digest  
interface vlan1  
  ip ospf message-digest-key 1 md5 <key string>  
!  
! RIP  
! Configure authenticated RIP.  
router rip  
  network 172.28.0.0/16  
  network 221.189.62.0/24  
key chain rip-key-chain  
  key 1  
    key-string piano8trip  
    accept-lifetime 08:30:00 Jan 09 2010 21:00:00 Mar 17 2011  
    send-lifetime 09:00:00 Jan 09 2010 20:00:00 Mar 17 2011  
  key 2  
    key-string hop78run  
    accept-lifetime 08:30:00 Mar 16 2011 21:00:00 May 22 2012  
    send-lifetime 20:30:00 Mar 16 2011 20:00:00 May 22 2012  
interface vlan1  
  ip rip authentication mode md5  
  ip rip authentication key-chain rip-key-chain  
!  
! VRRP  
! Configure authenticated VRRP.  
router vrrp 1 vlan1  
  virtual-ip 172.28.78.23 master  
  enable  
interface vlan1  
  ip vrrp authentication mode text  
  ip vrrp authentication string dog2jump  
!
```

Chapter 2 | Blocking Network Attacks

Introduction

There are a number of well-known LAN-based attacks, both Denial of Service (DoS) attacks and information stealing attacks. In this chapter we will consider each of the well known attacks and discuss how to configure AlliedWare Plus™ switches to combat these attacks.

These include:

- Address Resolution Protocol (ARP) spoofing
- VLAN hopping
- Double-tagged VLAN hopping
- Dynamic Host Configuration Protocol (DHCP) starvation and rogue server attacks
- Root Bridge spoofing
- MAC flooding
- DoS attacks

List of terms

VLAN hopping

Breaking VLAN security by tricking a router into putting users' traffic into a VLAN to which they should not have access.

ARP spoofing

Tricking a router into sending data to the wrong host, by deliberately causing its ARP cache entries to associate the wrong MAC address with certain IP addresses.

MAC flooding

Filling a switch's MAC table so that it will flood all packets like a hub.

DHCP starvation

Causing a DHCP server to allocate most of its IP leases to fictitious hosts, thereby leaving it without sufficient available leases to serve to legitimate hosts.

Rogue DHCP server

A host masquerading as a DHCP server and serving deliberately invalid leases to hosts, either as a simple denial of service attack or to direct them to a bogus DNS server that will take them to phishing web sites or similar.

Mitigating common network attacks

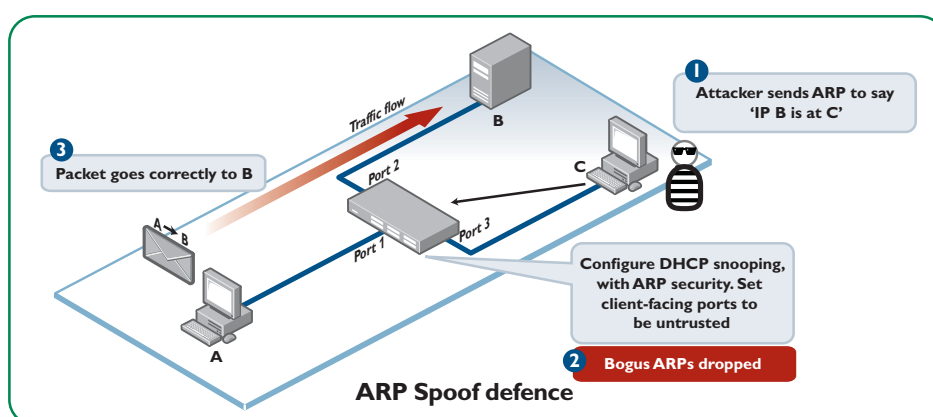
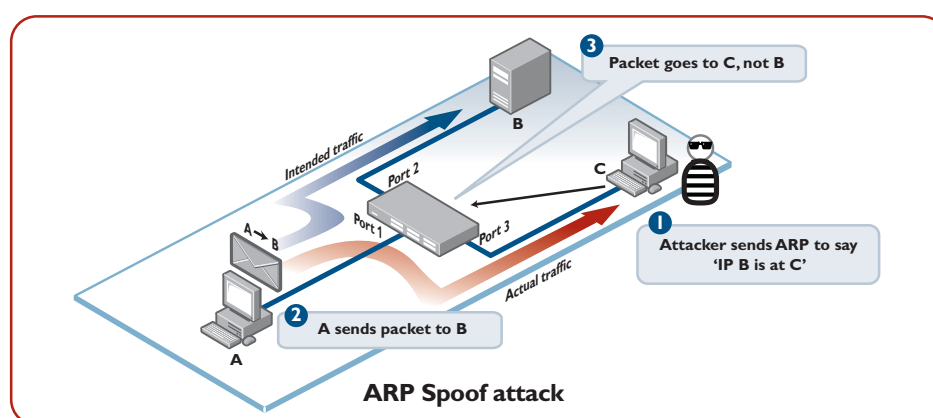
The need for network security is significantly increased due to increased mobility and the wide availability of various hacking tools. Let's consider some of the more common information stealing and denial of service attacks.

Address Resolution Protocol (ARP) spoofing attacks

One form of information stealing attack is ARP spoofing. A malicious host sends a bogus reply to a network server, claiming to be a genuine host desiring information. Once the switch has an incorrect entry in its ARP table, the malicious host starts to receive data intended for the genuine recipient.

Allied Telesis switches can use DHCP Snooping with ARP Security to protect your network from ARP spoofing attacks. All ARP replies arriving on untrusted ports are checked to ensure they contain legitimate network addressing information, safeguarding your network and ensuring online information reaches its intended destination.

In the example ARP spoof attack below, a hacker sends an ARP reply to a host's ARP request for server B. The hacker (C) falsely claims to be that server, tying their own MAC address to the IP address owned by the server. The bogus ARP message then also adds an entry to the switch's ARP table. When a message arrives for the valid device, B, this bogus ARP entry diverts it to C. For more information about DHCP Snooping, please see ["What is DHCP snooping?"](#) on page 154.



VLAN hopping attacks

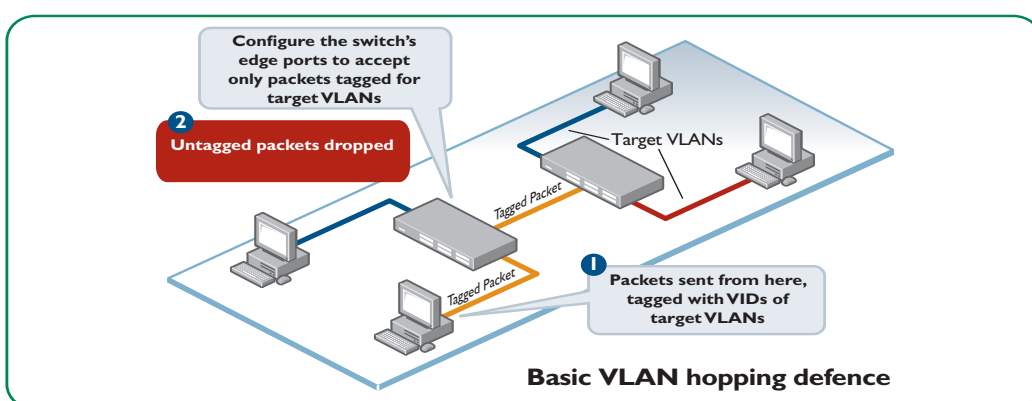
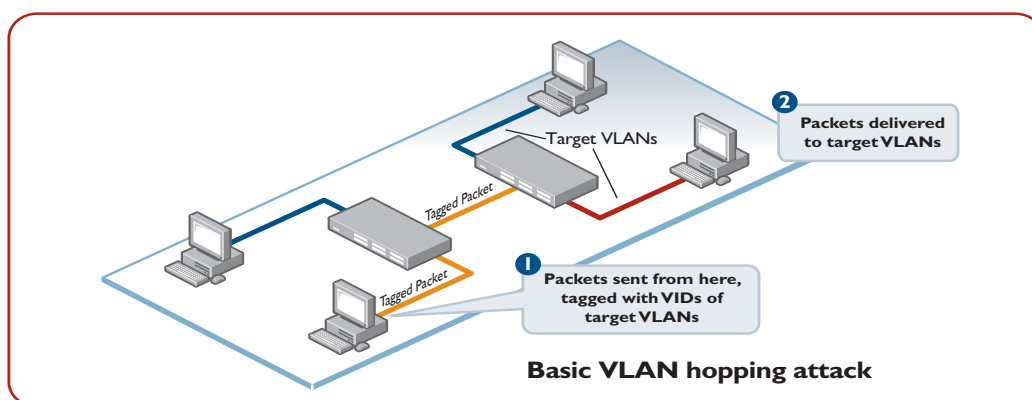
VLANs aim to provide a degree of network security via user segmentation. To bypass this, a malicious host wishing to gain access to an unauthorized VLAN can simply send a tagged packet into the network with the VLAN identifier of the target VLAN, which typically the switch will forward to that VLAN. A more complex variation on the VLAN attack is to send a double-tagged packet with the outer tag of the originating VLAN and an inner tag of the target VLAN. The switch will strip off the outer tag and pass the packet on to the target VLAN identified by the inner tag.

Allied Telesis switches eliminate basic and double-tagged VLAN hopping attacks by using ingress filtering to drop all tagged packets, since workstations attached to edge ports should not send tagged packets into the network, as shown in the diagrams below.

Simple VLAN hopping attack

In this VLAN hopping attack example, a malicious user in one VLAN gains unauthorized access to a different VLAN by sending tagged packets into the network with the VLAN ID (VID) of the targeted VLAN. If ingress filtering is not enabled, a device looks at the VLAN tag and passes the packet on to the targeted VLAN, despite the ingress port not being a member of that VLAN.

The defence against this attack is to ensure that ingress filtering is enabled on all ports, so that packets are dropped if they are tagged for VLANs that are not configured on the ports. By default AlliedWare Plus enables ingress filtering on all ports.



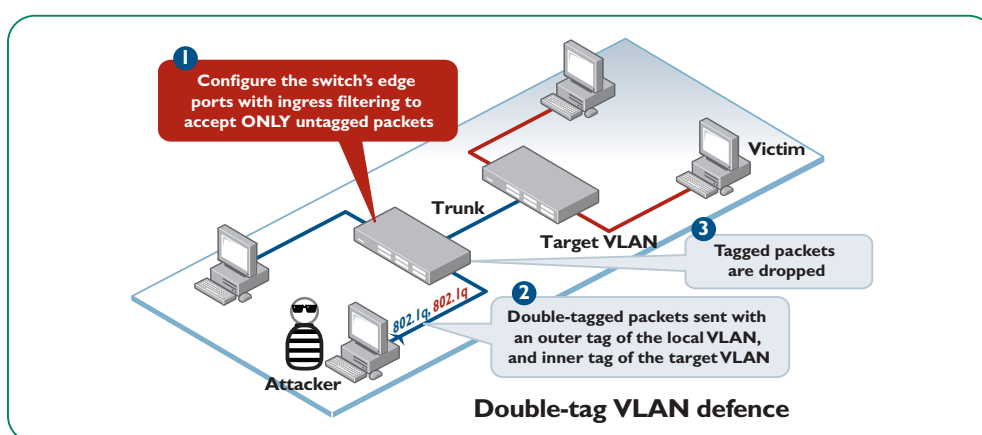
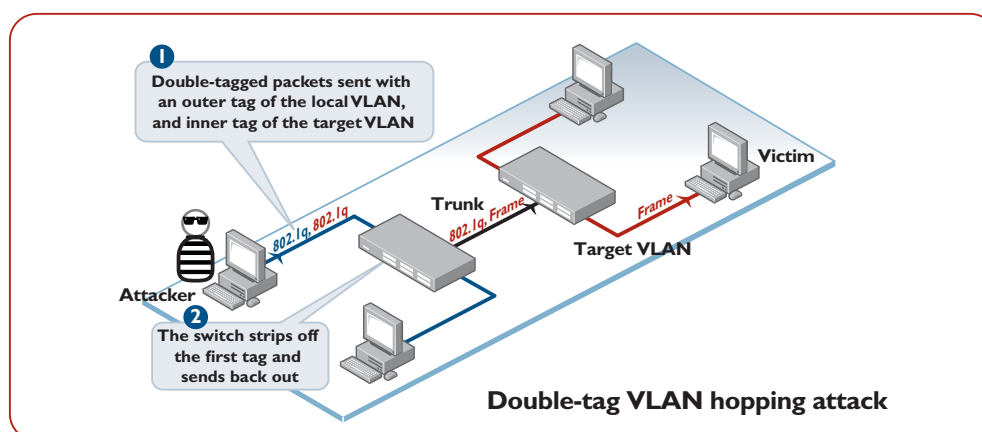
Double-tagged VLAN hopping attack

If there is a native VLAN configured on the trunk link that connects two switches, then if a double-tagged packet arrives on an access port, and its outer tag has the same VID as the native VLAN on the trunk port, then the outer VLAN will be stripped off, and the packet sent down the trunk link with its inner tag still in place.

When the packet arrives at the other end of the link, the receiving switch sees it as tagged with the VID of the inner tag. The switch will then forward the packet to ports which are members of the VLAN with that VID.

There are two ways to defend against this attack:

1. Use Ingress Filtering to drop all tagged packets on access ports, since workstations attached to edge ports should not send tagged packets into the network.
2. Do not configure a native VLAN on the trunk link between switches.



Dynamic Host Configuration Protocol (DHCP) attacks

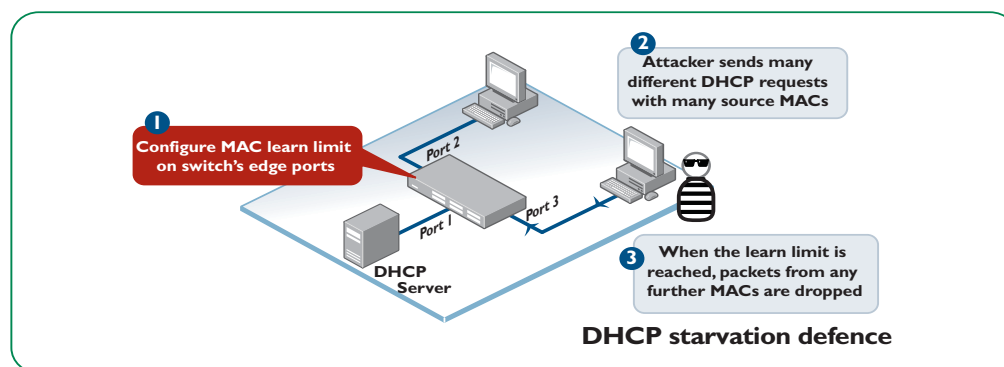
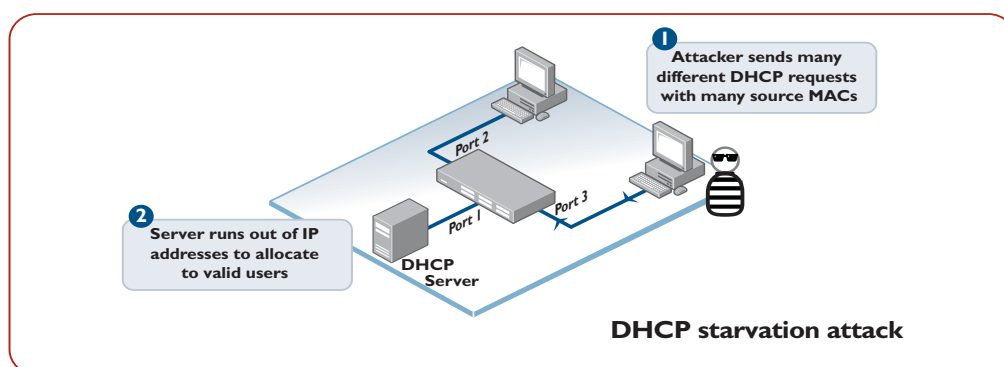
DHCP servers allocate IP network addresses to hosts, allowing them to access resources on the network.

Two forms of DHCP attack can compromise user's network access.

1. DHCP starvation attack

A malicious user inundates the DHCP server with countless requests from different bogus MAC addresses, which results in the server running out of IP addresses. Genuine users are unable to gain a network address and therefore network access.

Allied Telesis switches can use port security to stop malicious users sending multiple MAC addresses to the DHCP server, as shown in the diagrams below. Options are available for corrective action, including notifying the network administrator and/or disabling the switch port of the offender.



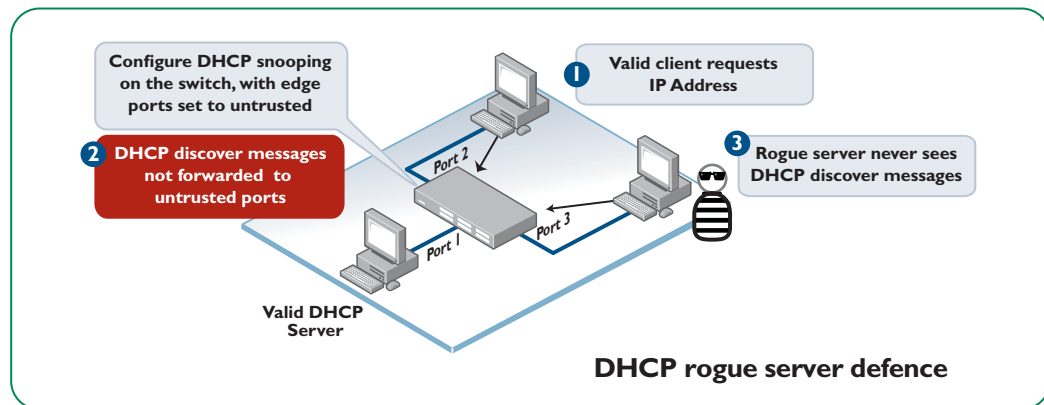
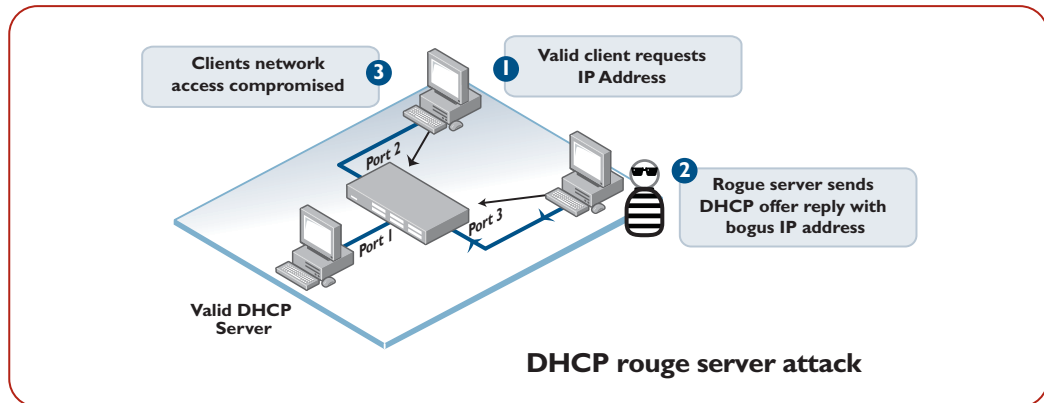
2. DHCP rogue server attack

A malicious user disguises himself as a DHCP server and responds to DHCP requests with a bogus network address, compromising the network access of genuine users.

Allied Telesis switches can avoid DHCP rogue server attacks by using DHCP Snooping to designate which ports may accept DHCP server response packets. If a rogue server is attached to an 'untrusted' port, its response packets will be dropped, rendering it useless.

This type of attack can also be accidental—for example when someone connects the WAN side of a wireless router onto the LAN.

For more information about DHCP Snooping, see [page 153](#)



Secure Configuration of Spanning Tree Protocol

The spanning tree protocol has no inbuilt security, so it is quite vulnerable to attack. There are two protection mechanisms available in the Alliedware Plus implementation of spanning tree that should always be enabled: STP root guard, and STP BPDU guard.

Protecting against Root Bridge spoofing attacks

Root Bridge spoofing is a data-stealing and denial-of-service attack. The attacker sets up a device that transmits STP hello packets with a very low priority so that their device gets elected as the Root Bridge. Once it is elected, the attacker will be able to see a lot of the data on the network, allowing them to steal that data. Root Bridge spoofing also disrupts the network.

To protect against this attack, you can configure specific ports on each switch to shut down if they receive BPDUs (Bridge Protocol Data Units) with a lower priority than the switch's own Bridge ID. Configure this setting only on ports that you know should never be root ports on the switch.

Enable STP root guard.

Configure ports that should never be root ports to shut down if they receive low priority BPDUs:

```
awplus(config)# interface <port-list>
awplus(config-if)# spanning-tree guard root
```

Disabling edge ports that receive spanning tree packets

Edge ports are those that connect to workstations or printers etc., rather than other switches. The devices that are connected to edge ports should never send spanning tree BPDUs. If an edge port receives BPDUs, then that is either a wiring error, whereby another switch has been connected to a port to which it should not be, or it is a deliberate malicious attempt to subvert the spanning-tree operating in the network. Edge ports can be configured to block all traffic (by entering the STP blocking state) if they receive BPDUs.

Enable STP BPDU guard.

Configure edge ports to shut down if they receive BPDUs:

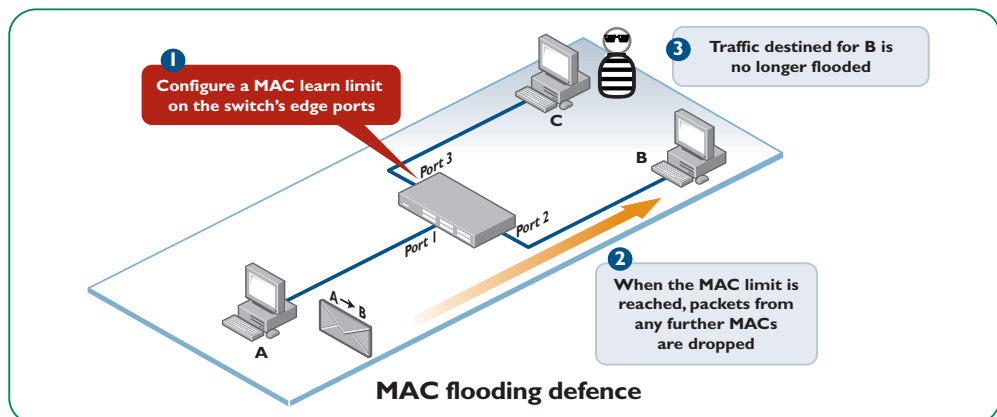
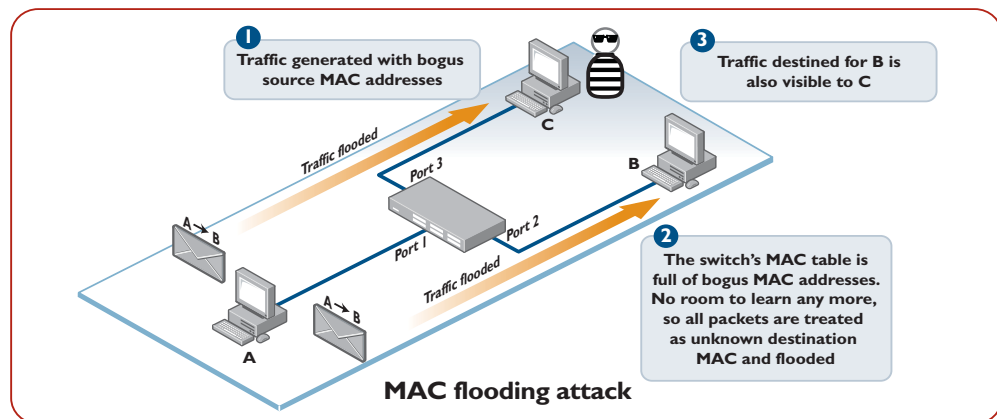
```
awplus(config)# interface <port-list>
awplus(config-if)# spanning-tree portfast bpdu-guard
```

If you do not use this option, then if the port receives a BPDU it will begin to negotiate spanning tree with the device sending the BPDU.

Protecting Against MAC Flooding Attacks

MAC flooding attack

Information stealing can be facilitated using a MAC flooding attack, which provides a source of accessible data. A malicious host sends packets from thousands of different bogus source MAC addresses, which fills the forwarding database. Once full, legitimate traffic is flooded and becomes widely accessible, as the switch does not have room to learn any more specific destination addresses in the forwarding database.



Allied Telesis switches provide two security measures which guard against a MAC flooding attack. The first is host authentication, where authenticating ports will only accept traffic from the MAC addresses of authenticated hosts. The second is port security, which controls how many MAC addresses can be learnt on a specific port, as shown in the diagrams above. Configurable options when limits are breached are to drop the untrusted data, notify the network administrator, or disable the port.

Configure port security (limit MAC address learning).

1. Enable port security on the ports that you want to limit.

```
awplus(config)# interface <port-list>
awplus(config-if)# switchport port-security
```

2. Specify the maximum number of MACs to learn on the ports.

```
awplus(config-if)# switchport port-security maximum <maximum>
```

3. Specify the action to take when the maximum number of MACs on a port is exceeded.

```
awplus(config-if)# switchport port-security violation
{shutdown|restrict|protect}
```

where:

- **shutdown** means that the port will be shut down when the MAC limit is exceeded.
- **restrict** means that an SNMP notification will be sent to inform the network manager that the limit has been exceeded.
- **protect** means that any packets from MAC addresses beyond the configured limit will be dropped. This is the default action.

Protecting Against Denial of Service Attacks in the LAN

Keeping productivity high requires reliable network access, and there are a number of DoS attacks that can threaten to thwart information availability. The attacks target devices, causing them to reduce performance, while others attempt to send a storm of data at a specific victim or to consume online resources.

Allied Telesis switches are capable of mitigating all of these attacks using DoS defence, which for the majority of these attacks is implemented in the switch's hardware, so does not affect network performance.

DoS attack prevention

The x600 and x610 Series switches are able to prevent a range of common DoS attacks from passing through the switch to attack other hosts in the network.

Enable DoS protection.

Configure DoS protection on a per-port basis.

```
awplus(config)# interface <port-list>
```

```
awplus(config-if)# dos {ipoptions|land|ping-of-death|
    smurf broadcast <local-ip-broadcast-addr>|synflood|teardrop}
    action {shutdown|trap|mirror}
```

Good Practice for Switch Security

The following are a set of quite simple configuration steps you can perform that will further reduce the risk of security violations in your network.

1. Close down and corral off unused ports.

If there are any ports on the switch that are not yet in use, then make sure nobody can use them to access the network.

There are two good practices that can be performed in this regard:

- Shut these ports down.

```
awplus(config)# interface port1.0.14
awplus(config-if)# shutdown
```

- Put the ports into a dead-end VLAN, so that even if they are mistakenly enabled, the traffic that enters the ports cannot get any further than the local switch.

To do this, create a Private Edge VLAN that has no IP address and no promiscuous port. Make this VLAN the **native** VLAN on the unused ports. That way, traffic entering the ports has no way to be forwarded on by the switch at either Layer 2 or Layer 3.

```
awplus(config)# vlan database
awplus(config-vlan)# private-vlan 3 isolated

awplus(config)# interface port1.0.2
awplus(config-if)# switchport access vlan 3
awplus(config-if)# switchport mode private-vlan host
```

2. Avoid using VLAN 1 as the management VLAN for switches in the network.

VLAN 1 is the default VLAN on Allied Telesis, and other vendors' switches—it is the VLAN that is the most likely to be the native VLAN on any switch port that has not been configured with security in mind, so it is the VLAN that an attacker is most likely to be able to get their traffic into.

You can instead create another specific VLAN for device management, minimize the number of ports in the network that are members of this VLAN, and isolate this VLAN from the VLANs containing user traffic. This reduces the chances of an attacker being able to get management access to the switches.

3. Use syslog to provide a detailed audit trail in the event of a suspected security breach or other problem.

Configure the switch to log all activity to a syslog server:

```
awplus# configure terminal
awplus(config)# log host <syslog-server-IP-address> level
informational
```

4. Use NTP to synchronize switch system time.

Investigating any events that happen on the network is easier if the system time on all switches is synchronized. The most effective way to synchronize the time on all the switches is to use NTP.

A possible configuration to securely synchronize the switch to a timer server would be:

```
awplus(config)# ntp authenticate
awplus(config)# ntp authentication-key 23 md5 secretKey
awplus(config)# ntp trusted-key 23
awplus(config)# ntp server <server-IP-address> key 23
```

5. Configure your switch to output a banner message when users connect.

This banner should give notice to anyone who connects to a switch that it is for authorized use only and any use of it will be monitored. Courts have dismissed cases against those who have attacked systems without banners. Having no banner on a switch may lead to legal or liability problems.

```
awplus# configure terminal
awplus(config)# banner login
Type CNTL/D to finish.
<Type in the text for the banner>
awplus(config)# exit
awplus# exit
```

6. Avoid directed broadcast forwarding.

If possible, do not enable directed broadcast forwarding. Directed subnet broadcasts are a technique commonly used by DoS attacks (and for spreading viruses).

A directed subnet broadcast occurs when a host sends a packet to the broadcast address of a subnet that is not the host's own subnet (for example, the host 192.168.2.45 sending a broadcast to 192.168.3.255, to attack all devices in the 192.168.3.0 subnet.) For the message to be broadcast, the router that provides the gateway from the 192.168.2.0/24 subnet to the 192.168.3.0/24 subnet must turn the packet from a unicast to a broadcast and forward it onto the 192.168.3.0/24 subnet. This act on the part of the gateway router is commonly referred to as directed broadcast forwarding.

The switches are capable of performing directed broadcast forwarding, but by default that capability is disabled. It should only be enabled if it is strictly required.

Configuration Script for Blocking Network Attacks

```
!
! Blocking network attacks
!
! Secure configuration of Spanning Tree Protocol
!
! Enable STP root guard.
interface <port-list>
    spanning-tree guard root
!
! Enable STP BPDU guard.
interface <port-list>
    spanning-tree portfast bpdu-guard
!
! Protecting against MAC-flooding attacks
!
! Configure port security (limit MAC address learning).
interface <port-list>
    switchport port-security
    switchport port-security maximum <maximum>
    switchport port-security violation {shutdown|restrict|protect}
!
! Protecting against DoS attacks in the LAN
!
! Enable DoS protection.
interface <port-list>
    dos {ipoptions|land|ping-of-death|smurf broadcast
        <local-ip-broadcast-addr>|synflood|teardrop} action {shutdown|
        trap|mirror}
!
!Configure DHCP snooping
!
service dhcp-snooping
interface <trusted interface list>
    ip dhcp snooping
    arp security
```

Chapter 3 | AAA

Introduction

AAA stands for authentication, authorization, and accounting. The acronym is used as an umbrella term to refer to all interactions that a switch has with security services.

As far as network engineers are concerned, their most frequent interaction with the acronym AAA is in the form of AAA commands.

It is this command-related use of the acronym AAA that we will be considering in this chapter. The chapter will look at the purpose of the AAA commands, and explain other terms that are used in descriptions of the operation of the AAA commands.

Then, we will move on to consider Authentication in particular, to give an overview of the authentication activities in which LAN switches participate, and to consider some specifics of the configuration and operation of a LAN switch's authentication feature-set.

List of terms

Authentication

Deciding whether a client is allowed access.

Authorization

Deciding what level of access a client is allowed - what services they are allowed to use.

Accounting

Keeping a record of the client's session, and collecting statistics on their data usage.

Method list

A list of servers that the switch's AAA process can use. The switch can run through the list until one of the servers responds.

VTY

A generic term used in the AW+ command-line to refer to the 'Virtual Terminals' that service Telnet and SSH sessions.

AAA Commands

Authentication, Authorization, and Accounting are multiple activities, which can be performed in multiple contexts, interacting with multiple servers. Moreover, if one server fails, it may be necessary to have defined fall back options to other servers.

There are potentially a large number of combinations that are possible between all these activities, servers, and contexts. To enable these combinations to be configured in a consistent, intuitive fashion, network equipment vendors have introduced some concepts that are used to provide some structure to the configuration of AAA services.

First, we will introduce these structures, and then look at how they are brought together to create an intuitive mechanism for configuring AAA services.

Server Groups

The two protocols most commonly used for Authentication, Authorization, and Accounting are RADIUS and TACACS. When using these protocols, the switch will exchange data with a RADIUS or TACACS server.

- For **authentication**, the switch will send user credentials to a RADIUS or TACACS server, and listen for the server's response to those credentials.
- For **accounting**, the switch sends accounting messages to the server, and the server uses those to accumulate usage records of network services.
- For **redundancy**, a network will often contain more than one RADIUS or TACACS server. Different servers might be used for different activities. A network might use RADIUS for 802.1x authentication, but TACACS for authenticating users logging into the management interfaces of the switch itself.

To enable a set of servers to be conveniently referenced from AAA commands, the concept of a server group has been introduced to the switch command line.

A group of RADIUS servers is defined by the command:

```
awplus(config)# aaa group server radius <group name>
```

which moves the command-line into RADIUS server group mode. In this mode, servers can be added to the group by the command:

```
awplus(config-sg)# server {<hostname>|<ip-address>}[auth-port  
    <0-65535>][acct-port <0-65535>]
```

For example:

```
awplus(config-sg)# server 192.168.1.1 auth-port 1812 acct-port  
    1813
```

The conditions relating to defining of the RADIUS server group are:

- Before a server can be added to a group, it must first have been configured by the command:


```
awplus(config)# radius-server host {<host-name>|<ip-address>}[acct-port <0-65535>][auth-port <0-65535>][key <key-string>] [retransmit <0-100>][timeout <1-1000>]
```
- A given server can belong to multiple groups. By default, all RADIUS servers configured on the switch belong to a default RADIUS server group simply called 'RADIUS'.

Similarly, TACACS servers are defined with the command:

```
awplus(config)# tacacs-server host {<host-name>|<ip-address>}[key [8]<key-string>]
```

Method Lists

The construct that defines how an authentication, authorization, or accounting event will be handled in a particular context is referred to as a method list.

A method list configures the set of methods that the event can be handled by, in the order that the methods will be tried. 'Method' is possibly a bit of a misnomer. In reality, a 'method' is a **server type**.

For example, you may wish that when a user logs into the management interface of the switch, their username/password is authenticated as follows:

- first it is sent to set of RADIUS servers for authentication
- if none of the RADIUS servers reply, then the local user list in the switch itself is checked.

To configure this behaviour, you create a method list that lists the RADIUS 'method' followed by the local 'method'. From this, it is evident that the term 'method' is really referring to a server type.

Configuring method lists

Within Alliedware Plus, it is possible to create method lists for two types of activities:

- authentication
- accounting

Alliedware Plus does not currently support authorization.

The method lists for these two activities can be created for four different contexts:

1. 802.1x
2. MAC-based authentication
3. Web-based authentication
4. Switch management session login

For the first three of these contexts, Alliedware Plus supports just one method list, called **default**. For management session login, it is possible to create a default method list, and any number of other, named, method lists.

For 802.1x, MAC-auth and web-auth, the method available for authentication is RADIUS. It is necessary to define which RADIUS server group is being used.

The commands to create an authentication method list for 802.1x, MAC-auth and web-auth are:

```
awplus(config)# aaa authentication dot1x default group {<group-name>|radius}

awplus(config)# aaa authentication auth-mac default group {<group-name>|radius}

awplus(config)# aaa authentication auth-web default group {<group-name>|radius}
```

Points to note:

- For any one of these authentication types, the authentication will not operate until the default authentication method list has been defined.
- The commands above effectively enable those three authentication types.
- If the server group 'radius' is chosen, then all the RADIUS servers configured on the switch will be available to the authentication method.

For authentication of the login to management sessions on the switch, the 'local' method is available, as well as RADIUS and TACACS. So, the syntax of the command for creating a login method list is:

```
awplus(config)# aaa authentication login {default|<list-name>}
                {[local][group {radius|tacacs+|<group-name>}]}
```

If you wish the username and password to be checked with a group of TACACS servers, and then checked in the switch's own user list if the TACACS servers do not respond, the method list would be configured as:

```
awplus(config)# aaa authentication login default group tacacs+
                user-servers local
```

The process of checking with the servers in the server group is described below in the section "[Checking multiple authentication servers](#)" on page 36.

Accounting method lists

The command for creating an accounting method list for one of 802.1x, auth-MAC, or auth-web is:

```
awplus(config)# aaa accounting <context> default {start-stop|
stop-only|none} {group {radius|<group-name>}}
```

where *<context>* is one of dot1x, auth-MAC, or auth-web.

For management login sessions, it is possible to create named method lists as well as the default method list:

```
awplus(config)# aaa accounting login {default|<list-
name>} {start-stop|stop-only|none} {group {radius|<group-
name>}}
```

The method list definition also defines whether the switch will send accounting start and/or stop messages or neither. There is a separate command **aaa accounting update** that controls whether or not RADIUS accounting update messages will be sent. This is a global command, so it controls the action of all accounting sessions, regardless of which method list they are controlled by. The details of RADIUS accounting are described in "[Chapter 5 | RADIUS](#)" on page 67.

Applying login method lists

For 802.1x, auth-MAC, and auth-web, there is only one method list, the default method list. However, for the authentication of users logging into management sessions on the switch, it is possible to create multiple named method lists.

The types of management session to which these method lists can be applied are:

- Console sessions on the switch's RS-232 port
- Remote CLI sessions via Telnet
- Remote CLI sessions via SSH

The method lists are applied to these session types by configuring the login method on the virtual interfaces via which these sessions access the switch.

The virtual interfaces are configured via the **line** command. The command to enter configuration mode for the console virtual interface is:

```
awplus# configure terminal
awplus(config)# line console 0
```

The command to enter configuration mode for the Telnet/SSH virtual interface is:

```
awplus(config)# line vty 0 4
```

Note: Telnet and SSH both use the same set of vty lines.

Within the interface configuration mode for these virtual interfaces, the command to apply an authentication method list is:

```
awplus(config-line)# login authentication <method list name>
```

To configure Telnet/SSH to use a RADIUS group 'trust', then check the local database, configure as

```
awplus(config)# aaa authentication login remote-login group
trust local
awplus(config)# line vty 0 4
awplus(config-line)# login authentication remote-login
```

Processing Authentication Requests

Checking multiple authentication servers

The logic by which a set of servers is checked is as follows:

1. The authentication request is sent to the first server in the list.
2. If the server responds (either to accept or reject the authentication request), no more servers are contacted.
3. If the server does not respond, the switch waits for timeout period. The timeout period defaults to 5 seconds, but can be configured, on a per-server basis, to a different value with the commands:

```
awplus(config)# radius-server host <ip-address> timeout
<timeout>
awplus(config)# tacacs-server timeout <seconds>
```

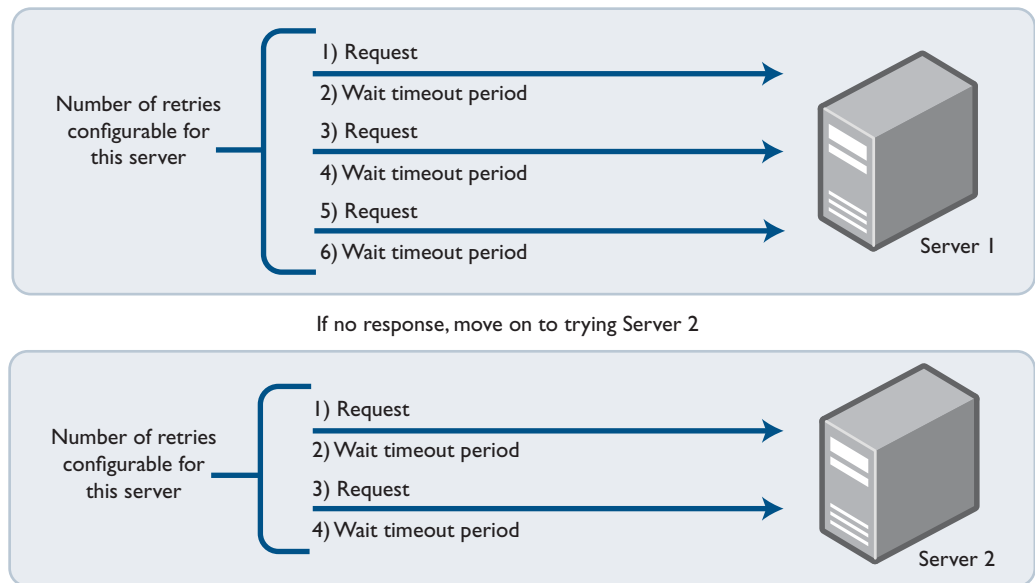
In the case of RADIUS, if no response is received within this time, then:

- the authentication request is sent to the server again.
- the switch again waits for the timeout period.

This cycle is repeated a number of times. By default, this number is 3, but can be configured, on a per-server basis, to a different value with the command:

```
awplus(config)# radius-server host <ip-address> retransmit
<number of retries>
```

4. If a full set of retries has been sent to a server, and still no response has been received, then the switch gives up on that server. It moves on to the next server in the group, and sends the request to that server. This process continues until a response has been received, or until all servers have been tried, and none has responded.



In the case of TACACS, if no response is received from the first attempt, the server is considered dead. This is because TACACS uses TCP which is a full connection protocol, if a connection cannot be established there is no purpose in retrying.

It is important to note that if a server's database does not contain a particular username, then it will respond with a reject message. The process of checking a series of servers is not a matter of looking for the server that knows of a user; it is just a matter of looking for a server that responds. A reject response is as valid as an accept response. As soon as the switch receives ANY response from a server, it will not check with any more servers in the group.

Generic authentication features

In subsequent chapters, we will examine each of the three authentication methods - 802.1x, MAC-auth, and Web-auth - separately. In the current section, we will look at some features that are common to all three authentication methods.

Critical ports

A critical port is one that has been configured as an authenticating port, but will treat all supplicants as authorized if the switch cannot communicate with the RADIUS server(s).

The intention of this feature is reduce the consequences of a loss of communication to the RADIUS server(s). If important devices were being prevented from accessing the network because authentication could not be completed, then the problems this causes may be worse than the risk of a malicious host being able to access the network.

The network administrator has the option to configure a port as critical. They thereby take the risk that a malicious host **might** gain access to the network via this port if the RADIUS server(s) go down. But they accept that risk in the interests of not blocking

access to the legitimate host that should be attached to the port. In general, it would be prudent to ensure that physical access to a critical port is well controlled, to make it very difficult for a malicious host to be attached to the port.

The command to configure a port as critical is:

```
awplus(config-if)# auth critical
```

This command is entered in interface configuration mode for the physical port interface.

Quiet period

Whilst authentication protects the network from malicious hosts, the authenticating switch itself is open to Denial of Service (DoS) attacks from malicious hosts. A malicious host attached to an authenticating switch could try to tie up the switch's processing resources by sending in rapid series of authentication requests.

To guard against this type of attack, the authentication process in Alliedware Plus implements a quiet period. The quiet period defines the length of time that the switch will simply ignore incoming authentication requests after an authentication failure has occurred.

By **default**, the quiet period is **60 seconds**, but it can be configured to other values by the command:

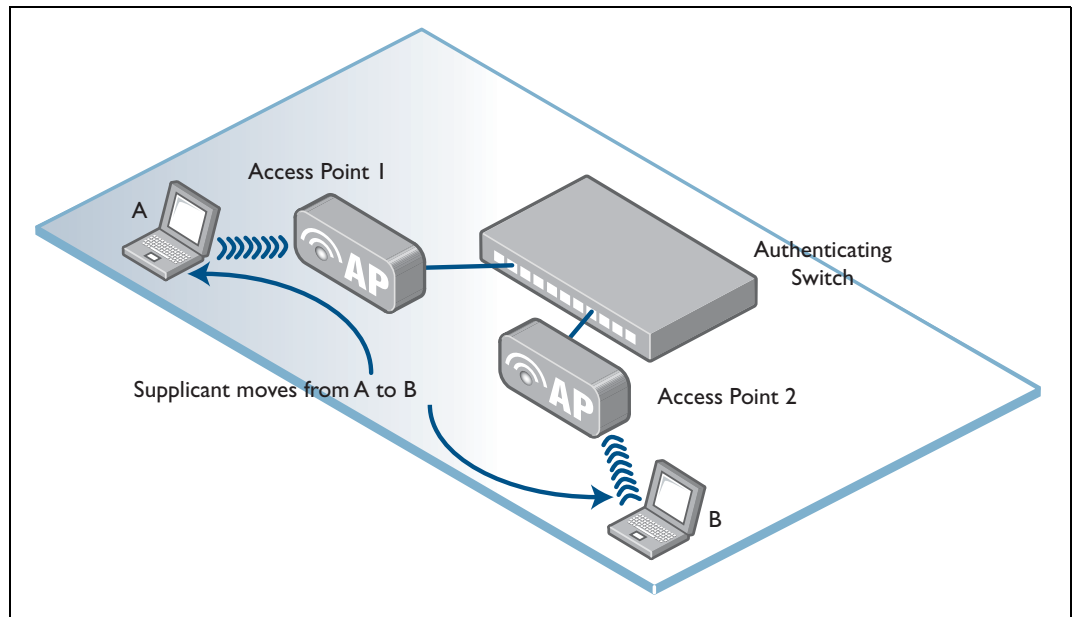
```
awplus(config-if)# auth timeout quiet-period <1-65535>
```

This command configures the quiet period on a per-port basis, and is entered in interface configuration mode for the physical port interface.

Roaming authentication

Problem

Wireless Ethernet users are mobile, and can roam from one Access Point to another. A user can be authenticated to a switch in behind the access points. By default, an authenticated session is associated with a specific switch port which can make it inconvenient for the user to re-authenticate when moving between access points that are attached to different ports.



Solution

Under roaming authentication, when a supplicant MAC is seen to move from one port to another, the following information about the supplicant is **transferred** from the original port to the new one:

- Authentication status and method
- Supplicant MAC and IP address (if an authenticated interface is configured for Web authentication)
- Supplicant name
- Authorized dynamic VLAN ID and RADIUS server
- Re-authentication timer

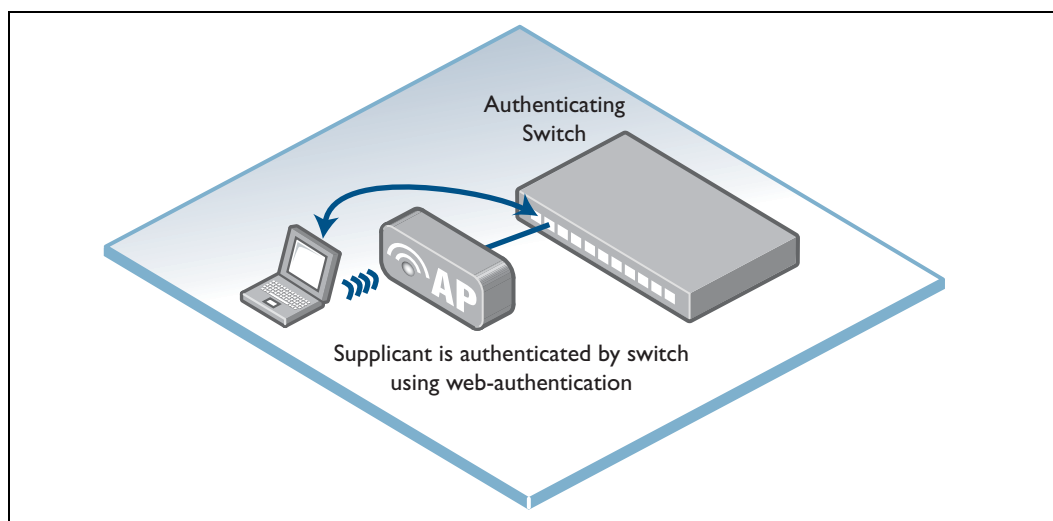
Therefore, the supplicant does not need to re-authenticate – the transition between the ports is transparent to the user.

Roaming authentication in wireless and wired environments

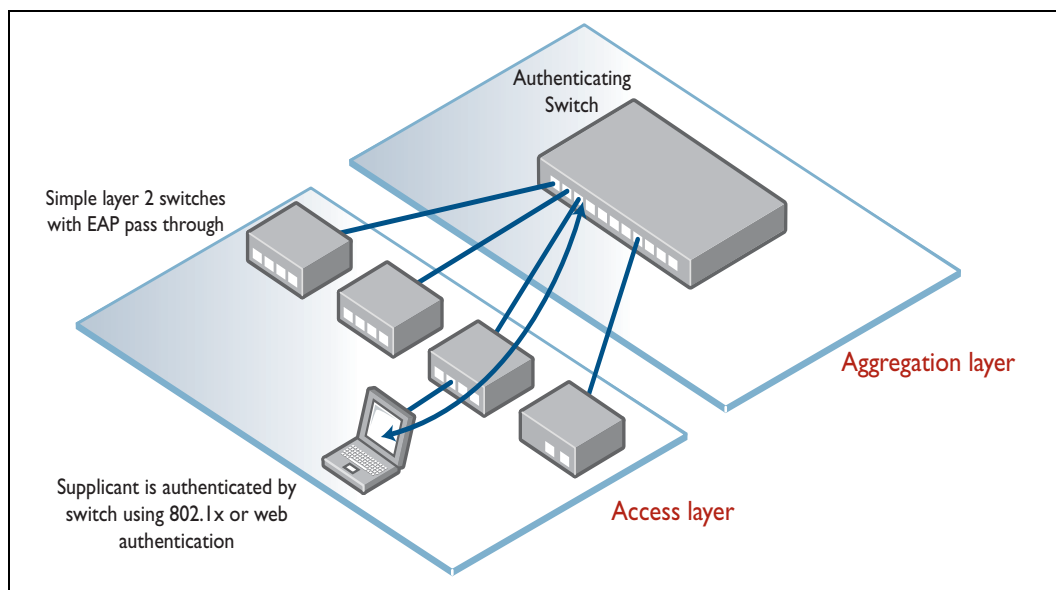
In a wireless environment, and 802.1x authentication is being used, the standard practice is for the wireless access point to be the authenticator. This enables session encryption keys to be sent to the access point and used to encrypt the wireless data session.

It will not be common to use roaming authentication for dot1x authentication on switches in a wireless environment.

In some school and public-access environments, there will not be 802.1x authentication at the access point, but web-authentication at the switch behind the access point. This is the situation where roaming authentication will be most used.

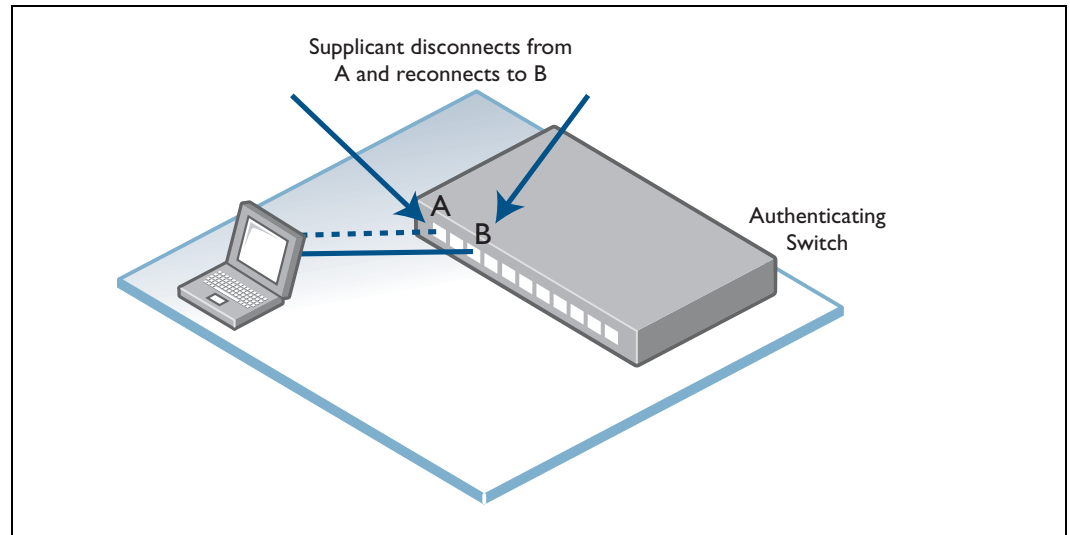


In a wired environment, where simple EAP pass-through switches are used at the edge of the network, then 802.1x or web-authentication could be performed on the authentication-capable switch at the aggregation layer.



In this case, roaming authentication for dot1x or web auth will enable a user to unplug from one edge switch and plug into another without needing to re-authenticate. Roaming authentication can support the case that the supplicant is connected directly to the authenticating switch.

The roaming authentication for disconnected ports feature enables a supplicant to be disconnected from a port on the authenticating switch and connected to another port of the switch without needing to re-authenticate.



Configuring roaming authentication

1. Enable roaming authentication on a per-port basis:

```
awplus(config)# interface port1.0.3
awplus(config-if)# auth roaming enable
```

2. If you wish to support roaming authentication with port disconnection on that port, a further command is required:

```
awplus(config)# interface port1.0.3
awplus(config-if)# auth roaming disconnect
```

Note: That roaming authentication only supports the case that the supplicant has moved between two ports that have identical authentication configuration. The two ports must also be in the same VLAN

Chapter 4 | X.509 Certificates

Introduction

The most popular method currently in use for achieving secure data transactions is the use of public/private key pairs, backed up with X.509 certificates.

An X.509 certificate binds a name to a public key value. The role of the certificate is to associate a public key with the identity contained in the X.509 certificate.

This chapter describes:

- Public/Private key pairs.
- How these key pairs secure transactions without pre-sharing of keys.
- What X.509 certificates are and how they are processed.
- The central role that X.509 certificates play in key-pair security.
- Examples of X.509 certificates being used in different applications.

List of terms

Asymmetric encryption

A form of encryption where keys come in pairs. The key used to decrypt the data is different to that which is used to encrypt the data.

Public/private key pair

In asymmetric encryption the public key of a key pair is freely distributed, and is only used to encrypt data.

The private member of the key pair is never divulged and is used to decrypt the data.

Digital signature

A number stored inside an X.509 certificate file that uniquely identifies the identity of the owner of a public/private key pair.

X.509 Certificate

Verifies the identity of the possessor of the private key corresponding to the certificate's public key.

Aspects of Security

In secure data transactions, there are three important requirements that must be satisfied:

1. **Encryption**—scrambling the contents of the packets with a sufficiently uncrackable algorithm, so that anyone eavesdropping on the conversation cannot work out the actual contents of the packets.
2. **Validation**—ensuring that the participants in the transaction are who they say they are.
3. **Tamper prevention**—ensuring that the packets that are transferred are not altered along the way.

The most popular method currently in use for achieving all three of these requirements in a reliable manner is the use of public/private key pairs, backed up with X.509 certificates.

Let us look at how this method works, and how it reliably achieves all the three requirements.

Encryption

Most encryption methods require an algorithm and a key (or multiple keys). The encrypting device feeds the data and key(s) into the encryption algorithm and scrambles the data. The decrypting device feeds the encrypted data and the same key(s) into the companion decryption algorithm, and recovers the original unscrambled data.

The important point is that both ends of the conversation need to have the same key(s). The distribution of the key(s) is a tricky problem to solve.

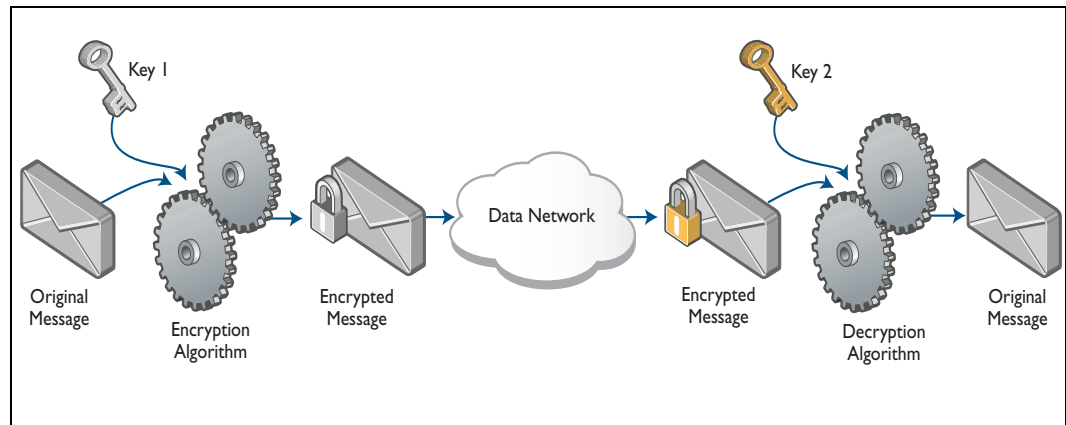
If the keys are shared at the start of the data transaction, they would have to be sent in an unencrypted form (as encrypted communication cannot begin until the keys are exchanged). But, then an eavesdropper would be able to see the keys, and steal them. Having stolen the keys, the eavesdropper could decrypt all the data transferred in the session, and the encryption was pointless.

The keys could be shared by some other completely separate means—sent in an SMS message, written on paper and sent in the post, couriered on a flash stick etc. But these methods are all rather slow and manual—and prone to human error. Moreover, they are still somewhat vulnerable to interception and key stealing.

It would be most desirable to have a completely secure way of exchanging keys at the start of the data session itself. Surprising as it may seem, there is actually a way to achieve this. The piece of magic that makes this possible is known as **asymmetric encryption**.

Asymmetric encryption

Asymmetric encryption algorithms are ones in which the key used to decrypt the data is different to that which is used to encrypt the data. These algorithms use key pairs. Data encrypted with one member of the pair can only be decrypted with the other member of the pair, and vice versa.



At first glance, this might not seem to be so powerful. In fact it might appear that we have simply introduced a new class of overly complicated encryption algorithms. But with the addition of one more simple idea, asymmetric encryption becomes very powerful.

Public/private pairs

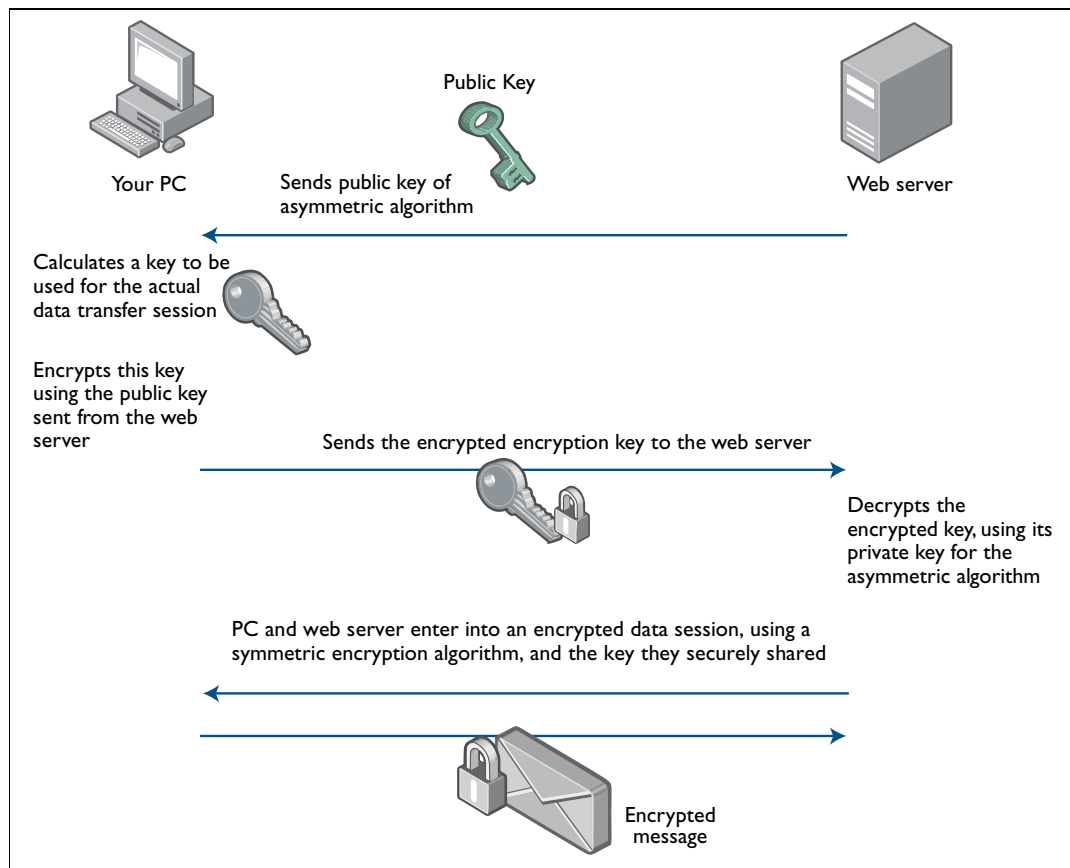
The additional idea is to treat the key pair as a public/private pair. The public member of the pair is freely distributed, with no attempt to hide it from eavesdroppers, because its only job is to encrypt. The private member of the key pair is never divulged. The owner of the key never reveals it to anyone. The private key is required to decrypt anything that has been encrypted by the public key.

Let us examine why this concept of public/private key pairs is so powerful. The best way to examine it is to consider an example data transaction. In fact, let us consider a very familiar transaction that almost all of us have experienced, and which uses public/private keys (even if we had not realised it). The example transaction is that of using your credit card to buy goods from an e-commerce website.

When your computer begins its session with a secure website:

1. The web server sends you its public encryption key.
2. Your computer then computes a key that will be used as the encryption key for the rest of the data session. Note that the encryption algorithm that will be used for the rest of the data session is a standard symmetric algorithm; it is only the key exchange that is secured by the asymmetric algorithm. Your computer computes a key that will be used for the subsequent encrypted data session.

3. Then, your computer encrypts this key using the public key that the web server sent to it.
4. This encrypted key is then transmitted to the web server. It does not matter who intercepts this message, and takes a copy of it; the **only** key that can decrypt the message is the web server's private key.
5. Only the web server has a copy of the private key, so no eavesdropper will be able to decrypt the message and learn the key that you and the web server will be using for your data transfer.



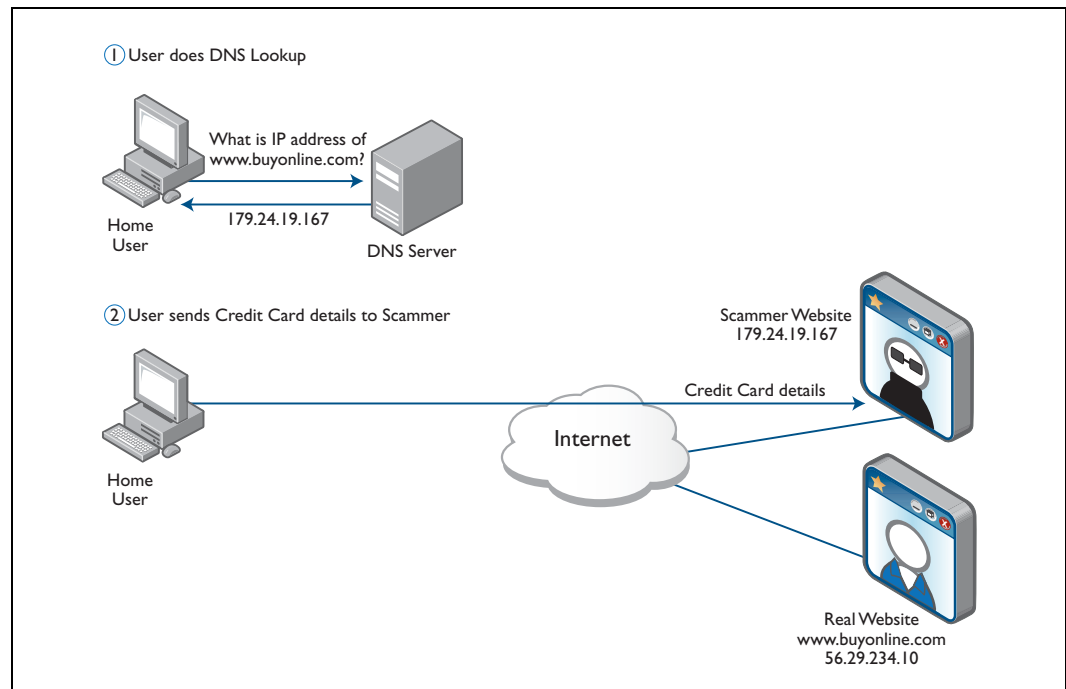
The asymmetric encryption algorithms, along with the idea of treating the keys as a public/private pair, has cracked the problem of how to securely transfer the key that will be used for a data session that is being encrypted with a symmetric algorithm.

That is **encryption** dealt with. Now let us look at how adding X.509 certificates into the process satisfies the **validation** requirement.

Validation

In the example above we see how your PC can exchange an encryption key with the web server in a way that keeps it safe from being stolen. But, how do you know that the web server to whom you sent your credit card details was actually the server it purported to be?

A sophisticated scam might have installed false records into DNS servers, so that when you directed your browser to the URL of the trusted on-line store, your traffic was actually being sent to the scammer's fake look-alike site.



A scam to steal credit card details, using a bogus DNS server.

This is where X.509 certificates and digital signatures come in.

Definition An X.509 certificate is an electronic file that verifies the identity of the owner of a public/private key pair.

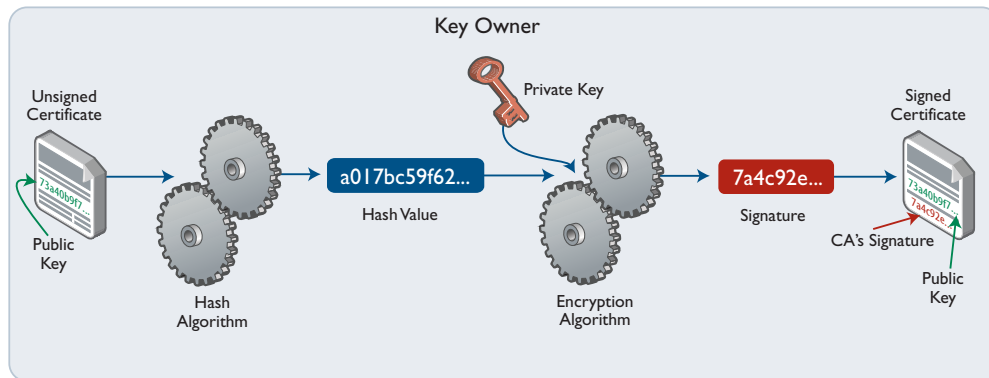
The file contains information like the owner's domain name, some of their physical address, an email address, the public key, and the algorithm that the key pertains to. The certificate also contains fields stating the dates of the beginning and end of its period of validity. There are a large number of other fields that can be, but do not have to be, present in a certificate.

One field that every certificate **must** contain is the **digital signature**. The digital signature is a number that is computed as follows:

- All the rest of the contents of the certificate are fed into a hash algorithm, which generates a single number, which is the hash (rather like a checksum) of those contents.
- The resulting hash value is encrypted using the private key of a public/private key pair.

Most importantly, the private key used in the creation of the signature is typically **not** the certificate owner's private key. Instead it is the private key of a third party—a highly trusted entity known as a Certification Authority (CA).

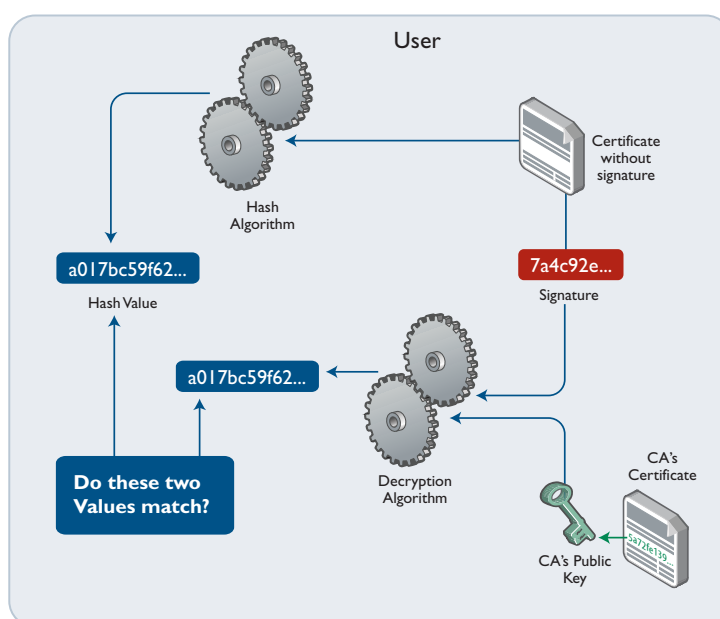
The fact that the certificate has been signed by this CA proves that this CA is satisfied that the certificate owner is who they claim to be. The CA has signed the certificate and given it to the owner.



But, how does your PC know the signature is valid? It is just a number contained in a file, so how can your PC work out that this number was encrypted by the CA?

To do so, your PC needs to already have a copy of the CA's own certificate for the private key they used for signing the web server's certificate. In fact, Microsoft Windows XP ships with a set of X.509 certificates from a number of highly trusted certification authorities.

The CA's certificate will, of course, contain the CA's public key. Using this public key, your PC decrypts the signature. Then, the PC can calculate the hash on the rest of the contents of the certificate, and check that this matches the decrypted signature.



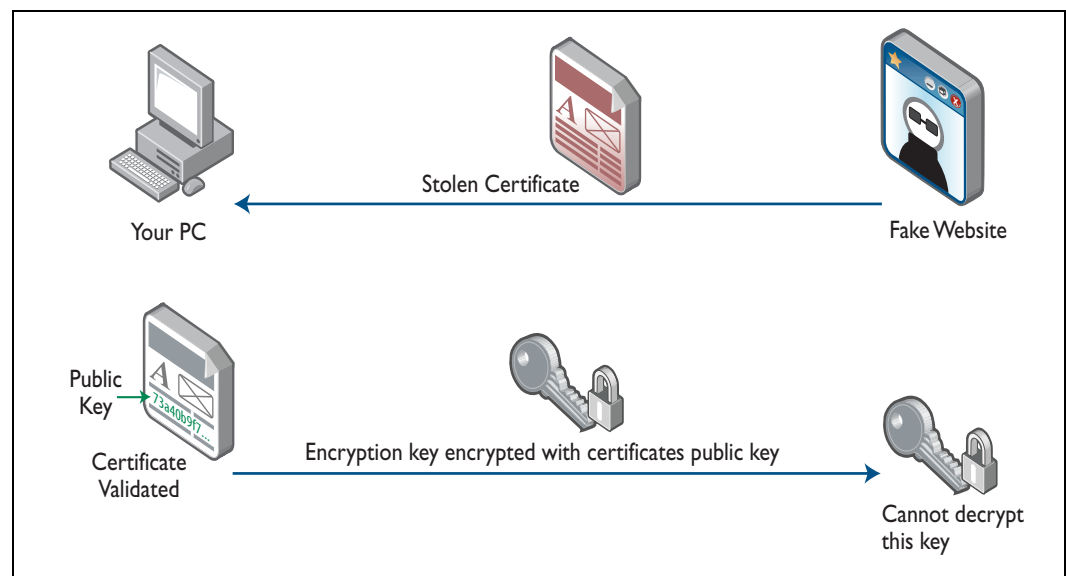
If this all checks out, then your PC:

- knows that the web server's certificate was signed by the CA it claims to have been signed by.
- knows that the CA is satisfied that the website is who they claim to be.
- will follow suit, and trust that the website is who they claim to be.

Certificate signing

"But", you might be thinking, "once the website sends out a copy of its certificate, then anybody can get hold of it, including scammers. Couldn't the scammers just send out the stolen certificate of the website they are spoofing?"

In fact, you would be correct. A certificate is fully available to the public domain. But, the really important point is that only the true owner of the certificate has a copy of the private key corresponding to the public key contained in the certificate. So if the scammer sent a stolen certificate to your PC, then certainly your PC would initially trust the scammer's site. But, when your PC sent the scammer an encryption key encrypted with the certificate's public key, the scammer would not possess the necessary private key to decrypt the key sent from your PC. So, the actual data transfer session would not be able to proceed.



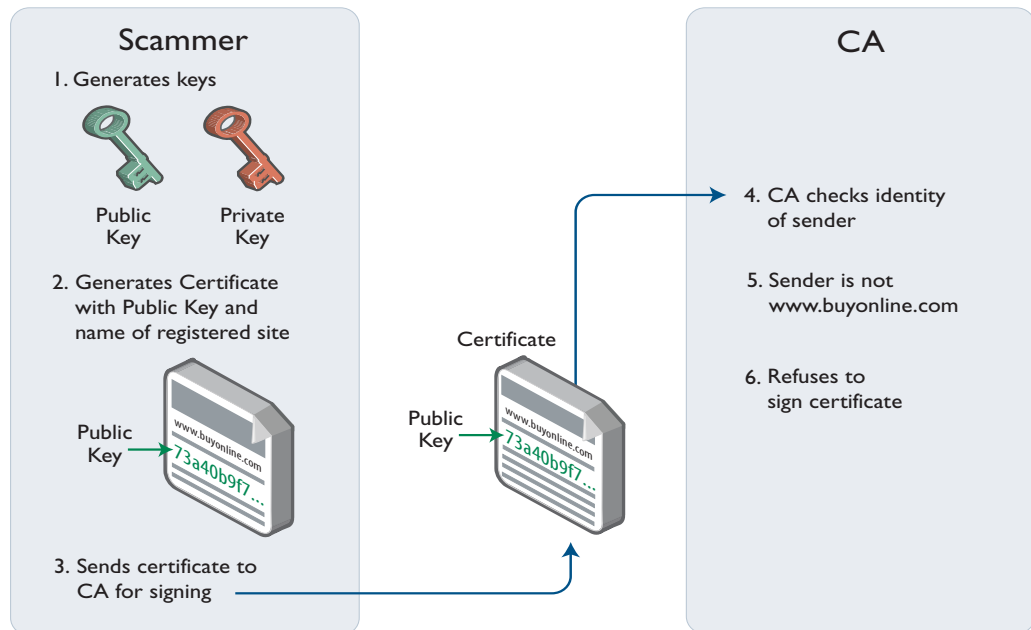
It is probably not entirely accurate to say that a certificate verifies the identity of the entity that sends the certificate to you.

Definition A more accurate statement is that the certificate verifies the identity of the possessor of the private key corresponding to the certificate's public key.

The certificate prevents a scammer from creating their own public/private key pair, and then sending out a certificate containing the public member of that key pair, but showing the owner identity as being that of some valid on-line store. This is because the scammer would not be able to persuade a trusted CA to sign the certificate.

Certainly, we do need to trust in the competence and honesty of the Certification Authorities. If scammers could bribe or trick CAs into signing false certificates, then the whole web of trust falls apart. It is a simple fact of life that no electronic security system can entirely eliminate the human element.

However, the CA's entire ability to stay in business is based on the integrity and trustworthiness of their systems, so they have very strong incentives to continue to deserve our trust.



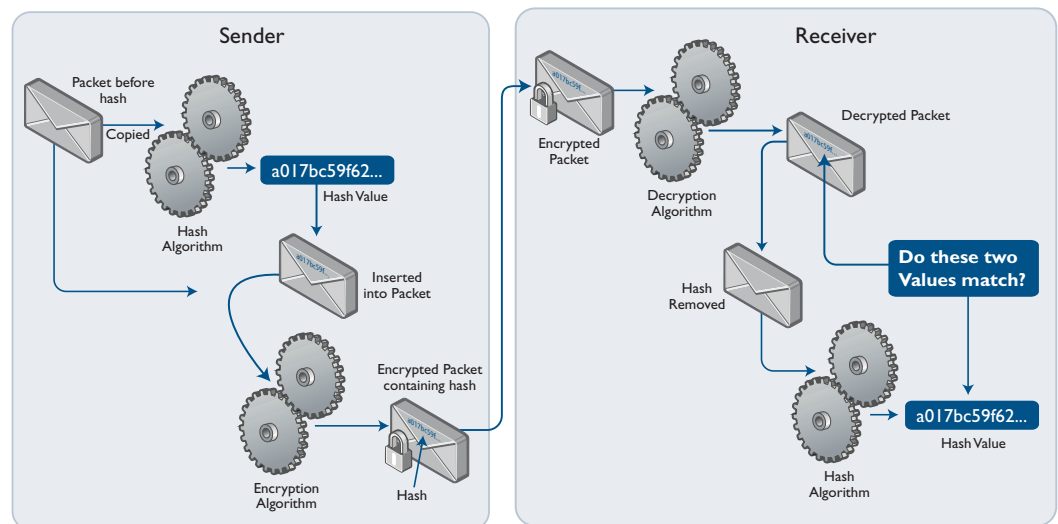
Tamper prevention

There are two aspects of tamper prevention we need to consider. The first is the prevention of tampering with X.509 certificates. The second is the prevention of tampering with the encrypted data session that occurs after the certificate exchange.

The key to tamper prevention is encrypted hashes.

As we have seen above, the signature in a certificate is an encrypted hash. A hash is calculated on the contents of the certificate, then the hash is encrypted using the CA's private key. If a man-in-the-middle intercepts the certificate and alters its contents, then they could recalculate the hash easily enough. But they do not possess the CA's private key, so they cannot correctly encrypt the new hash. So, when your PC receives the certificate, and validates the signature, the signature will not check out, and the certificate will be discarded.

In the subsequent encrypted data transaction, the web server and your PC can agree to use a hash algorithm like MD5 or SHA to calculate a hash on each packets' contents prior to encryption. The hash value can then be included in the packet, and encrypted along with the rest of the packet. Again, if the content of the packet is altered somewhere along its journey, the hash will not be able to be re-encrypted correctly (as only your PC and the web server know the encryption key). When the packet arrives at its destination, the hash contained in the altered packet will no longer be correct, and the packet will be discarded.



Example – 802.1x Authentication with X.509 Certificates

The local RADIUS server within AlliedWare Plus can authenticate 802.1x supplicants either via username and password, or by using X.509 certificates.

This example describes the following steps:

- Creating the server and user certificates using AlliedWare Plus local RADIUS server.
- Configuring ports as authenticator ports on [page 54](#).
- Installing certificates on the supplicant workstation on [page 54](#).
- Setting up the PC's NIC card as an 802.1x supplicant on [page 62](#).
- Attaching the PC NIC to the switch on [page 63](#).

Creating the server and user certificates using AlliedWare Plus local RADIUS server

1. Enable the server

```
awplus(config)# radius-server local
awplus(config-radsrv)# server enable
```

2. Add the switch to the client (NAS) list for the RADIUS server

```
awplus(config-radsrv)# nas 127.0.0.1 key awplus-local-radius-
server
awplus(config-radsrv)# exit
```

When you enable the RADIUS server, this also sets up the switch as a certificate authority, and creates a root Certificate Authority X.509 certificate on the switch. This certificate can be viewed using the command:

```
awplus(config)# show crypto pki certificates local-ca
```

```
awplus#show crypto pki certificates local-ca
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 0 (0x0)
Signature Algorithm: sha1WithRSAEncryption
Issuer: O=Allied-Telesis, CN=AlliedwarePlusCA
Validity
Not Before: Apr 17 05:42:09 2009 GMT
Not After : Apr 12 05:42:09 2029 GMT
Subject: O=Allied-Telesis, CN=AlliedwarePlusCA
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
```

3. Create a RADIUS group and a RADIUS user.

Create a RADIUS user group specifically for the purpose of associating a VLAN with the user. When the user is authenticated on a port, this is the VLAN to which the port will be dynamically allocated:

```
awplus(config)# radius-server local
awplus(config-radsrv)# group Engineers
awplus(config-radsrv-group)# vlan 40
awplus(config-radsrv-group)# exit
awplus(config-radsrv)# user Engineer01 password secret group
Engineers
awplus(config-radsrv)# exit
```

4. Create X.509 certificate. write the certificate write the certificates

1. Create a certificate for the user:

```
awplus(config)# crypto pki enroll local user Engineer01
Enrolling Engineer01 to local trustpoint...OK
```

5. Write the certificates to files, and upload them to a TFTP server to be transferred to the supplicant workstation.

1. Write the Certificate Authority certificate to a PEM file:

```
awplus(config)# crypto pki export local pem url tftp://
10.32.4.73/lrad.pem
Copying..
Successful operation
```

2. Write the user certificate to a PK CS12 file:

```
awplus(config)# crypto pki export local pkcs12 Engineer01
tftp://10.32.4.73/Engineer01.pfx
Copying..
Successful operation
```

Configuring ports as authenticator ports

1. Configure the ports to perform 802.1x authentication.

```
awplus(config)# int port1.0.1-1.0.24
awplus(config-if)# dot1x port-control auto
awplus(config-if)# auth dynamic-vlan-creation
awplus(config-if)# spanning-tree portfast
awplus(config-if)# exit
```

2. Add the switch as a RADIUS server to be used for 802.1x authentication.

```
awplus(config)# radius-server host 127.0.0.1 key awplus-local-
radius-server
awplus(config)# aaa authentication dot1x default group radius
```

3. Create VLAN for user to be allocated to.

```
awplus(config)# vlan database
awplus(config-vlan)# vlan 40
```

The switch is now configured to act as a RADIUS server and 802.1x authenticator.

Installing certificates on the supplicant workstation

You must install both the switch's Certificate Authority certificate and the user's certificate into the PC.

The switch's Certificate Authority certificate must be installed into the PC so that the PC will recognise the switch as a trusted Certificate Authority. Once the PC recognises the switch as a trusted Certificate Authority, it will:

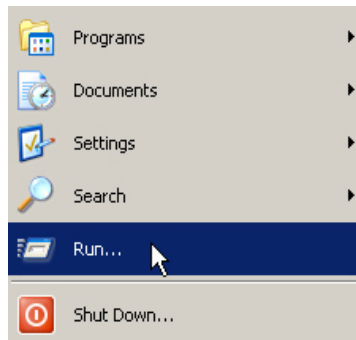
- Recognise the user's certificate as having been signed by a trusted certificate authority (as the user's certificate has been signed by the switch).
- Successfully validate the switch's certificate during the 802.1x authentication.

The PC is configured to request the switch's certificate during authentication, so that it can validate that it is connecting to a trusted authenticator. If the switch's certificate is already installed into the PC as a trusted certificate authority's certificate, then when it receives that certificate again during the 802.1x authentication, it will recognise that certificate as belonging to a trusted authenticator.

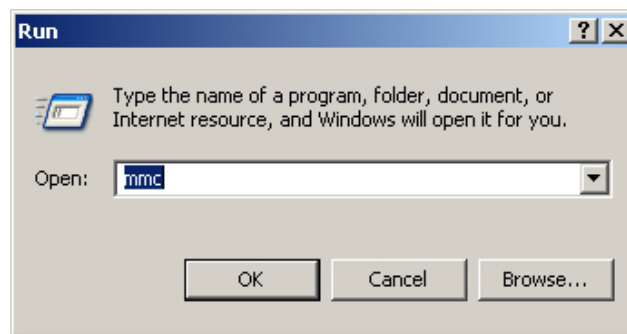
The user's certificate must be installed into the PC so that it can be sent to the switch during the 802.1x authentication.

Preparing to install certificates

1. Select **Run...** from your system **Start** menu.

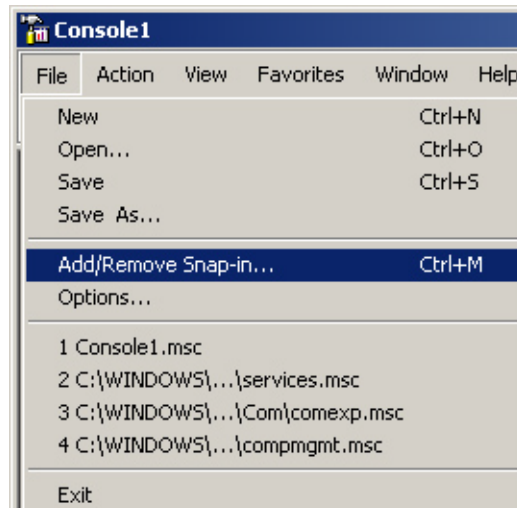


2. Type in **mmc**, and click **OK**.



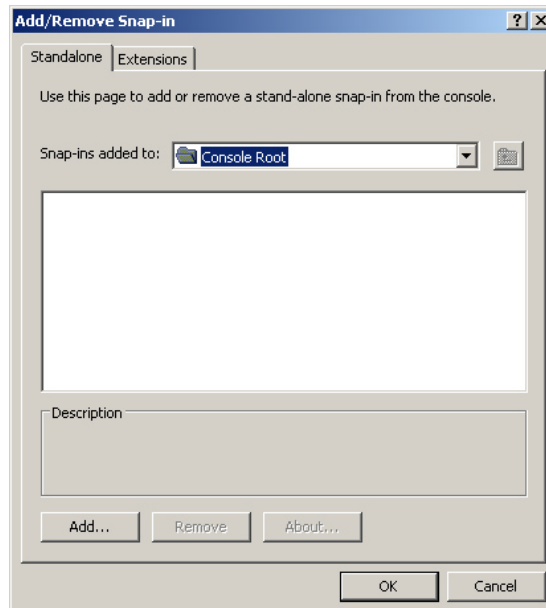
The system **Console** opens.

3. Select **File > Add/Remove Snap-in...**



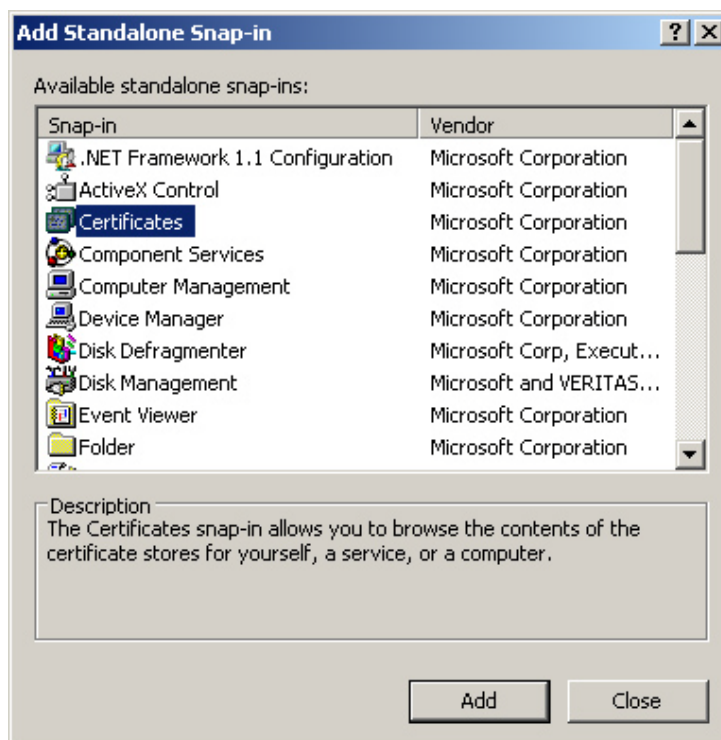
The **Add/Remove Snap-in** window opens.

4. Click **Add...**



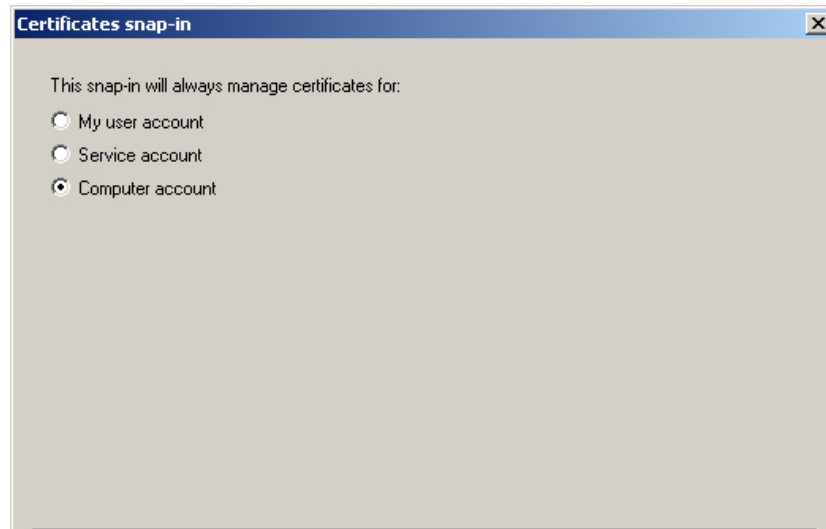
The **Add Standalone Snap-in** window opens.

5. Select **Certificates**, and then click **Add...**



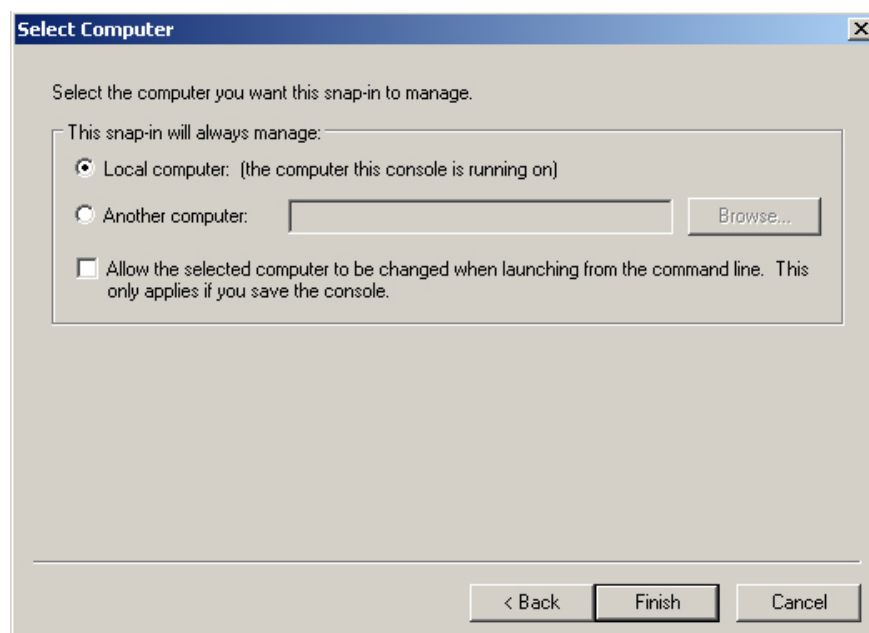
The **Certificates snap-in** window opens.

- Choose **Computer account** and then click **Next**.



The **Select Computer** window opens.

6. Choose **Local Computer** and click **Finish**.



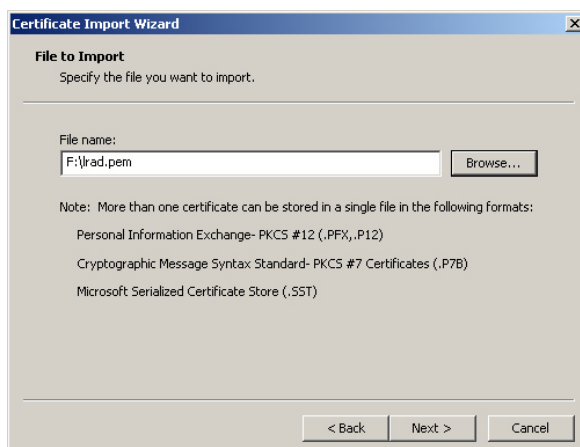
The snap-in is now installed into the System Console. You can now start installing the certificates that you exported from the switch.

Installing the certificates

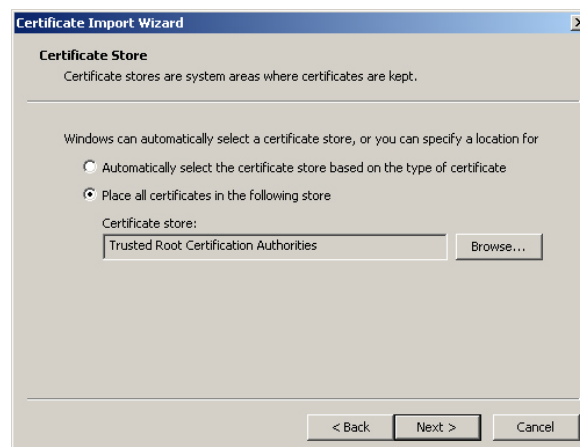
1. Within the console, right-click on **Certificates** under **Trusted Root Certificates**. Then select **All Tasks > Import...**
2. The **Certificate Import Wizard** begins. Click **Next**.



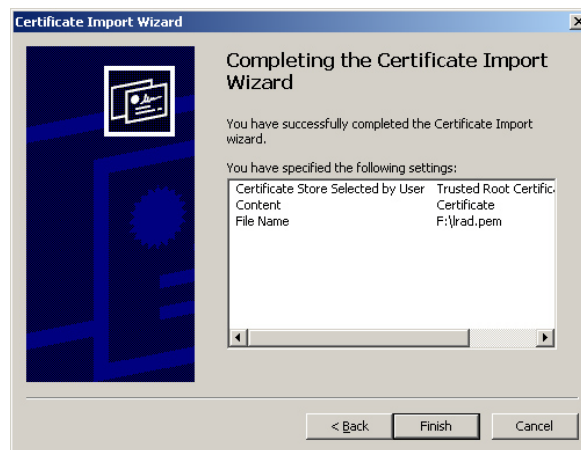
3. In the **File to Import** window, specify the file to which you exported the switch's Certificate Authority certificate. Click **Next**.



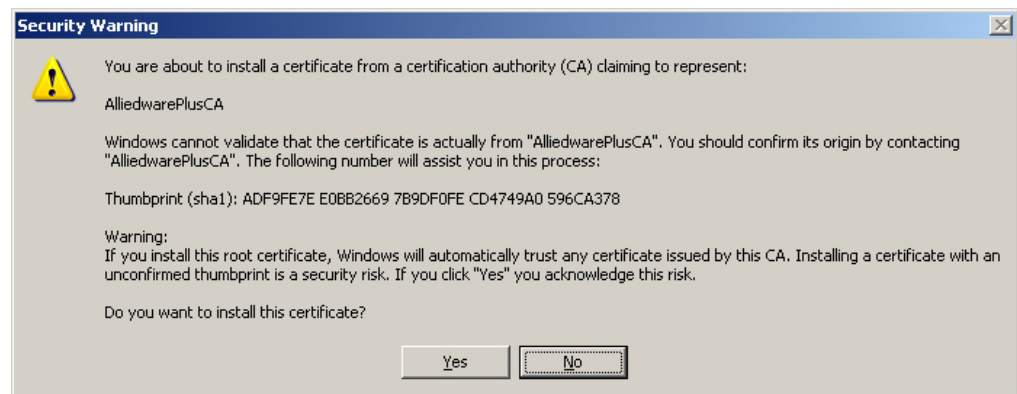
4. In the **Certificate Store** window, leave the default setting and click **Next**.



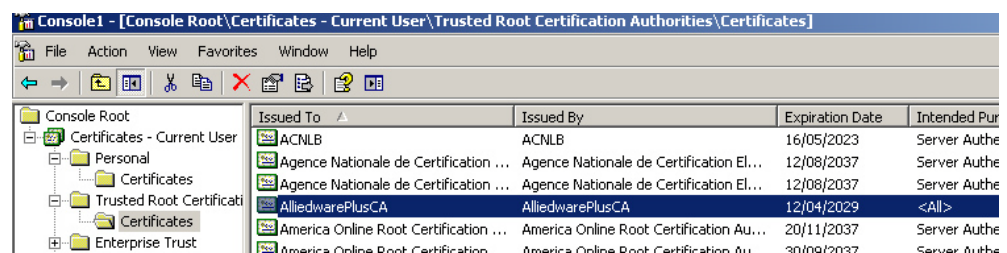
5. The Certificate Import Wizard is now complete. Click **Finish**.



6. A **Security Warning** may display. Since you just created this Certificate Authority on the switch, you know that you can trust it, so click **Yes** and proceed.

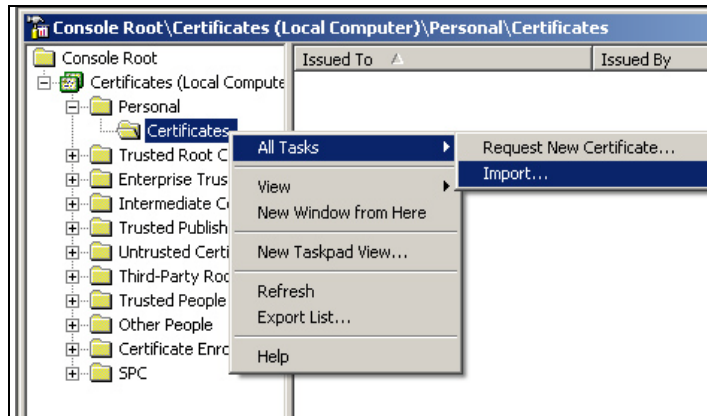


7. The certificate is now installed into the list of Trusted Root Certificates, as shown below:



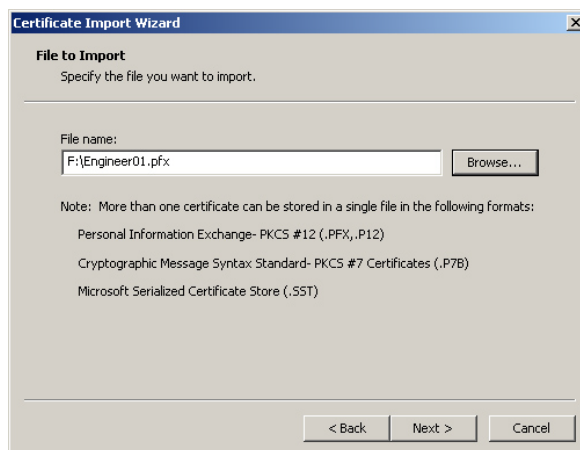
Install the user's certificate

1. Right-click on **Certificates** under **Personal**. Then select **All Tasks > Import...**

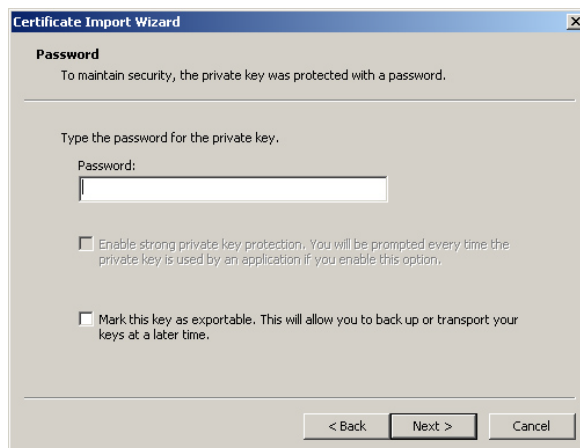


The **Import Certificate Wizard** opens for the second time.

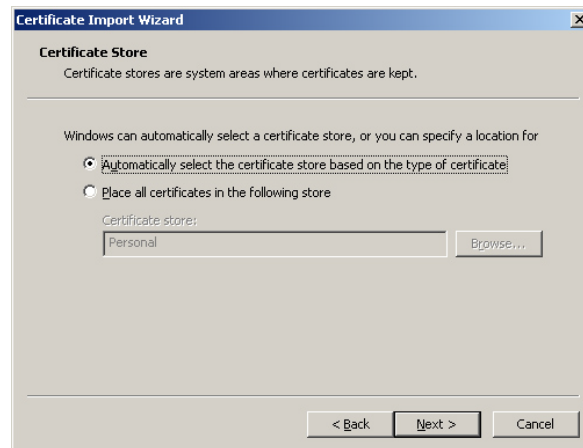
2. Work through this wizard again. This time, specify the file to which you exported the user's certificate.



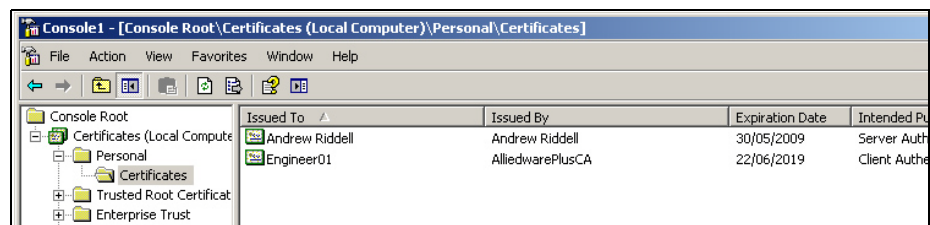
3. The wizard will now prompt you to enter the password that protects the certificate file. The certificate file was not protected with a password, so leave the **Password** field blank, and click **Next**.



4. Choose **Automatically select the certificate store based on the type of certificate** and click **Next**.



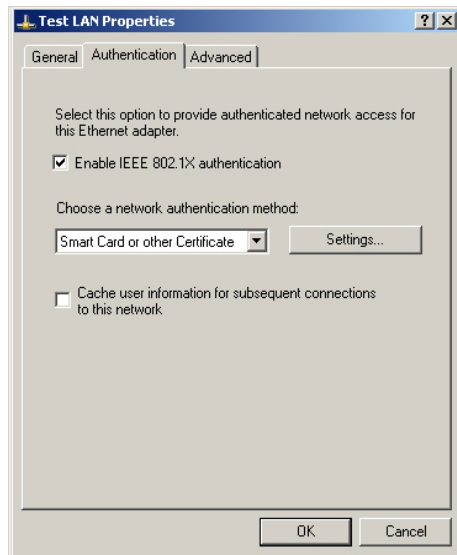
5. The certificate now appears in the **Certificates** store.



The certificates have now been successfully installed on to the PC.

Setting up the PC's NIC card as an 802.1x supplicant

1. Open the NIC's properties window, and select the **Authentication** tab.
 - Select **Enable IEEE 802.1x authentication**.
 - Choose **Smart Card or other Certificate** from the drop down box. Do **not** choose **Protected EAP (PEAP)**.

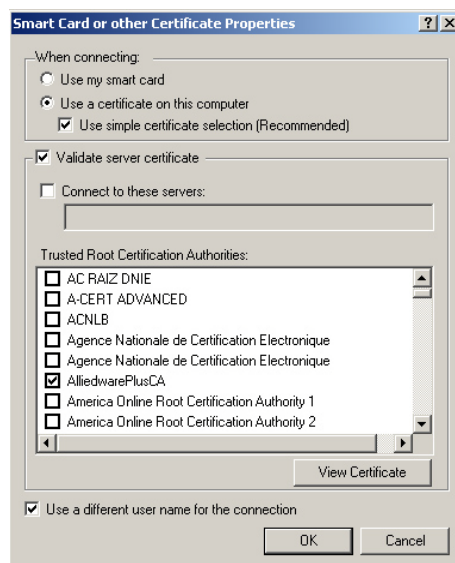


2. Click **Settings...**

The **Smart Card or other Certificate Properties** window opens.

Choose **Use a certificate on this computer**, and select:

- Use simple certificate selection.
- Validate server certificate.
- **AlliedwarePlusCA** from the list of **Trusted Root Certificate Authorities**
- Use a different user name for the connection.



Attaching the PC NIC to the switch

Attach the PC NIC to an authenticating port on the switch. The switch and the PC will exchange certificates, and authentication will succeed. To verify that the PC has been successfully authenticated, use the command:

```
awplus# show dot1x supplicant
```

This will produce output similar to the following:

```
Interface port1.0.1
  authenticationMethod: dot1x
  totalSupplicantNum: 1
  authorizedSupplicantNum: 1
    macBasedAuthenticationSupplicantNum: 0
    dot1xAuthenticationSupplicantNum: 1
    webBasedAuthenticationSupplicantNum: 0
    otherAuthenticationSupplicantNum: 0

Supplicant name: Engineer01
Supplicant address: 0002.b363.319f
  authenticationMethod: 802.1X
  portStatus: Authorized - currentId: 7
  abort:F fail:F start:F timeout:F success:T
  PAE: state: Authenticated - portMode: Auto
  PAE: reAuthCount: 0 - rxRespId: 0
  PAE: quietPeriod: 60 - maxReauthReq: 2
  BE: state: Idle - reqCount: 0 - idFromServer: 6
  CD: adminControlledDirections: both - operControlledDirections: both
  CD: bridgeDetected: false
  KR: rxKey: false
  KT: keyAvailable: false - keyTxEnabled: false
  dynamicVlanId: 40
```

You can see that this output displays the:

- User name under which the PC was authenticated (Supplicant name).
- MAC address of the PC (Supplicant address).
- ID of the VLAN that the port was dynamically allocated to (dynamicVlanId).

Using X.509 Certificates to Authenticate a VPN Tunnel

X.509 certificates are used to provide a public key technique to seed the ISAKMP negotiation between the VPN access concentrator (AT router) and VPN remote host (XP computer). They provide a more secure authentication method than the alternate practice of using a pre-shared key.

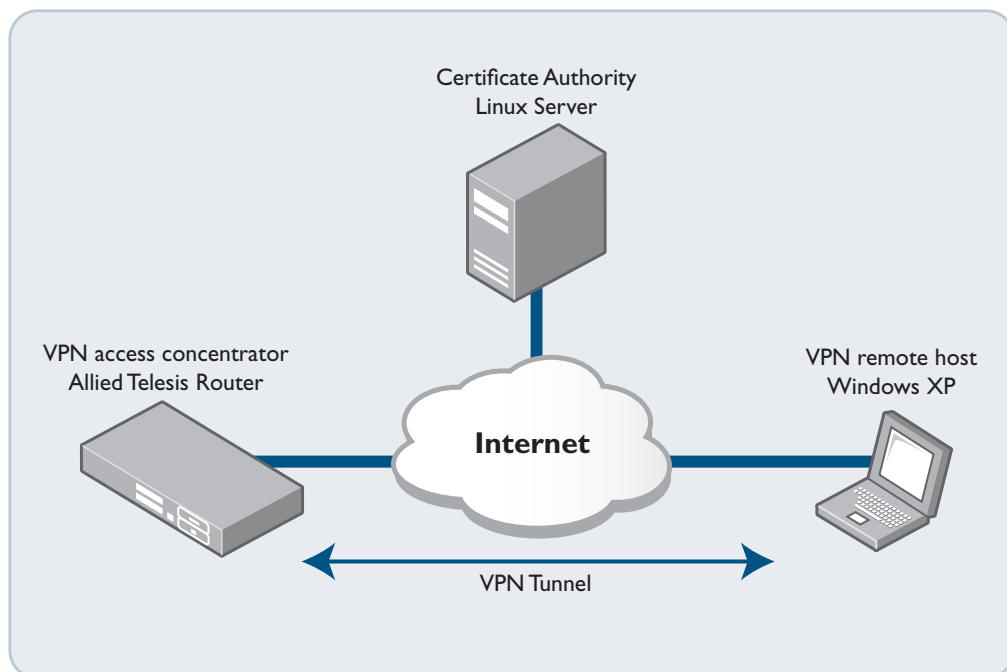
Solution overview

Certificates are required for the following roles:

1. Certificate Authority (CA)—this is the trusted certificate.
2. VPN access concentrator—Allied Telesis router (validated by CA).
3. VPN remote host—Windows XP client (validated by CA).

The Certificate Authority role can be provided by a Linux server, a Windows server, or a third party Certificate Authority paid service.

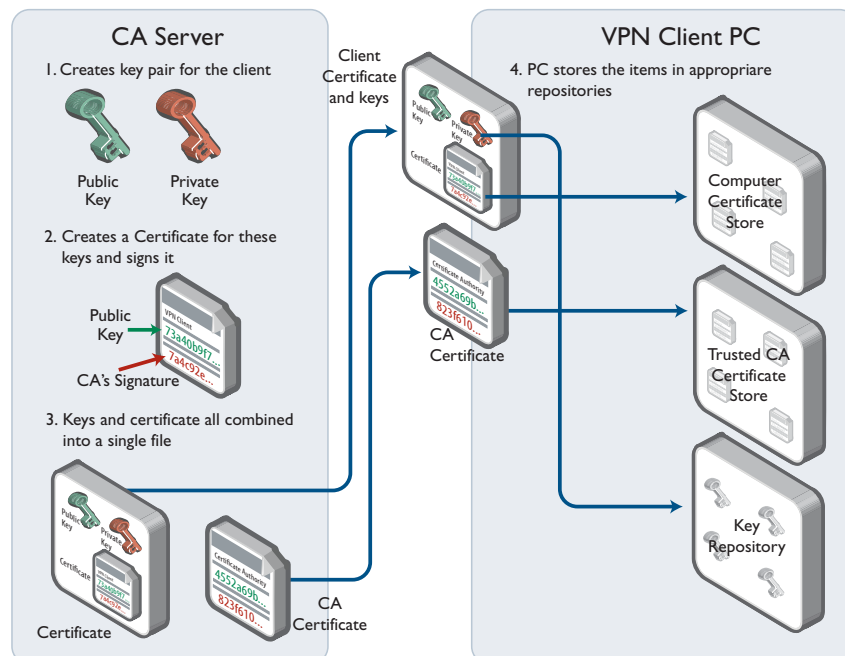
This solution uses a Linux server as the Certificate Authority.



Summary of trust relationship that the certificates create

Just for clarity, here is a brief overview of what is being done to get the right certificates in the right places, and how that enables the VPN access concentrator and the remote VPN client to trust each other's encryption keys:

- The CA certificate is loaded into the **Trusted Root CA** certificate lists on both the VPN access concentrator and the remote VPN client, they both agree they trust this CA.
- On the VPN access concentrator, an RSA key pair is created, and then an unsigned certificate to verify the VPN access concentrator's ownership of that RSA key pair.
- The unsigned certificate is uploaded to the CA server. The CA signs the certificate, and the new, signed certificate is loaded back onto the VPN access concentrator.
- The VPN access concentrator now has a certificate, verifying its ownership of its RSA key pair, signed by a certificate authority that the remote VPN client also trusts.
- The Windows PC cannot create an RSA key pair, so the CA creates an RSA key pair, and a signed certificate for that key pair, on behalf of the remote VPN client.
- The RSA key pair and signed certificate are loaded onto the remote VPN client.
- The remote VPN client now has its own RSA key pair, and a certificate verifying its ownership of that RSA key pair.
- The certificate is signed by a CA that the VPN access concentrator trusts.

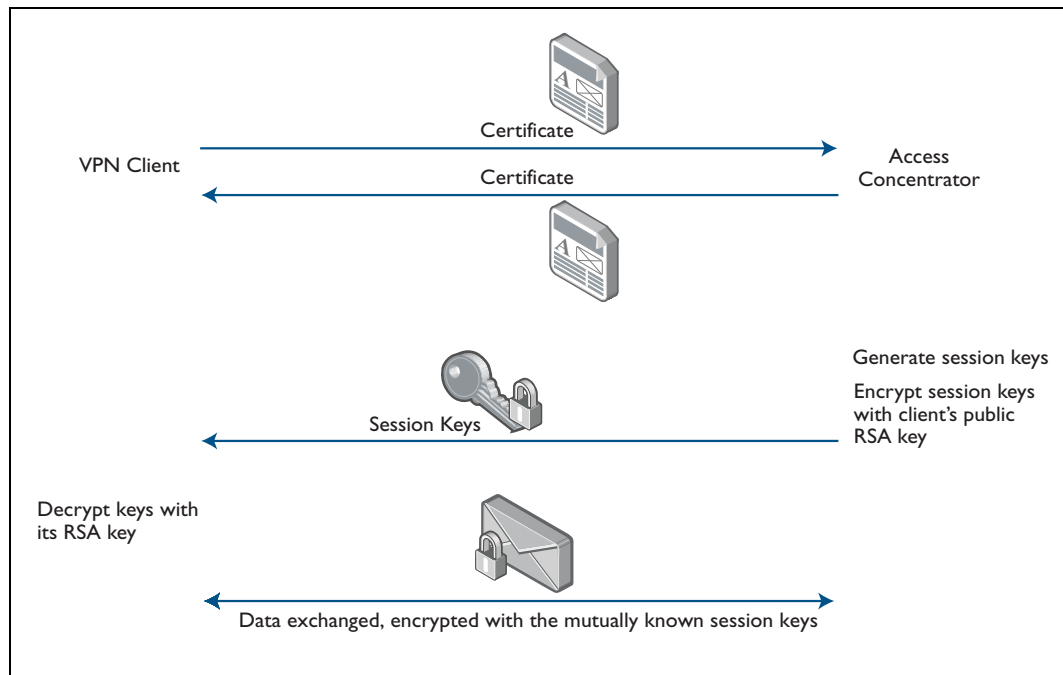


- The VPN access concentrator and remote VPN client each trust that the other device is the true owner of the public key that it sent because they also send each other certificates that verify that they are the true owners of those RSA encryption keys. The VPN access concentrator and remote VPN client trust each others' certificates because the certificates are both signed by the CA they both have the CA's certificate into their list of trusted CAs.

Secure session key exchange

When the VPN access concentrator and remote VPN client want to exchange keys for an encrypted data session:

- They send their public encryption keys to each other.
- One device sends session keys to the other, encrypted with the public key it received from the other device.
- The recipient can then decrypt the session key using its private key.



Chapter 5 | RADIUS

Introduction

The main purpose of the RADIUS protocol is to enable the authentication of network users stored on a server known as a RADIUS server database.

When users connect to the network, the device the users connect to can challenge the users for authentication and pass on the authentication to the RADIUS server to check. Based on the result of the check against the database, the RADIUS server informs the device whether or not to allow the connected user access to the network.

The RADIUS protocol is also used for transferring configuration and accounting information between a Network Access Server (e.g. a router or switch) that desires to authenticate its links and a shared RADIUS server.

This chapter describes the development, purpose, and key elements of the RADIUS protocol.

List of terms

TLV

Type Length Value.

Describes the format of a data element (attribute) in a RADIUS packet.

NAS

Network Access Server.

A generic term for an authenticator that requests the RADIUS server to verify the credentials of hosts that are connecting to it.

Shared Secret

A string of characters known to both devices in a RADIUS conversation, which is used in encrypting some pieces of the exchanged data.

RADIUS proxy

A server that uses the RADIUS protocol to communicate with NASs, but passes the received credentials through to one or more back-end servers that perform the actual validation of the credentials.

RADIUS Overview

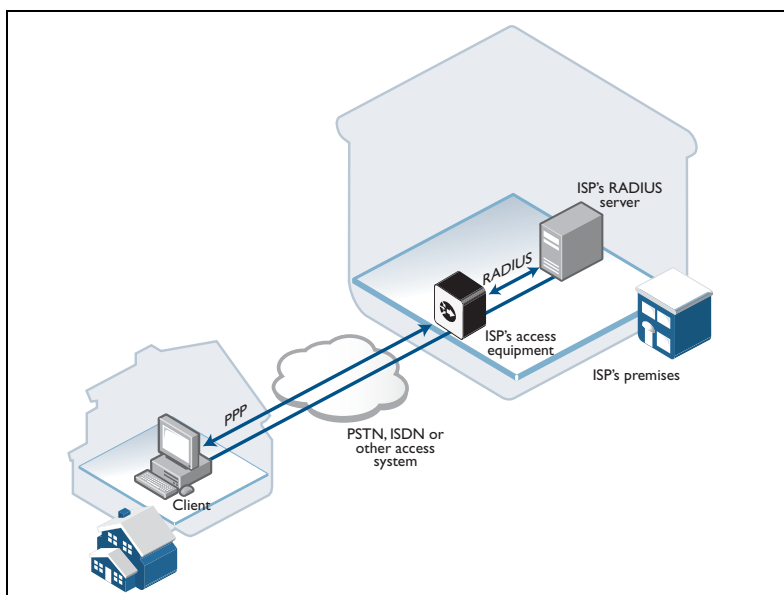
The acronym RADIUS stands for Remote Authentication Dial In User Service. This name harks back to an earlier time, when “Dial In User Services” were commonplace. Despite this name, the RADIUS protocol is by no means falling into disuse. In fact, RADIUS is increasingly a key component in modern Enterprise and Service-Provider security solutions.

The main purpose of RADIUS is to enable network users’ **authentication** credentials to be stored in a central database. Then, when users connect to the network, the access devices via which they connect can challenge them for authentication information and pass this information to the central database to be checked. Based on the result of the check, the central server informs the access switch whether or not to allow the connected user to access the network.

In fact, a RADIUS server can do more than just provide access **authorization**. It can also send back other parameters relevant to the connected user, like an IP address to be allocated to the user, a VLAN into which to place their traffic, a privilege level to allocate to a management session, etc. Moreover, RADIUS provides an **accounting** service. Access devices can tell the RADIUS server how long a user has been connected and/or how much traffic they have sent and received while connected.

Original use of RADIUS - PPP dialup

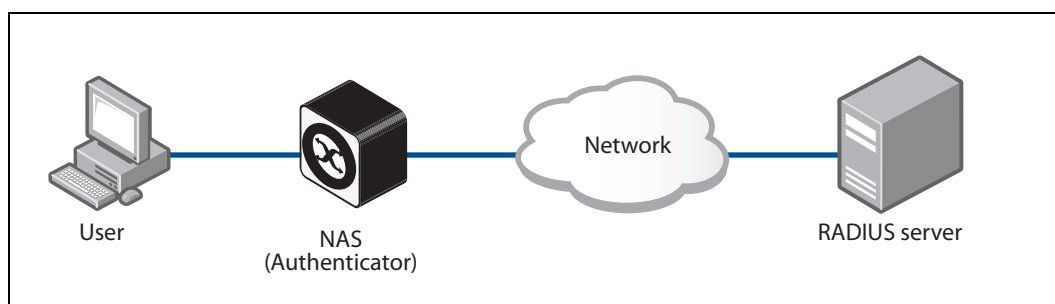
Originally, RADIUS was used for the authentication of users dialling into an (ISP) Internet Service Provider. A PPP (Point-to-Point Protocol) connection would be established between the remote client and the ISP's access equipment. The ISP's equipment would receive the client's username and password by either of the PPP authentication protocols—PAP (Password Authentication Protocol) or CHAP (Challenge Handshake Authentication Protocol)—and then pass those to their RADIUS server to be checked. The RADIUS server's response would be sent back to the client as a PAP or CHAP allow or deny message.



Over time, RADIUS has been adapted to more general network access authentication applications. Implementations of network access authentication using RADIUS invariably follow a model similar to the original PPP dialup application. To describe the component devices that take part in the authentication process, the protocol uses a generalised terminology, that is applicable to any system that uses RADIUS.

RADIUS terminology and components

The central players in a RADIUS interaction are the RADIUS server, where the database of user authentication credentials resides, and the Network Access Server (NAS), which is the access device that the user connects to.

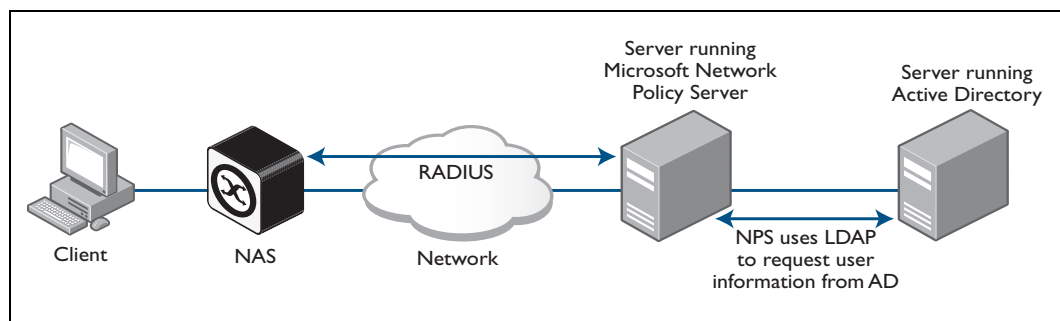


The RADIUS server is configured with a list of the IP addresses of valid NASs that are allowed to send authentication requests to it. The server will not accept requests from devices that are not on the NAS list.

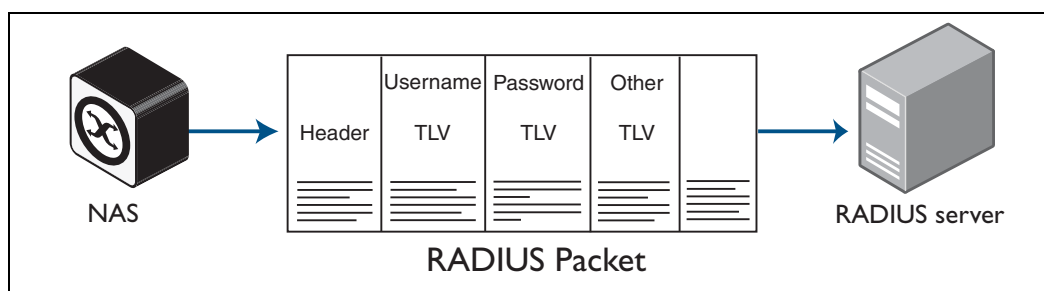
Each NAS has an associated **shared secret**, a pre-shared key that is used to authenticate the requests. The authentication mechanism is described below in "[RADIUS packet exchange](#)" on page 70.

The server also needs access to a list of user authentication credentials. This list could be stored within the server itself, or it could be accessed from another server.

For example, the Microsoft Network Policy Server does not hold its own list of user authentication credentials, but will request user information from an Active Directory server.



The server and the NAS communicate with each other by the exchange of attributes. Every piece of information they exchange is treated as an attribute—even usernames and passwords. Studying the exchange of attributes is central to understanding the operation of the RADIUS protocol.



Understanding the RADIUS protocol

The RADIUS protocol uses UDP packets to provide communication between the NAS and the RADIUS server. There are two UDP ports typically used as the destination port for RADIUS authentication packets—ports 1645 and 1812. Port 1812 has been more commonly used in recent years.

Similarly, ports 1646 and 1813 are used for RADIUS accounting. The protocol, the core packet types, and the core attributes, were defined in a series of RFCs between 1997 and 2000. The RFCs for RADIUS authentication were 2058, 2138, 2865, and 2868. The RFCs for RADIUS accounting were 2059, 2139, 2866, and 2867.

The RFCs define six RADIUS packet types and over 50 attributes:

Packet Type	Description
1	Access-Request
2	Access-Accept
3	Access-Reject
4	Accounting-Request
5	Accounting-Response
11	Access-Challenge

Let us first look at the packet exchange in a RADIUS conversation, then consider some of the core attributes. We will then look at security aspects of the protocol and follow that with a brief survey of some of the other attributes.

RADIUS packet exchange

A RADIUS exchange is initialised by the NAS when a user or client device has requested access to the NAS. The NAS obtains the requestor's authentication credentials, puts them into a RADIUS Access-Request packet, and sends that packet to the RADIUS server. If the server has not been configured to receive requests from the NAS, it will silently discard the Access-Request packet.

If the server accepts the request from the NAS, it then considers the authentication credentials provided in the Access-Request packet. It might look up the user in its own database, or it may have to consult other servers for verification of the user.

If the server decides that the user is invalid or is not allowed access to the NAS, then it responds with an Access-Reject, and the NAS will block the user or device that was requesting access.

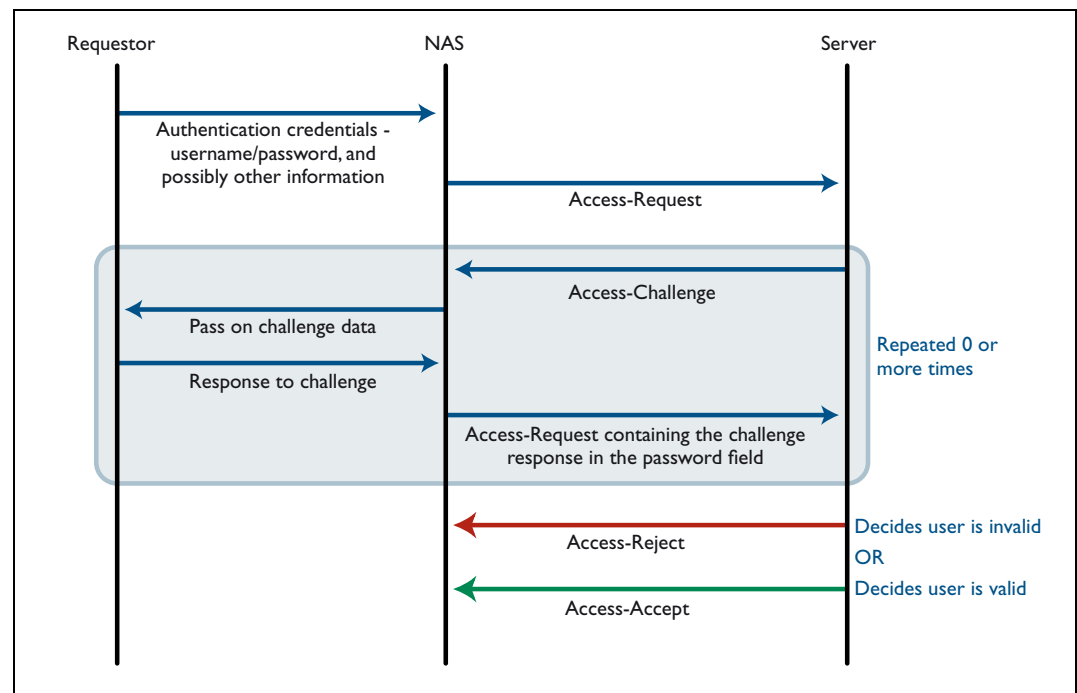
If the server is configured to accept the user as valid, but it needs to obtain more information to verify that the requestor is not an imposter masquerading as a valid user, it might enter into a challenge/response exchange.

In a challenge/response exchange:

1. The server sends an Access-Challenge packet to the NAS. This packet typically contains a randomly generated number that the NAS passes on to the requestor.
2. The requestor is expected to encrypt this number using software or a smart card that would only be available to a valid user.
3. The result of the encrypting of the number is passed back to the NAS, which then puts this result into the password field of a new RADIUS Access-Request packet that is sent to the server.
4. At this point, the server may decide to accept or reject the requestor, or it may decide to send another challenge (going back to step 1. above).

Eventually, the server will decide to accept or reject the requestor, after 0 or more challenge/response exchanges. As described above, if the server rejects the user, it sends an Access-Reject packet back to the NAS.

If the server decides to accept the requestor, it sends an Access-Accept packet to the NAS. This packet will typically contain a set of attributes that the NAS can apply to the access session as appropriate.



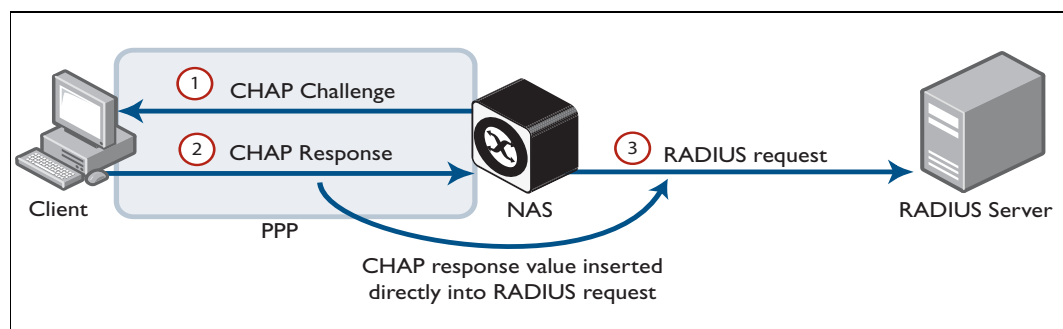
Core RADIUS attributes

In this section a selected set of commonly used attributes are described. Each attribute is identified by its RFC-defined name, followed by its attribute ID in parentheses.

Core RADIUS Attribute	Description
User-name (1)	<ul style="list-style-type: none"> User-name has attribute ID 1, which indicates how central it is to the whole operation of RADIUS. RADIUS is about identifying and validating users, so the first thing you need to have is a user-name. User-names are strings of at least 3 characters. The upper limit on the length of a user-name string that RADIUS can accept is 253, which is the upper limit on the length of all RADIUS attributes.
User-password (2)	<ul style="list-style-type: none"> This attribute can be used either for the password that a requestor entered at their initial authentication attempt, or for the response to an Access-Challenge. The user-password attribute is always carried in encrypted form in a RADIUS packet. The encryption of the password is performed as an MD5 hash of a rather complicated combination of the password, the NAS's shared secret, and a per-packet random value called the request authenticator. See "Password encryption" on page 74.
CHAP-password (3)	<ul style="list-style-type: none"> In the CHAP authentication protocol, the requestor does not actually send its password to the authenticator. Instead, it sends an encrypted string that is an MD5 hash of a combination of the password and a challenge value sent by the authenticator. If the NAS is using CHAP to authenticate the requestor, it does not receive the requestor's password, but sends the chap challenge result to the RADIUS server, instead. See the "Chap authentication diagram" on page 73.
Framed_IP-Address(8)	<ul style="list-style-type: none"> Dial-in users making PPP connections to the NAS are often dynamically allocated an IP address to be used for the duration of their connection. The Framed-IP-Address attribute, is the attribute that the server sends back to the NAS to inform it of the address to allocate to the dial-in user.
Service-type(6)	<ul style="list-style-type: none"> This attribute is mostly relevant when the NAS is authenticating a user who wishes to open a management session on the NAS itself. It is typically sent by the server back to the NAS in an Access-Accept packet, to indicate what level of management access the NAS should give to the user. For example, AlliedWare Plus maps the Service-Type value 6 (referred to as Administrative) to a privileged management session. The Service-Type attribute can be sent in an access-request packet to request a level of service, but that usage is not so common.
NAS-port-type(61)	<ul style="list-style-type: none"> This attribute identifies the type of port on which the requestor is accessing the NAS. It is sent by the NAS to the server in Access-Request packets, as the server may use this value to choose between different access policies. For 802.1x sessions, the NAS-Port-Type sent by the NAS is Ethernet (15).
Tunnel-type(64)	<ul style="list-style-type: none"> This attribute was originally defined in RFC2868, for use with various tunnelling methods like L2TP, GRE, IPSEC, etc. For the purposes of 802.1x VLAN assignment, RFC3580 defined "VLAN" (13) as a new value for this attribute. The use of this attribute for VLAN assignment will be discussed below in "Extensions in later RFCs" on page 77.

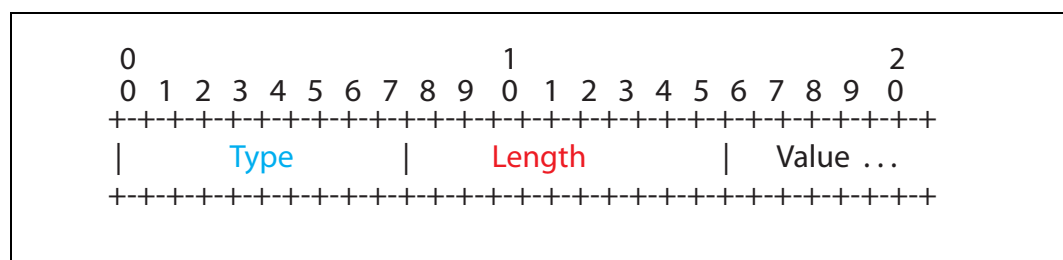
Core RADIUS Attribute	Description
Tunnel-medium-type(65)	<ul style="list-style-type: none"> This is another attribute that was originally defined in RFC2868 for use with tunnels over IP networks. Its purpose was to indicate the medium over which the tunnel was being created. Again, this attribute has been seconded for use in 802.1x VLAN assignment.
Tunnel-private group(81)	<ul style="list-style-type: none"> This attribute was also defined in RFC2868 for use with tunnels, but was given another use in the context of 802.1x VLAN assignment by RFC3580.

Chap authentication diagram



TLV format

Attributes are carried within RADIUS packets in the form of TLVs (Type Length Values). Every attribute's TLV starts with a Type field, which holds the attribute ID number. Then it has a Length field, which holds a one-byte number that represents the whole length of the TLV (not just the length of the attribute value). Then it ends with a Value field, which holds the actual value of the attribute itself.



Security aspects of the RADIUS protocol

As the RADIUS protocol is involved in network security, and carries user authentication information, it is a prime target for security attacks. In particular, RADIUS messages are likely to be the subject of information stealing attacks or session spoofing attacks. Therefore, it is important that the protocol be secure. There are three main security elements in the protocol: shared secret, authenticator, and password encryption.

These security processes are shown in diagrams on [page 76](#).

Shared secret

As mentioned earlier, every NAS and server are configured with a pre-shared key, called the **shared secret**. This is a string, with no particular format. The RFCs define no maximum or minimum lengths defined for the shared secret, but it is recommended that it be at least 16 characters in length, and most RADIUS implementations put a maximum length on the shared secret they will accept—maybe 64 characters or 128 characters.

There is no mechanism within the protocol for choosing and sharing the secret between the NAS and the server. The secret must be manually generated and separately configured on the NAS and on the server.

The shared secret itself never appears in any RADIUS packets, but as described below, it is used as an input to the algorithms used for creating encrypted values that are carried in the packets.

Authenticator

The authenticator is a random 16-byte value that is generated by the NAS. The NAS creates a new authenticator value for each Access-Request that it sends. The authenticator is sent in the Access-Request packet.

The response packets that come back from the server contain a value called the Response Authenticator. This value is created by performing an MD5 hash on a string that is created by concatenating the:

- packet type identifier (Access-Accept = 2, Access-Reject = 3, etc.)
- session ID
- authenticator sent in the Access-Request packet
- attribute fields in the packet
- shared secret that the server shares with the NAS to which it is responding

When the NAS receives the response packet, it performs the same hash on the same values, and verifies that it comes up with the same result. If not, then it must assume that the response packet has been spoofed, and silently discards it. See the diagram "[The RADIUS authentication processes](#)" on page 76.

Password encryption

The value placed in the user-password TLV of an Access-Request packet is not simply an exact copy of the password sent from the requestor to the NAS. Instead the NAS:

- concatenates together the shared secret and the authenticator that it has randomly generated for this request.
- performs an MD5 hash on this concatenated string, to create a 16-byte value.
- performs an XOR between this MD5 result and the requestor's password.
- places the result of this XOR into the user-password TLV.

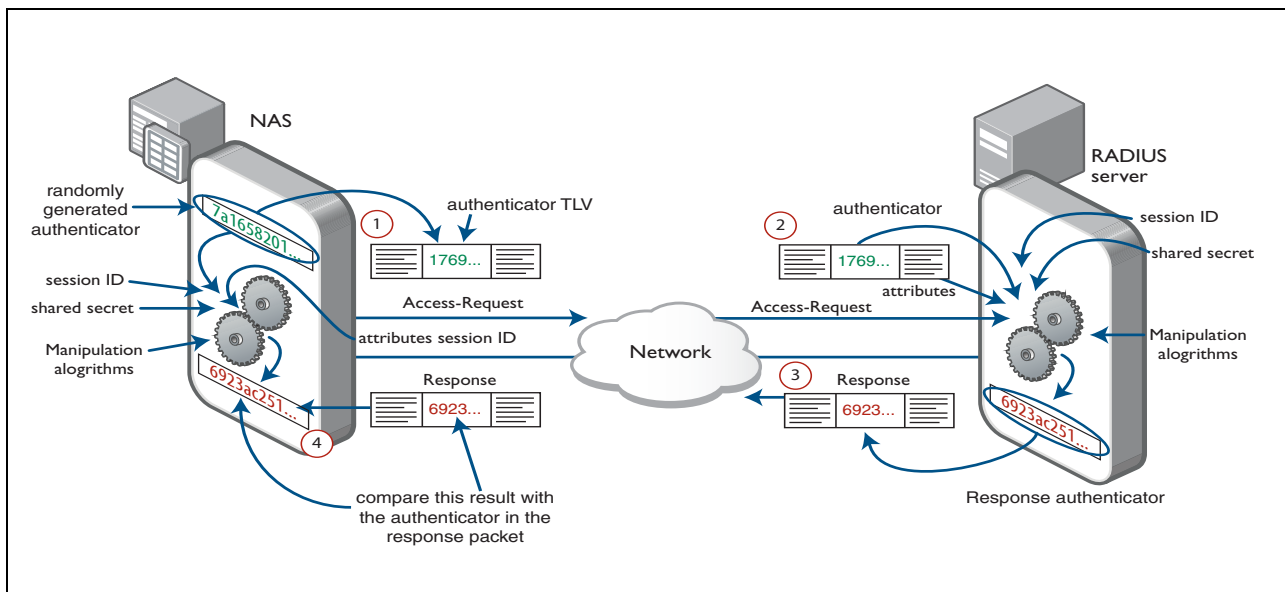
If the password is more than 16 bytes long, then the NAS:

- concatenates together the shared secret and the result of the first XOR.
- performs an MD5 hash on this concatenated string to create a 16-byte value.
- performs an XOR between this MD5 result and the second 16 bytes of the requestor's password.
- appends the result of this XOR to the user-password TLV.

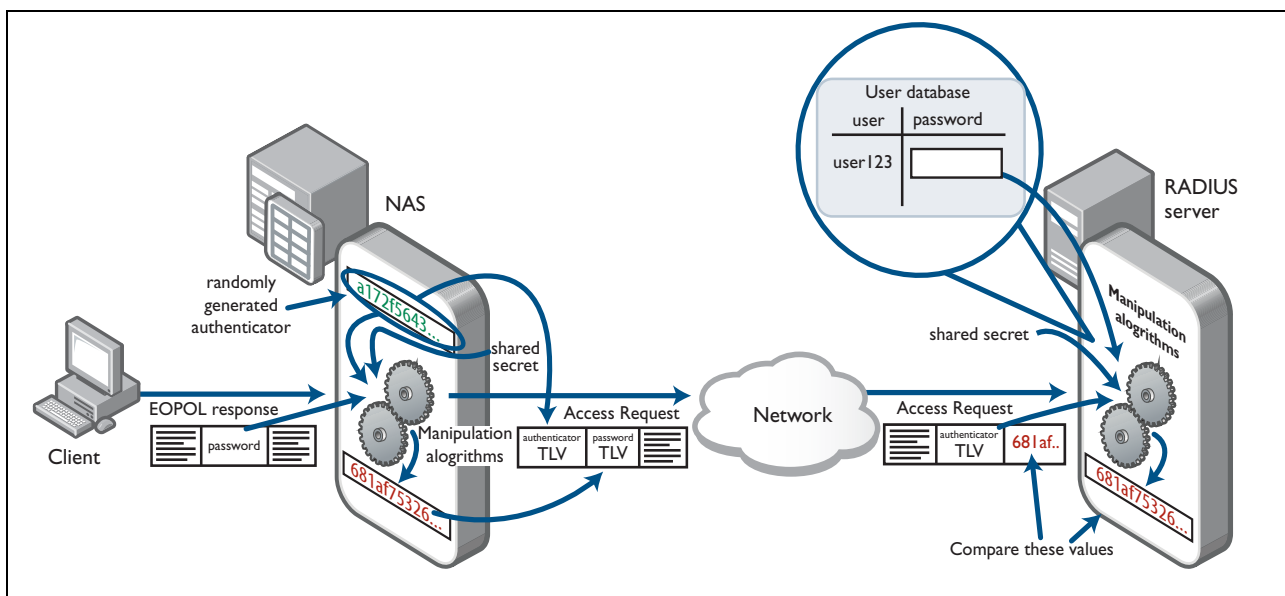
The process continues until an XOR (one or the other but not both) has been performed with the whole of the requestor's password.

When the server validates the Access-Request, it retrieves the user's password from the user credentials database, and performs the same manipulation upon that password. If the result matches the value in the user-password field of the Access-Request, then the password sent by the requestor is deemed to be correct. See the following RADIUS process diagrams.

The RADIUS authentication processes



The RADIUS password encryption processes



Extensions in later RFCs

Implementations of RADIUS servers and NASs rapidly diverged from the functionality defined in the original RFCs. Over time, a number of RFCs have been written in order to try and apply some level of standardisation to all these vendor-lead extensions to RADIUS.

The most obvious manner in which RADIUS has been extended has been the proliferation of attributes. A full list of all the attributes that have been standardised in RFCs is provided at, <http://freeradius.org/rfc/attributes.html>. Very usefully, this page provides a link to the relevant section of the RFC in which each attribute has been defined.

Another significant area of RADIUS extension has been the support for EAP and 802.1x. As described in the section "[Communication with the authentication server](#)" on page 119 the key for EAP support in the protocol was the creation of the EAP attribute. This meant a NAS could simply receive an EAP packet from a requestor, place this EAP packet into the EAP attribute of a RADIUS packet, and send that to the RADIUS server. The server would, of course have to know how to interpret the EAP packet, but the NAS did not have to understand it at all. Less successfully, a number of additional RADIUS packet types have been created to support additional functions like:

- requesting the NAS to reboot.
- requesting the server to change the password for a user.
- having the server send out strings that the NAS will present to requestors as a menu.
- using the RADIUS server to serve out information like messages of the day, IPX routes, etc.
- enabling the RADIUS server to base its accept/reject decisions on not just the validity of user credentials, but on other policies like the total number of current active users, bandwidth usage, number of dynamic tunnels in operation, etc.

A number of these functions are described in RFC 2882. Not many of them have remained in popular usage.

RADIUS proxy

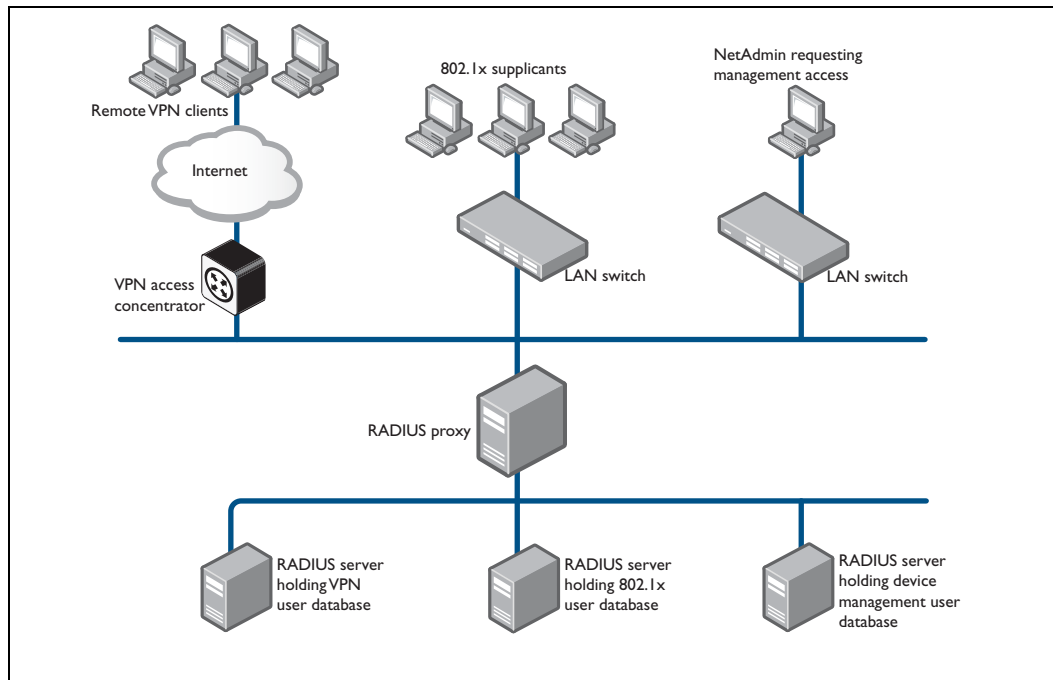
As mentioned above in the section "[RADIUS terminology and components](#)" on page 69, user credentials can be stored in a user database that does not reside on the RADIUS server itself.

Communication from the RADIUS server to such an external user database server is frequently performed by LDAP (Lightweight Directory Access Protocol). However, the external user database can reside on another RADIUS server, and the communication to that server can simply use RADIUS.

If a RADIUS server communicates with a NAS, but also acts as a client to another RADIUS server, it is said to be acting as a **RADIUS proxy**.

There a variety of situations where RADIUS proxy is useful.

Consider a case where multiple RADIUS servers have been set up to hold user databases for different purposes—authenticating switch management sessions, authenticating VPN connections, authenticating 802.1x sessions, etc. However, for convenience, all the NASs in the network are configured with only one address as their RADIUS server. In that case, the NASs send their requests to one RADIUS server, and that server acts as a proxy for all the servers that are holding the different user databases.



RADIUS accounting

For a variety of reasons—billing, capacity planning, usage auditing, etc. the manager of a network may wish to collect information about how long users are connected to the network, and how much data they transfer while connected.

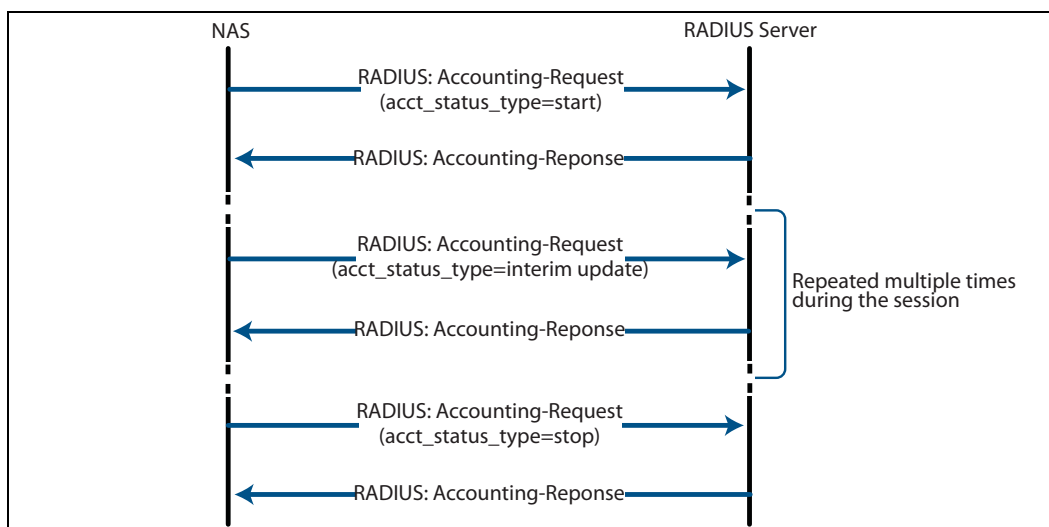
The RADIUS protocol provides a facility for this information collecting, known as RADIUS accounting. The RADIUS accounting protocol exchange is quite simple.

There are only two types of RADIUS accounting packet—Accounting-**Request** and Accounting-**Response**. The Accounting-Request packets are always sent from the NAS to the server. The Accounting-Response packets are always sent from the server to the NAS, and are effectively acknowledgements (ACKs) of the Accounting-Request packets.

The Accounting-Request packets always carry the attribute **Acct-Status-Type**. The most commonly used values of this attribute are:

- **Start**—a packet marking that a session is beginning.
- **Stop**—a packet marking that a session is ending.
- **Interim update**—packets sent periodically during the session to give update reports on the statistics that are being collected.

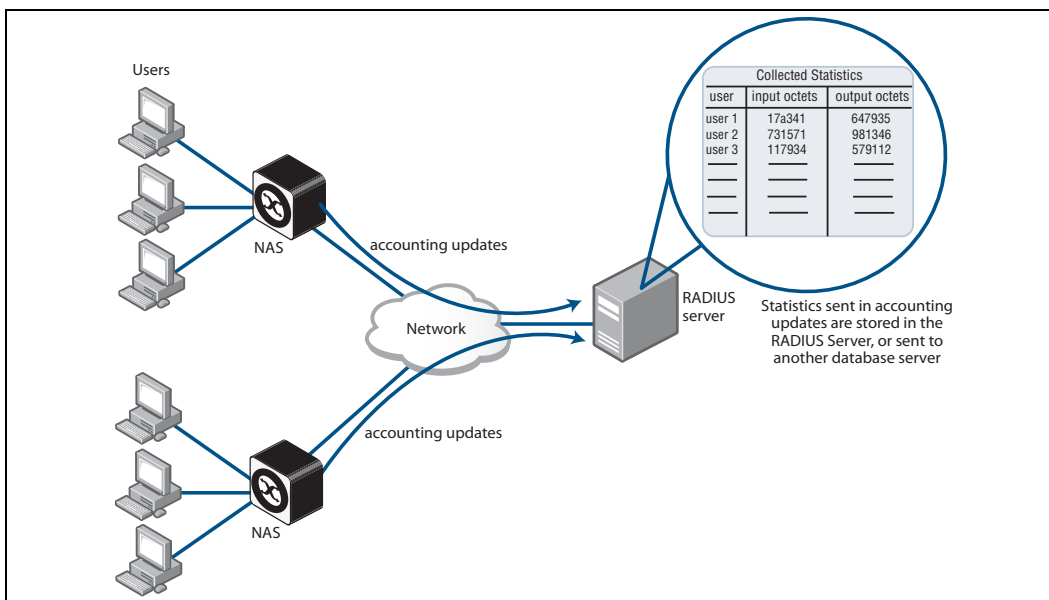
A typical RADIUS accounting session follows the pattern below:



The statistics that can be exchanged in the session are:

- Input octets
- Output packets
- Input packets
- Session duration
- Output octets

There is no requirement to exchange all these statistics, NAS implementations are at liberty to choose which statistics they will send. Each of these statistics has a corresponding attribute type.



The attributes are sent in Interim-Update and Stop accounting request packets. Each accounting session has a unique session ID, which is chosen by the NAS. The session ID is carried in an Acct-Session-ID attribute, which should be present in every packet involved in the session.

The accounting packets typically do not use the same UDP port as the authentication packets. The default port for RADIUS accounting is **1813**.

RADIUS accounting on AlliedWare Plus

AlliedWare Plus switches send three types of RADIUS accounting packet:

1. Accounting **start** packets, to inform the RADIUS server that a session has begun.
 - These packets tell the server the session ID, the name of the user authenticated for this session, and the type of port the session is using (Ethernet or “virtual”)
 - These packets are sent if the accounting method list is configured with start-stop
2. Accounting **stop** packets, to inform the RADIUS server that a session has ended.
 - For login sessions—sent when the user logs out or is automatically logged out due to idle timer; shows how long the user was logged in.
 - For dot1x/ auth-mac sessions—sent when workstation disconnects or when EAPOL-logoff sent.
 - For web-auth sessions—sent when the workstation disconnects or the user clicks the logoff button in the web-auth browser page; shows how much data was exchanged in the session.
 - These packets are sent if the accounting method list is configured with start-stop or stop-only.
3. Accounting **interim-update** packets, to give the RADIUS server an update on data sent/received during a dot1x/ auth-mac/auth-web session.

The interim-update packets:

- are never sent for login sessions
- are sent periodically
- have a configurable send
- have a period time

The table below provides a description for each of the RADIUS accounting packet types: Login, Dot1x, MAC auth and Web auth.

Login	Description
Start Packet	NAS port type = virtual NAS port number = session ID
Stop Packet	Reports duration of session
Interim update packet	Not sent
Dot1x, MAC auth and Web auth	
Start Packet	NAS port type = Ethernet NAS port number = ifindex of supplicant's port
Stop Packet	Reports packets and bytes sent& received on supplicant's port during session
Interim update packet	Reports packets and bytes sent and received on supplicant's port so far in session

Chapter 6 | TACACS PLUS

Introduction

The acronym TACACS stands for Terminal Access Controller Access-Control System. This rather tautological-sounding phrase is an accurate description of what the protocol is. It is the system by which an access controller implements its access control. The use of the word Terminal in the phrase does date the protocol a bit. The protocol is used to control access to all manner of devices these days, very few of them being terminals.

The purpose of TACACS is to communicate AAA information between network devices and a central server. TACACS is built around the concept of Authentication, Authorization, and Accounting, and implements these three processes quite separately from each other.

List of terms

Session

A single act of information exchange with a TACACS server. Several sessions can occur in one TCP connection.

Service

The type of action that a TACACS session relates to.

Evolution from TACACS to TACACS+

The original TACACS protocol was developed decades ago, and was used in the ARPANET (Advanced Research Projects Agency Network), the network that became the basis for the Internet. It was used for validating the identity of users connecting remotely into ARPANET nodes. As was typical of the time, the protocol was simple, and highly insecure.

In the 1980s, Cisco Systems picked up TACACS, and worked on improving it. Their improved version became known as XTACACS (a contraction of eXtended TACACS). Cisco introduced XTACACS to the market in 1990. The specification eventually became public as RFC1492 in 1993. The introduction section of this RFC is a brief, but rather amusing, life story of TACACS up to that point.

Cisco did not stop at XTACACS. They revamped the protocol significantly and effectively created a new protocol, which they called TACACS+. The specification of TACACS+ is documented in 1997 in an Internet draft: **draft-grant-tacacs-02.txt**. It has never achieved full RFC status.

This chapter will describe only TACACS+, as TACACS and XTACACS have fallen into disuse, and there is no longer any value in studying the operation of those protocols.

Overview of TACACS+

The TACACS+ protocol consists of three separate sets of packet definitions—one for each of the processes of Authentication, Authorization, and Accounting.

The three processes are able to operate quite independently of each other, with no process dependent on one of the others having already been performed. Moreover, a client can use different servers for different processes. For example, a client can use one server to authenticate a user that is logging in, a different server to request authorization for that user to carry out certain actions, and yet another server for the accounting records that keep track of how long the user was connected.

Protocol description

TACACS+ uses Transmission Control Protocol (TCP) as its transport protocol. The default port number to which TACACS+ packets are sent is 49. TACACS+ servers listen on port 49 for all session types—Authentication, Authorization, and Accounting.

Sessions

A TACACS+ client communicates to the server in conversations that are called sessions, with each session carrying out an individual AAA exchange. The act of authenticating a user or authorising a command occupies a single session.

Each such session has a session identifier. Each packet within the session has a sequence number. The sequence numbers in a session always start from 1, and increment with each packet.

There is not necessarily a one-to-one relationship between TACACS+ sessions and the TCP connections over which they are carried. Multiple sessions may occur within a single TCP connection. A client may hold open a TCP connection to a server, and carry out a series of authorization sessions through this connection over a period of time.

As we will see below, authorization and accounting sessions are very brief—just a single request and a single reply. Authentication sessions are often much lengthier, as the server may elicit a number of items of information about the user before finally deciding to permit or deny their Access-Request.

TACACS+ header

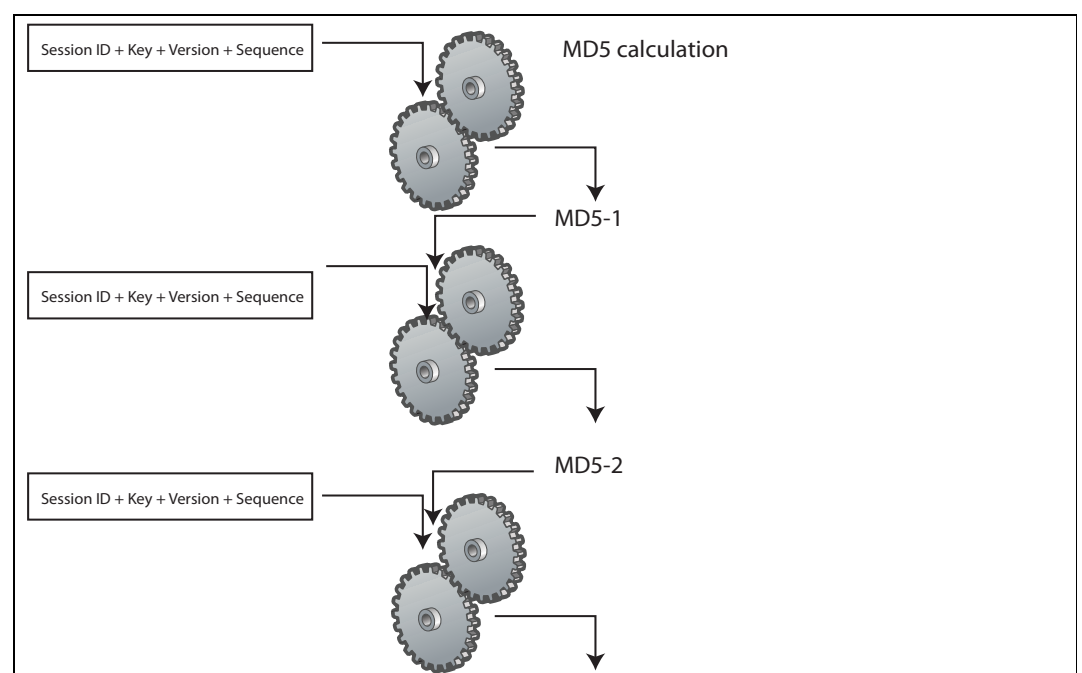
The TACACS+ header, that appears immediately after the TCP header, and precedes the packet body, has seven fields:

1. major version: 4 bits
2. minor version: 4 bits
3. type - 1 byte: indicates whether this packet is for authentication, authorization or accounting
4. seq_no - 1 byte: the sequence number within the current session
5. flags - 1 byte: currently the only flag is that indicating whether or not the body is encrypted
6. session_id - 4 bytes: ID number of the current session
7. length - 4 bytes: the total length of the TACACS+ packet body (not including the header)

Encryption

Whilst TACACS+ does support the exchange of unencrypted packets, the standard preference is to encrypt the packets. If encryption is used, then the full body of the TACACS+ packet is encrypted. The body of the packet refers to all packet contents beyond the TACACS+ header.

The encryption method is based on a pre-shared key and makes use of MD5 in an interesting way. A string is created by combining the session ID, the pre-shared key, the version number and the packet's sequence number within the current session. Then, an MD5 hash of this string is calculated. Then, another string is created by combining the original string with the resulting MD5 hash, and this new string is then hashed. Then, the original string is combined with the second MD5 hash output, and an MD5 hash is performed on this third string, and so on.



When enough MD5 hashes have been created that the combined length of the 16-byte strings MD5-1, MD5-2, MD5-3,... is greater than or equal to the length of the packet body, then the MD5 hashes are combined together in one long string. This string is then XORed with the packet body to create the encrypted packet contents.

Authentication

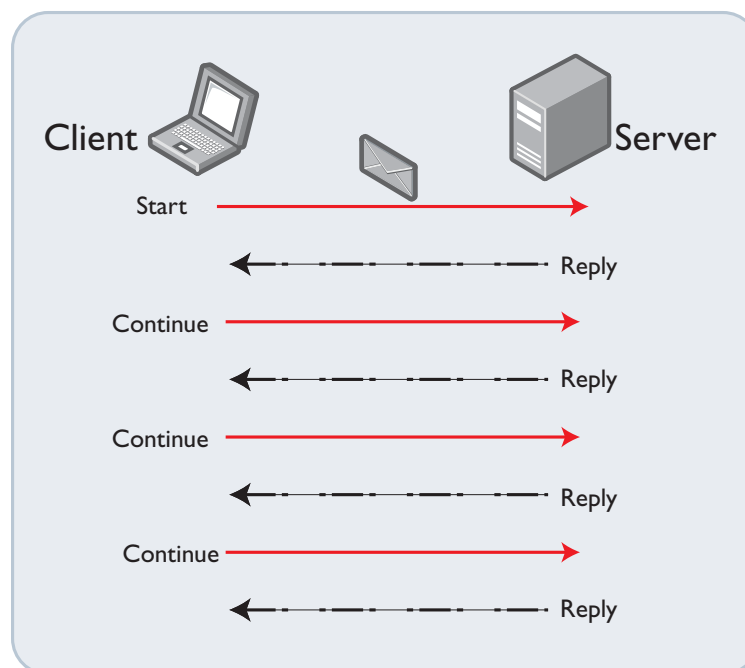
TACACS+ supports just a fixed set of authentication methods. It does not support EAP, and so cannot support arbitrary authentication methods. In particular, this means that TACACS+ cannot be used with 802.1x.

The fixed set of authentication methods supported by TACACS+ are: ASCII (plaintext), PAP, CHAP, ARAP, and MSCHAP.

Authentication session packet exchange

An authentication session always starts with a START packet from the client, to which the server will always reply with a REPLY packet. If the server requires more information from the client before deciding to allow or deny the authentication request, that further information will be requested in the REPLY packet. The client responds with a CONTINUE packet, to which the server will respond with another REPLY packet. This pattern continues until the server has all the required information and can finally reply with a Pass or Fail.

Every packet sent from the client will always elicit a REPLY from the server, as shown in the diagram below:



There are a couple of packets that can break this pattern:

- The client can send a **CONTINUE** message containing an abort message. In this case, the session is terminated immediately, with no need for the server to acknowledge the abort message.
- The server can send a **REPLY** packet containing a **RESTART** message. In this case, the session is begun again, with a new **START** packet from the client. The sequence number is rewound back to 1, but the session ID is not changed.

The RESTART is typically used when the server will not accept the authentication method specified in the client's first START packet. Usually, the RESTART packet will contain a list of the authentication methods that the server will accept.

Authentication actions

An authentication session can be used for three separate purposes:

1. A user Access-Request.
2. To change a user's password.
3. To request authentication credentials to send in an authentication request to another system.

The type of authentication session that is being initiated is always specified by the client in an 'action' field in the START packet.

The operation of these three different kinds of authentication session are subtly different, as we will see further below. But before looking at those details, let us consider some other aspects of TACACS+ authentication.

Privilege level

The START packet specifies the privilege level that the user expects to operate at. This has a value from 0-15, which, of course, mirrors the range of privilege levels available under CISCO IOS.

Service

Another field in the START packet is the service field. This indicates what activity on the client has elicited an authentication request. Among others, the service can specify a standard login, a PPP session establishment, an X.25 session establishment, a firewall proxy session, and a transition to a higher level of privilege. This last service type is referred to as the 'Enable' service, as it is commonly invoked by a user entering the IOS enable command. It could also be invoked by a user entering the UNIX su command.

User Access-Request sessions

The primary purpose of a user Access-Request session is for the client to send the username and password of an access requestor, so that the server can decide whether the user's credentials are valid. As described above, the type of access being requested is not necessarily a login to a device's management interface; it can also be the establishment of a network connection (PPP or X.25).

For 4 of the 5 possible authentication methods (PAP, CHAP, MS_CHAP, ARAP), the session **must** consist of a single START and a single REPLY packet. The START packet provides the username and password, and the REPLY indicates whether the server will or will not grant the Access-Request.

However, for ASCII login sessions, the session consists of a series of REPLY+CONTINUE exchanges between the client and server. For ASCII login sessions, the username is not necessarily provided in the initial START packet, and the password never is. The server will send REPLY packets with `getUser` and `getPass` to obtain the username and password.

The server can also use a **getData** request to obtain any arbitrary information it desires. Typically, a `getData` request is accompanied with a message text, with the intention that the client display this message (invariably a question) to the user, so the user can type a response that the client sends back to the server in a CONTINUE packet.

Usually, the final REPLY packet in a user Access-Request session will contain the server's verdict, delivered as either a Pass or a Fail. But the server can send the value **follow** in a reply. This indicates that the server cannot service the authentication request so the client should try another server. The Follow message is accompanied by a list of one or more servers, and can also indicate which protocol (for example, RADIUS, Kerberos, BSD R-command) and key the client should use with each of those servers.

Password change sessions

Password changing can only be performed for the ASCII and ARAP authentication methods; it is not supported for PAP, CHAP, or MS-CHAP. The session is brief—the client contacts the server with a START packet indicating that it wishes to change the password for a user.

In the case of the ARAP authentication method, the START packet also contains the old and new passwords. The server has all the information it needs right there in the START packet. It checks that the old password is correct, and the new password conforms to password requirements, etc. Then it sends a single REPLY to the client to inform it whether the password change request has been accepted or not. In the case of ASCII authentication, the old and new passwords are not provided in the START packet but are requested by the server in `getData` and `getPass` REPLY packets.

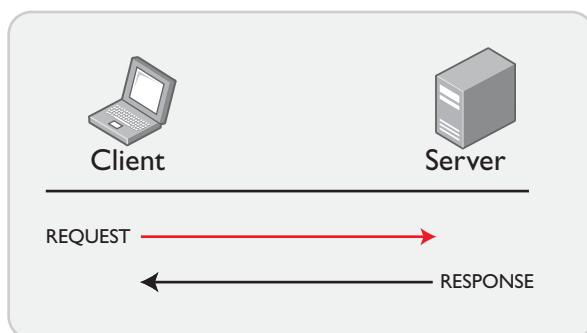
Authentication credential request sessions

These sessions are also referred to as Outgoing sessions. They occur when the client device is establishing a PPP connection with another remote device and needs to authenticate itself to that device. Rather than hold the authentication credentials locally, the client may store them on the TACACS+ server and request them from the server when they are required.

As the authentication credential request session can only be used for the case of PPP authentication, then the only authentication methods for which it is supported are PAP, CHAP, and MS-CHAP. The session must consist of only a single START packet and a single REPLY packet. The START packet specifies the username whose authentication response is required. The REPLY packet provides this response. In the case of PAP, the response is just the user's password. In the case of the CHAP methods, the response is the full MD5-hashed challenge response.

Authorization

An authorization session must always consist of a single pair of packets: a REQUEST packet from the client, and a RESPONSE packet from the server.



TACACS+ supports the authorization of all manner of actions. Among other things, the client can request the server to decide whether:

- A given user, logged into the client, may execute a given command (with a specified set of arguments).
- A given address from an IP address pool can be allocated to a given remote user connected to the client by a PPP or X.25 link.
- The network link (PPP, X.25) to a given remote user can be used for routing.
- A given route can be attached to the network link (PPP, X.25) to a given remote user.
- A given higher-layer protocol (IP, iPX, ATALK, FTP, Telnet, RLOGIN etc) can be used over the PPP link to a given remote user.
- The client can dial back to a given remote user that has established a PPP or X.25 connection to the client.

The RESPONSE packet from the server will inform the client whether the authorization request is granted or denied—expressed as Pass or Fail.

In the case of a request to authorize a user's command, a Pass response can also specify a change to the arguments for the command. The server might specify one or more arguments to add to the set listed in the REQUEST packet, or a set of arguments that must completely replace the set in the REQUEST packet.

Authorization to start a shell session

The attributes that have been defined for TACACS+ authorization packets do not include an attribute that explicitly states that the client device wishes to start up a shell session for a logged in user. The convention has arisen that such an authorization request is sent as a request to authorize null command.

The “I request authorization to start a shell session for user X” request is sent with the attributes “service=shell” and “cmd=”. Taken literally, it looks like a request to authorise a blank command, but the convention is that it is actually a request to start up a shell session for the logged in user whose username will also be present in the request packet.

The server will respond to such a request with a Pass or Fail verdict. If the server sends “Pass”, it will typically also include some parameters that will govern the nature of the shell session the user will be presented with. Typical attributes include privilege level, timeout, a connection access ACL number, whether the user may use the ESC key.

Accounting

As with authorization, an accounting session must always consist of a single pair of packets—a REQUEST packet from the client, and a RESPONSE packet from the server.

The REQUEST message will be one of three types:

1. Start
2. Stop
3. Update (referred to as a Watchdog message in TACACS+ jargon)

The Start message will usually contain a start time, and the Stop message will contain a stop time. Update messages include elapsed times and bytes/packets sent and received.

The REQUEST will also specify the user and the service that the accounting information relates to, as that information identifies the client connection being accounted.

AlliedWare Plus Implementation of TACACS+

The implementation of the TACACS+ client in AlliedWare Plus supports the following features:

- Authentication for user Access-Request sessions, using the ASCII authentication method.
- Authentication of enable password on a per-user basis, using the ASCII authentication method.
- Accounting to provide an audit trail of user exec sessions.
- Accounting of CLI commands executed within a user exec session, configurable on a per privilege level basis.
- Authorization requests to open a shell session for a logged in user. The attributes that AlliedWare Plus will process in the authorization response are:
 - **privilege level**—sets the user's privilege level in the range 1 to 15, with 15 being the highest.
 - **idletime**—sets how long the switch will wait, after the last user input into the shell session.
 - **timeout**—sets an absolute upper limit on the duration of the shell session, after which the session will be closed and the user will be forcibly logged out.

Note that Authorization:

- Cannot be performed independently of the authentication process. There are no authorization commands available.
- Must be configured on the same server as authentication.
- Is not applicable if enable password authentication has been configured.

With the AlliedWare Plus TACACS+ client implementation, all traffic that passes between the TACACS+ client and a TACACS+ server is encrypted using the TACACS+ key defined for that server. It is not possible to disable TACACS+ encryption. TACACS+ encrypts the entire payload of packets, which means that it encrypts the user's password between the client and the server.

AlliedWare Plus TACACS+ configuration example

The following configuration example describes how to configure the AlliedWare Plus TACACS+ client.

1. Configure the TACACS+ server IP address.

```
awplus(config)#tacacs-server host 172.10.10.1
```

Up to a maximum of four TACACS+ servers can be defined. If the first server cannot be contacted within the tacacs-server timeout period, the switch will attempt to contact the next configured server. This process will continue until one of the configured servers respond or until the list of servers has been exhausted.

2. Configure the global TACACS+ key for encrypting TACACS+ sessions.

```
awplus(config)#tacacs-server key tacacspass
```

The TACACS+ global key is used to encrypt sessions against all configured TACACS+ servers. The TACACS+ global key can be overridden by configuring a TACACS+ server host specific key.

3. Configure the default method list for TACACS+ login authentication.

```
awplus(config)#aaa authentication login default group tacacs+ local
```

The default login authentication method list is applied to all exec sessions without the need for additional configuration. Alternatively, named method lists can be configured and applied separately to console and VTY lines with the above command. Specifying the parameter **local** after **group tacacs+** means that if none of the configured TACACS+ servers can be reached, then the local user database will be used to authenticate users during login.

4. Configure the switch to use TACACS+ enable password authentication.

```
awplus(config)#aaa authentication enable default group tacacs+ local
```

AlliedWare Plus supports only a default method list for enable password authentication, which means it is applied globally to all users. A user is configured on a TACACS+ server with a maximum privilege level. When they enter the **enable <privilege level>** command they are prompted for an enable password which is authenticated against the TACACS+ server. If the password is correct and the specified privilege level is equal to or less than the user's maximum privilege level, then they are granted access to that level.

If the user attempts to access a privilege level that is higher than their maximum configured privilege level, then the authentication session will fail and they will remain at their current privilege level.

Specifying the parameter **local** after **group tacacs+** means that if none of the configured TACACS+ servers can be reached, then the local user database will be used to authenticate enable passwords.

5. Configure the default method list for TACACS+ login accounting.

```
awplus(config) #aaa accounting login default start-stop group tacacs+
```

The default login accounting method list will be applied to all exec sessions, without the need for additional configuration. Alternatively named method lists can be configured and applied separately to console and VTY lines with the above command.

By specifying the **start-stop** parameter that means that a Start packet will be sent when a user exec session begins, and a Stop packet will be sent when the user exec session closes. Alternatively **stop-only** can be specified, to only send an accounting Stop packet when the user exec sessions close.

6. Configure TACACS+ command accounting for privilege level 1, 7, and 15.

```
awplus(config) #aaa accounting commands 1 default stop-only group
tacacs+
```

```
awplus(config) #aaa accounting commands 7 default stop-only group
tacacs+
```

```
awplus(config) #aaa accounting commands 15 default stop-only group
tacacs+
```

AlliedWare Plus supports only a default method list for command accounting, which means it is applied globally to all user exec sessions. AlliedWare Plus command accounting is configurable per privilege level, meaning that only commands at the specified privilege level will be accounted to the TACACS+ server.

To configure AlliedWare Plus to account exec-mode (privilege level 1) commands, intermediate privileged exec-mode (privilege level 7) commands, and privileged exec-mode (privilege level 15) commands, each level must be specified separately.

Chapter 7 | Web-Authentication

Introduction

Web-authentication is a simple way to provide secure guest-user access to a network. It is also known as Captive Portal. It is used in a wide range of environments including Wi-Fi hot spots, hotels, universities, and business centres.

In basic terms, if the switch detects an unauthorized user web browsing, then irrespective of the IP configuration on their PC, they are re-directed to a Web-authentication login page. At this point, the user is required to enter a username and password before they can begin to web browse.

The main benefits of this solution come from not requiring additional customer knowledge, software or special configuration. Users are able to quickly and easily gain access to the network regardless of the type of device or operating system used.

This chapter:

- describes what Web-authentication is
- explains how Web-authentication is used
- examines Web-authentication operation
- and provides examples on how to manage traffic of unauthenticated supplicants.

List of terms

ARP intercept

When ARP intercept is enabled, the switch will process ARP packets coming in from supplicants. Instead of forwarding the ARPs, the switch will respond to them as though it possessed the IP address being requested in the ARP.

Guest VLAN

In a secure network, the default behaviour is to deny any access to supplicants that cannot be authenticated. However, it is often convenient to allow unauthenticated users to have limited access. A popular solution is to define a limited-access VLAN, called the Guest VLAN, and assign unauthenticated users into that VLAN.

Auth-fail VLAN

In some networks, it is useful to be able to differentiate between users who have not even attempted to authenticate themselves, and those who have tried and failed. This is achieved by defining an auth-fail VLAN in addition to the Guest VLAN. When a user has had sufficient unsuccessful authentication attempts, they are assigned to the Auth-fail VLAN.

What is Web-authentication?

Web-authentication is a convenient alternative to 802.1x authentication. It's commonly used to authenticate users in educational institutions, where regular users' workstations are not managed by the network administrator. Web-authentication enables the switch to detect an unauthenticated workstation web browsing into the network, then redirect the user's web browser to its own authentication web page.

Web-authentication works like this:

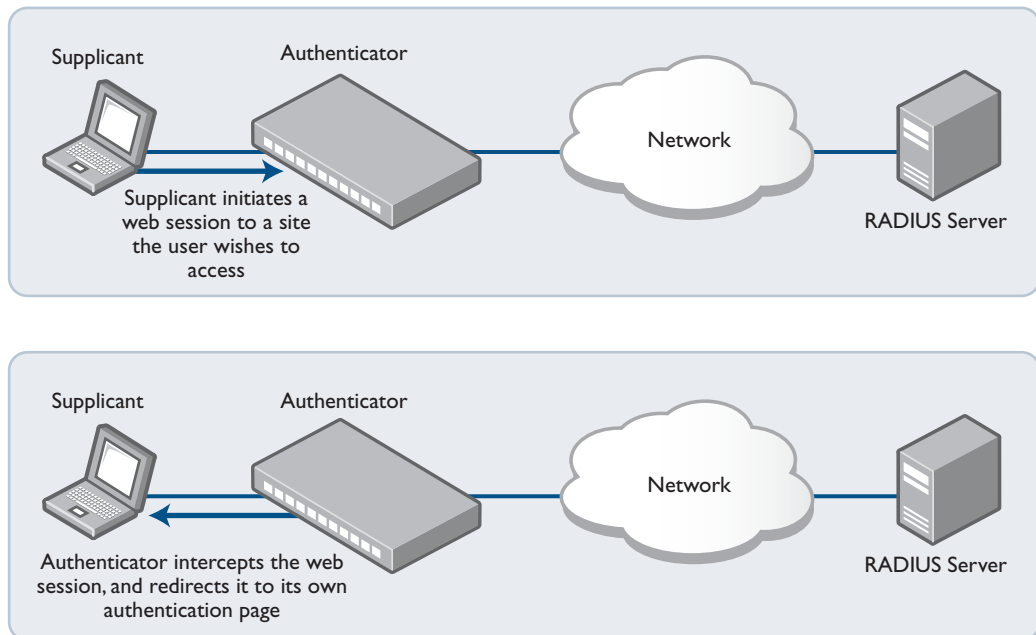
- The user enters their username and password into the web page, which the switch then sends to a RADIUS server for checking.
- If the RADIUS server accepts the user's credentials, the switch then allows their traffic into the network.

The Web-authenticating switch interacts with a RADIUS server in the same way as an 802.1x authenticator. The two methods can easily be used together in the same network, using the same RADIUS server.

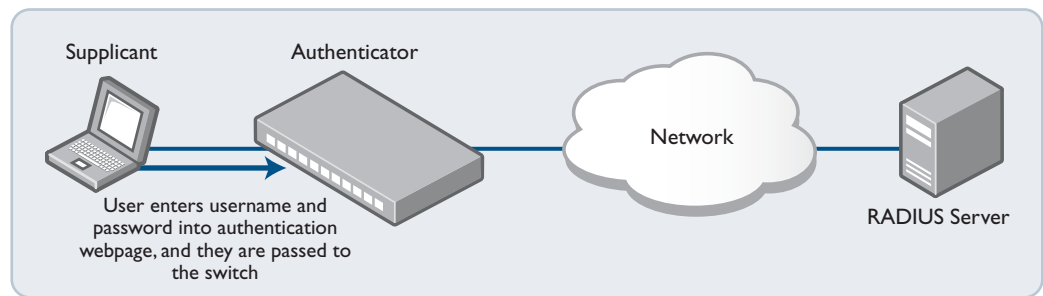
Web-authentication Basics

Conceptually, the operation of Web-authentication is quite simple:

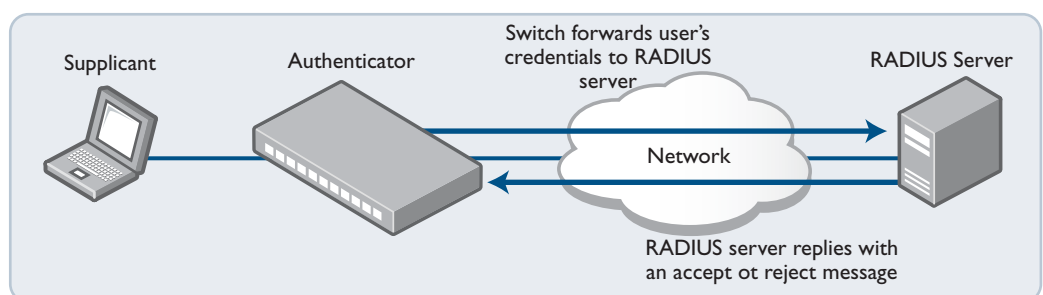
- I. The authenticating switch receives HTTP or HTTPS traffic from an unauthenticated supplicant. It intercepts the supplicant's web session and redirects it to its own internal web server.



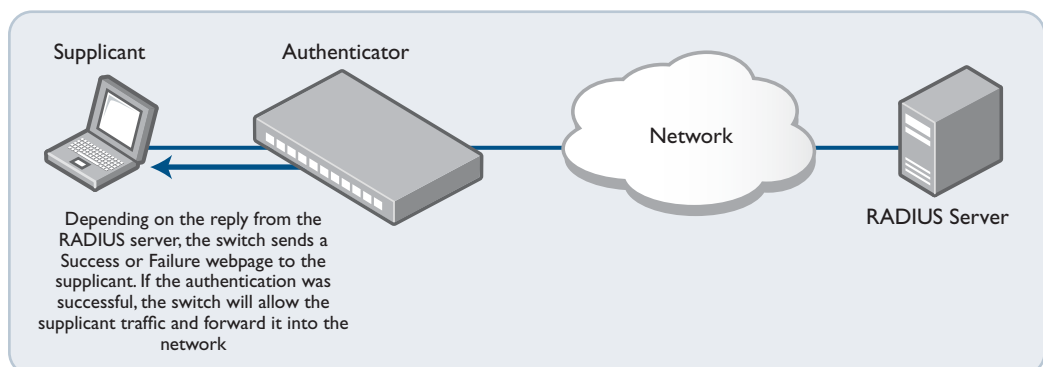
- The web server serves up an authentication page on which the user enters their username and password.



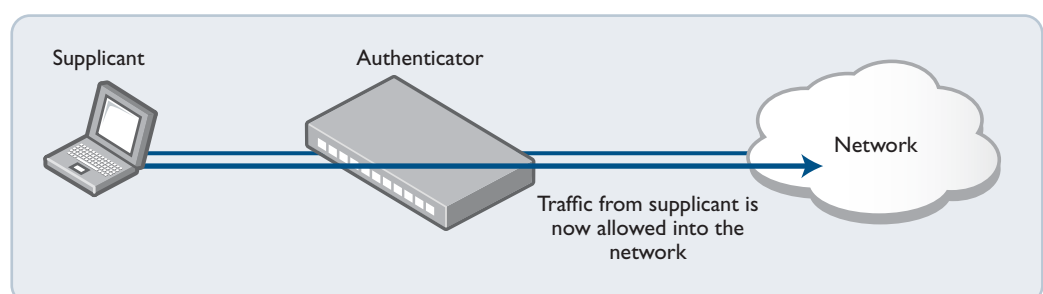
- The username and password are sent to a RADIUS server, which informs the authenticating switch whether or not the supplicant is authenticated.



- The user is then informed of the RADIUS server's verdict.



- If the supplicant has been successfully authenticated, the authenticating switch will give the supplicant workstation access to the network.



Configuring Web-authentication

Web-authentication can be configured on a switch in four simple steps:

1. Configure a RADIUS server.

```
awplus(config)# radius-server host <server-ip-address> key
<shared secret>
```

2. Instruct Web-authentication to use the configured RADIUS server.

```
awplus(config)# aaa authentication auth-web default group
radius
```

3. Define the IP address that the Web-authentication service will be accessed.

```
awplus(config)# auth-web-server ipaddress <ip-address>
```

4. Configure ports for Web-authentication.

```
awplus(config)# interface port1.0.1-1.0.20
awplus(config-if)# auth-web enable
```

Choosing the Web-authentication server address

When you come to configure Web-authentication, you need to answer some questions:

Questions What IP address should I specify as the Web-authentication server address? Is it OK to use just any IP address that is configured on one of the switch's VLANs, or is the choice more constrained than that?

Answer You must use the IP address that is configured on the VLAN that the **supplicant's packets will arrive** on.

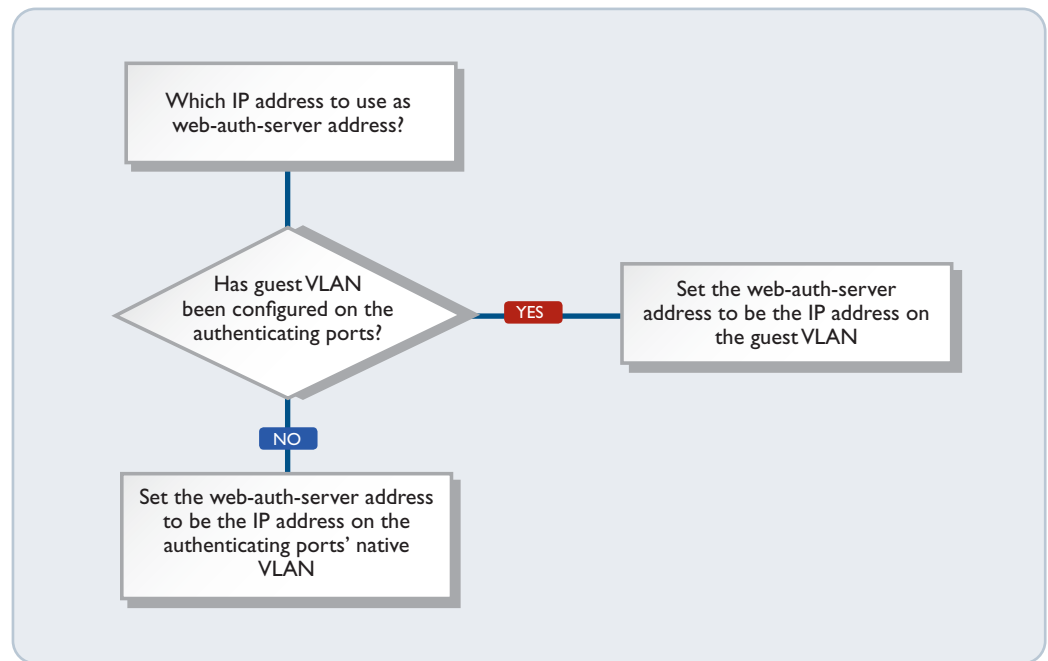
At first glance, that seems a simple answer. If the supplicant-connected ports are access ports in VLAN10, then you would expect that you configure the IP address on VLAN10 as the Web-authentication server address. In fact, that is the correct choice, **unless** a guest VLAN has been configured on the supplicant-connected ports.

The logic that the switch uses in deciding which VLAN to associate non-authenticated supplicants' packets with is:

- If guest VLAN has been configured on the port where the packet arrives, then associate the packet with the guest VLAN.
- Otherwise associate the packet with the port's native VLAN.

To reiterate, if you configure the supplicant-connected ports with guest VLAN, then use the IP address on the guest VLAN as the IP address of the Web-authentication server. Otherwise use the IP address on the supplicant-connected ports' native VLAN.

The diagram below illustrates how to decide which IP address to use as the Web-auth-server address:



Configuration Example 1: Using a guest VLAN

```

VLAN database
  VLAN 20 name guest
  VLAN 10 name edge
  VLAN 30 name core

radius-server host 192.168.30.129 key verysecret
aaa authentication auth-Web default group RADIUS
auth-Web-server ipaddress 192.168.20.1

int vlan10
  ip address 192.168.10.1/24
int vlan20
  ip address 192.168.20.1/24
int vlan30
  ip address 192.168.30.1/24

int port1.0.1-1.0.20
  switchport access vlan 10
  auth-Web enable
  auth guest-vlan 20

int port1.0.21-1.0.22
  switchport access vlan 30
  
```

Configuration Example 2: Not using a guest VLAN

```
VLAN database
  VLAN 10 name edge
  VLAN 30 name core

radius-server host 192.168.30.129 key verysecret
aaa authentication auth-web default group radius
auth-web-server ipaddress 192.168.10.1

int vlan10
  ip address 192.168.10.1/24
int vlan30
  ip address 192.168.30.1/24

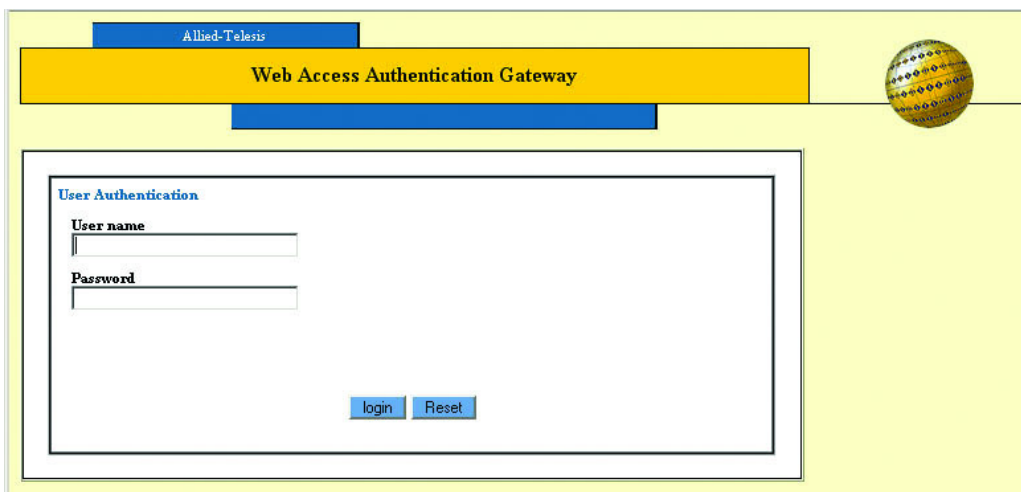
int port1.0.1-1.0.20
  switchport access vlan 10
  auth-Web enable

int port1.0.21-1.0.22
  switchport access vlan 30
```

Starting a Web-authentication Session

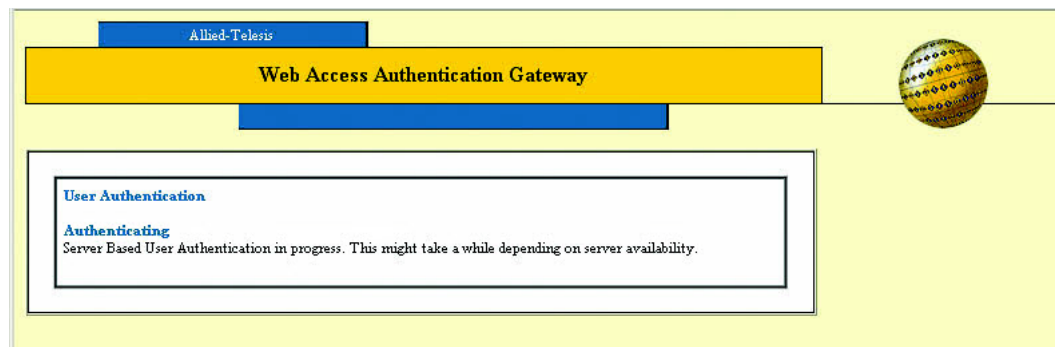
Let us look at what the user actually sees in a Web-authentication session:

1. The user starts their Web browser, and browses to a page they wish to view. Shortly thereafter, the address in the browser's address bar automatically changes to the address of the authenticating switch's authentication page.
2. In the switch's authentication page, the user enters their **User name** and **Password**, and clicks **login**.

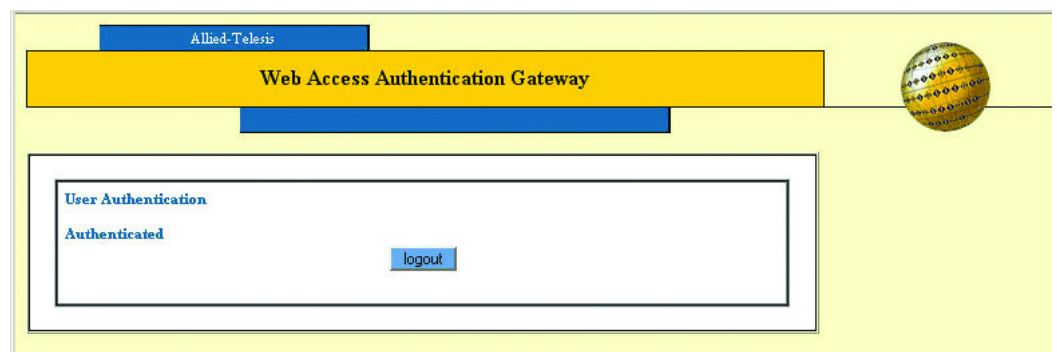


The screenshot shows a web browser window displaying the 'Web Access Authentication Gateway' page. The page has a yellow background. At the top, there is a blue bar with the text 'Allied-Telesis' and a yellow bar with the text 'Web Access Authentication Gateway'. To the right of the yellow bar is a small globe icon. Below these is a white box titled 'User Authentication' containing two input fields: 'User name' and 'Password'. At the bottom of the white box are two buttons: 'login' and 'Reset'.

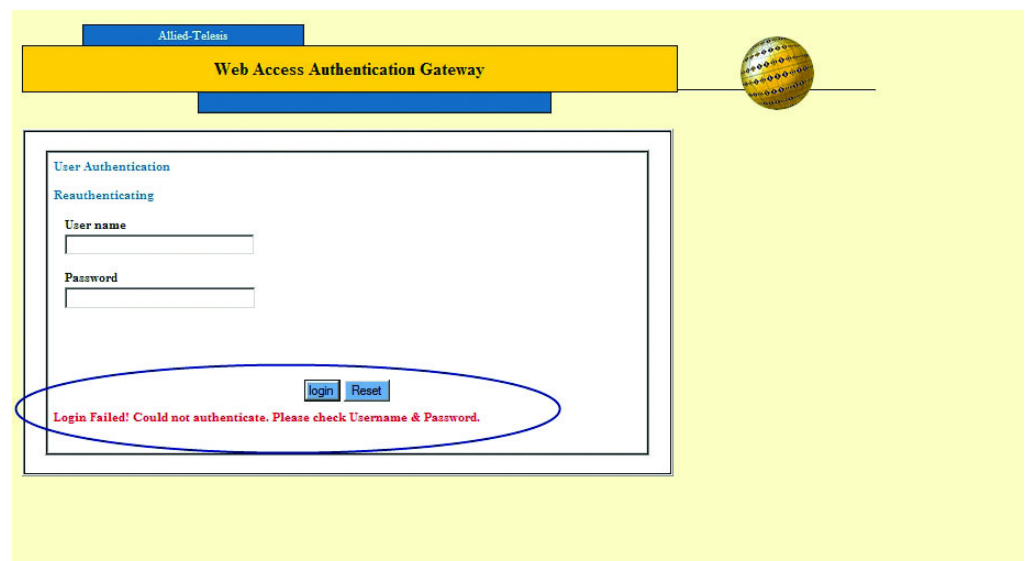
3. The switch displays a page that informs them that authentication is in progress.



4. Once authentication is complete, the authentication result is displayed.



- If the user enters a username/password combination that is **not** accepted by the RADIUS server, the switch presents an invitation to check the username and password. If the user enters incorrect usernames/passwords several times, the authentication fails. The number of times a user can try to login is configurable but it is set to 3 by default.



Understanding the Web-authentication Features

While the authentication process is essentially quite simple as it has been described so far, there are a number of implementation details that it glosses over.

To understand how to use Web-authentication effectively, we'll take a closer look at:

- Protocol support features
- Secure authentication (SSL)
- Ping-poll monitoring of supplicant presence
- Managing traffic of unauthenticated supplicants

Support for protocols underlying Web-authentication

Web-authentication does not use a dedicated protocol like 802.1x, with a standards-defined set of messages for authentication conversation. Instead, the switch overlays the Web-authentication process on top of the web browser communication process. The browser communication process was not designed for authentication and is itself reliant on IP addressing, ARP, and DNS.

The authentication needs to occur in a seamless manner for all users, irrespective of their IP and DNS setting, and before they have full access to the network.

To make this possible, the switch needs to provide facilities that enable the user's PC to access the authentication web page.

There are a following features of Web-authentication work together to achieve this:

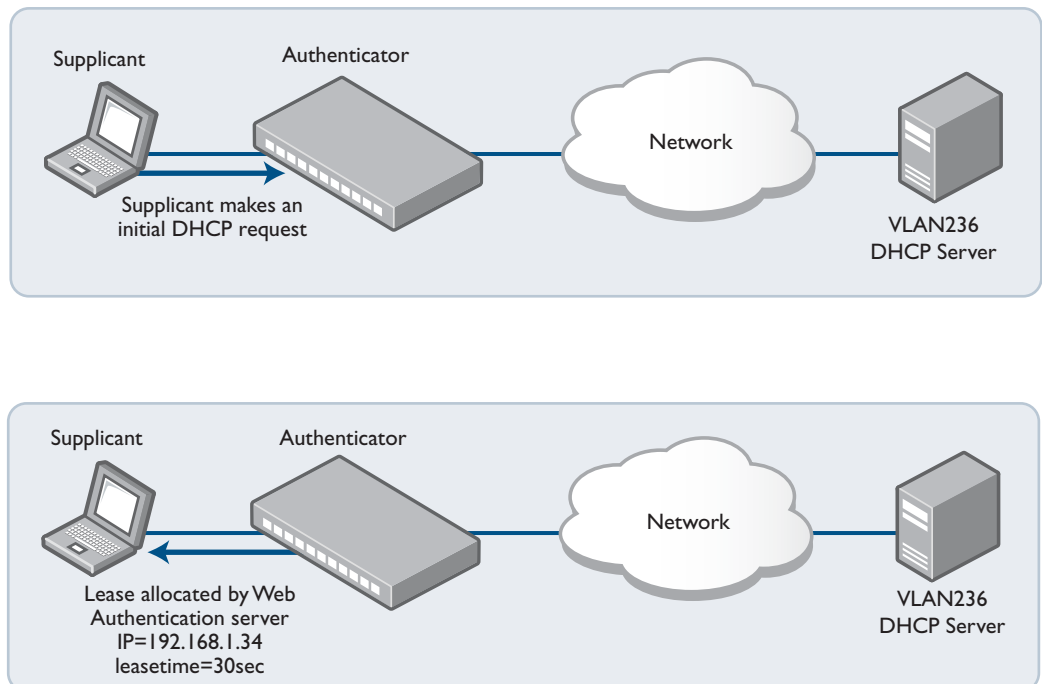
- DHCP server for Web-authentication
- Interception of clients' ARPs
- Proxy DNS response

DHCP server for Web-authentication—for supplicants using DHCP

To initiate a web browsing session, the supplicant needs an IP address. If the supplicant has been configured to obtain its IP address by DHCP, then the authenticating switch needs to ensure that the supplicant will be served an IP address.

The simplest way to achieve this, is to have the Web-authentication process itself act as a DHCP server. This avoids forwarding the supplicant's DHCP request to any other DHCP server. Therefore, there is a DHCP server built in to Web-authentication.

This DHCP server is dedicated to serving IP addresses to be used by Web-authentication clients.



This DHCP service is configured by the command:

```
awplus(config)# auth-web-server dhcp ip address <ip-address/
prefix-length>
```

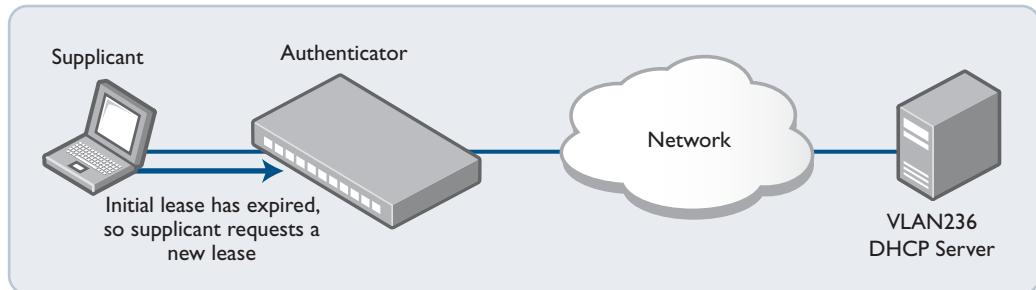
The IP address specified in this command is the IP address of the Web-authentication service. If the Web-authentication service's IP address has not already been configured by the command **auth-web-server ip address <ip-address>**, then this command configures the service's address.

If the Web-authentication service's IP address had already been configured by the command **auth-web-server ip address <ip-address>**, then the IP address in the **auth-web-server dhcp** command must be the same as that already configured. By default, this DHCP server serves leases of 30 second's duration. The lease duration can be changed by the command **auth-web-server dhcp lease <20-60>**. The short lease is deliberate. It facilitates the transition to a new VLAN/subnet after authentication. The supplicant is unaware that the switch transitions it to another VLAN, with another DHCP server, after authentication succeeds.

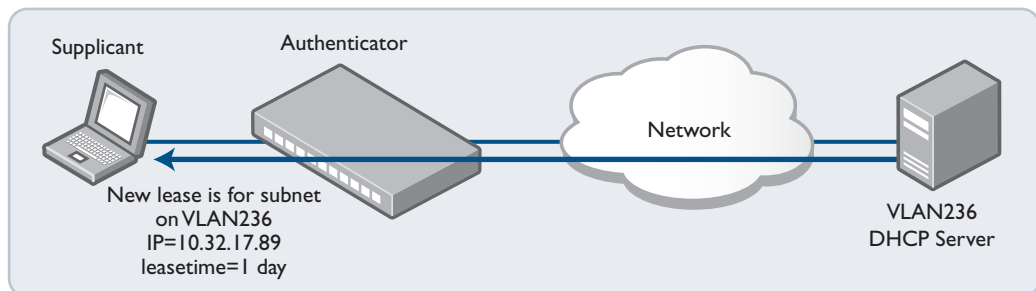
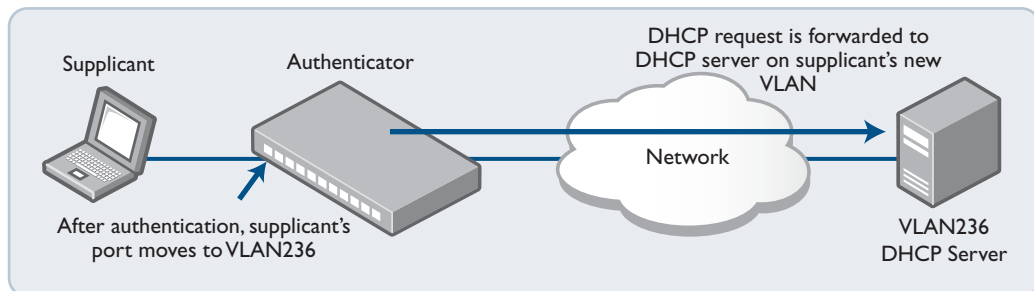
Similarly, there is no mechanism by which the switch signals to the supplicant to say "I have just assigned you to VLAN 236, you now need to obtain a DHCP lease from the DHCP server on that VLAN".

How can we force the supplicant to request a new DHCP lease after the completion of the authentication process? There is no mechanism by which the supplicant's web browser signals down to the DHCP client process to say "I've just completed an authentication session, you need to request a new DHCP lease".

The solution is to ensure that the lease allocated by the dedicated Web-authentication DHCP service is of a very **short duration**. This way the lease will expire within a short time from the completion of the authentication process, resulting in the supplicant requesting a new lease.



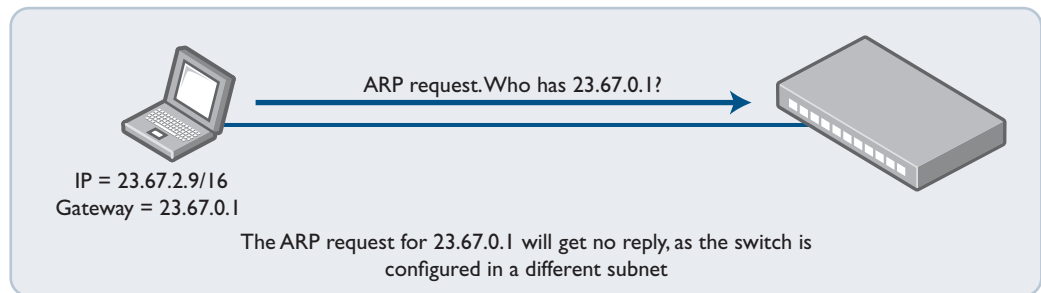
- This new request will now be serviced by the DHCP server on the supplicant's new VLAN.



Interception of clients' ARPs—for supplicants with static addresses

If the supplicant has been configured with a static IP address, then it is more than likely that the supplicant's IP configuration bears no relation to the Web-authentication server address. A computer's IP communications will always be preceded by sending out ARP requests for host addresses in its local subnet, or for its gateway address.

If the IP address and gateway address have been statically configured on the computer, and the subnet used in this static configuration is different to that on the authenticating switch, then the ARP requests will receive no reply, and the PC will not begin IP communication.

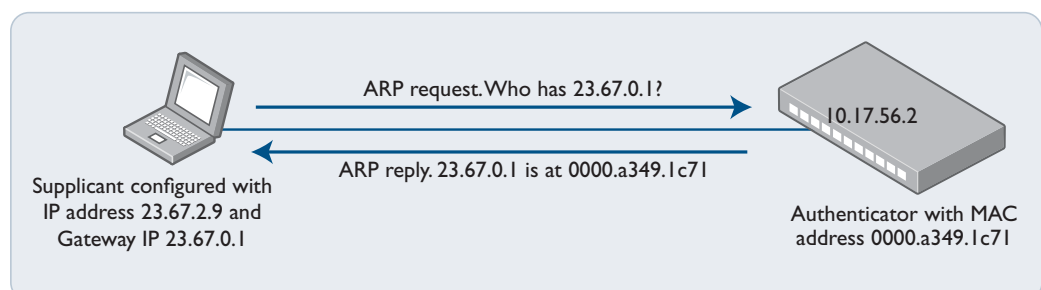


To deal with any arbitrary IP configuration on the supplicants, Web-authentication needs a method for replying to arbitrary ARP requests. This is the **ARP interception** feature.

ARP interception can operate in three modes:

```
awplus(config)# auth-web-server mode {intercept|none|promiscuous}
```

1. **Intercept** – will respond to ARP requests for any IP address that is in the same subnet as the switch's own IP address. Will provide its own MAC address in the ARP reply, irrespective of what IP address (within its own subnet) was being requested.
2. **None** – will only respond to ARP requests for its own IP address.
3. **Promiscuous** – will respond to **any** ARP request. Will provide its own MAC address in the ARP reply, irrespective of what IP address was being requested. When this mode is configured, the Web-authentication server can interoperate with **any** static IP configuration on a supplicant.

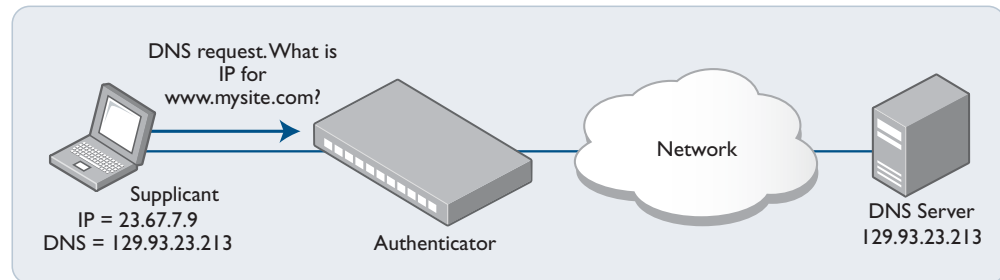


In promiscuous mode, the switch will send its own MAC address in response to an ARP request for ANY address, no matter whether the requested address bears any relation to the switch's own IP address on the interface where the ARP is received.

Proxy DNS response

Typically, an HTTP session from a web browser is preceded by a DNS request for the IP address of the web site the user wishes to browse to. If the DNS request receives no reply, the web browser will never progress on to connecting an HTTP session.

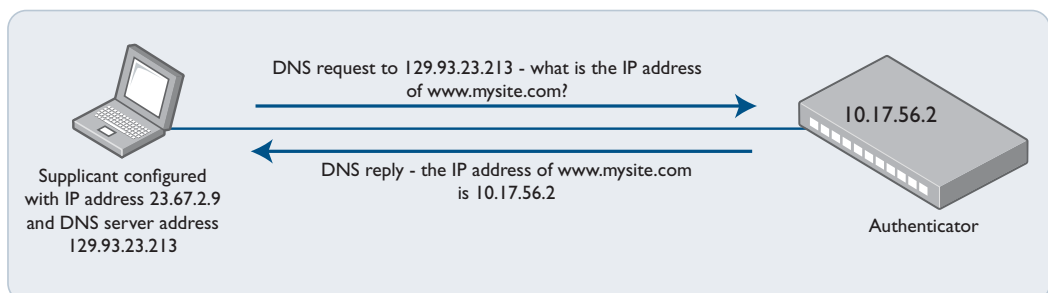
The Web-authentication server needs a mechanism to reply to DNS requests, so that the Web-authentication session can begin.



A web browser must request a DNS Server for the IP address corresponding to a URL. But the switch will not forward the request if the supplicant is not yet authenticated

The three modes listed also control the operation of the proxy DNS replies.

1. **Intercept** – responds to DNS requests whose source IP address is within the same subnet as the IP address on the switch. The IP address provided as the resolution of the DNS lookup is the switch's own IP address, so that the subsequent HTTP traffic will be directed to the switch.
2. **None** – the **default**. Does not respond to DNS requests.
3. **Promiscuous** – responds to DNS requests from any source IP address. The IP address provided as the resolution of the DNS lookup is the switch's own IP address, so that the subsequent HTTP traffic will be directed to the switch.



In promiscuous mode, the switch will reply to ANY DNS request from an authenticated supplicant, regardless of whether the destination IP address of the DNS server bears any relation to the switch's own IP address. The DNS reply from the switch will always specify its own IP address as the URL that was being requested.

Secure Authentication

The Web-authentication service can be configured to use a secure HTTPS connection. This ensures that the username and password are sent from the supplicant to the switch in encrypted form and cannot be snooped by anyone eavesdropping on the session.

Secure Web-authentication requires two steps:

1. Create an SSL certificate for the switch using the command:

```
awplus(config)# crypto pki enroll local
```

2. Configure the Web-authentication service to use HTTPS instead of HTTP:

```
awplus(config)# auth-web-server ssl
```

Once the Web-authentication service has been put into secure mode, the service will always use HTTPS for Web-authentication sessions, irrespective of whether or not the user directs their browser to an HTTPS session.

- For example, if the user directs their browser to <http://mysite.com>, the Web-authentication service will automatically redirect them to <https://<Web-authentication server address>>.

By default, the Web-authentication service uses TCP port 443 for HTTPS sessions, but it can be configured to use a different port, using the command:

```
awplus(config)# auth-web-server sslport <1-65535>
```

Copying a certificate onto the switch

Instead of using the self-created certificate, it is also possible to create a certificate elsewhere, and copy that certificate onto the switch.

The command to copy the certificate onto the switch is:

```
awplus # copy tftp://<tftp server address>/<certificate file name> Web-auth-https-file
```

The certificate could be created, for example, by using openssl, which is available for multiple different operating systems.

Note that the file that is copied onto the switch must:

- be in PEM format
- contain both the certificate and the corresponding private key.

If you use a program that creates separate files for these, you need to combine them into one file. The order within the file does not matter – the key could be first or the certificate could be first.

The **openssl** commands used to create a key pair and a certificate are:

- Create the private key:

```
openssl genrsa -out privkey.pem 2048
```

- Create a self-signed certificate for this key:

```
openssl req -new -x509 -key privkey.pem -out cacert.pem
```

- This will result in you being prompted for a number of parameters, like **organisation name**, **email address**, etc. Enter whatever values you want for these parameters.

`Privkey.pem` and `cacert.pem` are text files. Use a text editor to combine the content of these files together into a single file. The order within the file does not matter – the key could be first, or the certificate could be first.

Once the file has been copied onto the switch to be the Web-authentication HTTPS file, the output of the command **show auth-Web-server** will show:

```
awplus#show auth-Web-server
Web-authentication server
Server status: enabled
<SNIP>
Certification: user    <-----
<SNIP>
```

If you wish to remove the certificate that you have copied onto the switch, and go back to using the switch's self-generated certificate, use the command:

```
awplus# erase web-auth-https-file
```

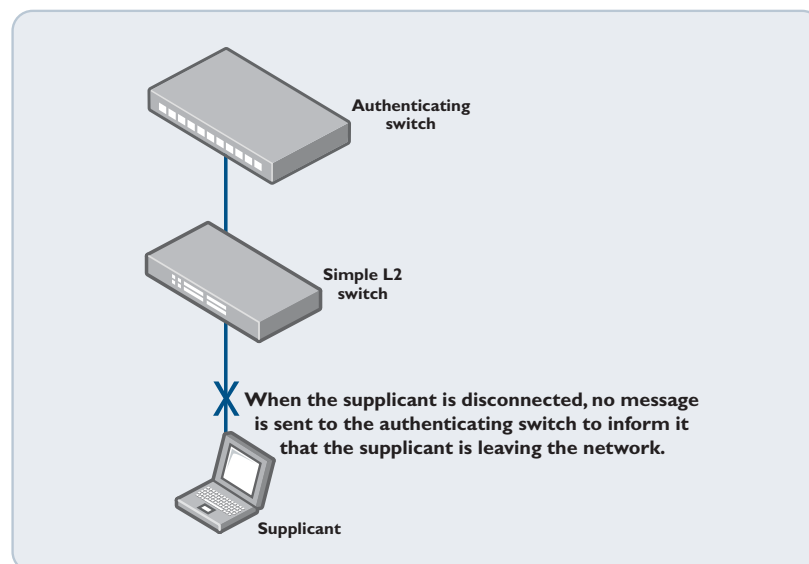
Ping-poll Monitoring of Supplicant Presence

A supplicant's authenticated session on the network must eventually come to an end. How does the authenticator decide that a supplicant's session has ended, and so remove it from the list of authenticated supplicants?

Sometimes it is obvious that the supplicant's session has ended, if the:

- supplicant unplugs from a port.
- user clicks the **logout** button they were provided with on the "Authentication Success" page as described in ["Starting a Web-authentication Session" on page 98](#).

Other times it is not obvious. Consider the case that a supplicant is not directly connected to the authenticating switch, but is connected to another switch that lies between itself and the authenticating switch, and the user simply disconnects their workstation.



If the network administrator wishes to ensure that the authenticating switch detects the supplicant's disconnection quickly, rather than waiting for the next expiration of the re-authentication period, then they can use **ping polling** to monitor the supplicants. This feature is enabled by the command:

```
awplus(config)# auth-web-server ping-poll enable
```

Once ping polling has been enabled, the Web-authentication service will automatically ping-poll every Web-auth supplicant once they have been authenticated.

By default the ping-poll has a:

- polling **interval** of 30 seconds
- **timeout** of 1 second (i.e. the switch waits 1 second for the ping response before deciding the ping has failed)
- **failcount** of 5 (i.e. if a given supplicant fails to respond to five pings in a row, its authenticated session is terminated)

These default values can be altered by using the commands:

```
awplus(config)# auth-web-server ping-poll interval <1-65535>
awplus(config)# auth-web-server ping-poll timeout <1-30>
awplus(config)# auth-web-server ping-poll failcount <1-100>
```

The currently configured values of these parameters can be seen by using the command:

```
show auth-web-server
awplus#show auth-web-ser
Web-authentication server
<SNIP>
HTTP Redirect: enabled
  Session keep: enabled
  PingPolling: enabled
PingInterval: 30
  Timeout: 1
  FailCount: 5
  ReauthTimerRefresh: disabled
```

Checking the IP addresses of the supplicants

To verify the IP addresses of the supplicants that the switch is ping-polling, use the command `show auth-web supplicant brief`.

```
awplus#show auth-web supplicant brief
Interface port1.0.19
authenticationMethod: Web
<SNIP>
Interface   VID Mode  MAC Address      Status   IP Address      Username
=====
port1.0.19  150  W  001c.7e95.d6bb  Authenticated  192.168.150.9  andrewr
```

Ping-poll and promiscuous mode

Note that if you are using promiscuous mode to enable workstations with arbitrary IP addresses to be authenticated, there is no guarantee that the ping poll process will successfully send pings to those supplicants.

If the supplicant's IP address is not in the same subnet as the IP address the switch has on the VLAN that the supplicant is allocated into, then the ping poll process will not be able to send pings to that supplicant. We recommend that you **do not** use the ping poll feature if you are using promiscuous mode, it risks the possibility of certain supplicants' authentication being terminated on a regular basis.

```
2010 Jun  4 18:46:09 user.notice awplus 802.1X[1044]: port1.0.19:
Supplicant andrewr logoff, Mac 001c.7e95.d6bb
2010 Jun  4 18:46:09 user.notice awplus 802.1X[1044]: port1.0.19:
Supplicant andrewr unauthorized, Mac 001c.7e95.d6bb
```

Managing Traffic of Unauthenticated Supplicants

The forwarding, blocking, and VLAN classification of traffic that arrives at the switch from unauthenticated supplicants is not entirely straightforward, and is subject to configuration. For the most part, the switch does not make a distinction between supplicants who have not yet attempted authentication, and those that have tried to authenticate, but failed.

The case in which there is a distinction drawn between those two classes of unauthenticated supplicant is when the auth-fail VLAN has been configured. Even when the auth-fail VLAN is not configured, the treatment of unauthenticated supplicants' traffic will differ, depending on whether or not the guest VLAN is configured.

We will take four different configuration combinations in turn, and look at the treatment of unauthenticated supplicants' traffic in each case of the cases where the port where the traffic arrives is configured with:

1. No Guest VLAN or Auth-fail VLAN
2. Guest VLAN, but no Auth-fail VLAN
3. Auth-fail VLAN, but no Guest VLAN
4. Auth-fail VLAN, and Guest VLAN

No Guest VLAN or Auth-fail VLAN

Traffic Type	How Traffic is Processed
HTTP packets to Web-auth server address	Sent to CPU <ul style="list-style-type: none"> Processed by Web-authentication
DHCP	Sent to CPU <ul style="list-style-type: none"> Processed by Web-auth DHCP server (if configured) or <ul style="list-style-type: none"> Processed by switch's standard DHCP server or <ul style="list-style-type: none"> Relayed to another DHCP server
DNS	Sent to CPU <ul style="list-style-type: none"> Intercepted by Web-authentication (if intercept configured) or <ul style="list-style-type: none"> Forwarded to another DNS server or <ul style="list-style-type: none"> Dropped
ARP	Sent to CPU. Supplicant's ARP is learnt. <ul style="list-style-type: none"> Normal reply to ARP requests for switch's own IP address Intercepted by Web-authentication (if intercept configured) or <ul style="list-style-type: none"> Dropped

Traffic Type	How Traffic is Processed
Other packets	<p>If auth-Web forwarding configured</p> <pre>auth-web forward {arp dhcp dns ...}</pre> <p>packets matching criteria will be forwarded with native VLAN (not routed to other VLANs).</p> <p>All other packets dropped.</p>

Guest VLAN but no Auth-fail VLAN

Traffic Type	How Traffic is Processed
HTTP packets to Web-auth server address	<p>Sent to CPU</p> <ul style="list-style-type: none"> Processed by Web-authentication
DHCP	<p>Sent to CPU</p> <ul style="list-style-type: none"> Processed by Web-auth DHCP server (if configured) <p>or</p> <ul style="list-style-type: none"> Processed by switch's standard DHCP server <p>or</p> <ul style="list-style-type: none"> Relayed to another DHCP server if guest VLAN 'routing' enabled
ARP	<p>Sent to CPU. Supplicant's ARP is learnt.</p> <ul style="list-style-type: none"> Normal reply to ARP requests for guest VLAN IP address Intercepted by Web-authentication (if intercept configured) <p>or</p> <ul style="list-style-type: none"> Dropped
Other packets, not destined to switch's own IP address	<ul style="list-style-type: none"> L2 switched within guest VLAN L3 switched to other VLANs if guest VLAN 'routing' enabled <p>Note: If guest VLAN 'routing' option is configured, take care to use ACLs to constrain what packets can be routed where. For example, you would probably configure ACLs to allow the traffic to be L3 forwarded to a DHCP server, a DNS server and possibly a NAC remediation server, or network domain controller.</p>
Other packets destined for switch's own IP address	<ul style="list-style-type: none"> Dropped

Auth-fail VLAN, but no Guest VLAN

In this case traffic from supplicants who are deemed to have failed authentication is treated differently to traffic from supplicants who are deemed to not yet have fully tried authentication.

The definition of a supplicant having “failed authentication” is when the supplicant’s number of failed authentication attempts has reached the value configured by the command:

```
awplus(config) # auth-web max-auth-fail <0-10>
```

By default, the value is 3.

Traffic from as-yet unauthenticated supplicants

This traffic is associated with the Native VLAN on the port on which the traffic arrives. It is treated in exactly the same way as described in ["No Guest VLAN or Auth-fail VLAN" on page 109](#).

Traffic from supplicants which has failed authentication

- This traffic is associated with the auth-fail VLAN configured on the ingress port
- The traffic is L2 switched within the auth-fail VLAN

Auth-fail VLAN, and Guest VLAN

Traffic from as-yet unauthenticated supplicants

This traffic is associated with the guest VLAN on the port on which the traffic arrives. It is treated in exactly the same way as described in ["Guest VLAN but no Auth-fail VLAN" on page 110](#).

Traffic from supplicants which has failed authentication

This traffic is associated with the auth-fail VLAN configured on the ingress port. It is treated in exactly the same way as described for the case of auth-fail VLAN and no guest VLAN, above.

Monitoring the Operation of Web-authentication

There is no specific debugging available for Web-authentication. The conversation between Web-authentication and a RADIUS server can be output by the command:

```
awplus# debug RADIUS all
```

An audit trail of Web-authentication events is kept in the system log. Successful and unsuccessful login attempts; and logoffs all generate entries in the system log.

```
2010 Jun  4 18:50:54 daemon.notice awplus radiusd[1712]: Login OK:
[andrewr] (from client 127.0.0.1 port 5019 cli 00-1c-7e-95-d6-bb)

2010 Jun  4 18:50:56 user.notice awplus 802.1X[1044]: port1.0.19: Web-
authentication successful for andrewr, IP 10.32.4.78, Mac
001c.7e95.d6bb

2010 Jun  4 18:52:31 daemon.notice awplus radiusd[1712]: Login
incorrect: [tester] (from client 127.0.0.1 port 5019 cli 00-1c-7e-95-
d6-bb)

2010 Jun  4 18:52:33 user.notice awplus 802.1X[1044]: port1.0.19: Web-
authentication failed for tester, IP 192.168.101.6, Mac 001c.7e95.d6bb

2010 Jun 15 18:35:00 user.notice awplus 802.1X[1046]: port1.0.19:
Supplicant and rew'r unauthorized, Mac 001c.7e95.d6bb
```

A list of all currently authenticated auth-Web supplicants can be seen from the commands:

```
awplus# show auth-web supplicant
awplus# show auth-web supplicant brief
```

Chapter 8 | 802.1x

Introduction

The IEEE Standard 802.1x provides a method of restricting access to networks based on authentication information. 802.1x provides port-based network access control for devices connected to the Ethernet. This allows a network controller to restrict external devices from gaining access to the network behind an 802.1x controlled port. External devices that wish to access services via a port under 802.1x control must firstly authenticate themselves and gain authorization before the 802.1x controlled port will forward any packets originating from, or destined for, the external device.

Port access control is achieved by making devices attached to a controlled port authenticate themselves via communication with an authentication server before these devices are allowed to access the network behind the controlled port.

This chapter provides an overview of the operation of 802.1x, and then describes the different authentication methods used by 802.1x. The configuration of 802.1x on AlliedWare Plus™ is described, including aspects like dynamic VLAN allocation and Guest VLAN. There are also notes on configuration of supplicants and servers. Advice on troubleshooting 802.1x on AlliedWare Plus is also provided.

List of terms

EAP

Extensible Authentication Protocol. A protocol that can carry out an arbitrary authentication process between two end-point devices, without an intermediate relay device needing to understand the content.

TLS

Tunnel Layer Security. An encrypted authentication exchange.

Supplicant

A device using 802.1x to request access to a network.

The role of 802.1x in networks

Networks have two important requirements:

1. **Security:** Authentication and Authorisation
2. **Flexibility:** The ability for users to roam

Networks need a device authentication method that is highly secure, but not tied to a port's physical location. Network resources presented to a given user need to be determined from their authentication credentials.

802.1x user authentication satisfies these requirements. It is relatively uncomplicated and has very little impact on network performance. It is a protocol that is medium-independent — being equally as effective on wireless connections (802.11i) and wired connections. 802.1x user authentication is rapidly becoming an expected component on networks.

What is 802.1x?

802.1x is an IEEE standard providing a mechanism for authenticating devices attached to a LAN port or wireless device. Devices wishing to access services behind a port must authenticate themselves before any Ethernet packets are allowed to pass through. The protocol is referred to as 802.1x because it was initially defined in the IEEE standard 802.1x, published in 2001 and revised in 2004 and again as the current 802.1x 2010 standard.

802.1x System Components

There are three main components to a system using 802.1x port authentication control:

1. Authenticator

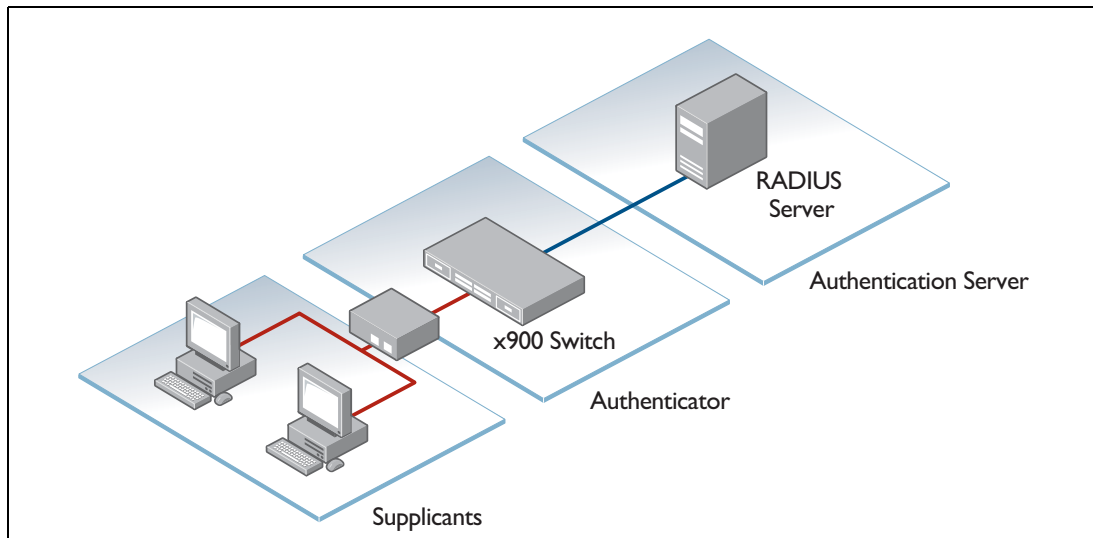
The device that wishes to enforce authentication before allowing access to services that are accessible behind it. An example of this is a switch that has 802.1x port authentication control enabled.

2. Supplicant

The client that wishes to access services offered by the authenticator's system. An example of this is a Windows XP Professional PC with an 802.1x client.

3. Authentication server

The device that uses the authentication credentials supplied by the supplicant, to determine if the authenticator should grant access to its services. The Allied Telesis implementation of 802.1x supports the use of a RADIUS authentication server using Extensible Authentication Protocol (EAP) in conjunction with RADIUS.

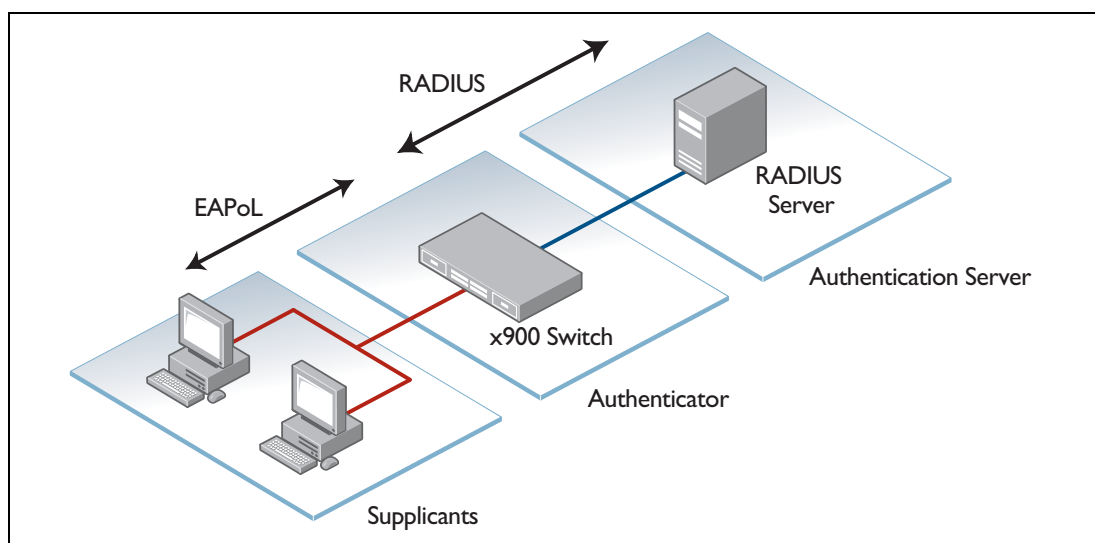


802.1x component protocols

There are two protocols involved in the authentication conversation:

1. EAPoL exchanged between the supplicant and authenticator.
 - EAPoL—Extensible Authentication Protocol over LAN—is the protocol defined in IEEE802.1x.
2. RADIUS exchanged between the authenticator and authentication server.
 - RADIUS has received specific extensions to interoperate with EAPoL.

The diagram below illustrates where EAPoL and RADIUS protocols are used in the authentication conversation:



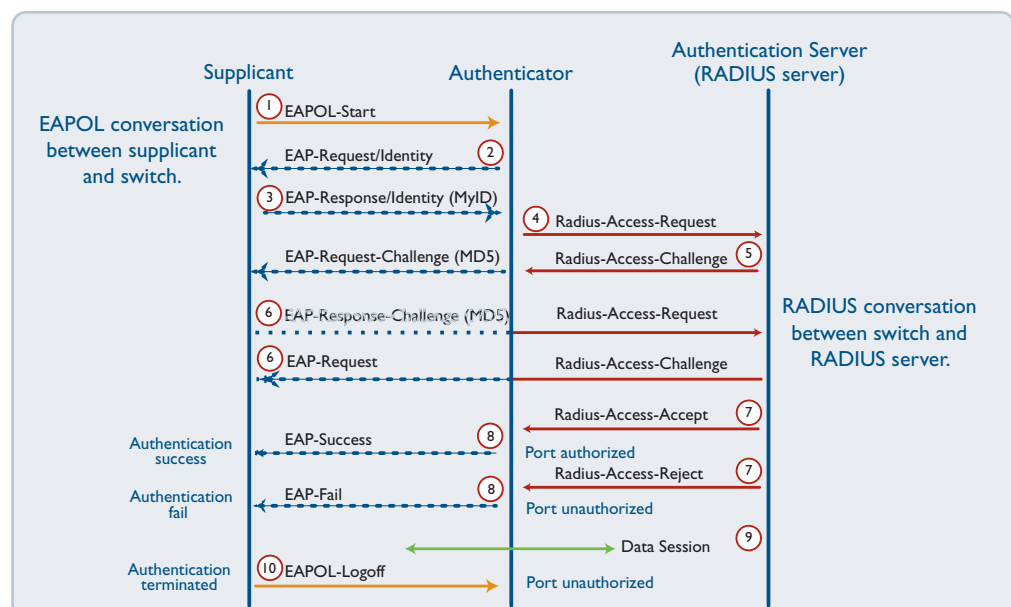
Basic steps in an 802.1x conversation

Step	Action
1	The supplicant informs the authenticator that it wants to initiate the conversation.
2	The authenticator requests the supplicant's credentials.
3	The supplicant sends username/password or X.509 certificate.
4	The authenticator wraps the supplicant's reply into a RADIUS packet and sends it to the RADIUS server.
5	The RADIUS server chooses an authentication method, and sends an appropriate request to the supplicant as a 'challenge'.
6	The RADIUS server and supplicant exchange some messages, ferried by the authenticator.
7	The RADIUS server eventually decides if the supplicant is allowed access and the RADIUS server sends an Access-Accept or Access-Reject message to the Authenticator.
8	The authenticator sends an EAPoL-Success or EAPoL-Fail to the supplicant.
9	The supplicant has a session using the network (if accepted).
10	When the session is over, the supplicant sends a log-off message.

Example message sequence

The diagram below illustrates an exchange using the EAP-MD5 authentication method, which is the simplest authentication method supported by 802.1x.

The EAPoL logoff message, of course, is not sent immediately after the other messages in the diagram, but is sent later on, at the end of the supplicant's data session, when it wishes to disconnect from the network.



Some important concepts

Allied Telesis access switches play the role of authenticator. The authenticator actually plays very little part in the authentication conversation. It simply ferries messages back and forth between the supplicant and the RADIUS server, but the actual authentication conversation takes place between the supplicant and the authentication server.

The authenticator:

- Wraps EAP packets from the supplicant inside the RADIUS packets and sends them on to the RADIUS server.
- Extracts EAP packets from the RADIUS server's replies, and sends them to the supplicant.
- Has very little knowledge of the content of these packets.

The authentication decision is performed by the RADIUS server. It informs the authenticator of its verdict in an Access-Accept or Access-Reject packet and the authenticator must follow the instructions and enforce the ruling (allow or deny access) of the RADIUS server

There are many different modes of EAP: EAP-MD5, EAP-TLS, EAP-TTLS, PEAP, etc.

Different EAP modes mean that there is different content within the EAP packets and different numbers of EAP packets exchanged in conversation. The authenticator has little awareness of different EAP modes.

Understanding EAP

Fundamentally, 802.1x is the EAPoL protocol and understanding EAP is central to understanding 802.1x. EAP existed before 802.1x, and was adapted into EAPoL by the IEEE802.1x standard. EAP has evolved significantly as new modes have been developed. Let us look at the evolution of the protocol.

Evolution of the protocol

The protocols and authentication methods associated with 802.1x have become a bit of an acronym soup. It is not always easy to get a clear view on where all the elements like MS-CHAP, PEAP, TLS, TTLS, EAP, etc. fit into the picture, and how they relate to each other.

The best way to understand where all the elements fit in is to look at the evolution of the protocol. As with most things, 802.1x started off relatively simple, and has become more complex as improvements have been added.

In the beginning there was the PPP Extensible Authentication Protocol

The Extensible Authentication Protocol (EAP) that underlies 802.1x was originally defined in quite a different context—namely as part of the Point-to-Point Protocol (PPP) suite. Traditionally, the PPP protocol had two authentication methods—PAP and CHAP. During the PPP link negotiation process, the PPP peers would negotiate with each other whether they would use PAP, CHAP, or no authentication. Then, when the link negotiation was complete, if they had agreed to perform authentication, the authentication exchange (just 2 or 3 packets) would occur. The nature of the exchange would differ slightly between PAP and CHAP.

However, as other authentication methods, in particular One-time Password (OTP) and Generic Token Card (GTC) came along, it became evident that PPP needed a better mechanism for incorporating new authentication methods.

EAP was originally defined in RFC2284 and updated by RFC 5247. The idea with EAP is that during link negotiation, the peers simply negotiate to perform authentication by EAP. Then during the authentication phase, they have a new negotiation, using the EAP protocol, as to which particular authentication method they will use. Then they perform the authentication according to the agreed method.

It should be kept in mind that this original definition of EAP just defined a few packets that were exchanged at a point in the negotiation of a PPP connection. These are the humble beginnings from which the whole current machinery of 802.1x grew.

EAP protocol details

Given that EAP underlies 802.1x, it is worth looking at some details of the protocol.

The protocol defines four message types. The Type field is one octet and identifies the structure of an EAP Request or Response packet. The first 3 Types are considered special case Types. The remaining Types define authentication exchanges. The Nak Type is valid only for Response packets, it MUST NOT be sent in a Request. The Nak Type MUST only be sent in response to a Request which uses an authentication Type code. All EAP implementations MUST support Types 1-4.

Message Type	Description
1	Request
2	Response
3	Success
4	Failure

Request, Success, and Failure packets are sent from the Authenticator to the peer being authenticated (in the language of RFC2284, this is referred to as 'the peer'). Response packets are sent from the peer to the Authenticator.

The data within Requests or Responses can be one of a number of types:

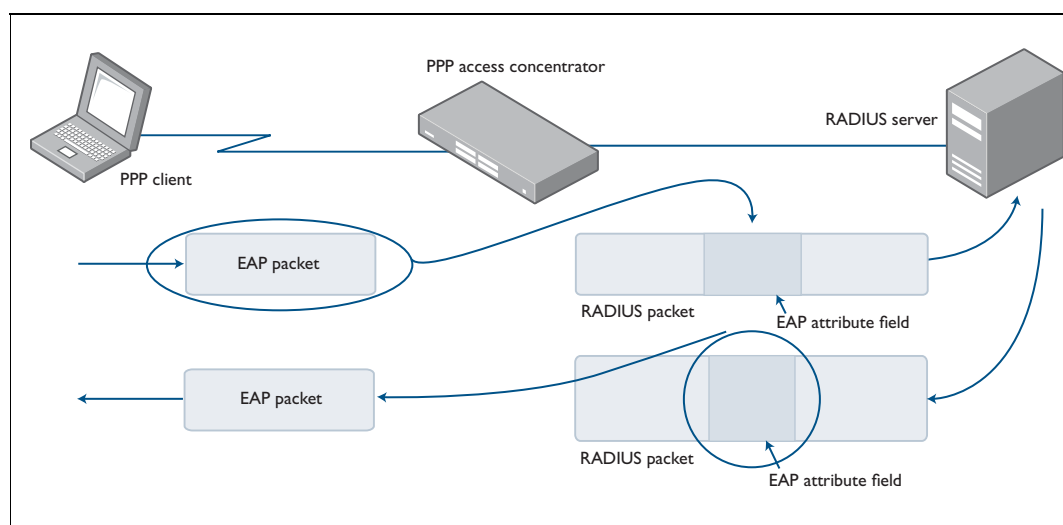
Data Type	Description
1	Identity
2	Notification
3	Nak (can only be in a Response packet)
All other values indicate that the packet is part of a particular authentication method	
4	MD5-Challenge
5	One-Time Password (OTP) (RFC 1938)6
6	Generic Token Card

Communication with the authentication server

One of the prime purposes for introducing EAP into PPP was to enable new authentication methods to be introduced without having to upgrade the device at the Authenticator end of the PPP connection. If the Authenticator operates by simply passing EAP authentication packets through to a 'back end' authentication server, then the Authenticator does not need to understand the details of the authentication methods that are being used within EAP.

In this way, the peer device and the authentication server are both free to agree on an authentication method they both understand, and they don't need to worry whether the Authenticator in between understands the chosen authentication method.

As new authentication methods come along, servers and clients can be upgraded to use these methods, irrespective of the capabilities of the network infrastructure over which they are communicating. By far the most popular authentication server type used in conjunction with EAP is RADIUS. In RFC2869, a new RADIUS attribute type (Type 79) was introduced specifically to pass EAP packets between the Authenticator and the RADIUS server.



Simply, the authenticator takes each EAP packet that it receives from the peer, puts the whole packet (minus the Layer 2 header) into an attribute-79 field of a RADIUS packet, and sends it through to the RADIUS server. Similarly, the RADIUS server will send the reply as an attribute-79 field of a RADIUS packet. The authenticator extracts the EAP packet from the RADIUS packet and sends it to the peer.

In this way, the authentication conversation is carried out between the peer and the RADIUS server, with the authenticator just acting as a packet relay.

The 802.1x standard defines EAPoL

At the heart of the IEEE 802.1x standard is the definition of a protocol called Extensible Authentication Protocol over LAN (EAPoL). This standard has taken the simple EAP protocol defined in RFC2284, extracted it from its context within PPP, and used it as the basis of a new stand-alone protocol running over Ethernet.

Because EAPoL does not operate as a stage within a PPP connection negotiation, it required some new message types to be added, so that it could operate stand-alone.

In fact, there are five message types defined in EAPoL:

Message Type	Description
0	EAP Packet
1	EAPoL Start
2	EAPoL Logoff
3	EAPoL Key
4	EAPoL Encapsulated ASF Alert

The EAP packet (Type 0) simply encapsulates an RFC2284 EAP packet. The majority of packets transmitted in an EAPoL exchange will be Type 0 packets, as the purpose of EAPoL is primarily to carry EAP packets across Ethernet.

The EAPoL Start packet (Type 1) had to be added as part of enabling the protocol to stand alone. In the PPP context, the EAP negotiation did not require a 'start' packet. Within PPP, as soon as the initial link negotiation has completed, the authentication phase starts straight away; there is no need for any particular 'lets start authentication' packet.

The EAPoL loggoff packet (Type 2) was also required to enable the protocol to stand alone. When a PPP connection terminates. The peers exchange PPP link-level packets to agree that the session is ended. It is not appropriate for the PPP authentication protocol (like EAP) to be involved in the connection-termination packet exchange. However, in the case of EAPoL, when a client device wishes to inform the Authenticator that it wishes to terminate its authenticated session, then EAPoL is the protocol it has available for signalling this termination. The EAPoL logoff message type was introduced to enable signalling of session termination.

The Type 3 and Type 4 EAPoL message types are for more specialised purposes. The Type 3 packet enables the exchange of encryption keys if the EAPoL session is to be encrypted.

The Type 4 packet is used to send network-management notifications (like SNMP traps) through a port, even if the port has not authenticated the client attached to it.

Some new terminology

In RFC2284, the term used for the device requesting to be authenticated was 'peer'. This is a term that arises naturally out of the PPP, as PPP standards frequently refer to the two ends of a PPP connection as peers. However, an 802.1x authentication conversation really is not a peer-to-peer interaction at all.

The devices exchanging the EAP packets are no longer peer routers, but a workstation (or other end-user device) and a LAN switch. The conversation itself is not particularly of a peer-to-peer nature. The client device is asking access permission, and the authentication server is sitting in judgement over it.

The 802.1x standard has done away with the term 'peer'. Instead, it refers to the client device as a **supplicant**, which is far more descriptive of its role in the conversation.

For the rest of this discussion, we will use the term supplicant.

Levels of Increasing Security in 802.1x

In essence, EAP (and so, of course, EAPoL) is a mechanism for the supplicant and authentication server to negotiate and then perform authentication.

Having this protocol in common usage has provided an opportunity for authentication methods of increasing security to be defined and deployed. Let us look at some of the authentication methods that have been used within 802.1x.

EAP-MD5

The MD5-challenge authentication method, which was defined along with the original EAP definition in RFC2284, was the first method to be used widely with 802.1x.

In this method, the supplicant has to send its username in clear text, and the password is transmitted as an MD5 hash.

This sending of the username in clear text gives potential crackers a head-start in breaking into the network. Given a stolen username (easily sniffed), the password can then be cracked by means of a dictionary attack (trying a long list of possible passwords until one works).

Moreover, MD5-challenge authentication provides no mechanism for the supplicant to verify that it is communicating with a genuine authentication server. Hence, the industry has looked for more secure authentication methods.

EAP-TLS

TLS stands for Tunnel Layer Security. The evolution of TLS constitutes a separate strand of the story of the elements that have come together in 802.1x.

TLS was defined in RFC2246. It grew out of the SSL protocol, which is still a very popular method for secure HTTP transactions. In the SSL protocol, the web server sends an X.509 certificate to the web client to prove its validity. Then the web transaction can be encrypted, based on the server's public/private keys.

TLS extended this by defining a requirement that the two ends of the communication both send each other X.509 certificates to establish their identities.

This provides an altogether higher level of security than the MD5 challenge. Instead of a password that is vulnerable to a dictionary attack, the supplicant uses a digital certificate to establish its identity. The certificate is encrypted using the server's public key and so can only be decrypted by the server that holds the corresponding private key.

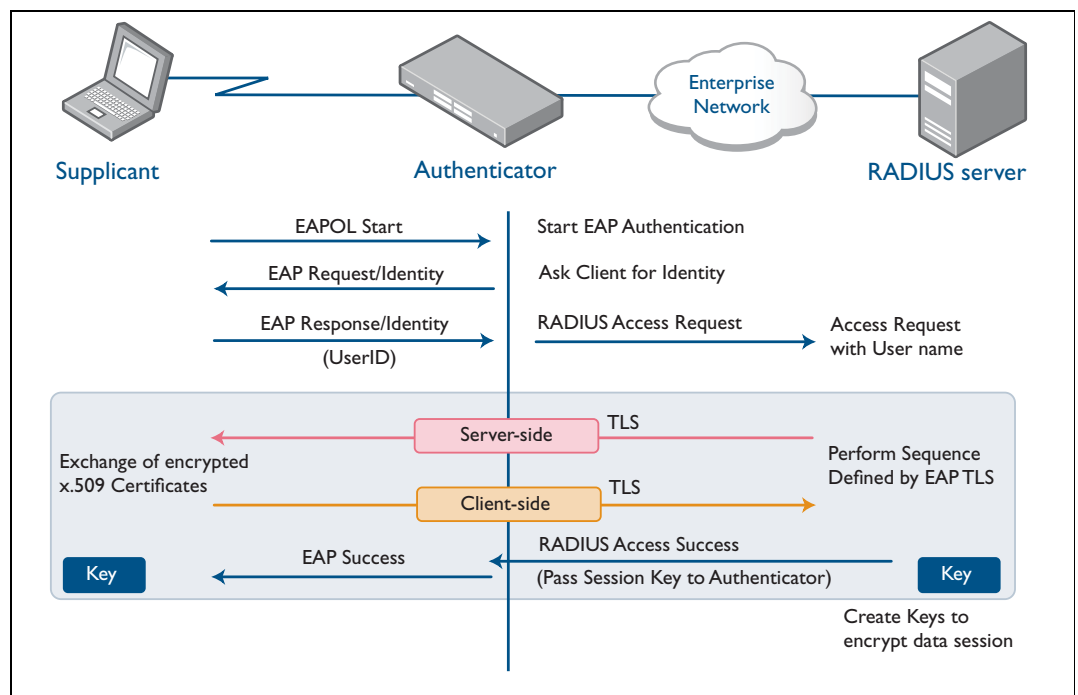
The use of TLS as an authentication method within EAP was defined in RFC2716. This RFC allocates the value 13 as the ID number used within EAP to identify the TLS authentication method.

The key negotiation process gives TLS an extra advantage, like SSL: the negotiated keys can be used to encrypt the actual data communication that the supplicant engages in after authentication. This is particularly useful when 802.1x is being used in a wireless

environment, where it is important to encrypt all communication to/from the supplicant device, to avoid eavesdropping.

Although the encryption keys are negotiated with the authentication server, it would be highly inefficient if the authentication server was required to encrypt/decrypt all subsequent data to/from the supplicant. Instead, the authentication server communicates these keys to the Authenticator. As the Authenticator is the device through which all of the supplicant's data accesses the network, it makes sense for the Authenticator to be the device that is performing the encryption/decryption.

The sequence of packet exchange in the EAP-TLS authentication exchange is as below.



EAP-TLS provides a good level of security for the authentication process, but it has the drawback that it requires all the supplicants to have X.509 certificates. This requirement can add a significant overhead to the administration of the network. Not only do certificates need to be created, and distributed to all the supplicants, there is an ongoing maintenance task of revoking certificates of users that leave an organisation. Also, it is not at all well suited to a guest-user or temporary user deployment, where the process of creating and distributing the certificate can take too long if the user only requires access for a brief period.

The industry needed to look for an authentication method that would provide the security of the TLS method, but the convenience of the password-based methods.

Two such methods have been developed in recent years: EAP-TTLS and PEAP.

EAP-TTLS

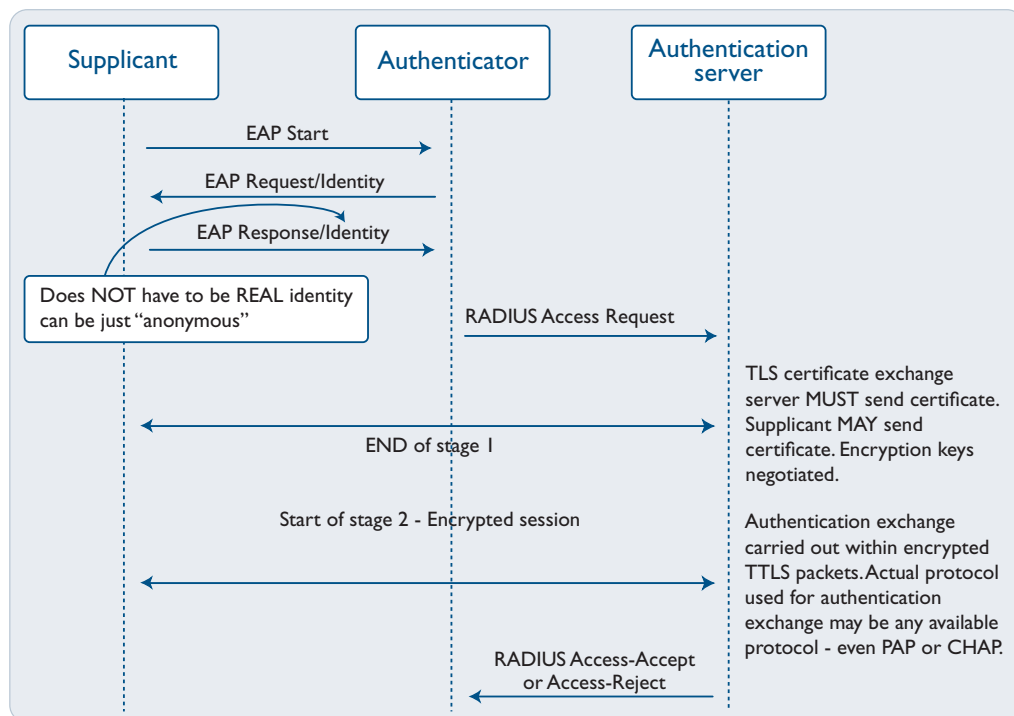
TTLS stands for Tunnelled Tunnel Layer Security.

Defined in RFC5281, it specifies a 2-stage authentication process:

1. The first stage is actually a TLS exchange. However, in the TLS phase of TLS only the server **must** provide its X.509 certificate. It is optional whether the supplicant provides its certificate.

In this phase, the supplicant validates the identity of the server, and the server **might** validate the identity of the supplicant. But, most importantly, a set of encryption keys are negotiated.

2. In the second stage, the encryption keys negotiated in phase (1) are used to encrypt the conversation in which the supplicant is authenticated. The authentication method used in this second phase is flexible. The authentication method might be PAP or CHAP or MS-CHAP, or even EAP-TLS. It is left entirely up to the supplicant and authentication server to negotiate what method they will use.



EAP-TTLS enables the supplicant to use a relatively simple password-based authentication method, but the encrypted nature of the authentication conversation makes the process far more secure.

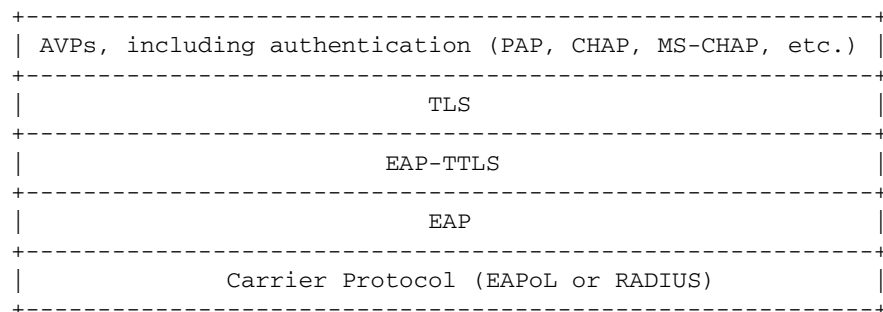
In fact, the data exchange carried out in the second phase of the EAP-TTLS exchange can be more than just authentication. The protocol allows all manner of data to be exchanged in this phase. This has opened the door for Network Access Control, in which the server interrogates the supplicant about a variety of aspects of the state of its software before it grants the supplicant access to the network. This NAC interrogation is performed by exchanging data within the second phase of the TTLS exchange.

Protocol layering

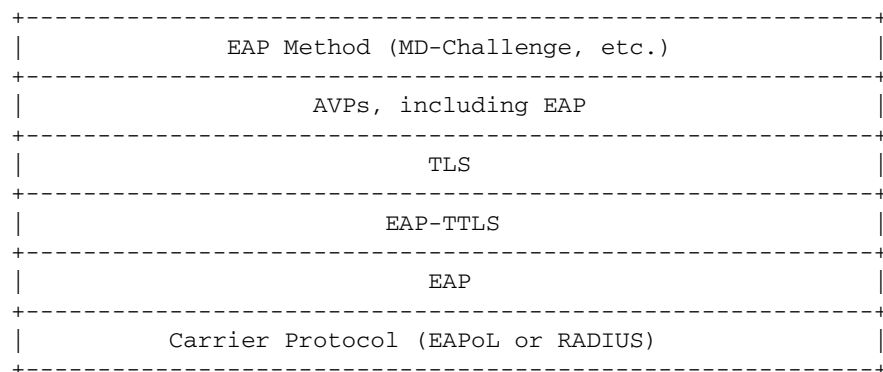
RFC5281 allocates the value 21 as the ID number used within EAP to identify the TTLS authentication method. It is evident, from the discussion above, that EAP-TTLS involves a fair amount of layering of protocols within protocols. In these protocol layer situations, a diagram can be a big help in visualising where all the various layers sit.

At the bottom of the stack is either EAPoL or RADIUS. One of the beauties of 802.1x is that the Authenticator can simply strip the Ethernet header and thin EAPoL header off the supplicant's packets, take all the rest of the packet contents, and embed these into an EAP-attribute field of a RADIUS packet.

The next layer up from the EAPoL or RADIUS carrier is EAP. EAP-TTLS is then encapsulated with EAP. The EAP-TTLS packets encapsulate TLS. TLS attribute-value pairs (AVPs) are then used to carry user authentication or other data.



When the user authentication protocol is itself EAP, the layering is as follows:



A note on the EAP identity exchange

Despite all the elaborate layering and data exchange that occurs within EAP-TTLS, the EAP conversation still must open with the server sending an identity request, and the supplicant responding with its identity. This transmission of a clear-text identity is a potential Achilles heel within this whole process. For this reason, RFC5281 specifically warns that the supplicant should not send its real identity in that initial identity exchange. The RFC recommends sending an identity like anonymous.

When sniffing an 802.1x exchange, you might often see a supplicant giving its identity as 'anonymous', and wonder how that can possibly be valid. The fact is, of course, that it is

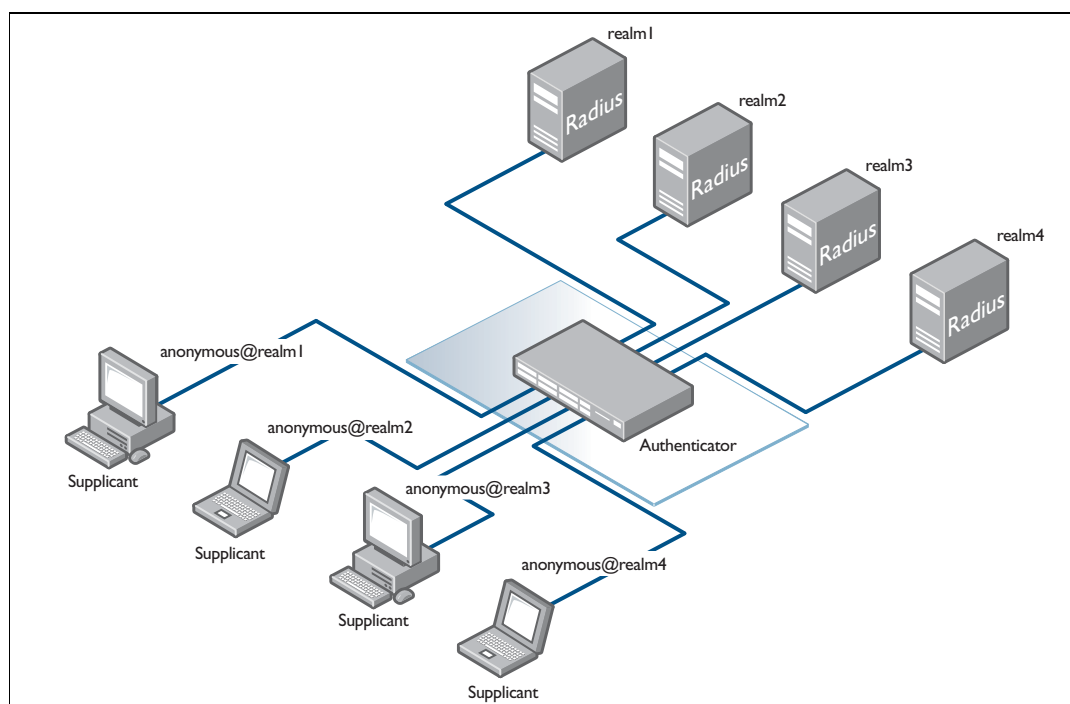
not meant to be a valid identity, it is just a placeholder. The supplicant's real identity is exchanged later on, within the encrypted second phase of the TTLS exchange.

The string that the supplicant sends in its identity response can be used for another purpose, quite separate from that of identifying the supplicant.

If the authenticator is servicing supplicants whose credentials are not all stored on the same authentication server, the Authenticator needs to rely on the supplicant to indicate which authentication server its EAP packets should be forwarded to. This is particularly the case when the Authenticator is a public point of access for clients of multiple different ISPs. The Authenticator must send each client's authentication details to the RADIUS server of the relevant ISP.

In that case, the identity sent by the supplicant will be of the form `<name>@<realm>`, where `<realm>` is typically the domain name of the owner of the RADIUS server. The Authenticator uses the `<realm>` part of the identifier to determine the address of the relevant RADIUS server—maybe by performing a DNS request, or by using a custom-configured list of realm-to-IP mappings.

RFC 5281 recommends that in this case, the identity sent by the supplicant should be of the form `anonymous@myisp.com`.



PEAP

Protected EAP operates in a very similar manner to EAP-TTLS. Again, it has a two-phase operation. Phase 1 uses TLS to negotiate encryption keys, then phase 2 performs the supplicant authentication within an encrypted conversation.

The main difference between PEAP and EAP-TTLS is that PEAP **requires** that the authentication method used in the second phase is a new (inner) EAP session. With EAP-

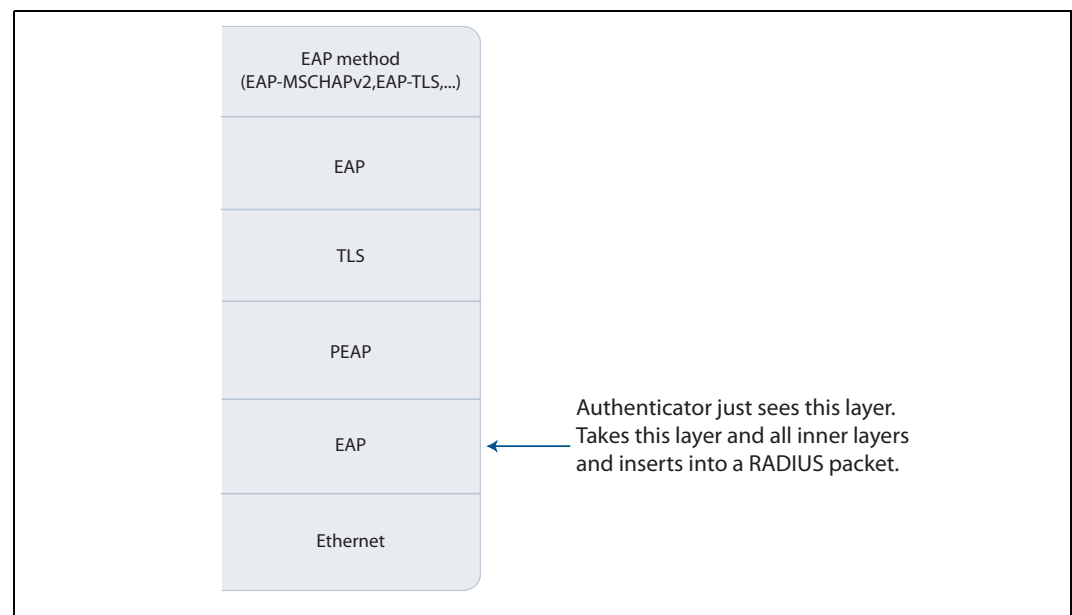
TTLS, the use of an 'inner' EAP session as the phase-2 authentication method is a possible option; with PEAP, it is the only option.

The value 25 has been allocated as the ID number used within EAP to identify the PEAP authentication method.

Protocol layering within PEAP

Like EAP-TTLS, PEAP is achieved by layering within a packet with EAP header. At the bottom of the stack is either EAPoL or RADIUS.

The next layer up from the EAPoL or RADIUS carrier is EAP.



This framework allows other supplicant↔server conversations to be transported in EAP particularly the Statement-of-Health (SOH) conversation for NAC.

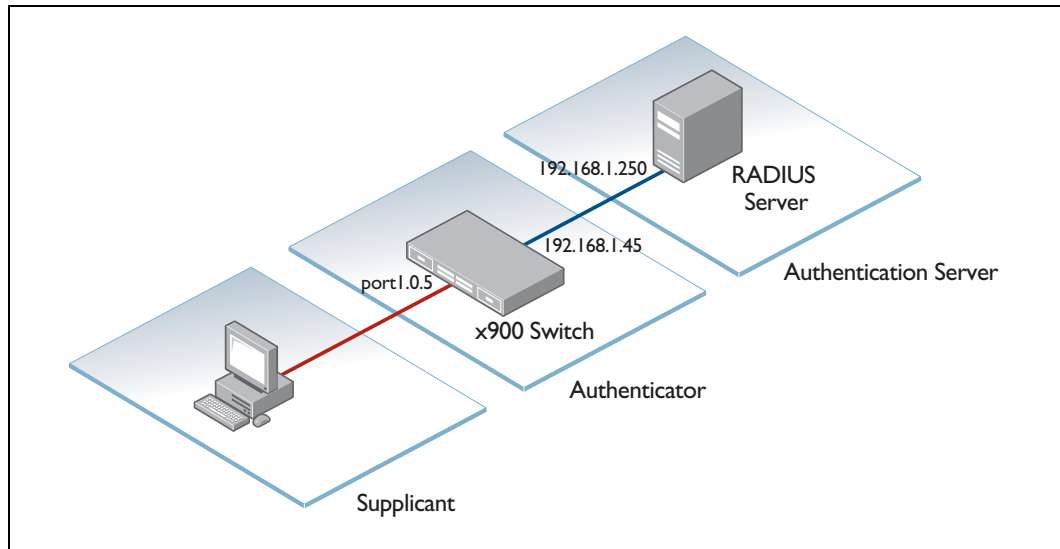
Table of authentication types

Process	EAP-MD5	EAP-TLS	EAP-TTLS	PEAP
Server Authentication	None	Public Key (Certificate)	Public Key (Certificate)	Public Key (Certificate)
Supplicant Authentication	Password Hash	Public Key (Certificate or Smart Card)	CHAP, PAP, MS-CHAP (v2), EAP	Any EAP like EAP-MS-CHAPv2 or Public Key
Creates encryption keys	No	Yes	Yes	Yes
Security Risks	Identity exposed, Dictionary attack, Man-in-the-middle (MitM) attack, Session hijacking	Identity exposed MitM attack	MitM attack	MitM attack

Configuring 802.1x

As a first 802.1x configuration example, consider a simple scenario:

- One supplicant
- One authenticating switch
- One RADIUS server



Let us look at the configuration required on all three devices:

Configuration for the switch operating as authenticator

1. Configure a RADIUS server for the switch to send requests to

```
awplus(config)# radius-server host 192.168.1.250 key
<secret-key>
```

2. Instruct 802.1x to use the configured RADIUS server

```
awplus(config)# aaa authentication dot1x default group radius
```

3. Configure port1.0.5 for 802.1x authentication

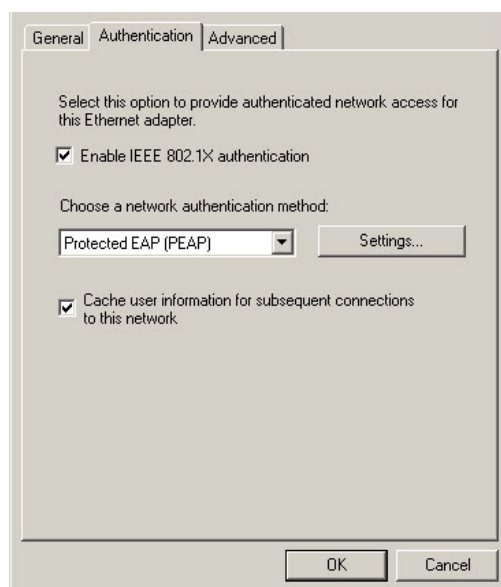
```
awplus(config)# interface port1.0.5
awplus(config-if)# dot1x port-control auto
awplus(config-if)# spanning-tree portfast
```

Supplicant configuration

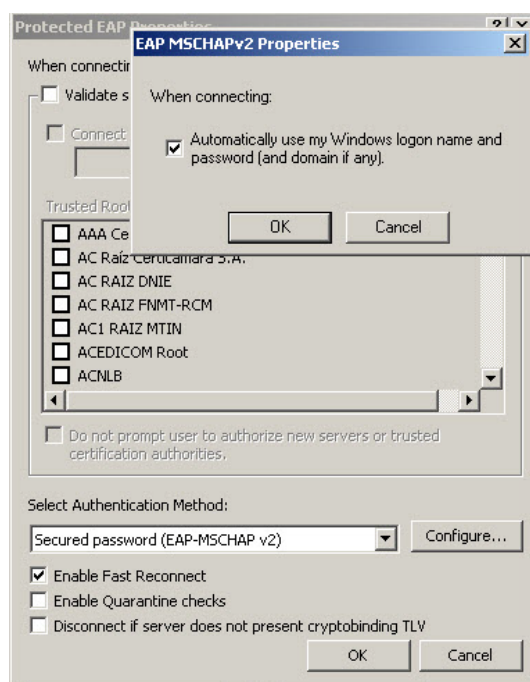
Setting up Windows 802.1x client.

- Under Windows XP the NIC card **properties** are accessed by:
Start >Settings >Network Connections >[NIC name]
- Under Windows 7/Vista it is accessed from the Network and Sharing centre.

NIC card properties



In the **Protected EAP Properties** window, click **Settings... > Configure...**



With the “Automatically use my Windows Logon name...” check box ticked, the PC will not need any user intervention in order to carry out 802.1x authentication. If the check box is not ticked, the PC will open a window asking for a username/password every time it needs to perform 802.1x authentications.

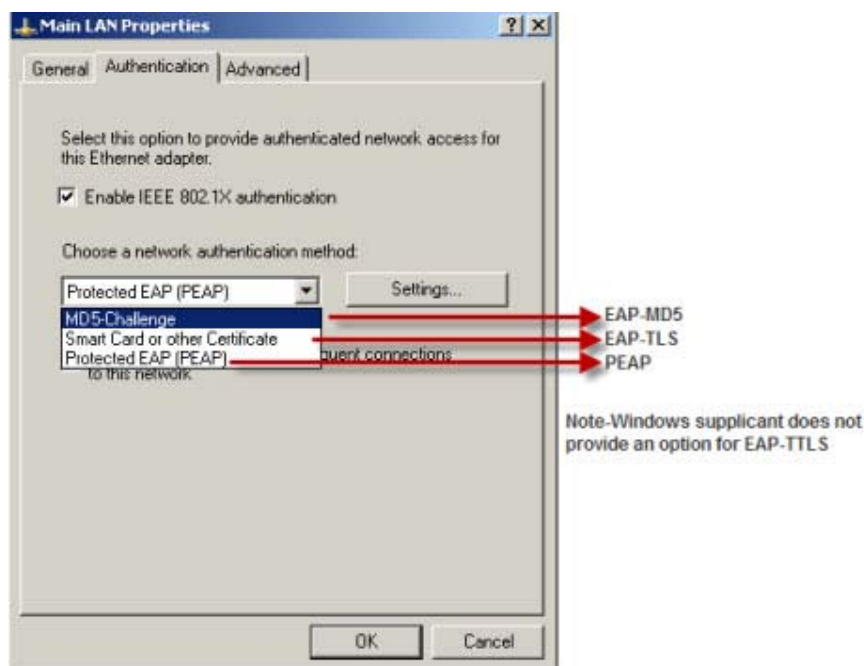
This is a particular problem at the time when the PC is logging into the network. It cannot log into the network until the network connection has been authenticated, but if it needs user input to authenticate the connection, it cannot proceed at login time, as the request for user input cannot be popped up at login time.

Configuring EAP types on a Windows XP supplicant

We have discussed the EAP authentication types:

- EAP-MD5
- EAP-TLS
- EAP-TTLS
- PEAP

The MS Windows XP 802.1x client does not name them all by these names. You can see a selection of authentication methods in the picture below:



Configuring Windows 2008 as a RADIUS Server

Within Windows 2008 Network Policy Server (NPS):

- Add a new RADIUS client, with an IP address and the secret key of the switch.
- Add a Connection Request Policy for local processing of requests.

Add Network Policy with settings:

- Condition: **User within user group to be 802.1x authenticated**
- Access Permission: **Grant Access**
- EAP Type: **PEAP**

Full details of configuring the Windows 2008 Network Policy Server are described in the Allied Telesis Solution document:

www.alliedtelesis.com/media/pdf/LAN_client_authentication.pdf.

Configuring AlliedWare Plus local RADIUS server as server

Enable the local RADIUS server:

```
awplus(config)# radius-server local
awplus(config-radsrv)# server enable
```

Configure the authenticating switch as a client:

```
awplus(config-radsrv)# nas 192.168.1.45 key <radius-secret>
```

Add users:

```
awplus(config-radsrv)# user <name1> password <password1>
awplus(config-radsrv)# user <name2> password <password2>
awplus(config-radsrv)# user <name3> password <password3>
```

Multi-supPLICANT modes

AlliedWare Plus can be configured to accept one or more supplicants downstream of a port. Three authentication host-modes are available:

- single-supPLICANT: the default state, only one supplicant allowed per port.
- multi-host: once the first host on a port is authenticated, all other downstream hosts are allowed without being authenticated (piggy-back mode).
- multi-supPLICANT: Multiple separate supplicants are individually authenticated on one port.

The command is (entered in interface configuration mode for a physical port interface):

```
awplus(config-if)# auth host-mode {single-supPLICANT|multi-
host|multi-supPLICANT}
```

This command controls how the switch deals with the situation where multiple authentication supplicants are downstream of a single port. This is possible if an EAP passes through a Layer 2 switch which has been connected to the port, and the supplicants are attached to that Layer 2 switch.

Single supplicant

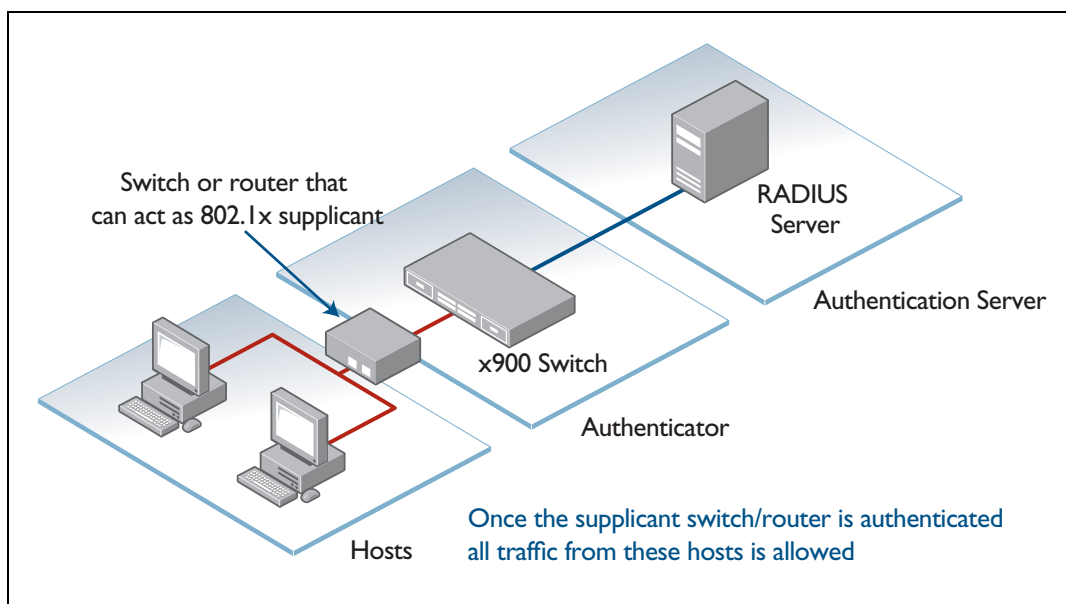
The first option that the command can set is single-host. With this option, only one supplicant may be authenticated on the port. Once that host has been authenticated, no other supplicants may be authenticated until the first supplicant's session has closed. This means, of course, that none of the other hosts downstream of the port will be able to send or receive traffic on that port.

This option is recommended when you know that there should only be one host connected to a port. By limiting the port to a single authenticated host, you guard against the consequences of someone accidentally or maliciously connecting a downstream switch to the port.

Multi-host

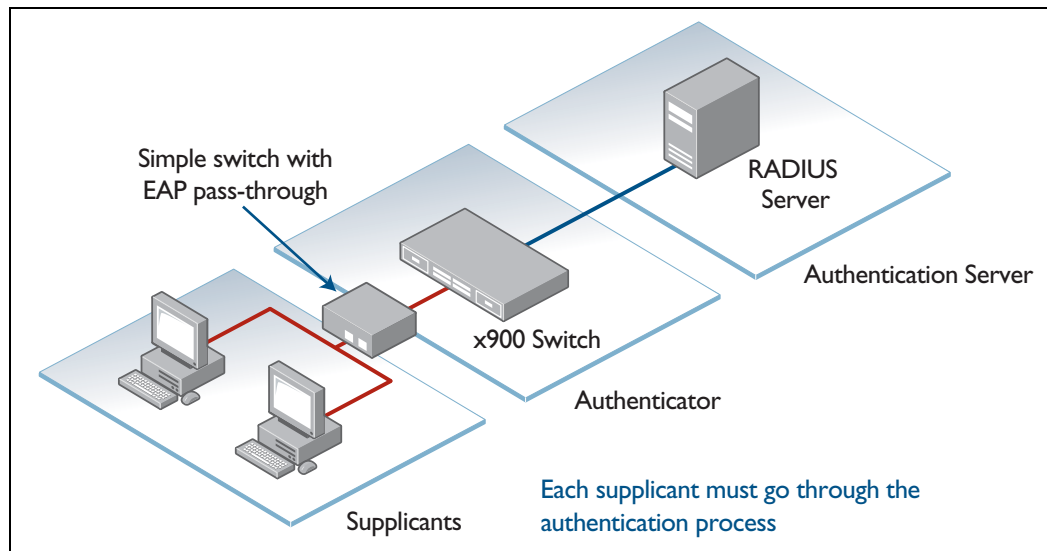
The next available host-mode option is multiple host mode (chosen by the parameter value multi-host). With this mode, once the first host has been authenticated on the port, all other downstream hosts are allowed without being authenticated. This is sometimes known as piggy-back mode. It is useful when the downstream switch attached to the authenticating port is an intelligent switch that can act as an authentication supplicant.

If you trust that malicious users cannot be connected to that switch but you do not know the identity of those users, then you can simply authenticate the switch and then allow its attached users to have network access. If the valid switch is disconnected and an invalid one is connected which is not configured with the correct authentication credentials, then the devices connected to the invalid switch will be blocked from accessing the network.



Multi-supPLICant

The final option is multi supplicant mode. In this mode, multiple separate supplicants can be individually authenticated on a single port. This mode is designed to support the situation where a lower-cost switch has been connected to an authenticating switch, in order to provide more connectivity at a lower cost per port.



Timers

Reauthentication

By default, once a supplicant is authenticated, it is not requested to re-authenticate until the start of the next session.

A port can be configured to periodically ask the supplicant to re-authenticate, for example every hour. The re-authentication period can be configured using the commands:

```
awplus(config)# interface port1.0.5
awplus(config-if)# auth reauthentication
awplus(config-if)# auth timeout reauth-period 3600
```

Enabling re-authentication is recommended. It increases the security of the solution, without any negative impact on data throughput or connection stability.

Quiet period

This feature protects the switch from a denial-of-service attack in which a malicious host hammers the switch with bogus authentication attempts, by ignoring authentication requests on a port for a period after a failed authentication attempt on that port. The default time period is 60 seconds. This time can be configured using the commands:

```
awplus(config)# interface port1.0.5
awplus(config-if)# auth timeout quiet-period 10
```

Server timeout

This is the timeout for awaiting a response from the RADIUS server, which defaults to 30 seconds. It can be configured using the commands:

```
awplus(config)# interface port1.0.5
awplus(config-if)# auth timeout server-timeout 120
```

Supplicant timeout

This is the timeout for awaiting a response from the supplicant which defaults to 30 seconds. It can be configured using the commands:

```
awplus(config)# interface port1.0.5
awplus(config-if)# auth timeout supp-timeout 2
```

802.1x VLAN Assignment

Dynamic VLAN assignment

Whilst the authentication of devices attaching to the network is primarily driven by security considerations, it has significant spin-off benefits.

Once a device has been authenticated, the network knows the identity of the device and/or its user. Decisions can be made, based on this identity. In particular, it is possible to decide what network environment, and level of access, to present to this device and its user.

The standard mechanism via which a user's network environment is controlled is VLAN membership. Once a user's packets are classified into a particular VLAN, the user's access to the network will be controlled by the constraints that have been put on that VLAN throughout the network.

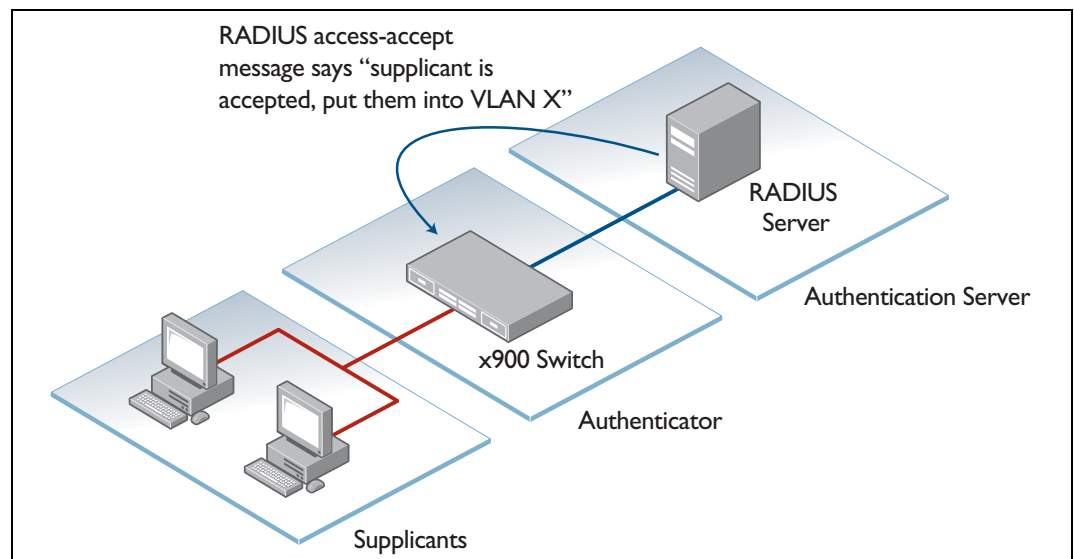
For this reason, it is now common for LAN switches to have the ability to dynamically assign the VLAN into which a device's traffic will be classified, once that device has been authenticated.

Dynamic VLAN assignment is achieved by a collaboration between the authenticator (the LAN switch) and the authentication server (the RADIUS server). When the RADIUS server sends back a RADIUS accept message to the authenticator, it can also include other attributes in that message that identify a VLAN to which the authenticated device should be assigned.

Dynamic VLAN assignment is a powerful extension to 802.1x, as it enables:

- Identity-based networking—the user gets the same environment no matter where they connect.
- Guest Access—guest users are allowed access to very limited parts of the network.

- NAC—level of access is based on a workstation's security status.



Authenticator configuration

In addition to the basic 802.1x configuration described in "[Configuration for the switch operating as authenticator](#)" on page 128, some further configuration is required to enable Dynamic VLAN creation on the switch. The VLANs that can be dynamically assigned must be present in the VLAN database:

```
awplus(config) # vlan database
awplus(config-vlan) # vlan x
awplus(config-vlan) # vlan y
awplus(config-vlan) # vlan z
awplus(config-vlan) # exit
```

Ports that accept VLAN membership dynamically have to be enabled for dynamic VLAN creation:

```
awplus(config) # interface port1.0.5
awplus(config-if) # auth dynamic-vlan-creation
```

Windows 2008 RADIUS server configuration

If a Windows 2008 server is being used as a RADIUS server for performing Dynamic VLAN assignment, then it must be configured with the VLAN attributes to inform the authenticator of the supplicant's VLAN.

Configuration on the Network Policy

Three RADIUS attributes must be set on the Network Policy:

1. Tunnel-Type—Virtual LAN
2. Tunnel-Medium-Type— 802
3. Tunnel-Private-Group-ID —<vlan id>

Tunnel-Type (64)

This attribute was originally defined in RFC2868, for use with various tunnelling methods like L2TP, GRE, IPSEC, etc.

For the purposes of 802.1x VLAN assignment, RFC3580 defined “VLAN” (13) as a new value for this attribute.

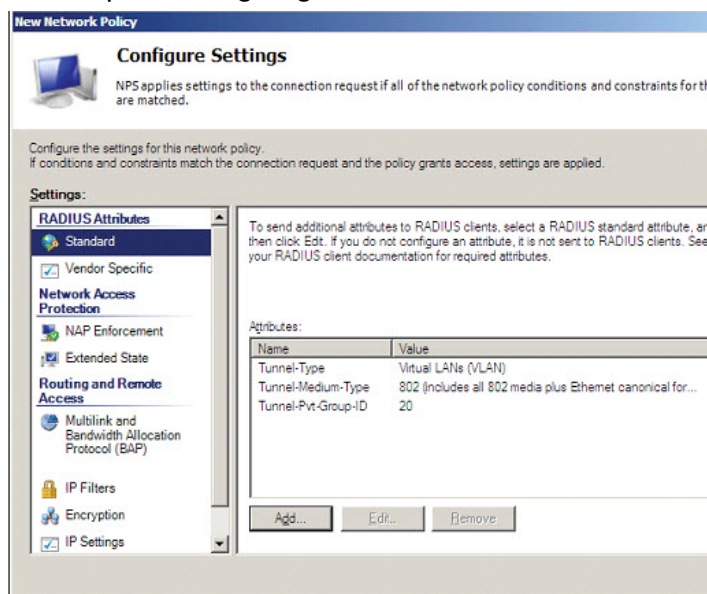
Tunnel-Medium-Type (65)

This is another attribute that was originally defined in RFC2868 for use with tunnels over IP networks. Its purpose was to indicate the medium over which the tunnel was being created. Again, this attribute has been seconded for use 802.1x VLAN assignment.

Tunnel-Private-Group-ID (81)

As with the two previous attributes, this one was also defined in RFC2868 for use with tunnels, but RFC3580 gave it another use in the context of 802.1x VLAN assignment.

An example of configuring these attributes on a network Policy is shown below:



Local RADIUS server configuration

To use the AlliedWare Plus local RADIUS server as the RADIUS server for Dynamic VLAN assignment, first configure the VLAN ID on a group, then associate each user that is to trigger dynamic VLAN assignment with the relevant. For example, to assign three different users to different VLANs, configure as follows:

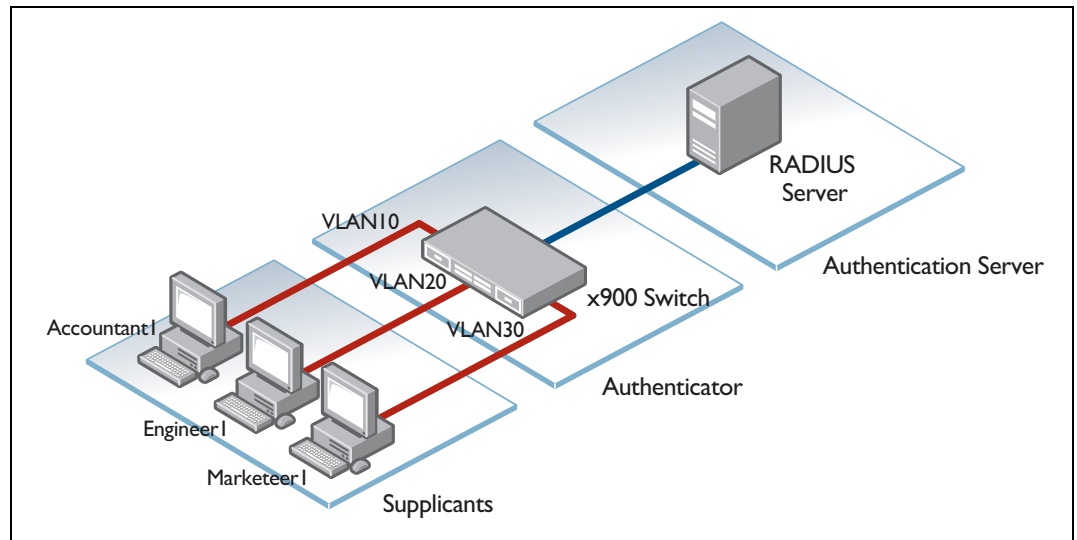
```
awplus(config)# radius-server local
awplus(config-radsrv)# group accounting
awplus(config-radsrv-group)# vlan 10
awplus(config-radsrv)# group engineering
awplus(config-radsrv-group)# vlan 20
awplus(config-radsrv)# group marketing
```

```
awplus(config-radsrv-group)# vlan 30
```

```
awplus(config-radsrv)#user Accountant1 password <password>
group accounting
```

```
awplus(config-radsrv)#user Engineer1 password <password>
group engineering
```

```
awplus(config-radsrv)#user Marketeer1 password <password>
group marketing
```



The three supplicants are assigned to different VLANs, VLAN 10, 20, and 30.

VLAN assignment application

A good example of using different VLAN assignment can be found in a school campus, where there are:

- Multiple, physically separated, networked switches
- Multiple VLANs with different access rights
- Many roaming network users, with different access rights

Consider a school that has three groups of users:

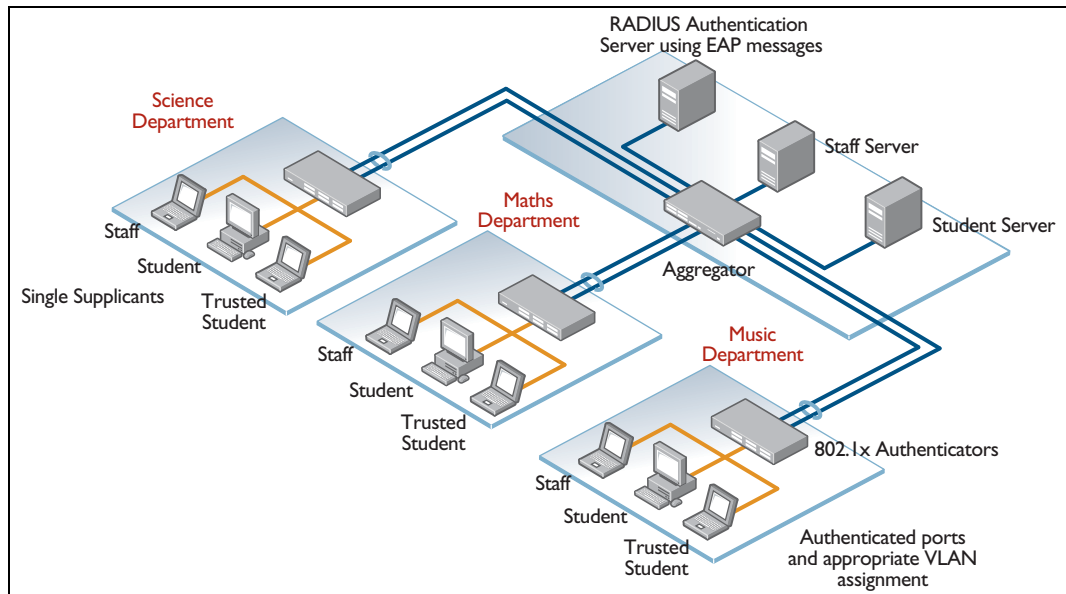
1. Students
2. Trusted students
3. Staff

Solution requirements

A member of one of the three groups can connect to any port on any edge switch, and immediately be assigned to the VLAN appropriate to their group.

Complete data separation is required between the three VLAN groups—no member of one group can exchange data with a member of another group.

School campus application



Configuration

- The RADIUS server is accessed via the “authentication” VLAN. This is the only VLAN with an IP address on each edge switch.
- All other ports provide simple Layer 2 switching after VLAN membership is confirmed.
- All VLANs must be pre-defined (i.e. created) before 802.1x supplicant ports can be dynamically added to the appropriate VLAN by 802.1x.
- The Allied Telesis device will be the authenticator, having port authentication and VLAN assignment enabled on ports 1 through 23.
- Port 24 is reserved for connection to the RADIUS server via the “authentication” VLAN—VLAN 1.

Configure the pre-defined VLANs

```
awplus(config)# vlan database
awplus(config-vlan)# vlan2 name students
awplus(config-vlan)# vlan3 name trusted-students
awplus(config-vlan)# vlan4 name staff
awplus(config-vlan)# exit
```

Do not add edge ports to VLANs in the configuration. 802.1x VLAN assignment will add ports dynamically.

Add tagged uplink ports to all the VLANs:

```
awplus(config-vlan)# int port1.0.24
awplus(config-vlan)# switchport mode trunk
awplus(config-vlan)# switchport trunk allowed vlan add 2,3,4
```

Assign a port and IP address to the “authentication” VLAN to allow RADIUS authentication.

```
awplus(config)# int vlan1
awplus(config-if)# ip address 10.0.10.254/24
awplus(config-if)# exit
```

Define the RADIUS authentication server:

```
awplus(config)# radius-server host 10.0.10.10 key testing123
awplus(config)# aaa authentication dot1x default group radius
```

Configure port authentication on ports 1 through 23:

```
awplus(config)# int port1.0.1-1.0.23
awplus(config-if)# dot1x port-control auto
awplus(config-if)# auth dynamic-vlan-creation
```

Dynamic VLAN assignment with multiple supplicants

In multi-supplicant mode, what happens if two supplicants downstream of the same port are assigned to different VLANs?

The **auth dynamic-vlan-creation** command has two parameters that govern the operation in this situation: rule and type.

The rule
parameter

The first parameter is the **rule** parameter.

For SBx908 and x900 Series switches (the situation is different for the x600 and x610 Series, as we will see below) it is not possible to assign different VLANs to untagged traffic from different supplicants. On the SBx908 and x900, dynamic VLAN assignment effectively says ‘the one untagged VLAN to be used on the authenticating port is VLAN x’. So, if the first supplicant is authenticated and assigned VLAN 45, then the authenticating port will classify all untagged traffic arriving on the port into VLAN 45. But if a second supplicant downstream of the same port then authenticates, and the RADIUS server assigns VLAN 56 to that supplicant, the switch then faces a dilemma. It is already using VLAN 45 as the untagged VLAN on that port; it cannot use VLAN 56 as well.

There are two ways that the switch can resolve this situation. It can:

1. Allow the second supplicant to access the network, but assign its data to VLAN 45.
2. Block the second supplicant from having network access.

The rule parameter configures which of these choices the switch will opt for. If rule is set to permit, then option (1) above is chosen. If rule is set to deny, then option (2) above is chosen.

The type parameter

The second parameter is the type parameter.

The type parameter applies only to the x600 and x610 Series switches. This is because these switches support MAC-based VLANs, whereas the x900 Series and SBx908 do not.

The effect of the type parameter is to make use of the x600 and x610's MAC-based VLAN support to provide a better solution to the case where different supplicants downstream of a single port are dynamically allocated to different VLANs.

If type is set to the value single, then the MAC-based VLAN capability is not used, and the port's behaviour in the different-dynamic-VLANs situation will be controlled by the rule parameter.

However, if type is set to multi, the x600/x610 switch brings the MAC-based VLAN capability into play. This capability enables it to support multiple different untagged VLANs on the same port. This is achieved by associating VLAN membership with the source MAC address of the incoming packets.

So, when different supplicants downstream of a single port are dynamically assigned different VLANs, the x600/x610 switch simply builds a table that maps supplicants' MAC addresses to their dynamically assigned VLANs.

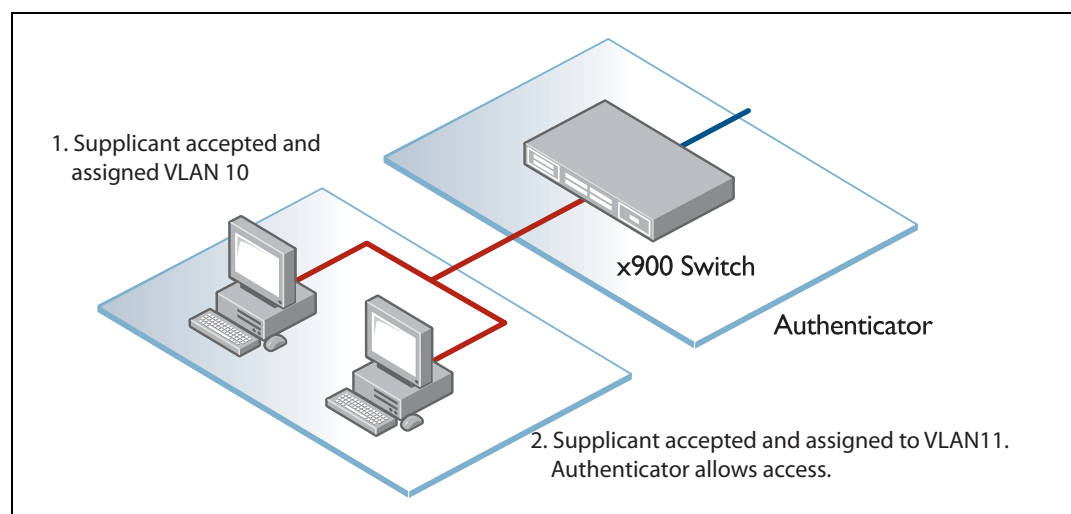
The combination of these parameters results in three options for handling the case where different VLANs are assigned to supplicants on the same ports.

Option 1 **Deny access to supplicant assigned a different VLAN.**

If the first supplicant authenticated on the port is assigned VLAN X, then any supplicants subsequently assigned a different VLAN are denied access. This is the default state when dynamic VLAN creation is enabled.

This is configured with:

```
awplus(config-if)# auth dynamic-vlan-creation rule deny
```

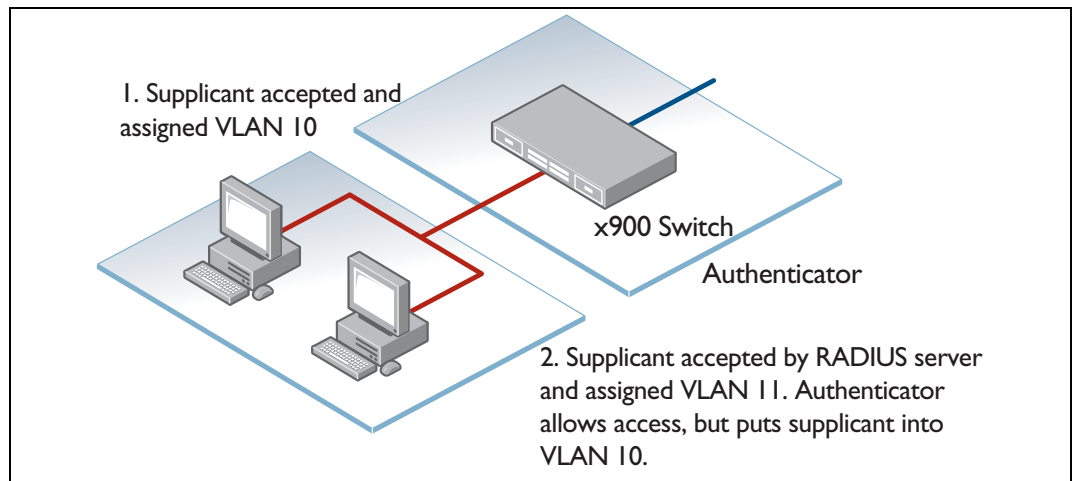


Option 2 Force all supplicants into the same VLAN

If the first supplicant authenticated on the port is assigned VLAN X, then any supplicants subsequently assigned a different VLAN are allowed access, but forced into VLAN X

This is configured with:

```
awplus(config-if)# auth dynamic-vlan-creation rule permit
```

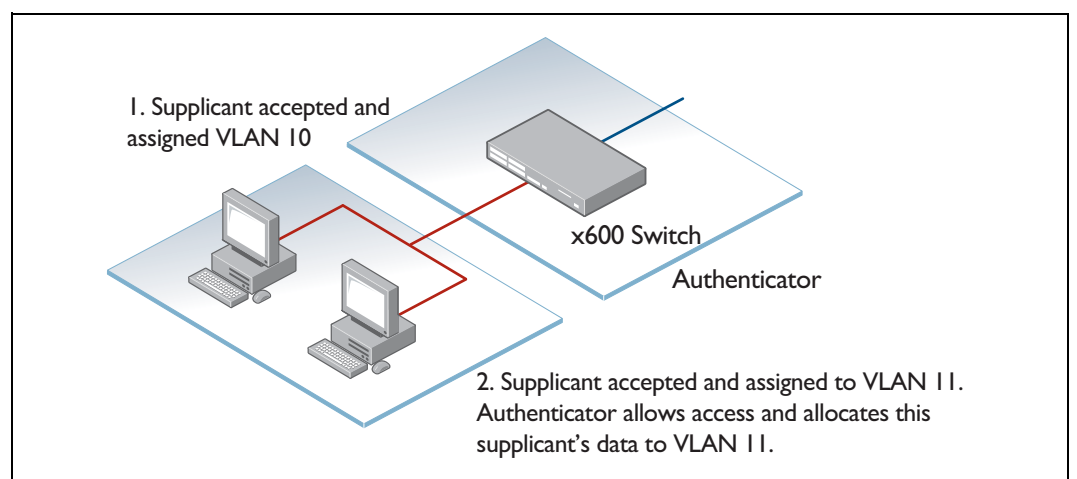


Option 3 Dynamically assign multiple VLANs to one port

On the x600 and x610 switches, it is actually possible to assign different VLANs to different supplicants downstream of the **same** port.

This is configured with:

```
awplus(config-if)# auth dynamic-vlan-creation rule permit type multi
```



The x600/x610 switch can assign VLAN membership to packets based on source MAC:

- Packets from MAC of supplicant 1 are assigned to VLAN10
- Packets from MAC of supplicant 2 are assigned to VLAN11

This feature is not supported on x900 and SwitchBlade x908 switches.

Using a guest VLAN

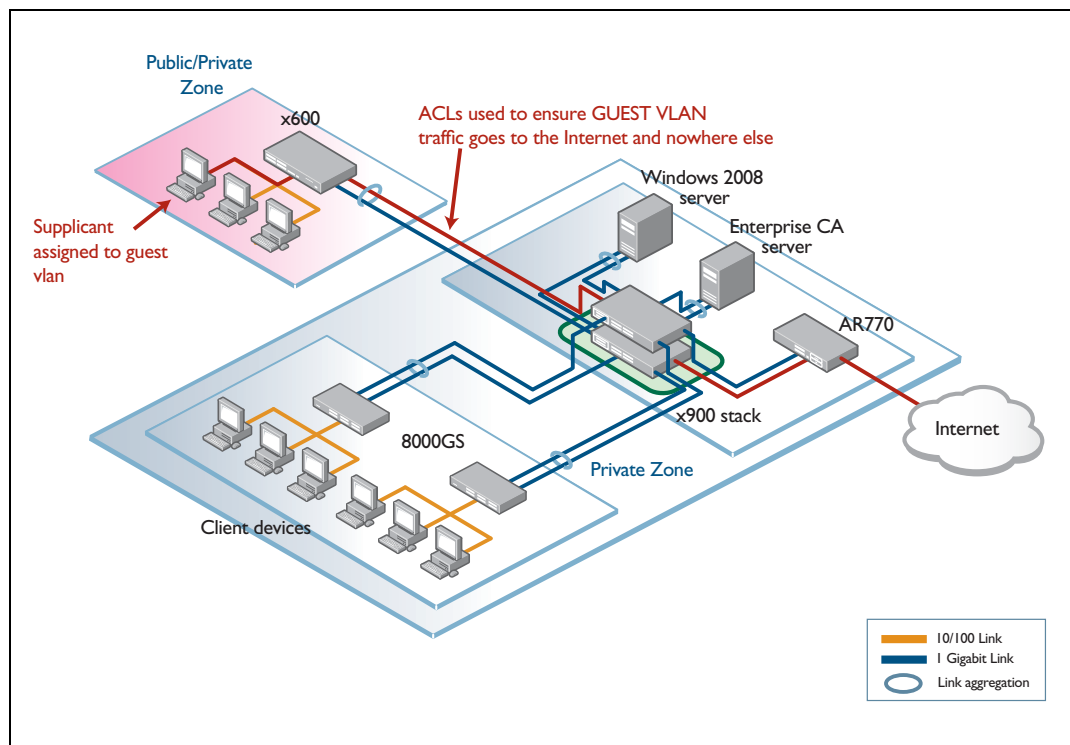
Whilst you need to authenticate the users who will have access to the important services within your network, you might also want to provide some basic level of access to users who fail to authenticate.

For example, visitors to an enterprise will often need to have Internet access. It would be desirable to have a secure, convenient way to provide this Internet access via the corporate LAN.

By default, 802.1x denies access to users who fail authentication.

Guests are not known to the RADIUS server, so fail authentication. The solution is to provide a Guest VLAN which is configured with:

```
awplus(config)# interface port1.0.5
awplus(config-if)# auth guest-vlan <vlan id>
```



If a supplicant attempts authentication and fails **or** does not even attempt authentication (no 802.1x client in the PC) then they are dynamically assigned to the guest VLAN.

Verify the operation of 802.1x

When a supplicant has been authenticated on a port the details of the authentication can be seen with:

```
show dot1x supplicant int port1.0.5

Interface port1.0.5
 authenticationMethod: dot1x
 totalSupplicantNum: 1
   authorizedSupplicantNum: 1
   macBasedAuthenticationSupplicantNum: 0
   dot1xAuthenticationSupplicantNum: 1
   WebBasedAuthenticationSupplicantNum:
   otherAuthenticationSupplicantNum: 0
 Supplicant name: Engineer01
 Supplicant address:
 0002.b363.319f
 authenticationMethod: 802.1X
 portStatus: Authorized - currentId: 9
 abort:F fail:F start:F timeout:F success:T
 PAE: state: Authenticated - portMode: Auto
 PAE: reAuthCount: 0 - rxRespId: 0
 PAE: quietPeriod: 60 - maxReauthReq: 2
 BE: state: Idle - reqCount: 0 - idFromServer: 8
 CD: adminControlledDirections: both - operControlledDirections:
 both
   CD: bridgeDetected: false
   KR: rxKey: false
   KT: keyAvailable: false - keyTxEnabled: false
 dynamicVlanId: 20
 assignment enabled
```

Annotations in the original image:

- <--- Authenticated by 802.1x (points to authenticationMethod: dot1x)
- <--- Supplicant name (points to Supplicant name: Engineer01)
- <---MAC of authenticated device (points to Supplicant address: 0002.b363.319f)
- <--- VLAN assigned, if dynamic VLA (points to dynamicVlanId: 20)

When a supplicant has been authenticated, and assigned to a VLAN, the port they authenticated on will then be seen to be a member of that VLAN.

```
show vlan 20

VLAN ID  Name          Type State   Member ports
(u)-Untagged, (t)-Tagged
=====
20  Engineering    STATIC ACTIVE  port1.0.5(u)

show vlan 30

VLAN ID  Name          Type State   Member ports
(u)-Untagged, (t)-Tagged
=====
30  Marketing      STATIC ACTIVE  port1.0.5(u)
```


And the x600 and x610 VLAN tables will show these as MAC-based VLAN memberships.

```
show platform table vlan
[Instance 1.0]
VLAN table
-----
Mac Based Vlan Information:
Index Mac          Vid Prio
-----
212 0002.b363.319f 20  0
996 000e.2e5f.a7fc 30  0
```

Troubleshooting

This section provides a summary of the 802.1x troubleshooting commands in available AlliedWare Plus:

`show dot1x diagnostics` `awplus# show dot1x diagnostics`

This command outputs a set of counters for events that have occurred, on a per-port basis including how many times:

- a supplicant has sent logoff before completing authentication
- a supplicant has been successfully authenticated on the port
- there has been a timeout while waiting for a response from a supplicant

This command is useful for tracking down if the conversations with supplicants are going smoothly, or if there are a number of occasions when EAPoL packets arrive from a supplicant at unexpected points in the authentication process.

`show dot1x statistics interface` `awplus# show dot1x statistics interface`

This command outputs a set of counters for how many of each type of EAPoL packet have been received on a port, including how many invalid and errored frames.

This command is useful for working out whether the port is successfully sending and receiving frames, and if frames are being dropped due to errors.

`debug dot1x` `awplus# debug dot1x`

This command turns on a mode where trace-level information is output during authentication conversations. Be aware that this is a very verbose output.

It is mostly useful to capture this as part of escalating an issue to ATI support.

show RADIUS `awplus# show radius`

This command shows the status of communication with all configured RADIUS servers. Servers are listed as **alive** or **dead**, depending on whether they are responding to requests

show RADIUS
statistics `awplus# show radius statistics`

This command outputs counters for:

- each type of RADIUS packet sent or received
- a number of different errors seen in RADIUS packets
- how many times a timeout has occurred when waiting for RADIUS server response

This is useful for working out whether the switch is successfully sending and receiving RADIUS packets, and if packets are being dropped due to errors.

debug RADIUS `awplus# debug radius`

This command turns on a mode where trace-level information is output during authentication conversations.

Be aware that this is a very verbose output. It is mostly useful to capture this as part of escalating an issue to ATI support.

Simple Definitions for 802.1x Port Authentication

Port authentication

802.1x port authentication is an IEEE standard for port-based Network Admission Control, part of the IEEE 802 (802.1) group of protocols. It provides authentication to devices attached to a LAN port, establishing a connection or preventing access from that port if authentication fails.

- It provides authentication to 802.1x supplicant devices attached to a LAN port.
- It allows access from that port if authentication succeeds and prevents access from that port if authentication fails.
- There are three roles: supplicant, authenticator, and authentication server.

802.1x VLAN Assignment

As described above, but now a port can be dynamically assigned to a VLAN that the RADIUS server has defined.

VLAN Assignment is based on the user's credentials. The EAP—Extended Authentication Protocol is used to allow RADIUS to authenticate hosts and specify VLAN membership

Data separation may be achieved—hosts run on separate VLANs

It removes the need for MAC-based VLANs.

Authentication methods

The common authentication methods are:

- MD5
- EAP-TLS or EAP—uses certificates
- EAP-TTLS—encrypts authentication exchange
- PEAP—very similar to EAP-TTLS, but forces encrypted authentication to use EAP

Chapter 9 | Tri-authentication

Introduction

One of the keys to a good security system is to eliminate chinks in the armour. In the context of device authentication, this means building a network where no device gets access via a network edge port without authentication.

To achieve this, the network needs to implement not only 802.1x and Web-authentication, but also a third authentication method—MAC-based authentication.

In this chapter, we will look at why MAC-based authentication is necessary, and how it works. Then we will look at how 802.1x, Web-based, and MAC-based authentication can be combined to provide Tri-authentication—a comprehensive barrier against unauthorized access to a network.

List of terms

MAC-based authentication

Authenticating devices using their MAC address as the identifier.

MAC-based VLANs

VLANs in which membership is determined on a packet-by-packet basis, based on the packet's source MAC address.

Why is MAC-based Authentication Required?

The authentication mechanisms provided by 802.1x and Web authentication are powerful and effective. But, they are not universally applicable. Web authentication is only applicable to devices that have a human user who opens the web browser and types in a username and password when requested. 802.1x authentication is only possible from devices whose software implements an 802.1x supplicant.

There are plenty of network-connected devices, like printers, scanners, fire-alarm monitors etc., that have neither a human user nor implement an 802.1x supplicant. In a network that ensures all access is authenticated, there needs to be a mechanism for authenticating these devices.

Fortunately, all Ethernet transceivers have a unique identifier—their **MAC address**. Hence, even without user input of a username/password, any Ethernet device will automatically identify itself simply by virtue of the source MAC address in the packets it sends. The method that has been developed for authenticating these devices uses the MAC address as the identifier, and so is called MAC-based authentication.

How does MAC-based authentication work?

In essence, MAC-based authentication works little differently from 802.1x or Web-based authentication.

Here are the main steps:

1. The supplicant is connected to the switch.
2. The switch (acting as the authenticator) receives an ID from the supplicant.
3. The switch passes the supplicant's ID to a RADIUS server in an Access-Request packet
4. The RADIUS server returns an Access-Accept or an Access-Deny. The Access-Accept can be accompanied with other attributes, for dynamic VLAN assignment.

The unique aspects of MAC-based authentication are in steps 2 and 3.

MAC-based authentication does not involve a process whereby the switch sends an ID request to the supplicant. The switch receives the ID from the supplicant by simply looking at the source MAC in the packets being sent from the supplicant.

The MAC address of the supplicant is a single identifier. But a RADIUS access-request requires both a username and a password. The workaround employed by MAC-based authentication is simply to use the MAC address as both username **and** password.

The switch extracts the source MAC address from the supplicant's packets and puts it into a string of the form xx-xx-xx-xx-xx-xx, using lower-case letters for any hex digits in the range a-f. This string is then used as both the username and the password in the RADIUS access-request packet. The supplicant MAC address is also sent in the attribute 31 “calling-station-id” as usual.

Configuring MAC authentication

Under AlliedWare Plus, there are two steps to setting up MAC authentication.

1. Define the authentication method list that is used for MAC authentication.

There is only one method list that can be created for MAC authentication—the **default** method list. Moreover, the only authentication server type that can be used is RADIUS.

The typical command for defining the method list is:

```
awplus(config)# aaa authentication auth-mac default group
radius
```

2. Enable MAC authentication on the port(s) that are to perform this authentication:

```
awplus(config)# interface port1.0.2
awplus(config-if)# auth-mac enable
awplus(config-if)# spanning-tree edgeport
```

On the RADIUS server, it is necessary to create user entries where both the username and password are the MAC address of the supplicant, in the form xx-xx-xx-xx-xx-xx. For example on the AlliedWare Plus local RADIUS server, the configuration would be:

```
awplus(config)# radius-server local
awplus(config-radsrv)# user xx-xx-xx-xx-xx-xx password xx-xx-
xx-xx-xx-xx
```

The supplicant, of course, requires no configuration, as the whole purpose of MAC-based authentication is to authenticate devices that cannot be configured for authentication.

It is also possible to configure the authentication protocol that the switch uses in its interaction with the RADIUS server. There are two choices of protocol: EAP-MD5 and PAP. The **default** method is PAP, and can be changed by using the command:

```
awplus(config-if)# auth-mac method [eap-md5|pap]
```

A superior alternative to legacy MAC-based VLANs

By combining MAC-based authentication, as described above, and dynamic VLAN assignment, you can configure a MAC-based VLAN solution. This solution is centrally managed—the list of allowed MAC addresses for each VLAN is stored on the RADIUS server. It does not have to be individually configured on each switch.

This is a more effective and convenient solution than the legacy MAC-based VLAN solution that was available on some networking hardware some years ago. The legacy solution had a fundamental problem: it could not simultaneously provide data separation between different MAC-based VLANs while allowing any given port on an edge switch to be a member of multiple MAC-address VLANs.

There were also a number of other issues with using legacy MAC-based VLANs:

- Multiple hosts on one physical port could actually be members of different VLANs, i.e. the port could be a member of several VLANs at once, which created an opportunity for some unauthorized access to other VLANs.
- All MAC addresses belonging to the VLAN had to be configured onto each switch.
- Any PC changes meant that the switch configurations had to be updated.

These problems do not exist when using MAC-based authentication with dynamic VLAN assignment, because the switch only assigns the VLAN to the port when the relevant supplicant is connected.

Tri-authentication operation

802.1x, Web-based, and MAC-based authentication methods are intended to authenticate different types of supplicants. However, it is not always possible to predict which type of supplicant will be connected to which port. So, to ensure all possibilities are covered, it is often necessary to configure all three authentication methods on the same ports.

AlliedWare Plus supports the configuration of any combination of two authentication methods, or all three authentication methods on the same port.

At first glance, you may think that having multiple authentication methods working on the same port would have them all competing and tripping over each other. Fortunately, it is not as bad as that.

The only potential conflict is MAC authentication conflicting with Web and/or 802.1x authentication. Whilst Web authentication and 802.1x use completely different packet types to each other, and so are easily distinguishable from each other, MAC authentication will operate on any packet type (although, it will in fact not operate on 802.1x EAPoL packets).

Certainly, MAC authentication will attempt to perform authentication based on the source MAC of any packets (except EAPoL packets) it receives from a supplicant. However, if the supplicant is to be authenticated by Web or 802.1x authentication, then the supplicant's MAC address should not be in the RADIUS user database. Therefore, the

MAC authentication will simply fail, and the switch will then wait for authentication attempts by other methods.

Therefore, when the supplicant sends EAPoL packets, or instigates Web browsing, the switch will proceed with the appropriate authentication process, irrespective of the fact that MAC authentication is also configured on the port.

If the supplicant's MAC address is present in the RADIUS server's user database, then the initial MAC authentication will succeed, and the supplicant will be authenticated. The supplicant will not then proceed to attempt Web or 802.1x authentication, as the whole purpose of MAC authentication is to support those devices that cannot perform Web or 802.1x authentication.

It is essential that you only add the MAC addresses of devices to the RADIUS server's database if you intend those devices to be authenticated by MAC authentication.

Multiple supplicants on tri-authenticating ports

If there are multiple supplicants accessing the network via a port configured with Tri-Authentication, there is no restriction on the combinations of different authentication methods that can be used to authenticate the different supplicants. The switch can support having some supplicants authenticated by 802.1x, some by MAC authentication and some by Web authentication, all at the same time.

The rules governing dynamic VLAN creation when there are multiple supplicants on the same port are described in ["Dynamic VLAN assignment with multiple supplicants" on page 139](#).

These rules apply equally to the tri-authentication case as to the 802.1x-only case.

Chapter 10 | DHCP Snooping

Introduction

The seemingly-simple act of a switch keeping track of the DHCP packets passing through it has a surprising number of security applications:

- Compensating for the inherent lack of security in the DHCP protocol.
- Preventing IP address spoofing.
- Preventing ARP cache poisoning.
- Providing an audit trail of who had which IP address and when.

This chapter will begin by describing the mechanics of DHCP snooping, and then move on to explain how it delivers the rather wide-ranging network security features listed above.

List of terms

DHCP Option 82

The DHCP Relay Agent Information Option. Inserts circuit specific information into a request that is being forwarded to a DHCP server. Specifically the option works by setting two sub-options: Circuit ID and Remote ID.

DHCP snooping

DHCP snooping provides an extra layer of security on the switch via dynamic IP source filtering.

DHCP snooping filters out traffic received from unknown or “untrusted” ports and builds and maintains a DHCP snooping database.

What is DHCP snooping?

DHCP snooping is the act of a Layer 2 or Layer 3 switch watching all the DHCP packets that pass through it. The switch maintains a database of information that it extracts from the packets, and it may insert the Option 82 field into the packets.

DHCP snooping is quite different to DHCP relay. DHCP relay is an integral part of the DHCP lease distribution process, which enables DHCP requests to cross Layer 3 boundaries. DHCP snooping, on the other hand, does not assist DHCP to perform its lease distribution task at all—it is focused on a quite separate set of security tasks.

DHCP relay changes the source and destination IP addresses on DHCP requests; DHCP snooping only alters packets to the minimal extent of inserting Option 82 information into them. DHCP snooping will filter out DHCP packets that it deems to be attempting to use DHCP as part of an attack on the network.

The basic mechanism of DHCP snooping

DHCP snooping forces all DHCP packets arriving at a switch to be sent up to the switch CPU before forwarding. The switch CPU is then able to look into the DHCP packets to see which IP addresses are being allocated to which clients. From this information, the switch maintains a database of the IP addresses that are currently allocated to downstream clients and the switch ports that the relevant clients are attached to.

As well as just storing information in the database, DHCP snooping can carry out other activities:

- Drop packets that it considers to be malicious.
- Insert Option 82 information into DHCP packets going from clients to the server.

Let us look at the details of the DHCP snooping database, and then at the detection of malicious packets, and the insertion of Option 82 information.

The DHCP snooping database

As DHCP snooping sees that a DHCP lease has been successfully allocated to a client, it creates an entry in the database. Each entry contains the following information:

Item	Description
Client IP	The IP address that has been allocated to the snooped DHCP client.
MAC Address	The MAC address of the snooped DHCP client.
Server IP	The IP address of the DHCP server.
VLAN	The VLAN to which the snooped DHCP client is connected
Port	The port to which the snooped DHCP client is connected.
Expires	The time, in seconds, until the DHCP client entry will expire.

The command **show ip dhcp snooping binding** displays the database content.

```
awplus#show ip dhcp snooping binding

DHCP Snooping Bindings:

Client IP      MAC Address    Server IP      VLAN Port  Expiry(s)  Type
-----
192.168.10.100 e0.18b3.d1a6 192.168.10.253 10 1.0.19 3207 Dyna

Total number of bindings in database: 1
```

The database is held in RAM, but is frequently backed up to permanent storage, so that the information will survive across switch reboots.

On a switch running AlliedWare Plus, you can choose which form of permanent storage the database will be backed up to. The choices are:

- NVS (the default)
- Flash
- SD card

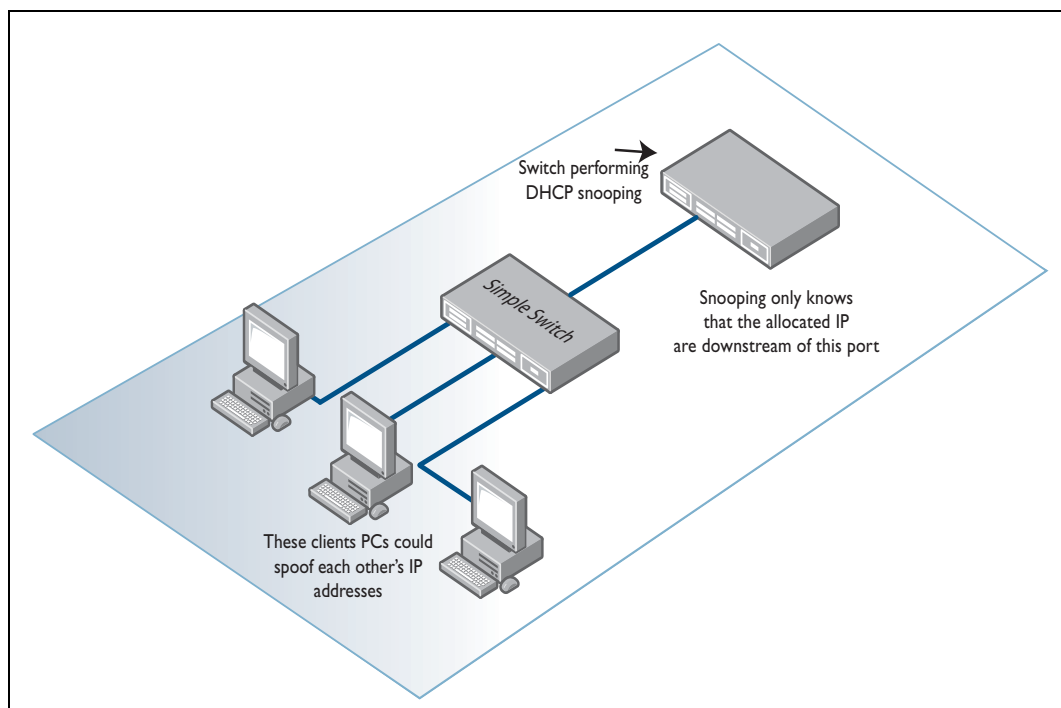
The command to specify the database storage location is:

```
awplus(config)# ip dhcp snooping database {nvs|flash|card}
```

The main purpose of the database is so that the switch can block IP address spoofing. The switch knows which clients, attached to which ports, are allocated which IP addresses. With this knowledge, it can then block packets arriving on port X with a source IP that it knows has been allocated to a client on port Y. Similarly, if a packet from a workstation arrives with a source IP address that does not appear anywhere in the DHCP snooping database, that packet is also seen to be not using its correctly allocated IP address, and is dropped.

The IP address spoofing protection is most effective when the client workstations are directly connected to the DHCP snooping switch. If there is another simple switch between the DHCP snooping switch and the workstations, then the client workstations downstream of any given port on the DHCP snooping switch could spoof each other's IP address.

The mechanics of how the switch performs the spoofing protection (also known as “IP source guard”) will be described in more detail in the section ["IP source guard" on page 161](#).



To support the case where one or more client PCs have been (validly) configured with static IP addresses, it is possible to add static entries into the DHCP snooping database. If this were not possible then the DHCP snooping database would be unaware of those workstations IP addresses and would drop their packets, as the packets would be arriving with source IP addresses that do not appear in the DHCP snooping database.

The AlliedWare Plus command to add a static entry to the DHCP snooping database is, for example:

```
awplus# ip dhcp snooping binding 172.16.1.202 0000.0000.00CA
      vlan 1 interface port1.0.2 expiry 3600
```

Entries are removed from the DHCP snooping database when:

- Their lease expires.
- The client explicitly releases a lease (this is configurable, and enabled by default).
- The port connected to the client goes down (this is configurable, and off by default).
- The entry is deleted with the command **no ip dhcp snooping binding <ipaddr>**.

Dropping malicious packets

Without DHCP snooping, there is no security built into the DHCP protocol. Clients and servers have no way to validate each other's identity. This opens networks up to DHCP-based attacks.

Well known DHCP-based attacks include:

Rogue DHCP server attack: a malicious workstation can act as a DHCP server, and reply to other workstations' requests. It can allocate them incorrect IP addresses (i.e. addresses incompatible with the route tables in the network). Even worse, the DNS server address that the rogue DHCP server provides to the workstations can be the address of a malicious DNS server that will direct users to imposter sites when they attempt to access their Internet banking for example.

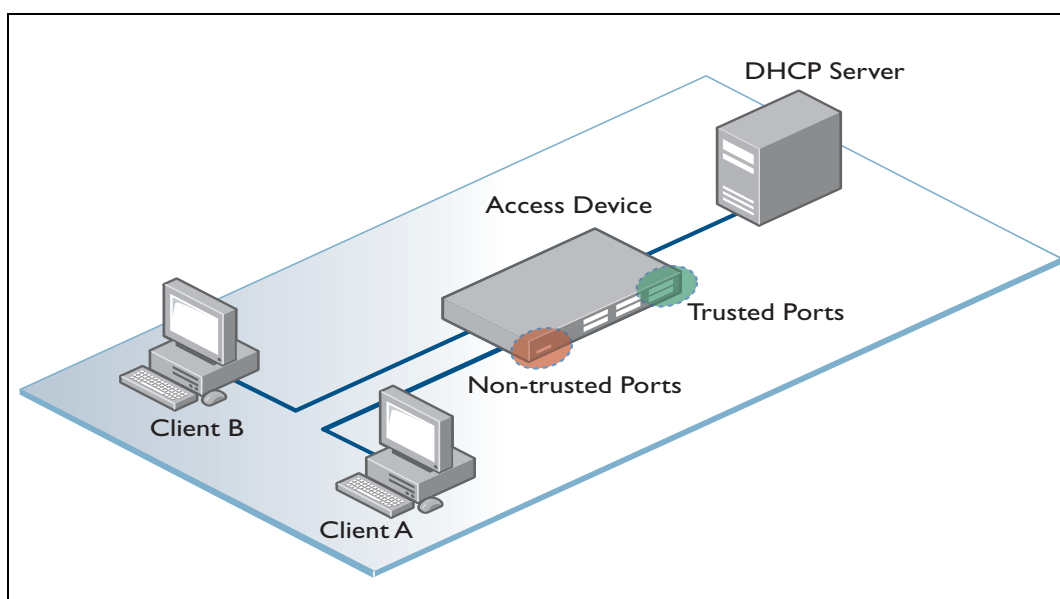
DHCP server exhaustion attack: a single malicious workstation can request innumerable leases from a DHCP server by pretending to be a number of different client devices. If it keeps on changing the client ID in its requests, the server will keep on allocating it yet more leases. This process can quickly use up the whole of the server's pool of available IP addresses, thereby denying other workstations in the network the opportunity of obtaining IP addresses.

Defending against DHCP-based attacks

Rogue server
attacks

DHCP snooping can make up for the protocol's lack of security, and defend the network against these attacks. To defend against rogue DHCP server attacks, DHCP snooping introduces the concept of trusted and untrusted ports.

Trusted ports are typically ports connected towards the core of the network. Untrusted ports are those connected towards workstations.



The idea is that the devices in the network core are under the control of the network administrator, and are trusted, but users workstations are under less control. This is because workstations could be infected with malware, or malicious users can deliberately

install attacking software onto their workstations. DHCP snooping applies a simple rule—DHCP packets that would come from a DHCP server (offers, Acks) will only be accepted if they arrive on a trusted port. Server-type DHCP packets received on an untrusted port are dropped.

In AlliedWare Plus, all ports are untrusted by default, and trusted ports need to be explicitly configured as trusted:

```
awplus(config)# int port1.0.24
awplus(config-if)# ip dhcp snooping trust
```

Server
exhaustion
attacks

There are two ways in which DHCP snooping defends against server exhaustion attacks:

1. Per-port lease limiting

Simply putting a limit on the number of leases that DHCP snooping allows to be allocated to workstations downstream of each untrusted port is effective in limiting the damage of exhaustion attacks. By default, DHCP snooping allows only one lease to be allocated per untrusted port.

This limit can be reconfigured on a per-port by the command:

```
awplus(config)# int port1.0.12
awplus(config-if)# ip dhcp snooping max-bindings <number of leases>
```

Once the number of leases downstream of a port has reached that port's limit, further requests from clients on that port are dropped.

2. MAC consistency checking

As mentioned above, lease limiting can reduce the damage of an exhaustion attack. But if a port has been configured for multiple leases, it would still be possible for an attacker on that port to steal up to that many leases.

Even more effective an approach is to recognise and drop packets that are part of an exhaustion attack attempt.

Commonly, in an exhaustion attack the client ID MAC address within a DHCP request packet will be different to the source MAC address of the packet. This is because the client ID is the parameter that the server looks at to decide whether the request is from a new client or from one that already has a lease. This forces the exhaustion attacker to use a series of different client IDs in its requests. But if it keeps changing the source MAC address of its requests, it will likely be trapped by port security which limits the number of MACs that can be learnt on a given port.

Unless it has passed through a DHCP relay, a valid DHCP request should always have the client ID MAC address equal to the packet's source MAC—it has no reason for them to be different.

As part of blocking server exhaustion attacks, DHCP snooping can be configured to drop requests in which the source MAC differs from the client ID MAC address.

This check is on by default and can be disabled with the command:

```
awplus(config)# no ip dhcp snooping verify mac-address
```

Note: This feature should not be used when the DHCP Snooper is upstream of a DHCP Relay, since the source MAC address of the packet does not match the client hardware address in the DHCP packets once they have been through a DHCP Relay.

DHCP Option 82 information

- DHCP Relay Agent Information Option 82 is an extension to DHCP and is defined in RFC 3046 and RFC 3993.
- DHCP Option 82 can be used to send information about DHCP clients to the authenticating DHCP server.
- DHCP Option 82 will identify the VLAN number, port number and, optionally a customer ID of a client, during any IP address allocation. When DHCP Option 82 is enabled on the switch, it inserts the above information into the DHCP packets as they pass through the switch on their way to the DHCP server.

The DHCP server can then store this information in a log that will provide an audit trail of not just the MAC address but also the network location (which port on which switch) of the workstation to which each lease is allocated.

Although Option 82 is titled the DHCP Relay Agent Information Option, the device that inserts the Option 82 information into a DHCP packet does not have to be acting as DHCP relay agent. A Layer 2 switch can insert the Option 82 information into the DHCP packets if snooping is enabled.

Really, the Option 82 information needs to be inserted into the DHCP packets by a switch at the edge of the network. This is because only the edge switch knows the information that uniquely identifies the subscriber that the IP address was allocated to. It is quite likely that the edge switch will be a Layer 2 switch rather than a DHCP relaying Layer 3 switch. Therefore, it makes sense for Layer 2 switches to be able to insert Option 82 information into DHCP packets.

Format of the Option 82 field

In the DHCP packet, the Option 82 segment is organized as a single DHCP option containing one or more sub-options that convey information known by the relay agent. The format of the option is shown below:

Code	Len	Agent Information Field					
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
82	N	i1	i2	i3	i4		iN
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

Each of the sub-options within the DHCP option are constructed as follows:

```

SubOpt Len      Sub-option Value
+---+---+-----+-----+-----+-----+-----+-----+
| X | N | s1 | s2 | s3 | s4 | | sN |
+-----+-----+-----+-----+-----+-----+-----+

```

The following table shows a list of the sub-options that are used for identifying the subscriber that the IP address was allocated to:

Sub-option	RFC	Description
1	RFC3046	Agent Circuit ID sub-option—used for defining the switch port and VLAN number of the port the client is attached to.
2	RFC3406	Agent Remote ID sub-option —used for defining the MAC address of the switch that added the Option 82 information.
6	RFC3993	Subscriber ID sub-option —can define client name, or switch name.

The RFCs do not specify the content of the sub-option values. These are just strings that the implementation can fill in any way they wish.

Example Packet

The following shows an extract of a DHCP Request packet that includes Option 82 details:

```

DHCP Message Type = DHCP Request

Option 82 - Agent Information (Option)
0000: 52 20 01 06 00 04 00 30 00 05 02 08 00 06 00 00 R .....
0010: CD 11 B2 52 06 0C 55 73 65 72 49 64 30 31 32 33 ...R..UserId012345
0020: 34 35

```

The following table provides an analysis of the strings in the above DHCP Request packet extract:

String Extract	Sub-option
<i>Green Italic</i>	Agent Circuit ID (sub-option 1)
Blue bolded	Agent Remote ID (sub-option2)
Red	Subscriber ID (sub-option 6)

The Agent circuit ID string **00 30 00 05** translates as:

HEX value 30 = vlan48

HEX value 05 = switch port 5

You can see that the Remote ID is just MAC address 0000.cd11.b252. The subscriber ID is a user-defined string “UserId012345”

Option 82 insertion in AlliedWare Plus DHCP snooping

By default, as soon as a switch is configured for DHCP snooping AlliedWare Plus will insert Option 82 information into DHCP packets that are received on untrusted ports.

The feature can be turned off and on by the command:

```
awplus(config)#(no) ip dhcp snooping agent-option
```

No subscriber ID sub-option is included in the Option 82 field inserted into the DHCP packets received on a port until the subscriber-ID value is configured for that port.

```
awplus(config)# interface port1.0.3
```

```
awplus(config-if)# ip dhcp snooping subscriber-id room_534
```

The remote ID sub-option will always be present in the Option 82 information that AlliedWare Plus inserts into DHCP packets. The default choice for the remote ID is the MAC address of the switch that is inserting the Option 82 information. The remote ID can, instead be set to any string you wish, by using the command:

```
awplus(config-if)# ip dhcp snooping agent-option remote-id  
    <remote ID>
```

This command is entered in interface mode for VLAN interfaces. You can configure different remote IDs to be inserted into DHCP packets received on different VLANs.

Filtering packets based on information gathered by DHCP snooping

A major contribution that DHCP snooping makes to network security is that the information stored in the DHCP snooping database can be used for intelligent, adaptive, packet filtering.

The packet filtering can be divided into two distinct types:

- Filtering IP packets based on source address—known as IP source guard.
- Filtering ARP packets based on sender address—known as Dynamic ARP inspection

Let us consider these two types of filtering in detail.

IP source guard

In a network in which IP addresses are allocated to workstations by DHCP, the DHCP snooping database in any given switch is an up-to-date record of the IP addresses on the workstations downstream of that switch. What is more, the database also records the ports those workstations are connected to. The switch thereby knows the source IP addresses that should be on the packets arriving on any workstation-directed ports. Workstations have no good reason to send packets with source IP addresses other than their allocated addresses. Packets that arrive with anything other than the source IP addresses that the DHCP snooping database believes should be seen arriving on their ingress can be treated as suspicious and dropped.

This filtering system is still workable even if there are some workstations, printers, etc. in the network that are configured with static IP addresses. Those devices can be covered by adding static entries into the DHCP snooping databases of the switches to which they are connected.

Hardware filtering

In Allied Telesis switches running AlliedWare Plus, the IP source guard filtering is performed in hardware, so it has no impact on forwarding performance.

From the configuration point of view, the key is to create hardware access lists that use the keyword **dhcp snooping** for the source IP address. An ACL that allows in packets that are sourced from IP addresses that are in the DHCP snooping database, and drops all other IP packets is configured as follows:

```
awplus(config)# access-list hardware acl1
awplus(config-ip-hw-acl)# permit ip dhcpsnooping any
awplus(config-ip-hw-acl)# deny ip any any
```

This ACL can then be applied to a set of ports:

```
awplus(config)# int port1.0.1-1.0.23
awplus(config-if)# access-group acl1
```

The keyword **dhcp snooping** is a place holder that will cause the switch to initially create a hardware filter with 0.0.0.0 in the source IP field. That 0.0.0.0 address can be replaced by a real IP address when DHCP snooping learns that an IP address has been allocated to a client downstream of a port that the ACL is applied to.

Of course, as discussed above, it is possible to have multiple workstations downstream of a given port. To support this case, the switch will attach multiple copies of the hardware filter to the port. For each port, the number of copies of the filter attached to the port is equal to the value of the max-bindings parameter on the port.

Dynamic ARP Inspection

In AlliedWare Plus, the Dynamic ARP inspection process is referred to as “ARP security”. Quite simply, when it is enabled, ARP security ensures that ARP packets received on untrusted ports are only permitted if they originate from an IP and MAC address from the DHCP snooping database.

The feature is configured on a per-VLAN basis.

```
awplus(config)# int vlan1
awplus(config-if)# arp security
```

Basic DHCP snooping configuration in AlliedWare Plus

Up until now, we have been looking at the features and capabilities of DHCP snooping. The features have been illustrated with examples of how they are configured on AlliedWare Plus, but we do not yet have an example of a fully operational DHCP snooping configuration.

Let us now build up a basic DHCP snooping configuration.

1. Enable the DHCP Snooping service.

```
awplus(config)# service dhcp-snooping
```

2. Configure port(s) connected to the DHCP server as trusted.

```
awplus(config)# int port1.0.24
awplus(config-if)# ip dhcp snooping trust
```

3. Configure ports 1-23 for DHCP snooping and allocate 5 clients per port.

```
awplus(config)# int port1.0.1-port1.0.23
awplus(config-if)# ip dhcp snooping max-bindings 5
```

4. Enable DHCP snooping on the VLAN itself.

```
awplus(config)# int vlan1
awplus(config-if)# ip address 192.168.50.254/24
awplus(config-if)# ip dhcp snooping
```

From there, you can proceed on to configure features like IP source guard, dynamic ARP inspection, checking of the match between DHCP packets' source MAC and client identifier MAC, remote-ID and subscriber-ID for Option 82, etc.

Monitoring using SNMP

It is possible to use SNMP traps to inform you of security violations related to DHCP snooping.

The violations that can generate SNMP traps are:

- Invalid BOOTP packet
- Invalid DHCP ACK
- Invalid DHCP release
- Max bindings on a port exceeded—attempt to add another database entry for the port when the number of entries for that port has already reached the port's max-bindings
- Insertion of Option 82 failed
- Invalid Option 82 info received
- Option 82 received on untrusted port
- BOOTP reply received on untrusted port
- Source MAC does not match client ID MAC inside the DHCP packet

The switch can be configured to send traps for these events as follows:

```
awplus(config)# int port1.0.10-1.0.20
awplus(config-if)# ip dhcp snooping violation trap
```

SNMP traps can also be sent for ARP security violations:

```
awplus(config)# int port1.0.10-1.0.20
awplus(config-if)# arp security violation trap
```