Mobility Task Force

Deliverable H

Testbed and reference design for inter-NREN roaming

Version:1.0

Editor: Klaas Wierenga (SURFnet)

Contributors: Paul Dekkers (SURFnet)
Hansruedi Born (SWITCH)
Sami Keski-Kasari (FUNET)
James Sankar (UKERNA)
TF Mobility Group

Table of Contents

1 Abstract

Wireless LAN technology (WLAN, according to IEEE 802.11*) is already an important part of the network access infrastructure at universities and research institutions connected to National Research and Education Networks (NREN's) in Europe.

The TERENA Task Force Mobility Group (TF-Mobility group) has assessed national solutions that enable transparent Wireless LAN access for nomadic wireless users at different locations on a national level and has identified the requirements and additional work required to scale each national solution to a European level.

In deliverable G[1], the "Preliminary selection for inter-NREN roaming" concluded that there will be no single national solution recommended as the European model. This is because no one solution outperforms, each solution has a number of strengths and weaknesses making such choices difficult. In addition there has been a considerable investment made to develop a variety of national solutions and there is a low likelihood that NREN's will abandon their solutions in favour of a European model. This deliverable concentrates on the interoperability issues between the national solutions; it describes interoperability test beds at institutions that have implemented one of the national roaming solutions and details to work undertaken to provide support for guest access based on the other roaming solutions.

Furthermore a reference implementation is presented on guest access for 802.1X-, Web-based redirection and VPN-based authentication.

The intended audience for this document are system administrators that want to implement a wireless infrastructure that allows for roaming access and all those who want to understand the issues and possible solutions involved in doing so .

This is a TF-Mobility group document.

The TF-Mobility group Terms of Reference are available at
http://www.terena.nl/tech/task-forces/tf-mobility.

A table of planned deliverables is available at:
http://www.terena.nl/tech/task-forces/tf-mobility/docs/DelList.pdf

This document follows the guidelines of the TF-Mobility group glossary of non-technical terms as outlined in
http://www.terena.nl/tech/task-forces/tf-mobility/Deliverables/DelB/DelB_v1-3-5.pdf

[1] http://www.terena.nl/tech/task-forces/tf-mobility/Deliverables/DelG/DelG-final.pdf

2 Introduction

The three main authentication solutions (802.1X, VPN and web-based redirection) have different characteristics that need to be taken into account when creating a interoperable solution.

802.1X based authentication is inherently different from Web based redirection and VPN-based authentication due to different demands on the wireless LAN networks. 802.1X enabled wireless networks require an encrypted channel (with dynamic WEP-keys), whilst the two other mechanisms are based on the concept of open, unencrypted access to the docking network. The result of these two contradicting requirements is that in order to support both types of network authentication two logically separated networks on the radio layer need to be constructed. In the case of an access point it means that the access point must be capable of using multiple SSID's and can support VLAN assignment, an example can be seen from the following docking network configuration:

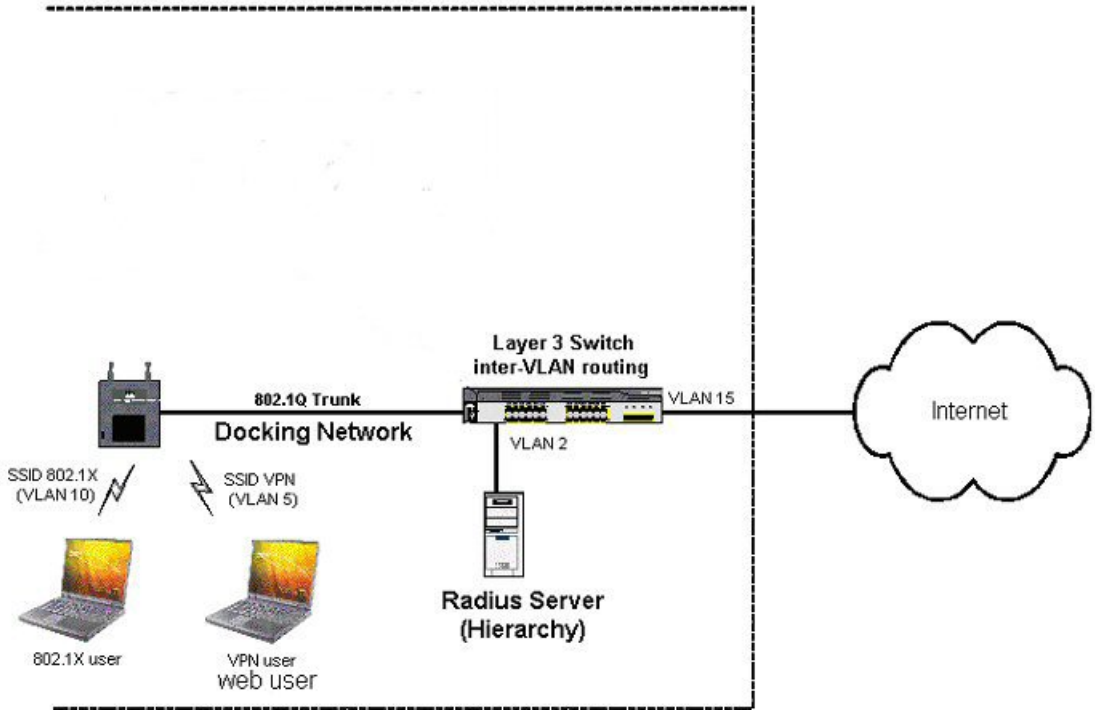


Figure 1: Network lay-out with multiple SSID's and VLAN assignment

It should be noted that is impossible in the vast majority of the current generation of access points to broadcast both configured SSID's at the same time.

Where it is imperative that all SSID's are visible (broadcasted) or when the access points are not capable of assigning VLAN's, an alternative solution is to create an infrastructure that is separated on layer 1, an example configuration is as follows:

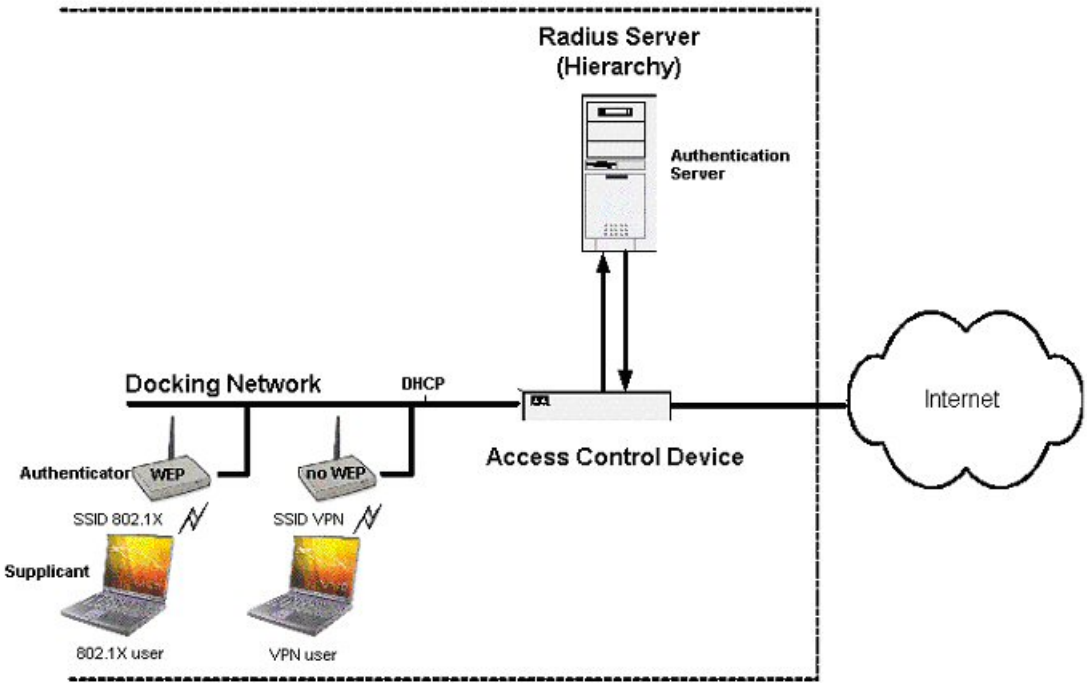


Figure 2: Network lay-out without multiple SSID's and VLAN assignment

For the remainder of this document the first setup will be assumed.

The Web-redirection and the 802.1X based approach share the use of a European scale RADIUS-hierarchy to forward user credentials for authentication at their home institution.

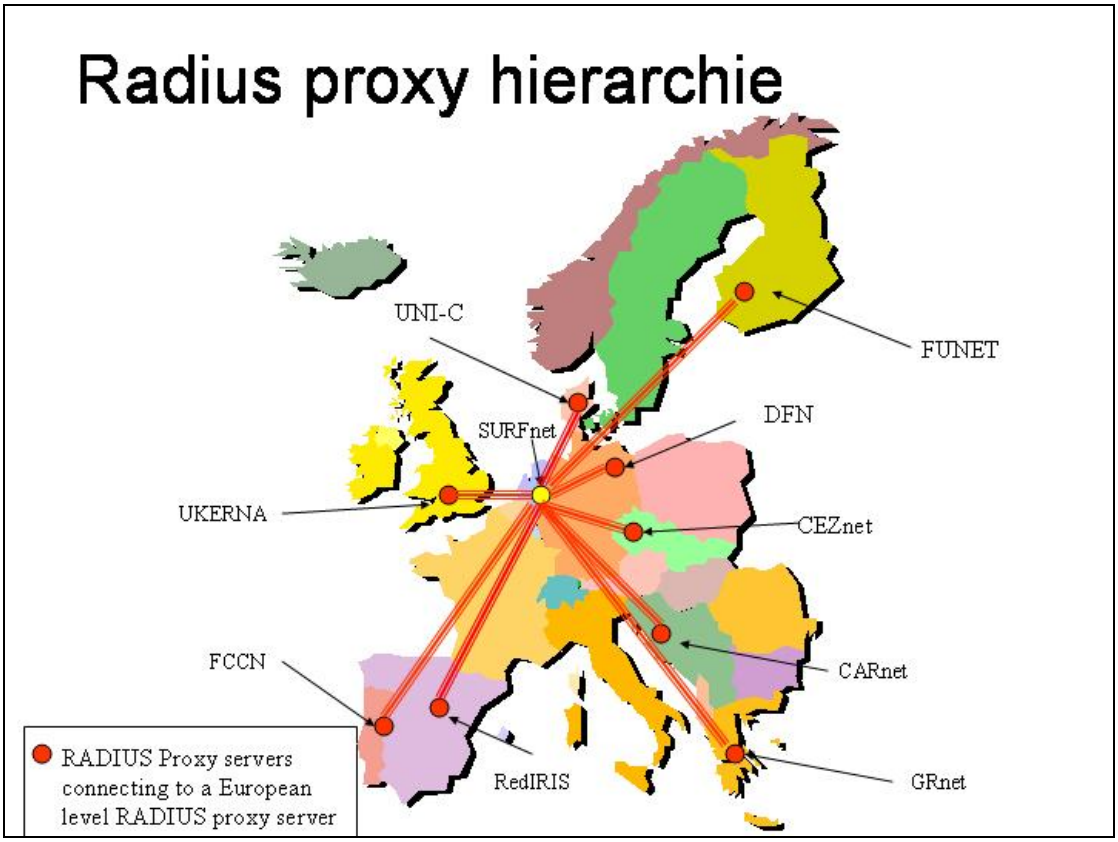


Figure 3: European proxy hierarchy (june 2004)

The guest users will enter their credentials that will be forwarded, based on the realm, through this RADIUS proxy hierarchy to their home institution RADIUS-server for authentication requests to be processed and the result forwarded back to the visited institution so that if successful will result in network access granted according to local policy. This European infrastructure is currently in place and a number of NRENs have built national RADIUS proxy hierarchies. For the remainder of this document it is assumed that this RADIUS proxy hierarchy infrastructure is in place.

For the VPN-based approach in order to make it possible for guests to access their home VPN-concentrator a block of IP address space has been allocated based on the Controlled Address Space for VPN Gateways (CASG) concept that has been discussed in detail in the TF-Mobility deliverable G.

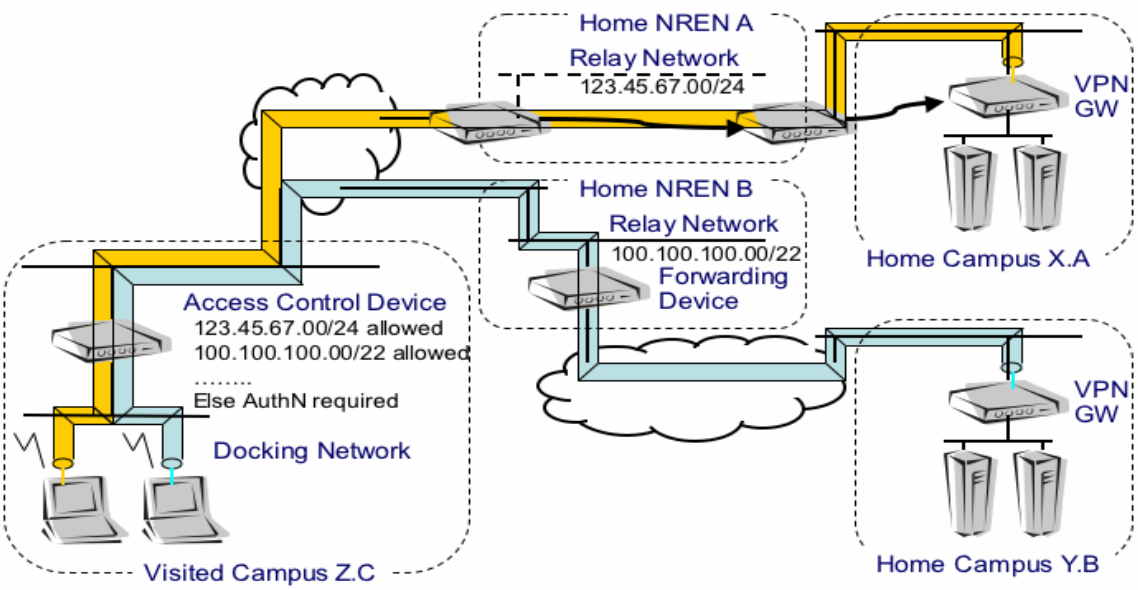


Figure 4: Address Space for VPN Gateways

It is assumed that guest users get restricted network access with their VPN-client to their home institution's VPN-server as long as it uses an IP-address from the CASG. For the remainder of this document it is assumed that the CASG is in place.

3 Testbed design for national roaming solutions

In order to design and document the reference architecture, three local test beds had to be created, to represent each of the major national roaming solutions in use today. These solutions are at SURFnet in the Netherlands (802.1X), at the Technical University of Tampere in Finland (Web-based redirection) and at SWITCH in Switzerland (VPN-based). A test bed was created to attempt to extend support to guest users from one of the other two solutions or a guest user from another NREN that was using a similar method deployed locally.

3.1 Test bed at 802.1X based docking network

Configuration

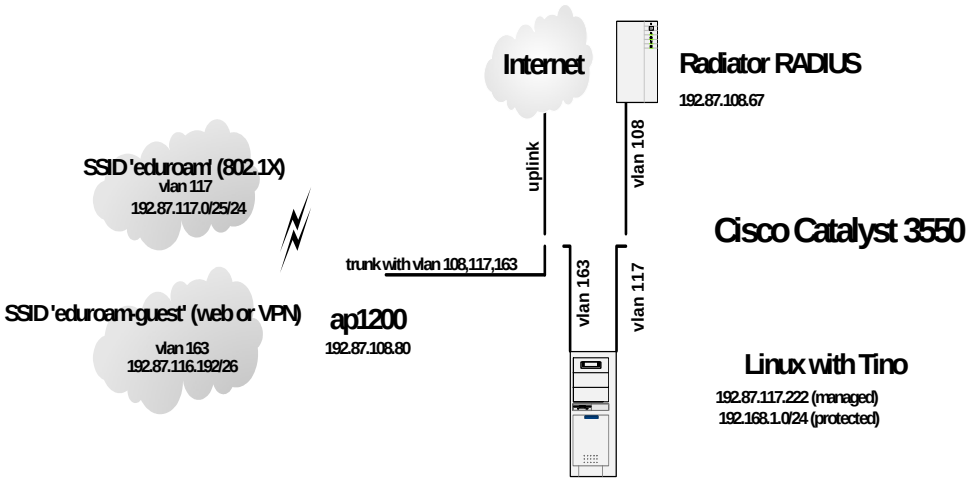


Figure 5: 802.1X based docking network

SURFnet uses 802.1X as the default authentication method, for this purpose a nationwide roaming service with the SSID broadcasted name “eduroam” has been launched to associate with a wireless network. This means that guest use by other 802.1X users is possible without any additional configuration. In order to support Web and VPN users a captive portal (created by Tino) has been implemented as well as an extra, non-802.1X based, wireless network.

The docking network contains access points that are capable of handling multiple SSIDs: SSIDs are mapped to VLANs (Table 1). Access points are connected to a layer 3 switch, which is responsible for inter-VLAN routing. VPN and Web users connected to the SSID broadcasted name “eduroam-guest” are granted (unauthenticated) access to VLAN 163. All sessions to destinations outside this VLAN are however intercepted by a session intercept device using Tino[2]. VPN sessions with a destination address that lies within the CASG address space are routed through. Web-based authentication

[2] http://www.cc.puv.fi/~teu/tino/

users need to fill in their user credentials on a webpage that is generated by Tino which are checked against the RADIUS-hierarchy, again, upon proper authentication all traffic is routed through.
802.1X users connected to the SSID “eduroam” are granted access to the Internet (and possibly to a number of local services to the discretion of the visited institution) upon successful authentication over the RADIUS-hierarchy.

SSID	VLAN-ID	Purpose	Parameters
	108	VLAN for access points, RADIUS server	
eduroam-guest	163	VPN and web user	Open, no WEP, DHCP
eduroam	117	802.1X user	802.1X, WEP, DHCP

Table 1: VLAN configuration

802.1X User at SURFnet

Depending on the operating system and the 802.1X client in use, the following procedure may differ slightly. In general an 802.1X user who wants to connect to the SURFnet docking network has to

- Connect their device to the docking network by associating with the broadcasted SSID “eduroam”.
- Logon to the 802.1X network (802.1X client sends credentials, after successful authentication the device receives IP configuration via DHCP).
- Depending on the 802.1X authentication credentials users are placed in a particular (employee, guest etc.) VLAN

VPN User at SURFnet

A VPN user who wants to connect to the SURFnet docking network has to

- Connect their device to the docking network by associating with the broadcasted SSID “eduroam-guest” to automatically receive an IP address and configuration via DHCP),
- Initiate their VPN client and login at the home institutions VPN server (access is granted to resources at the home institution as well as to the Internet (via home institution) via an encrypted tunnel).

Web User at SURFnet

A web user who wants to connect to the Internet at SURFnet has to

- Connect their device to the docking network by associating with the broadcasted SSID “eduroam-guest” to automatically receive an IP address and configuration via DHCP),
- Start a web browser, type in the details of any web page and the user will be automatically redirected to a login web page to enter their user credentials (username/password) and have access to additional information about the roaming system at the visited institution,
- Once the guest user has entered their credentials and successful authentication has occurred access to the Internet (and possibly some local services) is granted either for a specific time period or until a timeout occurs).

3.2 Test bed at VPN based docking network

Note: actual IP-addresses have been changed into 111.222.x.y

Configuration

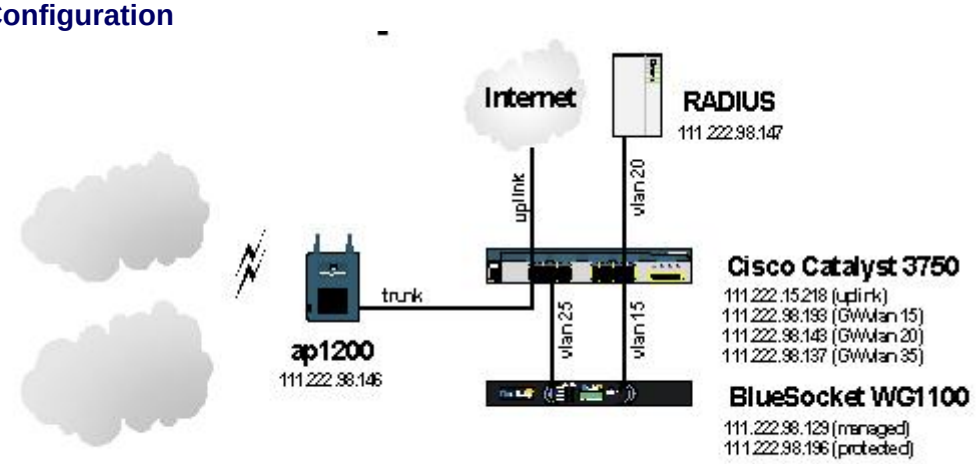


Figure 6: VPN based docking network

SWITCH has chosen a VPN-based approach for WLAN authentication. In order to support also 802.1X- and Web-based authentication a RADIUS-server (Radiator), a captive portal (BlueSocket) as well as an extra, 802.1X capable, wireless network have been implemented.

The docking network contains access points capable of handling multiple broadcasted SSIDs that are mapped to VLANs (Table 2). The access points are connected to a layer 3 switch, which is responsible for inter-VLAN routing. VPN users connected to SSID “web-vpn” are granted (unauthenticated) access to addresses in the CASG address range, implemented with an ACL on the captive portal (BlueSocket). 802.1X users connected to the SSID “802.1X” are then granted access to the Internet upon successful authentication via a RADIUS proxy hierarchy. The sessions of Web users are intercepted by the BlueSocket device and users are redirected to a login web page to enter their credentials. Upon proper authentication at their home institution via the RADIUS-proxy hierarchy, the user granted Internet access according to visited institution policy.

SSID	VLAN-ID	Purpose	Parameters
	20	VLAN for access points, RADIUS server	
web- vpn	25	VPN and web user	Open, no WEP, DHCP
802.1X	35	802.1X user	802.1X, WEP, DHCP

Table 2: VLAN configuration

802.1X User at SWITCH

Depending on the operating system and the 802.1X client in use, the following procedure may differ slightly. In general a 802.1X user who wants to connect to the SWITCH docking network has to

- Connect their device to the docking network by associating with a broadcasted SSID “802.1X” (VLAN 35),
- The user can then logon to the 802.1X network (the 802.1X client sends credentials, after successful authentication notebook receives an IP address and configuration via DHCP).

VPN User at SWITCH

A VPN user who wants to connect to the SWITCH docking network has to

- Connect their device to the open docking network with SSID “web-vpn” (VLAN 25) and automatically receive IP configuration via DHCP.
- Start their VPN client and login at the home institution (access is granted to resources at the home institution as well as to the Internet (via home institution) via an encrypted tunnel).

Web User at SWITCH

A web user who wants to connect to the Internet on the SWITCH docking network has to

- Connect their device to the open docking network with SSID “web-vpn” (VLAN 25) and automatically receives IP address and configuration via DHCP.
- Start a web browser, access any web page and is then automatically redirected to a login web page to enter their user credentials. The page also contains additional information about the roaming system at the visited institution),
- Once the guest user has entered their credentials and successful authentication has occurred access to the Internet (and possibly some local services) is granted either for a specific time period or until a timeout occurs).

3.3 Test bed at Web redirection based docking network

Note: actual IP-addresses have been changed into 111.222.x.y

Configuration

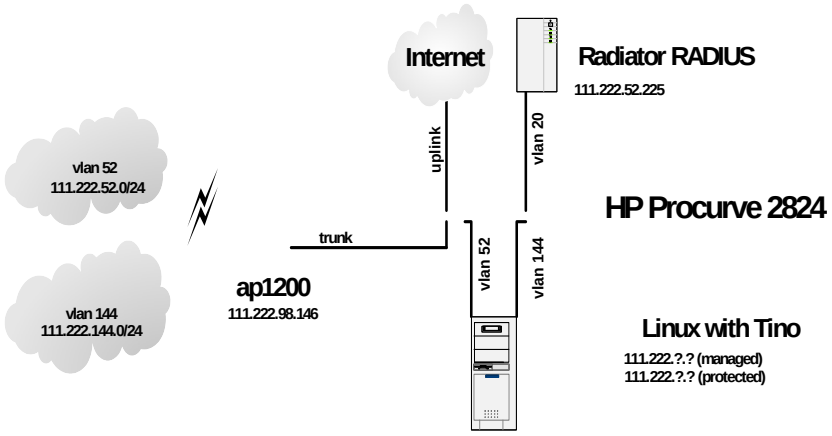


Figure 7: Web redirection based docking network

At the Technical University of Tampere (TUT) the preferred method of access to the wireless LAN is by using a captive portal (Tino) that authenticates against FUNET's RADIUS proxy hierarchy. Therefore to allow for guest users who usually use Web-based authentication at their home organisation, no extra effort is needed. In order to support also 802.1X and VPN users an extra, 802.1X-enabled, wireless network has been created and the configuration of the switch has been altered to allow for VPN-traffic pass through to addresses in the CASG address range.

The docking network contains access points capable to handle multiple broadcasted SSIDs that are mapped to VLANs (Table 3). Access points are connected to a layer 3 switch, which is responsible for inter-VLAN routing. Sessions of web users connected to SSID "web-vpn" are intercepted by Tino and upon proper authentication against the RADIUS-hierarchy routed through. VPN-users connected to SSID "web-vpn" are granted (unauthenticated) access to addresses in the CASG address range, implemented with an ACL on the layer 3 switch. 802.1X users connected to the SSID "802.1x" are granted access to the Internet upon successful authentication against the RADIUS-hierarchy.

SSID	VLAN-ID	Purpose	Parameters
	52	VLAN for access points, RADIUS server	
Web-vpn	144	VPN and web user	Open, no WEP, DHCP
802.1x	152	802.1X user	802.1X, WEP, DHCP

Table 3: VLAN configuration

802.1X User at TUT

An 802.1X user who wants to connect to the Internet at the TUT docking network has to

- Connect their device to the docking network by associating with the broadcasted SSID “802.1x”.
- The user logs onto the 802.1X network automatically as the 802.1X client sends credentials to authenticate. After successful authentication the device receives an IP address and configuration via DHCP and placed into VLAN 52 to be granted either for a specific time period or until a timeout occurs).

VPN User at TUT

A VPN user who wants to connect the TUT site has to

- Connect their device to the docking network by associating with the broadcasted SSID “web-vpn”, the device is then placed into VLAN 144, and automatically receives an IP address and configuration via DHCP,
- The user starts their VPN client and enter their credentials at the home institution (access is granted to resources at the home institution as well as to the Internet via home institution).

Web User at Remote FUNET docking network

A web user who wants to connect to the Internet at another FUNET docking network has to

- Connect their device to the docking network by associating with the broadcasted SSID “web-vpn”, the device id then placed into VLAN 144, and automatically receives an IP address and configuration via DHCP),
- The user opens their web browser and attempts to access a web page. The browser automatically redirects to a login web page for the guest user to enter their credentials and have access to additional information about the roaming system at the visited institution,
- Once the user has successfully entered their credentials at the web page and has been authenticated, the device is placed in VLAN 52 and access to the Internet (and possibly some local services) is granted either for a specific time period or until a timeout occurs).

4 Reference design

4.1 Introduction

In this chapter the setup at SURFnet will be documented in detail to provide a reference architecture. The network infrastructure described supports 802.1X- or Web-based authentication for both local and guest users as well as a method to authenticate VPN guest users by allowing to connect to a VPN-concentrator in the CASG address space range.
The configuration of the RADIUS-server, the switch, the access points and the captive portal will also be explained.

4.2 Layer 3 network design

Figure 8 shows the layer 3 topology of the SURFnet setup.

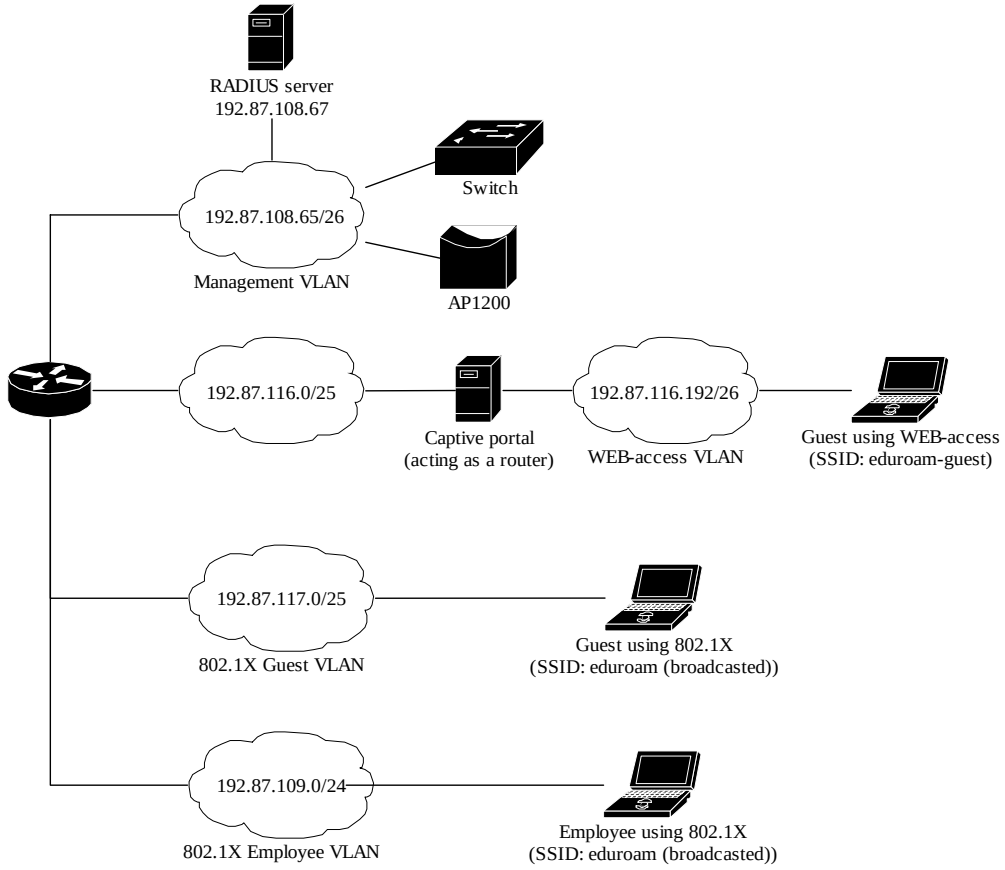


Figure 8: Layer 3 design of reference architecture

4.3 Layer 2 design

The layer 3 design is implemented using the following layer 2 setup.

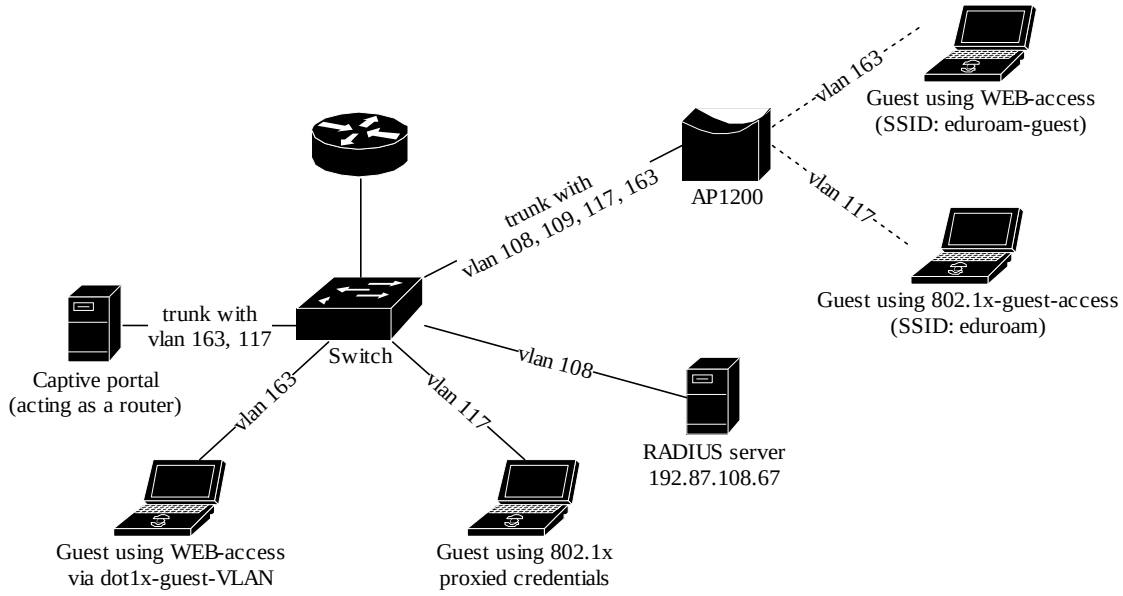


Figure 9: Layer 2 design of reference architecture

4.4 RADIUS configuration

RADIUS is used as authentication mechanism that can be used for 802.1x and Web-based redirection network access (the latter using a captive portal) and therefore has an important role in the network development. Without the RADIUS server no authentication could be done. As a result, choosing a suitable RADIUS product was SURFnet's first priority. SURFnet decided to choose Radiator for its RADIUS server software because it had powerful filtering mechanisms, flexible configurations, and support for multiple authentication methods.

In order to terminate EAP-authentication a SSL certificate is required. (In most cases there is already a PKI in place, so it can be assumed that this is not a problem.) Depending on the EAP-type used, client certificates may also be needed.

In the following examples there are two kinds of EAP that were configured at SURFnet. Firstly, EAP-TLS based on client-certificate authentication was established for SURFnet employees. Secondly, EAP-types based on EAP-TTLS and EAP-PEAP where these do not require client certificates but traditional mechanisms such as username/password authentication for non-SURFnet employees.

If the RADIUS server is just used for proxy-ing EAP-requests to another server and not for terminating EAP, the configuration can do so without the need for a SSL-certificate.

Clients

RADIUS is based on a client-server model. The NAS-devices (Access Points, switches etc.) forwards credentials to a RADIUS server but can also act as a client, and therefore need to be defined on the RADIUS server. Other RADIUS servers can act as a client as well, so every kind of RADIUS-request can be forwarded to another server.

The clients are configured within Radiator using the <Client>-clause.

```
<Client 192.87.108.80>
Secret 6.6obaFkm&RNs666
        Identifier ACCESSPOINT1
        IdenticalClients 192.87.108.75
</Client>
```

In this example there is a client definition for 192.87.108.80, an Access-Point. The secret is a series of (at best 16) characters that are used to encrypt the credentials sent in the RADIUS-request.

It is of course recommended to create a secret that can not be easily guessed since otherwise the RADIUS-message can be decrypted. This is not a big problem with EAP-authentication using 802.1X since the credentials are also transmitted over a SSL-encrypted tunnel between the client and the final

authentication server, but especially with regular credentials like the ones used with Web-based redirection authentication this is sensitive information that might be captured, therefore a reasonably complex secret and a SSL tunnel? is recommended.

The Identifier in the Client-definition can be used further on in the Radiator-configuration to filter on a specific request.

If there is more then one Client that should use this same secret and identifier definition, the IdenticalClients statement can be used. If there are really many clients with different IP-addresses, there is also the possibility for a “catch-all” client. This is the default-client that is used after all other client-definitions didn’t match. Define this client as:

<Client DEFAULT>

If this kind of configuration is used, it is worth filtering with firewall-rules on RADIUS packets. There are only a few places where a RADIUS-request should come from: the management VLAN with the NAS-devices (switches and access-points), and the RADIUS infrastructure where unknown requests can be sent to.

Realms and VLAN assignment

The processing of authentication of accounting requests is done by linear processing of the present <Realm>- or <Handler>-clauses in the Radiator configuration file. Handler-clauses are more potent than Realm clauses in terms of filtering things besides realms and are therefore the preferred method. A *realm* is the part behind a username to indicate the origin of a user. With RADIUS, the username is usually separated from the realm with a “@” so the complete username looks like a regular e-mail address.

A <Handler>-clause is terminated with a </Handler>.

Within a Handler many mechanisms can be configured that define what to do with the RADIUS request.

PROXY example

The most simple Handler is the definition for proxy-ing the request to another server using the “AuthBy RADIUS” definition within this Handler.

In this example a proxy-configuration is shown. First we have a Handler that matches on any request, as long as it does not come from the client with the identifier “Proxy-Identifier”. This is to prevent a proxy loop. When a request comes from a proxy-server, we should never forward it back to that proxy-server.

Another important part is the hostname to forward the request to. Multiple hostnames can be defined here for redundancy reasons: if the first host does not respond within 3 seconds, the second one is tried instead. The UDP-ports

to forward the RADIUS-request to can be defined in this “AuthBy RADIUS” mechanism as well.

```
<Handler Client-Identifier=/^(?!Proxy-Identifier$)/>
        <AuthBy RADIUS>
                Host 192.87.36.3
                Secret super_secret!
                AuthPort 1812
                AcctPort 1813
                StripFromReply Tunnel-Type, Tunnel-Medium-Type, \
                        Tunnel-Private-Group-ID
                AddToReply Tunnel-Type=1:VLAN,Tunnel-Medium-Type=1:Ether_802, \
                        Tunnel-Private-Group-ID=1:117
        </AuthBy>
</Handler>
```

It is good to know that for a “Host” both the IP-address and FQDN can be used. It’s more or less a personal preference of the RADIUS administrator with one remark that the hostnames are only looked up once at the Radiator (re)start. If the lookup fails, the Host cannot be used until the next restart. This can be tricky with a power outage, where for instance the DNS server is not yet available when Radiator already is.
While using hostnames one benefits from the administrative ease when an IP-address is changed. In that case, you only (or still) need a restart of the RADIUS server.

The last part in this <AuthBy RADIUS>-definition shows the addition of RADIUS-attributes to the RADIUS-response. These attributes can be used to define a VLAN that should be assigned to users that authenticate using this Handler. With StripFromReply the attributes that came from the proxy-server are stripped first to prevent malicious VLAN-assignments, afterwards the attributes are added with the proper values for the local network design. In this case, VLAN 117 should is used for guests.

Secure authentication with EAP-TLS

For SURFnet employees the EAP-TLS mechanism is used. Since there is already a PKI in place whereby every employee has a client certificate (to access for instance e-mail or the internet from anywhere outside the office) this was the most secure and easiest mechanism to use (since it is also by default available within Windows XP and Windows 2000).

In this example the AuthBy-definition is outside the Handler, and is referred to using the Identifier. (This is usefull if the AuthBy-definition is reused in another Handler for instance.)

```
<AuthBy FILE>
    Identifier ID4-SN-TLS
    Filename %D/surfnet-employees
    EAPType TLS
    EAPTLS_CAFile %D/cert/surfnet-ca-chain.pem
    EAPTLS_CertificateFile %D/cert/sn-radius-server-cert.pem
    EAPTLS_CertificateType PEM
    EAPTLS_PrivateKeyFile %D/cert/sn-radius-server-key.pem
    EAPTLS_PrivateKeyPassword (the secret that secures the private-key)
    EAPTLS_MaxFragmentSize 1024
```

```
        AutoMPPEKeys
        SSLeayTrace 1
        StripFromReply Tunnel-Type,Tunnel-Medium-Type,Tunnel-Private-Group-ID
        AddToReply Tunnel-Type=1:VLAN,Tunnel-Medium-Type=1:Ether_802,Tunnel-Private-Group-
ID=1:109,User-Name=%u
</AuthBy>
```

In this AuthBy-clause there is a file defined (surfnet-employees) in which every employee is listed. This way, we control the users that can access the infrastructure using EAP-TLS.

The next definitions define what to do with the EAP-request. First the “EAPType TLS” limits the use of this AuthBy-definition for TLS-only. We do not want regular password authentication here, just certificates. Next, the certificate files are configured and the secret that secures the private-key file can be provided. If there is no secret for the private key, this can be omitted.

The next part defines in what chunks the EAP-messages should be fragmented. Since some parts of the EAP-TLS challenge (which is just a normal Challenge like the ones used for securing a website) are to big to fit in a RADIUS request the packets should be fragmented.

The MPPE-keys portion is important for wireless access. With 802.1X, encryption is done at the edge of the network, between the Access-Point and the client. To provide this secure encryption, a unique key is created and encrypted using the MPPE-keys that are derived from the SSL-challenge. This is something that can both be done at the Access-Point and the Client end so that there is no need to transfer the WEP-key in plain text over the air. This, plus the fact that the key can be rotated within a period defined by either the Access-Point or the RADIUS server provides 802.1x users with a good level of security.

The last part of the AuthBy-definition shows again how to assign a proper VLAN.

```
<Handler Realm=surfnet.nl, EAP-Message=/.+/>
        AuthBy ID4-SN-TLS
</Handler>
```

The Handler above shows the referral to the AuthBy-definition and some filtering mechanisms to filter out the proper requests. If more things need to be filtered on, they can be added to this handler, as follows:

```
NAS-Port-Type=/^(Wireless-IEEE-802-11|Ethernet)$/
```

In this way, only requests with the proper NAS-Port-Types are allowed.

For Accounting purposes, a new handler should be defined in this case, that filters on

```
“Request-Type=Accounting-Request”
```

since the request does not match the Handler that filters on the EAP-Message.

EAP-TTLS or EAP-PEAP

When there is no PKI available and username/password authentication fits the requirements the EAP-mechanisms PEAP and TTLS can be used.

These two mechanisms look the same, they both set up an SSL tunnel on which the credentials can be transported. In the case of PEAP this is another EAP-method in an encrypted tunnel, while when using TTLS another RADIUS type is used that is more flexible because this enables the use of PAP-authentication based on a plain-text username and password, transported over a SSL-tunnel. PAP credentials can be verified with almost any backend. EAP-PEAP however uses EAP-MSCHAP for tunnelled authentication, and therefore requires a specific backend, like Active Directory.

When there is just a file with plaintext usernames and passwords (for guests for instance) both PEAP and TTLS can be used with the same realm.

Instead of a flat file a more flexible backend for user accounts is a database like MySQL, or LDAP.

```
<Handler TunnelledByPEAP=1, Realm=guest.surfnet.nl>
        <AuthBy FILE>
                Filename %D/peap-users
                EAPType MSCHAP-V2
        </AuthBy>
</Handler>

<Handler TunnelledByTTLS=1, Realm=guest.surfnet.nl>
        <AuthBy FILE>
                Filename %D/ttls-users
        </AuthBy>
</Handler>
```

In these Handlers with the filtering option “TunneledByPEAP” and “TunnelledByTTLS” define that the tunneled authentication (with the username and password in it) is handled here.

The “outer authentication”, where the SSL tunnel is set up, looks like the TLS handler.

```
<Handler Realm=group1>
        <AuthBy FILE>
                # the %D/users file can be empty, it’s there for normal PAP
                # authentication. This can however be used for the WEB captive
                # gateways.
                Filename %D/users
                EAPType TTLS, PEAP
                EAPTLS_CAFile %D/root.pem
                EAPTLS_CertificateFile %D/server.pem
                EAPTLS_CertificateType PEM
                EAPTLS_PrivateKeyFile %D/server.pem
                EAPTLS_PrivateKeyPassword serverkey
                EAPTLS_MaxFragmentSize 1024
                EAPAnonymous anonymous@group1
                AutoMPPEKeys
```

```
        </AuthBy>
</Handler>
```

General options

Before all clients and handler definitions there is some space for general configuration parameters for Radiator.

```
Trace 4

LogDir /var/log/radius
DbDir /etc/radiator

AuthPort 1812
AcctPort 1813
```

Here the location of logfiles (referred in the configuration as %L) and databases (%D) can be defined as well as the amount of log output is given (with Trace).
Also, there is an important section with “AuthPort” and “AcctPort”. These parameters define the ports used by the RADIUS server. The most common ports are 1812 for authentication and 1813 for accounting, however Radiator still uses the ports 1645 and 1646 as default.

4.5 Switch configuration

The next part shows a complete configuration for a Catalyst 3550 series switch.
Only relevant parts of the IOS-configuration are included.

```
!
aaa new-model
aaa authentication dot1x default group radius
!
vtp mode transparent
!
vlan 10,20,30,60,70
!
interface FastEthernet0/1
 description --- NORMAL SWITCHPORT IN VLAN 10 ---
 switchport access vlan 10
 switchport mode access
 no ip address
 spanning-tree portfast
!
interface FastEthernet0/12
 description --- SWITCHPORT WITH 802.1X: VLAN 10 WITH AUTH, VLAN 70 WITHOUT AUTH ---
 switchport access vlan 10
 switchport mode access
 no ip address
 dot1x port-control auto
 guest-vlan 70
 spanning-tree portfast
!
interface FastEthernet0/48
 description --- SWITCHPORT THAT IS THE TRUNK PORT WITH ALL VLANS TO CENTRAL SWITCH
---
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 10
 switchport mode trunk
 no ip address
!
interface Vlan10
 ip address 192.168.10.1 255.255.255.0
!
ip routing
ip route 0.0.0.0 0.0.0.0 192.168.10.254
!
radius-server host 192.168.100.10 auth-port 1812 acct-port 1813 key very_secret!
!
```

4.6 Wireless access using 802.1X

In this example the Aironet 1200 AP's from Cisco are used. This AP is one of the most functional avialable today because it supports 802.1X with user based VLAN assignment and enables the use of different VLANs to allow both Web based redirection and VPN to overcome interoperability issues. Also, these APs can be installed with different radio interfaces: 802.11a, 802.11b and 802.11g operating at the same time.

RADIUS

After the AP is provided with the proper IP-identity (on the BVI1 interface) RADIUS can be enabled

```
AP# configure terminal
AP(config)# aaa new-model
AP(config)# radius-server host 192.87.108.67 auth-port 1812 acct-port 1813 key <secret>
```

On the Access-Point we also need to define authentication-groups, for both eap-authentication, and RADIUS-accounting. This can be done with:

```
Ap(config)# aaa group server radius rad_eap
Ap(config-sg-radius)# server 192.87.108.67 auth-port 1812 acct-port 1813
Ap(config-sg-radius)# aaa group server radius rad_acct
Ap(config-sg-radius)# server 192.87.108.67 auth-port 1812 acct-port 1813
Ap(config-sg-radius)# aaa authentication login eap_methods group rad_eap
Ap(config-sg-radius)# aaa accounting network acct_methods start-stop group rad_acct
```

Basic wireless settings

First, a new SSID has to be created on the base station in the interface settings of Dot11Radio 0. On this SSID (eduroam), EAP-authentication and encryption must be enabled. We do this on the primary SSID, since 802.1X is the primary access-control mechanism.

In the interface settings the encryption strength must be defined per VLAN. Also, the key-rotation interval can be configured with broadcast-key and dot1x reauth-period.

```
Ap(config)# interface dot11Radio 0
Ap(config-if)# encryption vlan 117 mode ciphers wep40
Ap(config-if)# encryption vlan 109 mode ciphers wep40
Ap(config-if)# broadcast-key vlan 109 change 1800
Ap(config-if)# broadcast-key vlan 117 change 1800
Ap(config-if)# dot1x reauth-period 1800
Ap(config-if)# ssid eduroam
Ap(config-if-ssid)# authentication open eap eap_methods
Ap(config-if-ssid)# accounting acct_methods
Ap(config-if-ssid)# guest-mode
Ap(config-if-ssid)# vlan 117
```

The guest-mode statement in the configuration above is a bit misleading: it does not indicate that this SSID is to be used by guests; it’s just a statement to enable the broadcasting of the SSID. There is no good reason for not broadcasting an SSID, especially since in Windows XP SP1 broadcasted SSIDs are preferred over non-broadcasted SSIDs. We choose to broadcast this SSID because there are mostly Windows XP clients, and every Windows XP client is capable of doing 802.1X. The non-broadcasted SSIDs that we configure are for users of the WEB gateway or users that need VPN access. (In the SURFnet case guests often also have Wireless Client Software since they’re not able to do 1X, and therefore have not the problem with XP and non-bradcasted SSIDs)

An extra SSID can be configured with:

```
Ap(config)# interface dot11Radio 0
Ap(config-if)# ssid eduroam-guest
Ap(config-if-ssid)# description -- this SSID is wide-open for WEB/VPN guests
Ap(config-if-ssid)# vlan 163
```

NB: There is neither a WEP-key nor an authentication type defined for this SSID/VLAN pair.

VLAN interfaces

For every VLAN that should be wireless available a pair of interfaces must be configured: the dot11Radio and fastEthernet interface. The relationship between the interfaces is indicated with a bridge-groups.

Also, there is a maintenance interface, which is the “native VLAN”. The native VLAN is a Cisco-speak for “untagged” VLAN. With a “tagged”-VLAN there are tags for every frame on layer 2 that indicate what VLAN the packet belongs to. The untagged-VLAN lacks these, and in Cisco AP’s this one is commonly used for maintenance. Also, the bridge virtual interface, BVI1, that provides the AP with IP connectivity uses this native VLAN. The native-VLAN uses bridge-group 1, and has “native” as suffix on the encapsulation details.

In our examples we use:

```
interface Dot11Radio 0.108
!
 encapsulation dot1Q 108 native
 bridge-group 1
!
interface fastEthernet 0.108
!
 encapsulation dot1Q 108 native
 bridge-group 1
!
interface Dot11Radio 0.117
!
 encapsulation dot1Q 117
 bridge-group 117
!
interface fastEthernet 0.117
!
 encapsulation dot1Q 117
 bridge-group 117
!
```

And two other definitions are for the production VLAN (109) and WEB-guest VLAN (163).

```
interface Dot11Radio 0.109
!
 encapsulation dot1Q 109
 bridge-group 109
!
interface fastEthernet 0.109
!
 encapsulation dot1Q 109
 bridge-group 109
!
interface Dot11Radio 0.163
!
 encapsulation dot1Q 163
 bridge-group 163
!
interface fastEthernet 0.163
!
```

```
 encapsulation dot1Q 163
 bridge-group 163
!
```

4.7 Configuration of the captive portal

A captive portal provides users with a web based interface as soon as they try to connect a website on the internet. Initially the users get a layer 2 connection either by using Wireless LAN by associating with the broadcasted SSID “eduroam-guest” that was configured for this purpose or using a physical connection to a wired network, to the guest-VLAN on the switch. After this layer 2 connection, the client obtains a DHCP lease from the captive portal, which acts as a DHCP server as well.

Before authenticating no other internet traffic is possible, just a connection to the portal can be made: every website that a user tries to access results in a redirection to a web page to login on the portal.

There is just a single method for authentication available on this page, unlike with EAP it’s only username/password. There is also no mechanism for traffic separation: users already get an IP-address assigned by the DHCP server since there is a layer 3 connection possible from the start. (With 802.1X there is authentication before layer 2 is accessed.) This layer-2 and layer-3 connectivity also allows you to access other devices on the same network, and maybe even proxy packets through another machine to bypass your security.

Also, the data is not encrypted, so the username/password credentials that are entered on the website are relatively weak! It is possible to enforce the use of the SSL encrypted portal by redirecting the user one more time, but this can be done as well by a malicious client in the network: nothing keeps this user from providing a DHCP server and playing captive portal itself, and the identity of a certificate is hard to validate when there is no possibility to check on the IP-address.

Apart from these disadvantages it is easy for most users and especially guests since the only requirement is a browser. There are many kinds of captive portals, often commercially available.

A pretty simple and easy to configure web based authentication system is TINO. This system works under Linux, and uses various open source components to build the portal.

Linux installation

First it is assumed that there is a fresh Linux installation available (with a fresh kernel) on a machine with at least two network interfaces: one for the public internet, one for the network where users have to initially authenticate to be granted internet access.

The Linux flavor of choice is Debian Stable, (also preferred by TINO maintainers). With Debian comes APT, the Advanced Package Tool. With APT, you can update your complete Operating System with only two commands (apt-get update and apt-get upgrade). It is advised to do this after your Debian installation.

Because TINO runs from the webserver and needs to alter the firewall, it needs root permissions for one specific script. Install sudo with “apt-get install sudo” and edit sudoers with visudo. Add:

```
www-data        ALL=NOPASSWD: /usr/local/tino/firewall.pl
```

This firewall.pl script is dealt with later on.

Network configuration

To enable the routing functionality under Debian edit the /etc/network/options file, and change

```
ip_forward=no
```

into:

```
ip_forward=yes
```

To enable these settings restart the network environment with “/etc/init.d/networking restart” or make the changes manually (not persistent!) with:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Next, edit /etc/network/interface and make sure both interfaces have an IP-address. In this example, the eth1 interface is the users-network, with a private IP-range. (Of course this can be public as well). Append something like:

```
iface eth1
      address 192.168.1.1
      broadcast 192.168.1.255
      netmask 255.255.255.0
```

Next do a “ifdown eth1” and “ifup eth1” to make the changes effective.

DHCP and DNS

For TINO a DHCP server is required as well. Install this one with:

```
apt-get install dhcp3-server
```

Next, edit the /etc/dhcp3/dhcpd.conf file and make sure it contains something like:
ddns-update-style none;

```
option domain-name "guests.surfnet.nl";
option domain-name-servers 192.168.1.1;

default-lease-time 600;
max-lease-time 7200;

authoritative;

log-facility local7;

subnet 192.168.1.0 netmask 255.255.255.0 {
	range 192.168.1.10 192.168.1.20;
	option routers 192.168.1.1;
}
```

If you need a separate logfile for DHCP change /etc/newsyslog.conf and insert on top:

```
local7.debug    /var/log/dhcpd.log
```

Next, restart syslogd and start the dhcp3 server with:

```
/etc/init.d/syslogd restart
/etc/init.d/dhcp3-server start
```

As you noticed the server also has the role of DNS server, to enable DNS lookups on the client. Otherwise the client would not be able to find a valid IP-address, and the login-screen would never appear in the browser.

Install bind9 (or bind if you prefer) with:

```
apt-get install bind9
and start it with “/etc/init.d/bind9 start”
```

If you want the user to be able to type “logout” in the address bar and get the TINO logout screen make sure you create a zonefile and create a zone for the domain-name. Include an A-record for “logout”.domain-name, in this example “logout.guest.surfnet.nl” that points to the IP-address of the TINO-interface (192.168.1.1 for instance).

Apache

For TINO, there are some open source components needed. One of them is apache, the most popular webserver. This one can be installed with:

```
apt-get install apache
```

Also, if SSL encryption is required (usable for redirecting users that try to access SSL websites instead of a regular website) install apache’s SSL modules with:

```
apt-get install libapache-mod-ssl
```

Make sure you edit /etc/apache/httpd.conf. Search for the ScriptAlias section, and define:

```
ScriptAlias / /usr/local/tino/tino.cgi
```

(assuming that /usr/local/tino is the place where the tino files remain).
This way, every file that is requested on the website ends on having the login dialog. Otherwise, users requesting for example http://www.surfnet.nl/innovatie/wlan/ would end up having a 404 error, since they are redirected to the local port 80, but the requested document is not available.)

For SSL, a certificate is needed. If none is available, a self-signed certificate suffices, since a warning is given anyway. Put the certificates (server.key and server.crt) in the correct directories (/etc/apache/ssl.key and /etc/apache/ssl.crt) and append the following to the “LoadModule” section:
LoadModule ssl_module /usr/lib/apache/1.3/mod_ssl.so

and include directly under it:

```
<IfModule mod_ssl.c>
      Listen 443
      Listen 80
</IfModule>
```

Also, append the following section to the end of your configuration to enable the SSL-engine on the HTTPS port:

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
      SSLEngine on
      SSLCertificateFile /etc/apache/ssl.crt/server.crt
      SSLCertificateKeyFile /etc/apache/ssl.key/server.key
      ScriptAlias / /usr/local/tino/tino.cgi
</VirtualHost>
</IfModule>

apt-get install libauthen-radius-perl
apt-get install libcrypt-passwdmd5-perl
```

TINO installation

Apart from apache, TINO also needs two PERL modules for the RADIUS authentication backend. Install these with:
apt-get install libauthen-radius-perl
apt-get install libcrypt-passwdmd5-perl

Next, get TINO from:
http://www.cc.puv.fi/~teu/tino/ (version 1.2 is used in this document)
with for example “wget http://www.cc.puv.fi/~teu/tino/tino-1.2.tar.gz”.

Make the “/usr/local/tino” directory for TINO’s files with mkdir and untar the tarball fetched from the TINO website while in this directory with “tar -xvzf ~/tino-1.2.tar.gz”.

Then edit the just created /usr/local/tino/tino.pm and change the values of $radius_host and $radius_secret to match your RADIUS server. If you like, you can change the $html_heading variable as well to customize the title on the login page.

In order to work TINO needs some directories and files, create them with:
mkdir /var/spool/tino
touch /var/log/tino

Change the owner of this file and directory to match the one of the apache-user:

```
chown www-data:www-data /var/spool/tino /var/log/tino
```

TINO needs some scripts to run from cron, in order to clean up something. Edit /etc/cron.d/tino and add:

```
@reboot root /usr/local/tino/firewall.pl reset
* * * * * www-data /usr/local/tino/tino check
0 0 * * * root /usr/bin/savelog -p /var/log/tino &>/dev/null
```

The first line is for the firewall script, which is not yet in place.
Firewall configuration

The Linux firewall used is based on iptables. Make sure it is in the kernel or loaded as a module.
Add /usr/local/tino/firewall.pl with proper permissions (chmod 700 firewall.pl) and for instance the following contents:

```
#!/bin/sh
#
# This script is called with the following parameters:
#
# To open a rule: <"open"> <IP> <MAC>
# To close a rule: <"close"> <IP> <MAC>
# To reset the firewall: <"reset">
#
# <IP> is in dotted decimal form (xxx.xxx.xxx.xxx)
# <MAC> is on the form xx:xx:xx:xx:xx:xx

case "$1" in
  open)
   /sbin/iptables -A FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -A FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -I PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  close)
   /sbin/iptables -D FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -D FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -D PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  reset)
   /sbin/iptables –A FORWARD –s 192.168.1.0/24 –d CASG-range –j ACCEPT
   /sbin/iptables -P FORWARD DROP
   /sbin/iptables --flush
   /sbin/iptables --flush -t nat
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT \
                         --to-port 80
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j REDIRECT \
```

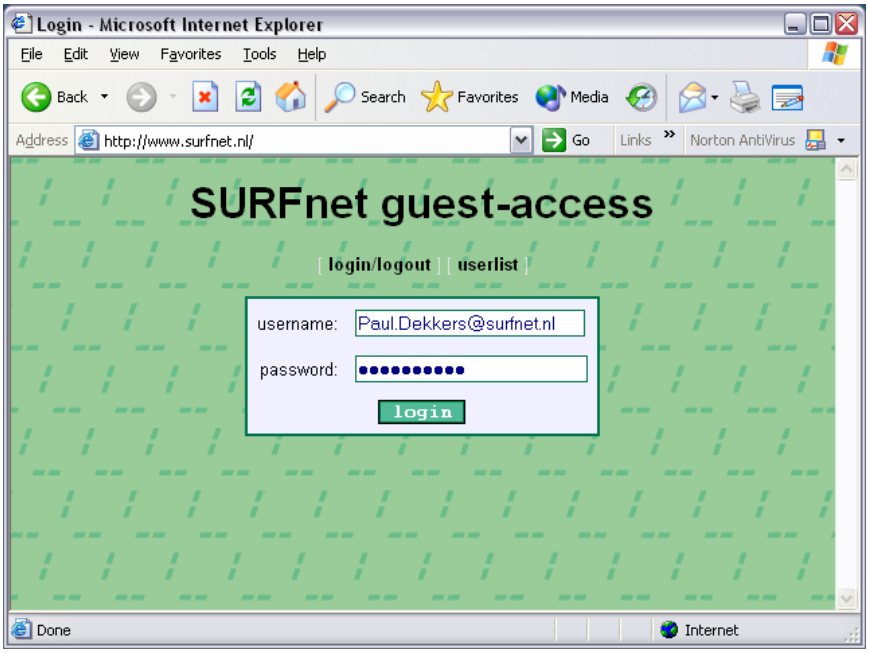
```
                --to-port 443
  # if you need MASQUERADING
  #/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
  ;;
esac
```

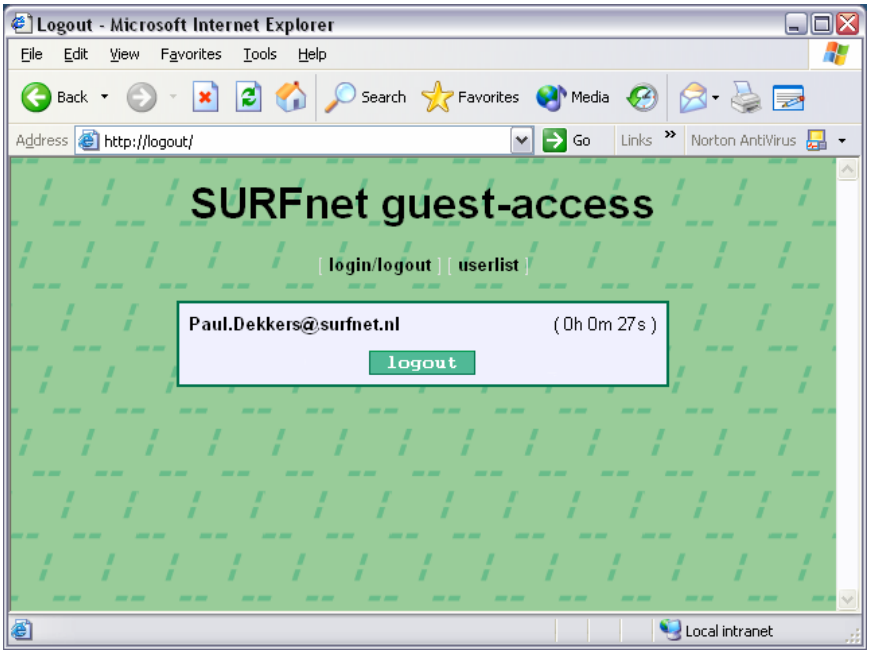
This script allows all traffic through to the CASG address-range but intercepts all other traffic and forwards it to the web-based authentication portal. Upon proper authentication a firewall is created that allows traffic through to the Internet.

Notice that this in fact is no Perl program but a simple shell script, but it is called by default from tino as /usr/local/tino/firewall.pl so it is easiest to keep this filename.

If you need NAT on the network (if it is a private IP-range, which is of course not encouraged) then enable the last line for MASQUERADING.

Next, look at the results!





4.8 VPN passtrough

VPN users get access via the captive portal. The firewall script must be altered in order to accept various VPN-servers. In the Tino setup provided in the previous paragraph all traffic was allowed to pass through to the CASG address-range. There is a danger in this, a user that manages to put a server in the CASG address-range can create a tunnel and bypass the portal. It might be argued that in fact it would be better to only allow specific protocols and ports that are necessary to connect to some VPN concentrator.
Altering for instance the FORWARD table to allow GRE traffic from and to a PPTP server and TCP traffic to port 1723 should be enough to allow PPTP, and allow UDP ports 500 to 500 and ESP and ASP traffic to the VPN servers for IPSEC. The downside to this is of course that you can not run just any kind of VPN software anymore.
Note also that in this configuration network address translation is used, this may introduce problems with some VPN-solutions, so for maximum compatibility with the VPN-based solution another approach may be chosen.

5 Conclusion

It has been acknowledged by the taskforce that currently there exists no single roaming solution that satisfies all the requirements that the various NRENs have. Recent developments however show that consensus is building towards a single, RADIUS-based, European roaming infrastructure based on 802.1X (and its successors).
Having said this, it is clear that various solutions will continue to co-exist for quite some time to come. This deliverable demonstrates that is is possible to build an infrastructure that allows NRENs to gradually move to this single infrastructure while at the same time providing ‘backward compatibility’ for their own users and guest users from other NRENs.
Time will show if and how fast this co-existence will seize to exist.

Appendix A: SURFnet configuration

RADIUS server

Radiator 3.9

```
Trace 4
LogDir /var/log/radius
DbDir /etc/radiator
AuthPort 1812
AcctPort 1813
<Client 192.168.10.2>
Secret very_secret!
Identifier AP-and-Switch
IdenticalClients 192.168.10.1
</Client>
<Client 192.168.100.1>
Secret super_secret!
Identifier Proxy-Identifier
</Client>
<Handler TunnelledByPEAP=1, Realm=group1>
<AuthBy FILE>
Filename %D/peap-users
EAPType MSCHAP-V2
</AuthBy>
</Handler>
<Handler TunneldByTTLS=1, Realm=group1>
RewriteUsername s/^([^@]+).*/$1/
<AuthBy FILE>
Filename %D/ttls-users
</AuthBy>
</Handler>
<Handler Realm=group1>
<AuthBy FILE>
# the %D/users file can be empty, it's there for normal PAP authentication
Filename %D/users
EAPType TTLS, PEAP
EAPTLS_CAFile %D/root.pem
EAPTLS_CertificateFile %D/server.pem
EAPTLS_CertificateType PEM
EAPTLS_PrivateKeyFile %D/server.pem
EAPTLS_PrivateKeyPassword serverkey
EAPTLS_MaxFragmentSize 1024
EAPAnonymous anonymous@group1
AutoMPPEKeys
</AuthBy>
</Handler>
# Proxy only requests that do not come from the proxy-server, to prevent loops.
<Handler Client-Identifier=/^(?!Proxy-Identifier$)/>
<AuthBy RADIUS>
Host 192.168.100.1
Secret super_secret!
AuthPort 1812
AcctPort 1813
StripFromReply Tunnel-Type,Tunnel-Medium-Type,Tunnel-Private-Group-ID
AddToReply Tunnel-Type=1:VLAN,Tunnel-Medium-Type=1:Ether_802,
Tunnel-Private-Group-ID=1:70
</AuthBy>
</Handler>
```

Tino

```
#!/bin/sh
#
```

```
# This script is called with the following parameters:
#
# To open a rule: <"open"> <IP> <MAC>
# To close a rule: <"close"> <IP> <MAC>
# To reset the firewall: <"reset">
#
# <IP> is in dotted decimal form (xxx.xxx.xxx.xxx)
# <MAC> is on the form xx:xx:xx:xx:xx:xx

case "$1" in
  open)
   /sbin/iptables -A FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -A FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -I PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  close)
   /sbin/iptables -D FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -D FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -D PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  reset)
   /sbin/iptables -P FORWARD DROP
   /sbin/iptables --flush
   /sbin/iptables --flush -t nat
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT \
                         --to-port 80
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j REDIRECT \
                         --to-port 443
   # if you need MASQUERADING
   #/sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
   ;;
esac
```

Cisco Catalyst

```
!
aaa new-model
aaa authentication dot1x default group radius
!
vtp mode transparent
!
vlan 10,20,30,60,70
!
interface FastEthernet0/1
 description --- NORMAL SWITCHPORT IN VLAN 10 ---
 switchport access vlan 10
 switchport mode access
 no ip address
 spanning-tree portfast
!
interface FastEthernet0/12
 description --- SWITCHPORT WITH 802.1X: VLAN 10 WITH AUTH, VLAN 70 WITHOUT AUTH ---
 switchport access vlan 10
 switchport mode access
 no ip address
 dot1x port-control auto
 guest-vlan 70
 spanning-tree portfast
!
interface FastEthernet0/48
 description --- SWITCHPORT THAT IS THE TRUNK PORT WITH ALL VLANS TO CENTRAL SWITCH
---
 switchport trunk encapsulation dot1q
 switchport trunk native vlan 10
 switchport mode trunk
 no ip address
```

```
!
interface Vlan10
 ip address 192.168.10.1 255.255.255.0
!
ip routing
ip route 0.0.0.0 0.0.0.0 192.168.10.254
!
radius-server host 192.168.100.10 auth-port 1812 acct-port 1813 key very_secret!
!
```

Cisco AP1200

```
AP# configure terminal
AP(config)# aaa new-model
AP(config)# radius-server host 192.87.108.67 auth-port 1812 acct-port 1813 key <secret>
Ap(config)# aaa group server radius rad_eap
Ap(config-sg-radius)# server 192.87.108.67 auth-port 1812 acct-port 1813
Ap(config-sg-radius)# aaa group server radius rad_acct
Ap(config-sg-radius)# server 192.87.108.67 auth-port 1812 acct-port 1813
Ap(config-sg-radius)# aaa authentication login eap_methods group rad_eap
Ap(config-sg-radius)# aaa accounting network acct_methods start-stop group rad_acct
Ap(config)# interface dot11Radio 0
Ap(config-if)# encryption vlan 117 mode ciphers wep40
Ap(config-if)# encryption vlan 109 mode ciphers wep40
Ap(config-if)# broadcast-key vlan 109 change 1800
Ap(config-if)# broadcast-key vlan 117 change 1800
Ap(config-if)# dot1x reauth-period 1800
Ap(config-if)# ssid eduroam
Ap(config-if-ssid)# authentication open eap eap_methods
Ap(config-if-ssid)# accounting acct_methods
Ap(config-if-ssid)# guest-mode
Ap(config-if-ssid)# vlan 117
Ap(config)# interface dot11Radio 0
Ap(config-if)# ssid eduroam-guest
Ap(config-if-ssid)# description -- this SSID is wide-open for WEB/VPN guests
Ap(config-if-ssid)# vlan 163
interface Dot11Radio 0.108
!
 encapsulation dot1Q 108 native
 bridge-group 1
!
interface fastEthernet 0.108
!
 encapsulation dot1Q 108 native
 bridge-group 1
!
interface Dot11Radio 0.117
!
 encapsulation dot1Q 117
 bridge-group 117
!
interface fastEthernet 0.117
!
 encapsulation dot1Q 117
 bridge-group 117
!
```

Appendix B: SWITCH configuration

Note: actual IP-addresses have been changed into 111.222.x.y

Cisco Catalyst 3750 switch

```
Model:        WS-C3750G-24TS
Software:    IOS (tm) C3750 Software (C3750-I5K2-M),
Version 12.1(19)EA1, RELEASE SOFTWARE (fc2)
!
vlan 15
 name VLAN-RESOURCE-1
!
vlan 20
 name VLAN-RESOURCE-2
!
vlan 25
 name VLAN-TERENA-WEB-VPN
!
vlan 35
 name VLAN-TERENA-802.1X
!
!
interface GigabitEthernet1/0/1
 description GE uplink internet
 ip address 111.222.15.218 255.255.255.252
 no switchport
 ip address
 ip pim sparse-mode
 ip mroute-cache distributed
 ip ospf cost 5
 duplex full
 speed 100
 no mdix auto
!
interface GigabitEthernet1/0/2
 description AP1200 trunk link
 switchport trunk encapsulation dot1q
 switchport mode trunk
 switchport nonegotiate
 no ip address
 no mdix auto
 spanning-tree portfast
!
interface GigabitEthernet1/0/3
 description  RADIUS server
 switchport access vlan 20
 switchport mode access
 no ip address
 no mdix auto
```

```
 spanning-tree portfast
!
interface GigabitEthernet1/0/4
 description BlueSocket managed interface
 switchport access vlan 25
 switchport mode access
 no ip address
 no mdix auto
 spanning-tree portfast
!
interface GigabitEthernet1/0/5
 description BlueSocket protected interface
 switchport access vlan 15
 switchport mode access
 no ip address
 no mdix auto
 spanning-tree portfast
!
interface Vlan15
 description VLAN-RESOURCE-1-GW
 ip address 111.222.98.193 255.255.255.192
 ip verify unicast reverse-path allow-self-ping
 no ip redirects
 no ip proxy-arp
 ip pim sparse-mode
 ip route-cache flow
 ip cgmp
 ip mroute-cache distributed
 ntp broadcast key 12
!
interface Vlan20
 description VLAN-RESOURCE-2-GW
 ip address 111.222.98.143 255.255.255.248
 ip verify unicast reverse-path allow-self-ping
 no ip redirects
 no ip proxy-arp
 ip pim sparse-mode
 ip route-cache flow
 ip cgmp
 ip mroute-cache distributed
 ntp broadcast key 12
!
interface Vlan35
 description VLAN-TERENA-802.1X-GW
 ip address 111.222.98.137 255.255.255.248
 ip verify unicast reverse-path allow-self-ping
 no ip redirects
 no ip proxy-arp
 ip pim sparse-mode
 ip route-cache flow
```

```
 ip cgmp
 ip mroute-cache distributed
 ntp broadcast key 12
!
```

Cisco Aironet 1200 Access Point

```
Model:         AIR-AP1220-IOS
Software:    IOS (tm) C1200 Software (C1200-K9W7-M)
        Version 12.2(13)JA1

!
aaa group server radius rad_eap
 server 111.222.98.147 auth-port 1812 acct-port 1813
!
!
aaa authentication login eap_methods group rad_eap
aaa authentication login mac_methods local
!
bridge irb
!
!
interface Dot11Radio0
 no ip address
 no ip route-cache
 !
 encryption vlan 35 mode wep mandatory
 !
 ssid 802.1x
    vlan 35
 !
 ssid web-vpn
    vlan 25
    authentication open
 !
interface Dot11Radio0.25
 encapsulation dot1Q 25
 no ip route-cache
 bridge-group 25
 bridge-group 25 subscriber-loop-control
 bridge-group 25 block-unknown-source
 no bridge-group 25 source-learning
 no bridge-group 25 unicast-flooding
 bridge-group 25 spanning-disabled
!
interface Dot11Radio0.35
 encapsulation dot1Q 35
 no ip route-cache
 bridge-group 35
```

```
 bridge-group 35 subscriber-loop-control
 bridge-group 35 block-unknown-source
 no bridge-group 35 source-learning
 no bridge-group 35 unicast-flooding
 bridge-group 35 spanning-disabled
!
interface FastEthernet0
 no ip address
 no ip route-cache
 duplex auto
 speed auto
 bridge-group 1
 no bridge-group 1 source-learning
 bridge-group 1 spanning-disabled
!
interface FastEthernet0.15
 encapsulation dot1Q 15
 no ip route-cache
 bridge-group 15
 no bridge-group 15 source-learning
 bridge-group 15 spanning-disabled
!
interface FastEthernet0.20
 encapsulation dot1Q 20
 ip address 111.222.98.146 255.255.255.248
 no ip route-cache
 bridge-group 20
 no bridge-group 20 source-learning
 bridge-group 20 spanning-disabled
!
interface FastEthernet0.25
 encapsulation dot1Q 25
 no ip route-cache
 bridge-group 25
 no bridge-group 25 source-learning
 bridge-group 25 spanning-disabled
!
interface FastEthernet0.35
 encapsulation dot1Q 35
  no ip route-cache
 bridge-group 35
 no bridge-group 35 source-learning
 bridge-group 35 spanning-disabled
!
interface BVI1
 no ip address
 no ip route-cache
!
radius-server host 111.222.98.147 auth-port 1812 acct-port 1813 key xxxx
!
```

end

BlueSocket Web-gateway

Model: WG-1100
Software: V3.0.0.12F

Datasheet and configuration details are available at http://ww.bluesocket.com

The BlueSocket configuration is based on multiple comma separated values files.

The following section is a summary of a bluesocket configuration based on a web-gateway concept.

The Bluesocket gateway is based on a role based solution where users can have different roles. Each role has its own access policy. A local database or external servers, such as RADIUS/LDAP can attached for user authentication (role change management).

- The Access policies for defined roles are as follows:

- unregistered (default):

Action	Service	Direction	Destination
Allow	ip	both ways	CASG
Deny	ip	both ways	any

- registered:

Action	Service	Direction	Destination
Allow	ip	both ways	any

- Defined destinations:

IP	Name	Subnet	Type
111.222.31.251	dns-server-switch	255.255.255.255	Host
333.222.30.0	VPN-Gateways-Surfnet	255.255.255.0	Network
111.222.40.0	VPN-Gateways-Finland	255.255.255.0	Network
111.222.50.0	VPN-Gateways-Switzerland	255.255.255.0	Network

- Destination groups defined:

Name	Member(s)
CASG	dns-server-switch, VPN-Gateways-Surfnet, VPN-Gateways-Finland, VPN-Gateways-Switzerland

- Defined Interface

IP	Subnet	Name
111.222.98.196	255.255.255.192	Protected Interface
111.222.98.129	255.255.255.192	Managed Interface

- Defined Accounting Server

IP	Name	Type
111.222.98.147	radius-switch	RADIUS

Radiator RADIUS-server

Model: RADIATOR
Software: 3.6

radius.cfg

```
LogDir          /var/log/radius
DbDir           /etc/radiator
DictionaryFile /etc/radiator/dictionary
PidFile /var/log/radius/radiusd.pid

AuthPort        1812
AcctPort        1813


<Client DEFAULT>
        Secret  xxxx
</Client>

#-----------------------------------------------
# handler for inner authentication of eap requests
#-----------------------------------------------

<Handler TunnelledByPEAP=1, Realm=university.ch>
        <AuthBy SQL>
                DBSource        dbi:mysql:radius
                DBUsername      xxxx
                DBAuth          xxxx
                RewriteUsername         s/^(.*)\\(.*)/$2/
                EAPType MSCHAP-V2
         </AuthBy>
</Handler>

<Handler TunnelledByTTLS=1, Realm= university.ch>
        <AuthBy SQL>
                DBSource        dbi:mysql:radius
                DBUsername      xxxx
```

```
            DBAuth          xxxx
            EAPType MSCHAP-V2
        </AuthBy>

</Handler>

#-----------------------------------------------
# handler for outer authentication of eap requests
#-----------------------------------------------

<Handler Realm= university.ch>
        <AuthBy SQL>
            DBSource        dbi:mysql:radius
            DBUsername      xxxx
            DBAuth          xxxx
            EAPType PEAP, TTLS
            EAPTLS_CAFile %D/certificates/demoCA/cacert.pem
            EAPTLS_CertificateFile %D/certificates/cert-srv.pem
            EAPTLS_CertificateType PEM
            EAPTLS_PrivateKeyFile %D/certificates/cert-srv.pem
            EAPTLS_PrivateKeyPassword whatever
            EAPTLS_MaxFragmentSize 1000
            AutoMPPEKeys
            SSLeayTrace 4
            EAPAnonymous %0
            EAPTLS_PEAPVersion 0
        </AuthBy>
</Handler>


#-----------------------------------------------
# handler for ccTLD
#-----------------------------------------------
#<Handler Realm = /.*\.ch/>
#       <AuthBy RADIUS>
#               Host ccTLD
#               AuthPort 1812
#               AcctPort 1813
#               Secret xxxx
#        </AuthBy>
#</Handler>

#-----------------------------------------------
# handler for root-radius-server
#-----------------------------------------------
#<Handler>
#       <AuthBy RADIUS>
#               Host euTLD
#               AuthPort 1812
#               AcctPort 1813
```

```
#               Secret xxxx
#       </AuthBy>
#</Handler>
```

Appendix C: Technical University of Tampere configuration

Note: actual IP-addresses have been changed into 111.222.x.y

Tino

Basic firewall script:

```
#!/bin/sh
#
# This script is called with the following parameters:
#
# To open a rule: <"open"> <IP> <MAC>
# To close a rule: <"close"> <IP> <MAC>
# To reset the firewall: <"reset">
#
# <IP> is in dotted decimal form (xxx.xxx.xxx.xxx)
# <MAC> is on the form xx:xx:xx:xx:xx:xx

case "$1" in
  open)
   /sbin/iptables -A FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -A FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -I PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  close)
   /sbin/iptables -D FORWARD -s $2 -m mac --mac-source $3 -j ACCEPT
   /sbin/iptables -D FORWARD -d $2 -j ACCEPT
   /sbin/iptables -t nat -D PREROUTING -i eth1 -s $2 -j RETURN
   ;;
  reset)
   /sbin/iptables -P FORWARD DROP
   /sbin/iptables --flush
   /sbin/iptables --flush -t nat
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT
--to-p
ort 80
   /sbin/iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 443 -j
REDIRECT --to-
port 443
   ;;
esac
```

Cisco AP1200

```
aaa group server radius radiator
 server-private 111.222.52.225 auth-port 1813 acct-port 1812 key xxxxxx
!
interface Dot11Radio0
 no ip address
```

```
 no ip route-cache
 !
 encryption vlan 52 mode ciphers tkip
 !
 ssid TH203
    vlan 52
    authentication open eap testi
    authentication key-management wpa optional
    accounting radiator
 !
 ssid TUT
    vlan 144
    authentication open
    guest-mode
 !
!
interface Dot11Radio0.52
 encapsulation dot1Q 52
 no ip route-cache
 no cdp enable
 bridge-group 52
 bridge-group 52 subscriber-loop-control
 bridge-group 52 block-unknown-source
 no bridge-group 52 source-learning
 no bridge-group 52 unicast-flooding
 bridge-group 52 spanning-disabled
!
interface Dot11Radio0.144
 encapsulation dot1Q 144
 no ip route-cache
 no cdp enable
 bridge-group 144
 bridge-group 144 subscriber-loop-control
 bridge-group 144 block-unknown-source
 no bridge-group 144 source-learning
 no bridge-group 144 unicast-flooding
 bridge-group 144 spanning-disabled
!

!
interface FastEthernet0.52
 encapsulation dot1Q 52
 ip address 111.222.52.222 255.255.255.0
 no ip route-cache
 no cdp enable
 bridge-group 52
 no bridge-group 52 source-learning
 bridge-group 52 spanning-disabled
!
interface FastEthernet0.144
```

```
 encapsulation dot1Q 144
 no ip route-cache
 no cdp enable
 bridge-group 144
 no bridge-group 144 source-learning
 bridge-group 144 spanning-disabled
!

HP Pro Curve 2824
vlan 52
   name ".52.0/24"
   untagged 3
   tagged 2
   exit
vlan 144
   name "pub-acc"
   untagged 4
   tagged 2
   exit
```