AUTHENTICATION AND AUTHORIZATION FOR MOBILE INTERNET USERS

Diplomarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Thomas Spreng
2006

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Acknowledgement

First of all I would like to thank Professor Dr. Torsten Braun for all of his support and for letting me write this diploma thesis in his research group *Computer Networks and Distributed Systems*. Also many thanks go to Dr. Marc-Alain Steinemann who supervised my work, gave me assistance whenever needed and had patience with me when my work was not progressing as fast as planned. Last but not least I would like to thank everyone else from the research group for all the inspiring discussions and interesting ideas we shared.

Contents

Chapter 1

Introduction

1.1 Motivation

Internet users' habits are changing in the way that many users prefer working with mobile devices. Many mobile devices already include wireless network adapters. In order to satisfy these needs, wireless docking networks become more and more important. In Swiss universities radio networks for mobile Ethernet users are mostly reserved to users of the respective university. The Swiss research network provider SWITCH [1] offers a workaround for connecting mobile devices by means of virtual private network (VPN) tunnels back to the user's home university. This workaround has several drawbacks, especially the installation and maintenance costs of the virtual private network gateways are high. Additionally, users do not get access to local services offered by the hosting university and cannot roam. It only works in universities that prepare their wireless local area networks (WLANs) for this workaround. SWITCH is implementing a Swiss-wide authentication and authorization infrastructure (AAI) [2] for universities that allows users to access enabled services from any place where Internet connection is available. Logging-on to wireless networks however is not addressed in the present solution. It is not sure whether this technology could be used for an easy access to wireless LAN networks but it shows at least how user-friendly a proposed solution could be. Users should be enabled to easily connect and use their mobile devices in all university campuses with the same security level they are used to in their home university.

1.2 Objectives

The goal of this thesis is to develop a concept and a prototype implementation of an authentication and authorization procedure for mobile Internet users. The focus is based upon the situation that exist among Swiss universities which means that already deployed infrastructures and technologies should be considered. Nevertheless, any solution must be able to be adapted to work in different scenarios, not only within universities. Five main objectives have been defined to be achieved in this diploma thesis:

1. Users of mobile devices should be able to access home and foreign wireless and wired LAN networks in a single authentication and authorization procedure. That means no

preliminary action is required for the user before accessing a network.

2. Users should be able to use the same credentials as they use in their home universities. Any type of a local user directory used for authentication purpose must be avoided since it will make any solution unscalable and dangerous in terms of privacy.

3. Users should be able to roam in the visited network without re-authenticating at each hand-over.

4. No third-party software should be used on the client side. This is also to support that the solution will be as platform and application independent as possible.

5. Users should have access to local services offered by the hosting organization such as printers for example. Access to such local resources may be handled differently in each organization since it depends on their policy, network topology and other things. The solution however should take this into account and show how local resources can be accessed.

While the thesis must focus on the main objectives mentioned above, there are also some other aspects that should be taken care of, especially when implementing a prototype. Since users will also be able to get network access in different foreign organizations, one must pay attention to topics like privacy, security and accounting. Otherwise many participants will most likely refuse to accept any solution. Such implementation criteria consists of the following points:

- Security:
  There are several layers where data security is important in such a scenario. In the authentication phase it is very important that the user credentials are not disclosed and cannot be sent to any malicious attacker. The same applies to the authorization phase where all the user attributes that are being exchanged must remain their privacy. Furthermore the user's network access session should be secured against any form of misuse. The next paragraph shows some more detailed information about the specific security threats.

- Low administration and maintenance costs:
  The maintenance costs of a running implementation should be kept as low as possible. Since several independent organizations may be involved in such a solution, administrative tasks such as bi-lateral agreements for example may result in high expenses and effort. Therefore it should taken care that those two cost factors remain at a reasonable level.

- Scalability:
  Every good design should be done with scalability in mind. There is no difference in that fact here. We do not know how many organizations and users will be involved in such a scenario hence it must be designed in a way that makes it work with many participating organizations and users. Such a system may also grow in the future. This must be taken care in the initial design process.

- Accountability:
  The origin situation is based upon network access for users among different universities. These participating universities may differ depending on the scenario where such a solution will be used. Libraries, other federal institutes, private organizations or commercial companies could take part, thus accounting information should be collected in a way that participating organizations could charge for the service they provide to foreign users.

- Guest user handling:
  Especially in a university scenario there are often situations where people that do not belong to any of the participating organizations need Internet access for a short period of time. It must be possible to include such guest users in an implementation in a convenient way.

- No use of closed or proprietary software and standards:
  Any form of proprietary software should be avoided for it makes further improvements and enhancements much more difficult and dependent of the owner. In addition to that, the involved third party software should be licensed in a way that it is free of charge for this use.

Security Considerations for Wireless Networks

On a wired network you have to physically attach your device to a port in order to receive or transmit network traffic. Users in such a network are much easier to control because it is possible to know who has physical access to the network and where he is located at. Besides that sniffing and eavesdropping can be made very difficult with the use of fully switched networks. However, a wireless network does not have such properties and therefore needs some special security considerations. The following security threats should be addressed when planning a wireless docking network:

- Eavesdropping:
  If you transmit any traffic on a wireless network everyone can receive and read those packets. That means that all sensitive data has to be encrypted in an appropriate way. This can be done by using transport protocols that offer such secure encryption such as HTTPS [3] or by establishing VPN tunnels.

- Masquerading and session hijacking:
  A malicious user may scan for 802.11 [4] frames and get the MAC and IP address (or other identity information) of a victim. The attacker then may make take over the session of the victim claiming the false identity. Before doing this, the hijacker either waits for the victim to leave the network or the malicious user can flood the victim with 802.11 MAC disassociate frames to forcefully log him out. If the access controller does not know that the user left the network or has been forced to, the attacker may take over that session until a new re-authentication is requested.

- Rogue access points and fake portals:
  If you associate with an access point (AP), you do not really know which device you are

actually connected to. It is just assumed that it is the corresponding wireless access point but it might also be a malicious user's wireless interface, pretending to be a valid access point. That way an attacker could set up a fake environment in order to get private data such as user name and password combinations. Therefore, an authentication procedure should always be mutual which means that not only the user authenticates himself to the network but also the according device to the user.

- Man-in-the-middle attacks:
  An attacker who positions himself between the victim and the access point might act as the access point to the user and as the user to the access point while intercepting all traffic between those two parties. The malicious user is then able to read, insert and modify any messages at will without the knowledge of either party.

- WEP algorithm flaws:
  A number of security flaws have been found in the WEP (Wired Equivalent Privacy) algorithm. There are active and passive methods to encrypt the traffic regardless of the length of the shared secret caused by a weak implementation of the WEP initialization vectors [5]. All these attacks can be performed even by low cost equipment, therefore 802.11 networks should not rely on WEP as a security mechanism.

1.3 Structure of the Document

After defining the concepts of authentication and authorization infrastructures and mobility management the following chapter 2 presents some protocols that might be used for such purpose and that have been evaluated for this thesis. Following that an overview about similar, already existing implementations is given. Chapter 3 shows an analysis of possible authentication and authorization schemes for mobile users. Those schemes are discussed and finally evaluated how they comply with the given goals and one is chosen as the proposed solution for this thesis. Then this solution is explained in detail in chapter 4 and its implementation is presented in chapter 5. Afterwards, some measurements are shown in chapter 6. Finally chapter 7 is the last one concluding the work and gives an overview about the future development of the proposed solution and its implementation.

Chapter 2

Related Work

In order to better understand the schemes that are discussed in the next chapter and the following chosen solution it is helpful to introduce some protocols and technologies that are relevant in such a scenario. At first the terms authentication, authorization and mobility management are explained because they are very often used later on in this document. After this introduction some network access and authorization protocols will be explained that have been evaluated for this thesis. Finally some already existing implementations will be shown that are used in similar scenarios.

The terms authentication and authorization are widely used in this document. Authentication is actually the process of verifying an identity claimed by an entity. This means to authenticate a user is the process to verify his identity at the organization he belongs to, often referred to as the identity provider. Authentication may not only be used in conjunction with persons but also with computers and other devices. This is very important since for an example a client may never be sure to what server he is connected to unless the server has been authenticated as well.

Authorization is often mentioned in combination with authentication and refers to the process of granting or denying access on a requested resource and requires an authentication process in advance. A protocol that serves the purpose of those two terms is usually called an authentication and authorization infrastructure (AAI).

Mobility management is a loose term used for all actions that are related to the mobility of a user. Information about the location of a mobile user, what base station he is connected to and similar data is covered by this term. Roaming is one of the most important procedures that emerges from mobility management.

2.1 IEEE 802.1X

IEEE 802.1X [6] is an open standards-based protocol for authenticating hosts on a network port. This port-based network access control makes use of the physical characteristics of an IEEE 802 LAN [7] environment in order to authenticate and authorize devices to a single point of attachment in a LAN environment. The protocol introduces the following terms to define this standard.

- Authenticator:

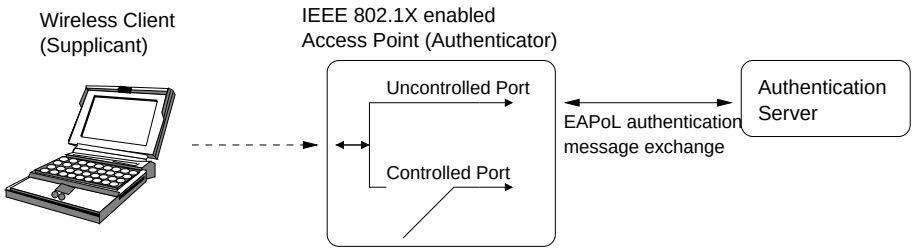

Figure 2.1: 802.1X Authorization State of the Controlled and Uncontrolled Port

A device that requests authentication from a node (supplicant) that is attached to a LAN port. It acts as an intermediary between the supplicant and the authentication server, requesting information from the client, verifying this information with the authentication server and relaying responses back to the client.

- Authentication Server:
  This device provides authentication services for the authenticator. Based on the information sent by the supplicant it decides whether it is authorized to access the network or not.

- Supplicant:
  A device that is attached to a port and requests authorization to access the network.

The operation of port-based access control implies two distinct states of access at the point of attachment, namely controlled and uncontrolled state as shown in figure 2.1. Any datagrams sent to the physical port are made available to both states of the port but only the uncontrolled port is switched on when a new supplicant attaches to the physical port. It is subject to the authorization procedure whether the status of the controlled port is then switched on or off. By default this decision is based on the result of authentication exchanges between the supplicant, authenticator and authentication server. The authenticator uses the uncontrolled port to communicate authentication protocol information with the supplicant. Protocol message exchanges between the authenticator and the authentication server can be done via controlled or uncontrolled ports. In a IEEE 802 LAN environment this authentication communication is done by EAPoL (EAP over LAN) messages.

2.2 Extensible Authentication Protocol (EAP)

The PPP Extensible Authentication Protocol (EAP) [8] is a protocol for point-to-point authentication that supports different authentication methods. The key features of EAP are:

- Flexibility:
  EAP acts as a framework for network authentication protocols. Whenever a new authentication method is implemented, only a new EAP protocol number needs to be reserved. The authenticating device (for example the access point) does not need to understand the

new protocol which means no changes are needed in order to introduce a new method. The EAP packets are just forwarded to the authentication server.

- Reliability:
  EAP offers a reliable data transfer since it is a media independent protocol, which means that the presence or absence of lower layer security is not taken into account. This is done by retransmission and elimination of duplicated packets.

- Media Independency:
  EAP is a media independent protocol. Normally it runs directly over the data link layer such as PPP or IEEE 802.

When a new network link is established, the authenticator sends a request to authenticate the client device. Normally the first query includes an identity request followed by requests for authentication information, whatever the authentication method will be. These requests will be answered by corresponding Response packets that contain the required information. Finally, the authenticator will send either an Success or a Failure packet back to the client.

The IEEE 802.1X standard defines how EAP is used over LAN, be it wired or wireless. In this case the EAP packets are encapsulated in Ethernet frames and not in PPP packets. This protocol defined by the IEEE 802.1X standard is called EAPoL (EAP encapsulation over LAN). Figure 2.2 shows an overview of the complete EAP protocol layering.

When an new client wants to access an 802.1X protected network, the authenticator forces the client's port in an uncontrolled state which means only 802.1X related traffic is forwarded. Then the client may send an EAP Start message, which will be answered by EAP Request packets from the authenticator. The EAP Response packets from the supplicant will then be forwarded to the authentication server. It uses a special authentication algorithm to verify the supplicant's identity. The authentication server responds then with either an EAP Reject or Success packet. In the latter case the authenticator will change the client's port in an authenticated state, which means that other traffic is forwarded as well and the client has access to the connected network.

Authentication Methods for EAP

Although there are dozens of methods, only some are being discussed here. These methods are: TLS (Transport Layer Security), TTLS (Tunneled Transport Layer Security), PEAP (Protected EAP), LEAP (Lightweight EAP) and SPEKE (Simple Password-authenticated Exponential Key Exchange).

Transport Layer Security (TLS)

EAP-TLS is a certificate-based authentication method, which provides mutual authentication. That is very important, especially when using wireless docking networks in order to prevent connecting to rogue access points. This is accomplished by signed certificates for both, clients and servers. Unfortunately this implies the setup of a full Public Key Infrastructure (PKI), which does not scale very well and is very cost intensive.

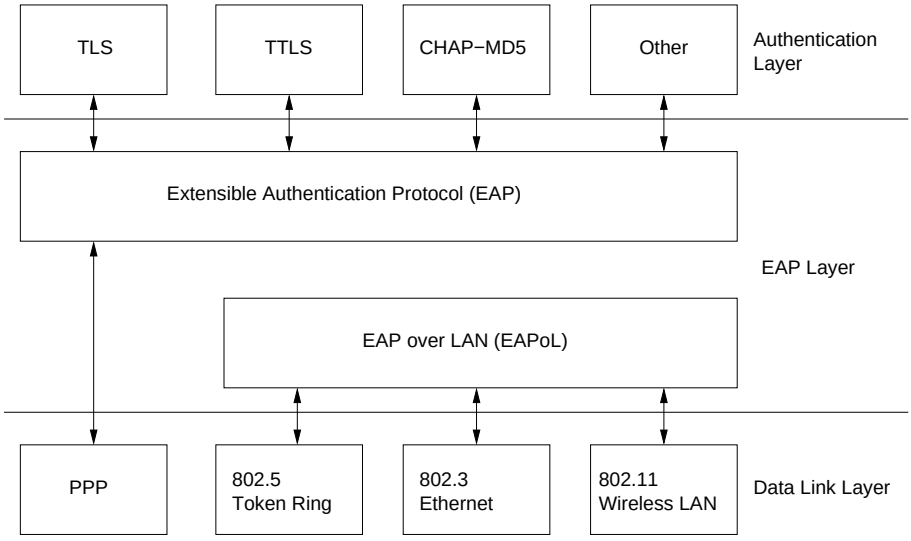

Figure 2.2: Extensible Authentication Protocol layers

An EAP-TLS conversation between a peer (for example a mobile device) and an authenticator looks as follows. When the mobile device requests access to the docking network, normal EAP negotiation takes place at first. After the authenticator has sent an EAP Request Identity packet, it may act as an pass through device between the peer and an EAP server (for example a RADIUS server). Then the client sends back its identity in a EAP Response message. The EAP server starts the EAP-TLS negotiation with an EAP-TLS Start packet (encapsulated in an EAP Request packet), which is responded by an EAP-TLS client_hello from the mobile device. This client_hello contains a session id among other things, which can be used to resume a former EAP-TLS connection. If there is no session to resume, the EAP server sends an EAP-TLS packet containing its certificate, a server_hello (with a new session id), a server_key_exchange, a server_hello_done message and a certificate request. The peer will then answer this with a EAP-TLS packet that holds its certificate, a client_key_exchange, a certificate_verify, a change_cipher_spec message and a TLS finished flag. Upon a successful certificate verification, the server sends back an EAP-TLS packet with a change_cipher_spec message and a TLS finished flag. After a final EAP Response and EAP Success message exchange the EAP-TLS session is established.

Tunneled Transport Layer Security (TTLS)

This method is an extension to TLS (explained in the previous paragraph). It also provides mutual authentication. Due to TLS is used to establish a tunnel between the client and the TTLS Server, other legacy authentication protocols can be used (besides certificate-based methods). The secure tunnel established by the TLS handshake is used to authenticate the client using other authentication infrastructures such as RADIUS. The TTLS packet payload includes the protocol used to authenticate the client which can be a PAP, CHAP or any other method. This

means that TTLS can be used to protect legacy authentication protocols against existing security threats by using a secure transport tunnel.

Protected Extensible Authentication Protocol (PEAP)

PEAP is very similar to TTLS. It also adds a TLS layer on top of EAP but it uses the secured session to protect a second EAP exchange. Common EAP methods supported by PEAP are EAP-MS-CHAPv2 and EAP-TLS. This protocol was developed by Microsoft and therefore has built-in client support in their operating systems. On the other hand client implementations for Unix based and Linux systems are in a more experimental state at the moment.

Lightweight Extensible Authentication Protocol (LEAP)

LEAP is a proprietary EAP authentication protocol developed by Cisco Systems, which means it can only be used in conjunction with their Access Points products. Like EAP-TLS it also provides mutual authentication. The authentication is based on a user name/password scheme. It uses the following authentication process. A mobile client connects to a LEAP enabled access point and sends an EAP Start packet. The authenticator sends an access request to the authentication server. Then the peer sends its user name to the access point, which forwards this message to the server. The authentication server sends a challenge back to the mobile client. The client computes this challenge with the Cisco LEAP algorithm and mixes it with the password. This encrypted password will be sent back to the authentication server, which validates the derived challenge and response value and sends back a success message to the client. Then the client sends an AP authentication challenge back to the server. If the network is successfully authenticated, the client sends a success message to the server, which will open the 802.1X protected network port. Finally, a WEP key will be generated and stored in the AP and the client for that session.

Simple Password-authenticated Exponential Key Exchange (SPEKE)

SPEKE is a password-based authentication method for EAP like LEAP but it is completely access point independent. It features mutual authentication using a password which is integrated in a normal Diffie-Hellman [9] exchange. That way the password is protected against sniffing and unconstrained brute force attacks from the network. Other active and passive network attacks such as man-in-the-middle and replay methods should be prevented by that method as well.

2.3 Remote Authentication Dial In User Service (RADIUS)

The Remote Authentication Dial In User Service (RADIUS) [10] protocol was developed by Livingston Enterprises, Inc as a distributed solution to meet the security requirements of remote computing. Distributed security is designed to separate user authentication and authorization from the communications process and to create a single, central location for user authentication data. This data can be accessed in different ways like via Network Information Service (NIS) or by querying a dedicated RADIUS database. The basic operation scheme is shown in figure 2.3.

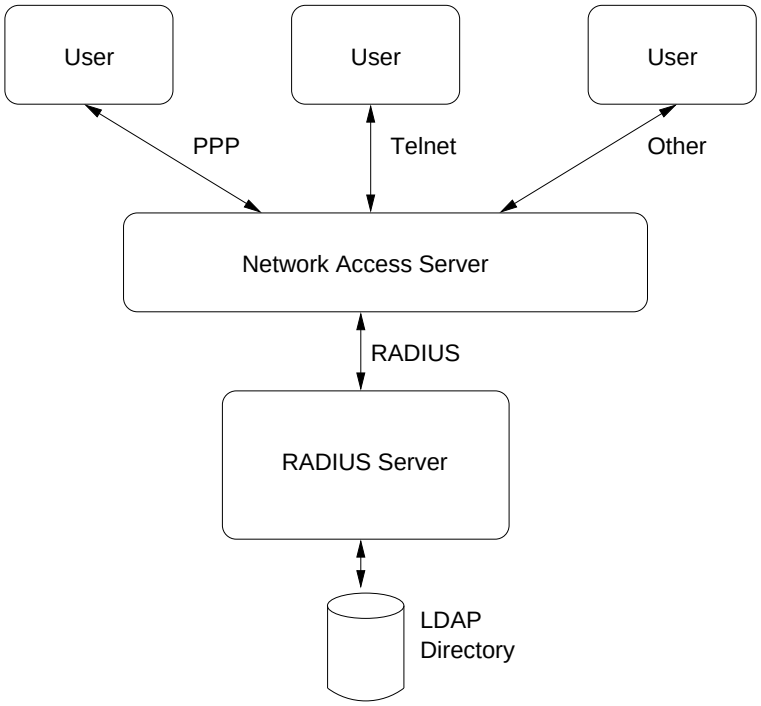

Figure 2.3: RADIUS Authentication Architecture

User authentication is performed by a series of communications exchanges between the RADIUS client, also referred to as the Network Access Server (NAS) and the authentication server (or AAA Server). When a user initiates access to a resource that requires authentication, the system requesting the authentication is the Network Access Server. It will send an Access-Request to an AAA Server, which contains the user's name and password, type of connection (port), NAS identity, and a message Authenticator. The password is hidden using a method based on the RSA Message Digest Algorithm MD5. If the server cannot be reached within a certain amount of time the request will be forwarded to an alternate server. When an Authentication Server receives such a packet, it will check whether the NAS is permitted to send access requests. If this validation is successful the user's credentials will be passed on to the appropriate authentication method. The AAA Server then sends either an Access-Reject or an Access-Challenge response back to the NAS, depending on the authentication success. These response packets may contain a text message, which will be prompted to the user for informational purpose. If the NAS supports challenge/response and the Access-Challenge packet contains a text message, it will be displayed and the user is prompted for a response. Then the RADIUS client re-sends its Access-Request including the user's response. The AAA Server may respond with an Access-Reject, another Access-Challenge or an Access-Accept message. The latter will only be sent if the AAA Server is satisfied with all supplied data from the Access-Request and Challenge-Response messages. All configuration information will be encapsulated in the Access-Accept packet. Optional accounting procedures may take place now. After this process the user is granted access to the

desired resource by the NAS.

Another key feature of the RADIUS protocol is proxy support. All AAA Servers may act as either a forwarding or a remote server. When a NAS sends an Access-Request packet, it can be forwarded to a remote server by the local RADIUS part, which will receive the reply from the remote AAA Server. Then this reply will be passed back to the client. This feature is important when it comes to designing an authentication infrastructure for wireless networks. Such a proxy functionality allows to roam between different access points without to authenticate locally.

For wireless docking networks RADIUS can be used in conjunction with IEEE 802.1X as an AAA (Authentication, Authorization and Accounting) infrastructure. Such an architecture includes the following parts. A mobile client seeking network access, a wireless access point (AP) that acts as an NAS, thus handling the message exchange between then mobile client and the hot spot controller. Both devices, namely the AP and the client's wireless network card must be 802.1X compliant. Finally, an EAP enabled RADIUS server is needed which acts as an AAA server.

When a mobile client associates with the network it provides its user credentials to the access point, which then passes them along to a RADIUS server for verification. The AAA server is responsible for the authentication, authorization and accounting functions for each client request. This server also helps to centralize public key operations, which are being used by some 802.1X variants (like PEAP and TTLS).

2.4 Diameter

The RADIUS protocol has some limitations as well as some vulnerabilities that are caused by the protocol design (and by poor implementation). In order to eliminate those shortcomings, the Diameter [11] protocol has now been introduced as a RADIUS replacement. Along with the protocol some applications has been defined as well, which support specific types of network access scenarios.

The following limitations of RADIUS caused the development of the Diameter protocol. Each attribute carried in a RADIUS message is defined by a variable-length 3-tuple (Attribute Type, Attribute Length, Attribute Value) with a maximum attribute length of 255. A Diameter attribute is encapsulated in a 5-tuple (Attribute Type, Flags, Attribute Length, Vendor-ID, Attribute Value) allowing an maximum attribute length of 16581375. Furthermore the RADIUS protocol uses a one byte identifier field, which is used to recognize retransmissions, thus allowing 256 messages between a RADIUS client and server. Diameter is designed to allow more than 4 billions messages. RADIUS is built upon UDP whereas Diameter uses TCP and SCTP (Stream Control Transmission Protocol) [12] as a transport layer protocol. This enables a server to regulate the data flow from its clients. The RADIUS protocol on the other hand is designed with merely no error handling. If a server does not respond to a request the NAS will simply forward the request to an alternate server assuming the latter is reachable. Since Diameter uses a connections oriented protocol such as TCP and its own keep-alive messages, fail-over handling is greatly improved. In case of a failure all retransmissions are done by the NAS itself, proxies do not have such a capability using RADIUS. This can be a drawback of RADIUS when it is being used in a large scale environment. Diameter handles the failures at the place where they

occur. RADIUS was implemented as a client/server protocol, which causes that all initiating messages have to come from the client. Diameter is more peer-to-peer oriented allowing server initiated messages as well. The RADIUS protocol is vulnerable to replay packets, which can be used for denial of service attacks. Diameter features a true end-to-end security for messages that are exchanged, which means that intermediate Diameter servers (such as proxies) cannot read any confidential information. RADIUS does not have such a security feature. The RADIUS protocol uses a shared key between to peers to encrypt the message payload regardless if the underlying protocol already ensures privacy. The Diameter protocol can secure its packets with either IP Security or Transport Layer Security.

As already mentioned before, Diameter features some applications that are built upon its own protocol. The data is carried in so called Attribute Value Pairs (AVPs), which is the same concept as in the RADIUS protocol (but somehow improved). These AVPs are used by the protocol itself as well as by the applications or even higher level programs that use Diameter. Currently, there are four different applications defined : the Mobile IP, the NASREQ (Network Access Server Requirements Next Generation), the CMS Security and the EAP application.

Diameter components are similar to those used in a RADIUS environment but they are named differently. The protocol defines clients, servers, relay-, proxy-, redirect- and translation agents. The client is the device that performs access control. This could be a NAS for example. The server is the part that handles the AAA functionality. Relay Agents route Diameter packets based on the information which is stored in the message. No information may be modified by the Relay Agents but they are allowed to add additional data. Proxy Agents are very similar but they may modify certain information hold in the message. A Redirect Agent provides other Diameter components with routing information that allows peers to resend their requests to the correct destination. Translation Agents translate Diameter messages to other protocols (such as RADIUS). This enables that other technology can be easily used in a Diameter environment.

2.5 Mobile IP

The design of the IP protocol assumes that an IP address uniquely identifies a host on the Internet. Incoming datagrams will be routed to the network this IP address belongs to and are finally delivered to the node itself. This implies that if this device changes its location and thus the point of attachment, the old IP address will no longer be valid and packets destined to it cannot be received anymore. In order to overcome these limitations the Mobile IP [13] (MIP) protocol was developed.

A mobile device on the Internet should be able to receive datagrams even after changing its location, which implies that the original IP address still must serve as a valid identification. In order to accomplish this, the MIP protocol introduces the following entities:

- The Mobile Node (MN):
  This is a host, which can change its point of attachment without altering its IP address. This is typically a notebook, PDA or a cellphone.

- The Home Agent (HA):
  This is a host, which acts as a router for the mobile node whenever it is not located in the

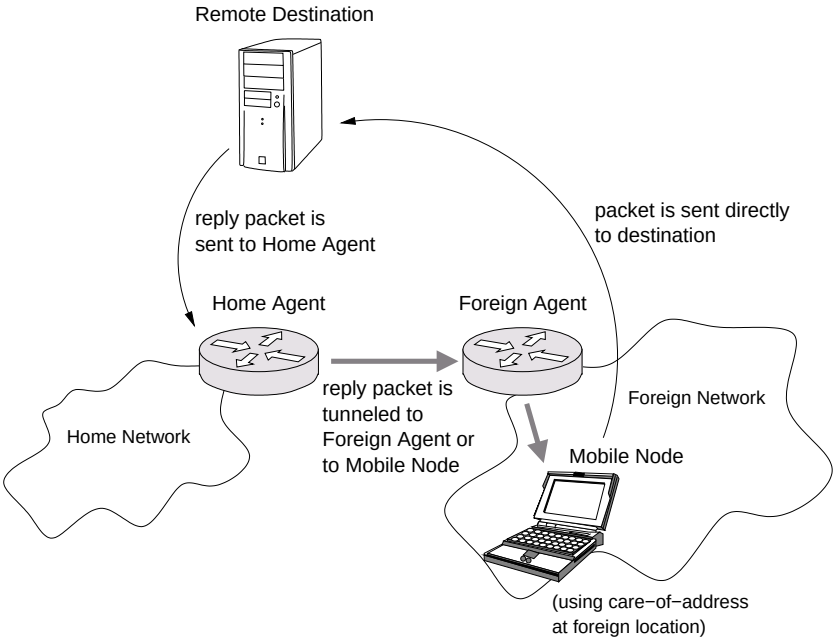

Figure 2.4: Mobile IP Operation With A Foreign Agent "care-of-address"

network its static address belongs to (the home network). All datagrams destined to the Mobile Node will be tunneled and thus delivered to the current address.

- The Foreign Agent (FA):
  That is a router on a foreign network, which may act as an IP tunnel endpoint for packets sent by Home Agents and serves as a default router for the Mobile Node.

A Mobile Node receives a static (home) IP address on its home network. As soon as the node change its location to a foreign network, a so called "care-of-address" is created and linked with the home address. All packets sent by the Mobile Node have the home address as the source and packets destined to the Mobile Node have it as the destination address in their IP headers. Due to this implementation the Mobile Node looks like a host with a static IP address to the other devices on the Internet.

The basic operation of the Mobile IP protocol works as follows and is also shown in figure 2.4. A Mobile Node associates with a new network. Then it either sends out an Agent Solicitation message or just listens passively for Agent Advertisement packets. As soon as it receives one of these messages, the Mobile Node determines if it is connected to a foreign or to the home network. In the latter case the node deregisteres at the Home Agent and operates without Mobile IP functionality. If the Mobile Node is located in a foreign network, it requests a "care-of-address" supplied either by the Foreign Agent or by an other server like a DHCP daemon. The Mobile Node then registers the new address at the Home Agent via Registration Request / Reply messages. Then the Home Agent knows that packets destined to the Mobile Node needs to be tunneled (IP-IP tunneling) to the "care-of-address". The tunnel endpoint may then be the

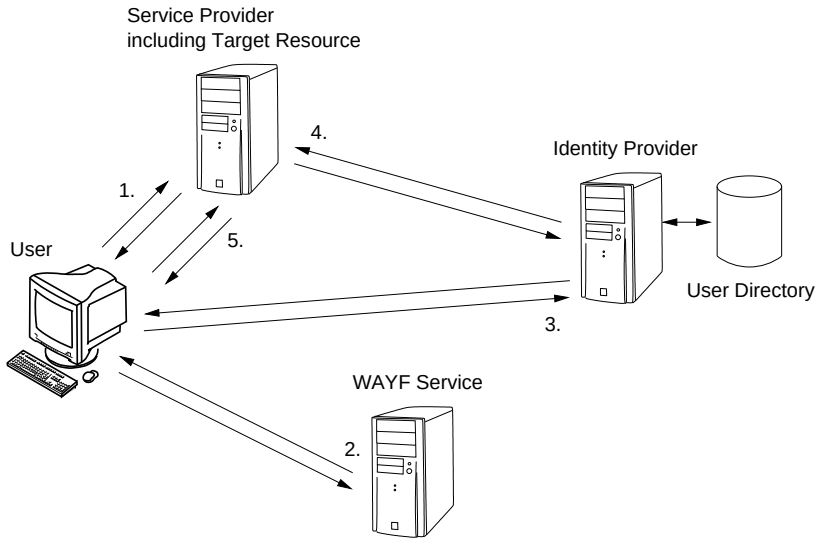

Figure 2.5: Shibboleth Authentication and Authorization Message Exchange

Mobile Node itself or a Foreign Agent located in the same network segment as the Mobile Node. Packets originated from the Mobile Node have the home IP address as the source address and are routed normally to their destination.

2.6 Shibboleth

Shibboleth [14] is a web-based authentication and authorization infrastructure (AAI) mainly developed by Internet2/MACE. A key aspect is the federated administration, which means that user identities and attributes are controlled and administrated by a single entity, the identity provider. Any resource relies on this entity to deliver information about a user in order to make authorization decisions. Also users are only authenticated at their respective identity provider, never at a resource locally. Information is meant to exchanged in a secure way using open, standard-based solutions like OpenSAML and OpenSSL.

A Shibboleth authentication and authorization procedure is shown in figure 2.5 and works as follows. If a user tries to access a Shibboleth-protected resource (also known as the service provider), it tries to authenticate the user first (figure 2.5, step 1.). Most likely this will not succeed unless the user has already visited this resource during the current session. Since the identity is unknown, the service provider part will redirect the user to the “Where Are You From” (WAYF) server (figure 2.5, step 2.). This service is maintaining a list of all participating organization and their corresponding identity providers within the Shibboleth federation. It is the only centralized service used in this infrastructure. The user then has to choose its home organization using a web front-end and immediately gets redirected to his respective identity

provider (figure 2.5, step 3.) where the authentication procedure takes place. Once authenticated the Handle Service at the identity provider creates a handle, which acts as a reference to the user and sends it to the service provider. The request to send this handle back to the resource is initiated by the WAYF server. The part, which is responsible for acquiring user handles at the service provider, is the Shibboleth Indexical Reference Establisher (SHIRE). Once the SHIRE has succeeded the impersonation checks on the handle, it passes it to the Shibboleth Attribute Requester (SHAR) component at the service provider. The SHAR then can send attribute query messages (AQM) to the identity provider to get information about this user (figure 2.5, step 4.). Based on these attributes the service provider may grant access to the resource (figure 2.5, step 5.).

While the design of the Shibboleth protocol is very nice in terms of federated administration, scalability and security aspects, it has one major drawback. The current implementation is completely browser based. This means that every user needs to use a HTTPS capable web browser in order to complete the authentication procedure since the user interaction is based on Security Assertion Markup Language (SAML) [15] browser profiles. These profiles work with HTTP redirects and HTTP POST and GET requests, hence a web browser is needed.

2.7 Eduroam

The Eduroam [16] project was initiated by the TERENA Task Force on Mobility [17], a European-wide research and education networking association. The goal was to design an international roaming solution for the different National Research and Educational Networks (NRENs) that are participating in this task force. The Eduroam infrastructure uses 802.1X as a network authentication protocol with a RADIUS based back-end on a European-wide scale. There are some top level RADIUS server where all the national RADIUS servers connect to. Each participating institution then connects its RADIUS servers to the national server. That way a hierarchical topology is built that consists of a maximum of three levels. If a user wants to get network access in an institution in foreign country, his RADIUS authentication messages will be proxied to the national NREN server fist. Since the server know that these messages are meant for another NREN, they are relayed to the top level RADIUS server where they are sent to the national RADIUS server where the user belongs to. Finally those messages are proxied to the server of his home institution. Reply packets are sent back using the same servers in the RADIUS hierarchy. Because these authentication messages may travel through a number of RADIUS proxy servers, only EAP authentication procedures are allowed that use encrypted user credentials. The list of supported EAP methods consists of EAP-TLS, EAP-SIM, EAP-TTLS and PEAP.

2.8 SWITCHmobile

This concept includes a VPN driven solution. Each organization must establish a docking network with some common properties. These include a DHCP server for automated IP configuration, no local authentication for addresses in the SWITCHmobile ACL (access control list) and no filtering for addresses outside the ACL. Additional requirements for WLAN networks

are SSID broadcast, no MAC filtering at the AP (access point) and no mandatory use of layer 2 encryption such as WEP (wired equivalent privacy). This docking network is separated from the private part of the university network and rather be placed in a demilitarized zone (DMZ). In order to access resources of the home organization a VPN tunnel has to be established between the mobile device and the VPN gateway at the home organization. If an organization wants to restrict the access of the docking network, the SWITCHmobile ACL is applied. This ensures that all VPN gateways can be reached without local authentication. This policy is executed by an access control device. Each organization has to implement this on its own, there exists no default solution.

2.9 Solution at IST (Technical University of Lisbon)

This concept also features a VPN based solution. But it differs in two major points from the SWITCHmobile concept. The network connection is always made between the mobile device and a local IPSec gateway and the authentication is based on client and server certificates. Normally this implies the deployment of a complete public key infrastructure (PKI). But they propose a more simplified solution where the client certificates do not need the non-repudiation property but instead have a short lifetime by default. This way the client private and public key can be created and maintained on a directory server at the local organization and there is no need for certificate revocation lists (CRLs). When a new mobile device is attached to the docking network it gets an IP address by a DHCP (dynamic host configuration protocol) daemon. Then the applicant is allowed to establish connections within the network segment of the access point (possibly a VLAN). Connections outside of this network are denied by a firewall by default. The only way to connect out, is to establish a VPN tunnel to an IPSec gateway. But in order to do so, a mobile user has to get the appropriate certificates from a captive web portal. For a bi-directional VPN connection two certificates are needed, one signed with the users public key and the other signed with the public key of the home organization. In addition to this certificates the user needs to obtain the private key from this portal as well (in order to decrypt the VPN traffic). With those credentials the user is allowed to establish a VPN connection to the IPSec gateway and to access the Internet. A solution for the distribution of user credentials among the organizations for the web portals is not covered by their article. A more detailed description can be found at project web page [18].

2.10 Solution at TUT (Tampere University of Technology)

This project is still in a very early phase of development. They plan to offer a combination of web-based authentication as well as a “classic” VPN solution. Mobile users are split into four different groups, namely students, staff, guests and roaming users. The docking networks are called Public Access Networks (PAN) and are considered hostile. They are separated from the private parts of the university network. Internet access is controlled by access control devices, other network parts like private intranets are protected by ACLs. When a user attaches his mobile device to such a PAN he receives a public IP address. Then the user has to start a web

browser and is redirected to a captive web portal. After entering the authentication information limited access is granted by an access control device. The user is considered logged out if the mobile device does not respond to a certain number of subsequent ping requests. Staff use a "classic" VPN connection to a specific VPN gateway possibly located in a private part of the university network. Guest users are handled almost like students. The difference is that they have to fill out some information at a guest registration page, which is located at the web portal. This application has to be approved and a validity lifetime has to be selected by an authorized person before the user gets access. Roaming user access is a combination of the student and the staff authentication. The mandatory part is similar to the students part but it is also possible to initiate a VPN connection to access the intranet. Detailed information can be found at the project web page [19].

2.11 NoCatNet

NoCat Network [20] is a wireless community network that has been started in Sonoma County, CA. The most interesting part is the NoCatAuth software. This is a complete wireless network access authentication implementation that acts as a captive web portal and as an access control device. Unauthorized users are redirected to the captive web portal where they need to authenticate themselves in order to get network access. The software is written in Perl and supports various authenticated modes. The software is meant to run on a Linux-based operating system and uses the IPTables [21] packet filter to control the network sessions.

Chapter 3

Analysis of Possible Authentication Schemes for Mobile Users

This chapter contains a collection of possible schemes for authentication and authorization of mobile Internet users in a docking network that have been analyzed for this thesis. At the end of this chapter all schemes will be evaluated against the given goals.

3.1 Open Network with Protected Resources

One possible solution is an open docking network where users don't have to authenticate in order to get Internet access. This docking network must be isolated from the private parts of the organizational network. Please take a look at figure 3.1 for a possible topology. The docking network is considered hostile because there is no control who is attached to it at a given time. Since access points (APs) used in such a docking network can be located in different physical locations like working pools, libraries and so on, it would be convenient to establish a virtual LAN (VLAN) segment for all docking networks. All APs should share the same service set id (SSID). Whether this SSID is broadcasted or not can be decided by each organization for itself. Since the propagation of 802.11 frames cannot be confined (at least on in an easy way), it is possible that the signal of an AP outside of a building is still strong enough to get associated. This way it is extremely easy for a person with a mobile Internet device to access the network from outside the building. That could be a reason not to broadcast the APs SSID even if this does not provide any security against such intruders.

Besides the common SSID the docking network requires a couple of other properties. The APs should be configured that no filtering at the MAC or IP layer is done and no mandatory use of WEP (Wired Equivalent Privacy) or any other data link layer encryption method is needed. In addition to that a DHCP (Dynamic Host Configuration Protocol) service must run inside the docking VLAN. Whether private or public IP addresses are used to configure clients can be decided by each docking network provider individually. One must keep in mind that some VPN suites like IPSec may not work through a NAT (Network Address Translation) gateway. The docking network is separated from other network segments (most likely a DMZ) by a firewall. Since we are allowing free access without authentication, some filtering is essential in order to reduce the risk of misuse. Only common used protocols or ports should be allowed to pass the

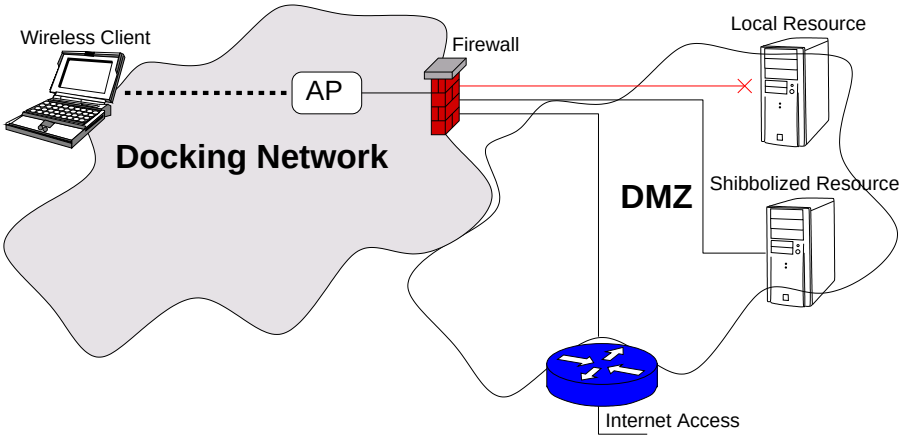

Figure 3.1: Network Topology for the Open Scheme

docking network. At least the ports 80 and 443 are needed to access HTTP and HTTPS services. Other protocols like SSH, FTP, NPP and SMB might also be of interest to the users especially for accessing local services. Another possibility is to block all outgoing packets from the docking network and enabling remote services by proxy usage only.

Restricted resources, like for example laboratories, that need authentication are handled the same way as if the user would access them from the Internet. That means authentication and authorization must be handled by an AAI infrastructure like Shibboleth.

Evaluation

This scenario actually complies with the primary goals of this thesis. Since we do not have to authenticate in order to get Internet access almost all primary goals are met. It does not matter where the mobile device is attached to the docking network; the procedure is everywhere exactly the same. Roaming is possible and no third party software is needed because we can assume that an mobile Internet device has a built-in DHCP client software. Local services are only restricted by the network topology and firewall policies. The main problem with this scenario is the security and the non existing accounting capability. Therefore the access control restrictions will possibly be very severe, which could make most local resources unusable. Because of the liability of misuse most organizations probably will never consider such a scheme.

3.2 Web-based Authentication using a Shibbolized Portal

In this solution the already working Shibboleth infrastructure is used in a combination with a captive web portal in order to grant Internet access to mobile users.

The docking network that is being used for this scheme can either be a normal LAN segment or a VLAN if it will be used in different locations. There are no restrictions on the (V)LAN topology; a portal can even be shared by different docking networks. Access points should all have the same SSID, whether it is broadcasted or not can be handled individually. The

docking network must be considered hostile and therefore rather be a part of the public (for example a DMZ) than of the private network segments and be separated from the other parts by a firewall. It should be kept in mind that firewall rules has to be changed from a remote location if a commercial product is used to control access to and from the docking network. Therefore it is recommended to use a open-source, Unix based server with a packet filter for this task, for example an OpenBSD or a Linux based system. Since this access control device is a single point of failure for the whole docking network and is a possible point of attack for hacking attempts, it should be secured as good as possible. To ensure ease of use for the mobile users, a DHCP service should be running inside the docking network. This can be done on the same physical machine as the firewall. The third service that is needed is a captive web server. Unauthorized HTTP/HTTPS requests are redirected to that server which is configured as a Shibboleth service provider. That means users will have to authenticate using the Shibboleth authentication and authorization infrastructure in order to get authorized at the portal. If this procedure is successful, the portal will trigger the firewall to grant access for the respective user. All these services can also be integrated into one device, which would act as an wireless firewall gateway (WFG).

Since Shibboleth's authentication procedure is done via HTTP redirects and HTTP(S) message exchanges between the client and the Shibboleth services, limited Internet access is already needed before any network authorization has taken place. More precisely, port 443 (HTTPS) for all Shibboleth identity providers as well as for the WAYF server has to be open by default. This means that a list of all identity providers has to be maintained on each portal. While this seems to have negative impact on the scalability of this solution (because of the N:M correlation), it is actually more of a problem of all Shibboleth service providers than of this scenario. Every service provider has to maintain such a list anyway so this access control list does not really decrease the scalability of this scheme. Upon a successful authentication the mobile user is authorized to get Internet access and the firewall applies the appropriate rules. Those rules must have a limited lifetime, which means that they will be flushed after a certain period of time in order to prevent session hijacking and other misuse. The portal itself will perform some checks on the authorized mobile Internet device in order to see if it is still connected to the docking network. These checks can consist of ping requests, ARP queries and packet counts. If a device fails to respond to these requests for a certain number of times, the user is considered logged out and the firewall rules will be flushed.

Network Access Procedure

If a new user tries to get Internet access the following procedures takes place.

- The mobile device is attached to the docking network (may be it wired or wireless).
- IP address and other network configuration options are delivered by a DHCP service.
- User opens a SSL capable web browser and is automatically redirected to the captive web portal.
- There are two main links at the portal. One for users that belong to an organization that is part of the Shibboleth federation and another one for guest users.

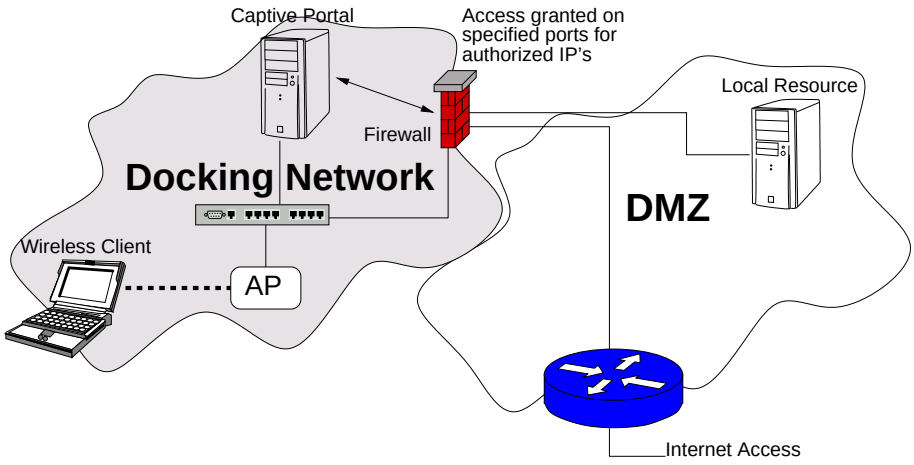

Figure 3.2: Network Topology using a Captive Shibboleth Portal

- The user follows the link, which leads to a Shibboleth protected target page (which is still on the portal server).
- AAI authentication takes place.
- Upon successful authorization the portal triggers the firewall to grant Internet access for the mobile device.
- Guest users have to fill out an application which will be granted or denied by an appropriate person. If their application is successful, they are lead to a network access page with local authentication.
- ARP look-ups and ping requests are done periodically by the access control device in order to check if the user is still attached to the docking network.
- If a certain number of subsequent tests fail, the user is considered logged out. Guest users also have a limited lifetime of their accounts and their access is denied as soon as this time period is over whether the ping requests fail or not.

Figure 3.3 shows how the authentication message exchange in this scheme takes place whenever a user needs to be authorized. The following enumeration explains the corresponding data flow on figure 3.3.

1. Unauthorized HTTP and HTTPS traffic is redirected to the captive portal. The user tries to access a Shibboleth protected service provider.
2. Browser is redirected to Shibboleth WAYF server in order to find the user's respective identity provider.

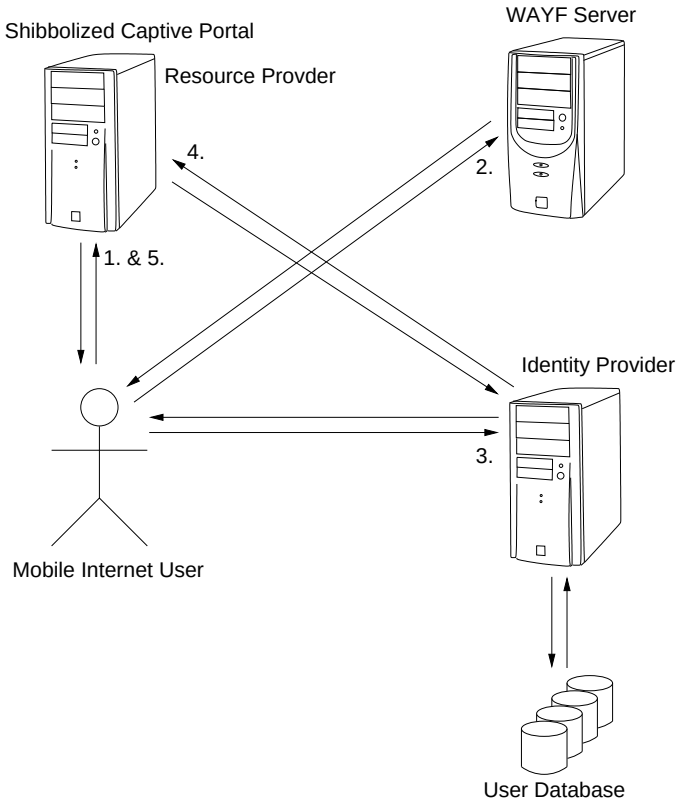

Figure 3.3: User Authentication Message Flow

3. The WAYF server redirects the browser to the users home organization's identity provider where the authentication procedure takes place (usually consists of a HTTPS login/password form).

4. Upon successful authentication the origin server creates an opaque handle for the user and contacts the service provider. The portal then requests some predefined attributes from the identity provider which are needed for the authorization process.

5. The user's browser is redirected back to the service provider and will be authorized based on the supplied attributes.

Evaluation

How does this scenario comply with the goals of this thesis? The authentication process is exactly the same at the home as well as at foreign organizations. Only a single procedure is needed to get Internet access. This already satisfies the first two primary goals that are demanded. Roaming is also possible without re-authentication as long as the user stays inside the same (V)LAN segment of the docking network. There are almost no requirements on the client side but a SSL capable web browser and a DHCP client. It is assumed that many mobile Internet

devices have those applications installed by default. After a successful authentication procedure, the user gets access using an IP address of the hosting organization. Therefore it is possible to access local services in the same manner as any other local user (but access to those resources might still be restricted by access control devices). All primary goals are met very well but security is one of the biggest concerns of this scheme. Since the packet filter rules will be based on either IP or MAC addresses, it is possible that someone could take over a running session by spoofing an already authenticated Internet address. On the other hand the administration costs are limited to the maintenance of the three services that are needed in the docking network. The scalability is mainly determined by the AAI that is being used. Another problem is that there is a special solution needed for guest users, which are not part of the Shibboleth federation.

3.3 IPSec Solution Based on Client Certificates

This scenario follows a different approach than the previous ones. It uses local VPN tunnel in order to authenticate outgoing traffic in conjunction with client certificates.

The docking network has more or less the same features than those described in the previous scenario. All APs have the same SSID, which should be broadcasted. There is a DHCP service and a captive web server running in the docking network segment. The difference concerning the infrastructure affects the firewall. In order to establish an IPSec VPN tunnel between the mobile client and the local gateway, (which can be on the same physical machine as the firewall) an IPSec stack and a ISAKMP (Internet security association and key management protocol) service is needed. Whether these services run on different servers or all on the same host can be decided by each organization individually.

Like in the previously described schemes, the IP address configuration is done automatically by the DHCP server and client. The user has to open its web browser and is redirected to the captive portal where the “usual” AAI authentication takes place. If this procedure is successful, a local SSL client certificate and key is generated on the portal and installed on the mobile client, which will be used for the IPSec authentication phase. The portal provides the user with all the information that is needed in order to configure its IPSec endpoint. Then it is possible to establish an IPSec tunnel between the mobile Internet device and the gateway. The firewall rules will pass any IPSec packets.

Network Access Procedure

The procedure to get network access for the user looks as follows.

- The client device gets associated with the docking network and is provided with IP configuration information from the DHCP service.
- The user opens web browser and gets redirected to the captive portal.
- Shibboleth service provider authentication and authorization takes place.
- A signed client certificate will be created and passed back to the user.

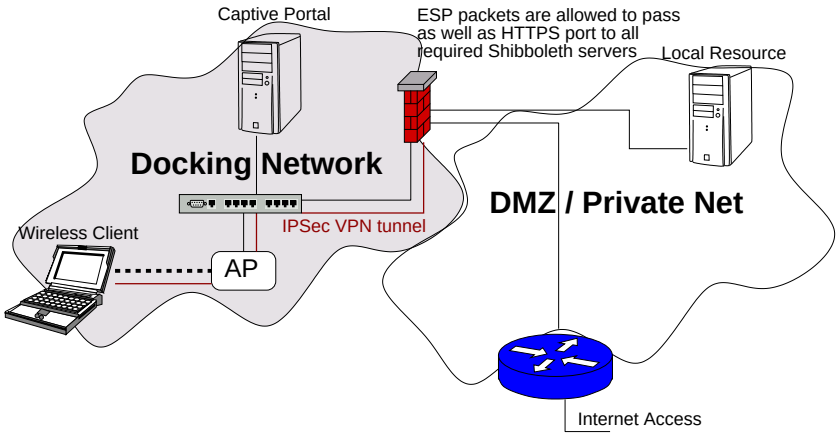

Figure 3.4: Network Topology of the IPSec Gateway Scheme

- The user has to configure its IPSec client in order to make it work with the provided gateway. This only needs to be once and then this configuration can be used on any docking network.
- If the IPSec policy is configured correctly, it should apply as soon as the user tries to connect to the outside of the docking network. Client and gateway will establish an IPSec tunnel and the traffic is allowed to pass the firewall.

Evaluation

The goal of a single and authentication procedure with the same credentials regardless where the mobile client is attached is satisfied. As long as the client certificate is valid, the user can roam around without re-authenticate all the time. A problem can be the software that is needed to establish a IPSec VPN tunnel on the client side. Modern operating systems like Windows 2000/XP, Linux and *BSD offer such programs by default (although they might not be perfectly compatible in a cross platform setup). Smaller devices like PDA's might be missing such software at all. But the software itself is not the only problem. The configuration of such a tunnel can be rather extensive and can conflict with already existing VPN configurations. In addition to that, a second tunnel to the user's home organization is not easy to establish. Therefore this scenario can cause conflicts with the already deployed SWITCHmobile infrastructure. Local services can be accessed since the IPSec tunnel ends at the hosting organization and the client has a local IP address. The advantage of this scenario is that the IPSec framework has proved its reliability and security for years.

3.4 802.1A using EAP-TLS and Client Certificates

The last scenario introduces a new authentication and authorization method; the Extensible Authentication Protocol Transport Layer Security (EAP-TLS). Like the IPSec based scheme, SSL

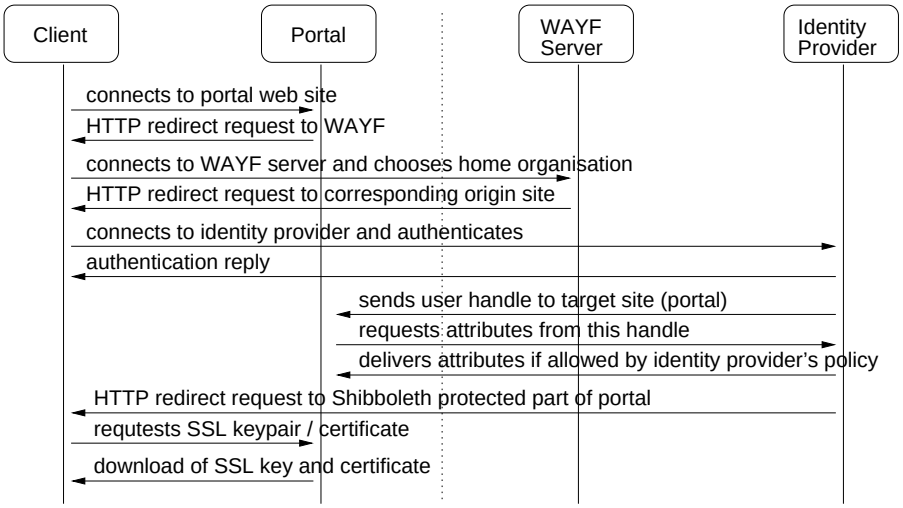

Figure 3.5: User Authentication Message Flow using the IPSec Scheme

client certificates are used to authenticate users as well but it introduces a central certification server. The docking network needs 802.1X capable access points in this scenario. They must be able to act as an EAP authenticator (Cisco Aironet, Orinocco APs are capable of that). Since we do not want to store the authentication information on each access point, we need a special server for that. This can be done by a Radius authentication service. This server does not need to be inside the docking network but must be accessible by the APs.

When a new client attaches to the docking network it gets associated to the AP first. Then a EAP initiation phase with the AP (authenticator) takes place. The access point forwards EAP-TLS requests to the local authentication server (TLS capable Radius server). If this is successful the client gets an EAP success message from the AP and is authorized to use the docking network. Such a EAP-TLS solution normally implies a fully functional public key infrastructure (PKI). But this can be simplified by using certificates signed by the certification server. This way no certificates have to be bought for the clients.

Unfortunately this authentication method is not fully supported by many operating systems yet. Windows XP / 2000 comes with a native client but for Unix based systems there is no such feature at this time. There exist some clients for Linux and BSD systems but they are not yet in a productive and stable state.

Network Access Procedure

When a new user tries to get network access using this scheme, the following procedure applies.

- The user needs to access the AAI protected certification server in advance.
- A signed client certificate will be created and passed back to the user.
- The client device gets associated with the docking network and EAP authentication takes place.

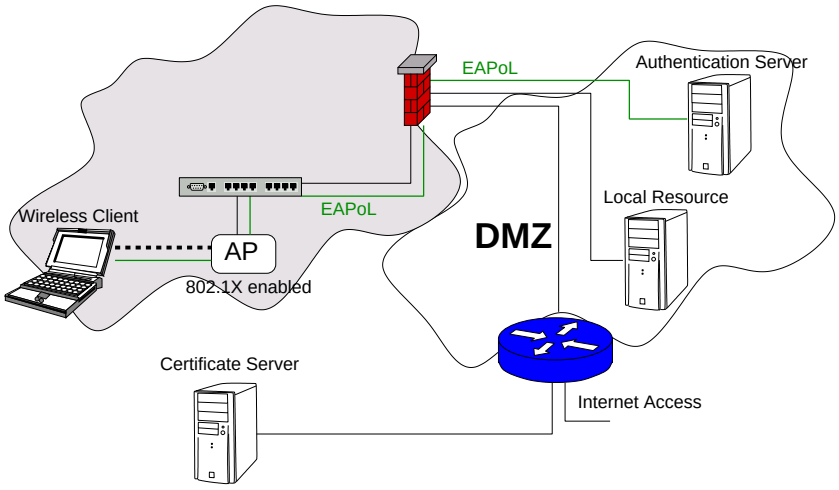

Figure 3.6: 802.1X Based Network Topology

- The AP forwards the EAP-TLS authentication request to the radius authentication server.
- If successful, client gets an associated and authenticated state at the AP and network access is allowed.

Evaluation

This scenario satisfies most of the primary goals. The authentication procedure is exactly the same wherever the user accesses a docking network. No user action required once the client is configured correctly. Roaming without re-authentication is also provided by this method. For Unix based OS the user has to install third party software in order to act as an EAP-TLS client. Local services can be used like any other local user. 802.1X may be still vulnerable to some session hijacking and man-in-the-middle attacks [22] but this scenario is considered a little more secure than the firewall based authentication solutions.

3.5 Rewriting SSL Proxy

This scenario uses a rewriting SSL proxy in order to enable access to remote web servers. The proxy actually functions like a limited SSL VPN gateway for HTTP(S) traffic whereas the browser acts as a VPN client. Hence this scenario is limited to HTTP and HTTPS services only. But unlike any other VPN solutions, absolutely no client installation or configuration is needed.

The docking network has the same properties as described in section 3.2. The server which does the proxying needs to be configured as an Shibboleth service provider. Each HTTP and HTTPS request sent by the user will have to pass the proxy, that means no traffic from any host but the proxy server is allowed to pass. The proxy itself is protected by a Shibboleth target site

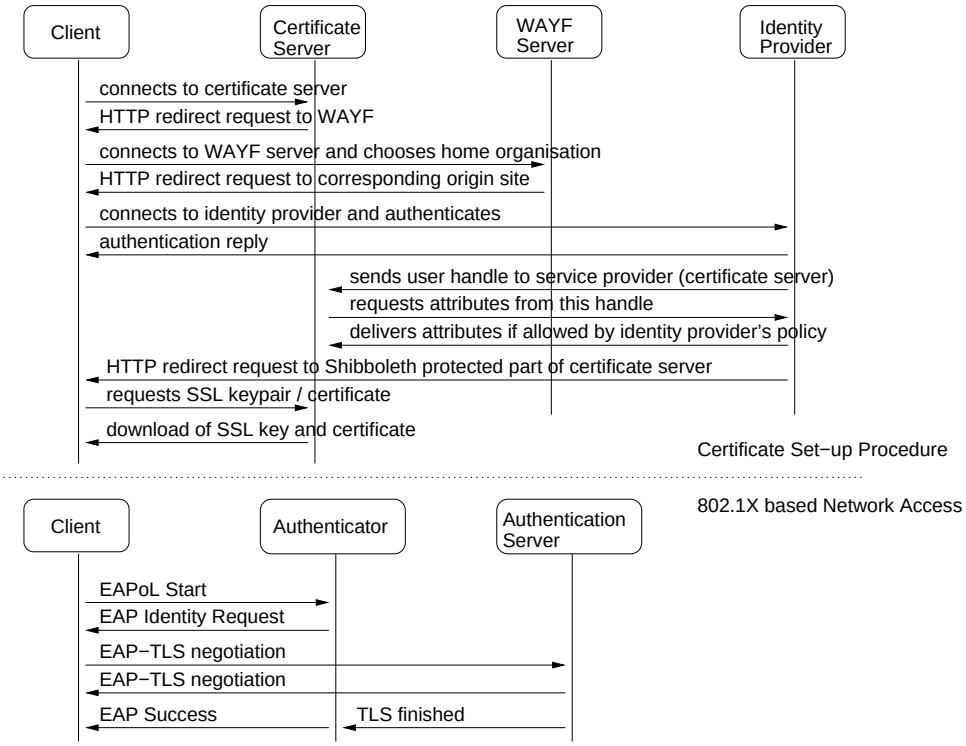

Figure 3.7: Certificate Set-up and Authentication Message Exchange

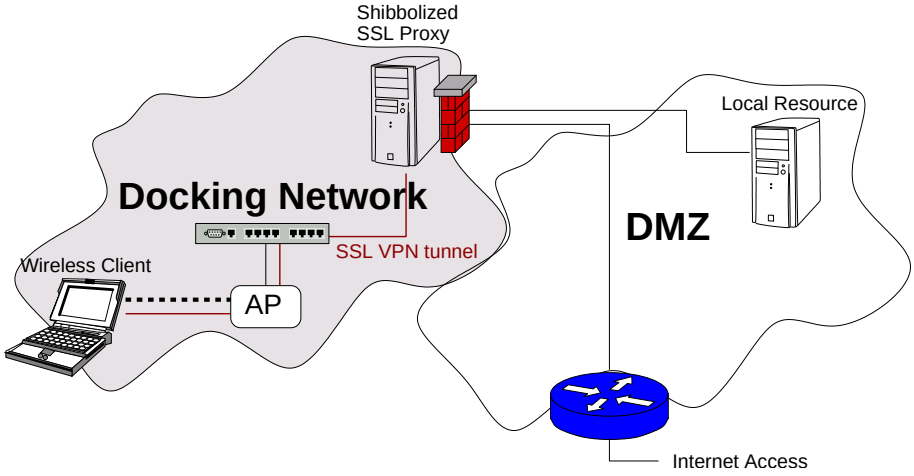

Figure 3.8: SSL Rewriting Proxy Network Topology

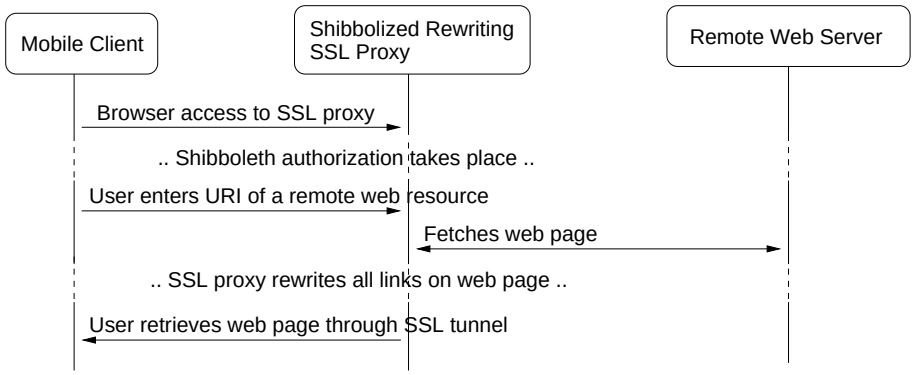

Figure 3.9: SSL Rewriting Proxy Web Resource Access Procedure

installation. Every user needs to authenticate using the Shibboleth AAI. Upon successful authorization the user is redirected to a CGI proxy script which is only accessible using then HTTPS protocol. Then the user may browse external web servers thought this SSL proxy. Unfortunately the proxy functionality is not that easy to implement. All links of the visiting web pages need to be rewritten to use the proxy instead of a direct connection. This does not only affect HTML code but also scripting languages like Javascript which are often used to change or generate HTML links on the client side. Therefore some web sites might not be displayed correctly.

Evaluation

This scenario satisfies many of the primary goals. The authentication procedure is exactly the same whether at the home or at a foreign organization. No software has to be installed or configured if it is assumed that all mobile nodes have a SSL-capable browser installed by default. Roaming without re-authentication is also possible. Access to local resources is limited to HTTP or HTTPS services. This is the big limitation of this solution. On the other hand it is very cheap to maintain and offers good security using SSL as a security framework.

3.6 Evaluation Summary

3.6.1 Third Party Software

Given on the fact that most operating systems do not have built in supplicant support for 802.1X using a common EAP authentication protocol, all schemes using that technology fail the goal of not requiring any third party software. There are systems that come with a 802.1X supplicant and an EAP client like for example Mac OSX and Microsoft Windows XP. But the Windows' EAP client only supports PEAP, MD5 and TLS so far and tests have shown that they only work together with Microsoft authentication servers. On the other hand most operating systems have free EAP clients that seem to work with other components without much hassle (for example SecureW2 and Xsupplicant).

VPN software, especially IPSec-based, is quite common in today's modern operating systems. The problem is again that the implementation of those VPN clients do not work very well if they are being used with different platforms. For example it is quite difficult to set up Microsoft's IPSec client to make it work with a Unix-based implementation. Also smaller devices like PDA's often do not come with a built in VPN software. This means that related schemes also do not fully satisfy the goal of involving no third party software.

Scenarios that are based on Web-based authentication only need a SSL-capable web browser installed on the mobile nodes. More or less all commonly used operating systems come with such a client. Unfortunately user friendliness comes at the cost of security. Such a solution would just provide access based on MAC and IP addresses (layer 3) which can be easily forged. Therefore, much care must be taken in order to avoid abuse and to the logging of such suspicious actions.

3.6.2 Authentication Procedure

The open network scheme has no initial authentication at all, thus the goal of only having a single authentication procedure is more than accomplished. Scheme two also satisfies this objective. The user has to authenticate once in order to get Internet access for the commonly used procedure from the AAI is used to accomplish that. Exactly the same applies to the rewriting SSL proxy scheme. All other schemes need an additional step by visiting a certification server from time to time in order to get a valid SSL certificate.

3.6.3 Credentials

All schemes use exactly the same credentials whether the user is at his home organization or at a foreign location.

3.6.4 Roaming

No schemes offer "true" roaming like for example MobileIP does but they all allow non-interactive re-authentication. If SSL certificates are used for authentication and everything is configured appropriately, there is no user interaction anymore to authenticate, thus roaming is possible without additional user input. The same applies to AAI based authentication schemes. Many infrastructures like Shibboleth feature non-interactive re-authentication by using session keys.

3.6.5 Local Services

All schemes use an IP address space which belongs to the local organization and therefore technically it would not be a problem to access local resources. It is more a matter of security of the docking network how access to other parts of the organization's network is granted. The stronger the security the more likely access to private resources can be given. Since IPSec offers the most secure authentication scheme, schemes four and five satisfy this goal the best. But nonetheless, access to local resources is possible with all proposed solutions.

3.6.6 Conclusion

If all these arguments are taken into account, a solution like the second one, a Shibboleth protected captive portal, meets the given goals the best. All other scenarios have some restrictions that makes them fail on some of the primary objectives. Hence the Shibbolized captive portal scheme is chosen. Even though it has some drawbacks as well, especially in terms of security, the overall functionality outmatches all the other schemes. The next chapter will provide more detailed information about the chosen solution, named Shibboleth Network Access Portal.

Chapter 4

Mobile User Authentication using a Shibboleth Network Access Portal

As already mentioned in the previous chapter, this solution is using a web-based authentication in order to grant access to the docking network for mobile Internet users. As the authentication and authorization back-end Shibboleth seems the most suitable one although other technologies like RADIUS or Diameter could be used as well. The advantage of using Shibboleth is that it is already deployed in many Swiss universities and features a web-based authentication, which is suitable for this scenario.

This chapter frequently uses the following terms:

- ACD:
  Short term for Access Control Device. This device is responsible for authorizing mobile nodes that are connected to the Docking Network to access the Internet.

- Docking Network:
  A network where mobile Internet users can attach their devices in order to get Internet access may it be wireless or wired. Normally datagrams leaving the docking network have to authorized by an ACD.

- Identity Provider:
  This is the authentication and authorization service provided by a user's home organization in a Shibboleth infrastructure. In short terms it is a location where a Shibboleth-protected resource can get authentication information as well as the user's attributes. It also provides an interface for the user to authenticate himself.

- Service Provider:
  That is a web resource that is protected by Shibboleth (also referred to as Shibbolized resource.) Only users authenticated at their home organization and authorized based on their attributes may access this resource.

- WAYF Server:
  The third component in a Shibboleth infrastructure. This is a service where a user is redirected to if its home organization is unknown. The acronym stands for "Where are you

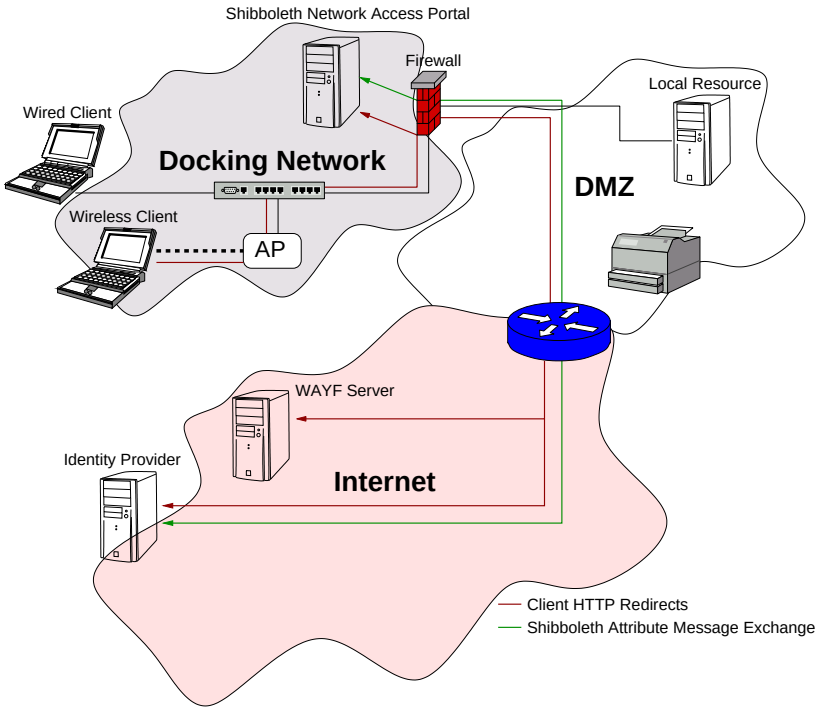

Figure 4.1: Shibboleth Network Access Portal Topology

from" Server and it provides a web interface for selecting a home organization. After the user has entered this information, he gets redirected to his respective home organization.

4.1 Network Topology

The mobile users should connect their devices in a separate network designed for this purpose, the so called docking network. This network is separated from the private parts of the organization's network. It is recommended that a IEEE 802.1Q VLAN [23] is deployed for the docking network. This has some administrative and security advantages when introducing new technologies for authenticating and authorizing users. A new separate VLAN may then be set up in parallel without physically altering of the Ethernet cabling. The Shibboleth Network Access Portal is located on the edge of the docking network acting as a router to the organization's demilitarized zone as show on figure 4.1. All data traffic is controlled by a firewall software that filters packets on the transport and network layers. The server also hosts the captive web portal, which is configured as a Shibboleth service provider to ensure only authenticated and authorized user may get access. For ease of use the portal runs a dynamic host configuration protocol (DHCP) service to automatically configure the IP address configuration settings of the clients.

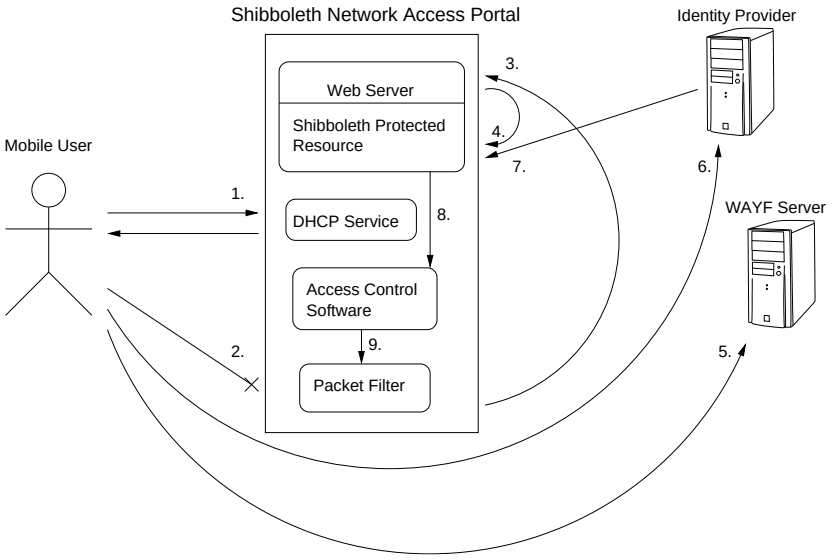

Figure 4.2: Authorization Process Sequence

4.2 Authorization Process

The complete authentication and authorization process consists of multiple stages. The first stage is the mobile client association and configuration followed by the Shibboleth authentication process. Finally the Shibboleth network access portal's authorization procedure takes place which will trigger the packet filter to open network access if successful. Figure 4.2 shows how the complete authorization procedure goes on.

1. A new client is connected to the network and sends a DHCPDISCOVER message to initiate the automatic IP configuration procedure. The DHCP service then sends IP settings information so that the clients network settings can be configured automatically.

2. Any HTTP or HTTPS traffic that passes the packet filter will be caught since the client is not yet authorized and does not have a valid network access session.

3. The HTTP or HTTPS packet will be rewritten with the destination address of the local web server. That way the client's web browser will display the captive web server part of the portal.

4. The client follows the HTML link to get network access which leads the user to a Shibboleth protected part of the web server. If the user does not have a valid session yet, the Shibboleth authorization procedure takes place.

5. Assuming the user did not have a valid session yet, he is redirected to the WAYF server where he must choose his Identity Provider. Packets destined to the WAYF server will be passed by the firewall by default.

6. The user is redirected to the identity provider of his home organization. There the user authentication procedure takes place and if successful the user is redirected back to the Shibboleth protected resource.

7. Upon a successful user authentication, the identity provider will send requested user attributes to the resource. Each user also may have an attribute release policy set up that defines what attributes are sent to which service providers. This release policy is part of the Shibboleth implementation.

8. If the user is authenticated and authorized by Shibboleth, the portal will make another authorization decision based on the supplied user attributes. This second procedure is much more fine-grained since it also allows to filter by attribute values. That way a role based access control is possible. Upon success a message is sent to the access control software of the portal that the client belonging to this session is granted network access.

9. Finally, the access control component will add firewall rules to give the sessions IP address full network access.

4.3 Access Control

Accessing the docking network does not need any authorization. Any device either associated with an access point or attached to an Ethernet port has full access to the docking network. Only packets leaving the network need to pass the Shibboleth network access portal and are blocked by default except packets used for the Shibboleth authentication process. The reason behind this is that the Shibboleth implementation is based on HTTP redirects, which requires the user to directly connect to the WAYF server and his respective identity provider during the authentication procedure. There is no way such a connection could be proxied or that the authentication messages could be relayed by an agent since the involved Shibboleth components require user interaction through a web browser. The portal is a Shibbolized resource combined with a packet filter and a captive web portal software. Figure 4.3 shows how the packet filter must be configured. HTTP and HTTPS traffic of unauthorized users is intercepted and redirected to the captive portal. There, users are directed to authenticate themselves at their identity provider. Actually, a link to a Shibboleth-protected area on the portal's web server is presented. By following this link the Shibboleth infrastructure makes sure they are authenticated and the captive portal gets all the required user attributes it needs to make an authorization decision. Based on that information the packet filter may be triggered to grant access to the Internet and various local resources. Since the Shibboleth network access portal can demand any attributes from the user's home organization (such as the organization name, unit, user role and so on) a role-based authorization procedure is possible. This means that the access is not only granted because a user is authenticated but also depends on what attributes he provides. One problem is that the user has to authenticate at his respective identity provider. This means he must be able to access to his home organization and the WAYF server already during an unauthenticated state. Therefore all Shibboleth network access portals must maintain a list of all identity providers and WAYF servers within the Shibboleth federation. This might become a scalability problem when

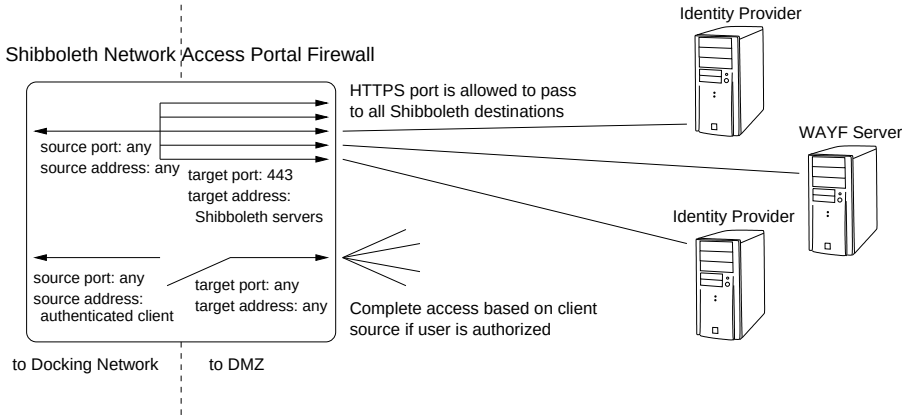

Figure 4.3: Shibboleth Network Access Portal Firewall State

there are hundreds and thousands of sites to maintain but in a Swiss-wide federation this will work without a problem since this can be done automatically. In fact the service provider part of Shibboleth already maintains such a list to establish trust relations to other components, which greatly simplifies this task since only this list needs to be parsed to get all these addresses.

Another function of the Shibboleth network access portal is to make sure that users who left the docking network get disconnected appropriately. Such a disconnect link is provided at the portal's web portal and when followed by the user all the corresponding packet filter rules will be flushed. But the problem is that you cannot rely on this manual log-out feature alone. One may forget to click such a link when leaving the network or even worse may be a victim of a session hijacking attack. Therefore a disconnect or re-authentication policy has to be applied. Each authorized access has a maximum lifetime after which a new re-authentication must be taken place. Since Shibboleth integrates Pubcookie [24] this process might be done without user interaction. This maximum lifetime can be reduced by the result of activity tests that will be run in the background. These tests may consist of ICMP requests, ARP look-ups and traffic measuring.

4.4 User Interface

When a new user wants to connect to the docking network he either plugs his device to a LAN port using a RJ45 cable or associates with an access point using the common SSID (service set identifier). Once connected the user has to open a web browser and connect to any server. Since the device is still in a unauthorized state any HTTP and HTTPS traffic is captured by the Shibboleth network access portal and redirected to the docking network portal. There two authentication options are presented as hyperlinks. The first and default link is to authenticate via Shibboleth for all users that are part of a participating organization and the second option is to authenticate as a guest user. If the first method is chosen the user is lead to a Shibbolized part of the portal which forces him to authenticate at his home organization if no active session exists. The second option is providing a local log-in mask for guest accounts. Once authenticated both

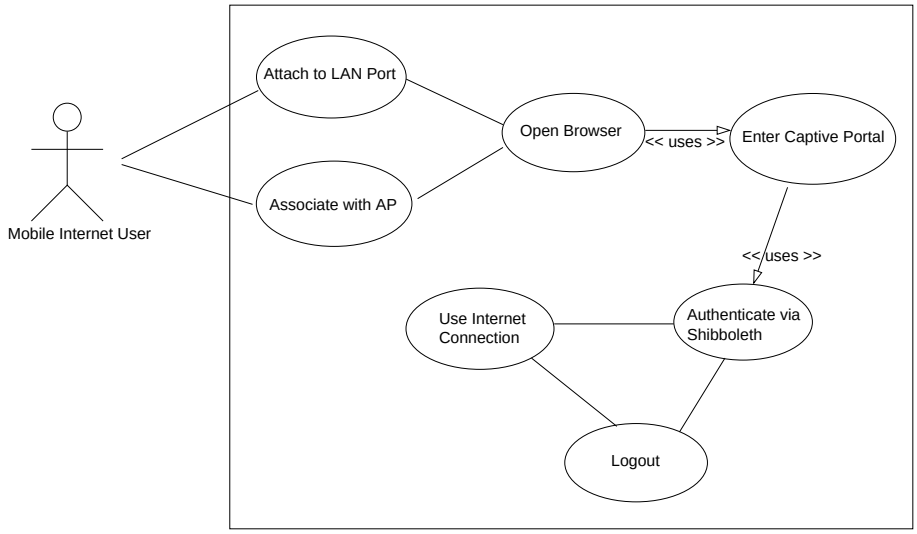

Figure 4.4: User Interaction Use Case

methods lead to a page which tells the user whether he is authorized to access the network or not. This decision is based on the configuration settings of the portal. As mentioned before access may be granted based on user attributes and not only on authentication state. If access is granted the user is presented a page with information about accessing the Internet and local resources. It also provides a link to log out when the user wants to disconnect from the network access. Otherwise Internet access may be practiced within the given rules.

4.5 Guest User Handling

People that are not member of the participating organizations should still be able to get network access if needed. There are more or less three possible solutions for this problem. One would be to add temporary accounts to a home organization's user directory. That would make them act like members of a participating organization and thus the docking network portal would not make any distinction between those two user groups. Another solution would be to define an identity provider within the Shibboleth federation that is responsible for all the guest accounts. The SWITCH VHO [25] already provides such a service. The third solution is to make the portal able to add temporary local accounts for accessing the network. These accounts have a defined lifetime after which they are deleted permanently. It is also possible to reduce the lifespan for the person who sets up such an account. This implies an administrative interface on the portal where such settings can be adjusted.

4.6 Interoperability with other Technologies

Since this scenario is not meant to be the only solution that will be used in the next years, it is more of a intermediate solution until one of the more secure, seminal authorization methods are widely implemented and adopted. Until this happens, this scenario will have its uses as a very user friendly and easy to use implementation for authenticating mobile Internet users. Since most enterprise level access points have multiple VLAN support, it is also possible to build up new solutions aside this one without any interference.

4.7 Security Concerns

As already mentioned in chapter 3, this scenario is vulnerable to some spoofing, session hijacking and man-in-the-middle attacks. Some of these threats can not be completely prevented to occur. Therefore, it would be important to passively detect attacks if possible or any other suspicious events. That way an operator could react upon such incidents and prevent any further abuse. Some more ideas about securing the user session will be discussed in chapter 6.

4.7.1 Considerations for WPA and 802.11i

Securing the communication channel between the mobile client and the wireless access point using WPA or 802.11i is not really possible at the moment. There are two ways of deriving keys in order to secure the communication. The first one is by setting up a pre-shared key. This is only viable in a local home or in an ad-hoc environment but it is useless in a larger scope. The second possibility is to establish a session key based on a 802.1X authentication. That way a dynamic per session key can be used between a single supplicant and the authenticator. Unfortunately Shibboleth needs at least limited Internet access to the WAYF server and the identity providers during the authentication phase but 802.1X prohibits any traffic except EAP messages on an unauthorized network port. There is no way of relaying or encapsulating Shibboleth authentication messages in EAP packets since the procedure is completely browser (HTTPS) based at the moment and requires user interaction. The only way to enhance this scenario with either WPA or 802.11i would be to write a new authentication protocol for Shibboleth.

4.7.2 Considerations for using a VPN tunnel

Another possible solution for securing the communication channel, is to use a local VPN tunnel between the client and the gateway of the docking network. Of course this would be possible but it would be more or less the same as the already described VPN scheme described in chapter 3. It would also require additional client software and configuration for accessing a docking network and therefore is not viable considering from users point of view.

Chapter 5

Implementation of a Shibboleth Network Access Portal

This chapter describes the proposed solution in more detail, what software is used and how the whole system is configured.

All components have been developed on a Debian [26] (3.1) Linux distribution operation system using a 2.4 kernel. The portal is using IPTables [21] to filter packets and thus limited to a Linux-based operating systems. All other software is available on various Unix-based systems and therefore it could be ported on another Unix-based platform with minimal changes.

The complete software of the Shibboleth network access portal can be roughly divided into three parts:

- Captive portal web page scripts
- Access control scripts
- Third party software

5.1 Software Components

In order to make the whole system work properly, all these components must be configured or written in a way to make them work together. First of all, lets take a look at the server and network architecture from a software perspective view. The portal server is located on the edge of the docking network acting as a firewall and gateway server. This means it needs to have at least two network interfaces, one connected to the demilitarized zone of the organizational network and the other one attached to the docking network.

As show in figure 5.1 a DHCP server is running, which listens on the docking network interface. When a new client joins the network, IP address configuration is done automatically by the server and the corresponding client on the user side. Since there is a DNS proxy service running as well on the portal, the DHCP server will provide its own IP address as DNS server option. That way the DNS port does not need to be open by default on the firewall. Aside of that the DHCP service does not use any special parameters but provide the clients with a valid IP address configuration. Along with that an Apache HTTP daemon is also bound to the docking

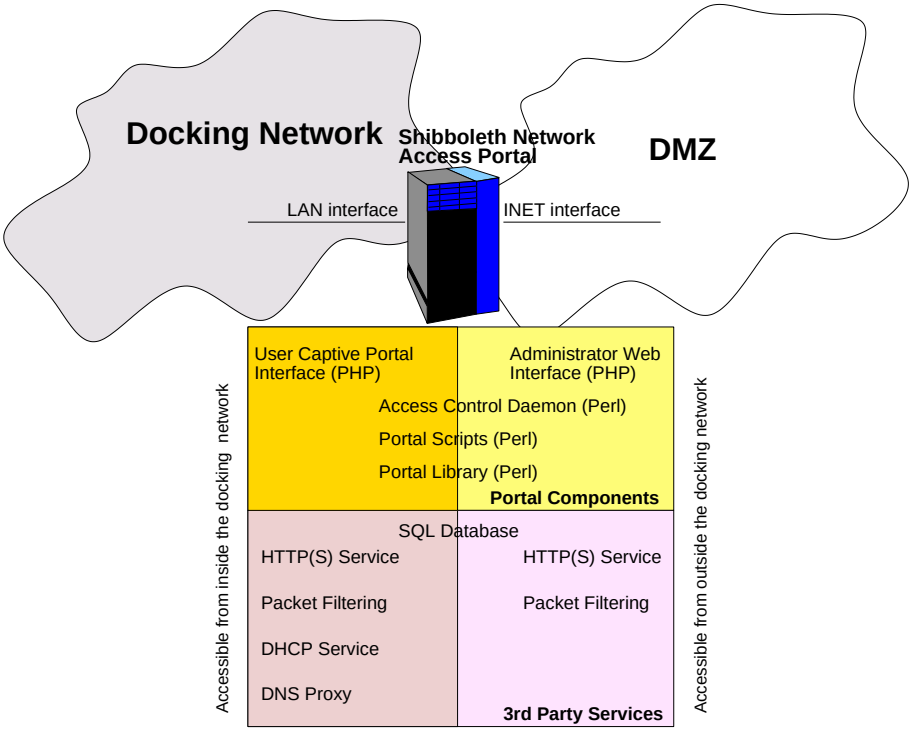

Figure 5.1: Software System Architecture

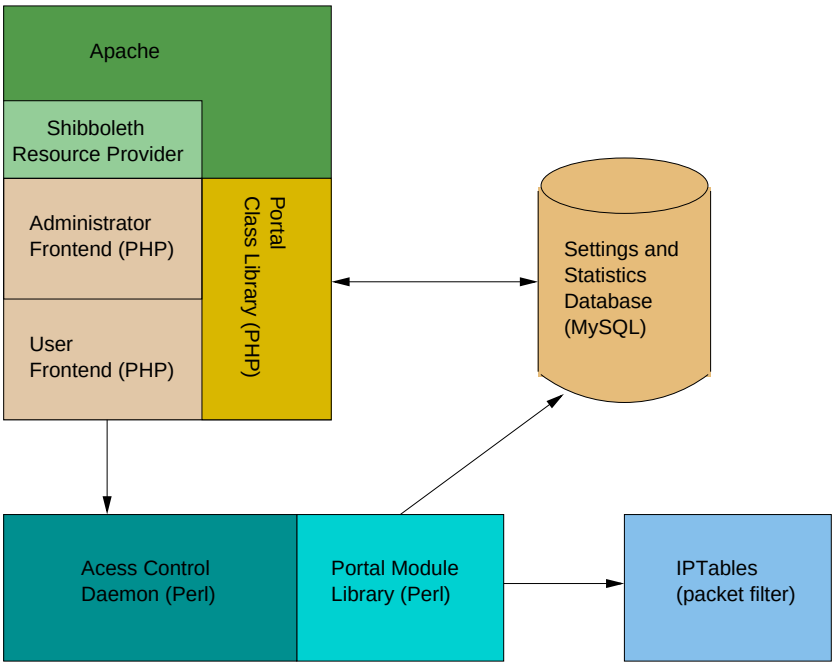

Figure 5.2: Software Components

network interface in order to provide the captive portal for the users. The administrative web page is only accessible via HTTPS on the interface which is connected to the the organizational demilitarized zone. The reason behind this separation is that only essential services run on the docking network interface in order to reduce all possible exploiting attempts. There is also a SQL database running on the portal but it is only used for internal data storage and will not accept any connections from the outside hence it can not be considered as a potential security risk. Furthermore an access control daemon and several libraries run on the portal server but they are not listening on any of the network interfaces.

Figure 5.2 gives an overview about the interaction of the software components. It shows how the PHP front-ends are embedded in the Apache web service and how the portal class library messages the access control daemon if a user has granted or denied access. Furthermore the portal module library used by the access control daemon stores and retrieves data about the current network access sessions. Finally, the portal class library also queries the database in behalf of the administrator front-end. Detailed message exchange and functions of this components are explained in the next few sections.

Figure 5.3 shows how the software components interact and exchange messages during a authorization phase. The interaction proceeds as follows.

1. The user connects to the captive web portal.
2. The PHP user front-end scripts generate a portal login welcome page.
3. The user follows a link in order to get network access. This link also makes him to proceed

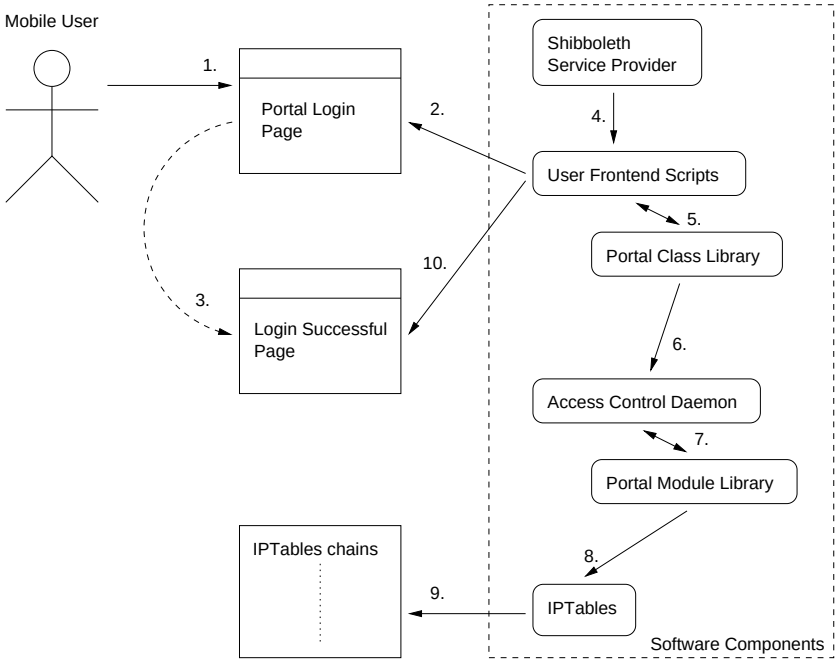

Figure 5.3: Software Message Exchange during the Authorization Phase

to a Shibboleth authentication phase.

4. Once the user is authenticated, the Shibboleth Service Provider part stores the Shibboleth user attributes in Apache environment variables where they are read by the user user front-end scripts.

5. The user front-end script passes all these attributes to a class from the portal class library, which will make an authorization decision based on these attributes.

6. If this was successful, the portal class library will trigger the access control daemon to enable network access for this user.

7. The access control daemon forwards this request to the corresponding module in the portal module library.

8. The module responsible for access control handling will execute an IPTables binary with arguments to add an entry for this client in the firewall table.

9. IPTables then will add entries in the corresponding IPTables chains that are responsible for passing packets between the two network interfaces of the Shibboleth Network Access Portal. This will grant full network access for the client.

10. As soon as the front-end script gets an successful return value from the method it has been called during 5., it will generate a login successful page for the user.

5.2 Portal Web Page Scripts

The portal web page scripts are used to generate the user interface using HTML pages served by the Apache web server. These scripts have been written in PHP and will run with version 4.2 and later. The PHP based software makes use of the PHP Extension and Application Repository [27] (PEAR) which is a framework for reusable PHP components. The language was chosen because I have already had lot of experience with coding web based application using PHP and it is very suited for web development in general.

The portal's web page scripts can be divided into three different parts, namely:

- Captive portal interface
- Administrator interface
- Portal class library

5.2.1 Captive Portal Interface

Once a new user connects to the docking network all unauthorized HTTP traffic gets redirected to the captive portal log-in page. The interface is kept as small as possible since the only function to the user is to either log in or log out. All user pages consist of the following files:

- DOCUMENT_ROOT/user/index.php
- DOCUMENT_ROOT/user/logout.php
- DOCUMENT_ROOT/user/logout_window.php
- DOCUMENT_ROOT/user/protected/index.php

The file 'user/index.php' just displays a log-in page if the user is not authenticated yet else it will display the current authorization status. Its only purpose is to inform the user about his status and present links to either log in or log out. The file 'user/logout.php' handles the log out functionality. It will terminate the user's PHP and browser session as well as it will close any existing network access for the current user. On the log-in page the user can open a small log-out and status window which will ensure that the session will remain as long as this small browser window stays open. This window is displayed by the file 'user/logout_window.php'. The last file users get in touch with is 'user/protected/index.php'. It handles the log-in procedure for getting network access on the portal. The directory '/protected/' is, as the name suggests, protected by Shibboleth and may only be accessed if the user successfully authenticates on his respective identity provider. The script itself reads all the attributes are provided by Shibboleth and store the users unique ID in the local database if it is a first time visit. The unique ID is an attribute provided by Shibboleth that, as the name already suggests, uniquely identifies a user. No other attributes are written to the database except the unique ID. All other attributes are only stored in the user's current session and will be discarded afterward. After the information has been stored or updated in the database, the script will try to authorize the user based on the provided attributes and the current access policy. If the policy and the attributes match, network access is granted and the user gets redirected back to the main page.

5.2.2 Administrator Interface

In order to adjust the portal settings or to monitor the users there is also an administrator interface, which consists of the following scripts:

- DOCUMENT_ROOT/admin/index.php
- DOCUMENT_ROOT/admin/logout.php
- DOCUMENT_ROOT/admin/database.php
- DOCUMENT_ROOT/admin/modify.php
- DOCUMENT_ROOT/admin/policy.php
- DOCUMENT_ROOT/admin/protected/index.php

The structure is also very similar to the users portal page. The file 'admin/index.php' is the main log-in and information display script. If the user is unauthorized it will redirect the browser to the log-in procedure, which is handled by the file 'admin/protected/index.php'. Only unique ID's that are marked as admins in the local database are authorized, all other users will be denied. The file 'admin/database.php' provides the logic for the database maintenance functions that are displayed on the main page. User modify functionality like granting admin privilege or forcing user to log out are handled by the script 'admin/modify.php'. Modifications to the local network access policy are done by the file 'admin/policy.php'. And finally there is also a script that handles the log-out procedure for administrators, namely 'admin/logout.php'.

5.2.3 Portal Class Library

Most functionality is not done by the portal scripts themselves but by the PHP portal library. This is an object oriented class library which provides encapsulation for the portal's core functionality. Since PHP doesn't feature very mature object oriented programming support, the library is kept pretty simple also in order to make it work with different PHP versions. The following figure shows all the classes and how they are connected to each other.

More detailed information about the Portal PHP Class Library can be found in the appendix.

5.3 Access Control Scripts

As already mentioned in the previous sections, there are some portal access control scripts written in Perl which handle the interaction with the operating system and the third party software mainly. While the PHP part of the portal software is responsible for the interaction with the users, the Perl written access control scripts are primarily used to access and modify the packet filter and the network address translation. There are five such access control scripts where the first four are mainly executed as command line programs and the last one acts as a background access control daemon:

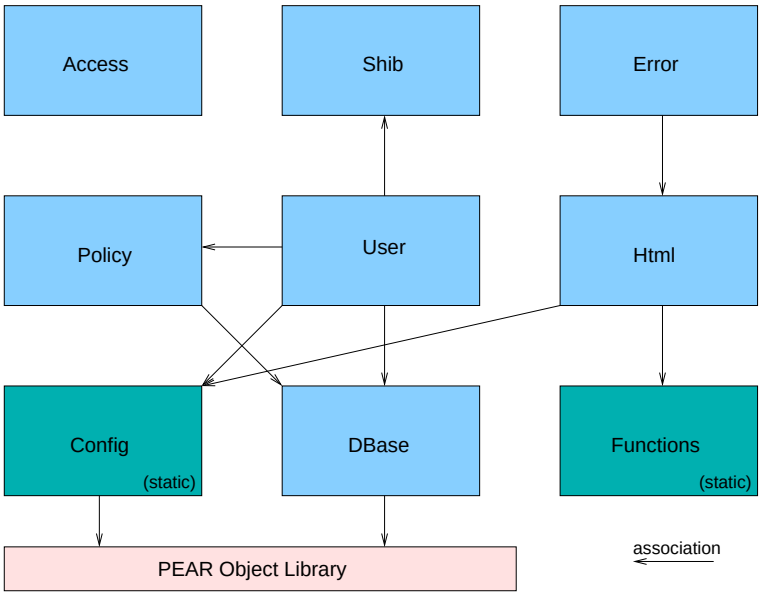

Figure 5.4: Portal PHP Class Library

- getHomeOrgIps.pl
- shibrules.pl
- addAdmin.pl
- iptrule.pl
- snapd.pl

5.3.1 Command Line Programs

The first one ('getHomeOrgIps.pl') is used to get a list of home organizations that are part of the current Shibboleth federation. The IP addresses of these home organizations will then be added to the IPTables rule by the second script ('shibrules.pl') that permits access to those sites per default. These are the only IP addresses that needs to be accessible from the docking network even for users that are not yet authorized. The 'addAdmin.pl' script is used to bootstrap the first administrator access. Since the administrator interface is also Shibbolized and hence can only be accessed by an user that is part of the Shibboleth federation, the first administrator privilege must be granted by this script. As soon as there is at least one administrator any further privileges can be handled via the administrative web interface. The fourth script ('iptrule.pl') basically just grants network access to a user. This script is also called by the PHP Access class whenever a user is authorized by the captive portal. Since this program must be run as root (because it alters the firewall table) it is recommended to use the 'sudo' package to give the PHP Access object the rights to run it as root. This is also suggested in the installation notes and there is already a sample sudo configuration, which can be used without any modification if needed.

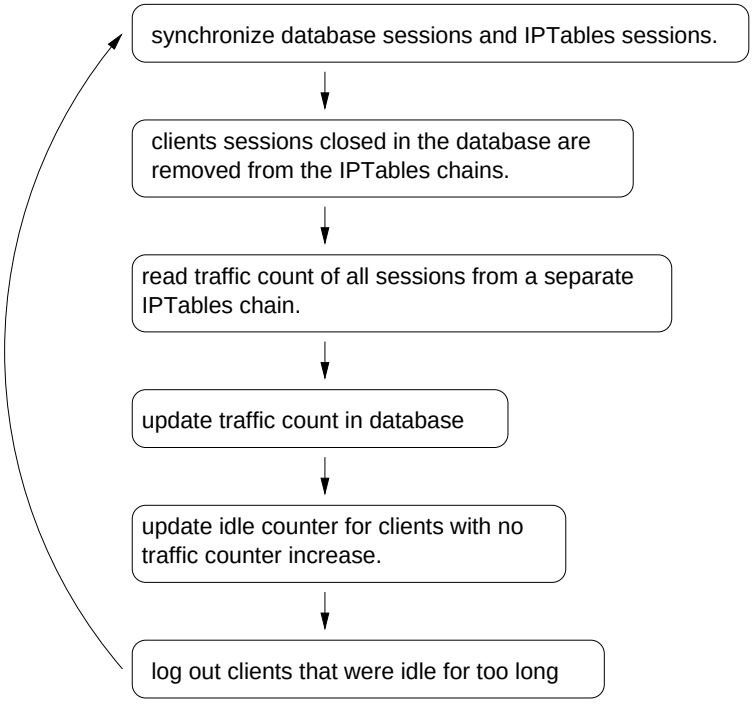

Figure 5.5: Access Control Daemon Main Loop Functionality

5.3.2 Access Control Daemon

The last access control script, called 'snapd.pl', acts as a daemon which means it detaches itself from the console and runs in the background. This daemon is the main access control program, it keeps track of all users and sessions that are currently on-line and it synchronizes the user database with the IPTables sessions in order to maintain a secure state of the access control chain. Another functionality of the script is to measure the real-time traffic that's is being generated by the user sessions. This data is used twofold, first of all it can be used to integrate accounting functionality into the portal since all network traffic is exactly measured for each user and it is also displayed in the administrator interface in a real time network graph. The second purpose of this data is that it is being used to analyze each users activity and if the user is inactive for a certain period of time, the network access session will be closed by the 'snapd.pl' daemon in order to prevent session hijacking attempts. All this access control daemon functionality is processed in one main loop as shown in figure 5.5. This loop handles all the important functions that are being used to make sure only authenticated and authorized clients with a valid session may get network access.

5.3.3 Portal Module Library

All these access control scripts use a custom Perl module library where the core functions are encapsulated. Figure 5.6 shows how the module library is built and how the classes are connected

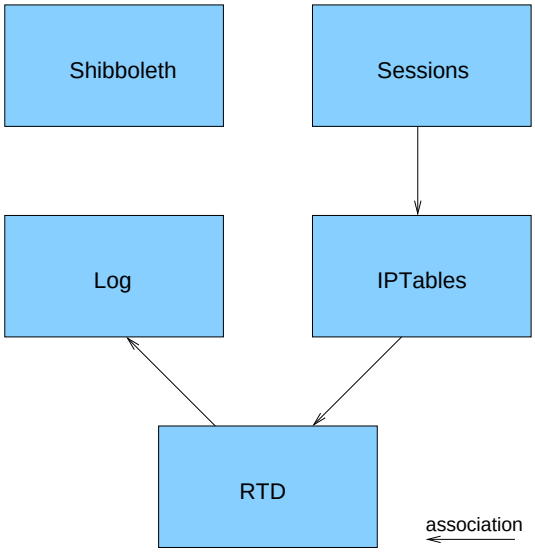

Figure 5.6: Portal Perl Module Library

to each other.

The appendix lists a documentation of the Perl Module Library and its public interfaces.

5.4 Third Party Software

In order to make the portal work as intended some third party software is also needed which have already been mentioned in the previous sections. The following presents a list of the required additional software used by the portal.

- Apache Web server
- IPTables packet filter
- DHCP Daemon
- MySQL Database
- DNS Proxy

The Web server used by this implementation is an Apache 2.0 with mod_php4. Any version from 1.3 would work also as long as it is able to parse PHP source code and can work with the Shibboleth modules. The web server acts as an interface between the users including administrators, the Shibboleth authentication infrastructure and the network access portal.

The actual access control on the portal is done by IPTables. This is a Linux specific packet filter which serves all the needs that are demanded by the portal software such as stateful filtering, network address translation and much more. One downside is that it bounds the portal

software to the Linux platform even though it would be possible to port it to other UNIX based platforms with some time and effort.

The DHCP daemon is actually not an essential part of the portal in terms of that it would not work without it. But from a user point of view it provides ease of use because the client's IP address configuration is done automatically without any interaction.

The MySQL database is used to store information about currently open sessions and is mostly queried by the administrator interface. In order to reduce redundant data and to provide data privacy only the users unique ID is store in the database. Any attributes provided by Shibboleth are only stored in the browser session and are never stored in the database. Besides the unique ID only data concerning session accounting is stored in the database, namely how many bytes has been transferred in and out and timestamps when the user has been logged in and out. This information can be looked at on the administrator interface.

The last third party software used in this implementation is a DNS proxy service. This way the DNS port (53) does not need to be open by default because the user connected to the docking network can query this service for DNS look-ups.

5.5 User Interfaces

As already mentioned in the previous sections there are two interfaces which are accessible via the web server by a browser. There is the user interface, also referred to as captive portal interface, and there is also an administrative interface. Now lets take a closer look at these web pages.

5.5.1 Captive Portal Interface

As soon as an unauthorized user tries to establish a HTTP session to a server outside of the docking network, the traffic gets redirected to the portal web server and the user's browser will display the captive portal pages. This user interface is kept very simple since the only functionality is to log out and log in. In addition to that the authentication and authorization procedure should take as little time as possible.

Figure 5.7 shows the very first screen that is presented to each unauthorized user. Only one link on this page is necessary for the users that is the one to get "Shibbolized network access". The other two links for debugging access and the administrator interface are just for convenient purpose only. As soon as the user clicks the "Shibbolized network access" link, the browser is redirected to the WAYF server and the respective identity provider where the Shibboleth authentication procedure takes place.

After this has been successfully completed, the user will be authorized by the portal based on the supplied attributes and the local policy. Once this is done the user is presented a page that tells the state of this authorization procedure as shown in figure 5.8. This state will either be access granted or access denied. On this page is a link to log out and close the session if desired but if the user would like to use this browser window for accessing external web server, it is also possible to open a small pop-up window, which will ensure that the session keeps open as long

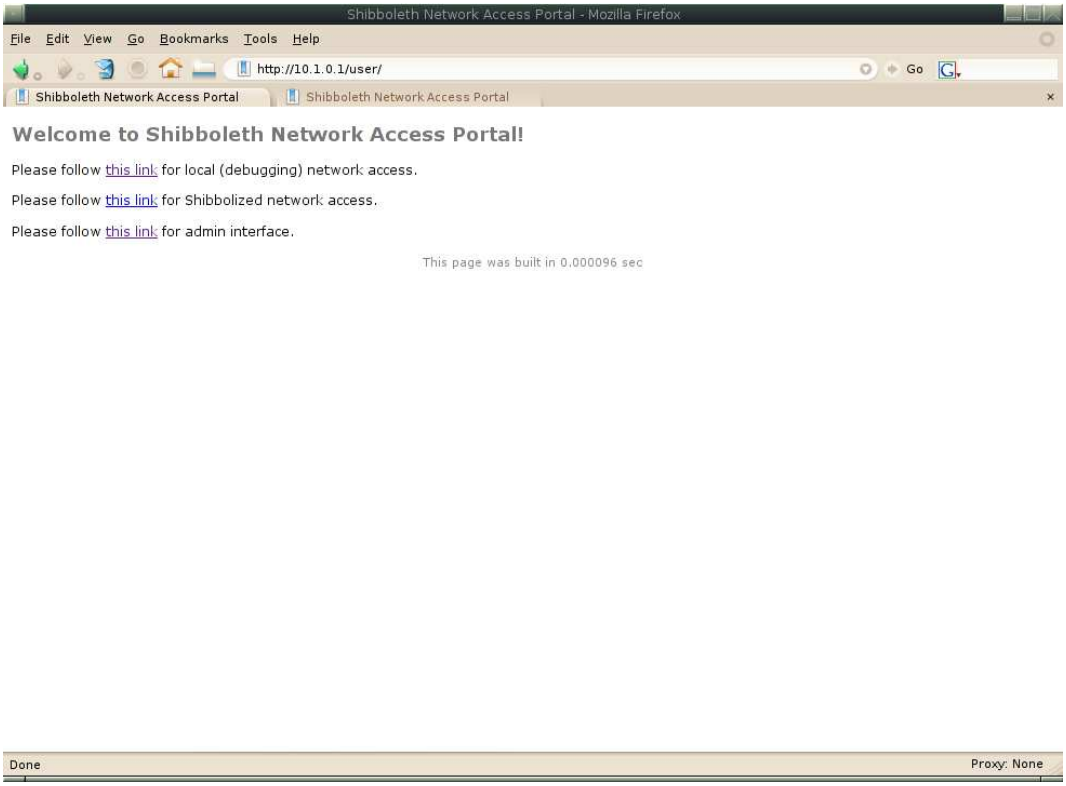

Figure 5.7: User Captive Portal Log-in Screen

as this pop-up window is not closed. These small status windows are shown in figures 5.9 and 5.10.

5.5.2 Administrator Interface

The administrative interface is only accessible for users that have the administrator privilege bit set in the portal database. This can either be granted by a command line script (used for bootstrapping the first administrator account) or via this admin web page. The following four functions are accessible by links on the left side of every page.

- list all users
- show on-line users
- configuration
- database tasks

The first link “list all users” presents a list of all registered users. That means it will show every user that has ever tried to use the docking network for Internet access. This screen is shown

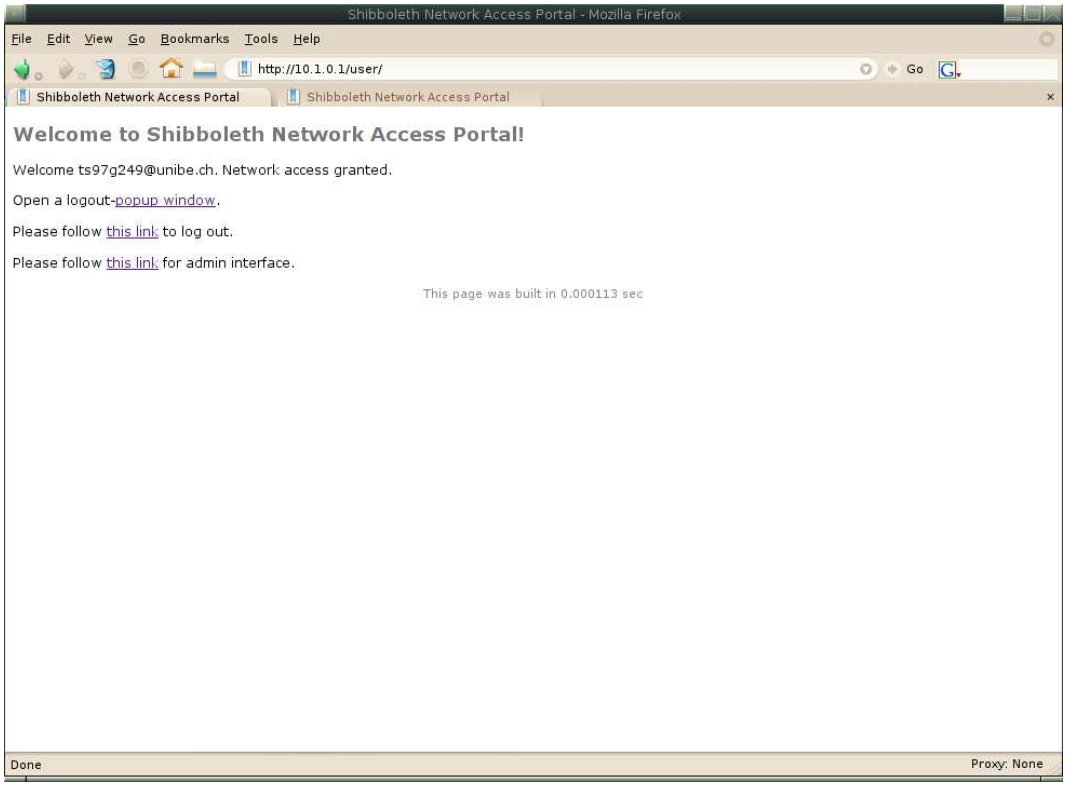

Figure 5.8: User Captive Portal Access Granted Screen

Figure 5.9: User Captive Portal Pop-up Screen

Figure 5.10: User Captive Portal Pop-up Log-out Screen

in figure 5.11. The following actions are available for each user listed on this page. First of all administrator privileges can be given or removed, secondly each user can be deleted including all the corresponding session data and lastly a link for accessing a detail user information screen is presented as well, which will look as in figure 5.13.

The second administrative section can be accessed by following the “show online users” link on the functions tab. This will present a user listing as shown in figure 5.12 containing all clients that are currently authorized and on-line including a small summery about the current sessions. The administrator can either follow a link to a more detailed information page, which also displays a real time traffic graph if that client is on-line at the moment as shown in figure 5.13. Additionally, the page shows information about the user and the session and it is also possible to force a client to log out from within this page.

The third administrative link on the functions tab is called “configuration”. This is where the fine grained policy for the user authorization can be set up. For each required Shibboleth attribute one or more rules can be defined for either granting or denying access if the rule applies.

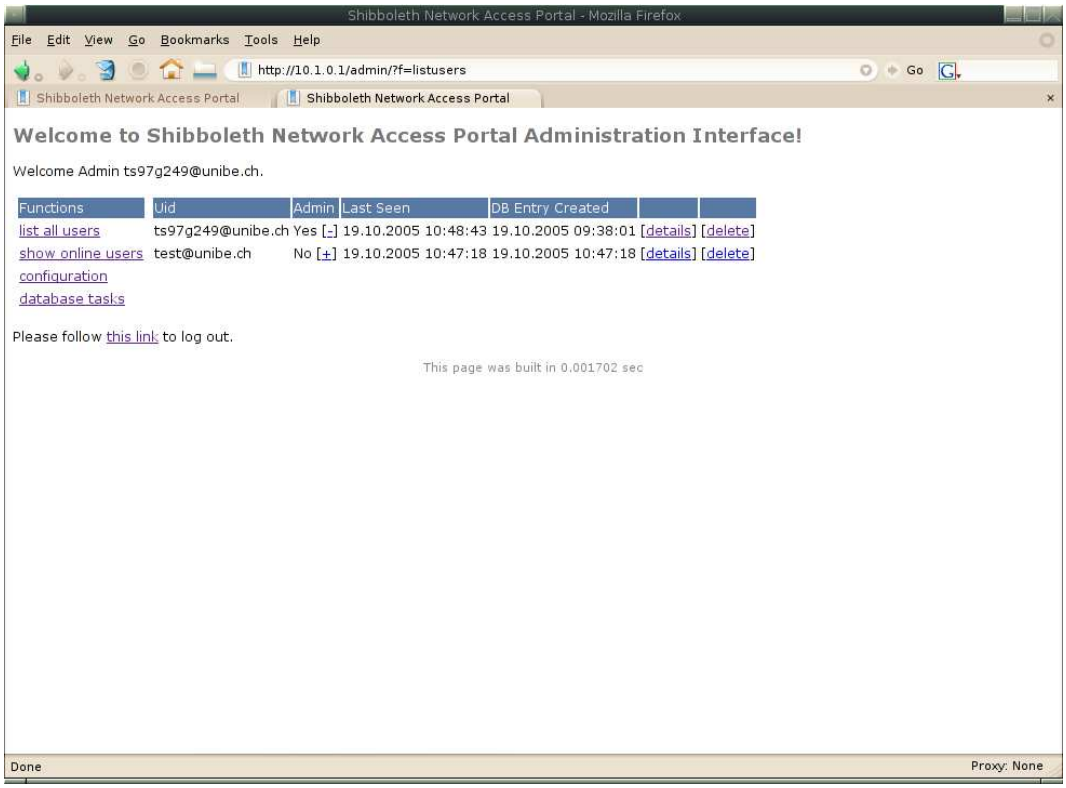

Figure 5.11: Administrator Interface List All Users Screen

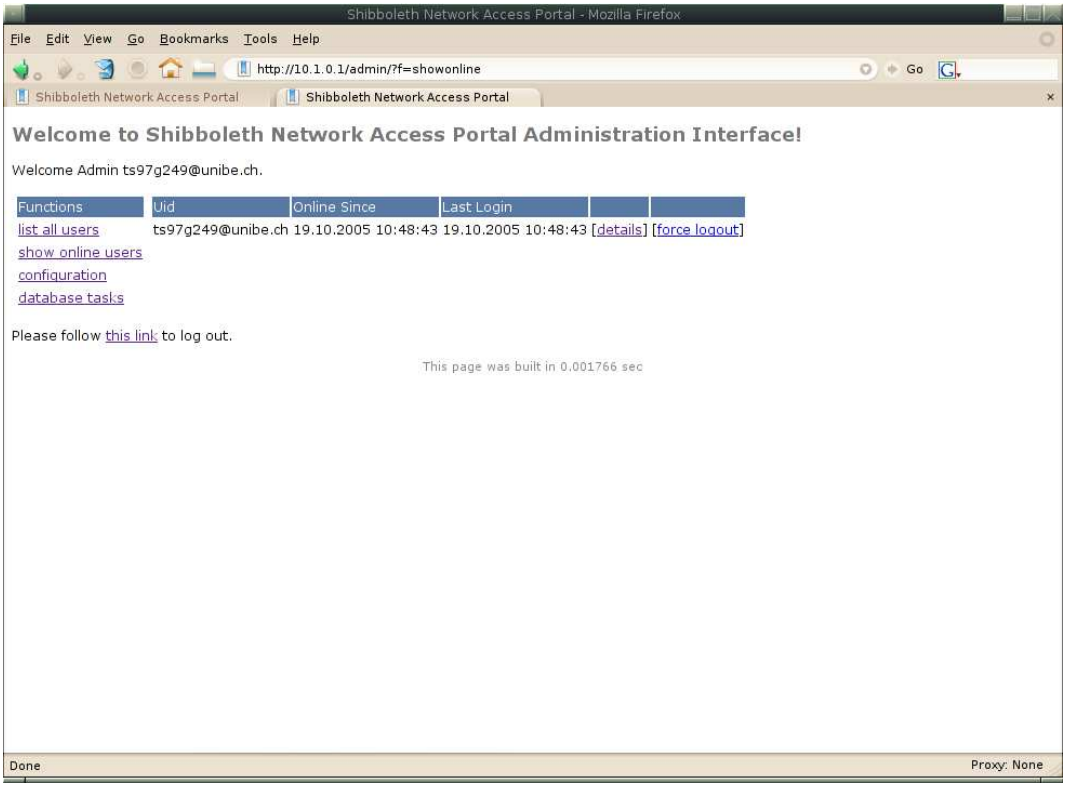

Figure 5.12: Administrator Interface Show On-line Users Screen

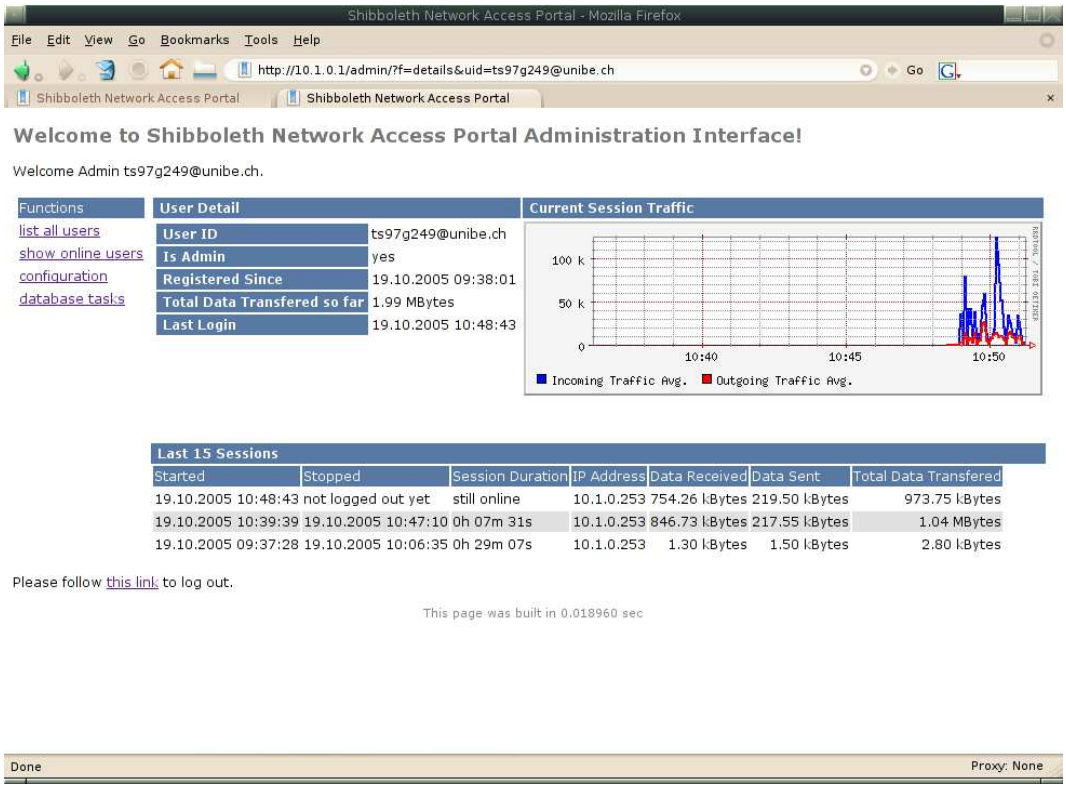

Figure 5.13: Administrator Interface Show User Detail Screen

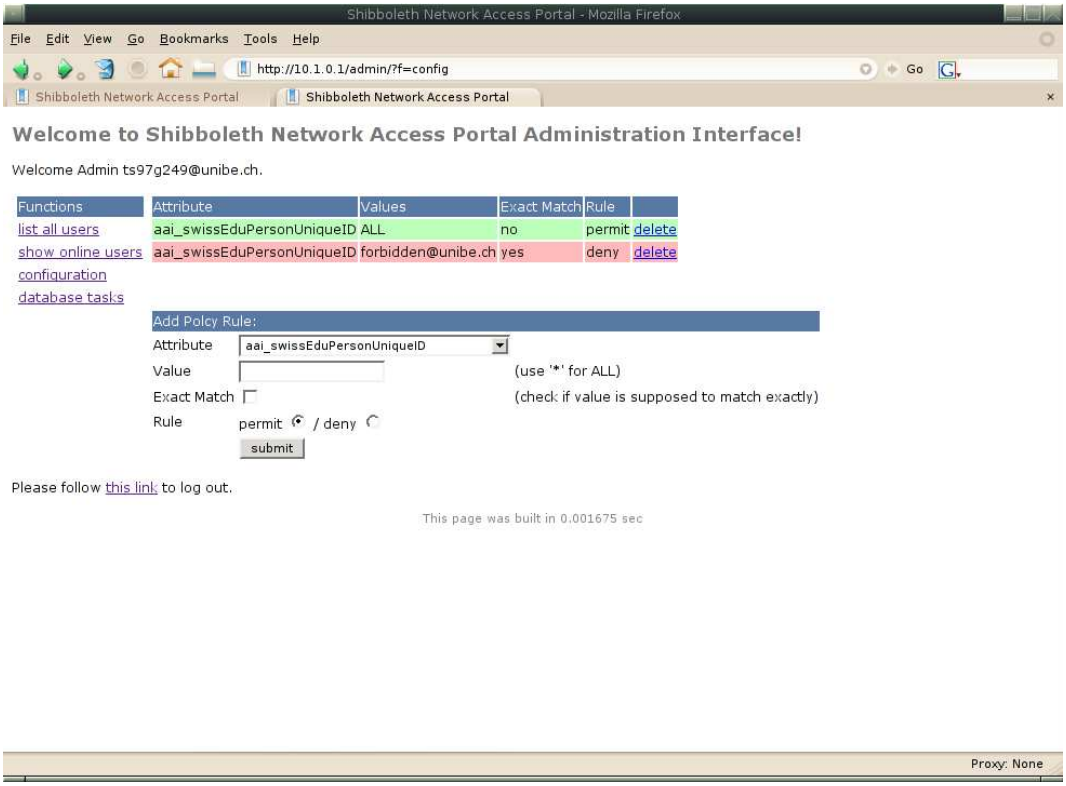

Figure 5.14: Administrator Interface Config Screen

The last administrative function is called “database tasks” and it consists of deleting old entries from the database. In order to make it possible for administrators to delete old portal database entries, may it be for privacy reasons or something else, there is just one function, which will delete old user and their session data.

Chapter 6

Performance Evaluation

Since this portal is about authenticating and authorizing users for network access, most performance tests do not apply here. It would not make sense to measure the response time the portal is delivering or the time in which the portal interface can be retrieved under heavy load. More important questions are how the access control daemon scales or how long the duration of a roaming session handover is.

6.1 Scalability of the Access Control Daemon

It is important to know how the access control daemon scales with an increasing number of concurrent clients. Upon these results one can know what hardware it would take in order to make sure that the portal is still working properly even if the whole docking network address space is covered by clients. The design of the access control daemon and the corresponding modules was done with scalability in mind, so this is not expected to be a problem. Figure 6.1 shows a scalability test of the access control daemon on a Pentium 4 running at 2GHz with 512MB RAM. The bars show how much time is needed in order process a certain number of clients simultaneously for a single loop iteration. The access control daemon's main functionality is handled in a main loop which continuously is processed until the daemon is killed. Figure 5.5 shows what functionality this loop contains. It is important that the access control daemon can handle all clients without a big delay in order to be able to react in time.

6.2 Session Handover Time

Since the proposed solution features re-authentication and re-authorization without user interaction and thus a seaming-less session handover between multiple docking network, it is very interesting to know what time such a procedure would take. One must notice that the timing is highly dependent on the underlying architecture and the involved technologies. The session handover can be divided into 4 different parts.

- 802.11 MAC layer handover
- IP layer reconfiguration

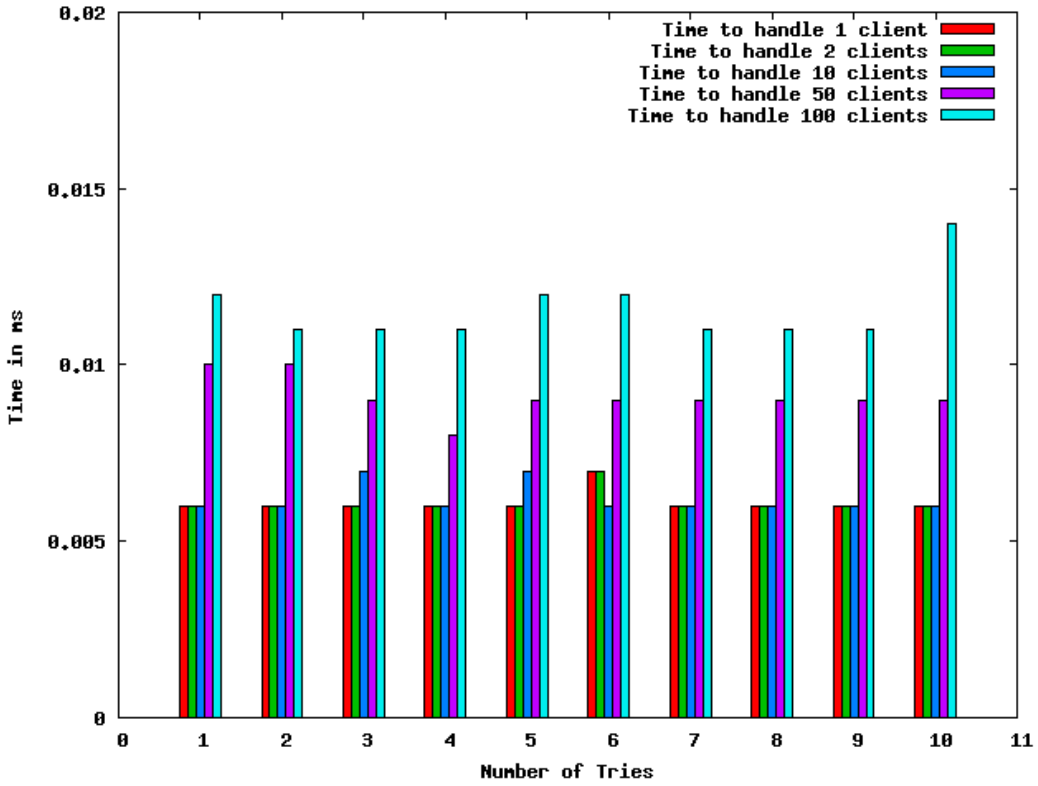

Figure 6.1: Access Control Daemon Scalability Test

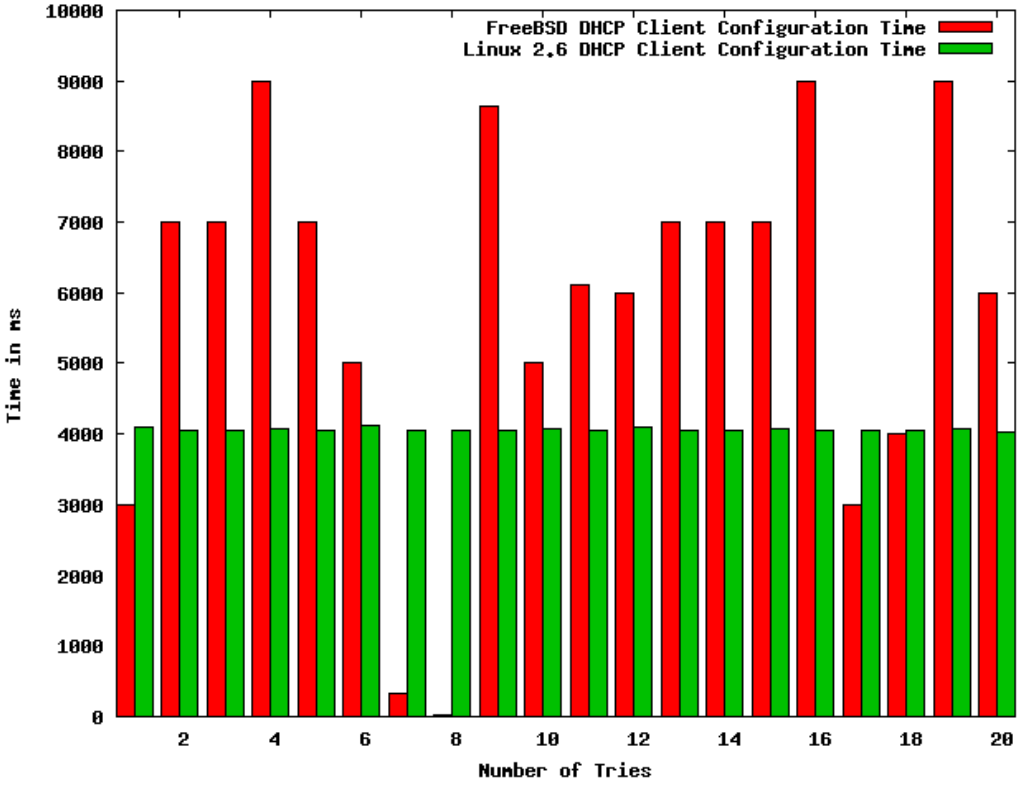

Figure 6.2: DHCP Client Configuration Times on Linux and FreeBSD

- Shibboleth re-authentication
- Portal re-authorization

The 802.11 MAC layer handover only applies if the docking network is indeed using a 802.11(b) layer of course. This measurement relies on that technology because it is the most widely used one but it up to the network operator to choose the underlying network type as long as it can be used with a TCP network stack. The 802.11(b) handover can be further divided into the discovery, search and handover execution phase. According to [28] this is also highly hardware dependent and can take from 1104ms up to 1920ms.

The second reconfiguration procedure (IP layer) is done by the DHCP protocol. In general it works as follows. The client sends a DCHPDISCOVER packet which will be responded by a DHCPOFFER packet from the server. Then the client answers with a DHCPREQUEST and if this requests is accepted by the server it will sent a final DHCPACK packet. Then the client may adjust its IP configuration according to the information that has been sent with the DHCPOFFER packet. Unfortunately this can take several seconds and the time it takes varies on different operating systems. This is shown in figures 6.2 where the DHCP configuration time is tested with a FreeBSD and a Linux client.

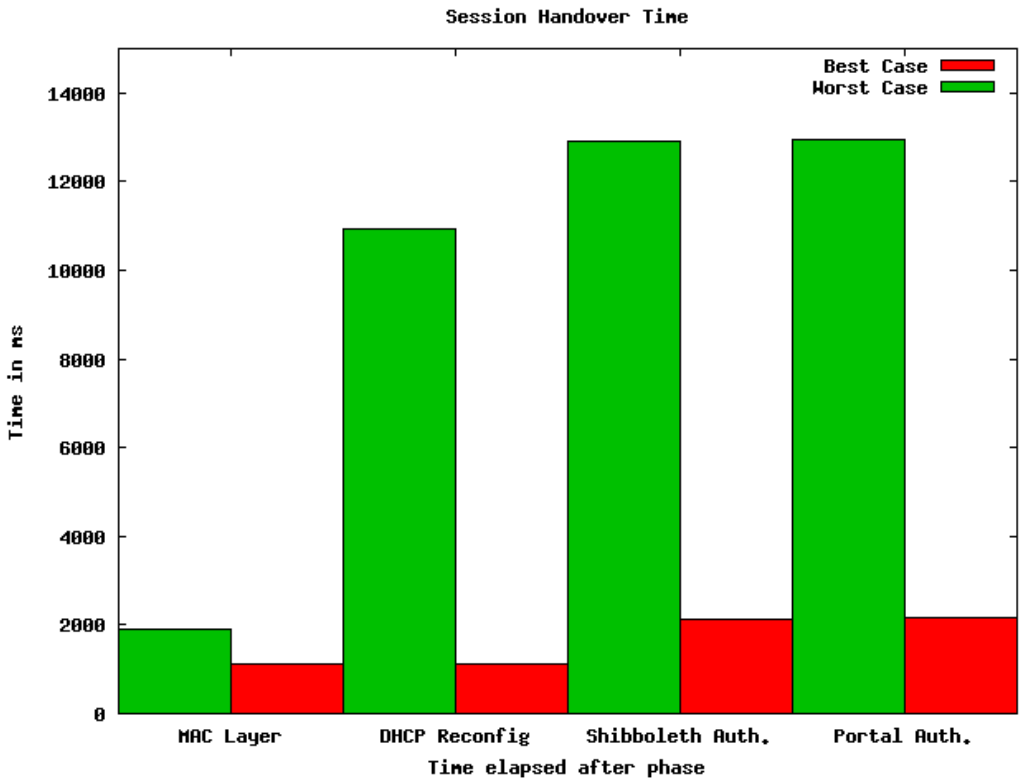

Figure 6.3: Complete Session Handover Time, Worst and Best Case

The third phase is the Shibboleth re-authentication procedure. This is based upon HTTP redirects and consists of the following messages. The client browser is forced to make a HTTP connection to the portal server and is redirected to the Shibboleth WAYF server. Then the client gets redirected further to the corresponding Shibboleth identity provider where a re-authentication takes place. Based upon the configuration of the identity provider the latter procedure can be done transparently without any user interaction. For this measurements it is assumed that this is done without any user interaction. Then the browser gets redirected back to the portal and the client is re-authorized and is granted access in the new docking network. Since this phase is handled by Shibboleth alone and the redirects may be different for several users, it is very difficult to get a general time frame for this procedure but typically this will happen within one or two seconds of time.

Last but not least the portals re-authorization is done but this can be neglected in relation to the other procedures because it will take place in under 20ms using “common” hardware. Figure 6.3 compares the best and the worst case of a session handover procedure timing after each of these four phases.

These measurements clearly show that the main time consumer of such a session handover is the DHCP re-configuration. The other parts do not play a significant role in term of time consumption.

Chapter 7

Conclusions and Outlook

As already mentioned earlier, the proposed solution has some security limitations that cannot be eliminated using these technologies in their current point of development. The main reason for that is that Shibboleth is completely based on HTTP redirects and therefore the user has to establish a direct connection to the corresponding identity provider before even being authenticated. That way limited Internet access is already needed before any authorization procedures can take place. In addition to that, the only resources that can be secured at the moment are web servers, there are no other interfaces that would make it possible to secure different resources other than web pages.

One possibility to improve those limitations would be to get rid of all those HTTP redirects that the browser has to make. One could try to implement something like a Shibboleth agent that would act as an interface to other authentication and authorization infrastructures like for example RADIUS or Diameter. This agent then would accept authentication credentials in a different form than Shibboleth provides and then would act as a client in the Shibboleth environment, doing the authentication in charge of the actual client. Then any response messages would be passed back to the connected interface. This way, one could easily connect a RADIUS based network authentication to a Shibboleth infrastructure. But there are some problems using such an agent. One point is that all the credentials needed to get decrypted and handled in clear text somehow at such a component, which is something that is not really acceptable from a privacy point of view. Another problem is that the actual authentication procedure is not really a part of Shibboleth itself. Normally it is just a web based HTML form which prompts the user for a user name and a password. But then either the agent must be aware of those authentication methods used in the federation or the identity providers must be modified to accept a generic form of authentication messages. A message flow in such a scenario is shown in figure 7.1. That way different network access procedures could be used such as any EAP based authentication protocols.

Another possibility to provide a secure network access procedure using Shibboleth would be to make it transport and relay any kind of authentication credentials. As of now, the authentication procedure is not part of the protocol itself, the only service it offers is the HTTP based WAYF component to find the right identity provider URL for each user. In order to make Shibboleth transport such an authentication procedure besides the browser based HTTP redirection, a new component is needed, called the Shibboleth authentication server. This part is respon-

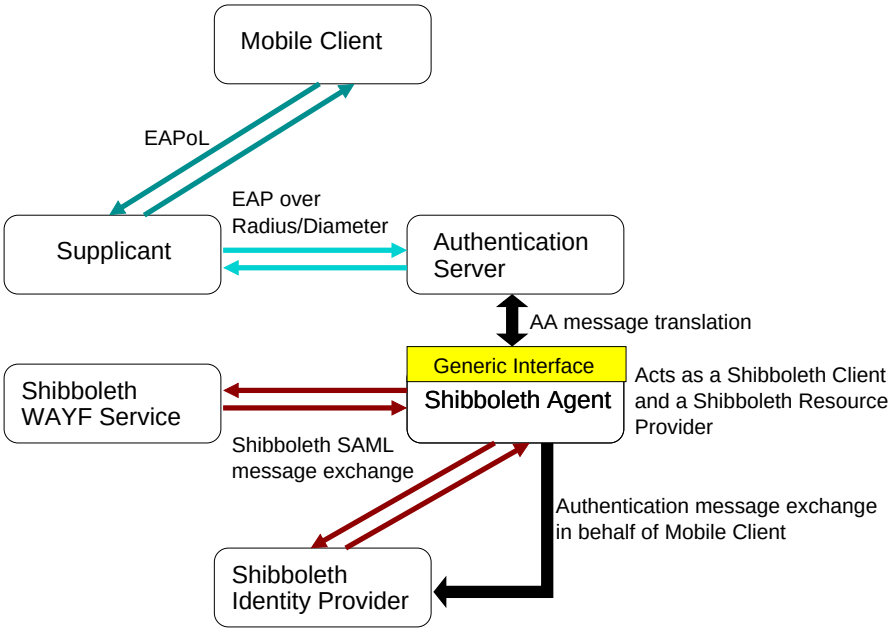

Figure 7.1: Shibboleth Agent Message Flow

sible for receiving any authentication protocol, most likely transported in EAP messages, and forwards it to the corresponding identity provider in order to authenticate the user. In order to easily integrate it in a network access scenario it would make sense that such an authentication server is able to read either Diameter or RADIUS tunneled EAP authentication messages for easy deployment in a 802.1X environment. Then this authentication server should be able to query the WAYF service where to authenticate the client if its needed. These query messages could be encapsulated into the SOAP protocol since it is already used by Shibboleth to transport SAML messages. This also implies that the WAYF service needs to be enhanced in a way that it offers such a SOAP based home organization query interface. The core components of the WAYF server does not need to be altered, only a new interface needs to be added. After querying the WAYF using SOAP objects, the Shibboleth authentication server just needs to forward the received authentication protocol to the identity provider. This authentication protocol can again be tunneled in SOAP messages if needed. It just must be taken care of the security related information that is being transported, hence the messages must be safe to any kind of altering, man-in-the-middle and other security related attacks. Therefore, it would make sense to tunnel them the same way than any other information is tunneled in a Shibboleth communication. A message flow in such an authentication procedure is shown in figure 7.2. Upon successful authentication the Shibboleth authentication server acts as a normal service provider and may request attributes about a user to use for an authorization decision. That way role based access is possible as it is in the proposed solution explained in chapter 5 and chapter 6.

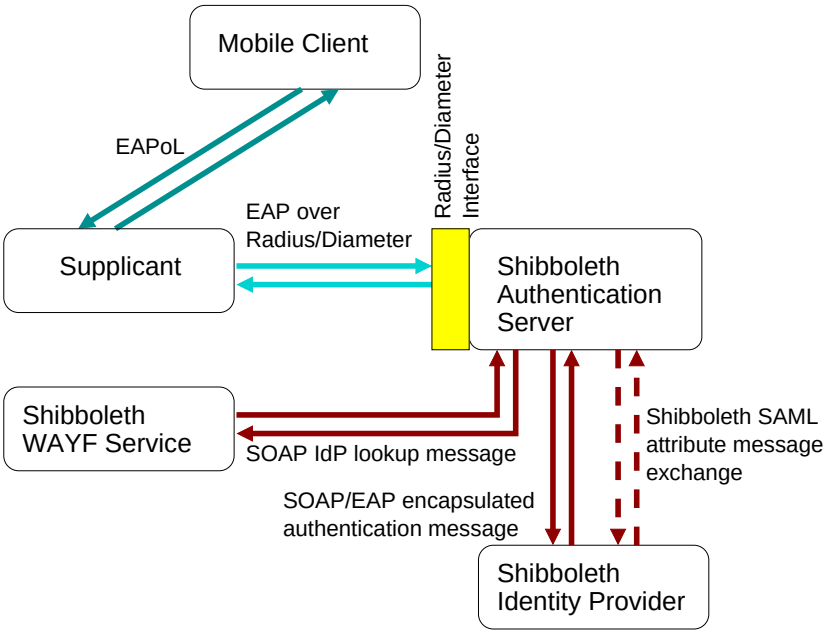

Figure 7.2: Shibboleth Authentication Server Message Flow

Bibliography

[1] SWITCH. SWITCHmobile. [Online]. Available: http://www.switch.ch/mobile/

[2] SWITCH. SWITCH Authentication and Authorization Infrastructure. [Online]. Available: http://www.switch.ch/aai/

[3] E. Rescorla. HTTP Over TLS (HTTPS). IETF RFC 2818. 2000.

[4] IEEE. ANSI/IEEE Standard 802.11, IEEE Standards for Local and Metropolitan Area Networks: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 1999.

[5] A. Stubblefield, J. Ioannidis and A. Rubin. Using the Fluhrer, Mantin, and Shamir attack to break WEP. ATT Labs Technical Report, TD4ZCPZZ, Revision 2. 2001.

[6] IEEE. ANSI/IEEE Standard 802.1X, IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control. 2001.

[7] IEEE. ANSI/IEEE Standard 802, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture. 2001.

[8] H. Levkowetz, J. Carlson, J. Vollbrecht and L. Blunk. Extensible Authentication Protocol (EAP). IETF RFC 3748. 2004.

[9] W. Diffie and M. E. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory, vol. IT-22. 1976.

[10] C. Rigney, A. Rubens, W. Simpson and S. Willens. Remote Authentication Dial In User Service. IETF RFC 2138. 1997.

[11] P. Calhoun, J. Loughney, E. Guttman, G. Zorn and J. Arkko. Diameter Base Protocol. IETF RFC 3588. 2004.

[12] L. Ong and J. Yoakum. An Introduction to the Stream Control Transmission Protocol (SCTP). IETF RFC 3286. 2002.

[13] C. E. Perkins. Mobile IP. IEEE Communications Magazine, vol. 35, no. 5, pp. 84-99. 1997.

[14] Internet2. The Shibboleth Authentication and Authorization Infrastructure. [Online]. Available: http://middleware.internet2.edu/shibboleth/

[15] OASIS. Assertions and Protocol for the OASIS Assertion Markup Language (SAML). Committee Specification 01. 2002.

[16] Licia Florio and Klaas Wierenga. Eduroam, providing mobility for roaming users. EUNIS Conference. 2005.

[17] TERENA. Trans-European Research and Education Networking Association. [Online]. Available: http://www.terena.nl/tech/task-forces/tf-mobility/

[18] Technical University of Lisbon. VPN based Network Access Description. [Online] Available: http://wifi.tagus.ist.utl.pt/description.pdf

[19] Tampere University of Technology. Solution at Tampere University of Technology. [Online]. Available: http://www.atm.tut.fi/public-access-roaming/

[20] NoCat.net. NoCatNet. [Online]. Available: http://nocat.net/

[21] Netfilter. IPTables Linux Packet Filter. [Online]. Available: http://netfilter.org/

[22] Arunesh Mishra and William A. Arbaugh. An Initial Security Analysis of the IEEE 802.1X Standard. Technical Report. Science CS-TR-4328. 2001.

[23] IEEE. ANSI/IEEE Standard 802.1Q, IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. 1998.

[24] Pubcookie. Pubcookie software for intra-institutional web authentication. [Online]. Available: http://www.pubcookie.org/

[25] SWITCH VHO. SWITCH Virtual Home Organization. [Online]. Available: http://www.switch.ch/aai/vho/

[26] Debian. Debian GNU/Linux operating system. [Online]. Available: http://www.debian.org/

[27] PEAR. The PHP Extension and Application Repository. [Online]. Available: http://pear.php.net/

[28] H. Velayos and G. Karlsson. Techniques to Reduce IEEE 802.11b MAC Layer Handover Time. KTH Technical Report TRITA-IMIT-LCN. ISSN 1651-7717. 2003.

[29] Torsten Braun and Marc Danzeisen. Secure Mobile IP Communication. Annual IEEE Conference on Local Computer Networks. 2001

Appendix A

Glossary

AAI Short term for authentication and authorization infrastructure.

Accounting Accounting measures the amount of resources that a user is using. This is mostly used for billing purpose.

ACD Stands for access control device. It is the device that makes the authorization decision for accessing a resource.

Authentication Authentication is the process that decides who a user actually is.

Authorization Authorization actually grants someone permission to do something or denies it.

DHCP Short term for dynamic host configuration protocol. It is used to automatically configure client IP address configuration settings.

DMZ Stands for demilitarized zone. It is a part of a network that has public accessible resources in it and usually isn't as secured as private parts of a network since it must be accessible from the Internet.

DNS Domain name service is the protocol used to resolv IP addresses from domain names and vica versa.

FTP File transfer protocol is an old protocol that is still being used to transfer large amounts of data.

LDAP Short for lightweight directory access protocol. It is used for accessing a directory service which is usually stored in a tree like datastructure and its optimized for reading data. It is often used for storing authorization and account data.

Identity Provider Identity providers are entities that maintain and assert authentication and authorization information about users that belong to their organization. That term is mainly used in Shibboleth and SAML environments.

IEEE The Institute of Electrical and Electronics Engineers is a non profit organization for the advancement of technology related to electricity. They also define a lot of standards which are used in computer network environments.

LAN Stands for local area network, which is an ethernet network that is not part of the internet address space.

NPP Network printing protocol is used to send print jobs to remote printers.

Perl Interpreted programming language that is heavily used in Unix based systems for administrative tasks.

PHP Short for PHP: Hypertext Preprocessor. It is a widely used scripting language that is being used for web development.

PPP Stands for point to point protocol which is mainly depricated now and is mostly used for tunneling dialup services.

SAML Acronym for security assertions markup language. It is a XML based framework for exchanging security information and it is being used by Shibboleth.

Service Provider A service provider is an entity that manages a protected resource in a Shibboleth or SAML environment.

SMB Stands for Samba, which is used for providing Windows based network services such active directory, Netbios name resolution or remote file access on Unix based platforms.

SOAP Simple object access protocol is used by Shibboleth to transport SAML messages.

SSH Stands for secure shell, which is used for opening a remote shell session over a secure tunnel.

SSID In WLAN computer networking, a service set identifier (SSID) is a code attached to all packets on a wireless network to identify each packet as part of that network.

SSL Short for secure sockets layer. This is a secure transport protocol mainly used to secure internet based communication.

VLAN Stands for virtual local area network.

WLAN Short for wireless local area network.

WEP Stands for wired equivalent privacy. It is a standard that tried to provide encryption support for 802.11 networks but it has major design flaws and is not recommended to use.

Appendix B

Source Code Interfaces

B.1 Portal Class Library Prototypes

This paragraph presents a closer look on the PHP classes and functions used by the portal library. It shows how each class and its public methods are declared and what they are for. It just shows the public interfaces of all class methods and their short documentation of the PHP portal class library.

Access class:

```
/**
 * This class handles the access control for the portal user. It is just
 * a wrapper to the functions used by the snapd.pl daemon.
 **/

class Access {

        /**
          * Grants network access to the user that belongs to the current
          * browser session.
          **/
        function grant();

        /**
          * Denies network access to the user that belongs to the current
          * browser session.
          **/
        function deny();

}
```

Config module:

```
/**
 * Reads the portals configuration file (etc/portal.conf) and stores
 * the appropriate fields in PHP global definitions.
 **/
```

```
require_once 'Config.php';

/* read and parse the config file */
$confFile = INC_PATH .'../etc/portal.conf';
$conf =& new Config;
$confRoot =& $conf->parseConfig($confFile, 'IniFile');

/* catch any errors */
if (PEAR::isError($confRoot)) {
    die('Error while reading configuration:  '.  $confRoot->getMessage());
}
$settings = $confRoot->toArray();

/* project title */
define('PROJECT_TITLE', $settings['root']['Global']['ProjectTitle']);

/* start page url */
define('BASE_URL', $settings['root']['Global']['Base_URL']);

/* database type */
define('DBTYPE', $settings['root']['Global']['DatabaseType']);

/* database type */
define('DBHOST', $settings['root']['Global']['DatabaseHost']);

/* database name */
define('DBNAME', $settings['root']['Global']['DatabaseName']);

/* database user account */
define('DBUSER', $settings['root']['Global']['DatabaseUser']);

/* databse user password */
define('DBPASS', $settings['root']['Global']['DatabasePass']);

/* aai policy attributes, please check /include/shib.inc for exact naming */
define('SHIB_POLICY_ATTRIBUTES', $settings['root']['Global']['ShibPolicyAttributes']);

/* path to the portal scripts */
define('SCRIPT_PATH', $settings['root']['Global']['ScriptPath']);
```

DBase class:

```
/**
 * Database wrapper class for PEAR DB.
 **/

require_once(INC_PATH . 'include/config.inc');
require_once('DB.php');

class DBase {

    var $db;

    /**
     * Default constructor
     **/
    function DBase();

    /**
```

```
        * Connects to the portal database using the credentials defined in
        * the portal.conf config file.
        **/
       function connect();

       /**
        * Query the database using the supplied SQL query command.
        **/
       function query($sql);

       /**
        * Disconnect from the portal database.
        **/
       function disconnect();

}
```

Error class:

```
/**
 * A simple error handling class
 **/

require_once(INC_PATH . 'include/html.inc');

class Error {

       /**
        * Displays a html error page containing the supplied
        * $msg and exit the current php script.
        **/
       function fatal($msg);

}
```

Functions module:

```
/**
 * Collection of static functions that don't belong anywhere else.
 **/


/**
 * Returns the current time with a precision of microseconds.
 **/
function getmicrotime();

/**
 * Returns a nicely formatted URL of the current PHP page.
 **/
function getcurrenturl();
```

HTML class:

```
/**
 * This class handles the HTML rendering. No other class or script
 * (except the Error class) should print out HTML code.
 **/
```

```
/* required files. */
require_once(INC_PATH . 'include/config.inc');
require_once(INC_PATH . 'include/functions.inc');


class Html {

    var $html;
    var $startTime;
    var $endTime;
    var $rrdGraphDir;

    /**
     * Default constructor
     **/
    function Html();

    /**
     * Adds HTML code to the current HTML rendering object.
     **/
    function add($html);

    /**
     * Generates HTML code for the portal page header. It will also
     * accept custom HTML header tags, if supplied.
     **/
    function header($custom = '');

    /**
     * Genereates HTML code for a common page footer.
     **/
    function footer();

    /**
     * Adds a logout link.
     **/
    function showLogout();

    /**
     * Generates HTML code for the administrator page header. This is
     * slightly different from the user's page header.
     **/
    function adminHeader();

    /**
     * Generates HTML code for the administrator page footer. This is
     * slightly different from the user's page footer.
     **/
    function adminFooter();

    /**
     * Displays all users. Like any other HTML rendering function that
     * needs to access the database, it expects a reference to a valid
     * database object as a parameter.
     **/
    function listUsers(&$db);

    /**
     * Displays a list of users that are currently online.
```

```
        * Expects a reference to a valid database object as a parameter.
        **/
        function showOnlineUsers(&$db);

        /**
        * Displays database tasks
        **/
        function showDBTasks(&$db);

        /**
        * Shows the current policy configuration and modification
        * interface. Expects a reference to a valid database object as a
        * parameter.
        **/
        function showConfig(&$db);

        /**
        * Displays the user detail page.
        * Expects a reference to a valid database object as a parameter.
        **/
        function showUserDetail(&$db);

        /**
        * Formats an integer representation of bytes into a nicely
        * formatted string.
        **/
        function formatByteString($bytes);

        /**
        * Outputs the HTML code of the current object.
        **/
        function output();

}
```

Policy class:

```
/**
 * This class handles modification and access to the fine grained policy
 * framework.
 */

require_once(INC_PATH . 'include/config.inc');

class Policy {

    var $id;
    var $db;

    /**
    * Default constructor
    **/
    function Policy(&$db);

    /**
    * Adds a new policy rule for the supplied attribute.
    **/
    function add($attr, $value, $match, $rule);
```

```
    /**
     * Deletes the policy rule with the supplied id.
     **/
    function del($id);

    /**
     * Checks if the supplied attribute/value pair is accepted by the
     * current policy or not. Returns true upon success otherwise false.
     **/
    function check($attr, $value);

    /**
     * Returns true if the supplied attribute is already covered by a
     * policy rule otherwise it will return false.
     **/
    function checkForRule($attr);

}
```

Shib class:

```
/**
 * This class handles the mapping of the Shibboleth attributes to the
 * PHP session variables.
 */

$SHIBBOLETH_HTTP_HEADER_SESSION_MAPPING=array(
  "HTTP_SHIB_SWISSEP_UNIQUEID"
         => "aai_swissEduPersonUniqueID",
  "HTTP_SHIB_PERSON_SURNAME"
        => "aai_surname",
  "HTTP_SHIB_INETORGPERSON_GIVENNAME"
        => "aai_givenName",
  "HTTP_SHIB_SWISSEP_DATEOFBIRTH"
        => "aai_swissEduPersonBirthdate",
  "HTTP_SHIB_SWISSEP_GENDER"
        => "aai_swissEduPersonGender",
  "HTTP_SHIB_INETORGPERSON_PREFERREDLANGUAGE"
        => "aai_preferredLanguage",
  "HTTP_SHIB_INETORGPERSON_MAIL"
        => "aai_mail",
  "HTTP_SHIB_INETORGPERSON_HOMEPOSTALADDRESS"
        => "aai_homePostalAddress",
  "HTTP_SHIB_ORGPERSON_POSTALADDRESS"
        => "aai_postalAddress",
  "HTTP_SHIB_INETORGPERSON_HOMEPHONE"
        => "aai_homePhone",
  "HTTP_SHIB_PERSON_TELEPHONENUMBER"
        => "aai_telephoneNumber",
  "HTTP_SHIB_INETORGPERSON_MOBILE"
        => "aai_mobileTelephoneNumber",
  "HTTP_SHIB_SWISSEP_HOMEORGANIZATION"
        => "aai_swissEduPersonHomeOrganization",
  "HTTP_SHIB_SWISSEP_HOMEORGANIZATIONTYPE"
        => "aai_swissEduPersonHomeOrganizationType",
  "HTTP_SHIB_EP_AFFILIATION"
        => "aai_eduPersonAffiliation",
  "HTTP_SHIB_SWISSEP_SWISSEDUPERSONSTUDYBRANCH1"
        => "aai_swissEduPersonStudyBranch1",
```

```
  "HTTP_SHIB_SWISSEP_SWISSEDUPERSONSTUDYBRANCH2"
        => "aai_swissEduPersonStudyBranch2",
  "HTTP_SHIB_SWISSEP_SWISSEDUPERSONSTUDYBRANCH3"
        => "aai_swissEduPersonStudyBranch3",
  "HTTP_SHIB_SWISSEP_SWISSEDUPERSONSTUDYLEVEL"
        => "aai_swissEduPersonStudyLevel",
  "HTTP_SHIB_SWISSEP_SWISSEDUPERSONSTAFFCATEGORY"
        => "aai_swissEduPersonStaffCategory",
  "HTTP_SHIB_EP_ORGDN"
        => "aai_swissEduPersonOrgDN",
  "HTTP_SHIB_EP_ORGUNITDN"
        => "aai_swissEduPersonOrgUnitDN",
  "HTTP_SHIB_EP_ENTITLEMENT"
        => "aai_swissEduPersonEntitlement"
);

class Shib {

    /**
     * Copies the values of the Shibboleth attributes from HTTP headers to
     * matching slots in the global session object $_SESSION.
     **/
    function storeInSession();

}
```

User class:

```
/**
 * This class handles all the users functionality like logging in and
 * out, adding and deleting users and so on.
 */

require_once(INC_PATH . 'include/config.inc');
require_once(INC_PATH . 'include/policy.inc');
require_once(INC_PATH . 'include/shib.inc');

class User {

    var $uid;
    var $id;
    var $db;

    /**
     * Default constructor. It will insert to supplied UserID if it
     * doesn't already exist in the database.
     **/
    function User($uid, &$db);

    /**
     * Returns the database id of the current user.
     **/
    function getId();

    /**
     * Returns all the attributes of the current user.
     **/
    function getAttributes();
```

```
    /**
     * Checks if the current user is authorized using the supplied
     * Shibboleth attributes with the current policy.
     **/
    function authorized();

    /**
     * Grants administrative privileges to the current user.
     **/
    function setAdmin($ok);

    /**
     * Checks if the user is an administrator or not.
     **/
    function isAdmin();

    /**
     * Removes the current user and all its session data from the
     * database.
     **/
    function del();

    /**
     * Inserts a new session entry for the current user.
     **/
    function sessionStart($ip);

    /**
     * Closes the current session entry in the database.
     **/
    function sessionClose($session_id = false);

    /**
     * Returns the current open session if there is one. Returns false
     * otherwise.
     **/
    function getOpenSession();

    /**
     * Static function that returns the number of administrators.
     **/
    function getNumAdmins($db);

    /**
     * Static function that checks if the supplied UniqueID is already
     * stored in the database or not.
     **/
    function checkUid($uid, &$db);

    /**
     * Static function that checks if the supplied user is successfully
     * authenticated. If the user has administrative privileges it will
     * also set $_SESSION['user_admin'] to true.
     **/
    function checkAuthentication($uid, $db);

}
```

B.2 Portal Module Library Prototypes

This paragraph gives a overview how the Perl module's public interfaces look like and what they are for.

IPTables module:

```
#
# This package handles the integration with the IPTables chains. It can
# add/modify/remove rules in certain IPTables chains.
#
package Portal::Iptables;

#
# default class constructor
#
sub new();

#
# initializes the object and sets some default values.
#
sub initialize();

#
# check if the supplied string is a valid IPv4 representation.
#
sub checkIpSyntax($);

#
# executes an iptables system command for the supplied rule arguments
# without passing it to the systems command shell for the sake of
# security.
#
sub rule(\%);

#
# checks if a rule that matches the supplied arguments already exists in
# the IPTables chain.
#
sub ruleExists(\%);

#
# builds and executes commands for granting access to the supplied
# client.
#
sub allowClient($);

#
# builds and executes commands for denying access to the supplied
# client.
#
sub denyClient($);

#
# builds and executes commands for adding a Shibboleth home organization
# destination rule.
#
sub allowDestination($);
```

```
#
# builds and executes commands for removing a Shibboleth home organization
# destination rule.
#
sub denyDestination($);

#
# builds and executes a command for adding accounting support for the
# supplied client.
#
sub addAccounting($);

#
# builds and executes a command for disabling accounting support for the
# supplied client.
#
sub delAccounting($);
```

Log module:

```
#
# A simple logging package
#
package Portal::Log;

#
# default class contructor
#
sub new();

#
# returns the supplied string without leading and trailing quote
# characters.
#
sub stripQuotes($);

#
# initializes the object and sets some default values
#
sub initialize();

#
# adds a default log entry using the supplied log level.
#
sub add($$);

#
# adds an error log entry with the supplied log level.
#
sub err($$);
```

RTD module:

```
#
# This package is used for handling real time data. This is used for
# accounting and administrative purpose.
#
package Portal::RTD;
```

```
#
# default constructor. It shouldn't be used, since all functions are static atm.
#
sub new();

#
# creates a new realtime round robin database for the supplied client.
#
sub create($);

#
# deletes the realtime round robin database for the supplied client.
#
sub delete($);

#
# updates the round robin database using the supplied send and receive
# byte count.
#
sub update($$$$$$);
```

Sessions module:

```
#
# This is the user session handling package where the core functionality
# of the portal access scripts is handled. This class is responsible for
# adding and deleting new sessions as well as for synchronizing the
# database with the IPTables chains.
#
package Portal::Sessions;

#
# default constructor.
#
sub new();

#
# returns the supplied string without leading and trailing quote
# characters.
#
sub stripQuotes($);

#
# initializes the object and sets some default values.
#
sub initialize();

#
# checks if the supplied client has an open session or not.
#
sub isOpen($);

#
# updates the byte counter in the portal database for the supplied
# client.
#
sub updateByteCount($$$);

#
```

```
# closes the session for the supplied client and perfom a logout.
#
sub logout($);

#
# returns a hash containing information about all currently open
# sessions that are stored in the portal database.
#
sub getOpenDBSessions();

#
# returns a hash containing all open IPTables sessions.
#
sub getOpenFilterSessions();

#
# reads the accounting chain for the supplied client and returns a hash
# containing the traffic count.
#
sub getByteCount();

#
# synchronizes the IPTables sessions and the portal database sessions.
#
sub synchronize();

#
# cleans up the object and gracefully closes the database connection.
#
sub cleanup();
```

Shibboleth module:

```
#
# This package handles the interaction with the Shibboleth components of
# the portal server. In order to work properly the portal software must
# be aware of all home organizations and must be able to parse some of
# the Shibboleth config files.
#
package Portal::Shibboleth;

#
# default constructor
#
sub new();

#
# initializes the object and sets some default parameters.
#
sub initialize();

#
# returns an array containing all IP addresses of all home organizations
# of the current Shibboleth federation
#
sub getHomeOrgIps();
```