Wireless LAN Authentication

DAWN (**D**hcp **A**uthentication for **W**ireless **N**etworks)

Athens University of Economy and Business

Dimitris Mistriotis

Supervisor: G. Polyzos

Table of Contents

1.0 Abstract

The following document describes an authentication scheme (DA Authentication module" for a Peer-to-Peer Wireless Network Confederation as described in relative papers.

In this scheme the client firstly acquires an IP address using a client implementation of the DHCP protocol something which can be considered generic for most modern operating systems (both in desktop and mobile areas). After acquiring IP address, the client is restricted in network level, having the power to do nothing else except authenticating. Authentication (log in) is done by using a web interface where a user name and password pair are provided. After a successful login, the previously unavailable services such as Internet usage are available for use as well as for accounting or other operations which may initiate. The usage of local resources may be stopped after the dhcp time lease expires or when a certain trigger is being pulled (such as a long time inactivity from user's part).

This project abstracts the P2P nature serving only local users until the final model is stabilized. Also it's whole structure is organized in such way that alternative methods of authentication (special treated users, signatures instead of passwords, etc) can be added easily in the future.

2.0 Authoring, contact information

This project has been deployed by Dimitris Mistriotis under surveillance of H. Eustathiou and professor G. Polyzos for Athens University of Business and Economics, at summer semester 2003. For further references questions related to this document, the source code or relative information, the following e-mail can be used: besieger@yahoo.com (Please allow up to 3 days for an answer).

3.0 Assumptions - design principles

3.1 Assumptions

Thinking from user's point of view, our desire is to provide maximum available usability and simplicity. In order to achieve this goal an attempt has been made to let users interact with various mechanisms using knowledge which they already have from previous home or office computer experience. So instead of using a typical client-server pair of applications, and therefore introducing users with a "new" program, which they need to "learn" how to use, a Web browser – CGI script pair is used instead. So required knowledge reduces to Web browser know-how, something that can be easily considered as zero effort from user's point of view, since we can be positive that most people owning a personal computer or similar equipment know how to use a Web browser.

From designer's point of view an intention to provide a "proof of concept" document instead of a "commercial product" was expressed. Main aim was to reach a working solution in tight time schedule. This affected some choices made during the process such as programming languages used. Of course someone continuing this project will be able to proceed into a full commercial-like solution with minimum effort.

3.2 Component Description

(Operating System, Programming Languages, DBMS)

First of all the chosen OS for deploying this project is GNU-Linux. This occurred for many reasons: first of all the availability of many different platforms using Linux kernel. Availability that varies from Intel based hardware to embedded devices area. So from the programmer's point of view, software developed in an easy to find personal computer, can be used with very few adoptions to completely different platforms and devices if needed. Secondly this project assumed that various different parts such as fire walling rules, information stored on databases connectivity with web interfaces and others, which had to be joined together forming a tight and well performing application. In order for that to be achieved, high demand for scripting languages which would "glue" these parts together rose from the very beginning of this project. So one way or another, a high demand for a *nix like operating system to be used

arose.

There is the argument of why choosing Linux instead of an another similar (to the purposes of the project) OS, such as NetBSD for example. The answer has to do with the high availability of information such as tutorials, articles in web-pages, books etc, as well as far more people with the ability to help if needed, which would help to solve arising problems faster than by using another OS. Also in later stages of deployment, it will be easier for to find a Linux "expert" to assist in future maintenance than a person with desired knowledge of another OS. The OS choice dictated the use of iptables as the fire walling program (interface with kernel) used in it as the standard one in this environment. But on the other hand the tools used are in one way or another available and in other OSs so the task of porting this application elsewhere is an easy one.

After choosing OS, the choice of a programming language arose. This project is concerned as a "proof of concept" one, which means that a solution to a problem is far more desired than full working code ready for the mass-market. So a fast prototyping language is needed which would help in speeding up development. Another important element on choosing programming language is good interaction with system calls and functions, as well as processing of input and output generated by various different utilities. These are some of the reasons that brought to the decision of adopting a scripting language as the premium tool for coding. Of course the host language must also be powerful enough to cope with things as database interaction and have abilities for Object Oriented programming (for good software design, maintenance) among other things such as code easy to read and understand. The choice was between Ruby and Python. The former was chosen for objective as well as personal reasons. The code is easier to read in Ruby because it reminds more of typical (Java, C++) coding conversions. A personal favor and experience in this language had something to do with choosing decision while code in Ruby seems to me more easy to read and understand (and perhaps reproduce in an another language if needed) than in everything else, that's because it's syntax resembles more traditional Object Oriented languages, but not in favor of flexibility.

Last choice was which database Software to use. Here MySQL was the best choice for many reasons. First of all a Relational DBMS is needed because Object Oriented attributes aren't needed for this project. MySQL compared with other databases in the RDBMS field is the fastest one available in the market today, has interfaces for many programming languages (including Ruby) as well as very good interfaces for using it, creating-importing backups, security features (precise ways of granting privileges) and good license scheme (free for non-commercial use, pricing which can be negotiated for commercial use), and of

course high availability of users and documentation.

So concluding we have an underlying OS which is extremely capable in many ways, while on the other hand many parts of it's operation may be altered in order to fit to what is desired for the developed application. This OS is backed up with one of the most capable RDBMS which can handle various data needed to be processed. At last but not least an Object Oriented programming language which can help into transforming ideas to working code (a) fast, (b) with high code quality, is used.

3.3 Design principles

Although as said this project has a “proof of concept” orientation, guidelines/directives and concepts of secure software design have been taken into account and followed as much as possible. Concerning the different processes the attempt to have minimum privileges as well as privilege segregation according to functionality has been taken. Only one process runs constantly with administrative (root user / superuser) privileges and this process doesn't accept input generated from external sources.

On the other hand input is being checked whenever possible in order to avoid situations such as cross-side scripting or buffer overflow attacks. Also measures against well known security breaches in the Linux world have been taken. One major issue is the effort to have easy to understand and maintain application code, commented as good as possible. By having this in hand it's very easy for someone to enhance security in the future with the minimum effort needed. This is one of the most important issues in developing secure application and it's the reason why many application or operating systems widely used fail to achieve the level of security needed by their users.

Last but not least: According to CERT statistics two out of three successful security breaches occurred because of errors which have to do with programming errors (such as those stated before). The remaining one third has to do with not correct configuration by system administrators. The only defense for this is initial secure configuration, which has also been taken into account.

4.0 Description of various components

4.1 Higher level presentation – User's point of view

In this section following a top-down approach, the internals of DAWN project will be explained. The start should be the higher level available: the end-user, human experience. From there on the whole project will be decomposed to it's parts explaining their design and how they interact with each-other. This is also the approach taken while developing the project: A human-centered concept of how things should work for the user was taken into consideration and this dictated the rest of the process. Screen shots have been taken from two commonly used desktop OSs: Windows XP and Red hat 9.0 in order to show ease of use as well as OS independence (from client's point of view).

User unknown to system (initial phase)

a. At the beginning user hasn't turned on a device or any equipment and hence is unknown to the system.

Newcomer phase

b. First of all the client-user turns on an wireless-equipped device (which fits in w-AD scheme), let's say a desktop computer. IP address is acquired using DHCP protocol, on the client's side. This is being done on the bootstrap phase of modern OSs.

c. We have entered authentication phase. From here on the user can do nothing else but authenticate, in order to use Internet or local services. For example user can't reach (via ping) www.google.com

To say the truth user can only perform single dns requests (only one host per time, not zone transfers). In order to authenticate user must start a web-browser and print a desired destination (let's say again google or aueb). Instead of seeing destination web address's contents, a redirection to login page is performed.

d. User supplies the user name and password pair provided by local

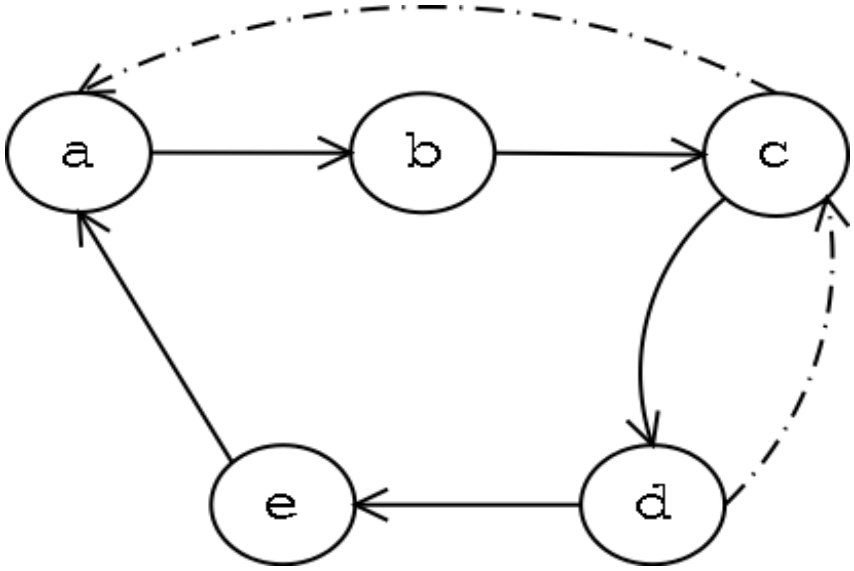
administrator in which user belongs.

If everything is OK, user sees a page saying that everything was performed correctly. After a short time period (some seconds), necessary for the system to update privileges and rights for the user, web browser will go and visit the page originally chosen by user. This is the end of the authentication phase and we enter the final one for this project.

If there is a problem (for example incorrect password),user will be informed about it with a simple page which also contains instructions of how to log-in correctly in case of a problem.

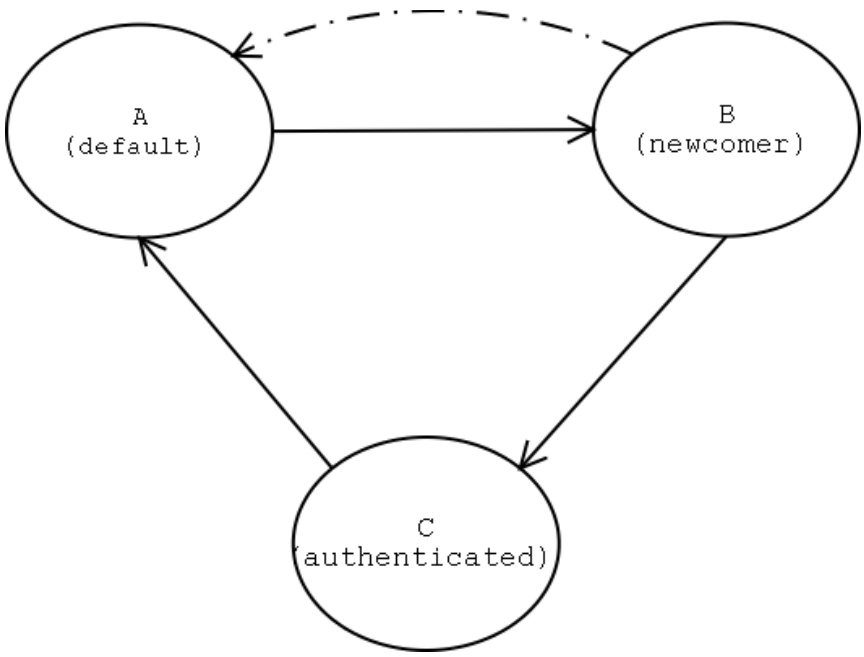
e. Now the user can view the original web page (which means that web capabilities have been acquired), as well as other Internet services.

When user leaves local-AD, something that the system will understand by not receiving a DHCP request for continuing using user's IP, all relative information is removed. Now we are again in phase (a). This may also happen if user hasn't authenticated properly and leaves local-AD. (from (d) to (a) change of states). So the state-diagram of user presented is the following one:



(here the thick lines represent “normal” usage, while the dashed-doted ones cases where user gives incorrect password or leaves AD without authenticating (c-d or c-a)

As we know this diagram can be optimized (or compressed) to a smaller, more efficient one which also represents the treatment towards user in this project:



As we can see from above:
A. By default the only thing that can be done is an issue of a DHCP request. This applies to devices which are bootstrapping as well as those who don't use their wireless devices (end of usage, shutdown, leaving AD etc).
B. From here the user may only authenticate and do nothing else except perform DNS queries, this happens as we can see in later sections because we want to capture user's destination so that user can be redirected there later. If user tries to do anything else (ftp ssh or telnet for example), he will be informed that logging in using a browser is required. Also the right to get lease time via a DHCP query is granted.
C. Here user has authenticated, so everything is open. “Everything” means access to Internet (via NAT) as well as to local machine (example for DHCP lease, as well as for future services such as statistics or whatever may arise)

This is in short the whole concept and scheme of this project. To supply an easy-to use authentication method from user's point of view, but with the maximum security and performance that can be achieved. In the next sector what happens in network layer (IP level) is discussed. The network level plays fundamental role in this project, everything else has been build around it and therefore it is the most important part of it.

4.2 Network Layer presentation

4.2.1 Examining Linux's fire walling capabilities from project's point of view.
(Short introduction to IP tables)

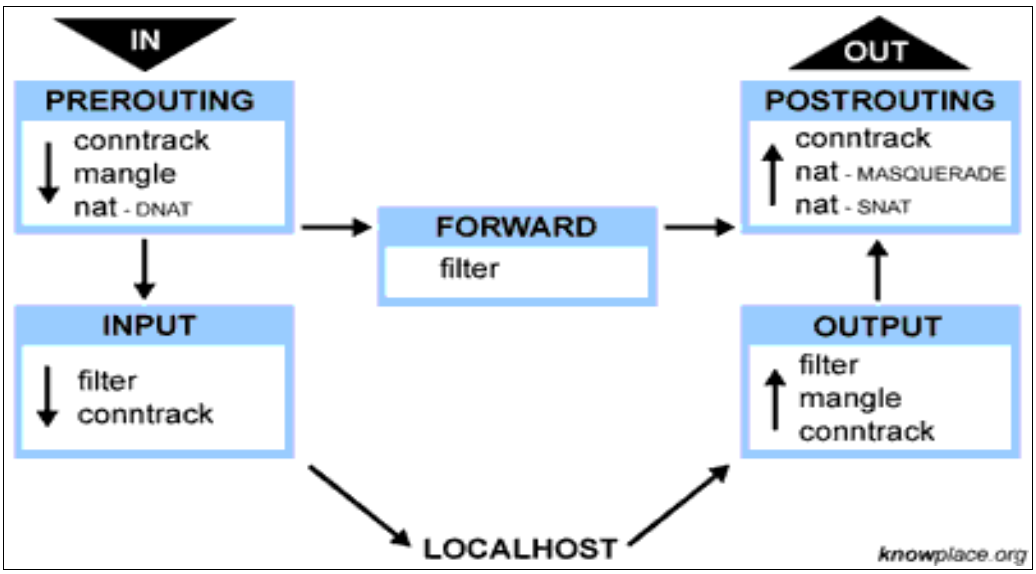
The aim here isn't to go too much into detail of how things are deployed in gnu-Linux environment or the design principles behind it. The basic points someone should know concerning this project about iptables and fire walling in Linux environment are the following:

- fire walling, NAT, and routing facilities are part of Linux's kernel, therefore actions are taken as fast as possible (concerning hardware, current processing load and other obvious restrictions).
- Interface is a program located under /sbin directory, is run in command line (or with a system command in a programming language) and accepts parameters following well known *nix semantics. Of course root privileges are needed and hence the need to have some processes running with such privileges.
- In modern kernels the philosophy of a stateful firewall is applied. This (concerning this project) means that rules can be applied easily having more in mind "what" should be done instead of "how" it should be done. So if allowance of connections to a web port are desired to be allowed, this can be done by issuing an one-line command such as: "/sbin/iptables -t tcp --dport 80 -j ACCEPT", and the internal mechanisms will handle all the information, while in a stateless fashion about 3 to 4 or more commands would be needed for the same results.
- As we will see later on behavior of iptables can be altered in order to fit user's needs for customization. As we've seen in previous section there are three categories of users. These facilities will be used in order to have a direct mapping of this project system's behavior.

A description of how connections are treated under iptables follows, more information can be found in various tutorials or articles in various web locations. One of the best available, written by authors of this software is the following one:

"http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO-6.html"

Abstracts of this document are used in the following pages. Having in mind this information someone can get into how things work and then this knowledge will be applied in this particular project.



Packet traversal in iptables environment (origin of image: http://www.knowplace.org/netfilter/syntax.html)

Every IP packet that comes through a network interface, or is about to leave one, has to pass a number of chains where the decision of what should be done with it will be taken. If the packet has to do with initializing a new connection, the PREROUTING chain will be the first one where decisions will be made. If the packet has localhost as it's destination address then it will follow through the INPUT chain and then it will be delivered to the local process. In case localhost is used as a router then the FORWARD chain will be consulted for routing related information. After that before leaving the host, POSTROUTING chain will be consulted. For packets generated by local programs,OUTPUT chain is consulted firstly, followed by POSTROUTING chain as above.

Also iptables has the concept of tables. Three tables exist: **filter**, **nat**, and **mangle** which process relative information. For example the filter table has to do with firewalling, and mangle with packer alteration/modification. The whole scheme can be demonstrated in the image at the top of this page.

At last as stated in iptables capabilities there is the capability to define new chains who have particular interest to different customized needs. An example of such behavior is a chain who will process authenticated user's traffic towards the local machine.

With this knowledge in mind it is easier to understand the scheme and the policies applied in the network layer. In the following section what is desired to be done concerning the user (section 4.1) are combined with the means available to achieve this goal (this section, 4.2.1). So the backbone of this scheme follows, while text starts to become more technical.

4.2.1 Constructing network layer behavior in Linux environment

As stated in previous section we have to cope with three categories of users (unknown-user, newcomer and authenticated) while the users interact with three main chains of iptables (INPUT, FORWARD and PREROUTING). In the following table the desired behavior for each user class – chain is demonstrated:

User class \ chain	**INPUT**	**FORWARD**	**PREROUTING**
Unknown (default)	Can issue a DHCP request. Can't do anything else.	Can't use the machine for forwarding	No rule applied
Newcomer	a. Can issue DHCP request (for more lease time). b. Can use local web server (so that user can authenticate). c. Can use local proxy (which redirects to web-server authentication page). d. Can perform a DNS single host query. (the reason why follows) e. Can't do anything else.	Can't use the machine for forwarding	All connections which are not targeted to localhost, are re-directed to local proxy process. This process redirects web-requests to login page, while shows a banner to all other services (such as ftp, ssh or irc)
Authenticated	No restrictions, can do anything	Connections are forwarded using local machine with NAT facilities	No rule applied (note: NAT has to do with postrouting)

Chains to handle policies specified in the previous page are demonstrated in section 6, more precisely in 6.1 in “prepare_iptables.sh” script. This script must be run before anything else in this project, so the best location for it is at system's initialization process. Also because Linux doesn't keep iptables related information for future reboots, rules need to defined at each startup. This means that there is no need to “stop” anything before shutdowns or reboots occur as well. Text in parentheses in this section shall be considered as reference to code in section 6.1, prepare_iptables.sh .

Two network interfaces were used. The following naming conversions that applied in many documents is to define **eth0** as name of the external interface (connected to Internet), while **eth1** is the name of the internal interface connected to the LAN via the wireless bridge (lines 11, 12). Line 11 follows:

```
11  client_interface=eth1
```

Instead of describing how every chain is defined for the previous nine combinations, one will be demonstrated, while the rest follow the same principles and syntax. The example will be the chain responsible for newcomer's prerouting behavior.

First of all the new chain (since it's not in the default ones) is defined in line 47:

```
47  $IPTABLES -t nat -N newcomer_prerouting
```

The rules are applied in lines 69 (a comment) and 71:

```
69  #newcomer_prerouting (nat table)
```

```
71  $IPTABLES -t nat -I newcomer_prerouting -i $client_interface -p tcp --
destination \! 192.168.0.1 -j DNAT --to-destination 192.168.0.1:$our_proxy_port
```

Rules are inserted in a stack (FILO) fashion, which happens by using the *-I* option. This doesn't affect behavior here, since we have only one rule but this is important in other chains. What we see here is that tcp connections *(-p tcp)* towards a machine other than local *(--destination \! 192.168.0.1)*, are redirected *(--to-destination)* to localhost, local proxy-port *(192.168.0.1:$our_proxy_port)* which will handle the request.

Some questions that rise here are the following: First of all: How can we direct a newcomer to this chain? Which brings us to how we can understand

that a newcomer has entered local-AD. Also after answering these questions, someone should ask how we remove the direction to this chain when user changes state. As we have seen before section 4.1) a newcomer may leave AD without authenticating (so there is the need to remove information related with him/her), or may become an authenticated user, where there is the need to alter information related with the newcomer state and direct to chains which have to do with authenticated users instead of newcomers.

Answers to these questions are given in following sections. Without altering the flow of this text, how everything is done has to do with iptables capabilities. By issuing the following command in dhcp_handler.rb, we give an extra-rule in the rule-stack for passing connection handling to an newcomer_prerouting chain, when a packet has a specific source address: *system("iptables -t nat -I PREROUTING -i #{Client_Interface} -p tcp -s #{@latest_IP_address} -j newcomer_prerouting")*

4.3 Database level Presentation

4.3.1 Overview

Up to now we have understood the desired behavior in the network layer. It is obvious that some data are generated through all the processes, which need to be re-used or altered. For example we want to record that a newcomer has entered local-AD, the IP address that was assigned by DHCP-server (daemon) and MAC address as well. Then after authentication information such as user name (which is now known) and needed for Accounting processes, must be stored somewhere. As stated in the first sections MySQL was used for these purposes.

Two databases are used: **dhcp_clients** and **local_AD**. The former is used for storing and retrieving information related to the current status of users (clients using local AD services) while the later stores information specific to local AD. For the moment these are configuration data such as the name of the wireless device, AD's name etc, while on the other hand there are user name-password pairs of local users. For the moment passwords are stored in plain text until a different method will be decided. This is also the place were communication with other Domains will be established, when they want to check user status.

Definition of dhcp_clients is at section 6.2. At 6.3 definition of local_AD database can be read, along with some sample data and default configuration. Grant tables are in section 6.4. Grant tables of database are designed with the least-privilege concept in mind, something that will be more obvious later on where we will see processes accessing them. For example all users must be located on local host, since there is no need for remote connections.

4.3.2 dhcp_clients database

As we can see (in section 6.3), this database consists of three tables: *Current_Clients* (line 26) and *Authenticated_IPs* (line 11) and *Original_Destination* (line 48). Below we can see the data elements of each table and the relative information that is stored.

4.3.2.1 DHCP clients table

IP address char(15)	MAC address char(17)
Username char(8)	Domain char(22)
User info char(30)	
Timestamp timestamp(14)	

Firstly beginning with Current Clients table. Each client is represented in a single row which consists of six columns:

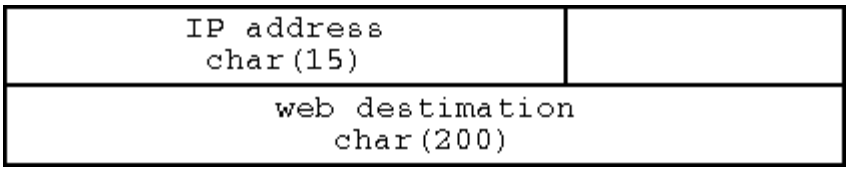
- IP address: which holds client's IP address.
- Mac Address: same as before but with MAC address.
- Username: As we've seen in section 4.1 user provides a similar with an e-mail address username. Here the part before the character “@” is included (username inside the domain).
- Domain: As before but now we have the domain part (example: aueb.domain.gr), which is the domain in which user belongs (part after the “@” character).
- User info: Can be described as the union of Username and Domain, or even better what user supplies when prompted so. Example: dimitris@aueb.domain.gr . We have more than needed data stored here and that's because we are still in a temporary situation where what will be used in the final scheme is not yet known. Therefore necessary redundancy is applied.
- Timestamp: When the user arrives a time stamp is applied. The time stamps here have the maximum length that MySQL can provide (from year to second), which explains their length (14 digits). These time stamps will be given to the user's browser (again as described in section 4.1) in order to be mixed with the plain text password before being hashed and sent back for processing. This process is necessary because it helps against the so called replay-attacks. If user replied through a browser with only an encryption of a password, then a third party watching the traffic could later be acquire user's identity with the following way: login with a “fake” browser, where instead of encrypting a plain text password, what was captured from watching a previous connection is being send. This attack has been used many times in the past in various cases. By demanding an answer which includes something unique (such as this time stamp), the whole scheme becomes safer against replay attacks. Since there is a different time stamp assigned with each user, relative data should be stored

here.

Because the previous table is more important for this project's scheme than the other's, an operational usage-demonstration follows:

1. When a user enters local-AD and gets an IP address, a query similar with the following is issued to MySQL:
   INSERT INTO Current_Clients **(**IP_address, MAC_address, Username, Domain, User_info, Timestamp**) VALUES (**'IP_address','MAC', NULL, NULL, NULL, **NOW());**
   Where NOW() represents the time stamp of query's issue.
2. After a successful authentication database must be altered in order to reflect recent changes, so a query similar to the following is going to be issued:
   UPDATE Current_Clients **SET** Username = 'login', Domain = 'domain', User_info = 'user' **WHERE** IP_address = 'IP' **LIMIT** 1 **;**
   "Limit 1" is placed having speed considerations in mind, since only one user has authenticated there is no need to traverse the table searching for a second one.
3. Finally when we are informed by the dhcp daemon that user has left local-AD we can remove relative data:
   DELETE FROM Current_Clients **WHERE** IP_address = 'IP_address' **LIMIT** 1 **;**
   Or something similar if we remove using user name (the WHERE part would be different)

4.3.2.2 Original Destination table



This table's structure is far more simpler, as well as the functionality that it serves. What we want here is to store which site user originally indented to visit when using the browser, before the authentication process. We have to fields:

- IP address: as user is "attached" with the IP provided, and
- web destination: where 200 characters (more than enough) are supplied.

4.3.2.3 Authenticated IP addresses table

IP address char(15)

Only one field:

- IP address: This table is used instead of using a formal Inter Process Communication mechanism. Reasons why have to do with easier and faster implementation, and uncertainty about which method fits best. When a user authenticates successfully the IP address is placed here. Process which has to do with applying different privileges in network level reads this table (by issuing a SELECT query) and then removes the data (a REMOVE query). So we have the mechanism of a pipe or of a FIFO queue in Linux environment.

4.3.3 local AD database

In this database, shown in section 6.3 two tables are stored: Configuration and local users.

4.3.3.1 Configuration table

Attribute char(30)
Value char(30)

This table holds configuration information about local AD as well as system's behavior. Each row represents an attribute – value pair:

- Attribute: the name of an attribute, such as wireless_device or AD name
- Value: for the value of a certain attribute, such as eth1 and “aueb.domain” for previously mentioned values.

4.3.3.2 Local Users table

Username char(30)	
Password char(15)	

Again we have a simple scheme here, as things are somehow premature for a final decision:

- Username: As stated in 4.3.2.1 (DHCP clients table), with an e-mail line fashion and
- Password: Stored in plain text format, until decided otherwise.

4.3.4 Grant tables

Grant tables, whose source code is on section 6.4, are designed having the least privilege concept in mind. Each process defined by a user name has rights to perform only the operations that is allowed to and nothing further. In order to see this in action, source code needs to be read, which lays in section 6.4.

4.4 Dhcp handler

4.4.1 Introduction

Up to now we have a concept of what should be done but we haven't interfere with how actions and decisions are taken. There are two points which will be mentioned. The former has to do with querying the dhcp server and reacting to specific events. The latter has to do with user authentication with the web interface. In this section reaction with dhcp server as well as other related issues will be addressed. One of the most important parts, which has to do with future P2P extensions of the project follows at last section.

4.4.2 How to handle the DHCP events

After some research several ways were found that could help into querying the dhcp server for specific information. Some seemed to fit more than others for various reasons. We will begin by describing the ones who were rejected, a path which will lead to the final choice made.

First of all dhcp server has the so called OMAPI support, where *“OMAPI is an programming layer designed for controlling remote applications, and for querying them for their state.”* as stated in it's man page. After studying the A.P.I. Structure the following problems arose: first of all it supplies only queries per single IP address: we can't ask for example what has changed since “last query”. The problem with this approach is that track should be kept with the previous state of every single IP address, issue one call for each one of them and then compare the results. The computational task is huge the resources needed in time and memory fairly large. Another element is that this API can be used only from C / C++ programs, limiting portability between languages. This can be solved in many ways, but adding to the previous reason makes this option to be rejected at once.

The second option was using ulog daemon. In IP tables a -log parameter can be used in order to log a certain packet. Ulog daemon can be configured of what should be done with that packet, actions such as informing a relational database such as MySQL. This scheme would be more than good for this project for various reasons: High level abstraction is the more obvious. The problem with ulog is that it sometimes “looses” - drops packets when operating

under heavy load. All packets are equal from ulog's “point of view”, but that doesn't happen with our project. Loosing an IP packet of an authenticated user may cause problems, such as annoyed clients. There wouldn't be a problem if heavy load (filling all buffers of ulog) was rare. But after discussions with other researchers of this project, a great number of packets originating or destined to clients, if not all, are going to be logged for accounting purposes. So ulog will always work under this circumstances. Using it would add obstacles to other researchers and cause problems in future development. As a result this option is also dropped.

A different approach followed, which was also the first attempt with working code. Dhcp server could provide debugging information stored into a file. Every single operation is recorded there if needed. After altering the initialization process of dhcp, all relative information was sent to a Unix pipe-file. A process was reading information from this file filtering changes interesting to this project, these information was passed (again with a pipe-file) for further processing. The advantages were that the relative information was supplied without loss of data or other problems. Disadvantages had to do with having to cope with a really messy approach: dhcp initialization script had to be altered, with no obvious reason why to someone studying it. This method is quite unorthodox: using data originally destined to debugging for other purposes, which means hard to explain to someone. Of course if using debugging data is really needed the problem of having dhcp handling (by this project) halted, adds complexity to the project. We are close to a solution, but not this one.

The final choice was to parse a status file generated by dhcp server, named “*leases.dhcp*”, which holds exact information such as IP addresses provided, lease times, MAC addresses etc. One big advantage is that this file is in pure text format containing only ASCII characters, this means that it's easy to be parsed and extract by familiar Unix/Linux utilities. After some research an extremely capable utility came into view, named dhcp status (“*dhcpstatus.pl*”). Dhcp status is part of most modern Linux distributions and it's functionality is to produce very well formated text based reports of Dhcp server, by parsing the “*leases.dhcp*” file. The different changes of Dhcp server are reported to our program by spotting differences between two sequential reports, using the generic Linux *diff* command, which produces differences between two text files, and parsing output in order to derive information.

Summarizing: We need to know when something changes about dhcp leases, because this information is important in handling clients states. The safest and most practical method is to parse text files generated and maintained by Dhcp server. What is needed is not file's contents but differences between

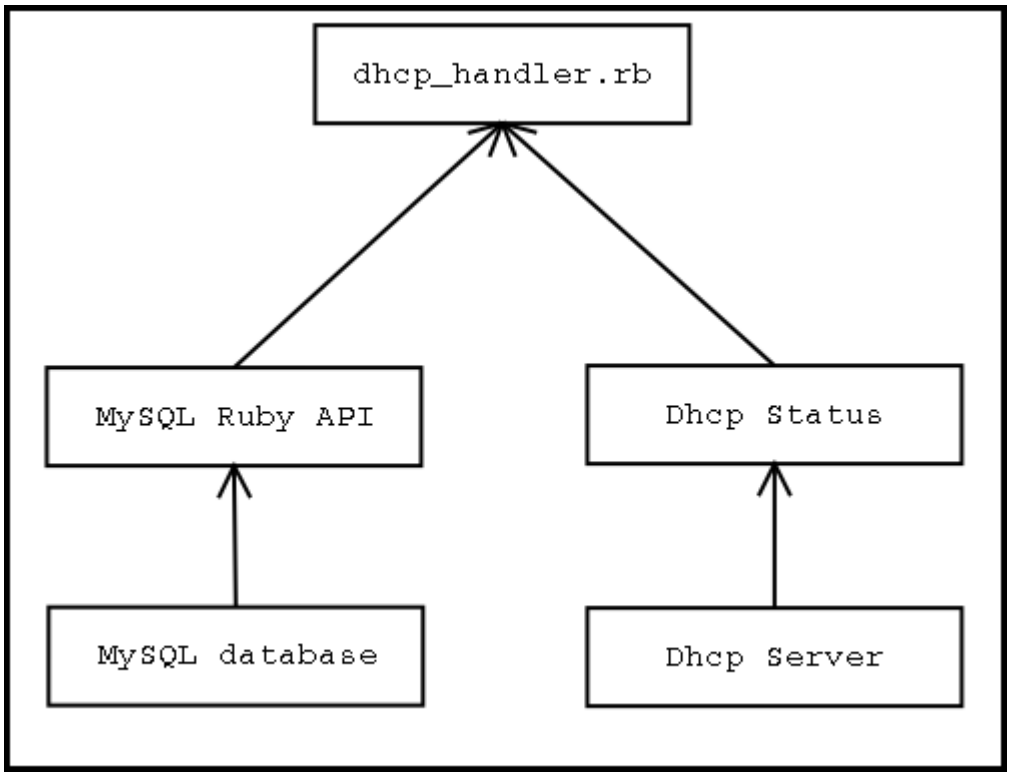
them. So we observe every two seconds by default what has changed, concerning the Dhcp server. Differences are handled accordingly.

4.4.3 Privilege Handling

One function that this program provides has to do along with coping with Dhcp events, is altering privileges of authenticated users. Decision has been made to place this responsibility here, because this is the only process of the whole project that needs to be run as “root”. By bringing this responsibility here, we are achieve to have a central point, from where things change, which makes administration easier.

4.4.4 Implementation

A source code analysis isn't provided here as with other parts of this project, since this is the place that most things tend to happen and code might change almost every time. Having in mind the organization of network and database layer and what is stated in previous paragraphs, one can understand source code with minimum effort. A diagrams showing various components needed by this program follows:



4.5 User Remover

4.5.1 Introduction

User remover, “user_remover.rb” with source code listed in section 6.8, is responsible for removing current users from system. This might occur for various different reasons such as: user decides to leave and informs for his/her intentions by using a log-off web page. Or maybe perhaps removal might occur by default after a ten minute inactivity. At this project it is only being used after an end of a Dhcp lease.

Source code of this program is deprived from dhcp_handler.rb and could have easily be a small function/class inside it. Reasons for this autonomy have to do with different conditions or triggers that might end to a user removal. Example processes responsible for accounting may remove a user after an inactivity period, as stated in previous paragraph. That's why this program is also tolerant in many conditions: A user removed by an accounting process will soon has his/her dhcp lease expired, so this program will be called twice for the same user. In that case only an error message will be shown.

4.5.2 Correct Syntax

Program may be called by one of the two following syntaxes:

user_remover **-i** ip_to_remove [reason]

or

user_remover **-u** username_to_remove [reason]

where the **’-i’** switch dictates that removal will be done based on user's IP address which must follow, while **’-u’** switch dictates that user removal will be done using user name, which must also follow. A **reason** might be supplied optionally, in case of tracking/logging facilities need it. Now no process is implemented to handle the 'reason' field, but is presented here in order to help future additions.

4.5.3 Implementation

Program begins executing at line 200 (beginning of main function), where first thing done is check of correct syntax. If there is a problem then help message similar with the previous paragraph is shown to user (line 45). In case of correct syntax constants are assigned (lines 20-37), from local database. Having everything in place an object to handle the removal is initialized (line 299), appropriate method called (line 230), and a destructor (line 231) before exit.

So everything is being done by an object of Client Remover class. In initialization (lines 65-76) a connection is made with local database so that sanity of input will be checked. What is desired is to extract an IP address – MAC address pair based either on an IP address or user name (provided when executing). Decision on which method will be followed is made in line 78. In case of IP address we want also user's class (authenticated or not) because rules will be removed from firewall depending on user-class. On the other hand if user name is supplied, it's sure we have to cope with an authenticated user hence class is considered known. Either case class is assigned in a constant value. These functions are performed in line 128.

After correct assignment of values, user is being removed where records are kept concerning him/her: local database (line 92) and iptables rule set (line 107).

4.6 CGI scripts

4.6.1 login page

4.6.1.1 Introduction

We have reached the point where a user is prompted with a login screen where the user name and password pair must be supplied so that local-ADs resources may be used. A screen-shot of this operation is illustrated in section 4.1, while source code, which is written in Ruby, is located at section 6.6. This point can be reached either by the proxy process described before, which will point a user here, or by a bookmark – manually, from the browser directly. This script generates html output, based on user's current status: a formal login page is displayed in case where everything seems to be normal, while an error page is shown otherwise. Something that must be mentioned is that the length of the source code is not related to it's complexity because of inclusion of a lengthy implementation of the MD5 algorithm, which is used as a “black box”, concerning our project.

4.6.1.2 Implementation

First of all we are in an area where many attacks are possible to occur during normal operation. According to secure programming guidelines, it is important not to supply any information to users concerning malfunctions as well as errors. So when a failure occurs user receives a “Server Error” web page. On the other hand valuable debugging information is stored in log files of http server (which is Apache at this implementation).

After loading the libraries needed (lines 16,17) and some constant values initialization (lines 19 to 28), an attempt to query the local database is made (line 30). What is requested is information about the specific IP address of cgi client. If it belongs to a local AD's user then a result is returned (else there is no result from the database server), also if this single IP address belongs to a non-authenticated user, user name field (line 42) should be null. This is the case where a proper login page should be displayed, so a “flag” constant is informed

(line 49, flag variable is set to “down” status). When at least one of the previous conditions is not matched, this error variable-flag is set to “up” value.

From this point through the end, things are pretty straightforward: if a problem has been spotted up to now (line 54) then (line 55) a web page is presented informing about it (html page, lines 64 to 93). Else (line 96) user is prompted with a log-in screen, requested to get on with the process.

As stated before, we want from users to supply an MD5 hash of the concatenation of their personal password with a fixed for each session time-stamp. Everything must be contained in the log-in page. So having retrieved the time-stamp at line 38, the following are supplied: a javascript based implementation of the MD5 algorithm (lines 107 to 367) and time-stamp (line 426). Also where the information will be sent to (login-cgi.rb) as we can see in the following part of this document, is supplied in line 401. Before data flow into target, password field is of course eliminated, while time-stamp is also re-transmitted as a token of “good will” from the user.

4.6.2 login cgi

4.6.2.1 Introduction

Along with “dhcp handler” this script is one of the most important parts of this project. The reason why has to do with it's functionality which will grow in importance as the project expands. From here connections will be initiated with other Administrative Domains in order to exchange information about each local AD and the guest users that want to use it.

Source code is in sector 6.7. When program starts some constant values are initialized. Then user must pass two “tests”. The former one is to supply the same timestamp that was assigned to the user. The idea behind this action is that the user must be originated from a page generated by login page script (previous sector). The latter “test” has of course to do with correct password. So the original password is loaded in plain text form from local database and then it is combined with the timestamp before hashed with the MD5 algorithm. Of course this hash must be the same with the hash supplied by user. If all these finish properly, local database will be informed for new user's status and privileges, while user will be informed for the result before being re-directed to his/her original destination.

Again here, as before users will not be informed in case of suspicious failure, but a “Server Error” page will be shown instead. In case of more innocent errors such as wrong password, an explanatory screen is shown instead.

4.6.2.2 Implementation

Program's main function begins at line 370 (actually with a comment that informs us about it), having constant definition as the first action (line 347), which calls a function defined between the 7th and 22th line of the program. What follows is some more object-variable initializations, such as and MD5 hash (line 379) which will re-calculate user's input in order to compare it against the one supplied, assignment of user's timestamp into STAMP value (line 381), and an input parser (line 386).

Input parser is declared and populated with code between lines 194 and 263. This class is responsible for checking input provided by users and assigns values to appropriate variables. This is the point where this CGI gets data “from the rest of the world” and is also the point from where most attacks against the whole system might occur. So most checks should be placed here. Now user supplied data are checked for “sanity”, for example correct size and hexadecimal characters for MD5 password. Results are returned in main, lines 388 to 391 (username, password, timestamp).

In line 397 is where a correct MD5 is constructed in order to be compared against the one supplied by user. Now local database is being queried in order to get a plain text password. But in future implementations this is the place where user's remote AD must be contacted and have a form of the password retrieved by it.

A correct log-in handler is initialized in case that it will be needed later on (line 401), but nothing is being done with it right now (only resource allocation). Objects of this class are responsible for: (1) Updating local database: Where user name is null (default value), authenticated user's name should be placed etc, and (2) Inform Dhcp handler, by inserting IP into Authenticated Table, so that this process will elevate user's privileges (local and Internet access).

At last two checks (one for timestamp and one for correct MD5 hash), are being made in line 403. In case of correctness, log-in handler object is being asked to perform actions described in previous paragraph, and an appropriate screen is being shown. If at least one check fails user an “error” web page is shown instead.

In case of correct log-in code in lines 29-101 is being executed. Some other functions, except plain html code, are the following: A different message is printed to user depending on his/her AD. We have two cases here, local and remote (lines 49-54). Also user's original destination (called “final” here), is being loaded from database (line 82), while user will be redirected there after a short time period which is needed in order to update user's privileges.

4.7 Initialization – Termination script

4.7.1 Introduction

After describing various components, the final one is the script that will start up Dhcp handler and setup firewall rules (default behavior, custom chain definition). One other process that must be also initiated is the Proxy one, but this is done by Dhcp handler, creating a sequence/hierarchy.

Since we are in Linux environment (Unix System V if being more specific), it's standards are to be followed. Code for this object is at section 6.10.

4.7.2 Overview

A script named “init_dawn.sh” is placed in **/etc/rc.d/init.d** directory, where scripts responsible for starting and stopping daemons are stored. Two links are created pointing to this file in the appropriate run level directory. In this case dawn should run on run level 3 (Multiuser with network support) so target directory is **/etc/rc.d/rc3.d** .

First symbolic link named “S99dawn” points to **/etc/rc.d/init.d/init_dawn.sh** and means Start (reason for 'S' letter), with priority 99 (lowest) where this link points. When daemons are called, target script is called with the word “start” as the first parameter so at boot time the command

/etc/rc.d/init.d/init_dawn.sh start

is issued.

Priority is the lowest one available, because we want other daemons to be already up and running, such as the MySQL database. If an attempt is made to start dawn before MySQL, for example, an error will occur.

The same fashion is being followed with the halting procedures, which occur before shutdown or reboot. A symbolic link named “K99dawn” is placed in the same directory for the same reasons. Semantics are the same, where 'K' means “kill”. Command issued is the same as before with “kill” instead of “start”, so it's the following one:

/etc/rc.d/init.d/init_dawn.sh kill

Script interprets the first argument given and acts accordingly as described in the following sector.

4.7.3 Implementation

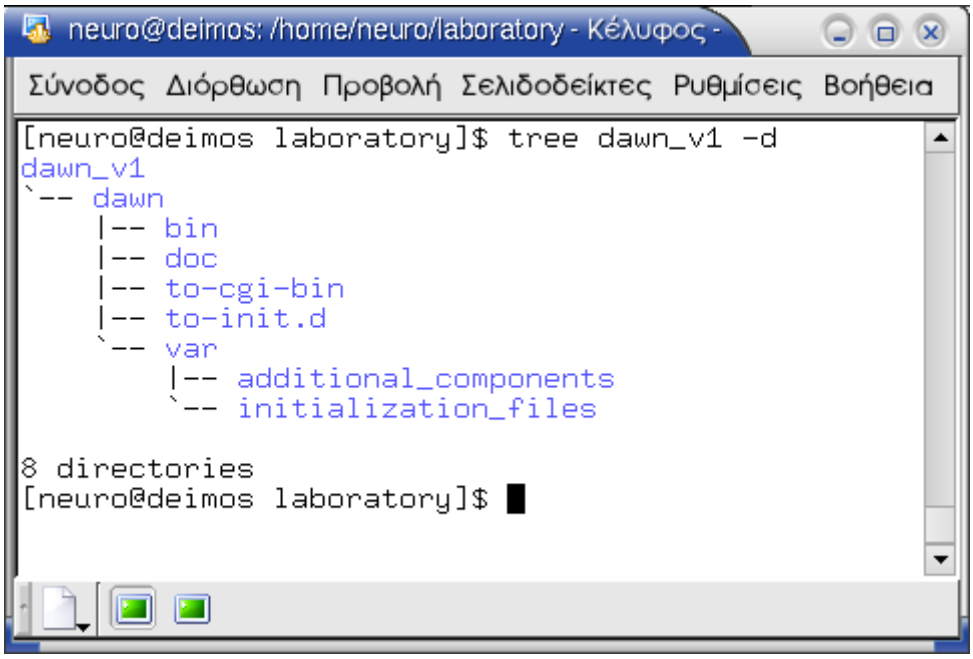
As we can see in line 42 command line arguments are interpreted with a case statement, according to the situation. If we need to “start” then function start_dawn (line 20) is called: Pid-file is deleted and a new empty one is created. Here process ID of dhcp_handler will be stored (lines 24-25). With this information the program will be terminated as we will see later on. Rules for iptables are loaded (line 27) and dhcp_handler is run by using nohup program (which traps all stop signals) and therefore makes it run as a daemon process (line 29).

If the desired action is to “stop” then stop_dawn function is called (line 33). Here process ID is read from pid-file and process is killed. What must be reminded is that dhcp handler which will be killed, is also responsible for stopping the connection Proxy, while first thing done by that program is to write it's process ID into the appropriate file.

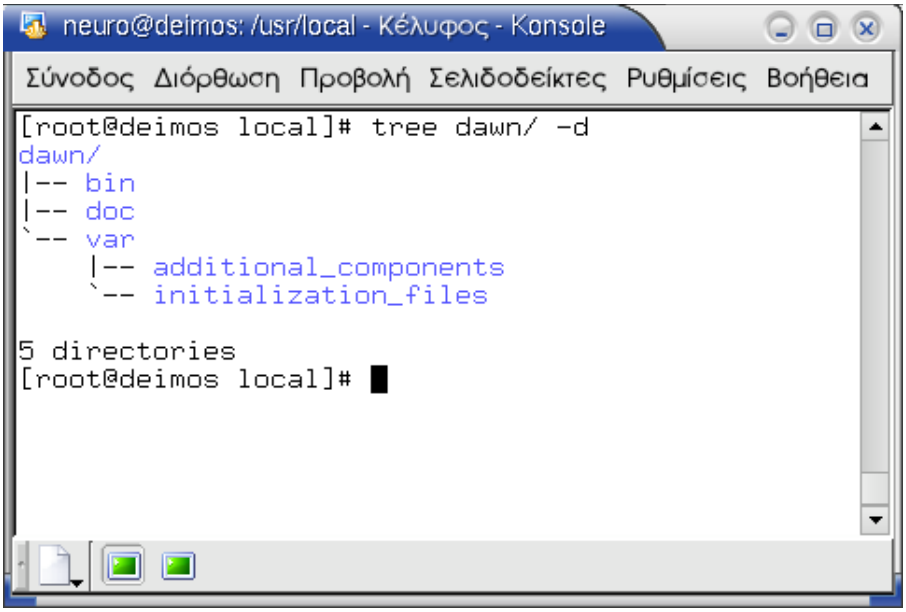
Locations of files used here follow well-known Linux semantics concerning filenames as well as locations or data stored in files. For example if we look at files under /var/log directory we will see only process Ids of relative with the filename daemons.

5.0 Installation procedure

Version 1.0 comes in .tgz format (created with tar and then gzipped). After extraction the following directory structure is created:



In dawn_v1 directory lies the installation script (Section 6.12) which will place files in appropriate directories. Most files are stored under **/usr/local** directory where additional programs are installed in Linux systems. Directory structure there is the following one:



Contents of each directory:

bin: All executables are placed here.
doc: Documentation, this document.
var: what can't be placed in other directories, for example source code for Java Proxy
additional components: Objects needed by dawn (Ruby language and Ruby-MySQL interface)
initialization files: placed here for future reference.

Two directories in original tar file aren't here:
to-cgi-bin: because cgi scripts of this project are placed in cgi scripts location
to-init.d: for the same reason as above but with initialization files.

6.0 Source Code

6.1 prepare_iptables.sh

```
#!/bin/bash

############################################################
# Script resonsible for setting up firewalling rules       #
# in order to understand the whole sceme see documentation #
#----------------------------------------------------------#
# Dimitris Mistriotis <besieger@yahoo.com>                 #
############################################################

#Constants definitions
client_interface=eth1
inet_interface=eth0
our_proxy_port=1003

#filename locations:
DEPMOD=/sbin/depmod
INSMOD=/sbin/insmod
IPTABLES=/sbin/iptables

echo " Inernet Interface: $inet_interface"
echo " Interface clients use: $client_interface"
echo " - Verifying that all kernel modules are ok"
/sbin/depmod Â-a
echo -en "ip_tables, "
$INSMOD ip_tables
echo -en "ip_conntrack, "
$INSMOD ip_conntrack
echo -en "ip_conntrack_ftp, "
$INSMOD ip_conntrack_ftp
echo -en "ip_conntrack_irc, "
$INSMOD ip_conntrack_irc
echo -en "iptable_nat, "
$INSMOD iptable_nat
echo -en "ip_nat_ftp, "
$INSMOD ip_nat_ftp
echo ". Done loading modules."
echo " enabling forwarding.."
echo "1" > /proc/sys/net/ipv4/ip_forward
echo " enabling DynamicAddr.."
echo "1" > /proc/sys/net/ipv4/ip_dynaddr

#make new chains
$IPTABLES -t filter -N default_input #make default-INPUT chain (for the
client_interface)
$IPTABLES -t filter -N default_forward
$IPTABLES -t filter -N newcomer_input
$IPTABLES -t filter -N newcomer_forward
$IPTABLES -t nat -N newcomer_prerouting
$IPTABLES -t filter -N authenticated_input
$IPTABLES -t filter -N authenticated_forward


#newcomer_input chain inserting rules in a stack-like fashion (LIFO)
$IPTABLES -t filter -I newcomer_input -j DROP
```

```
    54  $IPTABLES -t filter -I newcomer_input -i $client_interface -p icmp  -j
DROP
    55  $IPTABLES -t filter -I newcomer_input -i $client_interface -p udp --
dport \! 53 -j DROP
    56  $IPTABLES -t filter -I newcomer_input -i $client_interface -p udp --
dport 53 -j ACCEPT
    57  #53 udp, name server port
    58  $IPTABLES -t filter -I newcomer_input -i $client_interface -p tcp --
dport \! 80 -j DROP
    59  $IPTABLES -t filter -I newcomer_input -i $client_interface -p tcp --
dport 80 -j ACCEPT
    60  $IPTABLES -t filter -I newcomer_input -i $client_interface -p tcp --
dport 8008 -j ACCEPT
    61  $IPTABLES -t filter -I newcomer_input -i $client_interface -p tcp --
dport 1003 -j ACCEPT
    62  #change it to variable if this can be done :-)
    63
    64  #newcomer_forward chain which is far more simpler
    65  #(no forwarding allowed except web which will be altered at the nat
talbe)
    66  $IPTABLES -t filter -I newcomer_forward -i $client_interface -j DROP
    67  $IPTABLES -t filter -I newcomer_forward -i $client_interface  -p tcp --
dport 80 -j ACCEPT
    68
    69  #remove it after testing
    70  $IPTABLES -t filter -I newcomer_forward -i $client_interface  -p udp --
dport 53 -j ACCEPT
    71
    72  #newcomer_prerouting (nat table)
    73  #iptables -t nat -I newcomer_prerouting -i $client_interface -p tcp --
destination \! 192.168.0.1 --dport \! 80 -j DNAT --to-destination
192.168.0.1:$our_proxy_port
    74  $IPTABLES -t nat -I newcomer_prerouting -i $client_interface -p tcp --
destination \! 192.168.0.1 -j DNAT --to-destination 192.168.0.1:1003
    75  #iptables -t nat -I newcomer_prerouting -i $client_interface -p tcp --
dport \! 80 -j DNAT --to-destination 192.168.0.1:1003
    76
    77
    78
    79  #authenticated_input chain
    80  #$IPTABLES -t filter -I authenticated_input -j DROP
    81  $IPTABLES -t filter -I authenticated_input -i $client_interface -j
ACCEPT
    82  #allow everything
    83
    84  #default_input chain
    85  #allow _only_ dhcp requests
    86  $IPTABLES -t filter -I default_input -j DROP
    87  $IPTABLES -t filter -I default_input -i $client_interface -p udp --sport
68 --dport 67 -j ACCEPT
    88  #default_forward chain
    89  $IPTABLES -t filter -I default_forward -i $client_interface -j DROP
    90
    91
    92  #newcomer_forward chain
    93  $IPTABLES -t filter -I newcomer_forward -i $client_interface -j DROP
    94  #everything is dropped for everybody :-)
    95
    96
    97
    98  #make obligatory jump to this chain for users of $client_interface
    99  #in input and forward chains
   100  $IPTABLES -t filter -I INPUT -i $client_interface -j default_input
```

```
   101  $IPTABLES -t filter -I FORWARD -i $client_interface -j default_forward
   102  #Enabling SNAT (MASQUERADE) functionality on $inet_interface
   103  $IPTABLES -t nat -A POSTROUTING -o $inet_interface -j MASQUERADE
   104
   105
   106
   107  echo " FWD: Allow all connections OUT and only existing and related ones
IN"
   108  $IPTABLES -A FORWARD -i $inet_interface -o $client_interface -m state --
state ESTABLISHED,RELATED -j ACCEPT
   109
   110  $IPTABLES -A authenticated_forward -i $client_interface -o
$inet_interface -j ACCEPT
```

6.2 dhcp_clients.sql

This file was constructed by MySQL-dump utility, supplied with MySQL database package.

```
-- MySQL dump 8.22
--
-- Host: localhost    Database: dhcp_clients
---------------------------------------------------------
-- Server version       3.23.54

--
-- Table structure for table 'Authenticated_IPs'
--

CREATE TABLE Authenticated_IPs (
  ip_address char(15) NOT NULL default '',
  PRIMARY KEY  (ip_address)
) TYPE=MyISAM;

--
-- Dumping data for table 'Authenticated_IPs'
--



--
-- Table structure for table 'Current_Clients'
--

CREATE TABLE Current_Clients (
  IP_address char(15) NOT NULL default '',
  MAC_address char(17) NOT NULL default '',
  Username char(8) default NULL,
  Domain char(22) default NULL,
  User_info char(30) default NULL,
  Timestamp timestamp(14) NOT NULL,
  PRIMARY KEY  (IP_address)
) TYPE=MyISAM;

--
-- Dumping data for table 'Current_Clients'
--


INSERT INTO Current_Clients VALUES
('127.0.0.1','','none','none','none',20030328204346);
INSERT INTO Current_Clients VALUES
('192.168.0.1','','none','none','none',20030513141815);

--
-- Table structure for table 'Original_Destination'
--

CREATE TABLE Original_Destination (
  ip_address char(15) NOT NULL default '',
  web_destination char(200) default NULL,
  PRIMARY KEY  (ip_address)
) TYPE=MyISAM;
```

```
53
54  --
55  -- Dumping data for table 'Original_Destination'
56  --
```

6.3 local_AD.sql

Also created with MySQL dump utility.

```
-- MySQL dump 8.22

--
-- Host: localhost    Database: local_AD
---------------------------------------------------------
-- Server version       3.23.54

--
-- Table structure for table 'configuration'
--

CREATE TABLE configuration (
  Attribute char(30) NOT NULL default '',
  Value char(30) NOT NULL default '',
  PRIMARY KEY  (Attribute)
) TYPE=MyISAM;

--
-- Dumping data for table 'configuration'
--


INSERT INTO configuration VALUES ('wireless_device','eth1');
INSERT INTO configuration VALUES ('proxy_port','1003');
INSERT INTO configuration VALUES ('AD_name','aueb.domain.gr');
INSERT INTO configuration VALUES ('subnet','192.168.0.0');

--
-- Table structure for table 'local_users'
--

CREATE TABLE local_users (
  Username char(30) NOT NULL default '',
  Password char(15) NOT NULL default '',
  PRIMARY KEY  (Username)
) TYPE=MyISAM;

--
-- Dumping data for table 'local_users'
--


INSERT INTO local_users VALUES
('dimitris@aueb.domain.gr','dimitris1979');
INSERT INTO local_users VALUES ('Helias@aueb.domain.gr','wireless');
INSERT INTO local_users VALUES
('Visitor@another.domain.com','guesthere');

```

6.4 Grant Tables

```
     1  ###############################################################
     2  # Commands responsible for setting up MySQL GRANT tables      #
     3  # using GRANT statements instead of plain insert commands     #
     4  #-------------------------------------------------------------#
     5  # Dimitris Mistriotis 09 May 03 <besieger@yahoo.com>          #
     6  ###############################################################
     7
     8  #first one dhcp_handler:
     9  GRANT INSERT ON dhcp_clients.Current_Clients TO dhcp_handler@localhost
IDENTIFIED BY 'dhcp084';
    10  GRANT SELECT,DELETE ON dhcp_clients.Authenticated_IPs TO
dhcp_handler@localhost IDENTIFIED BY 'dhcp084';
    11
    12  #user remover:
    13  GRANT SELECT,DELETE ON dhcp_clients.Current_Clients TO
user_remover@localhost IDENTIFIED BY 'rmv23!';
    14
    15  #The following users connect from CGI-scripts (perfaps can somehow limit
only to that??)
    16
    17  #add authentication information
    18  GRANT INSERT ON dhcp_clients.Authenticated_IPs TO
add_auth_info@localhost IDENTIFIED BY 'adduser437';
    19
    20  #update dhcp information
    21  GRANT SELECT, UPDATE ON dhcp_clients.Current_Clients TO
dhcp_update@localhost IDENTIFIED BY 'update_now';
    22
    23  #read if a client has acquired IP address via dhcp in order to display
login page
    24  GRANT SELECT ON dhcp_clients.Current_Clients TO dhcp_read@localhost
IDENTIFIED BY 'read_now';
    25
    26  #only here privileges are performed outside the dhcp_clients database,
    27  #but on local_AD database where information (now only plain text
passwords) is stored about
    28  #each AD.
    29  GRANT SELECT ON local_AD.local_users TO local_password@localhost
IDENTIFIED BY 'pwd_manager';
    30
    31  #About Original_Destination table:
    32  #user who submits information
    33  GRANT INSERT ON dhcp_clients.Original_Destination TO
submit_web_page@localhost IDENTIFIED BY 'efstath';
    34  GRANT SELECT,DELETE ON dhcp_clients.Original_Destination TO
process_web_page@localhost IDENTIFIED BY 'process_now';
    35
    36  #About Attribure reading
    37  GRANT SELECT ON local_AD.configuration TO attribute_reader@localhost
IDENTIFIED BY 'attribute2003';
    38
    39
    40  #EOF
```

6.5 Proxy.java

This code has been produced by Helias Eustathiou (efstath@aueb.gr), on behalf of aiding this project. As we can see from the title, it's he proxy server response for newcomer user's TCP connections. If the connection has web attributes, HTML code redirecting to login page is supplied to the user, else a “banner”-message informing for login processes is supplied.

A web connection is identified by two basic characteristics: (a) browser sends some data, the desired web page, after TCP initialization, while other services such as ssh tend to wait for data (“banners”), and (b) these data include a “Host :” (line 29) string. Proxy tries (line 17) to read data (line 19) and if fails, we are in a non-web protocol, an error is caught (line 45).

```
 1  import java.io.*;
 2  import java.net.*;
 3  import java.util.*;
 4
 5  class ProxyThread extends Thread {
 6
 7          Socket s;
 8          String remoteIP;
 9          String getURL;
10          String host;
11
12          ProxyThread(Socket s) {
13                  this.s = s;
14          }
15
16          public void run() {
17                  try {
18                          s.setSoTimeout(500);
19                          BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
20                          DataOutputStream dos = new DataOutputStream(s.getOutputStream());
21
22                          try {
23                                  String line = br.readLine();
24                                  remoteIP = s.getInetAddress().getHostAddress();
25
26                                  StringTokenizer st = new StringTokenizer(line);
27                                  st.nextToken();
28                                  getURL = st.nextToken();
29                                  while (line.indexOf("Host: ") == -1) {
30                                          line = br.readLine();
31                                  }
32                                  st = new StringTokenizer(line);
33                                  st.nextToken();
34                                  host = st.nextToken();
35                                  saveToDB();
36                                  dos.writeBytes(response() + "\r\n");
37                                  s.close();
38
39                          } catch (InterruptedIOException x) {
40                                  dos.writeBytes("Please login to the local AD using your
browser.\r\n");
41                                  s.close();
42
43                          }
44
45                  } catch (Exception x) {
46                          x.printStackTrace();
47                  }
48          }
49
50          String response() {
```

```
    51
    52                  String page = "<HTML><HEAD><TITLE>Wait for login page to load</TITLE>" +
    53                          "<SCRIPT LANGUAGE=\"JavaScript\"><!--\r\nfunction redirect () { " +
    54                          "setTimeout(\"go_now()\",5000); }\r\nfunction go_now ()   {
window.location.href " +
    55                          " = \"http://192.168.0.1/cgi-bin/login-page.rb\"; }\r\n//--
></SCRIPT></HEAD>" +
    56                          "<BODY onLoad=\"redirect()\">\r\n<H1>Cannot redirect to login page.
</H1><BR><P>" +
    57                          "In order to log in your browser must have JavaScript enabled<P>" +
    58                          "By reading this page you use a browser without JavaScript
capabilities<P>" +
    59                          "(which are necessary to log-in) or you have JavaScript
disabled.<P>" +
    60                          "Correct this problem and try again.<P>or inform your local AD's
administrator for help</BODY></HTML>";
    61
    62                  return "HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n" + page;
    63
    64          }
    65
    66          void saveToDB() throws Exception {
    67
    68                  String command = "mysql -u submit_web_page --password=efstath dhcp_clients "
 +
    69                          "-e \"INSERT INTO Original_Destination (ip_address,
web_destination) VALUES " +
    70                          "('" + remoteIP + "', 'http://" + host + getURL + "');\"";
    71
    72                  long name = System.currentTimeMillis();
    73                  FileOutputStream fos = new FileOutputStream("" + name);
    74                  byte[] b = command.getBytes();
    75                  fos.write(b, 0, b.length);
    76                  fos.close();
    77                  File f = new File("" + name);
    78                  f.delete();
    79
    80                  String[] commands = new String[2];
    81                  commands[0] = "/bin/sh";
    82                  commands[1] = "" + name;
    83                  Process p = Runtime.getRuntime().exec(commands);
    84          }
    85
    86  }
    87
    88  class Proxy {
    89
    90          int port;
    91          String ip;
    92
    93          Proxy(int port, String ip) {
    94                  this.port = port;
    95                  this.ip = ip;
    96          }
    97
    98          void init() throws Exception {
    99
   100                  InetAddress ia = InetAddress.getByName(ip);
   101
   102                  ServerSocket ss = new ServerSocket(port, 100, ia);
   103
   104                  while (true) {
   105                          Socket s = ss.accept();
   106                          ProxyThread pt = new ProxyThread(s);
   107                          pt.start();
   108                  }
   109
   110          }
   111
   112          public static void main(String[] args) throws Exception {
   113                  int port = Integer.parseInt(args[0]);
   114                  String ip = args[1];
   115                  Proxy proxy = new Proxy(port, ip);
   116                  proxy.init();
   117
   118          }
   119  }
```

6.6 login-page.rb

```
     1  #!/usr/local/bin/ruby
     2  # in previous line location of ruby in current (and probably in most
linux) systems
     3  # may differ to yours, 100% sure in BSD installations
     4  =begin
     5  ************************************************
     6  * CGI script responsible for printing          *
     7  *   a login page to newcomers to local AD      *
     8  * output in html format, browser must support  *
     9  * javascript in order to interact              *
    10  *----------------------------------------------*
    11  * Dimitris Mistriotis 2003 (besieger@yahoo.com) *
    12  *                                              *
    13  ************************************************
    14  =end
    15
    16  require "mysql"
    17  require "cgi"
    18
    19  #define constants
    20  Host="localhost"
    21  Username="dhcp_read"
    22  Password="read_now"
    23  DB="dhcp_clients"
    24
    25
    26  $IP = ENV["REMOTE_ADDR"]
    27  $DB_reader = Mysql.new() #quering the database
    28  $DB_reader.connect(host=Host, user=Username, passwd=Password, db=DB)
    29
    30  _result=$DB_reader.query("SELECT Timestamp, Username FROM
Current_Clients WHERE IP_address= \'#{$IP}\'\ ")
    31
    32  #now first we will check the number of results it must be one
    33  #only one user per IP
    34  if (_result.num_rows ==1) then
    35  #check if user already connected
    36  #smash string
    37  _result.each do |row|
    38    $Stamp = row.to_s[/^[\d]{14,14}/]
    39    $Uname = row.to_s.delete($Stamp)
    40    end #each
    41
    42    if ($Uname != "") #if there is a username then an $Uname is an empty
string
    43    then
    44     ERROR_FLAG="UP" #constant assigned only once
    45    else
    46     ERROR_FLAG="DOWN"
    47     end
    48  else
    49    ERROR_FLAG="UP"
    50  end #if
    51
    52
    53  #so ready to show page to user
    54  if (ERROR_FLAG=="UP")
    55  then
    56
    57  # Code to handle what to do if a user has been here dy fault or
maliciously
```

```
  # should be placed here.

################################################################################
  # system (ALARM)
  #
  # ERROR PAGE FOLLOWS
  print "Content-type: text/html\r\n\r\n"
  print <<EOF
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
  <html>
  <head>
    <meta http-equiv="content-type"
   content="text/html; charset=ISO-8859-1">
    <title>Login Page -- error</title>
  </head>
  <body>
  <div style="text-align: center;"><big><big><span
   style="font-weight: bold;"><span style="text-decoration: underline;">Login
  Page</span></span></big></big><br>
  </div>
  <span style="font-weight: bold;"></span> <big
   style="color: rgb(255, 0, 0);"><span style="font-weight: bold;"><span
   style="font-weight: bold;"><br>
  </span></span>Error alrealy logged in or unknown IP</big><br>
  <br>
  According to your IP you have<br>
  1. Alreaddy logged in and you are here by accident-error<br>
  <span style="font-weight: bold;">or</span><br>
  2. You are probing- trying to access this system and you really should
  do<br>
  something else...<br>
  <br>
  <span style="font-weight: bold;"><span style="font-weight: bold;"><span
   style="font-weight: bold;"><br>
  </span></span></span>
  </body>
  </html>
  EOF

  else
  # LOGIN PAGE FOLLOWS part1
  print "Content-type: text/html\r\n\r\n"
  print <<EOF
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
  <html>
  <head>
    <title>Login Page - Welcome</title>

  </head>

  <script language="JavaScript" >

/*
 * A JavaScript implementation of the RSA Data Security, Inc. MD5 Message
 * Digest Algorithm, as defined in RFC 1321.
 * Version 2.1 Copyright (C) Paul Johnston 1999 - 2002.
 * Other contributors: Greg Holt, Andrew Kepert, Ydnar, Lostinet
 * Distributed under the BSD License
 * See http://pajhome.org.uk/crypt/md5 for more info.
 */

/*
 * Configurable variables. You may need to tweak these to be compatible with
 * the server-side, but the defaults work in most cases.
 */
var hexcase = 0;  /* hex output format. 0 - lowercase; 1 - uppercase        */
var b64pad  = ""; /* base-64 pad character. "=" for strict RFC compliance   */
var chrsz   = 8;  /* bits per input character. 8 - ASCII; 16 - Unicode      */

/*
 * These are the functions you'll usually want to call
 * They take string arguments and return either hex or base-64 encoded strings
 */
function hex_md5(s){ return binl2hex(core_md5(str2binl(s), s.length * chrsz));}
function b64_md5(s){ return binl2b64(core_md5(str2binl(s), s.length * chrsz));}
function str_md5(s){ return binl2str(core_md5(str2binl(s), s.length * chrsz));}
function hex_hmac_md5(key, data) { return binl2hex(core_hmac_md5(key, data)); }
```

```
function b64_hmac_md5(key, data) { return binl2b64(core_hmac_md5(key, data)); }
function str_hmac_md5(key, data) { return binl2str(core_hmac_md5(key, data)); }

/*
 * Perform a simple self-test to see if the VM is working
 */
function md5_vm_test()
{
  return hex_md5("abc") == "900150983cd24fb0d6963f7d28e17f72";
}

/*
 * Calculate the MD5 of an array of little-endian words, and a bit length
 */
function core_md5(x, len)
{
  /* append padding */
  x[len >> 5] |= 0x80 << ((len) % 32);
  x[(((len + 64) >>> 9) << 4) + 14] = len;

  var a =  1732584193;
  var b = -271733879;
  var c = -1732584194;
  var d =  271733878;

  for(var i = 0; i < x.length; i += 16)
  {
    var olda = a;
    var oldb = b;
    var oldc = c;
    var oldd = d;

    a = md5_ff(a, b, c, d, x[i+ 0], 7 , -680876936);
    d = md5_ff(d, a, b, c, x[i+ 1], 12, -389564586);
    c = md5_ff(c, d, a, b, x[i+ 2], 17,  606105819);
    b = md5_ff(b, c, d, a, x[i+ 3], 22, -1044525330);
    a = md5_ff(a, b, c, d, x[i+ 4], 7 , -176418897);
    d = md5_ff(d, a, b, c, x[i+ 5], 12,  1200080426);
    c = md5_ff(c, d, a, b, x[i+ 6], 17, -1473231341);
    b = md5_ff(b, c, d, a, x[i+ 7], 22, -45705983);
    a = md5_ff(a, b, c, d, x[i+ 8], 7 ,  1770035416);
    d = md5_ff(d, a, b, c, x[i+ 9], 12, -1958414417);
    c = md5_ff(c, d, a, b, x[i+10], 17, -42063);
    b = md5_ff(b, c, d, a, x[i+11], 22, -1990404162);
    a = md5_ff(a, b, c, d, x[i+12], 7 ,  1804603682);
    d = md5_ff(d, a, b, c, x[i+13], 12, -40341101);
    c = md5_ff(c, d, a, b, x[i+14], 17, -1502002290);
    b = md5_ff(b, c, d, a, x[i+15], 22,  1236535329);

    a = md5_gg(a, b, c, d, x[i+ 1], 5 , -165796510);
    d = md5_gg(d, a, b, c, x[i+ 6], 9 , -1069501632);
    c = md5_gg(c, d, a, b, x[i+11], 14,  643717713);
    b = md5_gg(b, c, d, a, x[i+ 0], 20, -373897302);
    a = md5_gg(a, b, c, d, x[i+ 5], 5 , -701558691);
    d = md5_gg(d, a, b, c, x[i+10], 9 ,  38016083);
    c = md5_gg(c, d, a, b, x[i+15], 14, -660478335);
    b = md5_gg(b, c, d, a, x[i+ 4], 20, -405537848);
    a = md5_gg(a, b, c, d, x[i+ 9], 5 ,  568446438);
    d = md5_gg(d, a, b, c, x[i+14], 9 , -1019803690);
    c = md5_gg(c, d, a, b, x[i+ 3], 14, -187363961);
    b = md5_gg(b, c, d, a, x[i+ 8], 20,  1163531501);
    a = md5_gg(a, b, c, d, x[i+13], 5 , -1444681467);
    d = md5_gg(d, a, b, c, x[i+ 2], 9 , -51403784);
    c = md5_gg(c, d, a, b, x[i+ 7], 14,  1735328473);
    b = md5_gg(b, c, d, a, x[i+12], 20, -1926607734);

    a = md5_hh(a, b, c, d, x[i+ 5], 4 , -378558);
    d = md5_hh(d, a, b, c, x[i+ 8], 11, -2022574463);
    c = md5_hh(c, d, a, b, x[i+11], 16,  1839030562);
    b = md5_hh(b, c, d, a, x[i+14], 23, -35309556);
    a = md5_hh(a, b, c, d, x[i+ 1], 4 , -1530992060);
    d = md5_hh(d, a, b, c, x[i+ 4], 11,  1272893353);
    c = md5_hh(c, d, a, b, x[i+ 7], 16, -155497632);
    b = md5_hh(b, c, d, a, x[i+10], 23, -1094730640);
    a = md5_hh(a, b, c, d, x[i+13], 4 ,  681279174);
    d = md5_hh(d, a, b, c, x[i+ 0], 11, -358537222);
    c = md5_hh(c, d, a, b, x[i+ 3], 16, -722521979);
    b = md5_hh(b, c, d, a, x[i+ 6], 23,  76029189);
    a = md5_hh(a, b, c, d, x[i+ 9], 4 , -640364487);
    d = md5_hh(d, a, b, c, x[i+12], 11, -421815835);
    c = md5_hh(c, d, a, b, x[i+15], 16,  530742520);
    b = md5_hh(b, c, d, a, x[i+ 2], 23, -995338651);

    a = md5_ii(a, b, c, d, x[i+ 0], 6 , -198630844);
    d = md5_ii(d, a, b, c, x[i+ 7], 10,  1126891415);
    c = md5_ii(c, d, a, b, x[i+14], 15, -1416354905);
    b = md5_ii(b, c, d, a, x[i+ 5], 21, -57434055);
    a = md5_ii(a, b, c, d, x[i+12], 6 ,  1700485571);
    d = md5_ii(d, a, b, c, x[i+ 3], 10, -1894986606);
    c = md5_ii(c, d, a, b, x[i+10], 15, -1051523);
```

```
    b = md5_ii(b, c, d, a, x[i+ 1], 21, -2054922799);
    a = md5_ii(a, b, c, d, x[i+ 8], 6 ,  1873313359);
    d = md5_ii(d, a, b, c, x[i+15], 10, -30611744);
    c = md5_ii(c, d, a, b, x[i+ 6], 15, -1560198380);
    b = md5_ii(b, c, d, a, x[i+13], 21,  1309151649);
    a = md5_ii(a, b, c, d, x[i+ 4], 6 , -145523070);
    d = md5_ii(d, a, b, c, x[i+11], 10, -1120210379);
    c = md5_ii(c, d, a, b, x[i+ 2], 15,  718787259);
    b = md5_ii(b, c, d, a, x[i+ 9], 21, -343485551);

    a = safe_add(a, olda);
    b = safe_add(b, oldb);
    c = safe_add(c, oldc);
    d = safe_add(d, oldd);
  }
  return Array(a, b, c, d);

}

/*
 * These functions implement the four basic operations the algorithm uses.
 */
function md5_cmn(q, a, b, x, s, t)
{
  return safe_add(bit_rol(safe_add(safe_add(a, q), safe_add(x, t)), s),b);
}
function md5_ff(a, b, c, d, x, s, t)
{
  return md5_cmn((b & c) | ((~b) & d), a, b, x, s, t);
}
function md5_gg(a, b, c, d, x, s, t)
{
  return md5_cmn((b & d) | (c & (~d)), a, b, x, s, t);
}
function md5_hh(a, b, c, d, x, s, t)
{
  return md5_cmn(b ^ c ^ d, a, b, x, s, t);
}
function md5_ii(a, b, c, d, x, s, t)
{
  return md5_cmn(c ^ (b | (~d)), a, b, x, s, t);
}

/*
 * Calculate the HMAC-MD5, of a key and some data
 */
function core_hmac_md5(key, data)
{
  var bkey = str2binl(key);
  if(bkey.length > 16) bkey = core_md5(bkey, key.length * chrsz);

  var ipad = Array(16), opad = Array(16);
  for(var i = 0; i < 16; i++)
  {
    ipad[i] = bkey[i] ^ 0x36363636;
    opad[i] = bkey[i] ^ 0x5C5C5C5C;
  }

  var hash = core_md5(ipad.concat(str2binl(data)), 512 + data.length * chrsz);
  return core_md5(opad.concat(hash), 512 + 128);
}

/*
 * Add integers, wrapping at 2^32. This uses 16-bit operations internally
 * to work around bugs in some JS interpreters.
 */
function safe_add(x, y)
{
  var lsw = (x & 0xFFFF) + (y & 0xFFFF);
  var msw = (x >> 16) + (y >> 16) + (lsw >> 16);
  return (msw << 16) | (lsw & 0xFFFF);
}

/*
 * Bitwise rotate a 32-bit number to the left.
 */
function bit_rol(num, cnt)
{
  return (num << cnt) | (num >>> (32 - cnt));
}

/*
 * Convert a string to an array of little-endian words
 * If chrsz is ASCII, characters >255 have their hi-byte silently ignored.
 */
function str2binl(str)
{
  var bin = Array();
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < str.length * chrsz; i += chrsz)
```

```
    bin[i>>5] |= (str.charCodeAt(i / chrsz) & mask) << (i%32);
  return bin;
}

/*
 * Convert an array of little-endian words to a string
 */
function binl2str(bin)
{
  var str = "";
  var mask = (1 << chrsz) - 1;
  for(var i = 0; i < bin.length * 32; i += chrsz)
    str += String.fromCharCode((bin[i>>5] >>> (i % 32)) & mask);
  return str;
}

/*
 * Convert an array of little-endian words to a hex string.
 */
function binl2hex(binarray)
{
  var hex_tab = hexcase ? "0123456789ABCDEF" : "0123456789abcdef";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i++)
  {
    str += hex_tab.charAt((binarray[i>>2] >> ((i%4)*8+4)) & 0xF) +
           hex_tab.charAt((binarray[i>>2] >> ((i%4)*8  )) & 0xF);
  }
  return str;
}

/*
 * Convert an array of little-endian words to a base-64 string
 */
function binl2b64(binarray)
{
  var tab = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
  var str = "";
  for(var i = 0; i < binarray.length * 4; i += 3)
  {
    var triplet = (((binarray[i   >> 2] >> 8 * ( i   %4)) & 0xFF) << 16)
                | (((binarray[i+1 >> 2] >> 8 * ((i+1)%4)) & 0xFF) << 8 )
                |  ((binarray[i+2 >> 2] >> 8 * ((i+2)%4)) & 0xFF);
    for(var j = 0; j < 4; j++)
    {
      if(i * 8 + j * 6 > binarray.length * 32) str += b64pad;
      else str += tab.charAt((triplet >> 6*(3-j)) & 0x3F);
    }
  }
  return str;
}


</script>


<body bgcolor="#cccccc" link="#000080" vlink="#6060c0" text="#000000"
 alink="#000099">

<div align="left">
<center>
<b>Welcome
 to local AD's login-page.</b><br>
  Please supply your <u>username</u> and <u>password</u> provided to you
by  your local AD administrator.<br>
  Username in user@remote.database.server format.<br>
  </center>
  <br>

<center></center>
  </div>

<center><br>

<table border="1" bgcolor="blue" cellspacing="0" cellpadding="0"
 align="center">
          <tbody>
            <tr>
              <td>

      <table border="0" width="100%" cellspacing="2" cellpadding="5">
```

```
                <tbody>
                  <tr>
                    <td bgcolor="white">

             <form name="auth_form"
  action="http://192.168.0.1/cgi-bin/login-cgi.rb" method="post">


                <table>
              <tbody>
                            <tr>
                              <td bgcolor="white"> Username </td>
                              <td bgcolor="white"> <input type="text"
  size="40" name="username" value=""> </td>
                            </tr>



                </tbody><tbody>
                                                    <tr>
                       <td bgcolor="white"> Password </td>
                             <td bgcolor="white"> <input type="password"
  size="40" name="msg" value=""> </td>
        <!-- msg=message to be encrypted-->                    </tr>

  EOF

  ############################################################
  # print stamp here and then the rest of the page follows #
  ############################################################
  print "<input type=\"hidden\" size=\"40\" name=\"timestamp\"
value=\"#{$Stamp}\">"

  print <<EOF
                <input name="password" size="40" type="hidden">

                </tbody>

                </table>


                <center>   <input type="submit" value="Submit"
  onclick="password.value = hex_md5( msg.value + timestamp.value ); msg.value =
'';"></center>
           </form>
                      </td>
                    </tr>


          </tbody>
        </table>
                </td>
              </tr>

    </tbody>
  </table>

  <p> <!--
  Source code of this page is a clone of the ideas/pages created by
  Paul Johnston (1998 - 2002), distributed under the BSD License
  I'd like to thank him for inspiration
  -->
      </p>
         </center>
          <br>

  </body>
  </html>

```

```
463  EOF
464
465  end #if
```

6.7 login-cgi.rb

```
     1  #!/usr/local/bin/ruby
     2  require 'cgi'
     3  require "mysql"
     4  require 'digest/md5'
     5
     6
     7  assign_constants = proc {
     8          begin
     9          _constants_db = Mysql.new()
    10          _constants_db.connect(host="localhost", user="attribute_reader",
password = "attribute2003", db="local_AD")
    11          _consants_results = _constants_db.query ("SELECT * FROM
configuration");
    12          _constants_db.close
    13          rescue MysqlError => connect_db_error
    14                  print "Problem during constant values assigment,
exiting\n"
    15                  print "Error number: #{connect_db_error.errno}. , Error
message: #{connect_db_error.error } \n"
    16                  exit(3)
    17          end
    18          _consants_results.each_hash do |row|
    19            case row["Attribute"]
    20            when "AD_name"
    21              AD_name = row["Value"]
    22            end #case
    23          end #do
    24          }
    25  #a Class which swows a correct-login page and redirects to webpage
    26  #user wanted originally to visit
    27  class Show_correct_log_in
    28
    29  #db constants:
    30  User_Page_Host = "localhost"
    31  User_Page_Username = "process_web_page"
    32  User_Page_Password = "process_now"
    33  User_Page_Database = "dhcp_clients"
    34  User_Page_Table = "Original_Destination"
    35
    36  def initialize(user_ip, username)
    37
    38  print "Content-type: text/html\r\n\r\n"
    39  print <<EOF
    40  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
    41  <html><head>
    42    <meta http-equiv="content-type" content="text/html; charset=ISO-8859-
1">
    43    <title>Correct Login</title></head>
    44  <body>
    45  <br>
    46  You've logged in <span style="text-decoration:
underline;">correctly</span> and you can now use #{AD_name}'s
services.<br><br><br>
    47  EOF
    48
    49  #if username ends in @AD_name then it's a local else it is a guest
    50  if (username[(username.index('@') +1)..username.size] == AD_name) then
    51    print "<br>You are a user originating from current AD<br>"
    52  else
    53    print "<br>You are a guest user<br>"
    54  end #if
```

```
    55
    56  final_destination = get_original_page(user_ip)
    57  if (final_destination == nil) then
    58  #redirect user to a predifined homepage
    59  print <<EOF
    60  redirecting you to our homepage <br>
    61  <SCRIPT LANGUAGE="JavaScript"><!--
    62  function redirect () { setTimeout("go_now()",5000); }
    63  function go_now ()   { window.location.href = "http://192.168.0.1"; }
    64  EOF
    65  else
    66  print "your ip address is: #{user_ip}<br>"
    67  print "you will be re-directed to: #{final_destination} in five
seconds<br>"
    68  print <<EOF
    69  <SCRIPT LANGUAGE="JavaScript"><!--
    70  function redirect () { setTimeout("go_now()",5000); }
    71  EOF
    72  print "function go_now ()   { window.location.href =
\"#{final_destination}\"; }\n"
    73  end #if
    74  print <<EOF
    75  //--></SCRIPT>
    76  <BODY onLoad="redirect()">
    77  </body></html>
    78  EOF
    79
    80  end #initialize
    81
    82  def get_original_page(ip_to_query)
    83    #connect to database and retrieve destination
    84    @web_db = Mysql.new()
    85    @web_db.connect(host=User_Page_Host, user=User_Page_Username,
passwd=User_Page_Password, db=User_Page_Database)
    86    @web_destination_query_result = @web_db.query("SELECT web_destination
FROM #{User_Page_Table} WHERE ip_address=\'#{ip_to_query}\'");
    87    #now the information can be deleted so that space will be left for
next one
    88    @web_db.query("DELETE FROM #{User_Page_Table} WHERE
ip_address=\'#{ip_to_query}\'");
    89    #and memory can be fred
    90    @web_db.close
    91    #now the single header can be un-wrapped from result object and passed
back to calling method
    92    if (@web_destination_query_result.num_rows == 0 ) then
    93      @result_page = nil
    94    else
    95      @result_page = @web_destination_query_result.fetch_row[0].to_s
#there can be only one row
    96    end #if
    97    @web_destination_query_result.free #finished also with result
    98    _return = @result_page
    99  end #get_original_page
   100
   101  end #Show_correct_log_in
   102
   103  class Error_exit
   104  #get an error code, present an error-page and do actions
   105  #appropriate i.e. log or just exit or alarm
   106
   107  def initialize (error_code)
   108  print "Content-type: text/html\r\n\r\n"
   109  print <<EOF
```

```
  110  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
  111  <html><head>  <meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1">
  112     <title>Login Error</title></head>
  113  <body>
  114  <div style="text-align: center; font-weight: bold;"><big
  115   style="font-weight: normal;"><span style="color: rgb(255, 0,
0);">ERROR</span></big><br>
  116  </div>
  117  <br>You might see this page for the following reasons:<br><ul>
  118    <li>You've entered <span style="text-decoration: underline;">incorrect
  119  password</span>, or<br>  </li>
  120    <li>You've used a <span style="text-decoration: underline;">cached
  121  version of the login page</span>, while you should download a new one,
  122  or <br>  </li>
  123    <li>You are feeding the login sceme with random or malicious data
(<span
  124   style="text-decoration: underline;">cross side scripting</span>)<br>
</li>
  125  </ul><br>
  126  You should try to fix these problems, contact support for more
  127  information.<br>
  128  <br><br>
  129  </body> </html>
  130  EOF
  131
  132  #actions according to error code
  133    case error_code
  134    when 1,2,3
  135      exit(error_code)
  136    end #case
  137  end #initialize
  138  end #Error_exit
  139
  140  class Correct_Loggon_handler
  141  #an object of this class is responsible for the following
  142  #a. update user log-on information
  143  # (update local database, where user==NULL now user = user supplied
name)
  144  #b. change access rights to the system
  145  #c. execute a "trigger" call if needed
  146  #because of security reasons this class may be initialized at the very
beginning
  147  #but the initializer must do nothing, the work must be done by functions
later on
  148  Dhcp_Host = "localhost" #some constants
  149  Dhcp_DB = "dhcp_clients"
  150  Dhcp_Username = "dhcp_update"
  151  Dhcp_Password = "update_now"
  152  Auth_Username = "add_auth_info"
  153  Auth_Password = "adduser437"
  154  Auth_Table = "Authenticated_IPs"
  155
  156  def initialize(ip_address, username)
  157    @canditate_IP = ip_address #caditate because we don't know if all are
Ok
  158    @candiatate_user = username #during init process
  159    @db_handler = Mysql.new() #but again don't connect yet
  160  end #initialize
  161
  162  def unlock_user()
  163  #does three tasks: updates user database, "unlocks IP" and
  164  # pulls a trigger _if_ needed
```

```
   165    update_database()
   166    unlock_IP()
   167  end #unlock_user
   168
   169  def update_database()
   170    #according to what is implemented in mySQL tables
   171    _at_position = @candiatate_user.index('@')
   172    login = @candiatate_user[0..(_at_position-1)]
   173    domain = @candiatate_user[(_at_position+1)..@candiatate_user.size]
#the rest of it
   174    @db_handler.connect(host=Dhcp_Host, user=Dhcp_Username,
passwd=Dhcp_Password, db=Dhcp_DB)
   175    @db_handler.query ("UPDATE Current_Clients SET Username =
\'#{login}\', Domain = \'#{domain}\', User_info = \'#{@candiatate_user}\' WHERE
IP_address = \'#{@canditate_IP}\' LIMIT 1")
   176    #only one IP address will be updated and therefore LIMIT's role
   177    @db_handler.close #no more needed
   178  end #update_database()
   179
   180  def unlock_IP()
   181    #insert this ip into Authenticated_IP table in local database
   182    #so that it will be processed by the appropriate program
   183    _auth_handler = Mysql.new()
   184    _auth_handler.connect(host=Dhcp_Host, user=Auth_Username,
passwd=Auth_Password, db=Dhcp_DB)
   185    _auth_handler.query ("INSERT INTO #{Auth_Table} (ip_address) VALUES
(\'#{@canditate_IP}\')")
   186    #if this fails the program might fail also
   187    _auth_handler.close
   188  end #unlock_IP()
   189
   190  private :update_database
   191  private :unlock_IP
   192  end #Correct_Loggon_handler
   193
   194  class Input_parser
   195  def initialize()
   196    @form_parser=CGI::new
   197    #a parser should check character-by character the input as a string
   198    #before proceeding in order to see if "illegal" characters are present
   199    #illegal = not legal (not in the set of those needed)
   200    #then a good r.e. check would help
   201    #this stuff is here ommited in order to do more work in the script
   202
   203    #set username:
   204    data = @form_parser['username'].to_s #.dump
   205    if (ok_username(data) == true)  then
   206      @username = data
   207    else     Error_exit.new(2)
   208    end #if (username check)
   209    #set hashed_password:
   210    data = @form_parser['password'].to_s #.dump
   211    if (ok_hashed_password(data) == true)  then
   212      @hashed_password = data
   213    else     Error_exit.new(2)
   214    end #if (password check)
   215    #set timestamp:
   216    data = @form_parser['timestamp'].to_s #.dump
   217    if (ok_timestamp(data) == true)  then
   218      @timestamp = data
   219    else     Error_exit.new(2)
   220    end #if (timestamp check)
   221
```

```
  end #initialize

  def ok_username(string_to_check)
    if ( (string_to_check =~ /[\w]+[@]([\w]+.)+[\w]+/) and
(string_to_check.size <=30) )
    then
      result = true
    else  result = false   end
  end #ok_username

  def ok_hashed_password(string_to_check)
    #password is an md5 encrypted hash so it consists of 32 hex characters
    #a-f letters in hex are in lowcase
    if (string_to_check =~ /[\da-f]{32,32}/)
    then
      result = true
    else  result = false   end
  end #ok_hashed_password

  def ok_timestamp(string_to_check)
    #timestamp consists of 14 digits in this sceme
    if (string_to_check =~ /[\d]{14,14}/)
    then
      result = true
    else  result = false   end
  end #ok_timestamp

  def get_username()
    result = @username
  end #get_username

  def get_hashed_password()
    result = @hashed_password
  end #get_password

  def get_timestamp()
    result = @timestamp
  end

  private :ok_username
  private :ok_hashed_password
  private :ok_timestamp
  end #Input_parser


  #two links will be established here: one for acquiring the timestamp
  #in order to check it with the password as well as to evaluate
  #it with the one sent by the login-page

  #Creating a wrapper for the dhcp clients database
  class Timestamp_dbwrapper
  #define constants
  Timestamp_Host="localhost"
  Timestamp_Username="dhcp_read"
  Timestamp_Password="read_now"
  Timestamp_DB="dhcp_clients"

  def initialize(ip_tosearch)
    @Timestamp_reader = Mysql.new()
    @Timestamp_reader.connect(host=Timestamp_Host,
user=Timestamp_Username, passwd=Timestamp_Password, db=Timestamp_DB)
    @result_Timestamp=@Timestamp_reader.query("SELECT Timestamp FROM
Current_Clients WHERE IP_address= \'#{ip_tosearch}\'")
```

```
  282  end #initialize
  283
  284  def return_timestamp()
  285    if (check_result() == "ok")
  286    then
  287      stamp = @result_Timestamp.fetch_row[0]
  288    else
  289      stamp = nil
  290    end #if
  291  end #return_timestamp
  292
  293  def shutdown
  294    @result_Timestamp.free
  295    @Timestamp_reader.close
  296  end #shutdown
  297
  298  def check_result()
  299    if (@result_Timestamp.num_rows ==1)
  300    then
  301      @result = "ok"
  302    else
  303      @result = "error"
  304    end #if
  305  end #check_result
  306
  307  private   :check_result
  308  end #class
  309
  310  class Password_dbwrapper
  311  # Final code must sure be altered here in order to
  312  # connect with (remote) users AD
  313  ######################################################
  314  #define constants
  315  Password_Host="localhost"
  316  Password_Username="local_password"
  317  Password_Password="pwd_manager"
  318  Password_DB="local_AD"
  319
  320    def initialize(username_to_search)
  321      @Password_reader = Mysql.new()
  322      @Password_reader.connect(host=Password_Host, user=Password_Username,
passwd=Password_Password, db=Password_DB)
  323      @result_Password=@Password_reader.query("SELECT Password FROM
local_users WHERE Username=\'#{username_to_search}\'")
  324      @parsed_once="no"
  325    end #initialize
  326
  327  def check_result_rows()
  328    if (@parsed_once == "no")
  329    then
  330      if (@result_Password.num_rows ==1)
  331        then
  332          @parsed_once="yes"
  333          @result = "ok"
  334      else
  335      #now we have a password violation or error user must be informed and
actions
  336      #must be logged as well as exit with an error
  337
##########################################################################
  338        @parsed_once="yes"
  339        @result = "not-ok"
  340        #Error_exit.new(2)
```

```
341       end #if
342     else
343     # programming error, just exit
344       exit(1)
345     end #if
346
347   end #check_result
348
349   def return_password
350     if (@result_Password.num_rows ==1)
351     then
352         password = @result_Password.fetch_row[0].to_s
353         if (password == nil) then #user unknown to database
354         Error_exit.new(2)
355         else
356         result = password
357         end #if
358     else
359         Error_exit.new(2)
360     end #if
361   end #return_password
362
363   def shutdown
364     @result_Password.free
365     @Password_reader.close
366   end #shutdown
367   private   :check_result_rows
368   end #class
369
370   #/----------------\
371   #| -->  main  <-- |
372   #\----------------/
373   #define constants
374   assign_constants.call
375   $IP = ENV["REMOTE_ADDR"] #global scope
376
377   #object initialization
378
379   md5_hash =Digest::MD5.new()
380
381   db1= Timestamp_dbwrapper.new($IP)
382   STAMP= db1.return_timestamp() #constant so that it can't be altered
383   db1.shutdown()
384
385   #user data as provided by login form:
386   user_data = Input_parser.new()
387
388   #check for correct format on the query
389   uname=user_data.get_username()
390   passwd=user_data.get_hashed_password()
391   timestamp=user_data.get_timestamp()
392
393   #connect to the local user database
394   db2= Password_dbwrapper.new(uname)
395   #because we want to free the plain-text password As Soon As Possible
396   #the hash will be calculated at once and then the wrapper will be
deleted
397   md5_hash.update(db2.return_password().to_s + STAMP)
398   hash = md5_hash.hexdigest
399   db2.shutdown()
400   #a call to garbage collector can be placed here
401   logg_in = Correct_Loggon_handler.new($IP,uname)
402
```

```
403  if ( (STAMP == timestamp) and (hash == passwd) )then
404    #the essential security check
405    logg_in.unlock_user()
406    Show_correct_log_in.new($IP, uname)
407  else
408    #timestamps do not match == attempt to violate for sure
409    Error_exit.new(1)
410  end #if
```

6.8 user_remover.rb

```
#!/usr/local/bin/ruby
=begin
#####################################################################
# Script responsible for removing user from database as well as     #
# from firewall rules                                               #
#-------------------------------------------------------------------#
# Dimitris Mistriotis <besieger@yahoo.com>                          #
#####################################################################
=end

begin
require "mysql"
  rescue LoadError => load_err
  print "Abscence of needed module in order to operate normally.\n Error
message follows\n"
  print "#{load_err} \n"
  exit(1)
end


assign_constants = proc {
        begin
        _constants_db = Mysql.new()
        _constants_db.connect(host="localhost", user="attribute_reader",
password = "attribute2003", db="local_AD")
        _consants_results = _constants_db.query ("SELECT * FROM
configuration");
        _constants_db.close
        rescue MysqlError => connect_db_error
                print "Problem during constant values assigment,
exiting\n"
                print "Error number: #{connect_db_error.errno}. , Error
message: #{connect_db_error.error } \n"
                exit(3)
        end
        _consants_results.each_hash do |row|
          case row["Attribute"]
          when "wireless_device"
            Client_Interface = row["Value"]
          end #case
        end #do
        }

#constant definitions, edit them in order to suit your system
IPtables = "/sbin/iptables"
#end of constant definitions



show_help = proc {
        print "Correct syntax:\n"
        print "user_remover -i ip_to_remove [reason]\n or \n"
        print "user_remover -u username_to_remove [reason]\n"
        print "in the optional reason field why the user is removed can
be specified, \n"
        print "(example end_of_lease) it doesn't alter program behavior,
but this may change in future releases.\n"
        exit(2)
        }

check_iptables = proc { print "Checking if iptables is present and this
```

```
program is able to use it\n" }
    55
    56  class Client_Remover
    57  #the whole code is organized in class although all will be used only
once for various reasons
    58  #for example someone may want to remove all users before shutdown,with
this class-organization it's easy
    59  #  constants for this class:
    60    Remover_Username = "user_remover"
    61    Remover_Password = "rmv23!"
    62    Remover_Host = "localhost"
    63    Remover_Database = "dhcp_clients"
    64    Remover_Table = "Current_Clients"
    65  def initialize
    66    print "Initializing Client Remover.\n"
    67    #open an always on connection to local database
    68    begin
    69    @db_handler = Mysql.new()
    70    @db_handler.connect(host=Remover_Host, user=Remover_Username,
passwd=Remover_Password, db=Remover_Database)
    71    rescue MysqlError => connect_error
    72         print "Problem while connecting to #{Remover_Database}
database.\n Error message: ", connect_error.error, "\n"
    73         exit(2)
    74    end #begin
    75
    76  end #initialize
    77
    78  def remove (method, data)
    79    case method
    80     when "ip"
    81       print "if it's to remove via IP then check IP and then go on\n"
    82       db_query_ip(data)
    83       #user_class = newcomer or authenticated
    84       #if we have reached this point verything is ok so:
    85       @final_IP = data
    86     when "username"
    87       print "if it's to remove via username get IP from user and then go
on\n"
    88       #if user is in database then final_IP = his IP
    89       user_class = authenticated
    90       #else exit error
    91    end #case
    92    database_remove(@final_IP)
    93    firewall_remove(@final_IP,@user_class)
    94  end #remove
    95
    96  def database_remove(ip_address)
    97    #issue a delete query
    98    begin
    99    _delete_result = @db_handler.query("DELETE FROM #{Remover_Table} WHERE
IP_address='#{ip_address}' LIMIT 1")
   100    rescue MysqlError => delete_error
   101    print "Could not delete from database, perhaps permissions problem\n"
   102    print "Error code: ", insert_error.errno, "\n"
   103    exit(2)
   104    end
   105  end #database_remove
   106
   107  def firewall_remove(ip_address,user_class)
   108    _user_MAC = get_mac()
   109    print "User's MAC #{_user_MAC}\n"
   110    print "Remove_Script mac to remove: #{_user_MAC}\n"
```

```
111     case user_class
112      when "newcomer"
113        print "removing newcomer, a user who hasn't authenticated\n"
114        #the opposite of the insertion rules, only -I becomes -D
115        system("iptables -t filter -D INPUT -i #{Client_Interface} -s
#{ip_address} -m mac --mac-source #{_user_MAC} -j newcomer_input")
116        system("iptables -t nat -D PREROUTING -i #{Client_Interface} -p tcp
-s #{ip_address}  -j newcomer_prerouting")
117        system("iptables -t filter -D FORWARD -i #{Client_Interface} -s
#{ip_address} -m mac --mac-source #{_user_MAC} -j newcomer_forward")
118
119      when "authenticated"
120        print "removing a user who has authenticated himshelf\n"
121          system("iptables -t filter -D INPUT -i #{Client_Interface} -s
#{ip_address} -m mac --mac-source #{_user_MAC} -j authenticated_input")
122          #more (08 June):
123          system("iptables -t filter -D FORWARD -i #{Client_Interface} -s
#{ip_address} -m mac --mac-source #{_user_MAC} -j authenticated_forward")
124
125     end #case
126   end #firewall_remove
127
128   def db_query_ip(ip_to_check)
129     #if ip is in database then a. set user class, b. return true
130     #select statement here queries for username
131     #if it's null but number of results is one (1) then it's a newcomer
132     #else if numner of results is zero then we have error
133     #or if number of results is one but != null then it's an authenticated
one
134
135     #if IP is in database then final_IP=data
136     _select_result = @db_handler.query("SELECT User_info, MAC_address from
#{Remover_Table} WHERE IP_address='#{ip_to_check}' LIMIT 1") #limited to one it
can't be more
137     if (_select_result.num_rows !=0) then
138       _select_result.each_hash do |row|
139         print "info: #{row["User_info"]}, mac: #{row["MAC_address"]} \n"
140         set_mac(row["MAC_address"])
141         if ( row["User_info"] != nil ) then
142           #the user is an authenticated one
143           print "authenticated user...\n"
144           @user_class = "authenticated"
145         else
146           print "Newcomer\n"
147           @user_class = "newcomer"
148         end #if
149       end #do |row|
150       _select_result.free #memory used
151     else
152       _select_result.free
153       #print error message and exit normally
154       print "No user with IP address #{ip_to_check} exists now on
system.\n"
155       exit(0)
156       #tolerance is shown here because a user might be requsted to be
removed from two
157       #reasons at once (e.g. end of lease and user request)
158     end #if
159
160   end #db_query_ip
161
162   def db_query_username(username_to_check)
163     #if user is in database then a. set user class b. set ip address
```

```
c.return true
   164    _select_result = @db_handler.query("SELECT IP_address, MAC_address
from #{Remover_Table} WHERE IP_address='#{username_to_check}' LIMIT 1") #limited
to one it can't be more
   165    if (_select_result.num_rows !=0) then
   166      @user_class = "authenticated" #no other possible choice!
   167      _select_result.each_hash do |row|
   168        print "ip: #{row["IP_address"]}, mac: #{row["MAC_address"]} \n"
   169        set_mac(row["MAC_address"])
   170        @final_IP = row["IP_address"]
   171      end #do |row|
   172    else  #no authentication has been made no username is present
   173      _select_result.free
   174      print "No user with Username #{username_to_check} exists now on
system.\n"
   175      exit(0)
   176    end #if
   177  end #db_query_username
   178
   179  def set_mac(current_mac_address)
   180    @user_mac = current_mac_address.upcase #iptables uses uppercase
letters
   181  end #set_mac
   182
   183  def get_mac()
   184    _temp = @user_mac
   185  end #get_mac
   186
   187  def shutdown()
   188    #close connection with local database
   189    @db_handler.close()
   190  end #shutdown()
   191
   192  private :firewall_remove
   193  private :database_remove
   194  private :db_query_ip
   195  private :db_query_username
   196  private :set_mac
   197
   198  end #Client_Remover
   199
   200  #--> Main <--#
   201  #check of command line data
   202  #number of arguments
   203  if ( (ARGV.size !=2) and (ARGV.size !=3) ) then
   204    print "Incorrect number of arguments\n\n"
   205    show_help.call
   206  end #if
   207
   208  case ARGV[0]
   209    when "-i"
   210      print "removing using ip\n"
   211      if ( ARGV[1] !~ /^\d\d\d.\d\d\d.\d*.\d*$/ ) then
   212        print "Incorrect IP format: #{ARGV[1]}\n" #showing the error to
the user
   213        show_help.call
   214      end #if
   215      $method = "ip"
   216    when "-u"
   217      print "removing using username\n"
   218      if ( ARGV[1] !~ /^[\w]+[@]([\w]+.)+[\w]+$/ ) then
   219        print "Incorrect username format: #{ARGV[1]}\n" #showing the error
to the user
```

```
220         show_help.call
221       end #if
222       $method = "username"
223   when // #any other case
224   print "Error in first argument\n"
225   show_help.call
226   end #case
227
228   assign_constants.call
229   handler = Client_Remover.new()
230   handler.remove($method,ARGV[1])
231   handler.shutdown()
```

6.9 dhcp_handler.rb

```
     1  #!/usr/local/bin/ruby
     2  =begin
     3
##############################################################################
#
     4  # Script responsible for cheking the DHCP server and than apply pre-
defined     #
     5  #policy  such as restrict until authentication, remove after expiration
etc      #
     6  #-------------------------------------------------------------------
--------#
     7  # Dimitris Mistriotis 2003 (besieger@yahoo.com)
#
     8  #
#
     9
##############################################################################
#
    10  =end
    11
    12  begin
    13  require "mysql"
    14   rescue LoadError => load_err
    15     print "Load error!, type: #{load_err} \n"
    16     print "Perhaps you haven't installed MySQL - ruby interface, \n"
    17     print "which is necessary to run most parts of DAWN. \n"
    18     print "Try visiting http://www.tmtm.org/ja/mysql/ruby/ for more
information\n"
    19     exit(1)
    20  end
    21
    22  #Some globally used constants:
    23
    24  DhcpStatus = "/usr/bin/dhcpstatus"
    25  IPtables = "/sbin/iptables"
    26  IFconfig = "/sbin/ifconfig"
    27  Pid_file = "/var/run/dawn.pid"
    28
    29  assign_constants = proc {
    30          begin
    31          _constants_db = Mysql.new()
    32          _constants_db.connect(host="localhost", user="attribute_reader",
password = "attribute2003", db="local_AD")
    33          _consants_results = _constants_db.query ("SELECT * FROM
configuration");
    34          _constants_db.close
    35          rescue MysqlError => connect_db_error
    36                  print "Problem during constant values assigment,
exiting\n"
    37                  print "Error number: #{connect_db_error.errno}. , Error
message: #{connect_db_error.error } \n"
    38                  exit(3)
    39          end
    40          _consants_results.each_hash do |row|
    41            case row["Attribute"]
    42            when "wireless_device"
    43              Client_Interface = row["Value"]
    44            when "proxy_port"
    45              Proxy_port = row["Value"]
    46            when "subnet"
    47              Dhcpd_Subnet = row["Value"]
```

```
    48            end #case
    49          end #do
    50          _consants_results.free
    51
    52          #get my ip by using ifconfig
    53          My_IP = `#{IFconfig} #{Client_Interface} |grep "inet
addr"`.to_s[/\d\d\d.\d\d\d.\d*.\d*/]
    54
    55          print "Showing Constants:\n"
    56          print "wireless_device = #{Client_Interface} \n"
    57          print "proxy_port = #{Proxy_port} \n"
    58          print "listening ip = #{My_IP} \n"
    59          }
    60
    61  temp_dir_create = proc {
    62          _date = `date +%d%m`
    63          _pid =Process.pid
    64          Temp_dir = "/tmp/" + _date.chop + "-" + _pid.to_s + "/"
    65          #defined as aconstant since many parts of the program will use
it
    66          print "Creating temporary directory #{Temp_dir}\n"
    67          #create it
    68          system "mkdir --mode=0600 #{Temp_dir}"
    69          }
    70
    71  class New_User_handler
    72  #  constants for this class:
    73    Username = "dhcp_handler"
    74    Password = "dhcp084"
    75    Host = "localhost"
    76    Database = "dhcp_clients"
    77    Table = "Current_Clients"
    78  def initialize()
    79    print "initializing new user handler\n"
    80    begin
    81    #what happens if we can't connect to the database
    82    #note: the connection will be always-on for performance reasons
    83    @db_handler = Mysql.new()
    84    @db_handler.connect(host=Host, user=Username, passwd=Password,
db=Database)
    85    rescue MysqlError => connect_error
    86         print "Problem while connecting to #{Database} database.\n"
    87         print "Error message: ", connect_error.error, "\n"
    88         exit(2)
    89    end #begin
    90  end #initialize
    91
    92  def network_restrict()
    93    print "network level #{@latest_IP_address}\n"
    94    #the rules on IPtables will be add as a stack (-I option)
    95    #so they are placed in reverse order
    96    system("iptables -t filter -I INPUT -i #{Client_Interface} -s
#{@latest_IP_address} -m mac --mac-source #{@latest_MAC} -j newcomer_input")
    97    #these three rules basically say this: Allow only web connections to
this IP - mac pair
    98    #redirection of web traffic to this host follows:
    99
   100    system("iptables -t nat -I PREROUTING -i #{Client_Interface} -p tcp -s
#{@latest_IP_address}  -j newcomer_prerouting")
   101    system("iptables -t filter -I FORWARD -i #{Client_Interface} -s
#{@latest_IP_address} -m mac --mac-source #{@latest_MAC} -j newcomer_forward")
   102  end #network_restrict
   103
```

```
  104  def add(ip_address,mac_address)
  105    @latest_IP_address = ip_address
  106    @latest_MAC = mac_address
  107    begin
  108    print "adding newcomer #{ip_address} to local database\n"
  109    @db_handler.query("INSERT INTO #{Table} (IP_address, MAC_address,
Username, Domain, User_info,  Timestamp) VALUES
('#{@latest_IP_address}','#{@latest_MAC}', NULL, NULL, NULL, NOW())")
  110    rescue MysqlError => insert_error
  111    #basically do nothing with it because it's the first target of a DoS
attack
  112    print "Error code: ", insert_error.errno, "\n"
  113    end #begin
  114    network_restrict()
  115  end #add
  116
  117  private :network_restrict
  118  end #New_User_handler
  119
  120
  121  class DhcpStatus_handler
  122  #  constants for this class:
  123
  124    Dhcp_Last_time = "dhcpstatus_before.txt"
  125    Dhcp_This_time = "dhcpstatus_now.txt"
  126    Diff_file = "differences.txt"
  127
  128  def initialize()
  129    #chech if dhcpstatus exists and readable and executable
  130    #file initialization:
  131    system ("/usr/bin/dhcpstatus -s
#{Dhcpd_Subnet}>#{Temp_dir}#{Dhcp_This_time} 2>/dev/null")
  132    system ("touch #{Temp_dir}#{Dhcp_Last_time}")
  133    system ("touch #{Temp_dir}#{Diff_file}")
  134    #new user handler:
  135    @newcomer = New_User_handler.new
  136  end #initialize
  137
  138  def process_changes()
  139    #old file = previous check new one
  140    system ("mv #{Temp_dir}#{Dhcp_This_time}
#{Temp_dir}#{Dhcp_Last_time}")
  141    system ("/usr/bin/dhcpstatus -s
#{Dhcpd_Subnet}>#{Temp_dir}#{Dhcp_This_time} 2>/dev/null")
  142    system ("diff #{Temp_dir}#{Dhcp_Last_time}
#{Temp_dir}#{Dhcp_This_time} >#{Temp_dir}#{Diff_file}")
  143    if (File.stat("#{Temp_dir}#{Diff_file}").size? != nil) then #there are
data in the file
  144     @diff_file = File.open ("#{Temp_dir}#{Diff_file}")
  145
  146     #Here eof is handled as an exception raised so when we are out of
  147     #input (because of EOF) file will be automatically closed.
  148     while (true)
  149     begin
  150      @input = @diff_file.readline()
  151      print "---> #{@input}"
  152      if (@input =~ /IP address/) then
  153      #something has changed with an address
  154      @ip_address = @input[/\d\d\d.\d\d\d.\d*.\d*/]
  155      print "ip address: #{@ip_address}\n"
  156       if (@input =~ /FREE/) then
  157        #an ip address has changed from free to active (== has been
assigned)
```

```
158         #consume three lines of input and get MAC from the fourth
159         #for _count in 0..3
160         @mac_address = nil
161         while (@mac_address == nil)
162           @input = @diff_file.readline()
163           print "data consumed (while searching for MAC): #{@input} \n"
164           @mac_address = @input[/([\da-f][\da-f][:]){5,5}[\da-f][\da-f]/]
165         end #for
166         print "MAC address: #{@mac_address}\n"
167         #we have ip and mac so client's information can be processed
168         @newcomer.add(@ip_address,@mac_address)
169       else
170         #there are two cases now or the user has left local AD (and has
to be removed) or
171         #there is a re-request for dhcp so time information has changed
172         #in that case ip still remains active
173         print "user left AD or dhcp re-request\n"
174       _remove_flag = "down"
175       while ( !(@input =~ /IP address/) or (@diff_file.eof != true) )
176           @input = @diff_file.readline()
177           print "searching for remove flag-> #{@input} "
178           #the check will be performed here inside the loop
179           if (@input =~ /FREE/) then
180           #the address was active and now is free
181             print "Set remove flag up\n"
182             _remove_flag = "up"
183           end #if
184       end #while
185          if (_remove_flag=="up") then
186            system ("./user_remover.rb  -i #{@ip_address} end_lease")
187          end #if
188       #must be located in teh same folder with this program
189       end #if
190      end #if IP address
191     rescue EOFError => error
192      @diff_file.close()
193      break
194     end #begin
195     end #while
196     @diff_file.close
197    end #if
198
199  end #process_changes
200  end #class
201
202  class Authenticated_Users_handler
203  #Again class_constants
204  Auth_user = "dhcp_handler"
205  Auth_pwd = "dhcp084"
206  Auth_host = "localhost"
207  Auth_db = "dhcp_clients"
208  Auth_table = "Authenticated_IPs"
209  def initialize()
210    print "A class responsible for giving proper permissions to already
authenticated users\n"
211    @user_db_hander = Mysql.new()
212    begin
213    @user_db_hander.connect(host=Auth_host, user=Auth_user,
passwd=Auth_pwd, db=Auth_db)
214    #again this conncection will be always-on for performance reasons
215    rescue MysqlError => _connect_error
216         print "Problem while connecting to #{Auth_db} database.\n"
217         print "Error message: ", _connect_error.error, "\n"
```

```
218         exit(2)
219       end #begin
220     end
221
222     def process_changes()
223       _newcomers = @user_db_hander.query("SELECT * FROM #{Auth_table} LIMIT
10")
224       #limit is set to ten so that the script will never stop here
processing many newcomers
225       #the rest of them will be proccessed very soon
226       if (_newcomers.num_rows() > 0 ) then # Check and proceed ifonly there
are results to process
227         _newcomers.each() {|ip_address| rearrange_ip(ip_address)  }
228       end #if
229       _newcomers.free
230     end #process_changes
231
232     def rearrange_ip (ip_address_to_process)
233
234     #testing reasons:
235     #ip_address_to_process="192.168.0.254"
236       #here the opposite of the previous rules are applied (-I becomes -D)
237       _temp = `#{IPtables} -L |grep #{ip_address_to_process}`
238       _mac = _temp.to_s[/([\dA-F][\dA-F][:]){5,5}[\dA-F][\dA-F]/]
239       #because mac address is also needed i managed this work-around
240       #here the opposite of the previous rules are applied (-I becomes -D)
241       system("iptables -t filter -D INPUT -i #{Client_Interface} -s
#{ip_address_to_process} -m mac --mac-source #{_mac} -j newcomer_input")
242       system("iptables -t filter -I INPUT -i #{Client_Interface} -s
#{ip_address_to_process} -m mac --mac-source #{_mac} -j authenticated_input")
243       system("iptables -t filter -D FORWARD -i #{Client_Interface} -s
#{ip_address_to_process} -m mac --mac-source #{_mac} -j newcomer_forward")
244       system("iptables -t filter -I FORWARD -i #{Client_Interface} -s
#{ip_address_to_process} -m mac --mac-source #{_mac} -j authenticated_forward")
245       #removing from prerouting chain
246       system("iptables -t nat -D PREROUTING -i #{Client_Interface} -p tcp -s
#{ip_address_to_process}  -j newcomer_prerouting")
247
248       #these three rules basically say this: Allow only web connections to
this IP - mac pair
249       #redirection of web traffic to this host follows:
250       system("iptables -t nat -D PREROUTING -i #{Client_Interface} -p tcp -s
#{ip_address_to_process}  -j newcomer_prerouting")
251
252       #after the network part, this IP can be deleted from temp space
253       begin
254       @user_db_hander.query("DELETE FROM #{Auth_table} WHERE ip_address =
'#{ip_address_to_process}'")
255       rescue MysqlError => _err #in order to have error here a problem has
happened during instalation
256           print "Error message: ", _err.error, "\n"
257           exit(2)
258       end #begin
259
260     end #rearrange_ip
261
262     private :rearrange_ip
263     end #Authenticated_Users_handler
264
265     #signal handlers
266     trap ("SIGINT", "SIG_IGN")
267     trap ("SIGQUIT", "SIG_IGN")
268     #so when parent-initialization process ends this program will continue
```

```
to operate
   269  #but when it's killed then it will die:
   270  trap 9, proc { print "Terminating dhcp_handler: #{$$}\n"
   271                 print "Killing proxy with pid #{Java_proxy}\n"
   272                 system "kill -9 #{Java_proxy}"
   273  }
   274
   275  #/----------------\
   276  #| -->  main  <-- |
   277  #\----------------/
   278
   279  #Assign values to constants after reading them from local database.
   280  assign_constants.call
   281
   282  #before beginning program operations, Java proxy is being initialized
   283  Java_proxy = fork
   284  if (Java_proxy == nil) then
   285  #we are in child process
   286   exec("java Proxy #{Proxy_port} #{My_IP} 1>/dev/null 2>&1")
   287   #by calling exec, the same pid will be used, usefull on killing from
parent process
   288  end #if
   289
   290  #write pid to appropriate file
   291  system "touch #{Pid_file}"
   292  system "echo #{$$} >#{Pid_file}"
   293
   294
   295  temp_dir_create.call
   296  $main_parser = DhcpStatus_handler.new()
   297  $user_handler = Authenticated_Users_handler.new()
   298  while (true) #main loop
   299    print "." #I am alive dot
   300    sleep(2) #wait 2 seconds between procecing
   301    $main_parser.process_changes()
   302    $user_handler.process_changes()
   303  end #main loop
```

6.10 init_dawn.sh

```
#!/bin/bash

###############################################
# Shell script used to start/stop DAWN        #
# Must be placed in /etc/rc.d/init.d directory #
#---------------------------------------------#
# Dimitris Mistriotis <besieger@yahoo.com>    #
###############################################

#Some variables
pid_file=/var/run/dawn.pid
program_root=/usr/local/dawn
executable=$program_root/bin/dhcp_handler.rb
firewall=$program_root/bin/prepare_iptables.sh
log_file=/var/log/dawn.log



#actions defined as functions
function start_dawn()
{
echo starting DAWN
#clearing from possible unfinished sessions:
rm -rf $pid_file
touch $pid_file
#prepare firewall
$firewall >>$log_file
#execute the main file
nohup $executable 1>>$log_file 2>>$log_file &
return
}

function stop_dawn()
{
echo Terminating dawn
dawn_pid=`cat $pid_file`
kill -9 $dawn_pid
return
}

#do actions according to command line input
case $1 in
'start')
start_dawn
;;
'stop')
stop_dawn
;;
'help')
#printing a quick help message
echo help mode, usage:
echo $0 start
echo starts DAWN services, while
echo $0 stop
echo is used to stop DAWN from running
;;
*)
echo incorrect usage of program, try using help as first argument
;;
esac
```

6.11 import_commands.sh

Purpose of this script, used at installation time is to upload information relative with this project to MySQL database. Straightforward code with no decisions made.

```
 1  #!/bin/sh
 2  echo be sure for priviledges while importing as well as that
 3  echo mysql is up and operating
 4
 5  #Create databases and import table structure
 6  mysqladmin create local_AD
 7  mysql local_AD <$1/local_ad_information.sql
 8  mysqladmin create dhcp_clients
 9  mysql dhcp_clients <$1/dhcp_clients.sql
10
11  #now GRANT permissions
12  mysql <$1/configure_grant_tables.sql
```

6.12 install_script.sh

```
#!/bin/bash

###############################################
# Shell script responsible for installing DAWN #
#---------------------------------------------#
# Dimitris Mistriotis <besieger@yahoo.com>     #
###############################################

#Values Assigment
#target directory for main functions
program_root=/usr/local/dawn
#cgi-bin directory
cgi_bin=/var/www/cgi-bin/
#which considered as a standard among linux distributions
init_dir=/etc/rc.d/init.d/
run_level3_init=/etc/rc.d/rc3.d/
#and therefore they are hard-coded.

#set default owner of files
owner=root
#set mask for file installation default owner can rwx, group r-x, others
---
bin_filemask=047
other_filemask=046

if [ $UID -ne 0 ]; then
echo "You must be root in order ot perform DAWN installation"
exit 1
fi

echo DAWN installation script
echo Creating target directories
#refer to documentation for directory structure
mkdir -p $program_root 2>/dev/null
mkdir -p $program_root/bin 2>/dev/null
mkdir -p $program_root/doc 2>/dev/null
mkdir -p $program_root/var 2>/dev/null
mkdir -p $program_root/var/additional_components 2>/dev/null
mkdir -p $program_root/var/initialization_files 2>/dev/null
mkdir -p $cgi_bin 2>/dev/null

echo installing files
#... one by one
echo bin directory
install dawn/bin/user_remover.rb $program_root/bin --mode=$bin_filemask
--owner=$owner  --verbose
install dawn/bin/dhcp_handler.rb $program_root/bin --mode=$bin_filemask
--owner=$owner  --verbose
install dawn/bin/prepare_iptables.sh $program_root/bin --
mode=$bin_filemask --owner=$owner  --verbose
install dawn/bin/Proxy.class $program_root/bin --mode=$bin_filemask --
owner=$owner  --verbose
install dawn/bin/ProxyThread.class $program_root/bin --
mode=$bin_filemask --owner=$owner  --verbose
echo doc directory
install dawn/doc/dawn_documentation.sxw $program_root/doc --
mode=$other_filemask --owner=$owner --verbose
echo var directory
install dawn/var/Proxy.java $program_root/var  --mode=$other_filemask --
owner=$owner --verbose
echo cgi-bin
```

```
    54  install dawn/to-cgi-bin/login-cgi.rb $cgi_bin --mode=$bin_filemask --
owner=$owner  --verbose
    55  install dawn/to-cgi-bin/login-page.rb $cgi_bin --mode=$bin_filemask --
owner=$owner  --verbose
    56  echo init directory
    57  install dawn/to-init.d/init_dawn.sh $init_dir --mode=$bin_filemask --
owner=$owner  --verbose
    58
    59  echo OK with file installation, copying additional files
    60  #don't care about file permissions
    61  cp dawn/var/additional_components/*.*
$program_root/var/additional_components
    62  cp dawn/var/initialization_files/*.*
$program_root/var/initialization_files
    63
    64  echo initializing MySQL Database
    65  bash dawn/var/initialization_files/import_commands.sh
dawn/var/initialization_files
    66
    67  echo Creating symbolic links in $run_level3_init
    68  ln -s $init_dir/init_dawn.sh $run_level3_init/S99dawn
    69  chmod +x $run_level3_init/S99dawn
    70  ln -s $init_dir/init_dawn.sh $run_level3_init/K99dawn
71  chmod +x $run_level3_init/K99dawn
```

Appendix I - Short introduction to Ruby

This Appendix does not intent to be an introduction to Ruby programming language, or a complete tutorial by any means. Tend to look at it as cross-reference between Ruby and a typical Object Oriented language like Java, so some similarities in concepts and differences in syntax will be illustrated here.

References

Two documents can be considered essential:

- "Programming Ruby, The Pragmatic Programmer's Guide" and
- Ruby User's Guide

Fortunately these two can be obtained very easily: the former is installed with other Ruby documentation, so we can easily say that it follows every installation, the latter can be obtained by language's web site (www.ruby.org) under documents link.

Variable Scope

An issue that rises with all scripting languages is variable scope since every one tends to use different symbols. A table from "Ruby User's Guide" can be considered useful:

`$`	global variable
`@`	instance variable
`[a-z]` or `_`	local variable
`[A-Z]`	constant

Code blocks

This section is placed here because of questions risen from people reading ruby source code for first time. Code blocks begin with "begin" reserved word and end with "end" reserved word. Inside each block an exception might rise, which can be captured with "rescue" reserved word into a variable and

handled accordingly. This exception mechanism has been used extensively because it gives the ability to produce more elegant and easy to read source code.

Example:

```
begin
require "mysql"
 rescue LoadError => load_err
   print "Load error!, type: #{load_err} \n"
   print "Perhaps you haven't installed MySQL - ruby interface, \n"
   print "which is necessary to run most parts of DAWN. \n"
   print "Try visiting http://www.tmtm.org/ja/mysql/ruby/ for more
information\n"
   exit(1)
end
```

(originating from section 6.9)