**System Documentation**

Client Company: Plaintech

Project VIRT team 6

Authors:

Cyril Adjei 500695104

Annika de Graaf 500662415

Jast Hamelink 500693975

Abdel Ochan 500661495

Brian van der Raaij 500639928

Version: 0.1

Status: Concept

Date: 4 Januari 2015

ITopia is subsidiary of the

Hogeschool van Amsterdam

Versioning

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ver.** | **Status** | **Date** | **Author(s)** | **Changes** |
| 0.1 | Concept | 04/01 | Annika de Graaf | Added all the separate chapters into this one document |
|  |  |  |  |  |
|  |  |  |  |  |

**Approval**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Carried out by** | | **Checked** | | **Approved** | |
| **Ver.** | **Name** | **Date** | **Name** | **Date** | **Name** | **Date** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Summary

This document contains information on the working of the product and its separate units. Aside from the working of the product, this document describes how the product was build and how it can be changed if needed to fit Plaintech’s needs. Each chapter will describe a certain part of unit of the system.

Table of Contents

1. Introduction 4

2. Basic Server Installation 5

2.1 Operating System Installation 5

2.2 Default Services Installation and Optimisation 9

2.2.1 Accessing Server: PuTTY 9

2.2.2 Installing and Optimising MySQL 10

2.2.3 Installing and Optimising Apache2 11

2.3 Libvirt and KVM 11

2.3.1 Default Installation 11

2.3.2 Network Bridging 12

2.3.3 Libvirt Storage Pool 13

2.3.4 Configuring Libvirt for VNC 14

2.3.5 Installation of Virtual Fileserver 15

3. Mailserver Installation 17

4. Website Coding and Configuration 20

4.1 Website Components 20

4.2 Front-end Pages 20

4.2.1 Content 20

4.2.2 Layout 24

4.3 Servlets 25

4.3.1 Main Servlet 25

4.3.2 Database Connection Servlet 29

4.3.3 Information Pulling Servlet 30

5. Security Configuration 32

5.1 Installing and Optimising Fail2Ban 32

5.2 Disabling Root Login 34

6. Backup Script 35

7. Maintenance 37

# Introduction

.

# Basic Server Installation

In this chapter the installation of the main server and the needed services will be explained. This chapter will provide you with an easy to follow guide on installing all the software components needed for the main server.

## Operating System Installation

The Plaintech server uses Debian (wheezy) as its OS or operating system. This chapter will show step by step how to completely install Debian on a server.

To install an operating system on a server one will have to make sure to have a Debian install image available. The easiest way to do this is to put the downloaded image on a CD or USB device, preferably using an existing tool to do so to avoid complications, and boot this image onto the server.

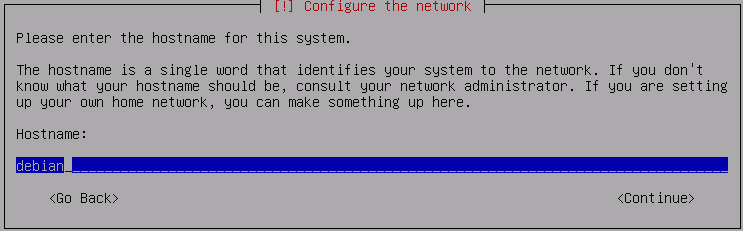
After successfully booting the server you will see the following screen;

Select first option, “Install”, from the menu and press Enter.

After completing the previous step, select English as language for the operating system and press Enter. When the language is set to English, the menu will guide you through a few steps to specify the kind of English used. When it asks for the country, please select the country in which the server is located. For the next two menus the default settings will do, but they can be changed if the user wishes so.

After the country and language specifications have been set, the user will get to the next menu. This menu asks for the network configuration but before the user can continue to the actual configuring of the network, they have to name the server. The name of the server does not really matter, but we strongly advice to pick a suitable one since it might help you identify the specific server if more servers are used.

Enter a valid Hostname for the server and press continue.



When it asks for a domain name, please enter the existing Plaintech domain.



Next up is the password for ‘Root’. Root is the administrator and has access to everything. Since we do not want other people to have access to the whole system, we need to use a strong password. Do not forget this password as it will be used to for changing some settings later on in the installation and remains the most important password on the server.

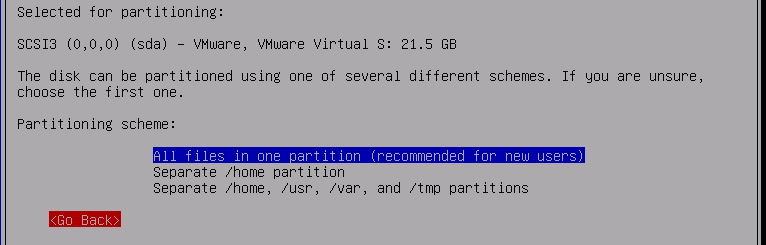


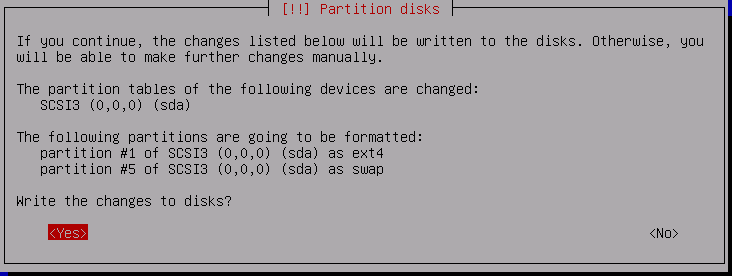
The system will ask for the name of the new user. This user basically has administrator rights and can become root. Enter a valid username. Afterwards you will have to enter a password for the new user. Remember that this is again a very important password as this account has by default access to root. This can be changed later on, but the password should be a strong one in any case.



For the next two menus we recommend the default settings.

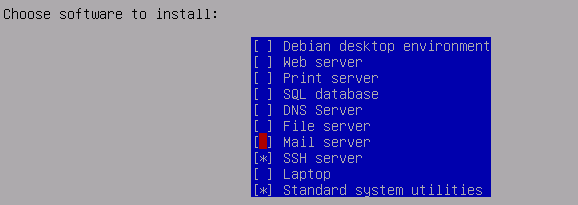
Make sure you select the correct hard drive to install to avoid probable data loss and choose the top option from the menu as shown below



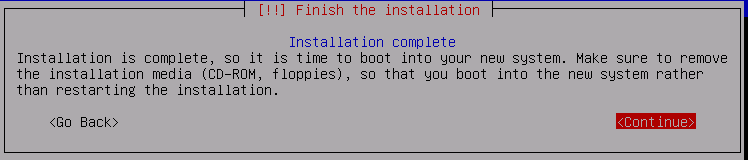
Press enter a few times and select yes from the screen as shown below.

The default settings are usually fine for the next screens. Press enter until the Select and install software window appears, as shown below. This can take a few moments to complete.

Select the options shown in the image below using the SPACEBAR. Press enter to continue the installation and select yes at the next shown screen.



When you see the screen below the default installation is done.



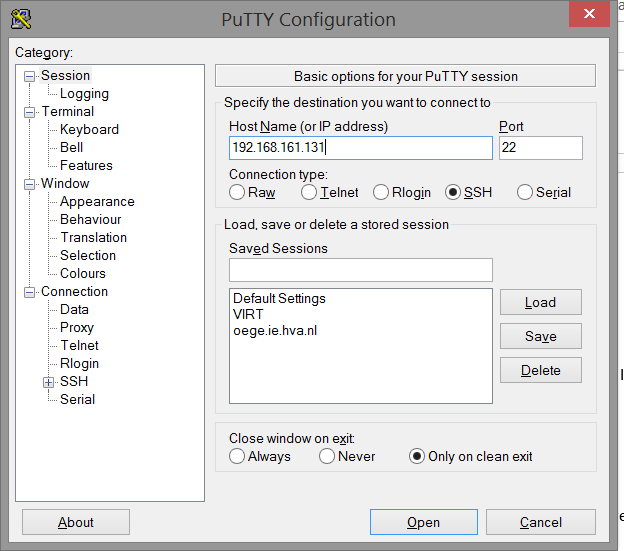
## Default Services Installation and Optimisation

For some programs to work properly they need certain services to run. In this part of the chapter we will show you how to install these services.

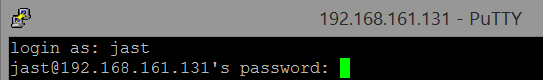
### Accessing Server: PuTTY

Even though this is not a service on the server, it is still an important one. As the server does not have a graphical interface, we will need a third party program to connect to the server. This program does not necessarily have to be PuTTY, but any program in which we can access the command line will do. We chose for PuTTY since it is a free, lightweight program that does the trick, but again, this is entirely up to you. Since we installed the SSH service together with the operating server we can use PuTTY to connect to the server. Download PuTTY from their website and execute the file with the .exe extension.

Enter the server’s IP address in the Host Name field. And press ‘open’ to connect.



After connecting with the server, log in with your credentials and you’re ready to go.



### Installing and Optimising MySQL

MySQL is used to run a database on a server. We will install this service trough PuTTY (Chapter 2.1).

Make sure you are logged in with a root account on the server. Use the command ‘*sudo su’* to activate your root rights followed by your password. Without being able to access the root account, one cannot install anything. Make sure your operating system is running the latest version by using ‘*apt-get update’* and ‘*apt-get dist-upgrade’* afterwards, these commands search for updates and install them if there are any which can take a while to complete.

To install the MySQL services use the command ‘*apt-get install mysql-server mysql-client’* and press ‘Y’ when prompted to start the installation.

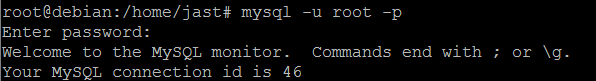
MySQL has its own user accounts, which are not related to the normal user accounts. The next step is setting up a secure password for the MySQL root account with the following command.

*Mysqladmin –u root –h localhost password ‘examplepassword’*

**

Where *examplepassword* should be replaced with your password.

To login to MySQL and use its functionalities type ‘*mysql –u root –p’* followed by your password as shown below.



Once installed, the database can be created and hosted on the server. To add/create a database, the SQL language can be used.

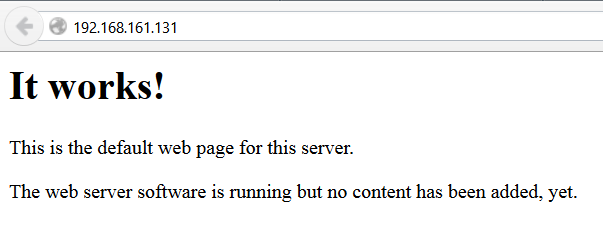
### Installing and Optimising Apache2

To run a website on your server you will need a webserver. For this purpose we will use Apache2.

Make sure you are logged in with a root account on the server. If you’re not sure how to do this, please refer for chapter 2.2.2.

Installing apache2 is relatively easy. Simply run the command ‘*apt-get install apache2’* and press ‘Y’ when prompted.

To test is the installation completed successfully enter your server IP address in any browser. If you see the message shown below you have successfully installed apache2!



## Libvirt and KVM

To run virtual machines we will need Libvirt & KVM.

Make sure you are logged in with a root account on the server. If you’re not sure how to do this, please refer to chapter 2.2.2.

### Default Installation

To install the services needed for Libvirt & KVM run the following command:

|  |
| --- |
| # apt-get install qemu-kvm libvirt-bin |

After the installation is complete you should grant rights to the account(s) which you want to be able to use libvirt. You do this by running the following command:

|  |
| --- |
| usermod -G libvirt -a username |

### Network Bridging

To allow multiple virtual machines to run on the main server you will need to create a bridge on your network card to allow virtual machines to access the internet.

At first, you will need a small piece of software, which can be installed by using the following command:

|  |
| --- |
| apt-get install bridge-utils vlan |

After installing the software go to the */etc/network/* folder on the server and make sure to create a backup of the file *interfaces* in case anything goes wrong.

Edit */etc/network/interfaces* with your favorite text editor to make it look like the example below and make sure you use the correct IP addresses.

|  |
| --- |
| # The loopback network interface  auto lo  iface lo inet loopback  # Management interface  auto eth0  iface eth0 inet manual  auto br-lan  iface br-lan inet static  address 10.10.0.10 #your ip address here  netmask 255.255.255.0 #your subnetmask here  gateway 10.10.0.254 #your default gateway here  bridge\_ports eth0  bridge\_stp off  bridge\_fd 0  bridge\_maxwait 0 |

To make your changes active type the following commands:

|  |
| --- |
| ifdown eth0 |
| killall dhclient |
| ifup br-lan |

To check if your changes have taken effect, run this command:

|  |
| --- |
| brctl show |

As a final check reboot the server and make sure it can still contact external IP addresses.

### Libvirt Storage Pool

To give Libvirt a place to install virtual machines we will have to assign libvirt a storage pool.

First you will have to create a disk partition to save all the virtual machines on. To check the current available partitions use the following command: fdisk –l

Look for available unused disk space and the name of the disk.

To create a new partition run the following commands:

|  |
| --- |
| fdisk /dev/sda #Where sda is the harddisk you want to modify |
| n |
| p #to make a primary partition |
| 3 #select a unused partition number |
| Enter #to select the first cylinder |
| Any number in mb to select where the new partition ends |

Next, you will have to label this partition for later use. We use the parted tool for this as shown below.

|  |
| --- |
| Parted /dev/sda3 #to select the new partition |
| mklabel |
| Labelname |
| Quit #to end the configuration |

Create a new folder for the partition to be mounted:

|  |
| --- |
| Mkdir /mnt/vms #file location can be anywhere |

Now we create a temporary xml file to create the storage pool:

|  |
| --- |
| Nano createpool.xml |
| <pool type='disk'>  <name>createpool</name>  <source>  <device path='*/dev/sda3*'/>  <format type='*labelname*'/>  </source>  <target>  <path>*/mnt/vms*</path>  </target>  </pool> |

To add the storage pool use:

|  |
| --- |
| virsh pool-define createpool.xml |

To make sure the pool added successfully type:

|  |
| --- |
| virsh pool-list --all |

You will notice that the just created pool isn’t running yet. We will want it to start and keep it starting automatically after reboots:

|  |
| --- |
| virsh pool-start |
| virsh pool-autostart *createpool* |

To verify the pool has been created successfully run:

|  |
| --- |
| virsh pool-info createpool |

You can delete the xml file we created before:

|  |
| --- |
| rm createpool.xml |

### Configuring Libvirt for VNC

To create a virtual machines there are a few things we will need.

To start, download the wanted operation system image from the internet. Move this downloaded .iso file to the ‘*/var/lib/libvirt/images’* folder.

Now since we will be using VNC for the virtual machines we will have to put Libvirt in listen mode.

Use the command below to edit the Libvirt configuration files.

|  |
| --- |
| Nano /etc/libvirt/libvirtd.conf |

Look for the line of text ‘*#listen\_tcp = 0’* and change it so it looks like the example in the image below.



Next open the libvirt-bin config file.

|  |
| --- |
| Nano /etc/default/libvirt-bin |

Change the sixth line to look like this which will make Libvirt automatically start in listen mode.



To activate the changes restart the Libvirt service with the command below.

|  |
| --- |
| /etc/init.d/libvirtd restart |

### Installation of Virtual Fileserver

To install virtual machines we make use of virt-install. To install this use the command:

|  |
| --- |
| apt-get install virtinst |

Now, to install the virtual machine run the following commands. The variables are self-explaining.

|  |
| --- |
| virt-install \ |
| --name virt6deb2 \ |
| --ram 256 \ |
| --vcpus 1 \ |
| --os-type=linux \ |
| --os-variant=debianwheezy \ |
| --cdrom /var/lib/libvirt/images/debian-7.7.0-amd64-netinst.iso \ |
| --disk pool=disks,size=10 --network=bridge:br-lan \ |
| --virt-type kvm \ |
| --noautoconsole --graphics vnc,listen=0.0.0.0,password=password |

Make sure your new virtual machine is running with the following command:

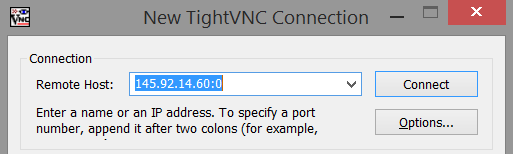
|  |
| --- |
| Virsh list |

To check at what port your virtual machine is running run use the following command:

|  |
| --- |
| Virsh vncdisplay virt6deb2 |

The output you get +5900 is the actual port where the server is running on.

To connect with the server open any VNC client, fill in the IP address and port and press connect.



If everything went good you will now see the splash screen of the installation of the operating system. Follow the guide in chapter 1 to complete the installation.

After the installation has completed install samba and cifs on the server:

|  |
| --- |
| apt-get install samba |
| apt-get install cifs-utils |

Also create users to use with samba:

|  |
| --- |
| smbpasswd –a username |

Create a folder you want to share anywhere on your system:

|  |
| --- |
| Mkdir /shared |

Edit the samba config file:

|  |
| --- |
| nano /etc/samba/smb.conf |

Add the following lines on top if the file:

|  |
| --- |
| [Sharezor] path = /shared  browseable = yes guest ok = no read only = no |

Reboot the samba services:

|  |
| --- |
| /etc/init.d/samba restart |

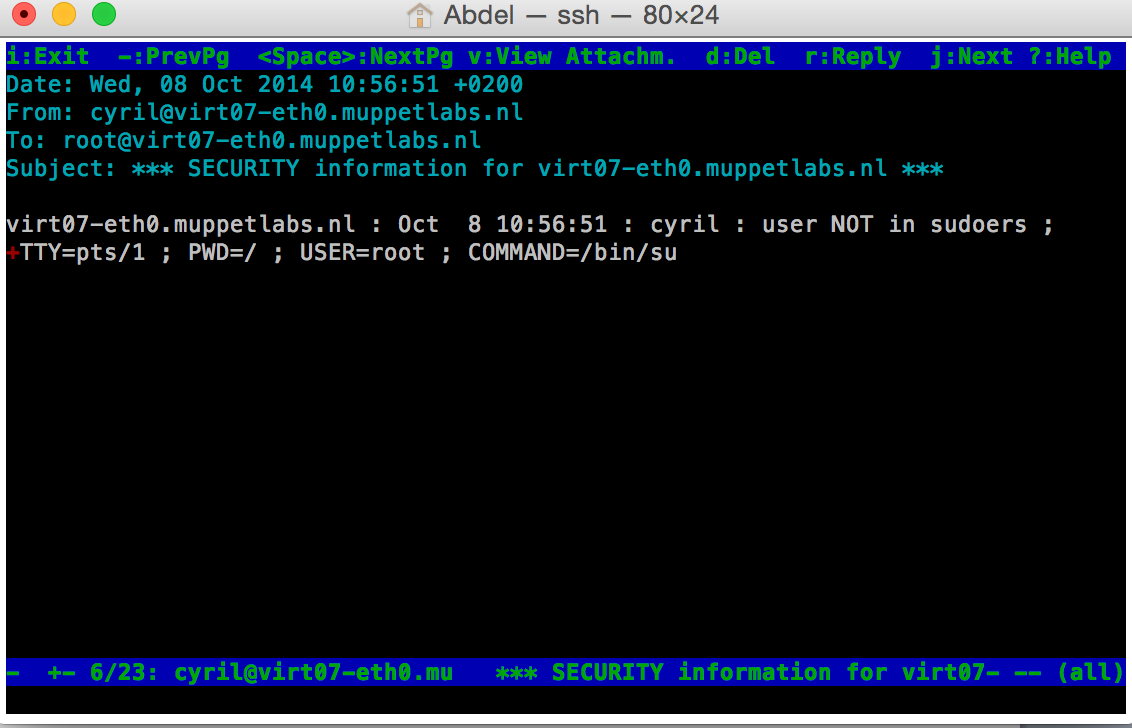
To mount the newly created share on the main debian server run the following command:

|  |
| --- |
| mount -t cifs //145.92.14.61 /sharezor /mnt/share -o user=username |

# Mailserver Installation

The mail server seemed like a requirement for us, because we used Fail2ban, it seemed like a good idea to, for example, let the administrator(s) know when a certain IP has blocked. Not only is it useful for Fail2ban, but if you redirect all mail from the root to an external email address, the administrator(s) can be up-to-date with system bugs and errors even without being logged into the server.

The software package we are using for the mail server is called Mutt. Mutt is software that works as mailbox, with this we are able to read all the system errors that the root account receives.



To send e-mails to the outside world you’ll need something called Smart Relaying. We use a certain URL for that, relay.muppetlabs.nl, this URL allows us to send an email with the root account of our server to the outside world. Useful things to do with this for example are:

* Letting a (hacked) user know that their account has been banned
* Sending notification(s) to server admin(s)

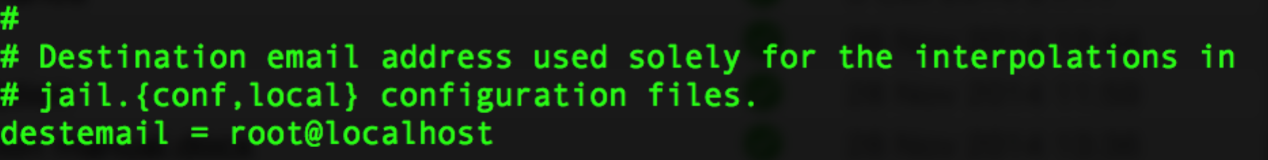
To make this work you need to create a small simple file in root called “.muttrc”, the contents should look like this:



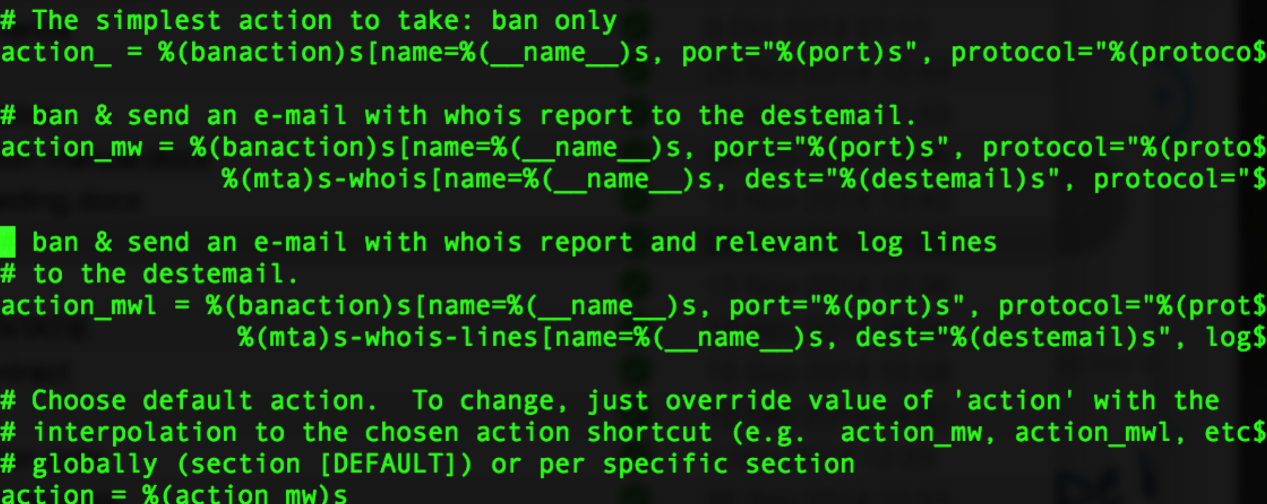
Mutt automatically uses these settings that you only have to save in root.

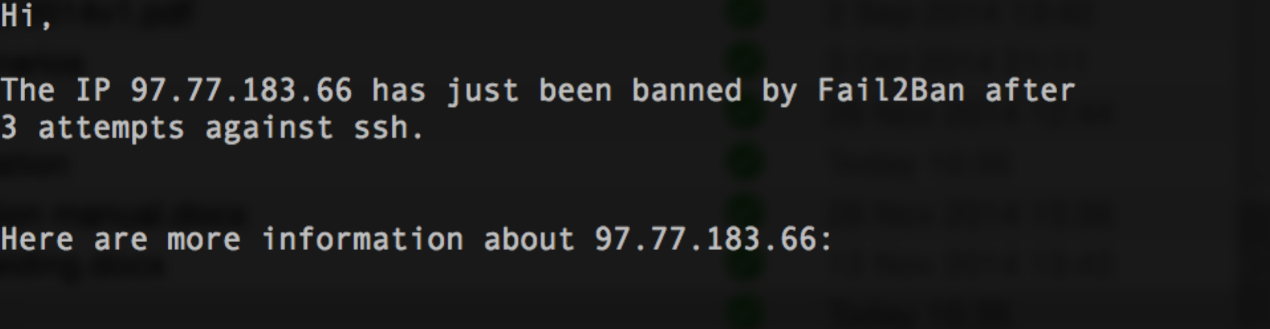
Exim4 is also required to complete your package, though you do not have to configure it after you install it, that’s where ‘sendmail’ comes into play. Installing sendmail comes with some configurations, though it can happen that sendmail doesn’t install everything. If it so happens that sendmail doesn’t create all the files required for you to get everything working, installing postfix is suggested. This can override the settings of both sendmail and Exim4 with the correct options selected and set it up automatically for you in a matter of seconds. This will result in a working system that is able to send e-mails to the outside world and forward them.

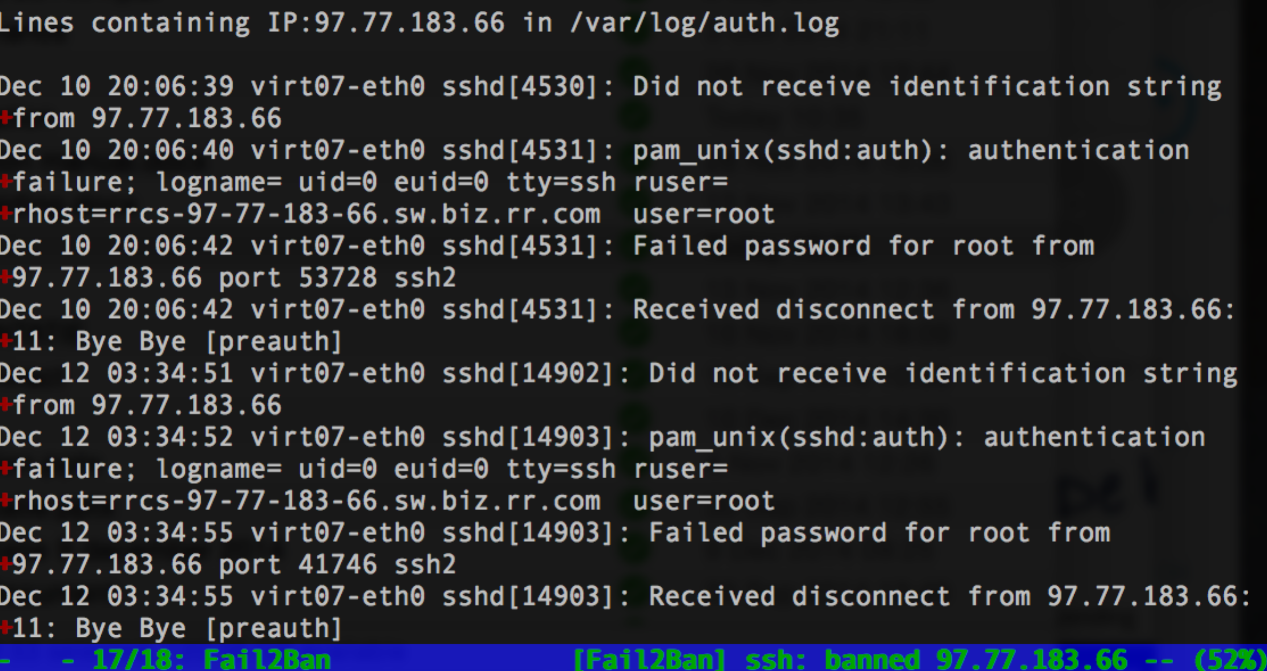
Another useful function that comes with fail2ban is the ability to report to a user account with a mailbox. This means you can get a detailed report who tried to perform a malicious attack called brute force on the server.



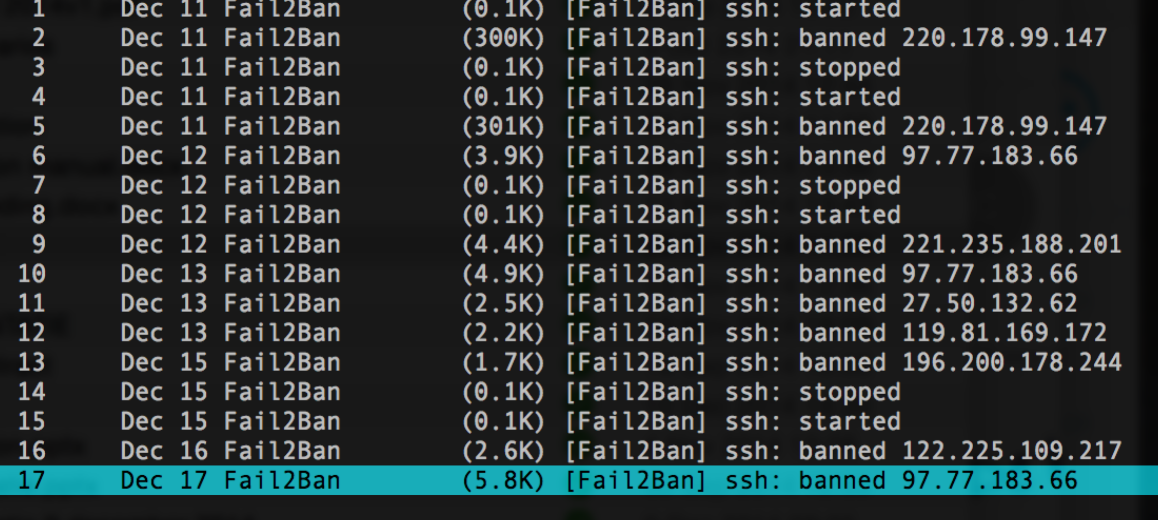
You can decide how much information you want to receive depending on the action you choose. (M, MW, MWL). MWL contains a log so you can see what account the attacker has been trying to access, this is also the most detailed action.







If you use mutt or mail (you must have mailutils installed!!) you will be able to see your emails. If you have enabled fail2ban successfully and allow it to send you reports, you should be able to see the following:



# Website Coding and Configuration

The website is the “front-end” part of the product. This is the part customers and potential customers will see and it will be the interface which they will use to access Plaintech’s services.

## Website Components

The website consists of multiple components, as you can see on the picture to the right. Basically, the upper half of the picture contains files which are “back-end” files and thus will not be seen by the customers, but are required to make the website work. The lower half of the picture contains “front-end” content that will be seen by the customers. All the files which are required for the website can be changed according to the wishes and needs of Plaintech. To make any changes, however, one needs to know where in the files the changes should be made. The next chapters will discuss the pages and how they could be changed.

## Front-end Pages

As mentioned in the previous chapter, there are both back-end files, which will not be seen by the customer and front-end files, which will be seen by the customer. In this chapter the front-end files will be explained so that Plaintech can change the website’s content according to their wishes.

### Content

The front-end of the website mainly consists of .html files. These files contain the information displayed in the browser, in other words this is what the customer sees. The content of the website is what generally changes most, as this is what the company wants their customers to know. The biggest part of each .html file is the same in our structure, as these are the head, the header and the footer.

Let’s take a look at the head first:

The head starts with a title, this can be changed to anything.

After that a set of characters, or charset, is defined and the links to the .css documents, the style documents, are provided. These links are very important as without them the website looks plain and basically only contains text. The changes that can be made to the style documents will be discussed in the next chapter.

Underneath the links, there is a block of code which is used for (outdated) Internet Explorer versions. The code is not necessary, but might come in handy when dealing with older browser versions.

However this part of the code does not contain any text content, it is a very important part of the code.

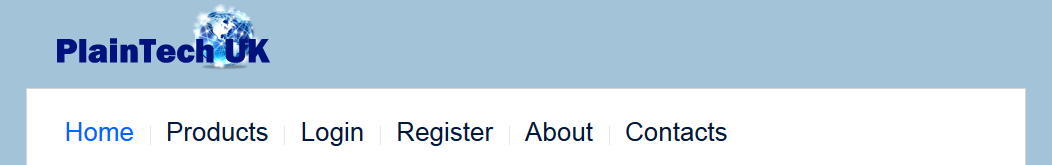
Next up is the header:



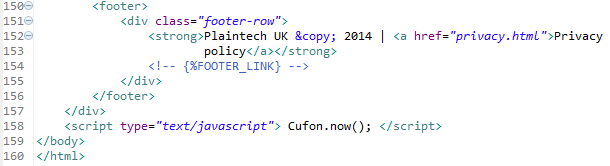
The header mainly contains the menu and the top bar of the website.

The only element in the header that is likely to be changed is the menu and its links.

To change the links of the menu, the part circled in red must be changed. Text of the links can be changed here by changing the black words, and links can be added or deleted by creating an exact copy of one of the <li> elements, or deleting a line containing a <li> element. Links usually only get changed when a new page is added to the website, or a page is removed or changed.

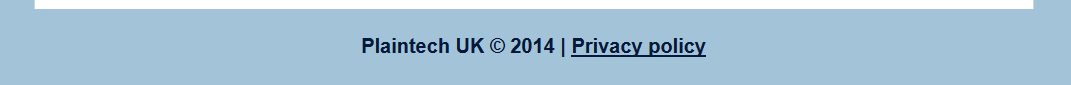
This is what the menu looks like with the code displayed above: 

Last of the three parts of the documents that are mostly the same on each individual page is the footer. The footer can be found all the way on the bottom of each document.

As you can see in the picture on the left, the footer is only a small piece of code.

All that is in the footer right now is the copyright disclaimer and the link to the privacy policy. For a lot of companies, this is all a footer will do. Sometimes more links are provided in the footer, but because the Plaintech website is a relatively small one, this is not necessary. However, if in the future the website will be expanded, more links might be handy. After the footer is closed (after the </footer> tag) the document is closed (</body></html>). This signifies the end of the page and nothing more will be put on this page.

With the code displayed on the previous page, this is what the footer looks like:



Aside from the content which is, more or less, the same in each document there is a lot more. This is the actual content. The content of a webpage is the part that changes most. The content of the index page, the page that is the first thing the customers see when entering the website, is one of the most viewed parts of every website and therefore is very important.

This is what the website looks like right now:



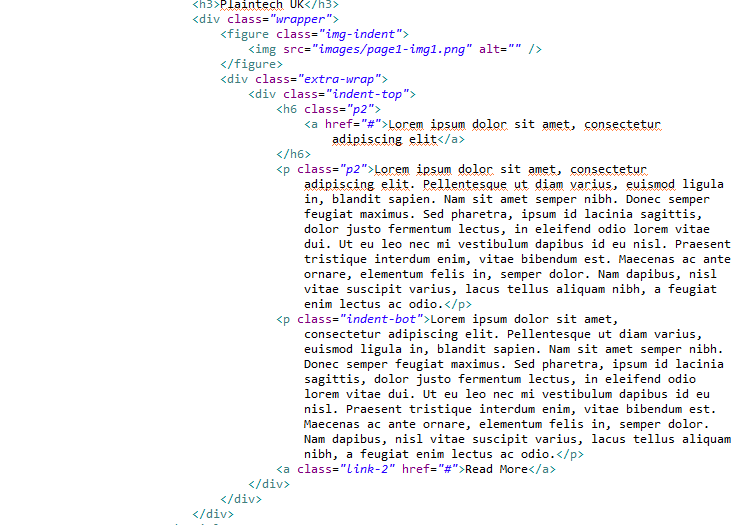
There is a lot on this page that probably will be changed in the future. So how can you change it? Let’s take a look at the code.

First of all let’s take a look at the three boxes which indicate three levels of VPS’es. The code for these boxes is the following:

In the picture on the right, only two of the three boxes their code are fully shown, however the third one is exactly the same.

As you can see, the boxes are contained by a lot of different <div> tags. These tags are mainly used for the style and should not be touched if you’re only changing the content.

The whole piece of code that lies within the red square is the first box.

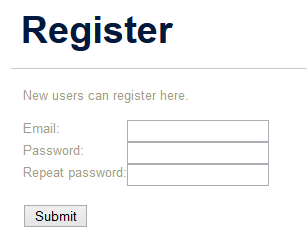
The text (in black) can be changed easily, everything else should only be changed if one wants to change the style.

The big part underneath the boxes which introduces Plaintech UK is very easily changeable too. The picture on the left displays the code.

This basically works the same as the former part with the boxes. The black text is just displayed as text and can be changed to whatever Plaintech wants it to be.

Since most pages have these boxes or big blocks of text, which can be changed according to the wishes of Plaintech as explained above, we will not show them all. There are, however, a few different elements on other pages which can be considered exceptions. These elements can be changed, however it can be more difficult to do so.

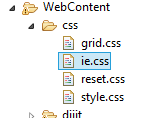
The register form is one of these exceptions.

Even though the register form looks easy, the code to create it might be considered not so easy.

The code for only the first two fields already is quite a lot:

To add a new field to the already existing ones, one needs to create a new block of code like these. When you do so, make sure that the blue text fits the new field. For example, do not give a “date of birth” field the id and/or name “surname”. This is confusing and might end up in unnecessary errors.

### Layout

Beside the actual conent, the layout of a website is crusial as well. Layout is proven to be even more important than text, as this is what catches the customer’s eye first. So how do you change the style of the website.

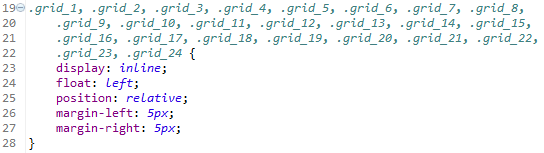
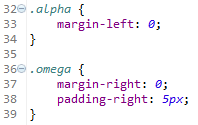
The layout of a website is mainly managed through .css documents, or Cascading Style Sheet documents.

These documents contain all the details on the style used for the website.

For the Plaintech website we used three local style sheets and one external. The external style sheet is used by Dojo, which we used for certain elements within the website. Since we cannot change this style sheet, we will not discuss it in this chapter. As you can see in the picture on the right, there are four style sheets available to us for the website. We will only be using grid.css, reset.css and style.css. The only reason ie.css is there is to make sure that some elements will be displayed differently in older web browsers, as these often cannot cope with the new coding of websites.

Let’s first of all take a look at the grid.css document. This document takes care of the placement of content on the website. For example, the boxes with the different types of VPS’es in the previous chapter are managed by the grid.css.

The code used to keep the first box in place is, for example:



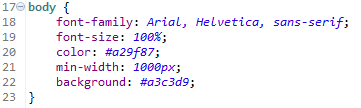


All grid.css does is place elements in the right places as shown above.

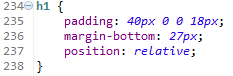
Reset.css doesn’t do much, and should not be touched so we will not discuss it. All it does is restore basic settings once certain elements are used.

Style.css is the biggest style sheet and the most important one. All style configuration except for the grid is done in this file.

A few examples from the style.css document are:



This is the configuration for the body. As you can see the font, the size of the font, the colour of the font, the minimal width and the background colour are defined. All the blue variables can be changed to whatever Plaintech desires.



This is the configuration for the biggest headings. As you can see the space around the heading and its position are defined.

The whole style.css document is filled with these small blocks of code to define what each element of the content should look like. All variables can be changed according to the wishes of Plaintech UK.

## Servlets

The servlets are the back-end part of the website. They make the website communicate with the server and make sure that certain actions are performed. We used three servlets to perform the actions. MyServlet.java is the main servlet and DBConnection.java and ServerInfoPull.java are the supporting servlets. To know how they work, and how they can be changed and/or optimised, let’s first look at MyServlet.java.

### Main Servlet

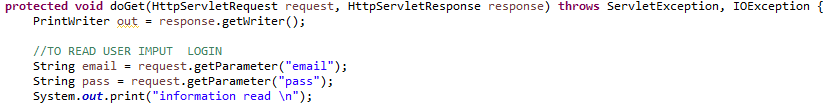
MyServlet.java is the servlet that the webpages connect to first when an action needs to be performed, whatever that action might be. If it cannot perform the action on its own and needs one of the other files it will open these files and run through them.

The first thing that can be seen aside from the imports which the java code uses is the following code:



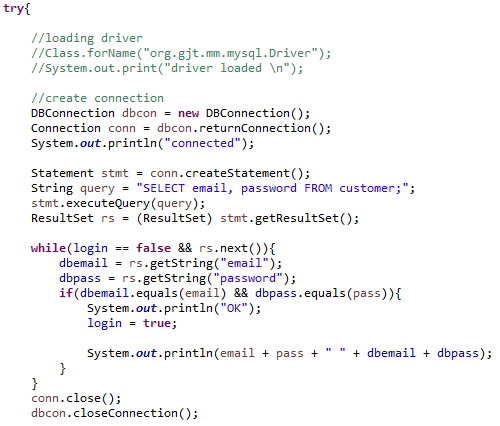
The Strings and Boolean that are seen here will be used later on, but are declared already.

The following code will be executed when the user tries to log in:



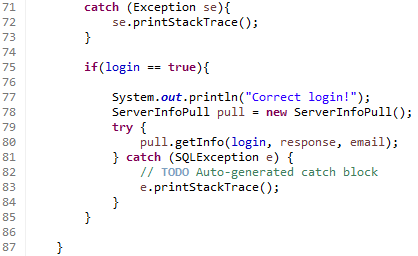
Once the user put in their email and their password, which are asked as their user credentials, the code will read them.

Then it will execute the next bit of code:

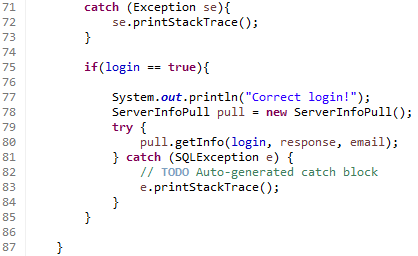
This piece of code will open the servlet in which the connection to the database is established. How the connection is established will be explained in the next chapter.

Once it ran through the code of that servlet, the servlet will return the connection to the main servlet.

Once there is a connection with the database, a query will be send to the database to select everything from the database so that the user’s credentials can be matched to the information in the database so that the user can log in if their credentials match. The information that the database returns, or results, will be put in a “ResultSet”. Once the information is received, a while loop is used to compare the user’s credentials to the results. Once the credentials match with the received information or the loop looped through all the results, the loop gets closed. Once the loop is closed, the connection to the database will be closed as this is no longer needed and the next piece of code will be executed.

This piece of code will only be executed if the user’s credentials do not match with the results from the database. It prints the exception if something went wrong so we can find the problem. If the user’s credentials did match the results from the database and it quit the loop because of it, this piece of code will be skipped and the next piece of code will be executed.

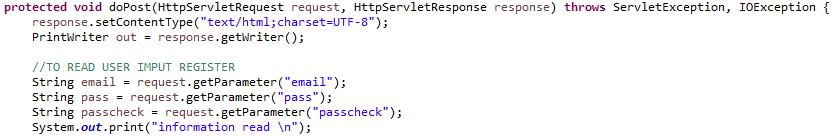
The next piece of code gets executed when the login credentials of the user match the information in the database:

When the user is successfully logged in, information of the user’s virtual server needs to be available to the user.

The main servlet will open the servlet that will get information of the server and run through it. How the other servlet works is explained in chapter 4.3.3.

To be able to run through it, the servlet gives some information, namely the login info, a response and the email of the user. If this works, the information can be used in the other servlet. If not, the catch will be ran through and the exception will be printed.

The next piece of code will be execute when a user tries to register on the website:



The email and the password that the user fills in will be read as well as the second password which will function to check the first one. Once it read the information, the next piece of code will be executed.

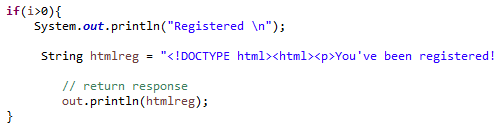
In this piece of code the passwords will be compared and if they match, the code is run through. The servlet that opens the connection with the database will be opened and ran through, after which the connection is returned to the main servlet. How the connection is made will be shown in the next chapter.

Once the connection is returned, a query that is defined in the next piece of code will be send to the database with the information that the customer filled in. This will then be saved into the database and can be used for logging in next time.

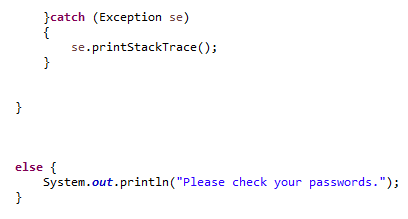


After this code is executed, the next piece of code will be executed.

This next piece of code merely checks if the registration worked. If it did work, then it will display a line saying that the user has been registered.

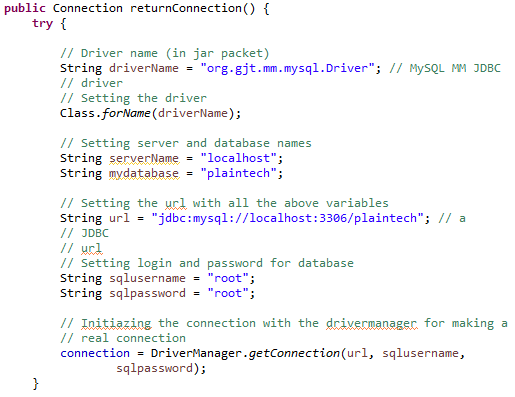


After the code has been run through, there is a little bit of code left in this servlet. This piece of code catches the remaining exceptions. This way if there is a problem it will always be displayed.



### Database Connection Servlet

As mentioned in the previous chapter, the DBConnection.java servlet is used to create a connection between the website and the database.

The following code creates the connection:

First the driver gets loaded since this is needed to create the connection.

The name of the server and the name of the database are put in a string so that we do not have to type the server name and the name of the database over and over again.

Then the exact location of the database is given in a URL. This needs to be the address from which the database can be accessed. After that the username and password that are used to access the database are put into a string, again so that we do not have to type it over and over again. Make sure that the user of which the credentials are used to access the database has enough rights, otherwise it will not work. Next the actual connection is established.

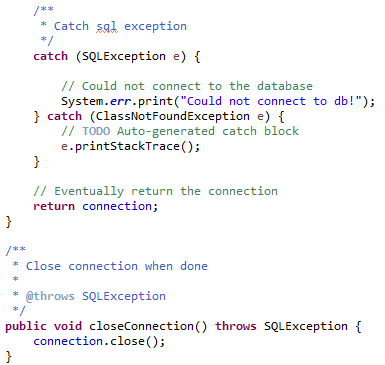
Once this part of the script is executed, the next part will be run through.

In this part of the script the exceptions, if there are any, will be caught.

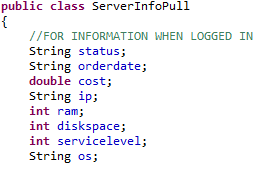
If the connection cannot be made, the problem will be shown as an error code because of the catch.

If the connection is established it will be returned so that it can be used in the main servlet.

At the end of the servlet the connection gets closed as it will not be used at the moment anymore.

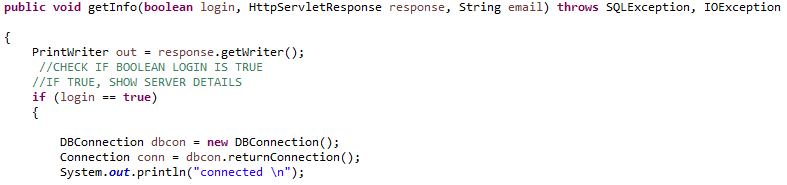


### Information Pulling Servlet

As mentioned in the former chapters, this servlet pulls information out of the database once the user is logged in. This servlet needs the other two servlets to work.

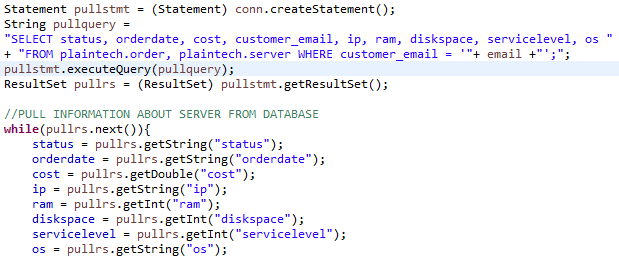
The first piece of code creates strings, doubles and integers for each piece of information we will need from the database. This is used so that we can later place the information and print it.

In the next piece of code the information from the main servlet gets used to check if the user is logged in correctly. If the user is logged in correctly, the servlet will open the servlet in which the connection to the database can be made and runs it (see the previous chapter).



After the connection is established it will return to the servlet and the next piece of code will be executed.

In this piece of code the query is created and send to the database. The information the database sends back will be put in a ResultSet.



Once information is received from the database and put in a ResultSet, the system runs through a while loop. It will keep lopping until all the results are put into the right strings, integers or doubles so that they are ready to be displayed.

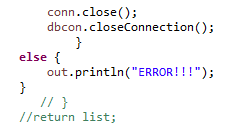
As soon as the system quits the loop, it will run through the next piece of code.

This piece of code is responsible for the printing out of the information that just has been received from the database.



It will print out a simple html table with the information in it. Later on, this shall be replaced with something neater.

Last but not least there is a small piece of code left to close this servlet, namely:



This piece of code closes the database connection as it is no longer needed and makes sure that any exceptions will be caught.

# Security Configuration

To keep the server and all the information on it secure, several security measures are required. As hackers or other people that are trying to access the server or the information for whatever reason have a lot of ways of attacking the server, there also are a lot of different measures that can be taken. We selected a few that we used on the server, however there are far more and different approaches to secure a server, so you should not limit yourself to the ones we used here. The server can never be a hundred percent secure and we are not responsible for any attacks or break-ins on the server, even if you use the security measures we suggested.

## Installing and Optimising Fail2Ban

Fail2Ban is a software package that is created to protect a server against malicious attacks like brute force attacks. To install Fail2Ban, one needs to be logged in as root. If you do not know how to do this, please refer to chapter 2.2.2.

To install the software package that is Fail2Ban, use the following command:

*Sudo apt-get install fail2ban*

Fail2ban has a default configuration file and is located at ‘/etc/fail2ban/jail.conf’. This file should be left alone and the configuration work should be done in a local copy of that file.

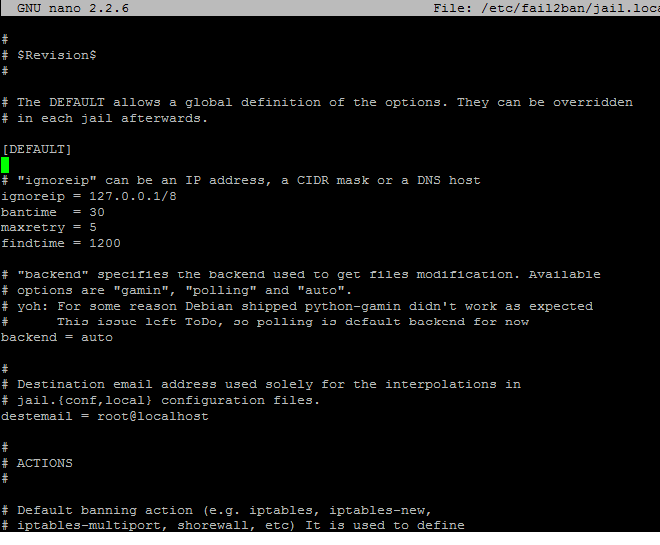
So we use the following command to copy the former file so we can freely edit it:

*Sudo cp /etc/fail2ban/jail.conf /etc /etc/fail2ban/jail.local*

We called the local copy ‘jail.local’. Now to open up the jail.local file:

*Sudo nano /etc/fail2ban/jail.local*

When the file is opened you see a default page, we’ve changed some values and added a few others. Please see the image on the next page.



Just below the [DEFAULT] rule there’s ‘ignoreip’, ‘bantime’ and ‘maxretry’ in the default page.  
We’ve adjusted the bantime from 600 seconds to 30 seconds because it’s a working project and it would be annoying if everyone gets banned once in a while.

We’ve added the line findtime = 1200.

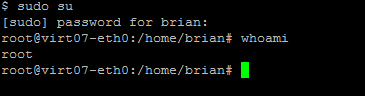
It might come in handy to know what these lines actually mean.  
**Bantime** is the actual time the user that tries to ‘brute force’ your server or typed their password incorrectly is banned. As explained above it’s just for 30 seconds now because it would take more time to unban our developers than to wait the 30 seconds.  
**maxretry** means the amount of time a user may get there password wrong or the amount of time a user could try to bruteforce your server.  
If the user has generated the “maxretry” within the last “**findtime**” the user gets banned.  
So the “findtime” is basically the amount of time the user gets to enter as many incorrect passwords as the value of “maxretry”

Now we need to restart fail2ban using the following command:

*Sudo service fail2ban restart*

## Disabling Root Login

First verify that you can switch to root (sudo su) with your account.

Code:  


Edit the ‘sshd\_config’ file

Code:  


Search the following line:



And change it to:

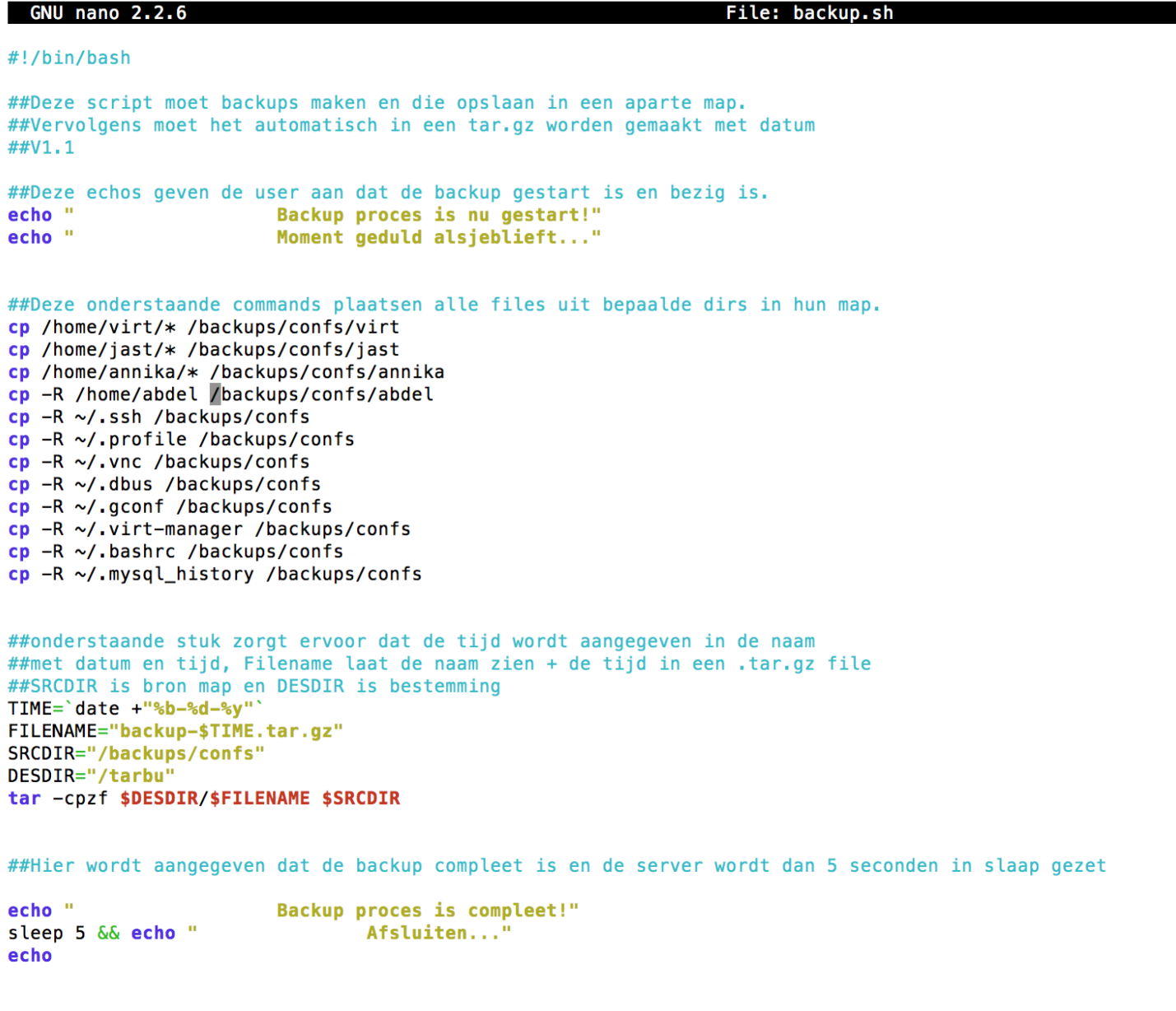


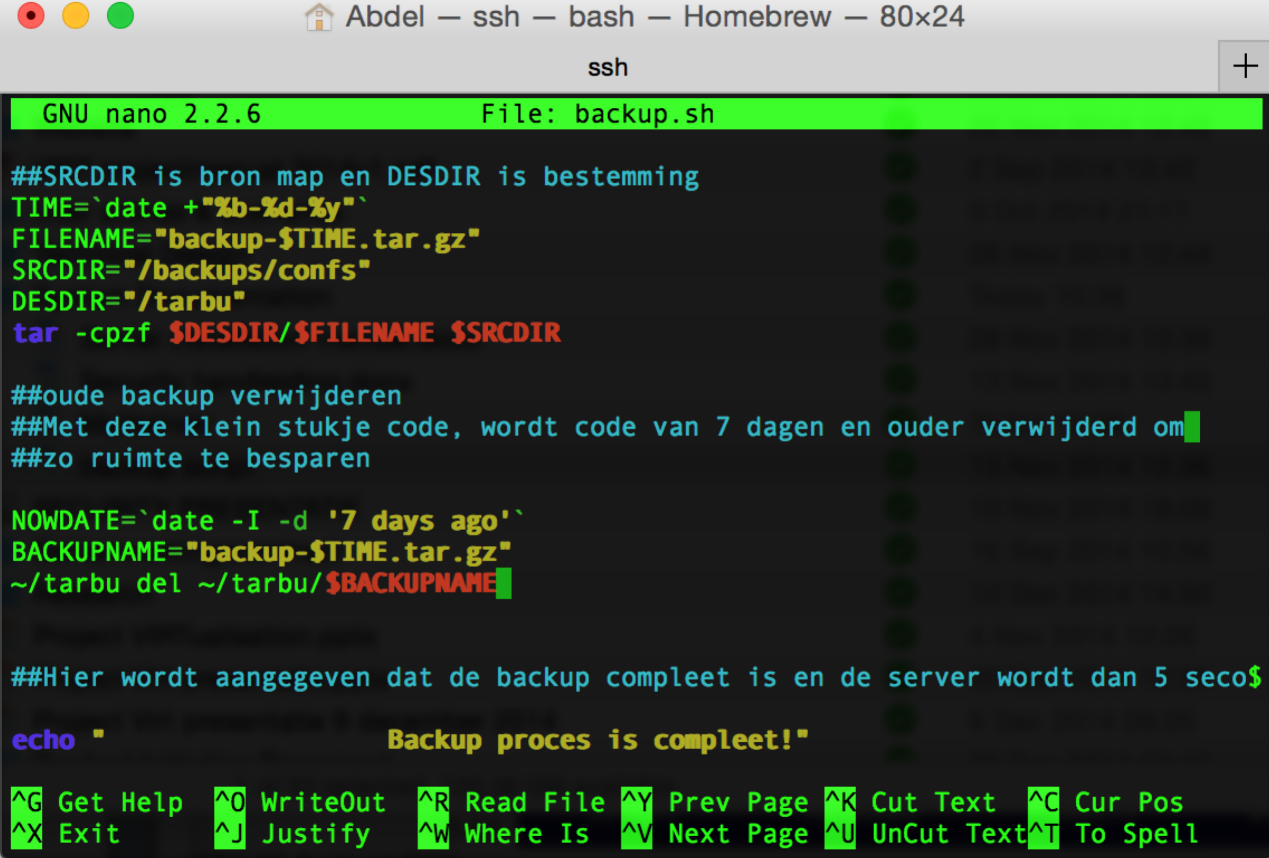
Now restart SSH with the following Command (in sudo su):  
*/etc/init.d/ssh restart*

And you’re all set.

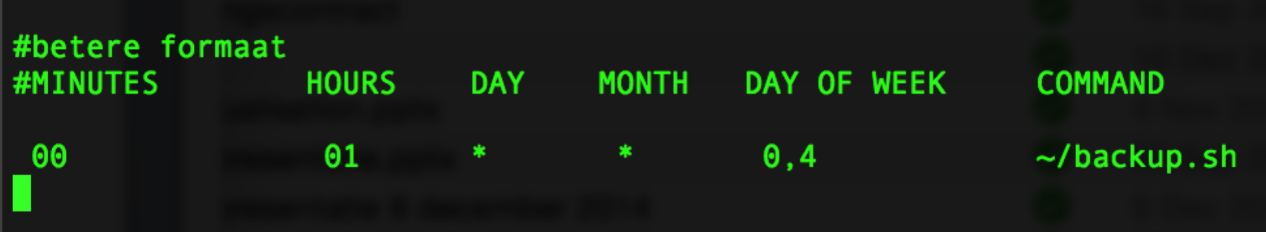
# Backup Script

The back-up system of the server is a requirement for 2 of the 3 SLAs we have with Plaintech UK. Medium and Heavy SLA’s. The script:

As you can see, some directories are being copies to a directory elsewhere on the server. You might have also noticed the ‘–R’. This stands for recursive copying. This mean it will copy any directory and subdirectory from the source you are using in the command line, which ensures that all files are copied. Below the command lines for the copying you can see 5 other lines. TIME indicates how the date is to be set on each document. Filename shows that the name of the file will be “backup <date>”. The source directory that is to be compressed into a tar.gz file is mentioned at SRCDIR. DESDIR is where the backup.tar.gz file will end up after being created. This directory is called /tarbu. The last line is to make the entire process happen. Tar is to execute the tar program, ‘-cpzf’ is to create, zip (gz stands for Gzip.) the ‘p’ stands for pasting and the “f” means it will use a file archive. Those are the red words in the same line.



The second part that comes after the part where the entire folder is compressed and then moved to a new directory, is the deletion part of the script. The amount of days can always be changed but for now it is set to 7 days ago. Though a smaller period might be better, in our current server this is more than enough because besides the script, we also regularly check and delete old backups so they will not take up too much space. We have those backups saved externally so they can always be added. This short piece of code detects file with the name backup-$TIME.tar.gz and deletes them if they are 7 days or older. The script runs depending on the times and date we choose to use in the ‘crontab’. (Command: *crontab –e*)



The last few lines of the script are the last few lines that tell you when an update is completed and then put the system to sleep for 5 seconds.

# Maintenance

Always make sure that the operating system as well as all the software on the server is up to date. Leaks that might be found in certain software or even the operating system are usually fixed in their next updates.

Not updating your machine leaves it vulnerable for malicious hackers and endangers the information stored on the server.