# abYsis

*Securing abYsis on hosted systems*

# Table of Contents

# Securing abYsis

Due to the varying nature of customer requirements and customers' IT infrastructure users should determine their own requirements. Below we provide some generic guidance regarding IT precautions that can be taken to secure abYsis but you should discuss with your IT department to determine what is required for your organisation and what best fits your IT structure.

## Firewalls

### Internal installations

An instance of abYsis would typically be kept behind a customer firewall to ensure it is not accessible by others. The firewall infrastructure will normally be configured by the customers' IT department and will ensure that no external access to abYsis is possible.

### AWS

If abYsis is installed on an external resource such as AWS then the AWS firewall needs to be configured.

Please refer to the section **Control Access through Firewall** in the document **abYsis - Using on AWS** (https://info.abysis.org).

## Username and Password Protection

### Creating a username and password

It is possible to protect the whole of your abYsis web site with a username and password. This will mean that only those with the username and password will have access. In line with common IT recommendations, the username and password that you use should be complex. There are a variety of ways to generate complex strings, including system commands and username/password managers.

Irrespective of how you generate them, record them securely as you will need them later to login in to the server web page.

Here is a 12 character example which we will use as the username:

```
7gw/eHUXuHpkKok0
```

And here is a 48 character example which we will use for the password.

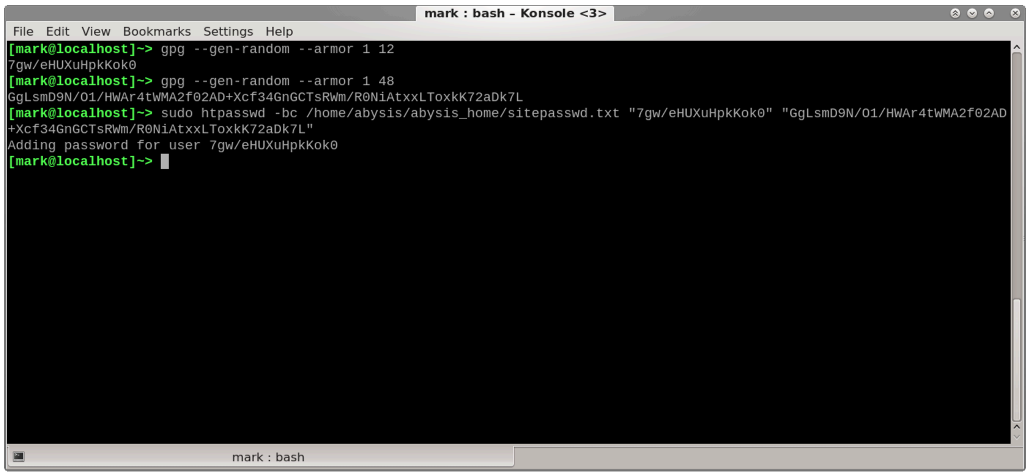
```
GgLsmD9N/O1/HWAr4tWMA2f02AD+Xcf34GnGCTsRWm/R0NiAtxxLToxkK72aDk7L
```

## Setting the username and password

To set the username and password. Use the following command. The command is all on one line and we have copied (shift-ctrl-c ) and pasted (shift-ctrl-p) the random strings from above into the command and put speech marks around them.

```
sudo htpasswd -b -c /home/abysis/abysis_home/sitepasswd.txt
"7gw/eHUXuHpkKok0"
"GgLsmD9N/O1/HWAr4tWMA2f02AD+Xcf34GnGCTsRWm/R0NiAtxxLToxkK72aDk7L"
```

This has now set the password


mark : bash – Konsole <3>
File Edit View Bookmarks Settings Help
mark : bash

```
[mark@localhost]~> gpg --gen-random --armor 1 12
7gw/eHUXuHpkKok0
[mark@localhost]~> gpg --gen-random --armor 1 48
GgLsmD9N/O1/HWAr4tWMA2f02AD+Xcf34GnGCTsRWm/R0NiAtxxLToxkK72aDk7L
[mark@localhost]~> sudo htpasswd -bc /home/abysis/abysis_home/sitepasswd.txt "7gw/eHUXuHpkKok0" "GgLsmD9N/O1/HWAr4tWMA2f02AD
+Xcf34GnGCTsRWm/R0NiAtxxLToxkK72aDk7L"
Adding password for user 7gw/eHUXuHpkKok0
[mark@localhost]~> 
```

## Adding password protection to the web site

Now you must add the commands to make the web server challenge you for these credentials.

To do this you need to edit the apache configuration file, abysis.conf. The abysis part of the filename is the same as the abysis part of the URL you use to access the server.

```
http://81.123.45.45/abysis/index.cgi
```

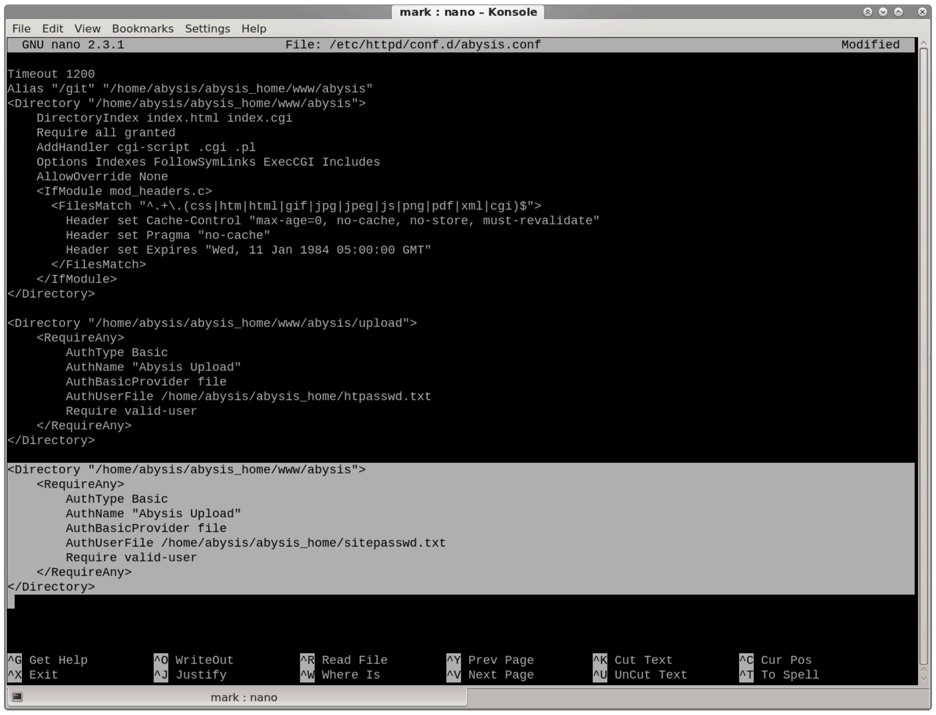
To edit this file use a text editor available on your Linux system. On our system we use an editor called nano (which may or may not be available on your own system).

```
sudo nano /etc/httpd/conf.d/abysis.conf
```

Add the following text to the end of the file. If your file has **<VirtualHost>** tags then make sure it is inside those **<VirtualHost>** tags.

```
<Directory "/home/abysis/abysis_home/www/abysis">
    <RequireAny>
        AuthType Basic
        AuthName "Abysis User"
        AuthBasicProvider file
        AuthUserFile /home/abysis/abysis_home/sitepasswd.txt
        Require valid-user
    </RequireAny>
</Directory>
```

Your file should now look like the following image (the inserted text won't be highlighted - that's just to show you where it is). Then save the file.



```
  GNU nano 2.3.1               File: /etc/httpd/conf.d/abysis.conf                        Modified

Timeout 1200
Alias "/git" "/home/abysis/abysis_home/www/abysis"
<Directory "/home/abysis/abysis_home/www/abysis">
    DirectoryIndex index.html index.cgi
    Require all granted
    AddHandler cgi-script .cgi .pl
    Options Indexes FollowSymLinks ExecCGI Includes
    AllowOverride None
    <IfModule mod_headers.c>
      <FilesMatch "^.+\.(css|htm|html|gif|jpg|jpeg|js|png|pdf|xml|cgi)$">
        Header set Cache-Control "max-age=0, no-cache, no-store, must-revalidate"
        Header set Pragma "no-cache"
        Header set Expires "Wed, 11 Jan 1984 05:00:00 GMT"
      </FilesMatch>
    </IfModule>
</Directory>

<Directory "/home/abysis/abysis_home/www/abysis/upload">
    <RequireAny>
        AuthType Basic
        AuthName "Abysis Upload"
        AuthBasicProvider file
        AuthUserFile /home/abysis/abysis_home/htpasswd.txt
        Require valid-user
    </RequireAny>
</Directory>

<Directory "/home/abysis/abysis_home/www/abysis">
    <RequireAny>
        AuthType Basic
        AuthName "Abysis Upload"
        AuthBasicProvider file
        AuthUserFile /home/abysis/abysis_home/sitepasswd.txt
        Require valid-user
    </RequireAny>
</Directory>


^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

Finally, restart **apache** so that abYsis is running with the new credentials with the following command.

```
sudo systemctl restart httpd
```

**Note**

- On your first visit to your abYsis web pages you will be asked for the username and password. Once you enter it your browser will usually remember it.
- You will however have to re-enter the username and password:-
- if you change your server settings (change ip address, domain name, change to HTTPS)
- if you clear your browser cache
- if you use a different browser or computer
- If you subsequently configure HTTPS access also, the apache settings you set for password protection should automatically be ported over to be used for HTTPS.

# HTTPS Installation

In order to provide HTTPS access to an abYsis server you need to install an SSL certificate from a Trusted Certificate Authority on the Apache web server that hosts the abYsis installation. There are many Trusted Certificate Authorities available and each will provide instructions for how to add a certificate to an Apache server.

Below is an example of how to set up a certificate from a free SSL provider, but of course you should confer with your IT team to find the optimal approach for your organisation.

## Requirements

The following are required when setting up HTTPS

- An open port 443. If you are using an internal server your IT team will know how best to configure this. If you are installing on AWS then refer to the "abYsis - Using on AWS" document.
- A static IP address
- A domain name attached to the IP address

## Let's Encrypt

This example uses Let's Encrypt (https://letsencrypt.org/) which provides free SSL certificates. It uses a program called **certbot** that handles the registration and installing of the SSL certificates as well as renewing the certificates when they expire.

You will need to enter a series of commands without errors. Once you have successfully reached the end your abYsis installation should be protected.

## Adding Let's Encrypt certbot

First, you will be installing some software called **snap** that supports the certbot program which installs the SSL certificate. Log on to your abYsis server (where abYsis has already been installed) using either the account used to install abYsis that has **sudo** permissions.

```
sudo yum install -y snapd
sudo systemctl enable --now snapd.socket
sudo systemctl enable snapd.seeded.service
sudo systemctl start snapd.seeded.service
sudo snap install core
sudo snap refresh core
sudo ln -s /var/lib/snapd/snap /snap
```

It is best practice to make sure no previous installation of certbot exists and then to install the latest version of certbot.

```
sudo snap remove certbot
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

### Activating Let's Encrypt

Then run certbot.

```
sudo certbot --apache -m "<email address> -n --agree-tos --domains "<domain name>"
```

Certbot will first check that it can find the apache conf file responsible for the domain name. Then it will request an SSL certificate from Let's Encrypt and register it both publicly and on your server. You need to provide an email address for the registration and the domain name that your server is using.

Finally, certbot will ensure that the certificate can be renewed when it is close to expiring.

If the certbot command completes without an error then your server should now be able to accept HTTPS requests.

## Note

- You would need to make sure your HTTPS port 443 is open on your abYsis server for people to access your encrypted pages.
- certbot ensures that normal HTTP traffic is automatically redirected to HTTPS.

- It may take a couple of hours for your certificate to be recognised and accepted by browsers
- An SSL certificate generated in this manner usually lasts for 90 days. The certbot program should automatically renew a certificate. To enable this it is advisable to ensure that HTTP port 80 is accessible by everyone. Your abYsis site should  be secure because of the redirect to HTTPS, but should be checked. A system such as Let's Encrypt should be able to confirm.