This post details how I set up this website and its server. This is in part for my reference. For my setup I went with FreeBSD. Because of issues I've had with CMSes in the past, I'm going with a static site based install with no PHP or Mysql. The HTTP server is Apache, SSL is present and configured with HTTP Strict Transport Security, and plain HTTP redirects to HTTPS. I want a setup that is pretty minimalistic and secure. I am running an email server, with spam filtering outsourced to a hosted spam filtering service. I have it installed on a fairly resource limited KVM VPS with about 256mb of ram.

Freebsd install

Because its a KVM VPS, I just loaded an ISO corresponding to the latest edition of FreeBSD. The install should work roughly the same as an install on physical hardware. The FreeBSD install is pretty simple. I do have a few notes and tips here.

- Uncheck "games" on the "Distribution Select" menu. These are not useful in a server install like the one we will be doing, but are checked by default and these take up a fair amount of space (problematic for a resource and storage constrained VPS like mine).
- Begin guided partitioning partitioning mode. Most of the defaults are reasonable here. Select the entire disk option and confirm that you want to erase the whole hard drive.
- Check the allocation the installer created to make sure it is reasonable and then select the commit option.

- Set a password. A secure, long password is particularly important here, because the is a remotely accessible system. FreeBSD will come with SSH preinstalled. A large number of the VPSes that have been hacked that I have seen have been compromised due to a bad password.
- Configure the network as per your need. The exact configuration options will depend on how your IP addresses are allocated, if you have been allocated/want to use a IPv6 address and so on.
- Set time as per your system, and desired time zone.
- Set sshd to start at boot. You need open ssh for remote administration.
- Enable crash dumps because this makes debugging issues easier.
- Set users as per your preference. It is probably a good idea to create at least one non-root user.

Initial Config

Ssh into the regular account created. Use the *su* command to escalate privileges. The su command allows you to switch users to the root user. The more common *sudo* command is not installed on FreeBSD by default, and thus it can not be used at this time. *su* requires no parameters because the root user is the default user to switch to.

```
su
```

Update your ports. This is the very first thing you should do, because

you will encounter errors if you try to install from the old ports that are included with the install by default. Ports are in effect source-based packages. Ports contain scripts to install, compile and update software on your FreeBSD install. The alternative would be to use binary packages through the *pkg* command, but for various reasons such as flexibility, I prefer ports over packages.

The following commands can be used to update the ports.

```
portsnap fetch
portsnap extract
```

The `portsnap fetch` command downloads the most recent set of ports from the FreeBSD server. The `portsnap extract` command uncompresses the ports, and moves them to the `/usr/ports` folder where they can be used by the system.

Install the portmaster tool to assist in port upgrades. It is a tool that can automate common tasks such as upgrading software installed through ports.

The following commands can be used to install portmaster.

```
cd /usr/ports/ports-mgmt/portmaster
make config-recursive
make install clean
```

The two make commands result in the downloading and compiling of the source code of the program.

Upgrade all ports using the portmaster tool as follows.

```
portmaster -a
```

Install OS patches (including security patches) with the following command

```
freebsd-update fetch
freebsd-update install
```

Mosh Shell

Mosh is an alternative remote access system to OpenSSH. It is more fault tolerant of intermittent connections, and has other benefits over raw OpenSSH. I have it installed on the VPSes that I use, because I find it preferable to SSH. Install it as follows.

```
cd /usr/ports/net/mosh
make config-recursive
make install clean
```

Git

Git is a commonly used version control program and is needed for a

number of tasks. It is used in a number of the ports, and is used to download config files later in the tutorial. Install it as follows.

```
cd /usr/ports/devel/git/
make config-recursive
make install clean
```

Lets Encrypt: Install the Lets Encrypt Client

Lets Encrypt is an service that allows you to generate free SSL certificates. There is an utility that is written in Python that can conduct the process of requesting SSL certificates for a domain name that has an A record tied to the IP address of your VPS.

Install this client as follows.

```
cd /usr/ports/security/py-letsencrypt/
make config-recursive
make install clean
```

Produce the Lets Encrypt Certificates

Use the following command to generate certificates.

```
letsencrypt certonly
```

This will open an interactive wizard that will request information corresponding to your domain and certificate. Unlike on certain Linux distributions, the Lets Encrypt tool on FreeBSD can not automatically install the certificates and configure the Apache web server to use SSL.

You can enter multiple domains/subdomains in a single certificate, but this increases the processing time it takes for Lets Encrypt to generate your certificate, and makes the resulting certificate file larger. If you want to have multiple domains or subdomains in your certificate enter them as follows when prompted to enter the domain you are generating certificates for separated by commas, as follows.

```
example.com,blah.example.com,foo.example.com,example2.com
```

The generated certificates are kept in `/usr/local/etc/letsencrypt/live/`

Apache

In this section we will install the Apache web server, and configure SSL properly, including Strict Transport Security and other security measures. Install the Apache web server with the following commands.

```
cd /usr/ports/www/apache24/
make config-recursive
make install clean
```

In order to cause the Apache web server to load when the system boots add the following line to the `/etc/rc.conf` file

```
apache24_enable="YES"
```

This can be done with the text editor of your choice. The *ee* text editor comes preinstalled with FreeBSD and is easy to use. It can be activated as follows

```
ee filename
```

ie.

```
ee /etc/rc.conf
```

Start the server with the following command.

```
service apache24 start
```

Create a folder for your document root. This can be done as follows.
`mkdir /usr/local/www/apache24/data/example.com`

Edit the `/usr/local/etc/apache24/httpd.conf` file.

Add the following to your configuration file. These are the web server modules that you will need to properly use SSL and other relevant features.

```
            LoadModule authn_socache_module
libexec/apache24/mod_authn_socache.so
            LoadModule socache_shmcb_module
libexec/apache24/mod_socache_shmcb.so
            LoadModule unixd_module libexec/apache24/mod_unixd.so
            LoadModule mpm_event_module
libexec/apache24/mod_mpm_event.so
            LoadModule ssl_module libexec/apache24/mod_ssl.so
            LoadModule authn_core_module
libexec/apache24/mod_authn_core.so
            LoadModule authz_core_module
libexec/apache24/mod_authz_core.so
            LoadModule log_config_module
libexec/apache24/mod_log_config.so
            LoadModule headers_module
libexec/apache24/mod_headers.so
            LoadModule unique_id_module
libexec/apache24/mod_unique_id.so
            LoadModule rewrite_module
libexec/apache24/mod_rewrite.so
            LoadModule autoindex_module
libexec/apache24/mod_autoindex.so
            LoadModule mime_magic_module
libexec/apache24/mod_mime_magic.so
            LoadModule dir_module libexec/apache24/mod_dir.so
            LoadModule setenvif_module
libexec/apache24/mod_setenvif.so
            LoadModule alias_module libexec/apache24/mod_alias.so
            LoadModule authz_host_module
libexec/apache24/mod_authz_host.so
            LoadModule mime_module libexec/apache24/mod_mime.so
```

```
            ServerName example.com:80
            Include etc/apache24/extra/httpd-ssl.conf
            DirectoryIndex index.html
```

Comment out all other LoadModule lines, so, as to disable other unneeded modules.

Edit the `/usr/local/etc/apache24/extra/httpd-ssl.conf` file in order to set the location of the SSL certificate and key.

Add the following to the file

```
        SSLProtocol -ALL +TLSv1.2
        SSLHonorCipherOrder On
        SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM
EECDH+ECDSA+SHA384 EECDH EDH+aRSA !aNULL !eNULL !LOW !3DES !RC4
!MD5 !EXP !PSK !SRP !DSS"
        Header always set X-Frame-Options DENY
        Header always set X-Content-Type-Options nosniff
        SSLCompression off
        SSLSessionTickets Off
        SSLUseStapling on
        SSLStaplingCache "shmcb:logs/stapling-cache(150000)"
        Header always set Strict-Transport-Security "max-
age=63072000; includeSubdomains; preload"
```

The above code does a few things. The `SSLProtocol -ALL +TLSv1.2` limits SSL/TLS protocols other than TLSv1.2, which is preferred for

security reasons. However, this does break compatibility with older web browsers, so, remove this if compatibility is a strong desire.

The `SSLHonorCipherOrder` line and the `SSLCipherSuite` line limit the types of encryption that can be used. This has the benefit of forcing the use of ciphers with stronger security attributes, and eliminating the possibility of downgrade attacks where in an attacker tries to force a client to use a lesser cipher.

The `SSLUseStapling` line and the `SSLStaplingCache` line enable SSL Stapling. This causes the server to send confirmation that the certificate in use has not been revoked. This has a number of security benefits, as it makes certificate revocation more efficient, and reduces the cost from the client side of checking for revocation.

The last line sets HTTP Strict Transport Security, which informs the browsers visiting the site that HTTPS is always the preferred way to connect with the server, and that plain HTTP should be refused. This is difficult to reverse, as even when this header is no longer being set clients will still insist on connecting over HTTPS. So, only keep this line if you are sure that you can always support HTTPS on the domains served by this server.

Next we will set up virtual domains. For each domain the server will serve add the following (appropriately edited based on the exact location of your folders and similar) in the

`/usr/local/etc/apache24/httpd.conf` file

```
      <VirtualHost *:80>
       ServerName example.com
       DocumentRoot /usr/local/www/apache24/data/example.com
       <IfModule mod_rewrite.c>
             RewriteEngine On
            RewriteCond %{HTTPS} off
            RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
       </IfModule>
      </VirtualHost>

        <IfModule mod_ssl.c>
        <VirtualHost *:443>
                ServerName theodorejones.info
                DocumentRoot
/usr/local/www/apache24/data/theodorejones.info
                SSLEngine on
                SSLCertificateFile
"/usr/local/etc/letsencrypt/live/theodorejones.info/fullchain.pem"

                SSLCertificateKeyFile
"/usr/local/etc/letsencrypt/live/theodorejones.info/privkey.pem"
        <Directory
/usr/local/www/apache24/data/theodorejones.info>
        Options All
        AllowOverride All
        </Directory>
        </VirtualHost>
        </IfModule>
```

The data the the server is serving for example.com is in `/usr/local/www/apache24/data/example.com` as a result of the above.

Reload the server with the following command in order to make the configuration changes take effect.

```
service apache24 reload
```

Server hardening

In this section we will make some changes to the Apache configuration to improve security. See the following article for some of the motivation behind this http://geekflare.com/apache-web-server-hardening-security/

Edit the `"/usr/local/etc/apache24/httpd.conf"` file. Add the following lines to it.

```
        #Hardening
        ServerTokens Prod
        ServerSignature Off
        FileETag None
        TraceEnable off
        Header edit Set-Cookie ^(.*)$ $1;HttpOnly;Secure
        Header always append X-Frame-Options SAMEORIGIN
        RewriteEngine On
        RewriteCond %{THE_REQUEST} !HTTP/1.1$
        RewriteRule .* - [F]
        Timeout 60
```

Opensmtpd

In this section we will configure the Opensmtpd email server. Opensmtpd is a SMTP server that is easy to configure. It is configured to listen on port 13456 because I use the MxGuardDog spam filtering service which forwards mail to that port. Change this to the desired port as accords with your needs.

Install opensmtpd as follows.

```
cd /usr/ports/mail/opensmtpd/ && make install clean
```

Edit the `/usr/local/etc/mail/smtpd.conf` file and add the following lines.

```
        listen on 198.251.80.156 port 13456

        table aliases file:/etc/mail/aliases
        accept from any for domain example.com deliver to maildir
        accept from local for any relay
```

The first line tells the mail server to listen on the IP address *198.251.80.156* on *13456*. Adjust according to your IP address and desired port.

Next set the SMTP server to run when the system boots.

Edit the `/etc/rc.conf` file.

Add the following lines

```
smtpd_enable="YES"
```

Start the service with the following command.

```
service start smtpd
```

Courrier IMAP

Courrier IMAP is a simple IMAP server. It allows users to retrieve their email through the IMAP protocol.

Install it with the following commands.

```
cd /usr/ports/mail/courier-imap;
make config
make install
```

Edit the `/etc/rc.conf` file and add the following in order to cause the IMAP server to load when the system boots.

```
courier_imap_imapd_enable="YES"
courier_authdaemond_enable="YES"
```

Start the IMAP server with the following two commands

```
/usr/local/etc/rc.d/courier-imap-imapd start
/usr/local/etc/rc.d/courier-authdaemond start
```

Daily Maintenance

Run these commands on a regular basis to keep the system updated. These commands will update the OS and the ports.

```
freebsd-update fetch
freebsd-update install
portsnap fetch update
portmaster -a
```