# HunnyPotR
# Design Details

"*Don't let the Muggles get you down.*"

*Harry Potter and the Prisoner of Azkaban*

# HunnyPotR Install

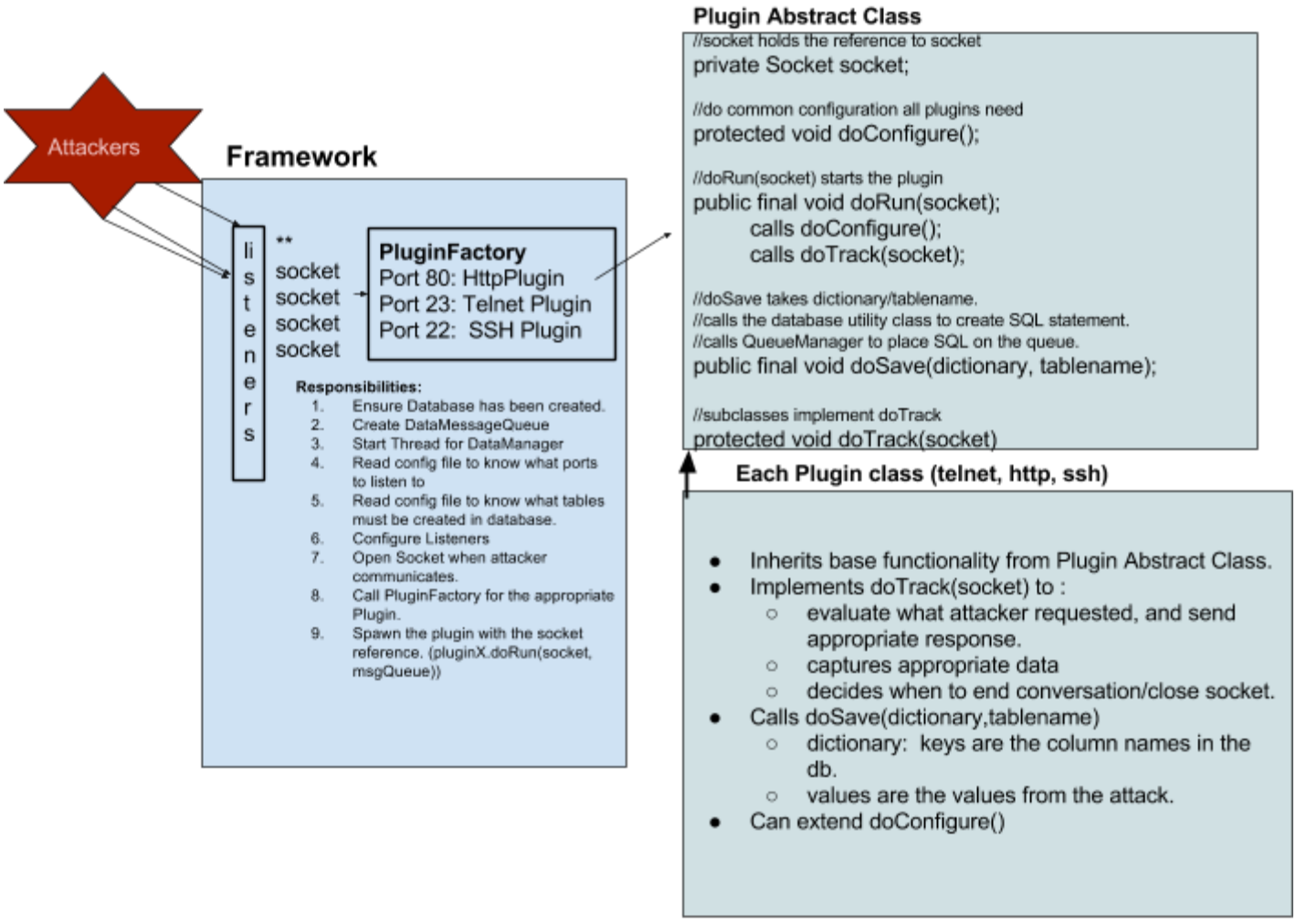
## Configuration

- When the honeypot is installed, the configuration file will be written to /etc/hunnypotr.
- When the administrator is installing the honeypot, if they choose to customize the configuration during the install phase, the configuration will be stored in the users home directory ~/hunnypotr.

# HunnyPotR Framework

## Handling Permissions

1. Multithread with authbind (linux)
2. Lower permissions on the spawn of the plugin




**Initially, just using Stream Sockets, but would like to use Raw Sockets to gain more information. Base Plugin would evaluate the packets in doRun();

# HunnyPotR Plugins

## Base Plugin Abstract Class

```
//socket holds the reference to socket
private Socket socket;

//do the common plugin configuration
protected void doConfigure();

//doRun(socket,msgQueue) starts the plugin
public final void doRun(socket){
        this.socket = socket;
        doConfigure();
        doTrack(socket);
}

//doSave takes dictionary/tablename.
//calls the database utility class to create SQL statement.
//calls QueueManager to place SQL on the queue.
public final void doSave(dictionary, tablename){
        MsgQueue queue = MsgQueue.getInstance();


//subclasses implement doTrack
protected  abstract void doTrack(socket)
```

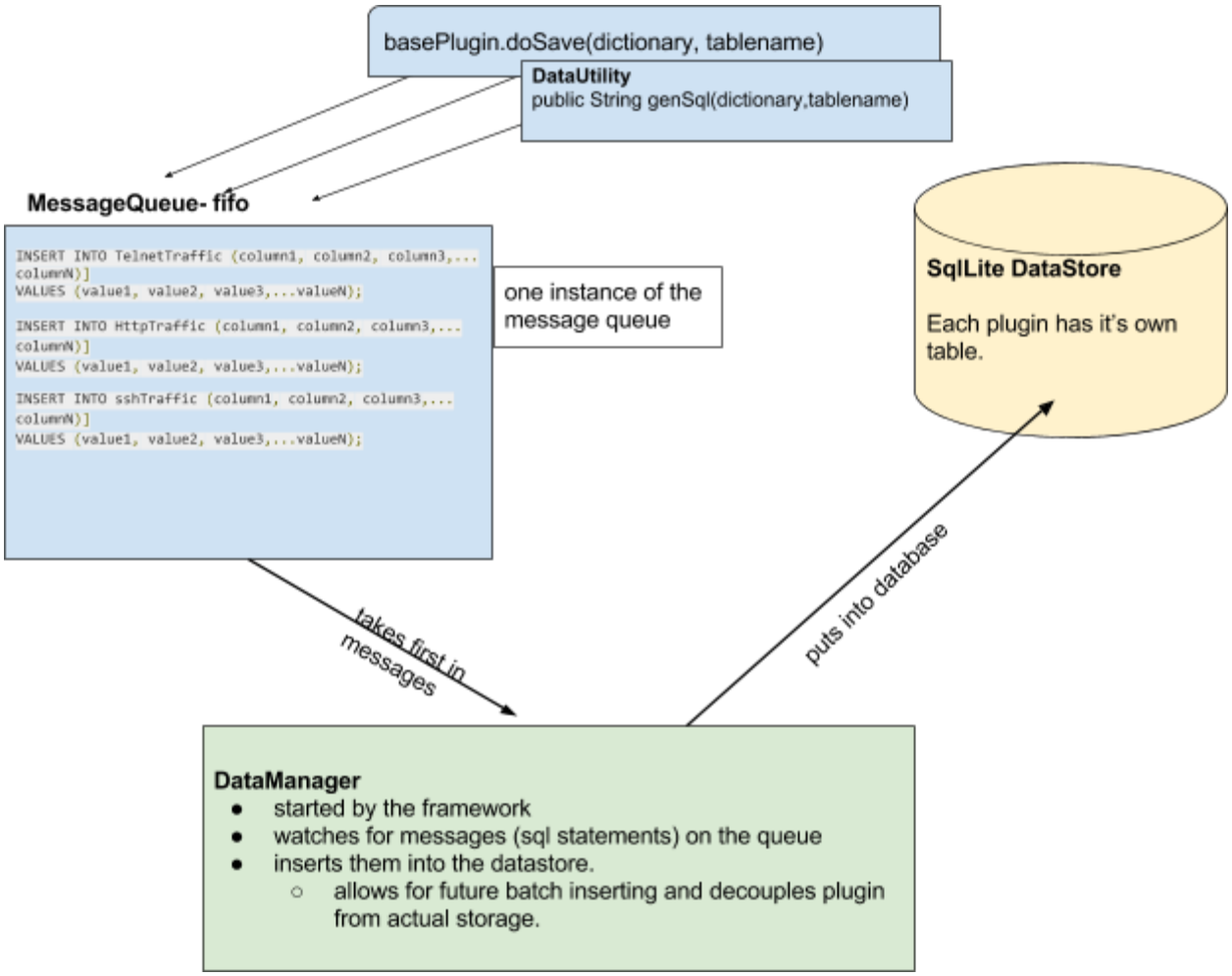
doRun() and doSave() are both “final” methods that are implemented by the base class. Subclasses should not touch these implementations. This will allow the product to eliminate coupling of the customer-written plugins from the framework and data storage mechanisms.

doTrack() will be an abstract method that the subclasses will be responsible for implementing. They will call doSave() from this method after they analyze the request, track the data being sent, and create the dictionary of key/value pairs

doConfigure() is called by doRun(), but can be extended by the subclasses to configure things out of the ordinary.

## Data Capture

- Simple Message Queue to handle the database sql statements.
- Base Plugin writes to queue when doSave is called. Actual plugins just call doSave(dictionary, tablename)
- Framework starts the DataManager thread who consumes the queue and inserts into database.  Multiple producers, one consumer.



## Logging

- Framework and each plugin will have it’s own log file.
- Name of the log file will be configurable for each plugin.
- Common log mechanism across the application to all store in a single directory.

## Deploying HunnyPotR

### Docker

# Reporting

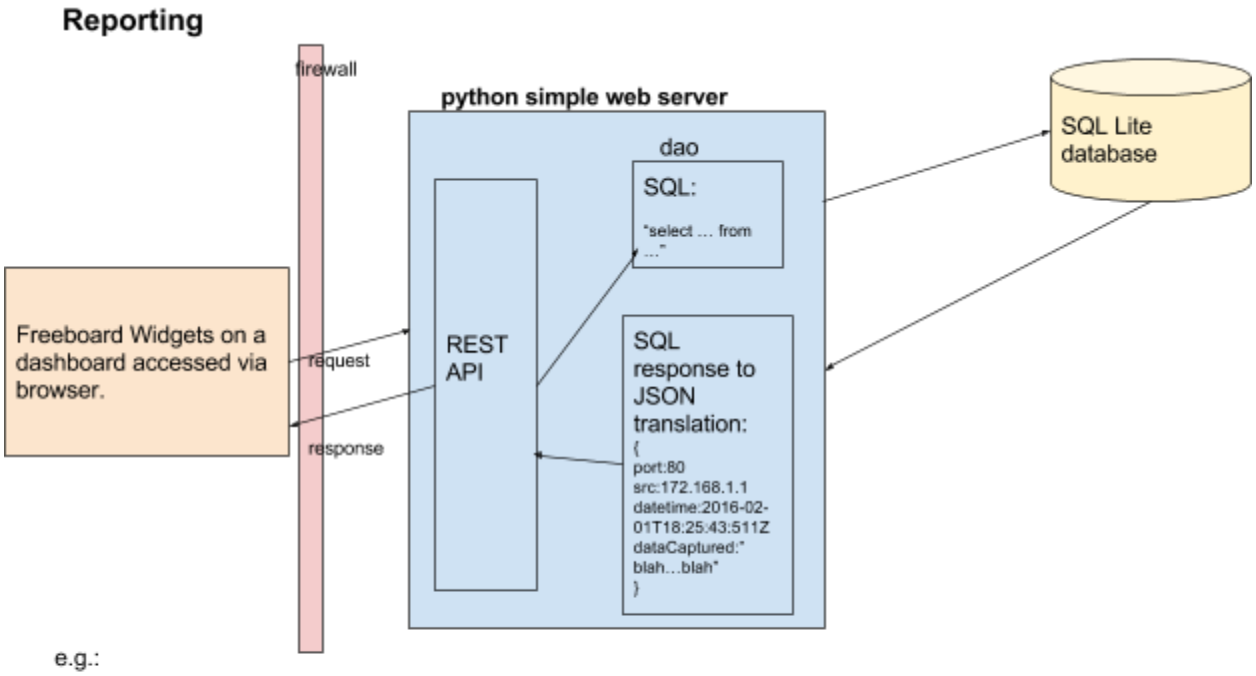
## Install of WebServer

## API to Restful Services

- Will work with customer to define details of the Rest API.

## Freeboard Dashboard

- calls the Rest API to populate widget.




e.g.:
http://hunnypotr/v1/analytics/ports/80/hours/24
http://hunnypotr/v1/analytics/ports/23/days/3
http://hunnypotr/v1/analytics/ports/22/weeks/1

## Items to Consider

1. When to enhance data by taking IP addresses and determining said location of the attack.
2. Handling the raw socket instead of just a streaming socket.  What extra data will this provide?  Initially will do simplest implementation, but designed with this in mind.