

---

# ScoringEngine Documentation

*Release latest*

**pwnbus**

**Jan 12, 2019**



---

## Contents

---

<b>1</b>	<b>Overview</b>	<b>1</b>
1.1	Why? . . . . .	1
1.2	How does it work? . . . . .	1
1.3	Screenshots . . . . .	3
<b>2</b>	<b>Installation</b>	<b>7</b>
2.1	Docker . . . . .	7
2.2	Manual . . . . .	8
<b>3</b>	<b>Configuration</b>	<b>15</b>
3.1	Location to config file . . . . .	15
3.2	Configuration Keys . . . . .	15
<b>4</b>	<b>Implemented Checks</b>	<b>17</b>
4.1	DNS . . . . .	17
4.2	Elasticsearch . . . . .	17
4.3	FTP . . . . .	17
4.4	HTTP(S) . . . . .	18
4.5	ICMP . . . . .	18
4.6	IMAP(S) . . . . .	18
4.7	LDAP . . . . .	18
4.8	MSSQL . . . . .	18
4.9	MySQL . . . . .	19
4.10	POP3(S) . . . . .	19
4.11	PostgreSQL . . . . .	19
4.12	RDP . . . . .	19
4.13	SMB . . . . .	19
4.14	SMTP(S) . . . . .	20
4.15	SSH . . . . .	20
4.16	VNC . . . . .	20
<b>5</b>	<b>Development</b>	<b>21</b>
5.1	Initial Setup . . . . .	21
5.2	Run Services . . . . .	22
5.3	Run Tests . . . . .	22
5.4	Modifying Documentation . . . . .	23



### 1.1 Why?

The goal of the ScoringEngine is to keep track of service up time in a blue teams/red team competition.

### 1.2 How does it work?

The general idea of the ScoringEngine is broken up into 3 separate processes, Engine, Worker, and Web.

#### 1.2.1 Engine

The engine is responsible for tasking *Checks* that are used to verify network services each round, and determining/saving their results to the database. This process runs for the entire competition, and will sleep for a certain amount of time before starting on to the next round.

#### 1.2.2 Worker

The worker connects to Redis and waits for *Checks* to get tasked in order to run them against . Once it receives a *Check*, it executes the command and sends the output back to the Engine.

#### 1.2.3 Web

The web application provides a graphical view of the Competition. This includes things like a bar graph of all team's scores as well as a table of the current round's results. This can also be used to configure the properties of each service per team.

## 1.2.4 External Resources

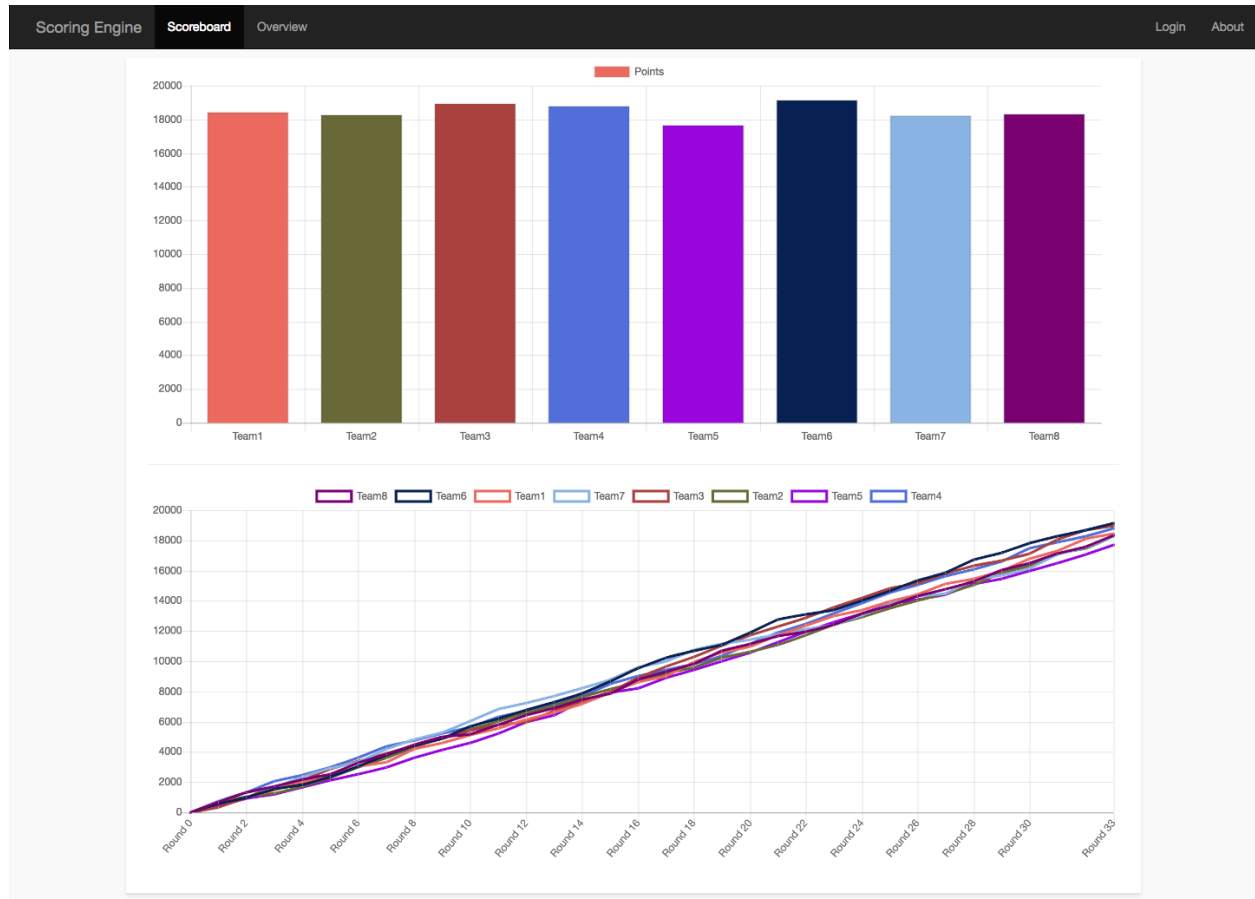
We currently use [MySQL](#) as the database, and [Redis](#) as the data store for tasks while they are getting scheduled.

## 1.2.5 Putting it all together

- The *Engine* starts
- The first *Round* starts
- The *Engine* tasks *Checks* out to the *Workers*
- The *Workers* execute the *Checks* and return the output to the *Engine*
- The *Engine* waits for all *Checks* to finish
- The *Engine* determines the results of each *Check*, and saves the results to the DB
- The *Engine* ends the *Round*
- The *Engine* sleeps for some time
- The second *Round* starts
- ...

## 1.3 Screenshots

### 1.3.1 Scoreboard



## 1.3.2 Overview

Scoring Engine	Scoreboard	Overview	Login	About
----------------	------------	----------	-------	-------

### Round 33

2017-12-28 12:38:43

Team Name	Current Score	SSH	HTTP	HTTPS	MySQL	FTPDownload	FTPUplod	DNS	Postgresql	POP3	IMAP	SMTP	VNC
Team6	19150	10.2.6.2:22	10.2.6.3:80	10.2.6.3:443	10.2.6.4:3306	10.2.6.5:21	10.2.6.5:21	10.2.6.6:53	10.2.6.7:5432	10.2.6.8:110	10.2.6.8:143	10.2.6.8:25	10.2.6.1:5900
Team3	18950	10.2.3.2:22	10.2.3.3:80	10.2.3.3:443	10.2.3.4:3306	10.2.3.5:21	10.2.3.5:21	10.2.3.6:53	10.2.3.7:5432	10.2.3.8:110	10.2.3.8:143	10.2.3.8:25	10.2.3.1:5900
Team4	18800	10.2.4.2:22	10.2.4.3:80	10.2.4.3:443	10.2.4.4:3306	10.2.4.5:21	10.2.4.5:21	10.2.4.6:53	10.2.4.7:5432	10.2.4.8:110	10.2.4.8:143	10.2.4.8:25	10.2.4.1:5900
Team1	18450	10.2.1.2:22	10.2.1.3:80	10.2.1.3:443	10.2.1.4:3306	10.2.1.5:21	10.2.1.5:21	10.2.1.6:53	10.2.1.7:5432	10.2.1.8:110	10.2.1.8:143	10.2.1.8:25	10.2.1.1:5900
Team8	18350	10.2.8.2:22	10.2.8.3:80	10.2.8.3:443	10.2.8.4:3306	10.2.8.5:21	10.2.8.5:21	10.2.8.6:53	10.2.8.7:5432	10.2.8.8:110	10.2.8.8:143	10.2.8.8:25	10.2.8.1:5900
Team2	18300	10.2.2.2:22	10.2.2.3:80	10.2.2.3:443	10.2.2.4:3306	10.2.2.5:21	10.2.2.5:21	10.2.2.6:53	10.2.2.7:5432	10.2.2.8:110	10.2.2.8:143	10.2.2.8:25	10.2.2.1:5900
Team7	18250	10.2.7.2:22	10.2.7.3:80	10.2.7.3:443	10.2.7.4:3306	10.2.7.5:21	10.2.7.5:21	10.2.7.6:53	10.2.7.7:5432	10.2.7.8:110	10.2.7.8:143	10.2.7.8:25	10.2.7.1:5900
Team5	17700	10.2.5.2:22	10.2.5.3:80	10.2.5.3:443	10.2.5.4:3306	10.2.5.5:21	10.2.5.5:21	10.2.5.6:53	10.2.5.7:5432	10.2.5.8:110	10.2.5.8:143	10.2.5.8:25	10.2.5.1:5900

Want a json formatted version of this data (including ip addresses)? [Here](#)

## 1.3.3 Team Services

Scoring Engine	Scoreboard	Overview	Services	team1user1	About
----------------	------------	----------	----------	------------	-------

### Team1

Place: 4

Score: 24100 points

Service	Host	Port	Status	Score Earned	Max Score	% Earned	Trending
SSH	10.2.1.2	22	UP	1900	4300	44	XXXXXX✓XX✓
HTTP	10.2.1.3	80	DOWN	2600	4300	60	XX✓XX✓XX✓XXX
HTTPS	10.2.1.3	443	UP	2200	4300	51	XX✓XX✓XX✓XX✓
MySQL	10.2.1.4	3306	UP	2000	4300	46	XX✓XX✓XX✓XX✓
FTPDownload	10.2.1.5	21	UP	1100	2150	51	XX✓XX✓XX✓XX✓
FTPUplod	10.2.1.5	21	UP	1200	2150	55	XX✓XX✓XX✓XX✓
DNS	10.2.1.6	53	DOWN	2000	4300	46	XX✓XX✓XX✓XX✓
Postgresql	10.2.1.7	5432	DOWN	2000	4300	46	XX✓XX✓XX✓XX✓
POP3	10.2.1.8	110	DOWN	2100	4300	48	XX✓XX✓XX✓XX✓
IMAP	10.2.1.8	143	DOWN	2200	4300	51	XX✓XX✓XX✓XX✓
SMTP	10.2.1.8	25	UP	2800	4300	65	XX✓XX✓XX✓XX✓
VNC	10.2.1.1	5900	UP	2000	4300	46	XX✓XX✓XX✓XX✓



### 1.3.4 Specific Service

Scoring Engine
Scoreboard
Overview
Services

team1user1
About

## SSH

Host: 10.2.1.2

Port: 22

Accounts

Username	Password
ttesterson	testpass
rpeterson	*****

Checks

Show 10 entries

Search:

Round	Result	Reason	Timestamp
43	UP	Successful Content Match	2017-12-28 12:49:41
42	DOWN	Job Timed Out	2017-12-28 12:49:41
41	DOWN	Unsuccessful Content Match	2017-12-28 12:49:41
40	UP	Successful Content Match	2017-12-28 12:49:41
39	DOWN	Unsuccessful Content Match	2017-12-28 12:49:41
38	UP	Successful Content Match	2017-12-28 12:49:41
37	UP	Successful Content Match	2017-12-28 12:49:41
36	DOWN	Unsuccessful Content Match	2017-12-28 12:49:41
35	DOWN	Unsuccessful Content Match	2017-12-28 12:49:41
34	DOWN	Unsuccessful Content Match	2017-12-28 12:49:41

Showing 1 to 10 of 43 entries

Previous 1 2 3 4 5 Next

### 1.3.5 Round Status

Scoring Engine
Scoreboard
Overview
Admin

whiteteamuser
About

Round Stats
Round Progress
Users
Settings
Teams
Team1
Team2
Team3
Team4
Team5
Team6
Team7
Team8

#### Current Round Status

Round Progress 55%

Team1 42%

Team2 57%

Team3 42%

Team4 42%

Team5 42%

Team6 57%

Team7 85%

Team8 71%

## 1.3.6 Admin Team View

Scoring Engine

Scoreboard

Overview

Admin

whiteteamuser - About

Round Stats

Round Progress

Users

Settings

Teams

Team1

Team2

Team3

Team4

Team5

Team6

### Team2

ICMP

SSH

Host: testbed\_ssh

Port: 22

Accounts

Username	Password
ttesterson	*****
rpeterston	*****

Environments

Matching Regex	Properties				
ttesterson	<table> <tr> <th>Name</th> <th>Value</th> </tr> <tr> <td>command</td> <td>ls -l /home</td> </tr> </table>	Name	Value	command	ls -l /home
Name	Value				
command	ls -l /home				
PID	<table> <tr> <th>Name</th> <th>Value</th> </tr> <tr> <td>command</td> <td>ps</td> </tr> </table>	Name	Value	command	ps
Name	Value				
command	ps				

Checks

	Round	Result	Reason	Command	Timestamp
🟢	2	Pass	Successful Content Match	sshpass -p 'otherpass' ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no 'rpeterston@testbed_ssh' -p 22 'ls -l /home'	2017-12-28 16:14:37
🟢	1	Pass	Successful Content Match	sshpass -p 'testpass' ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no 'ttesterson@testbed_ssh' -p 22 'ls -l /home'	2017-12-28 16:13:16

HTTP

HTTPS

MySQL

FTPDownload

FTPUplod

### 2.1 Docker

---

**Note:** It takes a minute or 2 for all of the containers to start up and get going!

---

#### 2.1.1 TestBed Environment

```
make rebuild-testbed-new
```

This command will build, stop any pre-existing scoring engine containers, and start a new environment. As part of the environment, multiple containers will be used as part of the testbed environment.

#### 2.1.2 Environment Variables

We use certain environment variables to control the functionality of certain docker containers.

**SCORINGENGINE\_OVERWRITE\_DB** If set to true, the database will be deleted and then recreated during startup.

**SCORINGENGINE\_EXAMPLE** If set to true, the database is populated with sample db, and the engine and worker containers will be paused. This is useful for doing development on the web app.

You can set each environment variable before each command executed, for example:

```
SCORINGENGINE_EXAMPLE=true make rebuild-new
```

### 2.1.3 Production Environment

Modify the bin/competition.yaml file to configure the engine according to your competition environment. Then, run the following make command to build, and run the scoring engine.

**Warning:** This will delete the previous database, exclude the ‘new’ part from the command to not rebuild the db.

```
make rebuild-new
```

Then, to ‘pause’ the scoring engine (Ex: At the end of the day):

```
docker-compose -f docker-compose.yml stop engine
```

To ‘unpause’ the engine:

```
docker-compose -f docker-compose.yml start engine
```

## 2.2 Manual

### 2.2.1 Base Setup

---

**Note:** Currently, the only OS we have documentation on is Ubuntu 16.04.

---

#### Install dependencies via apt-get

```
apt-get update
apt-get install -y python3.5 wget git python3.5-dev build-essential libmysqlclient-dev
```

#### Create engine user

```
useradd -m engine
```

#### Download and Install pip

```
wget -O /root/get-pip.py https://bootstrap.pypa.io/get-pip.py
python3.5 /root/get-pip.py
rm /root/get-pip.py
```

#### Setup virtualenvironment

```
pip install virtualenv
su engine
cd ~/
mkdir /home/engine/scoring_engine
virtualenv -p /usr/bin/python3.5 /home/engine/scoring_engine/env
```

### Setup src directory

```
git clone https://github.com/scoringengine/scoringengine /home/engine/scoring_engine/
↪src
```

### Install scoring\_engine src python dependencies

```
source /home/engine/scoring_engine/env/bin/activate
pip install -e /home/engine/scoring_engine/src/
```

### Copy/Modify configuration

```
cp /home/engine/scoring_engine/src/engine.conf.inc /home/engine/scoring_engine/src/
↪engine.conf
vi /home/engine/scoring_engine/src/engine.conf
```

### Create log file locations (run as root)

```
mkdir /var/log/scoring_engine
chown -R syslog:adm /var/log/scoring_engine
```

### Copy rsyslog configuration

```
cp /home/engine/scoring_engine/src/configs/rsyslog.conf /etc/rsyslog.d/10-scoring_
↪engine.conf
```

### Restart rsyslog

```
systemctl restart rsyslog
```

## 2.2.2 Web

### Install MySQL Server

```
apt-get install -y mysql-server
sed -i -e 's/127.0.0.1/0.0.0.0/g' /etc/mysql/mysql.conf.d/mysqld.cnf
systemctl restart mysql
```

## Setup MySQL

```
mysql -u root -p<insert password set during installation>
CREATE DATABASE scoring_engine;
CREATE USER 'engineuser'@'%' IDENTIFIED BY 'enginepass';
GRANT ALL on scoring_engine.* to 'engineuser'@'%' IDENTIFIED by 'enginepass';
```

## Install Nginx

```
apt-get install -y nginx
```

## Setup SSL in Nginx

```
mkdir /etc/nginx/ssl
cd /etc/nginx/ssl
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt
```

## Copy nginx config

```
cp /home/engine/scoring_engine/src/configs/nginx.conf /etc/nginx/sites-available/
↪scoring_engine.conf
ln -s /etc/nginx/sites-available/scoring_engine.conf /etc/nginx/sites-enabled/
rm /etc/nginx/sites-enabled/default
systemctl restart nginx
```

## Setup web service

```
cp /home/engine/scoring_engine/src/configs/web.service /etc/systemd/system/scoring_
↪engine-web.service
```

## Modify configuration

```
vi /home/engine/scoring_engine/src/engine.conf
```

## Install uwsgi

```
pip install uwsgi
```

## Start web

```
systemctl enable scoring_engine-web
systemctl start scoring_engine-web
```

## Monitoring

```
journalctl -f _SYSTEMD_UNIT=scoring_engine-web.service
tail -f /var/log/scoring_engine/web.log
tail -f /var/log/scoring_engine/web-nginx.access.log
tail -f /var/log/scoring_engine/web-nginx.error.log
```

## 2.2.3 Engine

### Install Redis

```
apt-get install -y redis-server
```

### Setup Redis to listen on external interface

```
sed -i -e 's/bind 127.0.0.1/bind 0.0.0.0/g' /etc/redis/redis.conf
systemctl restart redis
```

### Setup Engine service (run as root)

```
cp /home/engine/scoring_engine/src/configs/engine.service /etc/systemd/system/scoring_
engine-engine.service
```

### Modify configuration

```
su engine
vi /home/engine/scoring_engine/src/engine.conf
```

### Setup scoring engine teams and services

```
su engine
vi /home/engine/scoring_engine/src/bin/competition.yaml
source /home/engine/scoring_engine/env/bin/activate
/home/engine/scoring_engine/src/bin/setup
```

### Start engine service (must run as root)

```
systemctl start scoring_engine-engine
```

### Monitor engine

```
journalctl -f _SYSTEMD_UNIT=scoring_engine-engine.service
tail -f /var/log/scoring_engine/engine.log
```

## 2.2.4 Worker

### Modify hostname

```
hostname <INSERT CUSTOM HOSTNAME HERE>
```

### Setup worker service (run as root)

```
cp /home/engine/scoring_engine/src/configs/worker.service /etc/systemd/system/scoring_
↪engine-worker.service
```

### Modify configuration

Change REDIS host/port/password fields to main engine host::

```
vi /home/engine/scoring_engine/src/engine.conf
```

Modify worker to customize number of processes. Append ‘--concurrency <num of processes>’ to the celery command line. If not specified, it defaults to # of CPUs.

```
vi /home/engine/scoring_engine/src/bin/worker
```

### Start worker service (must run as root)

```
systemctl enable scoring_engine-worker
systemctl start scoring_engine-worker
```

### Monitor worker

```
journalctl -f _SYSTEMD_UNIT=scoring_engine-worker.service
tail -f /var/log/scoring_engine/worker.log
```

### Install dependencies for DNS check

```
apt-get install -y dnsutils
```

### Install dependencies for HTTP/HTTPS check

```
apt-get install -y curl
```

### Install dependencies for most of the checks

```
apt-get install -y medusa
```



### Install dependencies for SSH check

```
pip install paramiko
```

### Install dependencies for LDAP check

```
apt-get install -y ldap-utils
```

### Install dependencies for Postgresql check

```
apt-get install -y postgresql-client
```

### Install dependencies for Elasticsearch check

```
pip install requests
```

### Install dependencies for SMB check

```
pip install pysmb
```

### Install dependencies for RDP check

```
apt-get install -y freerdp-x11
```

### Install dependencies for MSSQL check

```
apt-get install -y apt-transport-https
curl -s https://packages.microsoft.com/keys/microsoft.asc | apt-key add -
curl -s https://packages.microsoft.com/config/ubuntu/16.04/prod.list | tee /etc/apt/
↪sources.list.d/msprod.list
apt-get update
ACCEPT_EULA=Y apt-get install -y locales mssql-tools unixodbc-dev
echo "en_US.UTF-8 UTF-8" >> /etc/locale.gen
locale-gen
```

### Install dependencies for SMTP/SMTPS check

```
cp /home/engine/scoring_engine/src/scoring_engine/checks/bin/smtp_check /usr/bin/smtp_
↪check
cp /home/engine/scoring_engine/src/scoring_engine/checks/bin/smtps_check /usr/bin/
↪smtps_check
chmod a+x /usr/bin/smtp_check
chmod a+x /usr/bin/smtps_check
```



## 3.1 Location to config file

### 3.1.1 Docker

---

**Note:** This file needs to be edited before running the make commands.

---

```
<path to source root>/docker/engine.conf.inc
```

### 3.1.2 Manual

---

**Note:** Need to restart each scoring engine service once the config is modified.

---

```
/home/engine/scoring_engine/src/engine.conf
```

## 3.2 Configuration Keys

---

**Note:** Each of these config keys can be expressed via environment variables (and take precedence over the values defined in the file). IE: To define round\_time\_sleep, I'd set SCORINGENGINE\_ROUND\_TIME\_SLEEP=3.

---

Key Name	Description
checks_location	Local path to directory of checks
round_time_sleep	Amount of time (seconds) the engine sleeps between rounds
worker_refresh_time	Amount of time (seconds) the engine will sleep for in-between polls of worker status
worker_num_concurrent_tasks	The number of concurrent tasks the worker will run. Set to -1 to default to number of processors.
worker_queue	The queue name for a worker to pull tasks from. This can be used to control which workers get which service checks. Default is 'main'
timezone	Local timezone of the competition
debug	Determines whether or not the engine should be run in debug mode (useful for development)
db_uri	Database connection URI
cache_type	The type of storage for the cache. Set to null to disable caching
redis_host	The hostname/ip of the redis server
redis_port	The port of the redis server
redis_password	The password used to connect to redis (if no password, leave empty)

---

## Implemented Checks

---

### 4.1 DNS

Queries a DNS server for a specific record

Custom Properties:

qtype	type of record (A, AAAA, CNAME, etc)
domain	domain/host to query for

### 4.2 Elasticsearch

Uses python requests to insert message and then query for same message

Custom Properties:

index	index to use to insert the message
doc_type	type of the document

### 4.3 FTP

Uses python ftplib to login to an FTP server, upload a file, login again to FTP and download file

*Uses Accounts*

Custom Properties:

remotefilepath	absolute path of file on remote server to upload/download
filecontents	contents of the file that we upload/download

## 4.4 HTTP(S)

Sends a GET request to an HTTP(S) server

Custom Properties:

useragent	specific useragent to use in the request
vhost	vhost used in the request
uri	uri of the request

## 4.5 ICMP

Sends an ICMP Echo Request to server

Custom Properties: *none*

## 4.6 IMAP(S)

Uses medusa to login to an imap server

*Uses Accounts*

Custom Properties:

domain	domain of the username
--------	------------------------

## 4.7 LDAP

Uses ldapsearch to login to ldap server. Once authenticated, it performs a lookup of all users in the same domain

*Uses Accounts*

Custom Properties:

domain	domain of the username
base_dn	base dn value of the domain (Ex: dc=example,dc=com)

## 4.8 MSSQL

Logs into a MSSQL server, uses a database, and executes a specific SQL command

*Uses Accounts*

Custom Properties:

database	database to use before running command
command	SQL command that will execute

## 4.9 MySQL

Logs into a MySQL server, uses a database, and executes a specific SQL command

*Uses Accounts*

Custom Properties:

database	database to use before running command
command	SQL command that will execute

## 4.10 POP3(S)

Uses medusa to login to an pop3 server

*Uses Accounts*

Custom Properties:

domain	domain of the username
--------	------------------------

## 4.11 PostgreSQL

Logs into a postgresql server, selects a database, and executes a SQL command

*Uses Accounts*

Custom Properties:

database	database to use before running command
command	SQL command that will execute

## 4.12 RDP

Logs into a system using RDP with an account/password

*Uses Accounts*

Custom Properties: *none*

## 4.13 SMB

Logs into a system using SMB with an account/password, and hashes the contents of a specific file on a specific share

*Uses Accounts*

Custom Properties:

share	name of the share to connect to
file	local path of the file to access
hash	SHA256 hash of the contents of the file

## 4.14 SMTP(S)

Logs into an SMTP server and sends an email

*Uses Accounts*

Custom Properties:

touser	address that the email will be sent to
subject	subject of the email
body	body of the email

## 4.15 SSH

Logs into a system using SSH with an account/password, and executes command(s)

---

**Note:** Each command will be executed independently of each other in a separate ssh connection.

---

*Uses Accounts*

Custom Properties:

commands	‘;’ delimited list of commands to run (Ex: id;ps)
----------	---

## 4.16 VNC

Connects and if specified, will login to a VNC server

*Uses Accounts (optional)*

Custom Properties: *none*



# CHAPTER 5

---

## Development

---

---

**Note:** Currently we support 2 ways of working on the Scoring Engine. You can either use the existing [Docker environment](#), or you can run each service locally using python 3. If you choose to do your development locally, we recommend using [virtual environments](#).

---

## 5.1 Initial Setup

These steps are for if you want to do your development locally and run each service locally as well.

### 5.1.1 Create Config File

```
cp engine.conf.inc engine.conf
sed -i '' 's/debug = False/debug = True/g' engine.conf
```

---

**Hint:** If debug is set to True, the web ui will automatically reload on changes to local file modifications, which can help speed up development.

---

### 5.1.2 Install Required Dependencies

```
pip install -e .
```

### 5.1.3 Populate Sample DB

```
python bin/setup --example --overwrite-db
```

## 5.2 Run Services

### 5.2.1 Web UI

```
python bin/web
```

Then, access `localhost:5000`

Table 1: Credentials

Username	Password
whiteteamuser	testpass
redteamuser	testpass
team1user1	testpass
team2user1	testpass
team2user2	testpass

---

**Note:** The engine and worker do NOT need to be running in order to run the web UI.

---

### 5.2.2 Engine

Both the engine and worker services require a redis server to be running. Redis can be easily setup by using the existing docker environment.

```
python bin/engine
```

### 5.2.3 Worker

```
python bin/worker
```

## 5.3 Run Tests

We use the `pytest` testing framework.

---

**Note:** The tests use a separate db (sqlite in memory), so don't worry about corrupting a production db when running the tests.

---

First, we need to install the dependencies required for testing.

```
pip install -r tests/requirements.txt
```

Next, we run our tests

```
pytest tests
```

---

**Hint:** Instead of specifying the tests directory, you can specify specific file(s) to run: *pytest tests/scoring\_engine/test\_config.py*

---

## 5.4 Modifying Documentation

We use `sphinx` to build the documentation.

First, we need to install the dependencies required for documentation.

```
pip install -r docs/requirements.txt
```

Next, we build our documentation in html format.

```
cd docs
make html
open build/html/index.html
```