# Contents

1 FEATURE HIGHLIGHTS ........................................... 3  
2 OVERVIEW OF FUNCTION .................................... 5  
3 SITEMIND SCORING SYSTEM .................................. 7  
4 DATA TAXONOMY .................................................. 9  
5 DATA SOURCES .................................................... 11  
6 PROCESS FLOW .................................................... 13  
7 DIRECTORY STRUCTURE ....................................... 15  
8 GETTING STARTED ................................................. 17  
  8.1 HTTPS WITH LETSENCRYPT .................................. 18  
9 DEBUGGING .......................................................... 19  
10 RUNNING LOCALLY ............................................... 21  
11 CODING CONVENTIONS ....................................... 23  
12 FUTURE DEVELOPMENT ......................................... 25  
13 ADMIN FEATURES ................................................. 27  
  13.1 SERVER CONFIGURATION .................................... 28
Domain research tool targeting media planners and researchers, specifically built for countering ad fraud and reducing its impact on media investment. Returns a result with up to 40 signals for any site typically in 1-3 seconds.
FEATURE HIGHLIGHTS

- intuitive ‘buy score’ system for website rating
- search any domain
- returns result usually in 1 to 2 seconds
- up to 150 data points per site from 5 different sources
- easy to use API with ready end-points for all common languages
SiteMind allows two different kinds of searches to be performed by the user:

- type-1: where a single domain name is the input
- type-2: where a comma separated list of domain names is the input

In both cases the system performs a series of operations resulting in up to 40 signals, which are then stored in a .csv file. Depending on the type of search, the result will then be returned either as a simple user interface, or a table with results for multiple sites.
CHAPTER 3

SITEMIND SCORING SYSTEM

The SiteMind scoring system takes widely accepted “red flags” from signals available from various data sources (see the sections below) and creates a single easy to understand score out of those flags.

The formula to calculate the score ranging from 0 to 100 is as follows:

$$100 - ((\text{CHECKS FAILED} / \text{CHECKS TOTAL}) * 100) = \text{SiteMind SCORE}$$

The score consist of 10 “flags”.

<table>
<thead>
<tr>
<th>VARIABLE NAME</th>
<th>FAILS WHEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCORE_CHECKS</td>
<td>Not enough signals to perform 4 checks</td>
</tr>
<tr>
<td>SCORE_UPSTREAM</td>
<td>More than 90% of the traffic coming from TOP5 Upstream</td>
</tr>
<tr>
<td>SCORE_UPSTREAMCHECK</td>
<td>No common sites in TOP5 Upstream</td>
</tr>
<tr>
<td>SCORE_TRUST</td>
<td>Web of Trust Trust score is less than 50</td>
</tr>
<tr>
<td>SCORE_TOPKEYWORDS</td>
<td>More than 90% of the traffic coming from TOP5 Keywords</td>
</tr>
<tr>
<td>SCORE_SEARCH</td>
<td>Less than 1% of traffic is coming from search</td>
</tr>
<tr>
<td>SCORE_PAGEVIEWS</td>
<td>More than 8 pageviews per visit on average</td>
</tr>
<tr>
<td>SCORE_YEARS</td>
<td>Domain was originally registered less than 2 years ago</td>
</tr>
<tr>
<td>SCORE_PRIVACY</td>
<td>Domain uses whois privacy guard</td>
</tr>
<tr>
<td>SCORE_BOUNCERATE</td>
<td>Site bouncerate is less than 10% on average</td>
</tr>
</tbody>
</table>
The below table shows all the signals that are currently available through SiteMind. All variables are available through the scan function in the resulting .csv file, or in the user interface resulting from a single site search.

**NOTE:** Different naming may be used in the user interfaces, and this is easily changed.

**VARIABLE NAME** is the name of the variable as it is found in the output file resulting from a search of scan.

**SOURCE** is the reference to where the data is originating from. In the case the filed says ‘sitemind’ it means that the signal is inferred from other data.

**COLUMN NUMBER** is only for development purpose and is used in the UI codes to to present a certain signal in a given place in the user interface.

<table>
<thead>
<tr>
<th>VARIABLE NAME</th>
<th>SOURCE</th>
<th>COLUMN NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCORE_CHECKS</td>
<td>sitemind</td>
<td>2</td>
</tr>
<tr>
<td>SCORE_UPSTREAM</td>
<td>sitemind</td>
<td>3</td>
</tr>
<tr>
<td>SCORE_UPSTREAMCHECK</td>
<td>sitemind</td>
<td>4</td>
</tr>
<tr>
<td>SCORE_TRUST</td>
<td>sitemind</td>
<td>5</td>
</tr>
<tr>
<td>SCORE_TOPKEYWORDS</td>
<td>sitemind</td>
<td>6</td>
</tr>
<tr>
<td>SCORE_SEARCH</td>
<td>sitemind</td>
<td>7</td>
</tr>
<tr>
<td>SCORE_PAGEVIEWS</td>
<td>sitemind</td>
<td>8</td>
</tr>
<tr>
<td>SCORE_YEARS</td>
<td>sitemind</td>
<td>9</td>
</tr>
<tr>
<td>SCORE_PRIVACY</td>
<td>sitemind</td>
<td>10</td>
</tr>
<tr>
<td>SCORE_BOUNCERATE</td>
<td>sitemind</td>
<td>11</td>
</tr>
<tr>
<td>ADMIN_CITY</td>
<td>whois</td>
<td>12</td>
</tr>
<tr>
<td>ADMIN_COUNTRY</td>
<td>whois</td>
<td>13</td>
</tr>
<tr>
<td>ALEXA_BOUNCERATE</td>
<td>alexa</td>
<td>14</td>
</tr>
<tr>
<td>ALEXA_INLINKS</td>
<td>alexa</td>
<td>15</td>
</tr>
<tr>
<td>ALEXA_LOADSPEED</td>
<td>alexa</td>
<td>16</td>
</tr>
<tr>
<td>ALEXA_PAGEVIEWS</td>
<td>alexa</td>
<td>17</td>
</tr>
<tr>
<td>ALEXA_RANK</td>
<td>alexa</td>
<td>18</td>
</tr>
<tr>
<td>ALEXA_SEARCHVISITS</td>
<td>alexa</td>
<td>19</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>VARIABLE NAME</th>
<th>SOURCE</th>
<th>COLUMN NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALEXA_TIMEONSITE</td>
<td>alexa</td>
<td>20</td>
</tr>
<tr>
<td>ALEXA_TOPCOUNTRIES</td>
<td>alexa</td>
<td>21</td>
</tr>
<tr>
<td>ALEXA_TOPKEYWORDS</td>
<td>alexa</td>
<td>22</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM1</td>
<td>alexa</td>
<td>23</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM1N</td>
<td>alexa</td>
<td>24</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM2</td>
<td>alexa</td>
<td>25</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM2N</td>
<td>alexa</td>
<td>26</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM3</td>
<td>alexa</td>
<td>27</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM3N</td>
<td>alexa</td>
<td>28</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM4</td>
<td>alexa</td>
<td>29</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM4N</td>
<td>alexa</td>
<td>30</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM5</td>
<td>alexa</td>
<td>31</td>
</tr>
<tr>
<td>ALEXA_UPSTREAM5N</td>
<td>alexa</td>
<td>32</td>
</tr>
<tr>
<td>CHECKS_TOTAL</td>
<td>alexa</td>
<td>33</td>
</tr>
<tr>
<td>CHECK_FALSE</td>
<td>alexa</td>
<td>34</td>
</tr>
<tr>
<td>CHECK_TRUE</td>
<td>alexa</td>
<td>35</td>
</tr>
<tr>
<td>TOP5_UPSTREAM</td>
<td>alexa</td>
<td>36</td>
</tr>
<tr>
<td>WHOIS_PRIVACY</td>
<td>whois</td>
<td>37</td>
</tr>
<tr>
<td>WHOIS_YEARS</td>
<td>whois</td>
<td>38</td>
</tr>
<tr>
<td>WOT_CHILDSAFETY</td>
<td>weboftrust</td>
<td>39</td>
</tr>
<tr>
<td>WOT_TRUST</td>
<td>weboftrust</td>
<td>40</td>
</tr>
</tbody>
</table>
While adding virtually any additional data source, SiteMind relies on three different data source by default.

- Alexa
- Web of Trust
- WHOIS

**ALEXA***

It is recommended to use the paid Alexa API. SiteMind uses web scraping method by default for demo and prototyping purpose.

**Web of Trust**

Web of Trust data is fetched using the WOT API, which provides a rich data taxonomy and is free to use to a substantial level of daily usage.

More information on the WOT API can be found here: https://www.mywot.com/wiki/API

You can apply for your own API key here: https://www.mywot.com/en/reputation-api

**WHOIS**

SiteMind provides a fully automated method for the “gold standard” way of fetching WHOIS records.

1. Gets to main record from the tld level registrar including the registrar that holds the sub-record
2. Gets the sub-record from the holding registrar
1. User provides input through the search field in the UI
   - > form_process.php
   - > run.sh

2. run.sh checks if there query is empty, single domain, or multiple comma separated domains
   - > sitemind.sh (“controller”)

3. Regardless if it's single or multi search the program cycle proceeds
   - > bin/api-fetch.sh
     - > bin/alexa_data.sh
     - > bin/whois_data.sh
     - > bin/wot_data.sh
     * > wo_data.py

4. Using the data in various .temp and .bash files a usable data format is created
   - > bin/api-build.sh

5. The data is provided in a comma separated format for multi searches
   - > data-export.sh

6. The data is further formatted for the UI building process
   - > data-cms.sh

7. The UIs are built each in a separate script
   - > cms/cms-scorecard.sh
   - > cms/cms-traffic.sh
   - > cms/cms-overview.sh
   - > cms/cms-upstream.sh
8. A finish cleanup is performed
   • `finish-cleanup.sh`
NOTE: In a multi-user system, each user has a self-contained replica of the program folder in the program root.

<table>
<thead>
<tr>
<th>FOLDER</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>Sitemind program root</td>
</tr>
<tr>
<td>/bin</td>
<td>Where non UI scripts reside</td>
</tr>
<tr>
<td>/cms</td>
<td>Where the UI scripts reside</td>
</tr>
<tr>
<td>/cms/graphics</td>
<td>Images for the UI</td>
</tr>
<tr>
<td>/cms/js</td>
<td>Javascripts for the UI</td>
</tr>
<tr>
<td>/cms/style</td>
<td>Style sheets for the UI</td>
</tr>
<tr>
<td>/cms/templates</td>
<td>Header and Footer for UI</td>
</tr>
</tbody>
</table>
The following installation instructions have been tested on Ubuntu 16.04 clean distro.

**Install dependencies:**

```bash
sudo apt-get update
sudo apt-get install -y apache2
sudo apt-get install -y php5
sudo apt-get install -y unzip
sudo apt-get install -y parallel
sudo apt-get install -y num-utils
sudo apt-get install -y git
```

**Getting the source files and setting it up:**

```bash
wget https://github.com/SiteMindOpen/SiteMind/archive/master.zip
unzip master.zip
sudo rsync -av ~/SiteMind-master/ /var/www/html
```

After the initial setup, as long as you create new users with SiteMind command line command ‘sm-user-new’, permissions will be handled automatically and is not something you need to think about.

**Creating an admin user:**

```bash
PASSWORD=$(openssl rand -base64 20); htpasswd ./etc/apache2/.htpasswd -cbB admin "$PASSWORD"; echo -e "Your password is $PASSWORD"
```

**Restart apache**

Ubuntu 14.04:

```bash
service apache2 restart
```

Ubuntu 16.04:
8.1 HTTPS WITH LETSENCRYPT

Letencrypt makes it incredibly easy (and fast) to setup functional https for your site.

Note that for the below to work, you need to have a valid domain name that is pointed to the server you’re initiating the below command from:

```
sudo git clone https://github.com/letsencrypt/letsencrypt /opt/letsencrypt
cd /opt/letsencrypt
./letsencrypt-auto --apache -d yoursite.com
```

NOTE: as part of the setup process, there will be a prompt asking if you want to redirect all requests to https. I think this should be on for most cases.
CHAPTER 9

DEBUGGING

For DATA related debugging change production_version to debug_function from line 24 in bin/api-fetch.sh. This will help you to identify issues with one part of the data fetching cycle getting stuck. This should happen very rarely as it has been debugged a lot.

For UI related download the program folder to a local machine and run a PHP server locally. This way you will very easily see any error messages that are coming up when the UI is loaded.

If you’ve setup properly, then you can easily see related error logs on the server-side using:

```
./sm-monitor
```
You have to run a PHP server from the Sitemind folder to be able to make queries from the UI:

```
php -s http://127.0.0.1:8000
```

If you’re a mac user, go the Sitemind folder and execute the below command:

```
sudo php -S 127.0.0.1:8000 && /Applications/Google\ Chrome.app/Contents/MacOS/Google\ →Chrome --app="http://127.0.0.1:8000/dev/index.html" --window-size="1000x800"
```

Alternatively you can run from the command line (in the Sitemind folder):

```
./run.sh domain.com
```
CODING CONVENTIONS

The code is almost 100% bash and certain principles have been followed where possible:

- code starts one tab intend deep
- each script (.sh file) represents a step in the process flow
- no more than 50 lines of code per script
- no more than 50 characters long lines of code
- functions first, program second, cleanup last
- minimal comments - instead self-explaining code

It should be very easy for anyone with beginner+ level in bash to modify the code that is already there, to add new code to improve current functionality, or add completely new functionality.
FUTURE DEVELOPMENT

• Create setup process where server is configured including SSL and a conf file is created at ~/.sitemindrc
• Make upstream sites clickable (yields a new search)
• Check for native advertising being a major source of traffic
• Add a 30 day cache to avoid redundant searches
• Make one-page report for export available with all the signals
• time-limited account creation
In the environment of the host machine, include the following alias commands:

```bash
alias sm-sync='/var/www/html/admin/bin/sync.sh'
alias sm-user-list='cat /etc/apache2/.htpasswd | cut -d: -f1'
alias sm-monitor='/var/www/html/admin/bin/monitor.sh'
alias sm-user-new='/var/www/html/admin/bin/user-new.sh'
alias sm-user-rm='/var/www/html/admin/bin/user-sh.sh'
alias sm-commit='/var/www/html/admin/bin/commit.sh'
alias sm-commit-version='cd ~/git/sitemind && /var/www/html/admin/bin/commit-version.sh'
alias sm-commit-log='git log --oneline --decorate --color'
alias sm-conf-nossl='vim /etc/apache2/sites-available/000-default.conf'
alias sm-conf-ssl='vim /etc/apache2/sites-available/000-default-le-ssl.conf'
alias sm-find-file='/var/www/html/admin/bin/sm-find-file.sh'
```

Usually you can find the file from ~/ under the name .bashrc. Add the above lines in to the file and next time you login to the host, the following commands will be available anywhere in your system:

In a Linux system you can do this typically by:

```bash
vim .bashrc
```

**sm-sync**

Syncs all the user accounts with /dev.

**sm-user-list**

Prints out a list of user accounts.

**sm-monitor**

Creates a report out of access and error logs from the on going day’s logs.

**sm-user-new**

Creates a new user in to the system and prints out a randomly generated password for the user.
EXAMPLE USAGE (where we want to create a user ‘john’):

```
sm-newuser john
```

sm-user-rm

Removes a user and all associated files from the system (Use with caution!).

## 13.1 SERVER CONFIGURATION

**sm-conf-nossl**

Opens up the no ssl (port 80) apache configuration file in vim editor.

**sm-conf-ssl**

Opens up the ssl (port 443) apache configuration file in vim editor.