Bimeta Client Manual

Blueintel

October 2, 2020

Background

This document describes the usage of the bimeta client created by Blueintel. Bimeta is Blueintel's way of storing information about organizational assets that are monitored for possible adversary activity.

Organizations store the following in bimeta:

- Lists of IP addresses, domains, and keywords to monitor.
- Personnel who receive alerts.

bimeta also makes querying threat intelligence data for your organization easy. The following data is currently available to be queried and additional data types will be added in the future:

- exposed github data
- mimicked domains (available soon)

The bimeta client program runs on Microsoft Windows, Linux, and Apple computers. Instructions to download bimeta will be provided to you during the onboarding process. All objects in bimeta have security permissions associated with them. You will only be able to manipulate objects for which you are authorized.

Installation

Follow these steps to install bimeta:

1. Download the .zip file from github or use "git clone" to make a copy of the github repository. The link to the github repository will be given you as part of the onboarding process.
2. If you downloaded the .zip file, unzip it.
3. Create a configuration file as described in the next section and place it in the same directory as the executable. The exact steps will vary based on your operating system.
4. bimeta is now installed. Run it by typing "./bimeta.py" to see the options available.
5. You can put the the bimeta directory in your path if you do not want to type the full path name to bimeta.py or the "./" when bimeta.py is in your current directory.
6. Use "git pull" at any time to update your copy of bimeta.
7. All commands below assume that bimeta.py is in your path.

bimeta Configuration File

Configuration of bimeta consists of creating a small text configuration file named "bimeta_config.toml" and placing it in the same directory as the bimeta client. You only need to change the apikey value in the configuration file to your API key. Make sure to include double quotes around your API key. You will receive your API key directly from a member of the Blueintel team during the onboarding process. A sample configuration file is shown below.

```
[bimeta_client]
apikey = "<your apikey here>"
ipAddr = "api.badpanda.org"
ipPort = "443"
```

Usage

If you don't need to know all the details of how to use bimeta and just want to get up and running quickly, skip ahead to the section titled "Common Tasks".

bimeta uses a VERB-NOUN scheme to specify the action you want to perform and the type of object you want to work with. For example, the following command retrieves information about the user with an email of "user@example.org":

```
bimeta.py get user --email user@example.org
```

The general form of a bimeta command is:

```
bimeta.py <verb> <noun> [<arguments>...]
```

Probably the most useful bimeta option to remember is the "–help" option to view the available options. For example:

```
bimeta.py --help
```

The `--help` option can also be used after each of the other options to find out what options are available next (e.g. bimeta get `--help` shows what can come after "get"). The bimeta options are described in the following sections.

Command Line

The following are the types of objects stored in bimeta:

- org - This is the organization to which you belong. Blueintel will create an organization record for you during the onboarding process. You will specify the organization name as an option to some of the commands.
- user - User belonging to an organization. Organizations may have multiple users. Each user may be configured to receive different kinds of communications from Blueintel.
- config - Configuration file associated with an organization. In the future, organizations may have more than one configuration file to solve specific needs, but currently the only configuration file used is named "badpanda.config". Additional sections below document the format of the "badpanda.config" file.
- tag - Additional information about a user in a key, value format (for example, "tech=true"). Users and organizations may have more than one tag. Examples of how to use tags are give below.
- github-alerts - blueintel monitors github for exposed data. You may use bimeta to retrieve the names of github repositories that may expose your organization's data.

The following verbs create/read/update/delete these objects:

- put - Create a new object.
- set - Update a previously created object.
- del - Delete an object.
- get - Retrieve information about an object.

Output from bimeta is always in JSON format.

Global Options

One or more global options may be specified on the command line as shown in Table 1.

Global Option	**Description**
--help (or just help)	This is the same help text that is displayed if no options are specified on the command line.
--config	Specifies the location the configuration file with your API key and the location of the bimeta server. Defaults to the bimeta executable directory.

Table 1: Global Options

Configuration Files

Maintain configuration files for your organization by using the "put/get/set/del config" commands. Configuration files contain the keywords and domain names that should be monitored for your organization. bimeta currently recognizes two types of configuration files: "githubsearch.config" and "badpanda.config". githubsearch.config specifies the criteria for searching github for data inadvertently exposed by your organization. "badpanda.config" specifies your domains and keywords that should be monitored for mimicked domains or domains closely related to your organization.

The typical workflow is to create a configuration file in a text file and then upload it to bimeta with either put (to create a new configuration file), or set (to update a configuration file). Table 2 lists the arguments that apply to the config commands and which verbs each option may be used with.

Short	**Long**	**Description**	**Verbs used in**
-o	--org	Name of organization	put/get/set/del
-n	--name	"githubsearch.config" or "badpanda.config"	put/get/set/del
-f	--file	Name of file to upload to bimeta. This is the filename of the file on your system you wish to upload.	put/set

Table 2: Configuration Options

Users

Maintain users for your organization using the "put/get/set/del user" commands. Create users when you want them to receive emails from blueintel. Users receive emails determined by the tags you associate with them. See the User Tags section below for details.

Table 3 lists the arguments that apply to users and which verbs each option may be used with.

Short	**Long**	**Description**	**Verbs used in**
-e	--email	Email of user	put/set/get/del
-n	--name	Name of user	put/set

-t	--title	Title of user	put/set
-o	--org	Organization to which user belongs.	put/set
-c	--clearance	Clearance of user (red, amber, yellow, green). You may only assign a clearance equal to or below your own clearance.	put/set
-i	--id	Unique identifier assigned by bimeta to a user when they are added. --id must be used when updating a user. The id may be retrieved using the get verb and the --email option.	get/set

Table 3: User Options

You may retrieve a list of users assigned to your organization using the following command and replacing ”<orgname>” with the name of your organization.

```
bimeta.py get user -o <orgname>
```

User Tags

You maintain tags related to an organization by using the ”put/get/del tag” commands. Only three user tags are in current use: ”tech”, ”comm”, and ”exec”. Table 4 lists the arguments that apply to the tag commands and which verbs each option may be used with. Tags may not be updated. Incorrect tags may be deleted and new tags added.

Short	**Long**	**Description**	**Verbs used in**
-e	--email	Email of user	put/get/del
-t	--tags	Comma delimited list of tags to add to user in the format ”tag-name1=value1,tagname2=value2,...”. Multiple tags may be specified in a single command by separating them with ”,”. Spaces are not allowed.	put
-n	--name	Comma delimited list of tag names to be deleted.	del

Table 4: User Tag Options

Use the following command to retrieve all the tags related to a user, replacing ”<email>” with the email of the user whose tags are to be retrieved.

```
bimeta.py get tag --email <email>
```

Common Tasks

The following subsections walk through common tasks you will need to do when working with bimeta. Some of the commands are line wrapped to fit on the page but each command must be typed on a single line.

Get Help

If you type bimeta with no options, it displays a help message. You can also specify --help at any point in the command. Examples of adding --help to the ”get” verb and the ”get user” command are shown below. Adding --help to other verbs and commands works similarly.

`bimeta.py get --help` - Displays valid objects for the ”get” verb.

`bimeta.py get user --help` - Displays valid options for the ”get user” command.

Add a User

Adding a user requires specifying a number of options. All the options in the example below are required. The text of the command is wrapped below to fit on the page but the entire command must be on a single line. You will typically set the tags to ”tech=true,comm=true” for that user to receive technical emails and intelligence reports from Blueintel. Change ”true” to ”false” for users that will not receive technical email (”tech=false”) or intelligence reports (”comm=false”). All options shown in the example are required.

```
bimeta.py put user --org example --email john.doe@example.com
        --name "John Doe" --title "Security Analyst"
        --clearance amber --tags "tech=true,comm=true"
```

Remove a User

Removing a user just requires specifying the user’s email address. Users need to be removed whenever they no longer work for your organization or they no longer need to receive emails from bimeta.

```
bimeta.py del user --email john.doe@example.com
```

Update a User

Updating a user requires two steps. First, find the user’s unique id using ”bimeta get user” and then update the user by specifying the id as one of the parameters.

Example:

```
bimeta.py get user --email <user email>  (id will be displayed)
```

```
bimeta.py set user --id=<id> ...
```

When updating a user, you only need to provide the options for the fields that are being updated. These are the same options used to add a user.

Create the badpanda Configuration File

Assuming that the badpanda.config file already exists on your computer, use the following command to add it to bimeta (line wrapped due to space limitations):

```
bimeta.py put config --org <orgname> --name badpanda.config
        --file <configuration file name>
```

The configuration file on your computer does not need to be named badpanda.config. Documentation for the format of the badpanda.config file is in the ”badpanda Configuration File Manual”.

Retrieve the badpanda Configuration File

Use a command similar to the following to retrieve your badpanda.config file:

```
bimeta.py get config --org <orgname> --name badpanda.config
```

This command sends the output to your screen. You may save the output to a file by adding `"> <output filename>"` to the command.

Update the badpanda Configuration File

Assuming that the updated version of the badpanda.config file already exists on your computer, use a command similar to the following to update it in bimeta (line wrapped due to space limitations):

```
bimeta.py set config --org <orgname> --name badpanda.config
    --file <configuration file name>
```

You must use "set" to update the configuration file. If you try to use "put" to update the configuration file, you will receive an error indicating that the configuration file already exists. The configuration file on your computer does not need to be named badpanda.config.

Assign Tag(s) to a User

Tags are assigned to users when they are added but sometimes the tags need to be changed. There are currently two tags in use, "tech" and "comm". "tech=true" means that that user will receive technical emails. "comm=true" means that that user will receive intelligence reports from blueintel. Changing either one to false will disable that type of email for that user. Tags are added and updated using the following command:

```
bimeta.py put tag --email <user email> --tags "<tags to set>"
```

Multiple tags may be specified by separating them with commas. For example "tech=true,comm=true". Tags are deleted using the following command:

```
bimeta.py del tag --email <user email> --name "<name of tag>"
```

Only specify the name of the tag when deleting it. Do not include the "=true/false".