
Burp-UI Documentation

Release 0.4.2

Ziirish

January 03, 2017

1	Documentation	3
1.1	Introduction	3
1.2	Requirements	4
1.3	Installation	5
1.4	Upgrading	8
1.5	Architecture	9
1.6	Usage	10
1.7	Manage	19
1.8	Celery	22
1.9	Gunicorn	22
1.10	Docker	26
1.11	bui-agent	27
1.12	Contributing	30
1.13	Changelog	32
1.14	FAQ	38
1.15	Step By Step	40
1.16	Developer Guide	43
	HTTP Routing Table	101

Burp-UI is a web-ui for [Burp](#) backup written in python with Flask and jQuery/Bootstrap. You may have a look at the [README](#) file first. There is also a dedicated FAQ.

Documentation

1.1 Introduction

Burp-UI is a web-based interface for **Burp**. Its purpose is to give you a *nice* way to monitor your backups with some dashboards, but you will also have the ability to download files from backups and to configure your burp-server.

The project also provides a full documented API so that you can develop any front-end you like on top of it. The core will take care of the communication with the burp server(s) for you.

Note: Although the **Burp**'s author and I exchange a lot, our products are totally distinct. So I would like people to understand some issues might be related to **Burp-UI**, but some other might be related to **Burp** and I may not be able to help you in the later case. There is a dedicated mailing-list for **Burp** related issues. You can find details [here](#)

1.1.1 Compatibility

Burp version Backend	1	2
< 1.3.48		
1.3.48 => 1.4.40	X	
2.0.0 => 2.0.16		
2.0.18 => 2.0.X protocol 1		X
2.0.18 => 2.0.X protocol 2		X*

* The protocol 2 is in heavy development Burp side so the support in **Burp-UI** is best effort and all features (such as server-initiated restoration) are not available.

1.1.2 Known Issues

Because it's an Open Source project, people are free (and encouraged) to open issues in the [bug-tracker](#). You will also find there the current opened issues.

There are also a few issues unrelated to the code itself:

1. SSH issue

People that would like to clone the repository over SSH will face an authentication failure even if they added a valid SSH key in their user settings. The reason is I only have *one* public IP address so I must use port redirections to have multiple SSH instances running. To fix the issue, you should configure your SSH client by adding the following lines in your `~/.ssh/config` file:

```
Host git.ziirish.me
Port 2222
```

1.2 Requirements

Please note that, **Burp-UI** must be running on the same server that runs the burp-server for some features.

Note: At the moment, **Burp-UI** and this doc is mostly debian-centric but feel free to contribute for other distributions!

1.2.1 Python

Burp-UI is built against python 2.7. The support for python 2.6 has been removed since it is not supported anymore by the CPython core team. Unit tests are ran against python 2.7 and python 3.4. If you encounter compilation errors with one of these version, feel free to report them.

1.2.2 Libraries

Some libraries are required to be able to compile some requirements:

```
apt-get install libffi-dev libssl-dev python-dev python-pip
# this package conflicts with the pip requirements
apt-get remove python-cffi
```

1.2.3 Debian Wheezy

It looks like some requirements are not automatically installed on *Debian Wheezy*. You can install them with the following command:

```
pip install "burp-ui[debian_wheezy]"
```

1.2.4 LDAP

For LDAP authentication (optional), we need extra dependencies. You can install them using the following command:

```
pip install "burp-ui[ldap_authentication]"
```

1.2.5 Local

For Local authentication (optional), we need extra dependencies as well. You can install them using the following command:

```
pip install "burp-ui[local_authentication]"
```


1.2.6 Redis

If you wish to use redis for Caching and/or managing uses sessions, you need additional dependencies:

```
pip install "burp-ui[unicorn-extra]"
```

1.2.7 Celery

The celery worker also needs additional dependencies that you can install using:

```
pip install "burp-ui[celery]"
```

1.2.8 SQL

If you need persistent data, you will need additional dependencies as well:

```
pip install "burp-ui[sql]"
```

1.2.9 Burp1

The burp1 backend supports burp versions from 1.3.48 to 1.4.40. With these versions of burp, the status port is only listening on the local machine loopback interface (ie. localhost or 127.0.0.1). It means you *MUST* run **Burp-UI** on the same host that is running your burp server in order to be able to access burp's statistics. Alternatively, you can use a bui-agent.

1.2.10 Burp2

The burp2 backend supports only burp 2.0.18 and above. Some versions are known to contain critical issues resulting in a non-functional **Burp-UI**: 2.0.24, 2.0.26 and 2.0.30 If you are using an older version of burp2 **Burp-UI** will fail to start.

1.3 Installation

Burp-UI is written in Python with the **Flask** micro-framework. The easiest way to install **Burp-UI** is to use `pip`.

```
pip install burp-ui
```

You can setup various parameters in the `burpui.cfg` file. This file can be specified with the `-c` flag or should be present in `/etc/burp/burpui.cfg`. By default **Burp-UI** ships with a sample file located in `$INSTALLDIR/share/burpui/etc/burpui.sample.cfg`. (*\$INSTALLDIR* defaults to */usr/local* when using `pip` **outside** a virtualenv)

Note: It is advised to copy the sample configuration in `/etc/burp/burpui.cfg` and to edit this file so that it is not overwritten on every upgrade.

Then you can run `burp-ui: burp-ui`

By default, `burp-ui` listens on localhost on port 5000.

You can then point your browser to `http://127.0.0.1:5000/`

1.3.1 Upgrade

In order to upgrade **Burp-UI** to the latest stable version, you can run the following command:

```
pip install --upgrade burp-ui
```

Note: If you encounter any issue after upgrading to the latest stable release, make sure you read the upgrading page.

1.3.2 Debian Wheezy

The version of `pip` available on *Debian Wheezy* does not support all the features needed to build and install the latest **Burp-UI** version.

Instead, you may want to run the following command either to install it from scratch or to upgrade your current version to the latest one:

```
easy_install --upgrade burp-ui
```

1.3.3 General Instructions

Restoration

Burp-UI tries to be as less intrusive as possible with **Burp** internals. In order to make the *online* restoration/download functionality work, you need to check a few things:

1. Provide the full path of the burp (client) binary file (field *burpbin* in burp-ui configuration)
2. Provide a burp-client configuration file (field *bconfcli* in burp-ui configuration)
3. Provide the full path of an empty directory where a temporary restoration will be made. This involves you have enough space left on that location on the server that runs **Burp-UI**
4. Launch **Burp-UI** with a user that can proceed restorations and that can write in the directory mentioned above
5. Make sure the client provided in 2. can restore files of other clients (option *restore_client* in burp-server configuration). The *restore_client* is the *cname* you provided in your client configuration file (see 2.)

Burp 2

When using the burp2 backend, **Burp-UI** can be executed on any machine as long as you can access the burp status port, but you will not be able to edit the burp server configuration file within the *settings* view of **Burp-UI**. You also need to configure a *restore_client* on your burp server corresponding to the client you will use through **Burp-UI**

1.3.4 Options

```
usage: burp-ui [-h] [-v] [-d] [-V] [-c <CONFIG>] [-l <FILE>]
              [-i <MIGRATIONSDIR>]
              ...

positional arguments:
  remaining
```

optional arguments:

```
-h, --help            show this help message and exit
-v, --verbose          increase output verbosity (e.g., -vv is more verbose
                        than -v)
-d, --debug           enable debug mode
-V, --version          print version and exit
-c <CONFIG>, --config <CONFIG>
                        burp-ui configuration file
-l <FILE>, --logfile <FILE>
                        output logs in defined file
-i <MIGRATIONS_DIR>, --migrations <MIGRATIONS_DIR>
                        migrations directory
```

1.3.5 Developer options

Since v0.4.0, [Burp-UI](#) uses the new Flask's *CLI* module. This change brings new options to help you debug your development environment:

Usage: flask run [OPTIONS]

Runs a local development server for the Flask application.

This local server is recommended for development purposes only but it can also be used for simple intranet deployments. By default it will not support any sort of concurrency at all to simplify debugging. This can be changed with the `--with-threads` option which will enable basic multithreading.

The reloader and debugger are by default enabled if the debug flag of Flask is enabled and disabled otherwise.

Options:

```
-h, --host TEXT        The interface to bind to.
-p, --port INTEGER     The port to bind to.
--reload / --no-reload Enable or disable the reloader. By default
                        the reloader is active if debug is enabled.
--debugger / --no-debugger Enable or disable the debugger. By default
                        the debugger is active if debug is enabled.
--eager-loading / --lazy-loader Enable or disable eager loading. By default
                        eager loading is enabled if the reloader is
                        disabled.
--with-threads / --without-threads Enable or disable multithreading.
--help                Show this message and exit.
```

Some options are redundant with [Burp-UI](#)'s one.

1.3.6 Example

By default, the embedded server listens on `127.0.0.1` on port `5000`, if you wish to change this, you would run something like:

```
burp-ui -- -h 0.0.0.0 -p 8080
```

Note: The `--` is mandatory.

1.4 Upgrading

This page is here to help you upgrading from previous versions of [Burp-UI](#) to the latest version. Each section presents major/breaking changes, new requirements and new options. For a complete list of changes, you may refer to the [CHANGELOG](#) page.

1.4.1 v0.4.0

- **Breaking** - Due to the use of the new Flask's embedded server, it is no longer possible to serve the application over SSL (HTTPS) anymore from within the Flask's server. You'll need to use a dedicated application server for this purpose such as gunicorn or a reverse-proxy.

Or you can use the `python -m burpui -m legacy [--help]` command that **SHOULD** be backward compatible (but note that no further support will be provided since it is not the Flask's default behavior anymore).

- **Breaking** - The database schema evolved between v0.3.0 and v0.4.0. In order to apply these modifications, you **MUST** run the `bui-manage db upgrade` command before restarting your [Burp-UI](#) application (if you are using celery, you must restart it too).

More details on the [Manage](#) and [Celery](#) pages.

- **Breaking** - Plain text passwords are deprecated since v0.3.0 and are now disabled by default. It means you should not manually add new users in your burp-ui configuration anymore with `login = password` but you should now use the `bui-manage` command instead.
- **Breaking** - The default `version` setting has been set to 2 instead of 1 since burp-2.0.52 is now the stable release.
- **New** - The `bui-manage` tool can now help you setup both [Burp](#) and [Burp-UI](#).
- **New** - The SQL requirements have evolved, you **MUST** run `pip install --upgrade "burp-ui[sql]"` if you wish to keep using persistent storage.

1.4.2 v0.3.0

- **New** - `bui-manage` tool: This tool is used to setup database (see [Manage](#)).
- **New** - `bui-celery` tool: This tool is used to run a celery runner (see [Celery](#)).
- **Breaking** - Configuration file format changed. Colons (:) must be replaced by equals (=). Besides, some settings containing spaces should be surrounded by quotes. *Note:* The conversion is mostly automatic, but you should keep an eye on it though.
- **New** - Basic authentication backend now supports hashed passwords (*Note:* plain text passwords are now deprecated and the support will be dropped in v0.4.0). You can create new users with the `bui-manage` tool, passwords generated through this tool are hashed. *Note:* Starting with v0.4.0, plain text passwords will be automatically hashed.
- **New** - Local authentication backend allows you to login using local accounts through pam.

1.5 Architecture

This section is a must-read in order to understand what is going on between [Burp](#) and [Burp-UI](#).

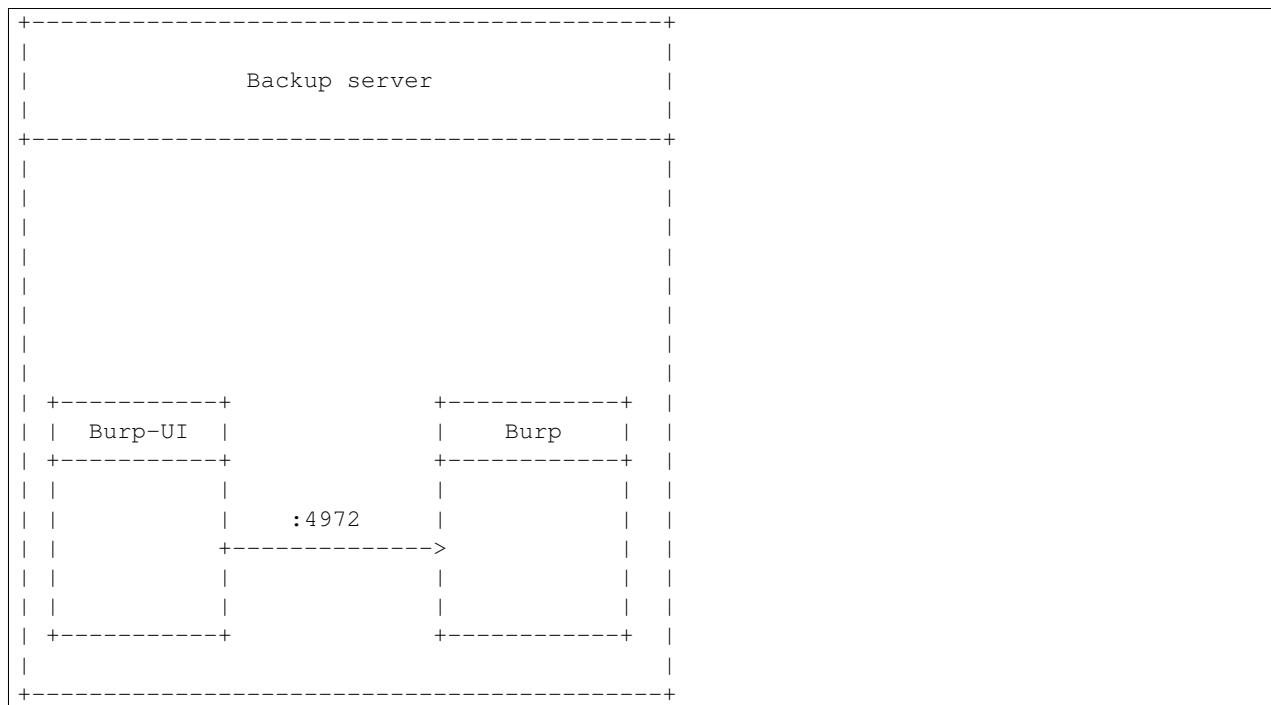
Both projects are lead by two different people so please report your issues to the right project.

The *Burp1* and *Burp2* backends behave slightly differently due to some changes in the core or [Burp](#). You can refer to the [Burp's documentation](#) for details.

1.5.1 Burp 1.x

If you are running [Burp](#) 1.x, you **MUST** install [Burp-UI](#) on the same host (or at least setup a bui-agent locally). This limitation is due to the fact burp 1.x only exposes its *status port* (the port 4972 by default) to localhost (either 127.0.0.1 or ::1). [Burp-UI](#) then just opens a connexion to `localhost : 4972` in order to communicate with [Burp](#).

Here is a little illustration:



1.5.2 Burp 2.x

If you are running [Burp](#) 2.x, you can host [Burp-UI](#) on a different server, but I don't recommend it if you wish to be able to use all the features.

The [Burp](#) 2.x *status port* has been completely reworked. It can now be published to remote hosts, but the *status protocol* is not compatible with 1.x.

The *status port* is now accessed through a [Burp](#) client thanks to the `-a m` flag. The client will then take care to open the connexion with the server securing the communication with SSL. The [Burp](#) server also supports basic *ACL* though the `restore_client` option.

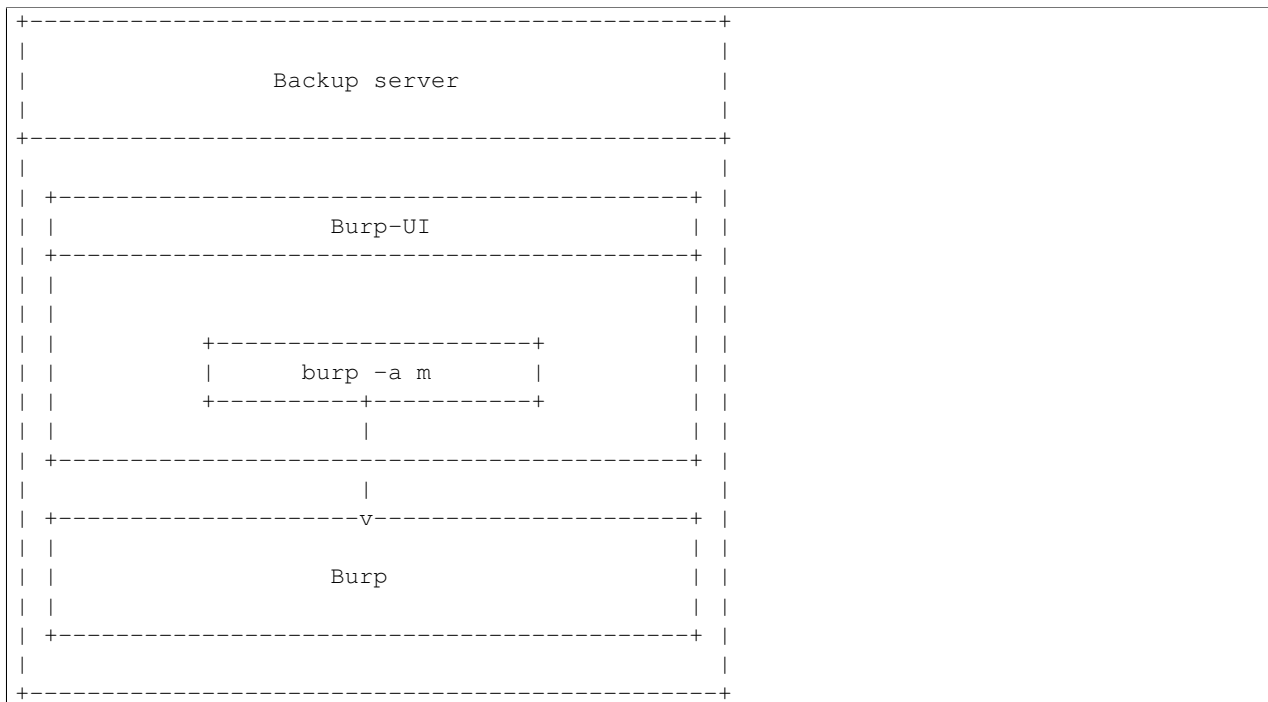
By default, a client will only be able to view its own reports/stats. If you want to be able to monitor other clients, you need to be added as a `restore_client` for those clients (this can be done by editing the client configuration file

in the *clientconfdir* directory on the server). Alternatively, you can add this option in the *burp-server.conf* file for this setting to be applied globally for all the clients.

Ok, now why did I tell you all this? Well, because the **Burp-UI**'s backend for **Burp** 2.x is basically just a wrapper around `burp -a m`.

It means that when you start **Burp-UI** with `version = 2`, the command `<burpbin> -c <bconfcli> -a m` (where `<burpbin>` defaults to `/usr/sbin/burp` and `<bconfcli>` defaults to `/etc/burp/burpui.cfg`). Of course this command will be ran with the same permissions and privileges as **Burp-UI** itself.

And here is a little illustration to summarize all this:



1.6 Usage

Burp-UI has been written with modularity in mind. The aim is to support **Burp** from the stable to the latest versions. **Burp** exists in two major versions: 1.x.x and 2.x.x.

Note: The version 2.x.x of **Burp** is currently in heavy development and should bring a lot of improvements, but also a lot of rework especially regarding the `status port` which is the main communication system between **Burp** and **Burp-UI**.

Both *Versions* are supported by **Burp-UI** thanks to its modular design. The consequence is you have various options in the configuration file to suite everybody needs.

There are also different modules to support *Authentication* and *ACL* within the web-interface.

Warning: **Burp-UI** tries to be as less intrusive as possible, nevertheless it ships with the ability to manage **Burp**'s configuration files. This feature **requires** **Burp-UI** to be launched on the **same** server that hosts your **Burp** instance. You also have to make sure the user that runs **Burp-UI** has **enough** privileges to edit those files.

1.6.1 Configuration

The `burpui.cfg` configuration file contains a `[Global]` section as follow:

```
[Global]
# burp server version 1 or 2
version = 2
# Handle multiple bui-servers or not
# If set to 'false', you will need to declare at least one 'Agent' section (see
# bellow)
standalone = true
# authentication plugin (mandatory)
# list the misc/auth directory to see the available backends
# to disable authentication you can set "auth = none"
# you can also chain multiple backends. Example: "auth = ldap,basic"
# the order will be respected unless you manually set a higher backend priority
auth = basic
# acl plugin
# list misc/acl directory to see the available backends
# default is no ACL
acl = basic
# You can change the prefix if you are behind a reverse-proxy under a custom
# root path. For example: /burpui
# You can also configure your reverse-proxy to announce the prefix through the
# 'X-Script-Name' header. In this case, the bellow prefix will be ignored in
# favour of the one announced by your reverse-proxy
prefix = none
```

Each option is commented, but here is a more detailed documentation:

- *version*: What version of **Burp** this **Burp-UI** instance manages. Can either be *1* or *2*. This parameter determines which backend is loaded at runtime.
(see [Versions](#) for more details)

- *standalone*: **Burp-UI** can run in two different modes. If it runs in standalone mode (meaning you set this parameter to *true*), you can only address **one** **Burp** server of the version specified by the previous parameter.

If this option is set to *false*, **Burp-UI** will run as a *proxy* allowing you to address multiple **Burp** servers. In this mode, you need to configure **at least one** *Agent* section in your configuration file. You also need to run one `bui-agent` per server.

(see [Modes](#) for more details)

- *auth*: What *Authentication* backend to use.
- *acl*: What *ACL* module to use.
- *prefix*: You can host **Burp-UI** behind a sub-root path. See the [unicorn](#) page for details.

There is also a `[UI]` section in which you can configure some *UI* parameters:

```
[UI]
# refresh interval of the pages in seconds
refresh = 180
# refresh interval of the live-monitoring page in seconds
liverefresh = 5
```

Each option is commented, but here is a more detailed documentation:

- *refresh*: Time in seconds between two refresh of the interface.
- *liverefresh*: Time in seconds between two refresh of the *live-monitor* page.

1.6.2 Production

The `burpui.cfg` configuration file contains a `[Production]` section as follow:

```
[Production]
# storage backend for session and cache
# may be either 'default' or 'redis'
storage = default
# session database to use
# may also be a backend url like: redis://localhost:6379/0
# if set to 'redis', the backend url defaults to:
# redis://<redis_host>:<redis_port>/0
# where <redis_host> is the host part, and <redis_port> is the port part of
# the below "redis" setting
session = default
# cache database to use
# may also be a backend url like: redis://localhost:6379/0
# if set to 'redis', the backend url defaults to:
# redis://<redis_host>:<redis_port>/1
# where <redis_host> is the host part, and <redis_port> is the port part of
# the below "redis" setting
cache = default
# redis server to connect to
redis = localhost:6379
# whether to use celery or not
# may also be a broker url like: redis://localhost:6379/0
# if set to "true", the broker url defaults to:
# redis://<redis_host>:<redis_port>/2
# where <redis_host> is the host part, and <redis_port> is the port part of
# the above "redis" setting
celery = false
# database url to store some persistent data
# none or a connect string supported by SQLAlchemy:
# http://docs.sqlalchemy.org/en/latest/core/engines.html#database-urls
# example: sqlite:///var/lib/burpui/store.db
database = none
```

1.6.3 Experimental

There is a `[Experimental]` section for features that have not been deeply tested:

```
[Experimental]
## This section contains some experimental features that have not been deeply
## tested yet
# enable zip64 feature. Python doc says:
# « ZIP64 extensions are disabled by default because the default zip and unzip
# commands on Unix (the InfoZIP utilities) don't support these extensions. »
zip64 = false
```

These options are also available in the `bui-agent` configuration file.

1.6.4 Security

The `[Security]` section contains options to harden the security of the application:


```
[Security]
## This section contains some security options. Make sure you understand the
## security implications before changing these.
# list of 'root' paths allowed when sourcing files in the configuration.
# Set this to 'none' if you don't want any restrictions, keeping in mind this
# can lead to accessing sensible files. Defaults to '/etc/burp'.
# Note: you can have several paths separated by commas.
# Example: /etc/burp,/etc/burp.d
includes = /etc/burp
# if files already included in config do not respect the above restriction, we
# prune them
enforce = false
# enable certificates revocation
revoke = false
# remember_cookie duration in days
cookietime = 14
# whether to use a secure cookie for https or not. If set to false, cookies
# won't have the 'secure' flag.
# This setting is only useful when HTTPS is detected
scookie = true
# application secret to secure cookies. If you don't set anything, the default
# value is 'random' which will generate a new secret after every restart of your
# application. You can also set it to 'none' although this is not recommended.
appsecret = random
```

Some of these options are also available in the `bui-agent` configuration file.

1.6.5 Modes

Burp-UI provides two modes:

- *Standalone*
- *Multi-Agent*

These modes allow you to either access a single [Burp](#) server or multiple [Burp](#) servers hosted on separated hosts.

Standalone

This mode is the **default** and the easiest one. It can be activated by setting the *standalone* parameter in the `[Global]` section of your `burpui.cfg` file to *true*:

```
[Global]
standalone = true
```

That's all you need to do for this mode to work.

Multi-Agent

This mode allows you access multiple [Burp](#) servers through the `bui-agent`. The architecture is available on the `bui-agent` page.

To enable this mode, you need to set the *standalone* parameter of the `[Global]` section of your `burpui.cfg` file to *false*:

```
[Global]
standalone = false
```

Once this mode is enabled, you have to create **one** [Agent] section **per** agent you want to connect to in your `burpui.cfg` file:

```
# If you set standalone to 'false', add at least one section like this per
# bui-agent
[Agent:agent1]
# bui-agent address
host = 192.168.1.1
# bui-agent port
port = 10000
# bui-agent password
password = azerty
# enable SSL
ssl = true

[Agent:agent2]
# bui-agent address
host = 192.168.2.1
# bui-agent port
port = 10000
# bui-agent password
password = ytreza
# enable SSL
ssl = true
```

Note: The sections must be called [Agent : <label>] (case sensitive)

To configure your agents, please refer to the [bui-agent](#) page.

1.6.6 Versions

Burp-UI ships with two different backends:

- *Burp1*
- *Burp2*

These backends allow you to either connect to a [Burp](#) server version 1.x.x or 2.x.x.

Note: If you are using a [Burp](#) server version 2.x.x you **have** to use the *Burp2* backend, no matter what [Burp](#)'s protocol you are using.

Burp1

Note: Make sure you have read and understood the requirements first.

The *burp-1* backend can be enabled by setting the *version* option to *1* in the [Global] section of your `burpui.cfg` file:

```
[Global]
version = 1
```

Now you can add *burp-1* backend specific options:

```
# burp1 backend specific options
[Burp1]
# burp status address (can only be '127.0.0.1' or '::1')
bhost = ::1
# burp status port
bport = 4972
# burp binary
burpbin = /usr/sbin/burp
# vss_strip binary
stripbin = /usr/sbin/vss_strip
# burp client configuration file used for the restoration (Default: None)
bconfcli = /etc/burp/burp.conf
# burp server configuration file used for the setting page
bconfsrv = /etc/burp/burp-server.conf
# temporary directory to use for restoration
tmpdir = /tmp
```

Each option is commented, but here is a more detailed documentation:

- *bhost*: The address of the [Burp](#) server. In burp-1.x.x, it can only be *127.0.0.1* or *::1*
- *bport*: The port of [Burp](#)'s status port.
- *burpbin*: Path to the [Burp](#) binary (used for restorations).
- *stripbin*: Path to the [Burp](#) *vss_strip* binary (used for restorations).
- *bconfcli*: Path to the [Burp](#) client configuration file (see restoration).
- *bconfsrv*: Path to the [Burp](#) server configuration file.
- *tmpdir*: Path to a temporary directory where to perform restorations.

Burp2

Note: Make sure you have read and understood the requirements first.

Note: The unicorn documentation may help you configuring your system.

The *burp-2* backend can be enabled by setting the *version* option to 2 in the [Global] section of your [burpui.cfg](#) file:

```
[Global]
version = 2
```

Now you can add *burp-2* backend specific options:

```
# burp2 backend specific options
[Burp2]
# burp binary
burpbin = /usr/sbin/burp
# vss_strip binary
```

```
stripbin = /usr/sbin/vss_strip
# burp client configuration file used for the restoration (Default: None)
bconfcli = /etc/burp/burp.conf
# burp server configuration file used for the setting page
bconfsrv = /etc/burp/burp-server.conf
# temporary directory to use for restoration
tmpdir = /tmp
# how many time to wait for the monitor to answer (in seconds)
timeout = 5
```

Each option is commented, but here is a more detailed documentation:

- *burpbin*: Path to the **Burp** binary (used for restorations).
- *stripbin*: Path to the **Burp** *vss_strip* binary (used for restorations).
- *bconfcli*: Path to the **Burp** client configuration file (see restoration).
- *bconfsrv*: Path to the **Burp** server configuration file.
- *tmpdir*: Path to a temporary directory where to perform restorations.
- *timeout*: Time to wait for the monitor to answer in seconds.

1.6.7 Authentication

Burp-UI provides some authentication backends in order to restrict access only to granted users. There are currently three different backends:

- *LDAP*
- *Basic*
- *Local*

To disable the *authentication* backend, set the *auth* option of the [Global] section of your *burpui.cfg* file to *none*:

```
[Global]
auth = none
```

You can use multiple backends, they will be sorted by priority or in the order they are defined if no priority is found. If a user is present in several backends, the first one that matches both login and password will be used.

Example:

```
[Global]
auth = basic,ldap
```

LDAP

The *ldap* authentication backend has some dependencies, please refer to the requirements page. To enable this backend, you need to set the *auth* option of the [Global] section of your *burpui.cfg* file to *ldap*:

```
[Global]
auth = ldap
```

Now you can add *ldap* specific options:

```
# ldapauth specific options
[LDAP]
# Backend priority. Higher is first
priority = 1
# LDAP host
host = 127.0.0.1
# LDAP port
port = 389
# Encryption type to LDAP server (none, ssl or tls)
# - try tls if unsure, otherwise ssl on port 636
encryption = tls
# specifies if the server certificate must be validated, values can be:
# - none (certificates are ignored)
# - optional (not required, but validated if provided)
# - required (required and validated)
validate = none
# SSL or TLS version to use, can be one of the following:
# - SSLv2
# - SSLv3
# - SSLv23
# - TLSv1
# - TLSv1_1 (Available only with openssl version 1.0.1+, requires python 2.7.9 or higher)
version = TLSv1
# the file containing the certificates of the certification authorities
cafile = none
# Attribute to use when searching the LDAP repository
#searchattr = sAMAccountName
searchattr = uid
# LDAP filter to find users in the LDAP repository
# - {0} will be replaced by the search attribute
# - {1} will be replaced by the login name
filter = (&({0}={1})(burpui=1))
#filter = (&({0}={1})(|(userAccountControl=512)(userAccountControl=66048)))
# LDAP base
base = "ou=users,dc=example,dc=com"
# Binddn to list existing users
binddn = "cn=admin,dc=example,dc=com"
# Bindpw to list existing users
bindpw = Sup3rS3cr3tPa$$w0rd
```

Note: The *host* options accepts URI style (ex: `ldap://127.0.0.1:389`)

Warning: The quotes (") around *base* and *binddn* are **MENDATORY**

Basic

In order for the *basic* authentication backend to be enabled, you need to set the *auth* option of the [Global] section of your `burpui.cfg` file to *basic*:

```
[Global]
auth = basic
```

Now you can add *basic* specific options:

```
# basicauth specific options
# Note: in case you leave this section commented, the default login/password
# is admin/admin
[BASIC]
# Backend priority. Higher is first
priority = 2
admin = pbkdf2:sha1:1000$12345678$password
user1 = pbkdf2:sha1:1000$87654321$otherpassword
```

Note: Each line defines a new user with the *key* as the username and the *value* as the password

Warning: Since v0.3.0, passwords must be hashed (see manage to know how to create new users with hashed passwords)

Local

In order for the *local* authentication backend to be enabled, you need to set the *auth* option of the [Global] section of your `burpui.cfg` file to *local*:

```
[Global]
auth = local
```

Now you can add *local* specific options:

```
# localauth specific options
# Note: if not running as root, then burp-ui must be run as group 'shadow' to
# allow PAM to work
[LOCAL]
# Backend priority. Higher is first
priority = 3
# List of local users allowed to login. If you don't set this setting, every
# local user will be able to login
users = user1,user2
```

1.6.8 ACL

Burp-UI implements some mechanisms to restrict access on some resources only for some users. There is currently only one backend:

- *Basic ACL*

To disable the *acl* backend, set the *acl* option of the [Global] section of your `burpui.cfg` file to *none*:

```
[Global]
acl = none
```

Basic ACL

The *basic* acl backend can be enabled by setting the *acl* option of the [Global] section of your `burpui.cfg` file to *basic*:

```
[Global]
acl = basic
```

Now you can add *basic acl* specific options:

```
# basicacl specific options
# Note: in case you leave this section commented, the user 'admin' will have
# access to all clients whereas other users will only see the client that have
# the same name
[BASIC:ACL]
admin = user1,user2
# Please note the double-quotes and single-quotes on the following lines are
# mandatory!
# You can also overwrite the default behavior by specifying which clients a
# user can access
user3 = '["client4", "client5"]'
# In case you are not in a standalone mode, you can also specify which clients
# a user can access on a specific Agent
user4 = '{"agent1": ["client6", "client7"], "agent2": ["client8"]}'
```

Warning: The double-quotes and single-quotes are **MENDATORY**

By default, if a user is named `admin` it will be granted the admin role. Here are the default grants:

1. `admin =>` you can do anything
2. `non admin =>` you can only see the client that matches your username
3. `custom =>` you can manually assign username to clients using the syntax `username = client1,client2`
or `username = '{"agent1": ["client1-1"], "agent2": ["client2-3", "client2-4"]}'` (if you are running a multi-agent setup)

1.7 Manage

Since *v0.3.0*, [Burp-UI](#) ships with a tool called `bui-manage`. This tool allows you to create new users and to manage database migrations.

This tool is actually a wrapper script that interacts with the core of [Burp-UI](#). You can use it like this:

```
bui-manage [wrapper options...] [--] <subcommand>
```

This page details the *subcommand* usage. The tool provides some inline help too:

```
bui-manage -h
usage: bui-manage [-h] [-c <CONFIG>] [-i <MIGRATIONS_DIR>]
                  [-m <agent|server|worker|manage>]
                  ...

positional arguments:
  remaining

optional arguments:
  -h, --help            show this help message and exit
  -c <CONFIG>, --config <CONFIG>
                        burp-ui configuration file
  -i <MIGRATIONS_DIR>, --migrations <MIGRATIONS_DIR>
                        migrations directory
  -m <agent|server|worker|manage>, --mode <agent|server|worker|manage>
                        application mode
```

```
# note the -- used to separate the wrapper from the actual command
bui-manage -- --help
Usage: flask [OPTIONS] COMMAND [ARGS]...

This shell command acts as general utility script for Flask applications.

It loads the application configured (either through the FLASK_APP
environment variable) and then provides commands either provided by the
application or Flask itself.

The most useful commands are the "run" and "shell" command.

Example usage:

$ export FLASK_APP=hello
$ export FLASK_DEBUG=1
$ flask run

Options:
--help  Show this message and exit.

Commands:
compile_translation  Compile translations.
create_user          Create a new user.
db                  Perform database migrations.
init_translation     Initialize a new translation for the given...
run                  Runs a development server.
setup_burp           Setup burp client for burp-ui.
shell                Runs a shell in the app context.
update_translation   Update translation files.
```

1.7.1 Database

To manage database migration, you first need to enable database support within your configuration file (see [Production section](#))

You will also need some extra requirements:

```
pip install "burp-ui[sql]"
```

Then you just have to run the following command to have your database setup:

```
bui-manage db upgrade
```

If your configuration is not in a *common* location, you can specify it like this:

```
bui-manage -c path/to/burpui.cfg db upgrade
```

If you did not install [Burp-UI](#) in a *common* location or you want to run it without installing it directly through the sources, you may need to specify the location of the *migrations* scripts like this:

```
bui-manage -c path/to/burpui.cfg -i path/to/migrations db upgrade
```

1.7.2 Users

You can create new users using the `bui-manage` file like this:


```
bui-manage create_user <new_username>
```

By default, the script will create new users for the Basic authentication backend. Without further details, a new password will be generated. You can either provide a password through the command line or tell the script to ask you what to setup using either the `-p` or `-a` options.

Examples:

```
bui-manage create_user user1
[*] Adding 'user1' user...
[+] Generated password: 7lVIanuJ
[+] Success: True

bui-manage create_user -p toto user2
[*] Adding 'user2' user...
[+] Success: True

bui-manage create_user -a user3
[*] Adding 'user3' user...
Password:
Confirm:
[+] Success: True
```

1.7.3 Configure

Since *v0.4.0*, the `bui-manage` tool is now able to help you setup both [Burp](#) and [Burp-UI](#) so they speak to each other.

The available options are:

```
bui-manage setup_burp --help
Usage: flask setup_burp [OPTIONS]

    Setup burp client for burp-ui.

Options:
  -b, --burp-conf-cli TEXT    Burp client configuration file
  -s, --burp-conf-serv TEXT   Burp server configuration file
  -c, --client TEXT           Name of the burp client that will be used by
                              Burp-UI (defaults to "bui")
  -h, --host TEXT             Address of the status server (defaults to ":::1")
  -r, --redis TEXT            Redis URL to connect to
  -d, --database TEXT         Database to connect to for persistent storage
  -n, --dry                   Dry mode. Do not edit the files but display
                              changes
  --help                      Show this message and exit.
```

The script needs the [Burp](#) configuration files to be readable **AND** writable.

Note: This script was initially developped to setup the docker image. I do not guarantee to be able to support it out of the docker context.

Note: This script only supports Burp 2.0.x.

The docker image uses this script like this:

```
bui-manage -c $BURPUI_CONFIG setup_burp -b $BURP_CLIENT_CONFIG \  
-s $BURP_SERVER_CONFIG -h $BURP_SERVER_ADDR -c $BURPUI_CLIENT_NAME \  
-r $REDIS_SERVER -d $DATABASE_URL
```

1.8 Celery

Since v0.3.0, **Burp-UI** supports asynchronous operations thanks to **Celery**. In order to use this feature, you need to enable it in the configuration (see Production section)

You will also need some extra requirements:

```
pip install "burp-ui[celery]"
```

Celery needs a *Broker* to communicate between the workers and your application. I chose **Redis** so you will need a working **Redis** server (Basically you just need to run `apt-get install redis-server` on Debian based distributions)

1.8.1 Runner

Once everything is setup, you need to launch a worker. **Burp-UI** ships with a helper script called `bui-celery`. You can use it like this:

```
bui-celery --beat
```

If your configuration is not in a *common* location, you can specify it like this:

```
bui-celery -c path/to/burpui.cfg -- --beat
```

Note: A systemd service example file is shipped in the *contrib* directory

Note: The `--beat` option is recommended since some operations need to be executed periodically

Note: The usage of a database is recommended to keep a track of executed tasks

1.9 Gunicorn

Starting from v0.0.6, **Burp-UI** supports **Gunicorn** in order to handle multiple users simultaneously because some operations (like the online restoration) may take some time and thus may block any further requests. With **Gunicorn**, you have several workers that can proceed the requests so you can handle more users.

You need to install gunicorn and gevent:

```
pip install "burp-ui[gunicorn]"
```

You will then be able to launch **Burp-UI** this way:

```
gunicorn -k gevent -w 4 'burpui:create_app(conf="/path/to/burpui.cfg")'
```

When using `gunicorn`, the command line options are not available. Instead, run the `Burp-UI create_app` method directly. Here are the parameters you can play with:

- `conf`: Path to the `Burp-UI` configuration file
- `verbose`: Verbosity level between 0 and 4
- `logfile`: Path to a logfile in order to log `Burp-UI` internal messages

Warning: You **MUST** set the `appsecret` option in your configuration file when using `gunicorn`. The default *magic* value 'random' cannot be used. If you don't change the settings the default value will be 'none' and your cookies won't be secured.

1.9.1 Advanced usage

`Gunicorn` supports further settings (see its [documentation](#) for details). For instance, you would probably like to use the `-c` flag with the sample configuration file bundled with `Burp-UI` in `contrib/gunicorn/burpui_config.py`.

Usage example:

```
gunicorn -c burpui_config.py 'burpui:create_app(conf="/etc/burp/burpui.cfg",logfile="/var/log/gunicorn.log")'
```

1.9.2 Daemon

If you wish to run `Burp-UI` as a daemon process, the recommended way is to use `Gunicorn`.

When installing the `gunicorn` package on `debian`, there is a handler script that is able to start several instances of `Gunicorn` as daemons.

All you need to do is installing the `gunicorn` package and adding a configuration file in `/etc/gunicorn.d/`.

There is a sample configuration file available [here](#).

If you are using this sample configuration file, make sure to create the `burpui` user with the appropriate permissions first:

```
# install the gunicorn package
apt-get install gunicorn
# copy the gunicorn sample configuration
cp /usr/local/share/burpui/contrib/gunicorn.d/burp-ui /etc/gunicorn.d/
# create the burpui user
useradd -m -r -d /var/lib/burpui -c 'Burp-UI daemon user' burpui
mkdir /etc/burp
# copy the burp-ui sample configuration file
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
mkdir -p /var/log/gunicorn
chown -R burpui: /var/log/gunicorn
```

You will also need a custom client configuration and you will have to create the certificates accordingly:

```
# create the configuration file used by burp-ui
cat >/var/lib/burpui/burp.conf<<EOF
mode = client
port = 4971
status_port = 4972
server = 127.0.0.1
password = abcdefgh
cname = bui-agent1
```

```
pidfile = /var/lib/burpui/bui-agent1.client.pid
syslog = 0
stdout = 1
progress_counter = 1
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /var/lib/burpui/CA-client
ssl_cert_ca = /var/lib/burpui/ssl_cert_ca-client.pem
ssl_cert = /var/lib/burpui/ssl_cert-client.pem
ssl_key = /var/lib/burpui/ssl_cert-client.key
ssl_peer_cn = burpserver
EOF
# generate the certificates
burp_ca --name bui-agent1 --ca burpCA --key --request --sign --batch
cp /etc/burp/ssl_cert_ca.pem /var/lib/burpui/ssl_cert_ca-client.pem
cp -a /etc/burp/CA/bui-agent1.crt /var/lib/burpui/ssl_cert-client.pem
cp -a /etc/burp/CA/bui-agent1.key /var/lib/burpui/ssl_cert-client.key
chown -R burpui: /var/lib/burpui/
```

Now you need to add the *bui-agent1* client to the authorized clients:

```
echo "password = abcdefgh" >/etc/burp/clientconfdir/bui-agent1
echo "restore_client = bui-agent1" >>/etc/burp/burp-server.conf
```

Finally, make sure you set `bconfcli: /var/lib/burpui/burp.conf` in your **Burp-UI** configuration file (`/etc/burp/burpui.cfg`), and then you can restart **Gunicorn**:

```
service gunicorn restart
```

If you want to take advantage of *advanced* features such as client add/removal and configuration files edition, you should set the permissions accordingly Burp-side.

First of all, add the following lines in your `/etc/burp/burp-server.conf`:

```
user = burpui
group = burpui
```

Then you need to fix some permissions:

```
chown -R burpui: /etc/burp/{burp-server.conf,burpui.cfg,CA,CA.cnf,clientconfdir,dhfile.pem,ssl_cert_
chgrp burpui /etc/burp
chmod g+rw /etc/burp
```

Finally you can restart your `burp-server`.

Note: The above commands are meant for *default* setup. You may need to adapt the paths.

1.9.3 Reverse-Proxy

You may want to add a reverse-proxy so **Burp-UI** can be accessed on port 80 (or 443) along with other applications.

Here is a sample configuration for Nginx:

```
server {
    listen 80;
    server_name burpui.example.com;

    access_log /var/log/nginx/burpui.access.log;
```

```

error_log    /var/log/nginx/burpui.error.log;

location / {

    # you need to change this to "https", if you set "ssl" directive to "on"
    proxy_set_header    X-FORWARDED_PROTO http;
    proxy_set_header    Host                $http_host;
    proxy_set_header    X-Forwarded-For     $remote_addr;

    proxy_read_timeout  300;
    proxy_connect_timeout 300;

    proxy_pass http://localhost:5000;
}

```

Sub-root path

You can host [Burp-UI](#) behind a sub-root path. For instance `/burpui`. To accomplish this, you can either setup your reverse-proxy to announce the desired *prefix*, or you can use the `prefix` option in your [Burp-UI](#) configuration file (see usage for details).

If you want to configure this reverse-proxy side, you need to announce the HTTP Header `X-Script-Name`.

Here is a sample configuration for Nginx:

```

server {
    listen 80;
    server_name example.com;

    access_log    /var/log/nginx/burpui.access.log;
    error_log     /var/log/nginx/burpui.error.log;

    location /burpui {

        # you need to change this to "https", if you set "ssl" directive to "on"
        proxy_set_header    X-FORWARDED_PROTO http;
        proxy_set_header    Host                $http_host;
        proxy_set_header    X-Forwarded-For     $remote_addr;
        # Our service is hosted behind the "/burpui" prefix
        proxy_set_header    X-Script-Name       /burpui;

        proxy_read_timeout  300;
        proxy_connect_timeout 300;

        proxy_pass http://localhost:5000;
    }
}

```

Apache sample:

```

ProxyPass /burp/ http://localhost:5000/burp/
ProxyPassReverse /burp/ http://localhost:5000/burp/
<Location /burp/>
    SetOutputFilter proxy-html
    ProxyPassReverse /burp/
    ProxyHTMLURLMap http://localhost:5000/ /

```

```
Require all granted
</Location>
```

Warning: If your *prefix* does not start with a *'/'*, it will be ignored.

1.9.4 Production

We can consider the [demo](#) as a production example of what you can setup/expect in your environment. It is using [Gunicorn](#) along with Nginx as described above.

In order to improve performances, [Redis](#) can be used to cache sessions and various API calls.

See the production section of the usage page.

1.10 Docker

Since the *v0.4.0*, a docker image is provided. It ships with the latest stable release of [Burp-UI](#) and supports the celery worker introduced in *v0.3.0* if you link it to a redis container.

1.10.1 Introduction

All you need is [docker](#) and [docker-compose](#). A *docker-compose.yml* file is provided. There are a few variables supported to setup your system:

- **BURPUI_CONFIG** - Specify where the [Burp-UI](#) configuration file is located. It defaults to *"/etc/burp/burpui.cfg"*.
- **BURPUI_VERBOSE** - Specify the log verbosity (between 0 and 4). It defaults to 0.
- **BURPUI_CLIENT_NAME** - Specify the name of the burp client that will be used by [Burp-UI](#). It defaults to *"bui"*.
- **BURPUI_UID** - uid of the *burpui* user you want to map in your host. It defaults to 5337.
- **BURPUI_GID** - gid of the *burpui* group you want to map in your host. It defaults to 5337.
- **BURP_CLIENT_CONFIG** - Specify the path of the burp client configuration file to use for the [Burp-UI](#) client. It defaults to *"/tmp/burp.conf"*. It means you won't have access to it outside of the container. It is intended to not override the */etc/burp/burp.conf* file if you already use it.
- **BURP_SERVER_CONFIG** - Specify the path of the burp-server configuration file. It defaults to *"/etc/burp/burp-server.conf"*.
- **DATABASE_URL** - Specify the URL of the database to connect to. It defaults to *"sqlite:///var/lib/burpui/store.db"*.
- **REDIS_SERVER** - Specify the address of the redis server. It defaults to *"redis:6379"*.
- **BURP_SERVER_ADDR** - Specify the address of the burp-server status port. If set to *"auto"* (the default), we will use the address of the docker host. Make sure your status port is listening on this interface.

The provided *docker-compose.yml* file suggests that you *mount* the */etc/burp* and */var/spool/burp* paths inside the container (this is automatic) so that [Burp-UI](#) is able to access some required files.

1.10.2 Requirements

The docker image only works with a burp server version 2.0.x.

Burp-UI will be launched with the user *burpui* inside the container. This user has the UID `$BURPUI_UID` and the GID `$BURPUI_GID` so you may want to create a user with the associated UID/GID in your host and make sure it has read **AND** write access to `/etc/burp` and `/var/spool/burp`.

For instance:

```
groupadd -g 5337 burpui
useradd -r -m -d /var/lib/burpui -c 'Burp-UI daemon user' -u 5337 -g 5337 burpui
```

1.10.3 Usage

All you have to do is to retrieve the *docker-compose.yml* file, edit the variables if needed and launch the docker containers. For instance you could do:

```
mkdir -p ~/workspace
cd ~/workspace
git clone https://git.ziirish.me/ziirish/burp-ui.git
cd burp-ui
docker-compose up -d
```

That's it. Really. Now profit and go to <http://localhost:5000/>

1.10.4 Troubleshooting

Here are some hints to help you troubleshoot your *Burp-UI* container.

Cannot launch burp process: Unable to spawn burp process

This means *Burp-UI* was not able to spawn a burp client that is able to communicate with the server. You can check the containers logs using the `docker-compose logs` command. If the output contains something like:

```
It looks like your burp server is not exposing it's status port in a way that is reachable by Burp-UI
You may want to set the 'status_address' setting with either '1.2.3.4', ':::' or '0.0.0.0' in the /etc
```

It means your burp-server is not exposing its status port. The above output gives you the instructions to fix it.

Note: You'll have to restart your burp-server to bind to the new *status_address*

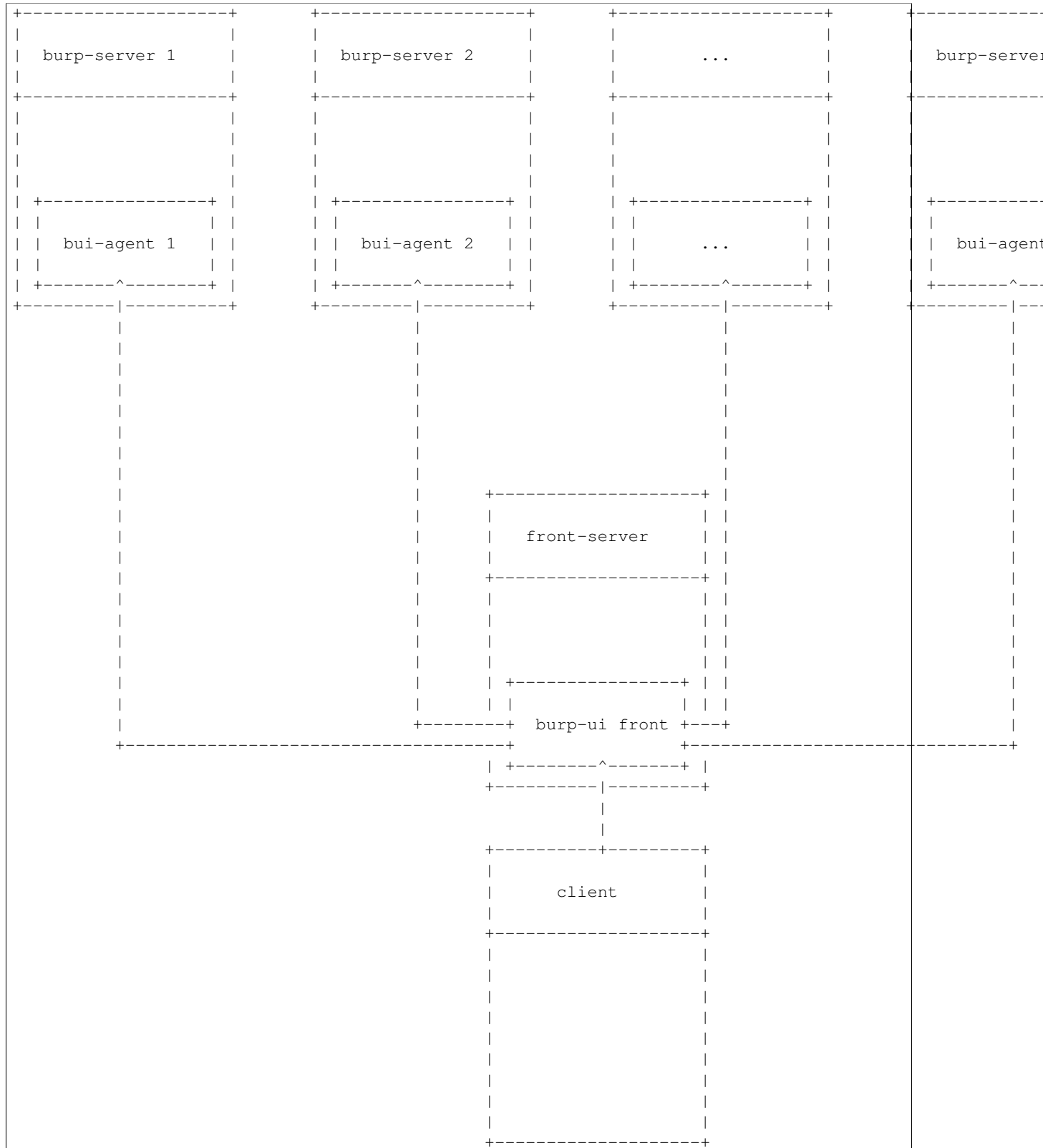
1.11 bui-agent

The bui-agent is a kind of proxy between a *Burp* server and your *Burp-UI* server.

It is useful when you have several servers to monitor and/or when you don't want (or can't) install the full *Burp-UI* on your server.

1.11.1 Architecture

The architecture is described below:



1.11.2 Requirements

The agent is powered by gevent. In order to install it, you can run the following command:

```
pip install "burp-ui[agent]"
```

1.11.3 Configuration

These agents must be launched on every server hosting a [Burp](#) instance you'd like to monitor.

They have a specific `buiagent.cfg` configuration file with a `[Global]` section as below:

```
[Global]
# On which port is the application listening
port = 10000
# On which address is the application listening
# ':::' is the default for all IPv6
# set it to '0.0.0.0' if you want to listen on all IPv4 addresses
bind = ::
# enable SSL
ssl = true
# ssl cert
sslcrt = /etc/burp/ssl_cert-server.pem
# ssl key
sslkey = /etc/burp/ssl_cert-server.key
# burp server version 1 or 2
version = 1
# agent password
password = password
```

Each option is commented, but here is a more detailed documentation:

- *port*: On which port is bui-agent listening.
- *bind*: On which address is bui-agent listening.
- *ssl*: Whether to communicate with the [Burp-UI](#) server over SSL or not.
- *sslcrt*: What SSL certificate to use when SSL is enabled.
- *sslkey*: What SSL key to use when SSL is enabled.
- *version*: What version of [Burp](#) this bui-agent instance manages. (see [Burp-UI versions](#) for more details)
- *password*: The shared secret between the [Burp-UI](#) server and bui-agent.

As with [Burp-UI](#), you need a specific section depending on the *version* value. Please refer to the [Burp-UI versions](#) section for more details.

1.11.4 Daemon

I have no plan to implement daemon features, but there are a lot of tools available to help you achieve such a behavior.

For instance, you can create a systemd service file containing:

```
[Unit]
Description=Burp-UI agent service
After=network.target

[Service]
```

```
ExecStart=/usr/local/bin/bui-agent
User=burpui
```

You can also have a look at how the demo works (it uses supervisor)

1.11.5 Example

Here is a full usage example:

```
# On the server called 'agent1'
agent1:~$ bui-agent -c path/to/buiagent.cfg

# On the server called 'agent2'
agent2:~$ bui-agent -c path/to/buiagent.cfg

# On the server called 'front'
front:~$ burp-ui -c path/to/burpui.cfg
```

This example uses three servers. You then only need to point your browser to <http://front:5000/> for instance, and the **Burp-UI** instance (front) will *proxify* the requests to the two agents for you.

1.12 Contributing

Contributions are welcome. You can help in any way you want, for instance by opening issues on the [bug tracker](#), sending patches, etc.

There is also a dedicated website. Currently it only hosts a [Discourse](#) instance where you can discuss with each other. No need to create another account, the one you use in the [bug tracker](#) can be imported automatically!

Feel free to use it and post your tips and remarks.

The address is: <https://burpui.ziirish.me/>

You can financially support the project if you find it useful or if you would like to sponsor a feature. Details on my [website](#).

1.12.1 Translating

Translations are very welcome! If you are willing to help, you will need some tools:

```
pip install Flask-Babel
```

Then you need to fork the project retrieve the sources:

```
git clone https://git.ziirish.me/<your_login>/burp-ui.git
cd burp-ui
```

You can have the list of available languages by running:

```
ls burpui/translations
```

New language

If your language is not listed, you can create a new translation running the following command:

```
./bui-manage init_translation <country_code> # where <country_code> can be "de", "ru", etc.
```

Update translation

If you want to update an existing (and/or un-complete) translation, you probably want to have a look at the *templates* files.

An un-translated file will contain things like:

```
<h1>Some title</h1>
```

The string *Some title* won't be translated as is. You need to update the template like this:

```
<h1>{{ _('Some title') }}</h1>
```

Then you can update the translation files with the following command:

```
./bui-manage update_translation
```

Translation

Now you can proceed the translation in the file *burpui/translations/<country_code>/LC_MESSAGES/messages.po*.

It looks like:

```
#: burpui/forms.py:18 burpui/templates/login.html:8
msgid "Username"
msgstr ""
```

You just have to put the translations in the *msgstr* line like:

```
#: burpui/forms.py:18 burpui/templates/login.html:8
msgid "Username"
msgstr "Utilisateur"
```

Once it's done, you can push the sources and create a Merge Request on GitLab:

```
git checkout -b translation-<country_code>
git add burpui/translations/<country_code>/LC_MESSAGES/messages.po
git commit -m "<country_code> translation"
git push -u origin translation-<country_code>
```

1.12.2 Issues / Bugs

If you find any issue while using Burp-UI please report it on the [bug tracker](#). All issues should contain the used command line to reproduce the problem, the debug output and both versions of burp and Burp-UI you are using.

You can get those informations using the following commands:

```
$ /usr/sbin/burp -v
burp-1.4.40
$ burp-ui -V -v
burp-ui: v0.1.0.dev (90deb82c7b0be35f1a70bb073c9926b5947c6a85)
$ burp-ui -v
```

Optionally your python version and your OS might be useful as well.

1.12.3 Questions

Ask questions in the [discussion forum](#). Do not use the issue tracker for this purpose.

Burp-UI has extensive online documentation please read the [doc](#).

1.12.4 Troubleshooting

In case you encounter troubles with Burp-UI, you should run it with the `-vvvv` flag and paste the relevant output within your bug-report. Please also give the version of burp **AND** Burp-UI. Since v0.0.6 you can use the `-V` or `--version` flag in order to get your version number.

1.12.5 Merge / Pull requests

I would like you to use [gitlab](#) for your Merge requests in order to take advantage of the automated tests I have been working on. You can login/register on my personal gitlab server with your github account.

1.12.6 Development

You will find any development information on the developer guide page.

1.13 Changelog

1.13.1 0.4.2 (12/16/2016)

- Fix: bui-agent was broken
- Fix: handle i18n exceptions
- Fix: enable db migration only when needed
- Fix: wrong escape in translation

1.13.2 0.4.1 (12/15/2016)

- **BREAKING:** Use the new Flask's embedded server by default means no more SSL (HTTPS) support without a dedicated application server
- Fix: [issue #156](#)
- Fix: [issue #157](#)
- Fix: [issue #165](#)
- Fix: [issue #176](#)
- Fix: [issue #181](#)
- Fix: [issue #182](#)
- Various bugfix
- [Full changelog](#)

1.13.3 0.4.0 (11/23/2016)

- **BREAKING:** The database schema evolved. In order to apply these modifications, you **MUST** run the `bui-manage db upgrade` command after upgrading
- **BREAKING:** Plain text passwords are deprecated since v0.3.0 and are now disabled by default
- **BREAKING:** The default *version* setting has been set to 2 instead of 1
- Add: new `bui-manage setup_burp` command
- Add: new `docker` image
- Add: manage user sessions
- Add: `French` translation
- Fix: issue #151
- Fix: issue #154
- Fix: issue #158
- Fix: issue #163
- Fix: issue #164
- Fix: issue #166
- Fix: issue #169
- Fix: issue #171
- Fix: issue #172
- Fix: issue #173
- Fix: issue #174
- Various bugfix
- `Full changelog`

1.13.4 0.3.0 (08/15/2016)

- **BREAKING:** New configuration file format to allow further improvements (The conversion is automatic, but LDAP settings might need some attention)
- **BREAKING:** Passwords are now *salted* for the *BASIC* authentication backend (The conversion is automatic too)
- **BREAKING:** If you plan to use the SQL storage along with gunicorn, you **MUST** add the `-preload` parameter (see the `gunicorn.d/burp-ui` file)
- Add: `Celery` support for asynchronous tasks
- Add: `SQLAlchemy` support for persistent storage
- Add: `RESTful` restore
- Add: `autoreload` config
- Add: remember some user settings
- Add: `client certificate` revocation
- Add: new `local` authentication backend

- Add: new filters on history API call
- Add: implement backend `keepalive`
- Add: allow to *disable* server-initiated restoration
- Fix: disable Basic-Auth login from UI to prevent some bugs with sessions
- Fix: issue #134
- Fix: issue #135
- Fix: issue #137
- Fix: issue #138
- Fix: issue #145
- Fix: issue #148
- Improvement: new asynchronous `backup-running` API call
- Security: restrict files that can be sent by the agent
- [Full changelog](#)

1.13.5 0.2.1 (05/17/2016)

- Add: allow to edit a server-initiated restoration
- Add: allow to cancel a server-initiated restoration
- Add: support for Burp labels
- Add: `server-initiated` backups
- Add: support sub-root path
- Add: new Burp 2 settings
- Improvement: better logging system
- Improvement: new security options
- Fix: issue #109
- Fix: issue #113
- Fix: issue #114
- Fix: issue #117
- Fix: issue #123
- Doc
- [Full changelog](#)

1.13.6 0.1.3 (02/20/2016)

- Fix: issue #107
- Fix: issue #108

1.13.7 0.1.2 (02/18/2016)

- Fix: duration computation
- Fix: issue #104
- Fix: issue #105
- Fix: issue #106

1.13.8 0.1.1 (02/17/2016)

- Fix: burp2 backend issue
- Fix: Debian wheezy compatibility
- Fix: sample configuration files location
- Better calendar readability

1.13.9 0.1.0 (02/15/2016)

- Add: [python 3](#) support
- Add: new fields in [backup](#) reports
- Add: [server-side](#) initiated restoration
- Add: percent done in [overview](#)
- Add: ability to [chain multiple](#) authentication backends
- Add: display versions [within the interface](#)
- Add: support for [zip64](#)
- Add: new [report](#)
- Add: new [calendar view](#)
- Add: “restart” option to debian init script thanks to @Larsen
- Add: Basic HTTP Authentication (mostly for the API)
- Add: self-documented API
- Fix: issue #81
- Fix: issue #87
- Fix: issue #88
- Fix: issue #92
- Fix: issue #95
- Fix: issue #99
- Fix: issue #100
- Fix: issue #101
- [demo](#)
- API refactoring

- Security fixes
- Bugfixes

1.13.10 0.0.7.3 (09/26/2015)

- Fix: issue #77
- Doc

1.13.11 0.0.7.2 (09/01/2015)

- Fix: issue #73
- Fix: issue #74
- Doc

1.13.12 0.0.7.1 (08/22/2015)

- Add: Burp-2 backend
- Add: sortable tables
- Add: ACL support
- Add: support client-side encrypted backups while performing an online restoration
- Add: multiple archive format
- Add: better Active Directory support
- Improvement: better config file parser
- Improvement: better logging with Unicorn
- Improvement: full support of server configuration file + clientconfdir
- Fix: issue #35
- Fix: issue #37
- Fix: issue #41
- Fix: issue #42
- Fix: issue #46
- Fix: issue #49
- Fix: issue #53
- Fix: issue #54
- Fix: issue #59
- Fix: issue #62
- Fix: issue #68
- Fix: issue #69
- Fix: issue #70

- Fix: issue #71
- Fix: issue #72
- doc on [readthedocs](#)
- Two merge requests from Wade Fitzpatrick (!1 and !2)
- API refactoring
- Security fixes
- Bugfixes
- [Full changelog](#)

1.13.13 0.0.6 (12/15/2014)

- Add: [gunicorn](#) support
- Add: init script for CentOS
- Add: init script for Debian
- Add: autofocus login field on login page
- Add: [burp-server](#) configuration panel
- Fix: issue #25
- Fix: issue #26
- Fix: issue #30
- Fix: issue #32
- Fix: issue #33
- Fix: issue #34
- Fix: issue #35
- Fix: issue #39
- Code cleanup
- Improve unit tests
- Bugfixes
- [Full changelog](#)

1.13.14 0.0.5 (09/22/2014)

- Add: multi-server support
- Fix bugs
- [Full changelog](#)

1.13.15 0.0.4 (09/07/2014)

- Add: ability to download files directly from the web interface
- [Full changelog](#)

1.13.16 0.0.3 (09/02/2014)

- Add: authentication
- [Full changelog](#)

1.13.17 0.0.2 (08/25/2014)

- Fix bugs
- [Full changelog](#)

1.13.18 0.0.1 (08/25/2014)

- Initial release

1.14 FAQ

1.14.1 Is there a demo somewhere?

Yes, you can play with [Burp-UI](#) at [demo.ziirish.me](#). Credentials are:

- *admin / admin* to play with [Burp-UI](#) as an administrator
- *demo / demo* to play with [Burp-UI](#) as a regular user

1.14.2 How to start using Burp-UI?

You may find all the basic informations to get started with [Burp-UI](#) in the [README](#) file. You can also read the step-by-step page to get started.

1.14.3 How to configure my *firewall*?

When running [Burp-UI](#) in standalone mode, the embedded webserver listens on port **5000** on all interfaces.

The [Burp-UI](#) agents listen on port **10000** by default.

Of course those are configurable.

1.14.4 What are the default credentials?

The default login / password is *admin / admin* with the basic authentication backend.

1.14.5 How does the online restoration feature work?

The online restoration feature works the same way as if you were running the burp client yourself. It means Burp-UI runs the following command:

```
burp -a r -b <number> -C <client name> -r <regex> -d /tmp/XXX -c <bconfcli>
```

It then generates an archive based on the restored files.

Because of this workflow, and especially the use of the `-C` flag you need to tell your burp-server the client used by Burp-UI can perform a restoration for a different client. You can refer to the restoration section of this documentation along with the version section for more details.

1.14.6 What does the server-initiated restoration feature do and how to make it work?

This feature asks the server to perform a restoration on the client the next time it sees it.

In order for this feature to work, your client **MUST** allows the server to do that. You have to set `server_can_restore = 1` (which is the default value) in your client configuration file (usually `/etc/burp/burp.conf`).

1.14.7 How can I start Burp-UI as a daemon?

There are several *init scripts* provided by some users available [here](#).

The recommended way to run Burp-UI in production is to use [Gunicorn](#). You can refer to the gunicorn section of this documentation for more details.

1.14.8 How to setup a reverse-proxy in front of Burp-UI?

The only way to run Burp-UI behind a reverse-proxy is to use [Gunicorn](#). You can refer to the gunicorn section of this documentation for more details.

1.14.9 Why don't I see all my clients using the burp-2 backend?

Starting with burp 2, you cannot see all the client through the status port unless you tell burp a particular client can see other clients statistics. See the general instructions for more details.

1.14.10 Are there any known issues?

There is a known issue section in this documentation.

1.14.11 Burp-UI does not work anymore since I upgraded it, what can I do?

Make sure you read the upgrading page in case some breaking changes occurred.

1.14.12 How can I contribute?

You can refer to the contributing section of this documentation.

1.15 Step By Step

Although **Burp-UI** tries to make **Burp** accessible to everyone, both products have their complexity.

In this *Step by Step*, I would like to introduce you different use-cases with their associated configurations, descriptions and comments. In every case, we will consider neither **Burp** or **Burp-UI** are installed and describe the steps to setup your server from Scratch.

Note: Again, this part of the doc is mostly debian-centric. If some users are willing to adapt these examples with other distros I would be very thankful.

1. *Burp1 server* with **Burp-UI**
2. *Burp2 server* with **Burp-UI**
3. *Multiple servers* with **bui-agents**

1.15.1 Burp1 server

In this scenario, we are going to install a **Burp** server version 1.4.40 which is the current stable version. We assume you are using the user *root* to run the following commands.

We begin with the installation of **Burp** itself.

First, we need some system requirements in order to compile **Burp** and to install **Burp-UI**:

```
apt-get update
apt-get install uthash-dev g++ make libssl-dev librsync-dev python2.7-dev \
git python-pip libffi-dev
```

Now we retrieve the **Burp** sources and then we compile and install it:

```
cd /usr/src
git clone https://github.com/grke/burp.git
cd burp
git checkout tags/1.4.40
./configure --disable-ipv6
make
make install
# we also install init scripts
cp debian/init /etc/init.d/burp
cat >/etc/default/burp<<EOF
RUN="yes"
DAEMON_ARGS="-c /etc/burp/burp-server.conf"
EOF
chmod +x /etc/init.d/burp
update-rc.d burp defaults
```

It is now time to install **Burp-UI**:

```
pip install --upgrade burp-ui
```

Now that everything is installed, let's configure our tools!

In order to perform online restorations, **Burp-UI** relies on a classical **Burp** client.

We need to define our client, and we also need to allow it to perform restorations for other clients. We will set it up globally. Our client will be named *bui*:

```
# burp-ui client's definition
cat >/etc/burp/clientconffdir/bui<<EOF
password = abcdefgh
EOF

# grant our client to perform restorations for others
echo "restore_client = bui" >>/etc/burp/burp-server.conf

# now we generate ou client configuration
cat >/etc/burp/burp.conf<<EOF
mode = client
port = 4971
server = 127.0.0.1
password = abcdefgh
cname = bui
pidfile = /var/run/burp.bui.pid
syslog = 0
stdout = 1
progress_counter = 1
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /etc/burp/CA-client
# SSL certificate authority - same file on both server and client
ssl_cert_ca = /etc/burp/ssl_cert_ca-client.pem
# Client SSL certificate
ssl_cert = /etc/burp/ssl_cert-client.pem
# Client SSL key
ssl_key = /etc/burp/ssl_cert-client.key
# SSL key password
ssl_key_password = password
# Common name in the certificate that the server gives us
ssl_peer_cn = burpserver
# The following options specify exactly what to backup.
include = /home
EOF
```

Our **Burp** server is now set up, we can start it:

```
/etc/init.d/burp start
```

Now we can configure **Burp-UI**. The package comes with a default configuration and init scripts. We copy them at the right place:

```
cp /usr/local/share/burpui/contrib/debian/init.sh /etc/init.d/burp-ui
chmod +x /etc/init.d/burp-ui
update-rc.d burp-ui defaults
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
```

The default configuration is plug and play for this case, we just have to start **Burp-UI**:

```
/etc/init.d/burp-ui start
```

Your server is now fully set-up, you can access **Burp-UI** by pointing your browser to: http://server_ip:5000/

The default user / password is: admin / admin

For further customization, you can refer to the usage page of this documentation.

1.15.2 Burp2 server

In this scenario, we are going to install a [Burp](#) server version 2.0.28. We assume you are using the user *root* to run the following commands.

We begin with the installation of [Burp](#) itself.

First, we need some system requirements in order to compile [Burp](#) and to install [Burp-UI](#):

```
apt-get update
apt-get install uthash-dev g++ make libssl-dev librsync-dev python2.7-dev \
git python-pip libffi-dev libyajl-dev libz-dev
```

Now we retrieve the [Burp](#) sources and then we compile and install it:

```
cd /usr/src
git clone https://github.com/grke/burp.git
cd burp
git checkout tags/2.0.28
./configure
make
make install
# we also install init scripts
cp debian/init /etc/init.d/burp
cat >/etc/default/burp<<EOF
RUN="yes"
DAEMON_ARGS="-c /etc/burp/burp-server.conf"
EOF
chmod +x /etc/init.d/burp
update-rc.d burp defaults
```

It is now time to install [Burp-UI](#):

```
pip install --upgrade burp-ui
```

Now that everything is installed, let's configure our tools!

In order to perform online restorations, [Burp-UI](#) relies on a classical [Burp](#) client.

We need to define our client, and we also need to allow it to perform restorations for other clients. We will set it up globally. Our client will be named *bui*:

```
# burp-ui client's definition
cat >/etc/burp/clientconfdir/bui<<EOF
password = abcdefgh
EOF

# grant our client to perform restorations for others
echo "restore_client = bui" >>/etc/burp/burp-server.conf
# Burp 2 is able to cache the manifests for better performances
echo "monitor_browse_cache = 1" >>/etc/burp/burp-server.conf

# now we generate ou client configuration
cat >/etc/burp/burp.conf<<EOF
mode = client
port = 4971
status_port = 4972
server = ::1
password = abcdefgh
cname = bui
pidfile = /var/run/burp.bui.pid
```

```

syslog = 0
stdout = 1
progress_counter = 1
network_timeout = 72000
ca_burp_ca = /usr/sbin/burp_ca
ca_csr_dir = /etc/burp/CA-client
# SSL certificate authority - same file on both server and client
ssl_cert_ca = /etc/burp/ssl_cert_ca-client.pem
# Client SSL certificate
ssl_cert = /etc/burp/ssl_cert-client.pem
# Client SSL key
ssl_key = /etc/burp/ssl_cert-client.key
# SSL key password
ssl_key_password = password
# Common name in the certificate that the server gives us
ssl_peer_cn = burpserver
# The following options specify exactly what to backup.
include = /home
EOF

```

Our **Burp** server is now set up, we can start it:

```
/etc/init.d/burp start
```

Now we can configure **Burp-UI**. The package comes with a default configuration and init scripts. We copy them at the right place:

```

cp /usr/local/share/burpui/contrib/debian/init.sh /etc/init.d/burp-ui
chmod +x /etc/init.d/burp-ui
update-rc.d burp-ui defaults
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg

```

We have to edit the default configuration in order to work with a **Burp-2** server:

```
sed -i "s/^version = .*/version = 2/" /etc/burp/burpui.cfg
```

That's it, the other default parameter should be able to handle such a setup. We can start **Burp-UI**:

```
/etc/init.d/burp-ui start
```

Your server is now fully set-up, you can access **Burp-UI** by pointing your browser to: http://server_ip:5000/

The default user / password is: admin / admin

For further customization, you can refer to the usage page of this documentation.

1.15.3 Multiple servers

[TODO]

1.16 Developer Guide

1.16.1 Development

If you wish to use the latest and yet unstable version (eg. `master`), you can install it using `pip` too, but I would recommend you to use a `virtualenv`.

To do so, run the following commands:

```
mkdir /opt/bui-venv
pip install virtualenv
virtualenv /opt/bui-venv
source /opt/bui-venv/bin/activate
pip install --upgrade https://burpui.ziirish.me/builds/burp-ui.dev.tar.gz
```

You can uninstall/disable this [Burp-UI](#) setup by typing `deactivate` and removing the `/opt/bui-venv` directory.

1.16.2 Hacking

For those of you who would like to hack on the project, I have split out the repository to keep a copy of all the external dependencies (JS and CSS) in a git submodule.

In order to run local debugging, you need to retrieve this git submodule.

To do so, run the following commands:

```
git clone https://git.ziirish.me/ziirish/burp-ui.git
cd burp-ui
git submodule update --init
```

1.16.3 create_app

The `create_app` method is the most important. This is where the application is created. It is mostly used when running your application through `gunicorn`.

`burpui.create_app(conf=None, verbose=0, logfile=None, gunicorn=True, unittest=False, debug=False, cli=False)`

Initialize the whole application.

Parameters

- **conf** (*str*) – Configuration file to use
- **verbose** (*int*) – Set the verbosity level
- **logfile** (*str*) – Store the logs in the given file
- **gunicorn** (*bool*) – Enable gunicorn engine instead of flask's default
- **unittest** (*bool*) – Are we running tests (used for test only)
- **debug** (*bool*) – Enable debug mode
- **cli** (*bool*) – Are we running the CLI

Returns A `burpui.server.BUIServer` object

1.16.4 API

Here are the different routes provided by the application. You can implement whatever front-end you like on top of it.

GET `/api/admin/auth/backends`

Returns a list of backends

GET method provided by the webservice.

Returns Backends

GET /api/admin/auth/users

Returns a list of users

GET method provided by the webservice.

Returns Users

PUT /api/admin/auth/users

Create a new user

POST /api/admin/auth/users

Change user password

DELETE /api/admin/auth/users

Delete a user

GET /api/admin/me/session

Returns a list of sessions

GET method provided by the webservice.

Returns Sessions

DELETE /api/admin/me/session

Delete a given session

Note: `id` is mandatory

GET /api/preferences/ui

Returns a list of prefs

GET method provided by the webservice.

Returns prefs

PUT /api/preferences/ui

Create prefs

POST /api/preferences/ui

Change prefs

DELETE /api/preferences/ui

Delete prefs

GET /api/settings/server-config

Reads the server configuration

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "boolean": [
    "daemon",
    "fork",
    "...",
  ],
  "defaults": {
    "address": "",
    "autoupgrade_dir": "",
    "ca_burp_ca": "",
    "ca_conf": "",
    "ca_name": "",
    "ca_server_name": "",
    "client_can_delete": true,
```

```
    "...": "...",
  },
  "integer": [
    "port",
    "status_port",
    "...",
  ],
  "multi": [
    "keep",
    "restore_client",
    "...",
  ],
  "placeholders": {
    "autoupgrade_dir": "path",
    "ca_burp_ca": "path",
    "ca_conf": "path",
    "ca_name": "name",
    "ca_server_name": "name",
    "client_can_delete": "0|1",
    "...": "...",
  },
  "results": {
    "boolean": [
      {
        "name": "hardlinked_archive",
        "value": false
      },
      {
        "name": "syslog",
        "value": true
      },
      { "...": "..." }
    ],
    "clients": [
      {
        "name": "testclient",
        "value": "/etc/burp/clientconffdir/testclient"
      }
    ],
    "common": [
      {
        "name": "mode",
        "value": "server"
      },
      {
        "name": "directory",
        "value": "/var/spool/burp"
      },
      { "...": "..." }
    ],
    "includes": [],
    "includes_ext": [],
    "integer": [
      {
        "name": "port",
        "value": 4971
      },
      {
```

```

        "name": "status_port",
        "value": 4972
      },
      { "...": "..." }
    ],
    "multi": [
      {
        "name": "keep",
        "value": [
          "7",
          "4"
        ]
      },
      { "...": "..." }
    ]
  },
  "server_doc": {
    "address": "Defines the main TCP address that the server listens on. The default is either '
    "...": "..."
  },
  "string": [
    "mode",
    "address",
    "..."
  ],
  "suggest": {
    "compression": [
      "gzip1",
      "gzip2",
      "gzip3",
      "gzip4",
      "gzip5",
      "gzip6",
      "gzip7",
      "gzip8",
      "gzip9"
    ],
    "mode": [
      "client",
      "server"
    ],
    "...": []
  }
}

```

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

POST /api/settings/server-config

Saves the server configuration

GET /api/settings/path-expander

Expands a given path

For instance if it's given a glob expression it will returns a list of files matching the expression.

GET /api/settings/clients

Returns a list of clients

GET /api/settings/options

Returns various setting options

PUT /api/settings/config

Creates a new client

GET /api/clients/backup-running

Tells if a backup is running right now

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "running": false
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

GET /api/clients/running

Returns a list of clients currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[ 'client1', 'client2' ]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Ask a specific client in order to know if it is running a backup

Returns The *JSON* described above.

GET /api/servers/report

Returns a global report about all the servers managed by Burp-UI

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "backups": [
    {
      "name": "AGENT1",
      "number": 49
    }
  ],
  "servers": [
    {
```

```

    "name": "AGENT1",
    "stats": {
      "linux": 4,
      "total": 349705,
      "totsize": 119400711726,
      "unknown": 0,
      "windows": 1
    }
  ]
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients/servers you are authorized to.

Returns The *JSON* described above.

GET /api/clients/report

Returns a global report about all the clients of a given server

GET method provided by the webservice.

The *JSON* returned is:

```

{
  "backups": [
    {
      "name": "client1",
      "number": 15
    },
    {
      "name": "client2",
      "number": 1
    }
  ],
  "clients": [
    {
      "name": "client1",
      "stats": {
        "total": 296377,
        "totsize": 57055793698,
        "windows": "unknown"
      }
    },
    {
      "name": "client2",
      "stats": {
        "total": 3117,
        "totsize": 5345361,
        "windows": "true"
      }
    }
  ]
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET /api/servers/stats

Returns a list of servers (agents) with basic stats

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    'alive': true,
    'clients': 2,
    'name': 'burp1',
  },
  {
    'alive': false,
    'clients': 0,
    'name': 'burp2',
  },
]
```

Returns The *JSON* described above.

GET /api/clients/stats

Returns a list of clients with their states

GET method provided by the webservice.

The *JSON* returned is:

```
{
  [
    {
      "last": "2015-05-17 11:40:02",
      "name": "client1",
      "state": "idle",
      "phase": "phase1",
      "percent": 12,
    },
    {
      "last": "never",
      "name": "client2",
      "state": "idle",
      "phase": "phase2",
      "percent": 42,
    }
  ]
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET /api/clients/all

Returns a list of all clients with their associated Agent if any

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "name": "client1",
    "agent": "agent1"
  },
  {
    "name": "client2",
    "agent": "agent1"
  },
  {
    "name": "client3",
    "agent": "agent2"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET /api/admin/me

Returns the current user informations

GET method provided by the webservice.

Returns Users

GET /api/misc/counters

Returns counters for a given client

GET method provided by the webservice.

Parameters

- **name** – the client name if any. You can also use the GET parameter

‘name’ to achieve the same thing

Returns Counters

GET /api/misc/monitor

Returns a list of clients that are currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    'client': 'client1',
    'agent': 'burpl',
    'counters': {
      'phase': 2,
      'path': '/etc/some/configuration',
      '...': '...',
    },
  },
]
```

```
{
  'client': 'client12',
  'agent': 'burp2',
  'counters': {
    'phase': 3,
    'path': '/etc/some/other/configuration',
    '...': '...',
  },
},
],
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET /api/misc/history

Returns a list of calendars describing the backups that have been completed so far

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "color": "#7C6F44",
    "events": [
      {
        "backup": "0000001",
        "end": "2015-01-25 13:32:04+01:00",
        "name": "toto-test",
        "start": "2015-01-25 13:32:00+01:00",
        "title": "Client: toto-test, Backup n°0000001",
        "url": "/client/toto-test"
      }
    ],
    "name": "toto-test",
    "textColor": "white"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Which client to collect data from

Returns The *JSON* described above

POST /api/misc/alert

Propagate a message to the next screen (or whatever reads the session)

GET /api/misc/about

Returns various informations about Burp-UI

GET `/api/swagger.json`
Render the Swagger specifications as JSON

GET `/api/doc`
Override this method to customize the documentation page

GET `/api/settings/ (server) /config/`
client/path: conf Reads a given client configuration

POST `/api/settings/ (server) /config/`
client/path: conf Saves a given client configuration

DELETE `/api/settings/ (server) /config/`
client/path: conf Deletes a given client

PUT `/api/restore/ (server) /server-restore/`
name/int: backup Schedule a server-initiated restoration

PUT method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns Status message (success or failure)

POST `/api/restore/ (server) /archive/`
name/int: backup Performs an online restoration

POST method provided by the webservice. This method returns a `flask.Response` object.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns A `flask.Response` object representing an archive of the restored files

GET `/api/client/ (server) /browseall/`
name/int: backup Returns a list of all ‘nodes’ of a given backup

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-05-21 14:54:49",
    "gid": "0",
    "inodes": "173",
    "selected": false,
    "expanded": false,
    "children": [],
    "mode": "drwxr-xr-x",
    "name": "/",
    "key": "/",
    "title": "/",
    "fullname": "/",
    "parent": ""
```

```
"size": "12.0KiB",
"type": "d",
"uid": "0"
}
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/ (server) /browse/`
name/int: backup Returns a list of ‘nodes’ under a given path

GET method provided by the webservice.

The *JSON* returned is:

```
[
{
  "date": "2015-05-21 14:54:49",
  "gid": "0",
  "inodes": "173",
  "selected": false,
  "expanded": false,
  "children": [],
  "mode": "drwxr-xr-x",
  "name": "/",
  "key": "/",
  "title": "/",
  "fullname": "/",
  "parent": "",
  "size": "12.0KiB",
  "type": "d",
  "uid": "0"
}
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/ (server) /report/`
name/int: backup Returns a global report of a given backup/client

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 394,
    "scanned": 394,
    "total": 394,
    "unchanged": 0
  },
  "duration": 5,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": true,
  "end": 1422189124,
  "files": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 1421,
    "scanned": 1421,
    "total": 1421,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,

```

```
    "total": 0,
    "unchanged": 0
  },
  "number": 1,
  "received": 1679304,
  "softlink": {
    "changed": 0,
    "deleted": 0,
    "new": 1302,
    "scanned": 1302,
    "total": 1302,
    "unchanged": 0
  },
  "special": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "start": 1422189119,
  "total": {
    "changed": 0,
    "deleted": 0,
    "new": 3117,
    "scanned": 3117,
    "total": 3117,
    "unchanged": 0
  },
  "totsize": 5345361,
  "vssfooter": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssfooter_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader_enc": {
    "changed": 0,
    "deleted": 0,
```

```

    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "windows": "false"
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/settings/config/ (client) /`
path: *conf* Reads a given client configuration

POST `/api/settings/config/ (client) /`
path: *conf* Saves a given client configuration

DELETE `/api/settings/config/ (client) /`
path: *conf* Deletes a given client

GET `/api/settings/ (server) /path-expander/`
client Expands a given path

For instance if it's given a glob expression it will return a list of files matching the expression.

GET `/api/settings/ (server) /server-config/`
path: *conf* Reads the server configuration

GET method provided by the webservice.

The *JSON* returned is:

```

{
  "boolean": [
    "daemon",
    "fork",
    "..."
  ],
  "defaults": {
    "address": "",
    "autoupgrade_dir": "",
    "ca_burp_ca": "",
    "ca_conf": "",
    "ca_name": "",
    "ca_server_name": "",
    "client_can_delete": true,
    "...": "..."
  },
  "integer": [
    "port",
    "status_port",
    "..."
  ],
}

```

```
"multi": [
  "keep",
  "restore_client",
  "...",
],
"placeholders": {
  "autoupgrade_dir": "path",
  "ca_burp_ca": "path",
  "ca_conf": "path",
  "ca_name": "name",
  "ca_server_name": "name",
  "client_can_delete": "0|1",
  "...": "..."
},
"results": {
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    },
    { "...": "..." }
  ],
  "clients": [
    {
      "name": "testclient",
      "value": "/etc/burp/clientconfdir/testclient"
    }
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
      "value": "/var/spool/burp"
    },
    { "...": "..." }
  ],
  "includes": [],
  "includes_ext": [],
  "integer": [
    {
      "name": "port",
      "value": 4971
    },
    {
      "name": "status_port",
      "value": 4972
    },
    { "...": "..." }
  ],
  "multi": [
    {
```

```

        "name": "keep",
        "value": [
            "7",
            "4"
        ]
    },
    { "...": "..." }
]
},
"server_doc": {
    "address": "Defines the main TCP address that the server listens on. The default is either '
    "...": "..."
},
"string": [
    "mode",
    "address",
    "..."
],
"suggest": {
    "compression": [
        "gzip1",
        "gzip2",
        "gzip3",
        "gzip4",
        "gzip5",
        "gzip6",
        "gzip7",
        "gzip8",
        "gzip9"
    ],
    "mode": [
        "client",
        "server"
    ],
    "...": []
}
}

```

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

POST `/api/settings/ (server) /server-config/`
path: *conf* Saves the server configuration

GET `/api/settings/ (server) /config/`
client Reads a given client configuration

POST `/api/settings/ (server) /config/`
client Saves a given client configuration

DELETE `/api/settings/ (server) /config/`
client Deletes a given client

PUT `/api/restore/server-restore/ (name) /`
int: *backup* Schedule a server-initiated restoration

PUT method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns Status message (success or failure)

POST `/api/restore/archive/` (*name*) /
int: *backup* Performs an online restoration

POST method provided by the webservice. This method returns a `flask.Response` object.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns A `flask.Response` object representing an archive of the restored files

GET `/api/restore/` (*server*) `/server-restore/`
name Reads the content of the 'restore' file if present

GET method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns The content of the restore file

DELETE `/api/restore/` (*server*) `/server-restore/`
name Remove the 'restore' file if present

DELETE method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/clients/` (*server*) `/running/`
client Returns a list of clients currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[ 'client1', 'client2' ]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Ask a specific client in order to know if it is running a backup

Returns The *JSON* described above.

GET `/api/client/browseall/ (name) /`
int: backup Returns a list of all ‘nodes’ of a given backup

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-05-21 14:54:49",
    "gid": "0",
    "inodes": "173",
    "selected": false,
    "expanded": false,
    "children": [],
    "mode": "drwxr-xr-x",
    "name": "/",
    "key": "/",
    "title": "/",
    "fullname": "/",
    "parent": "",
    "size": "12.0KiB",
    "type": "d",
    "uid": "0"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/browse/ (name) /`
int: backup Returns a list of ‘nodes’ under a given path

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-05-21 14:54:49",
    "gid": "0",
    "inodes": "173",
    "selected": false,
    "expanded": false,
    "children": [],
    "mode": "drwxr-xr-x",
    "name": "/",
    "key": "/",
    "title": "/",
    "fullname": "/",
    "parent": "",
    "size": "12.0KiB",
    "type": "d",
  }
]
```

```
    "uid": "0"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/report/ (name) /`

int: *backup* Returns a global report of a given backup/client

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 394,
    "scanned": 394,
    "total": 394,
    "unchanged": 0
  },
  "duration": 5,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": true,
  "end": 1422189124,
  "files": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 1421,
    "scanned": 1421,
    "total": 1421,
    "unchanged": 0
  },
  "hardlink": {
```

```
"changed": 0,
"deleted": 0,
"new": 0,
"scanned": 0,
"total": 0,
"unchanged": 0
},
"meta": {
  "changed": 0,
  "deleted": 0,
  "new": 0,
  "scanned": 0,
  "total": 0,
  "unchanged": 0
},
"meta_enc": {
  "changed": 0,
  "deleted": 0,
  "new": 0,
  "scanned": 0,
  "total": 0,
  "unchanged": 0
},
"number": 1,
"received": 1679304,
"softlink": {
  "changed": 0,
  "deleted": 0,
  "new": 1302,
  "scanned": 1302,
  "total": 1302,
  "unchanged": 0
},
"special": {
  "changed": 0,
  "deleted": 0,
  "new": 0,
  "scanned": 0,
  "total": 0,
  "unchanged": 0
},
"start": 1422189119,
"total": {
  "changed": 0,
  "deleted": 0,
  "new": 3117,
  "scanned": 3117,
  "total": 3117,
  "unchanged": 0
},
"totsize": 5345361,
"vssfooter": {
  "changed": 0,
  "deleted": 0,
  "new": 0,
  "scanned": 0,
  "total": 0,
  "unchanged": 0
}
```

```
},
  "vssfooter_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "windows": "false"
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

PUT `/api/backup/` (*server*) **/do-server-backup/**
name Schedule a server-initiated backup

PUT method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/backup/` (*server*) **/server-backup/**
name Tells if a 'backup' file is present

GET method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns True if the file is found

DELETE `/api/backup/ (server) /server-backup/`
name Remove the 'backup' file if present

DELETE method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/client/ (server) /report/`
name Returns a global report of a given backup/client

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 394,
    "scanned": 394,
    "total": 394,
    "unchanged": 0
  },
  "duration": 5,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": true,
  "end": 1422189124,
  "files": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 1421,
    "scanned": 1421,
    "total": 1421,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
```

```
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "number": 1,
  "received": 1679304,
  "softlink": {
    "changed": 0,
    "deleted": 0,
    "new": 1302,
    "scanned": 1302,
    "total": 1302,
    "unchanged": 0
  },
  "special": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "start": 1422189119,
  "total": {
    "changed": 0,
    "deleted": 0,
    "new": 3117,
    "scanned": 3117,
    "total": 3117,
    "unchanged": 0
  },
  "totsize": 5345361,
  "vssfooter": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssfooter_enc": {
    "changed": 0,
```

```

    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "windows": "false"
}

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/(server)/stats/`
name Returns a list of backups for a given client

GET method provided by the webservice.

The *JSON* returned is:

```

[
  {
    "date": "2015-01-25 13:32:00",
    "deletable": true,
    "encrypted": true,
    "received": 123,
    "size": 1234,
    "number": 1
  },
]

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

- **name** (*str*) – The client we are working on

Returns The *JSON* described above.

GET `/api/admin/auth/users/` (*name*)

Returns a list of users

GET method provided by the webservice.

Returns Users

PUT `/api/admin/auth/users/` (*name*)

Create a new user

POST `/api/admin/auth/users/` (*name*)

Change user password

DELETE `/api/admin/auth/users/` (*name*)

Delete a user

GET `/api/admin/me/session/` (*id*)

Returns a list of sessions

GET method provided by the webservice.

Returns Sessions

DELETE `/api/admin/me/session/` (*id*)

Delete a given session

Note: *id* is mandatory

GET `/api/misc/` (*server*) `/counters/`
client Returns counters for a given client

GET method provided by the webservice.

Parameters

- **name** – the client name if any. You can also use the GET parameter

‘name’ to achieve the same thing

Returns Counters

GET `/api/misc/` (*server*) `/history/`

client Returns a list of calendars describing the backups that have been completed so far

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "color": "#7C6F44",
    "events": [
      {
        "backup": "0000001",
        "end": "2015-01-25 13:32:04+01:00",
        "name": "toto-test",
        "start": "2015-01-25 13:32:00+01:00",
        "title": "Client: toto-test, Backup n°0000001",
        "url": "/client/toto-test"
      }
    ]
  },
  {
    "name": "toto-test",
```



```

    "textColor": "white"
  }
]

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Which client to collect data from

Returns The *JSON* described above

GET `/api/settings/path-expander/` (*client*)

Expands a given path

For instance if it's given a glob expression it will returns a list of files matching the expression.

GET `/api/settings/server-config/` (*path: conf*)

Reads the server configuration

GET method provided by the webservice.

The *JSON* returned is:

```

{
  "boolean": [
    "daemon",
    "fork",
    "..."
  ],
  "defaults": {
    "address": "",
    "autoupgrade_dir": "",
    "ca_burp_ca": "",
    "ca_conf": "",
    "ca_name": "",
    "ca_server_name": "",
    "client_can_delete": true,
    "...": "..."
  },
  "integer": [
    "port",
    "status_port",
    "..."
  ],
  "multi": [
    "keep",
    "restore_client",
    "..."
  ],
  "placeholders": {
    "autoupgrade_dir": "path",
    "ca_burp_ca": "path",
    "ca_conf": "path",
    "ca_name": "name",
    "ca_server_name": "name",
    "client_can_delete": "0|1",
    "...": "..."
  }
}

```

```
},
"results": {
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    },
    { "...": "..." }
  ],
  "clients": [
    {
      "name": "testclient",
      "value": "/etc/burp/clientconffdir/testclient"
    }
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
      "value": "/var/spool/burp"
    },
    { "...": "..." }
  ],
  "includes": [],
  "includes_ext": [],
  "integer": [
    {
      "name": "port",
      "value": 4971
    },
    {
      "name": "status_port",
      "value": 4972
    },
    { "...": "..." }
  ],
  "multi": [
    {
      "name": "keep",
      "value": [
        "7",
        "4"
      ]
    },
    { "...": "..." }
  ]
},
"server_doc": {
  "address": "Defines the main TCP address that the server listens on. The default is either '
  "...": "..."
},
},
```

```

"string": [
  "mode",
  "address",
  "...",
],
"suggest": {
  "compression": [
    "gzip1",
    "gzip2",
    "gzip3",
    "gzip4",
    "gzip5",
    "gzip6",
    "gzip7",
    "gzip8",
    "gzip9"
  ],
  "mode": [
    "client",
    "server"
  ],
  "...": []
}

```

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

POST `/api/settings/server-config/` (**path:** *conf*)

Saves the server configuration

GET `/api/settings/config/` (*client*)

Reads a given client configuration

POST `/api/settings/config/` (*client*)

Saves a given client configuration

DELETE `/api/settings/config/` (*client*)

Deletes a given client

GET `/api/settings/` (*server*) `/server-config`

Reads the server configuration

GET method provided by the webservice.

The *JSON* returned is:

```

{
  "boolean": [
    "daemon",
    "fork",
    "...",
  ],
  "defaults": {
    "address": "",
    "autoupgrade_dir": "",
    "ca_burp_ca": "",
  },
}

```

```
    "ca_conf": "",
    "ca_name": "",
    "ca_server_name": "",
    "client_can_delete": true,
    "...": "..."
  },
  "integer": [
    "port",
    "status_port",
    "..."
  ],
  "multi": [
    "keep",
    "restore_client",
    "..."
  ],
  "placeholders": {
    "autoupgrade_dir": "path",
    "ca_burp_ca": "path",
    "ca_conf": "path",
    "ca_name": "name",
    "ca_server_name": "name",
    "client_can_delete": "0|1",
    "...": "..."
  },
  "results": {
    "boolean": [
      {
        "name": "hardlinked_archive",
        "value": false
      },
      {
        "name": "syslog",
        "value": true
      },
      { "..." : "..." }
    ],
    "clients": [
      {
        "name": "testclient",
        "value": "/etc/burp/clientconfdir/testclient"
      }
    ],
    "common": [
      {
        "name": "mode",
        "value": "server"
      },
      {
        "name": "directory",
        "value": "/var/spool/burp"
      },
      { "..." : "..." }
    ],
    "includes": [],
    "includes_ext": [],
    "integer": [
      {
```

```

        "name": "port",
        "value": 4971
    },
    {
        "name": "status_port",
        "value": 4972
    },
    { "...": "..." }
],
"multi": [
    {
        "name": "keep",
        "value": [
            "7",
            "4"
        ]
    },
    { "...": "..." }
]
},
"server_doc": {
    "address": "Defines the main TCP address that the server listens on. The default is either '
    "...": "..."
},
"string": [
    "mode",
    "address",
    "..."
],
"suggest": {
    "compression": [
        "gzip1",
        "gzip2",
        "gzip3",
        "gzip4",
        "gzip5",
        "gzip6",
        "gzip7",
        "gzip8",
        "gzip9"
    ],
    "mode": [
        "client",
        "server"
    ],
    "...": []
}
}

```

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

POST `/api/settings/` (*server*) `/server-config`

Saves the server configuration

GET `/api/settings/` (*server*) `/path-expander`

Expend a given path

For instance if it's given a glob expression it will returns a list of files matching the expression.

GET `/api/settings/ (server) /clients`

Returns a list of clients

GET `/api/settings/ (server) /options`

Returns various setting options

PUT `/api/settings/ (server) /config`

Creates a new client

GET `/api/restore/server-restore/ (name)`

Reads the content of the 'restore' file if present

GET method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns The content of the restore file

DELETE `/api/restore/server-restore/ (name)`

Remove the 'restore' file if present

DELETE method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/clients/running/ (client)`

Returns a list of clients currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[ 'client1', 'client2' ]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Ask a specific client in order to know if it is running a backup

Returns The *JSON* described above.

GET `/api/clients/ (server) /backup-running`

Tells if a backup is running right now

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "running": false
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above.

GET `/api/clients/ (server) /running`

Returns a list of clients currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[ 'client1', 'client2' ]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Ask a specific client in order to know if it is running a backup

Returns The *JSON* described above.

GET `/api/clients/ (server) /report`

Returns a global report about all the clients of a given server

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "backups": [
    {
      "name": "client1",
      "number": 15
    },
    {
      "name": "client2",
      "number": 1
    }
  ],
  "clients": [
    {
      "name": "client1",
      "stats": {
        "total": 296377,
        "totsize": 57055793698,
        "windows": "unknown"
      }
    },
    {
      "name": "client2",
      "stats": {
```

```
        "total": 3117,
        "totsize": 5345361,
        "windows": "true"
    }
}
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET `/api/clients/ (server) /stats`

Returns a list of clients with their states

GET method provided by the webservice.

The *JSON* returned is:

```
{
  [
    {
      "last": "2015-05-17 11:40:02",
      "name": "client1",
      "state": "idle",
      "phase": "phase1",
      "percent": 12,
    },
    {
      "last": "never",
      "name": "client2",
      "state": "idle",
      "phase": "phase2",
      "percent": 42,
    }
  ]
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET `/api/clients/ (server) /all`

Returns a list of all clients with their associated Agent if any

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "name": "client1",
    "agent": "agent1"
```



```

    },
    {
      "name": "client2",
      "agent": "agent1"
    },
    {
      "name": "client3",
      "agent": "agent2"
    }
  ]

```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

PUT `/api/backup/do-server-backup/` (*name*)

Schedule a server-initiated backup

PUT method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/backup/server-backup/` (*name*)

Tells if a ‘backup’ file is present

GET method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns True if the file is found

DELETE `/api/backup/server-backup/` (*name*)

Remove the ‘backup’ file if present

DELETE method provided by the webservice.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns Status message (success or failure)

GET `/api/client/report/` (*name*)

Returns a global report of a given backup/client

GET method provided by the webservice.

The *JSON* returned is:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 394,
    "scanned": 394,
    "total": 394,
    "unchanged": 0
  },
  "duration": 5,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": true,
  "end": 1422189124,
  "files": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 1421,
    "scanned": 1421,
    "total": 1421,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "meta_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
  }
```

```
    "unchanged": 0
  },
  "number": 1,
  "received": 1679304,
  "softlink": {
    "changed": 0,
    "deleted": 0,
    "new": 1302,
    "scanned": 1302,
    "total": 1302,
    "unchanged": 0
  },
  "special": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "start": 1422189119,
  "total": {
    "changed": 0,
    "deleted": 0,
    "new": 3117,
    "scanned": 3117,
    "total": 3117,
    "unchanged": 0
  },
  "totsize": 5345361,
  "vssfooter": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssfooter_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "vssheader_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
```

```
"scanned": 0,
"total": 0,
"unchanged": 0
},
"windows": "false"
}
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on
- **backup** (*int*) – The backup we are working on

Returns The *JSON* described above.

GET `/api/client/stats/` (*name*)

Returns a list of backups for a given client

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "date": "2015-01-25 13:32:00",
    "deletable": true,
    "encrypted": true,
    "received": 123,
    "size": 1234,
    "number": 1
  },
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **name** (*str*) – The client we are working on

Returns The *JSON* described above.

GET `/api/misc/counters/` (*client*)

Returns counters for a given client

GET method provided by the webservice.

Parameters

- **name** – the client name if any. You can also use the GET parameter

‘name’ to achieve the same thing

Returns Counters

GET `/api/misc/history/` (*client*)

Returns a list of calendars describing the backups that have been completed so far

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "color": "#7C6F44",
    "events": [
      {
        "backup": "0000001",
        "end": "2015-01-25 13:32:04+01:00",
        "name": "toto-test",
        "start": "2015-01-25 13:32:00+01:00",
        "title": "Client: toto-test, Backup n°0000001",
        "url": "/client/toto-test"
      }
    ],
    "name": "toto-test",
    "textColor": "white"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Which client to collect data from

Returns The *JSON* described above

GET `/api/misc/ (server) /counters`

Returns counters for a given client

GET method provided by the webservice.

Parameters

- **name** – the client name if any. You can also use the GET parameter

‘name’ to achieve the same thing

Returns Counters

GET `/api/misc/ (server) /monitor`

Returns a list of clients that are currently running a backup

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    'client': 'client1',
    'agent': 'burp1',
    'counters': {
      'phase': 2,
      'path': '/etc/some/configuration',
      '...': '...',
    },
  },
  {
    'client': 'client12',
    'agent': 'burp2',
  }
]
```

```
'counters': {
  'phase': 3,
  'path': '/etc/some/other/configuration',
  '...': '...',
},
},
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode

Returns The *JSON* described above

GET `/api/misc/ (server) /history`

Returns a list of calendars describing the backups that have been completed so far

GET method provided by the webservice.

The *JSON* returned is:

```
[
  {
    "color": "#7C6F44",
    "events": [
      {
        "backup": "0000001",
        "end": "2015-01-25 13:32:04+01:00",
        "name": "toto-test",
        "start": "2015-01-25 13:32:00+01:00",
        "title": "Client: toto-test, Backup n°0000001",
        "url": "/client/toto-test"
      }
    ],
    "name": "toto-test",
    "textColor": "white"
  }
]
```

The output is filtered by the `burpui.misc.acl` module so that you only see stats about the clients you are authorized to.

Parameters

- **server** (*str*) – Which server to collect data from when in multi-agent mode
- **client** (*str*) – Which client to collect data from

Returns The *JSON* described above

GET `/api/misc/ (server) /about`

Returns various informations about Burp-UI

1.16.5 Backend

Here is the *backend* interface definition in order to implement a new backend.

class `burpui.misc.backend.interface.BUIbackend` (*server=None, conf=None*)

The `burpui.misc.backend.interface.BUIbackend` class provides a consistent interface backend for any burp server.

Parameters

- **server** (`burpui.server.BUIServer`) – Flask server instance in order to access logger and/or some global settings
- **conf** (*str*) – Configuration file to use

cancel_server_backup (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.cancel_server_backup()` function is used to delete the server-initiated backup file of a given client.

Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

cancel_server_restore (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.cancel_server_restore()` function is used to delete the server-initiated restoration file of a given client.

Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

clients_list (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.clients_list()` function is used to retrieve a list of clients with their configuration file.

Returns A list of clients with their configuration file

del_file (*path, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.del_file()` function is used to delete a file on a remote agent.

delete_client (*client=None, keepconf=False, delcert=False, revoke=False, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.delete_client()` function is used to delete a client from burp's configuration.

Parameters

- **client** (*str*) – The name of the client to remove
- **keepconf** (*bool*) – Whether to keep the conf (in order to just revoke/delete the certs for instance)
- **delcert** (*bool*) – Whether to delete the associated certificate

- **revoke** (*bool*) – Whether to revoke the associated certificate
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

expand_path (*path=None, source=None, client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.expand_path()` function is used to expand path of file inclusions glob the user can set in the setting panel. This function is also a *proxy* for multi-agent setup.

Parameters

- **path** (*str*) – The glob/path to expand
- **source** (*str*) – In which file are we working
- **client** (*str*) – The client name when working on client files
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of files or an empty list

get_all_clients (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_all_clients()` function returns a list containing all the clients with their states.

Parameters **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of clients

Example:

```
[
  {
    "last": "2015-10-02 08:20:03",
    "name": "client1",
    "state": "idle",
    "percent": null,
    "phase": null,
  },
  {
    "last": "2015-01-25 13:32:00",
    "name": "client2",
    "state": "idle",
    "percent": null,
    "phase": null,
  },
]
```

get_attr (*name, default=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_attr()` function returns the given attribute or default.

get_backup_logs (*number, client, forward=False, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_backup_logs()` function is used to retrieve the burp logs depending the burp-server version.

Parameters

- **number** (*int*) – Backup number to work on
- **client** (*str*) – Client name to work on
- **forward** (*bool*) – Is the client name needed in later process

- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns Dict containing the backup log

Example:

```
{
  "dir": {
    "changed": 0,
    "deleted": 0,
    "new": 17,
    "scanned": 30246,
    "total": 30246,
    "unchanged": 30229
  },
  "duration": 436,
  "efs": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "encrypted": false,
  "end": 1443767237,
  "files": {
    "changed": 47,
    "deleted": 2,
    "new": 2,
    "scanned": 227377,
    "total": 227377,
    "unchanged": 227328
  },
  "files_enc": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 0,
    "total": 0,
    "unchanged": 0
  },
  "hardlink": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 28,
    "total": 28,
    "unchanged": 28
  },
  "meta": {
    "changed": 0,
    "deleted": 0,
    "new": 0,
    "scanned": 58,
    "total": 58,
    "unchanged": 58
  },
  "meta_enc": {
    "changed": 0,
```

```
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "number": 576,
    "received": 11691704,
    "softlink": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 9909,
        "total": 9909,
        "unchanged": 9909
    },
    "special": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 1,
        "total": 1,
        "unchanged": 1
    },
    "start": 1443766801,
    "total": {
        "changed": 47,
        "deleted": 2,
        "new": 19,
        "scanned": 267619,
        "total": 267619,
        "unchanged": 267553
    },
    "totsize": 52047768383,
    "vssfooter": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "vssfooter_enc": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "vssheader": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    },
```

```

    "vssheader_enc": {
        "changed": 0,
        "deleted": 0,
        "new": 0,
        "scanned": 0,
        "total": 0,
        "unchanged": 0
    },
    "windows": "false"
}

```

get_client (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client()` function returns a list of dict representing the backups of a given client.

Parameters

- **name** (*str*) – Client name
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of backups

Example:

```

[
    {
        "date": "2015-01-25 13:32:00",
        "deletable": true,
        "encrypted": true,
        "number": "1",
        "received": 889818873,
        "size": 35612321050,
    }
]

```

get_client_filtered (*name=None, limit=-1, page=None, start=None, end=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_filtered()` function returns a list of dict representing the backups of a given client filtered by the given criteria.

Parameters

- **name** (*str*) – Client name
- **limit** (*int*) – Number of element to return, -1 for not limit
- **page** (*int*) – What page to retrieve
- **start** (*int*) – Return elements after this date
- **end** (*int*) – Return elements until this date
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of backups

Example:

```

[
    {
        "date": "2015-01-25 13:32:00",
        "deletable": true,
        "encrypted": true,
        "number": "1",
    }
]

```

```
        "received": 889818873,  
        "size": 35612321050,  
    }  
]
```

get_client_labels (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_labels()` function returns a list of labels (if any) for a given client.

Note: Labels are only available since Burp 2.0.34

Parameters

- **client** (*str*) – The client for which you want the labels
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of labels or an empty list

get_client_version (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_client_version()` function returns the client version used to connect to the server.

Parameters **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns Burp client version

get_clients_report (*clients, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_clients_report()` function returns the computed/compacted data to display clients report.

Parameters

- **clients** (*list*) – List of clients as returned by `burpui.misc.backend.interface.BUIbackend.get_all_clients()`
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A dict with the computed data

get_counters (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_counters()` function returns a dict of counters for a given client while it performs a backup.

Parameters

- **name** (*str*) – Name of the client for which you'd like stats
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A dict of counters

get_file (*path, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_file()` function is used to retrieve a file on a remote agent.

get_parser (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_parser()` function returns the parser of the current backend.

get_parser_attr (*attr=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_parser_attr()` function is used to retrieve some attributes from the Parser. This function is useful in multi-agent mode because the front-end needs to access the backend attributes through the agents.

Parameters

- **attr** (*str*) – Name of the attribute to retrieve
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns The requested attribute or an empty list

get_server_version (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_server_version()` function returns the server version (if any).

Parameters **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns Burp server version

get_tree (*name=None, backup=None, root=None, level=-1, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.get_tree()` function returns a list of dict representing files/dir (with their attr) within a given path

Parameters

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **root** (*str*) – Root path to look into
- **level** (*int*) – Level of the tree relative to its root
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of files/dir within the given path with their attr

Example:

```
[
  {
    "date": "2015-01-23 20:00:07",
    "gid": "0",
    "inodes": "168",
    "mode": "drwxr-xr-x",
    "name": "/",
    "parent": "",
    "fullname": "/",
    "level": -1,
    "size": "12.0KiB",
    "type": "d",
    "uid": "0",
    "folder": True,
    "children": []
  }
]
```

is_backup_running (*name=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_backup_running()` functions tells you if a given client is currently performing a backup.

Parameters

- **name** (*str*) – Name of the client
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns True or False

is_one_backup_running (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_one_backup_running()` function tells you if at least one backup is running.

Parameters **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of running clients

is_server_backup (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_server_backup()` function is used to know if there is a server-initiated backup file in place.

Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns True or False

is_server_restore (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.is_server_restore()` function is used to know if there is a server-initiated restoration file in place and retrieve its content in order to edit it.

Parameters

- **client** (*str*) – The name of the client to look for
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A dict representing the content of the server-initiated restoration file

read_conf_cli (*client=None, conf=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.read_conf_cli()` function works the same way as the `burpui.misc.backend.interface.BUIbackend.read_conf_srv()` function but for the client config file.

read_conf_srv (*conf=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.read_conf_srv()` function returns a dict of options present in the server config file.

Parameters

- **conf** (*str*) – Complementary configuration file (for instance, file inclusions)
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns Dict of options

Example:

```
{
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    }
  ]
}
```

```

    },
  ],
  "clients": [
    {
      "name": "client1",
      "value": "/etc/burp/clientconffdir/client1"
    },
    {
      "name": "client2",
      "value": "/etc/burp/clientconffdir/client2"
    },
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
      "value": "/srv/burp"
    },
  ],
  "includes": [],
  "includes_ext": [],
  "integer": [
    {
      "name": "port",
      "value": 4971
    },
    {
      "name": "status_port",
      "value": 4972
    },
    {
      "name": "max_children",
      "value": 5
    },
    {
      "name": "max_status_children",
      "value": 5
    },
  ],
  "multi": [
    {
      "name": "keep",
      "value": [
        "7",
        "4",
        "4"
      ]
    },
    {
      "name": "timer_arg",
      "value": [
        "12h",
        "Mon,Tue,Thu,Fri,17,18,19,20,21,22,23",
        "Wed,Sat,Sun,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23"
      ]
    },
  ]

```

```
    },  
    ],  
}
```

restore_files (*name=None, backup=None, files=None, strip=None, archive='zip', password=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.restore_files()` function performs a restoration and returns a tuple containing the path of the generated archive and/or a message if an error happened.

Parameters

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **files** (*str*) – A string representing a list of files to restore

Example:

```
['/etc/passwd', '/etc/shadow']
```

Parameters

- **strip** (*int*) – Number of parent directories to strip while restoring files
- **archive** (*str*) – Format of the generated archive (may be zip, tar.gz or tar.bz2)
- **password** (*str*) – Password for encrypted backups
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A tuple with the generated archive path and/or an error message

revocation_enabled (*agent=None*)

The `burpui.misc.backend.interface.BUIbackend.revocation_enabled()` function is used to know if the revocation feature is enabled or not.

Parameters **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns True or False

server_backup (*client=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.server_backup()` function is used to schedule a server-side initiated backup.

Parameters

- **client** (*str*) – Client name
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

server_restore (*client=None, backup=None, files=None, strip=None, force=None, prefix=None, restoreto=None, agent=None*)

The `burpui.misc.backend.interface.BUIbackend.server_restore()` function is used to schedule a server-side initiated restoration.

Parameters

- **client** (*str*) – Client name
- **backup** (*int*) – Backup number

- **files** (*str*) – List of files to restore
- **strip** (*int*) – Number of leading path to strip
- **force** (*bool*) – Whether to force overriding files or not
- **prefix** (*str*) – Where to restore files
- **retoreto** – Restore on an other client
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

status (*query*=`'\n'`, *timeout*=`None`, *agent*=`None`)

The `burpui.misc.backend.interface.BUIbackend.status()` method is used to send queries to the Burp server

Parameters

- **query** (*str*) – Query to send to the server
- **timeout** (*int*) – Query timeout in seconds
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns The output returned by the server parsed as an array

Example:

```
[
    "client1      2      i      576 0 1443766803",
    "client2      2      i      1 0 1422189120",
]
```

store_conf_cli (*data*, *client*=`None`, *conf*=`None`, *agent*=`None`)

The `burpui.misc.backend.interface.BUIbackend.store_conf_cli()` function works the same way as the `burpui.misc.backend.interface.BUIbackend.store_conf_srv()` function but for the client config file. It takes an extra parameter:

Parameters **client** (*str*) – Name of the client for which to apply this config

store_conf_srv (*data*, *conf*=`None`, *agent*=`None`)

The `burpui.misc.backend.interface.BUIbackend.store_conf_srv()` functions is used to save the new settings in the configuration file.

Parameters

- **data** (*dict*) – Data as sent by the web-form
- **conf** (*str*) – Force the file path (for file inclusions for instance)
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

1.16.6 Parser

Here is the *parser* interface definition in order to implement a new parser.

class `burpui.misc.parser.interface.BUIparser`

`burpui.misc.parser.interface.BUIparser` defines a generic interface for burp configuration files parser.

cancel_backup (*name=None*)

`burpui.misc.parser.interface.BUIparser.cancel_backup()` called by
`burpui.misc.backend.interface.BUIbackend.cancel_server_backup()` in or-
der to cancel a server-initiated backup.

Parameters **name** (*str*) – Client name

Returns A list of notifications to return to the UI (success or failure)

cancel_restore (*name=None*)

`burpui.misc.parser.interface.BUIparser.cancel_restore()` called by
`burpui.misc.backend.interface.BUIbackend.cancel_server_restore()` in
order to cancel a server-initiated restoration.

Parameters **name** (*str*) – Client name

Returns A list of notifications to return to the UI (success or failure)

is_client_revoked (*client=None*)

`burpui.misc.parser.interface.BUIparser.is_client_revoked()` is used to check if
a given client has it's certificate revoked or not.

Parameters **client** (*str*) – The name of the client to check

Returns True or False

list_clients ()

`burpui.misc.parser.interface.BUIparser.list_clients()` is used to retrieve a list of
clients with their configuration file.

Returns A list of clients with their configuration file

param (*name, obj='server_conf', client=None*)

`burpui.misc.parser.interface.BUIparser.param()` lookup for a given param in the conf.

Parameters

- **name** (*str*) – Param name
- **obj** (*str*) – Object to look param for
- **client** (*str*) – Search for a given client param

Returns The asked param

path_expander (*pattern=None, source=None, client=None*)

`burpui.misc.parser.interface.BUIparser.path_expander()` is used to expand path of
file inclusions glob the user can set in the setting panel.

Parameters

- **pattern** (*str*) – The glob/path to expand
- **source** (*str*) – What file we are working in
- **client** (*str*) – The client name when working on client files

Returns A list of files or an empty list

read_backup (*name=None*)

`burpui.misc.parser.interface.BUIparser.read_backup()` called by

`burpui.misc.backend.interface.BUIbackend.is_server_backup()` in order to test the existence of a server-initiated backup file.

Parameters `name (str)` – Client name

Returns A True if the file is found, else False.

read_client_conf (`client=None, conf=None`)

`burpui.misc.parser.interface.BUIparser.read_client_conf()` is called by `burpui.misc.backend.interface.BUIbackend.read_conf_cli()` in order to parse the burp-clients configuration files.

It works the same way as `burpui.misc.parser.interface.BUIparser.read_server_conf()`

read_restore (`name=None`)

`burpui.misc.parser.interface.BUIparser.read_restore()` called by `burpui.misc.backend.interface.BUIbackend.is_server_restore()` in order to read a server-initiated restoration file.

Parameters `name (str)` – Client name

Returns A dict describing the content of the file

read_server_conf (`conf=None`)

`burpui.misc.parser.interface.BUIparser.read_server_conf()` is called by `burpui.misc.backend.interface.BUIbackend.read_conf_srv()` in order to parse the burp-server configuration file.

Parameters `conf (str)` – Complementary configuration file (for instance, file inclusions)

Returns Dict of options

Example:

```
{
  "boolean": [
    {
      "name": "hardlinked_archive",
      "value": false
    },
    {
      "name": "syslog",
      "value": true
    }
  ],
  "clients": [
    {
      "name": "client1",
      "value": "/etc/burp/clientconffdir/client1"
    },
    {
      "name": "client2",
      "value": "/etc/burp/clientconffdir/client2"
    }
  ],
  "common": [
    {
      "name": "mode",
      "value": "server"
    },
    {
      "name": "directory",
```

```
        "value": "/srv/burp"
    },
],
"includes": [],
"includes_ext": [],
"integer": [
    {
        "name": "port",
        "value": 4971
    },
    {
        "name": "status_port",
        "value": 4972
    },
    {
        "name": "max_children",
        "value": 5
    },
    {
        "name": "max_status_children",
        "value": 5
    }
],
"multi": [
    {
        "name": "keep",
        "value": [
            "7",
            "4",
            "4"
        ]
    },
    {
        "name": "timer_arg",
        "value": [
            "12h",
            "Mon,Tue,Thu,Fri,17,18,19,20,21,22,23",
            "Wed,Sat,Sun,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23"
        ]
    }
],
]
```

remove_client (*client=None, keepconf=False, delcert=False, revoke=False*)

burpui.misc.parser.interface.BUIparser.remove_client() is used to delete a client from burp's configuration.

Parameters

- **client** (*str*) – The name of the client to remove
- **keepconf** – Whether to keep the conf (in order to just revoke/delete the cert)
- **keepconf** – bool
- **delcert** (*bool*) – Whether to delete the associated certificate
- **revoke** (*bool*) – Whether to revoke the associated certificate

Returns A list of notifications to return to the UI (success or failure)

server_initiated_backup (*name=None*)

burpui.misc.parser.interface.BUIparser.server_initiated_backup() called by *burpui.misc.backend.interface.BUIbackend.server_backup()* in order to create a server-initiated backup file.

Parameters *name* (*str*) – Client name

Returns A list of notifications to return to the UI (success or failure)

server_initiated_restoration (*name=None, backup=None, files=None, strip=None, force=None, prefix=None, restoreto=None*)

burpui.misc.parser.interface.BUIparser.server_initiated_restoration() called by *burpui.misc.backend.interface.BUIbackend.server_restore()* in order to create server-initiated restoration file.

Parameters

- **name** (*str*) – Client name
- **backup** (*int*) – Backup number
- **files** (*str*) – List of files to restore
- **strip** (*int*) – Number of leading path to strip
- **force** (*bool*) – Whether to force overriding files or not
- **prefix** (*str*) – Where to restore files
- **agent** (*str*) – What server to ask (only in multi-agent mode)

Returns A list of notifications to return to the UI (success or failure)

store_client_conf (*data, client=None, conf=None*)

burpui.misc.parser.interface.BUIparser.store_client_conf() is used by *burpui.misc.backend.BUIbackend.store_conf_cli()*.

It works the same way as *burpui.misc.parser.interface.BUIparser.store_conf()* with an extra parameter:

Parameters *client* (*str*) – Name of the client for which to apply this config

store_conf (*data, conf=None, client=None, mode='srv', insecure=False*)

burpui.misc.parser.interface.BUIparser.store_conf() is used to store the configuration from the web-ui into the actual configuration files. It is used by *burpui.misc.backend.BUIbackend.store_conf_srv()*.

Parameters

- **data** (*dict*) – Data sent by the web-form
- **conf** (*str*) – Force the file path (for file inclusions for instance)
- **client** (*str*) – Client name
- **mode** (*str*) – We actually use the same method for clients and server files
- **insecure** (*bool*) – Used for the CLI

Returns A list of notifications to return to the UI (success or failure)

Example:

```
[[0, "Success"]]
```

1.16.7 Auth

Here is the *auth* interface definition in order to implement a new authentication backend. It is composed by two classes.

class `burpui.misc.auth.interface.BUIhandler` (*app=None*)

The `burpui.misc.auth.interface.BUIhandler` class maintains a list of Burp-UI users.

Parameters `app` (`burpui.server.BUIServer`) – Instance of the app we are running in

user (*name=None*)

The `burpui.misc.auth.interface.BUIhandler.user()` function returns the `flask_login:flask_login.UserMixin` object corresponding to the given user name.

Parameters `name` (*str*) – Name of the user

Returns The corresponding user object

class `burpui.misc.auth.interface.BUIuser`

The `burpui.misc.auth.interface.BUIuser` class extends the `flask_login:flask_login.UserMixin` class.

is_active

Returns True if user is active, otherwise False

is_admin

If no ACL engine is loaded, every logged-in user will be granted admin rights :returns: True if the user is admin, otherwise False

is_authenticated

Returns True if a user is authenticated, otherwise False

login (*passwd=None*)

The `burpui.misc.auth.interface.BUIuser.login()` function checks if the provided user-name and password match.

Parameters `passwd` (*str*) – Password

Returns True if the name and password match, otherwise False

1.16.8 ACL

Here is the *acl* interface definition in order to implement a new acl backend. It is composed by two classes.

class `burpui.misc.acl.interface.BUIaclLoader` (*app=None*)

The `burpui.misc.acl.interface.BUIaclLoader` class is used to load the actual ACL backend

acl

Property to retrieve the backend

class `burpui.misc.acl.interface.BUIacl`

The `burpui.misc.acl.interface.BUIacl` class represents the ACL engine.

clients (*username=None, server=None*)

`burpui.misc.acl.interface.BUIacl.clients()` returns a list of allowed clients for a given user.

Parameters

- **username** (*str*) – Username to check

- **server** (*str*) – Server name. Used in multi-agent mode

Returns A list of clients

is_admin (*username=None*)

burpui.misc.acl.interface.BUIacl.is_admin() is used to know if a user has administrator rights.

Parameters **username** (*str*) – Username to check

Returns True if the user has admin rights, otherwise False

is_client_allowed (*username=None, client=None, server=None*)

burpui.misc.acl.interface.BUIacl.is_client_allowed() tells us if a given user has access to a given client on a given server.

Parameters

- **username** (*str*) – Username to check
- **client** (*str*) – Client to check
- **server** (*str*) – Server to check

Returns True if username is granted, otherwise False

servers (*username=None*)

burpui.misc.acl.interface.BUIacl.servers() returns a list of allowed servers for a given user.

Parameters **username** (*str*) – Username to check

Returns A list of servers

/api

GET /api/admin/auth/backends, 44	GET /api/clients/running, 48
GET /api/admin/auth/users, 44	GET /api/clients/running/ (client), 74
GET /api/admin/auth/users/ (name), 68	GET /api/clients/stats, 50
GET /api/admin/me, 51	GET /api/doc, 53
GET /api/admin/me/session, 45	GET /api/misc/ (server)/about, 82
GET /api/admin/me/session/ (id), 68	GET /api/misc/ (server)/counters, 81
GET /api/backup/ (server)/server-backup/ (name), 64	GET /api/misc/ (server)/counters/ (client), 68
GET /api/backup/server-backup/ (name), 77	GET /api/misc/ (server)/history, 82
GET /api/client/ (server)/browse/ (name) / (int:backup), 54	GET /api/misc/ (server)/history/ (client), 68
GET /api/client/ (server)/browseall/ (name) / (int:backup), 53	GET /api/misc/ (server)/monitor, 81
GET /api/client/ (server)/report/ (name), 65	GET /api/misc/about, 52
GET /api/client/ (server)/report/ (name) / (int:backup), 54	GET /api/misc/counters, 51
GET /api/client/ (server)/stats/ (name), 67	GET /api/misc/counters/ (client), 80
GET /api/client/browse/ (name) / (int:backup), 61	GET /api/misc/history, 52
GET /api/client/browseall/ (name) / (int:backup), 61	GET /api/misc/history/ (client), 80
GET /api/client/report/ (name), 77	GET /api/misc/monitor, 51
GET /api/client/report/ (name) / (int:backup), 62	GET /api/preferences/ui, 45
GET /api/client/stats/ (name), 80	GET /api/restore/ (server)/server-restore/ (name), 60
GET /api/clients/ (server)/all, 76	GET /api/restore/server-restore/ (name), 74
GET /api/clients/ (server)/backup-running, 74	GET /api/servers/report, 48
GET /api/clients/ (server)/report, 75	GET /api/servers/stats, 50
GET /api/clients/ (server)/running, 75	GET /api/settings/ (server)/clients, 74
GET /api/clients/ (server)/running/ (client), 60	GET /api/settings/ (server)/config/ (client), 59
GET /api/clients/ (server)/stats, 76	GET /api/settings/ (server)/config/ (client) / (path:conf), 53
GET /api/clients/all, 50	GET /api/settings/ (server)/options, 74
GET /api/clients/backup-running, 48	GET /api/settings/ (server)/path-expander, 73
GET /api/clients/report, 49	GET /api/settings/ (server)/path-expander/ (client), 57
	GET /api/settings/ (server)/server-config, 71
	GET /api/settings/ (server)/server-config/ (path:conf), 57
	GET /api/settings/clients, 47
	GET /api/settings/config/ (client), 71

```
GET /api/settings/config/(client)/(path:conf), 57 DELETE /api/restore/(server)/server-restore/(name), 60
GET /api/settings/options, 48 DELETE /api/restore/server-restore/(name), 74
GET /api/settings/path-expander, 47 74
GET /api/settings/path-expander/(client)DELETE /api/settings/(server)/config/(client), 59
69 59
GET /api/settings/server-config, 45 DELETE /api/settings/(server)/config/(client)/(path:conf), 53
GET /api/settings/server-config/(path:conf), 53
69 DELETE /api/settings/config/(client), 71
GET /api/swagger.json, 52 71
POST /api/admin/auth/users, 45 DELETE /api/settings/config/(client)/(path:conf), 57
POST /api/admin/auth/users/(name), 68 57
POST /api/misc/alert, 52
POST /api/preferences/ui, 45
POST /api/restore/(server)/archive/(name)/(int:backup), 53
53
POST /api/restore/archive/(name)/(int:backup), 60
60
POST /api/settings/(server)/config/(client), 59
59
POST /api/settings/(server)/config/(client)/(path:conf), 53
53
POST /api/settings/(server)/server-config, 73
73
POST /api/settings/(server)/server-config/(path:conf), 59
59
POST /api/settings/config/(client), 71
71
POST /api/settings/config/(client)/(path:conf), 57
57
POST /api/settings/server-config, 47
47
POST /api/settings/server-config/(path:conf), 71
71
PUT /api/admin/auth/users, 45
45
PUT /api/admin/auth/users/(name), 68
68
PUT /api/backup/(server)/do-server-backup/(name), 64
64
PUT /api/backup/do-server-backup/(name), 77
77
PUT /api/preferences/ui, 45
45
PUT /api/restore/(server)/server-restore/(name)/(int:backup), 53
53
PUT /api/restore/server-restore/(name)/(int:backup), 59
59
PUT /api/settings/(server)/config, 74
74
PUT /api/settings/config, 48
48
DELETE /api/admin/auth/users, 45
45
DELETE /api/admin/auth/users/(name), 68
68
DELETE /api/admin/me/session, 45
45
DELETE /api/admin/me/session/(id), 68
68
DELETE /api/backup/(server)/server-backup/(name), 65
65
DELETE /api/backup/server-backup/(name), 77
77
DELETE /api/preferences/ui, 45
45
```

A

acl (burpui.misc.acl.interface.BUIaclLoader attribute), 98

B

BUIacl (class in burpui.misc.acl.interface), 98

BUIaclLoader (class in burpui.misc.acl.interface), 98

BUIbackend (class in burpui.misc.backend.interface), 82

BUIhandler (class in burpui.misc.auth.interface), 98

BUIparser (class in burpui.misc.parser.interface), 93

BUIuser (class in burpui.misc.auth.interface), 98

C

cancel_backup() (burpui.misc.parser.interface.BUIparser method), 94

cancel_restore() (burpui.misc.parser.interface.BUIparser method), 94

cancel_server_backup() (burpui.misc.backend.interface.BUIbackend method), 83

cancel_server_restore() (burpui.misc.backend.interface.BUIbackend method), 83

clients() (burpui.misc.acl.interface.BUIacl method), 98

clients_list() (burpui.misc.backend.interface.BUIbackend method), 83

create_app() (in module burpui), 44

D

del_file() (burpui.misc.backend.interface.BUIbackend method), 83

delete_client() (burpui.misc.backend.interface.BUIbackend method), 83

E

expand_path() (burpui.misc.backend.interface.BUIbackend method), 84

G

get_all_clients() (burpui.misc.backend.interface.BUIbackend method), 84

get_attr() (burpui.misc.backend.interface.BUIbackend method), 84

get_backup_logs() (burpui.misc.backend.interface.BUIbackend method), 84

get_client() (burpui.misc.backend.interface.BUIbackend method), 87

get_client_filtered() (burpui.misc.backend.interface.BUIbackend method), 87

get_client_labels() (burpui.misc.backend.interface.BUIbackend method), 88

get_client_version() (burpui.misc.backend.interface.BUIbackend method), 88

get_clients_report() (burpui.misc.backend.interface.BUIbackend method), 88

get_counters() (burpui.misc.backend.interface.BUIbackend method), 88

get_file() (burpui.misc.backend.interface.BUIbackend method), 88

get_parser() (burpui.misc.backend.interface.BUIbackend method), 88

get_parser_attr() (burpui.misc.backend.interface.BUIbackend method), 88

get_server_version() (burpui.misc.backend.interface.BUIbackend method), 89

get_tree() (burpui.misc.backend.interface.BUIbackend method), 89

I

is_active (burpui.misc.auth.interface.BUIuser attribute), 98

is_admin (burpui.misc.auth.interface.BUIuser attribute), 98

is_admin() (burpui.misc.acl.interface.BUIacl method), 99

is_authenticated (burpui.misc.auth.interface.BUIuser attribute), 98

is_backup_running() (burpui.misc.backend.interface.BUIbackend method), 89

is_client_allowed() (burpui.misc.acl.interface.BUIacl method), 99

is_client_revoked() (burpui.misc.parser.interface.BUIparser method), 94

is_one_backup_running() (burpui.misc.backend.interface.BUIbackend method), 90

is_server_backup() (burpui.misc.backend.interface.BUIbackend method), 90

is_server_restore() (burpui.misc.backend.interface.BUIbackend method), 90

L

list_clients() (burpui.misc.parser.interface.BUIparser method), 94

login() (burpui.misc.auth.interface.BUIuser method), 98

P

param() (burpui.misc.parser.interface.BUIparser method), 94

path_expander() (burpui.misc.parser.interface.BUIparser method), 94

R

read_backup() (burpui.misc.parser.interface.BUIparser method), 94

read_client_conf() (burpui.misc.parser.interface.BUIparser method), 95

read_conf_cli() (burpui.misc.backend.interface.BUIbackend method), 90

read_conf_srv() (burpui.misc.backend.interface.BUIbackend method), 90

read_restore() (burpui.misc.parser.interface.BUIparser method), 95

read_server_conf() (burpui.misc.parser.interface.BUIparser method), 95

remove_client() (burpui.misc.parser.interface.BUIparser method), 96

restore_files() (burpui.misc.backend.interface.BUIbackend method), 92

revocation_enabled() (burpui.misc.backend.interface.BUIbackend method), 92

S

server_backup() (burpui.misc.backend.interface.BUIbackend method), 92

server_initiated_backup() (burpui.misc.parser.interface.BUIparser method), 96

server_initiated_restoration() (burpui.misc.parser.interface.BUIparser method), 97

server_restore() (burpui.misc.backend.interface.BUIbackend method), 92

servers() (burpui.misc.acl.interface.BUIacl method), 99

status() (burpui.misc.backend.interface.BUIbackend method), 93

store_client_conf() (burpui.misc.parser.interface.BUIparser method), 97

store_conf() (burpui.misc.parser.interface.BUIparser method), 97

store_conf_cli() (burpui.misc.backend.interface.BUIbackend method), 93

store_conf_srv() (burpui.misc.backend.interface.BUIbackend method), 93

U

user() (burpui.misc.auth.interface.BUIhandler method), 98