Burp-UI Documentation

Release 0.1.2

Ziirish

January 03, 2017

Contents

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.

CHAPTER 1

Documentation

1.1 Introduction

Burp-UI is a web-based interface for Burp. It's 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 Known Issues

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

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

1. SSL issue

My new SSL certificate seem to be unknown on older systems like debian wheezy. Thus, you may have some SSL failure while trying to clone my repository. In order to fix this error, you can run the following command as root that will add my certificate in your trust list:

```
echo -n | \
openssl s_client -showcerts -connect git.ziirish.me:443 \
-servername git.ziirish.me 2>/dev/null | \
sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >>/etc/ssl/certs/ca-certificates.crt
```

2. 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 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]"
```

1.2.2 LDAP

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

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

1.2.3 SSL

If you would like to use SSL, you will need the `python-openssl` package. On Debian:

```
aptitude install python-openssl
```

Alternatively, you can install the python package using the following command:

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

1.2.4 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 machine loopback (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.

1.2.5 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`.

On Debian, you can install `pip` with the following command:

```
aptitude install python-pip
```

Once `pip` is installed, you can install `Burp-UI` this way:

```
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 all interfaces (including IPv6) 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
```

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>]
               [-m <agent|server>]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         increase output verbosity (e.g., -vv is more than -v)
  -d, --debug           alias for -v
  -V, --version         print version and exit
  -c <CONFIG>, --config <CONFIG>
                        configuration file
  -l <FILE>, --logfile <FILE>
                        output logs in defined file
  -m <agent|server>, --mode <agent|server>
                        application mode (server or agent)
```

1.4 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.

1.4.1 Configuration

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

```
[Global]
# On which port is the application listening
port: 5000
# 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: false
# ssl cert
sslcert: /etc/burp/ssl_cert-server.pem
# ssl key
sslkey: /etc/burp/ssl_cert-server.key
# burp server version 1 or 2
version: 1
# 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
```

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

- *port*: On which port is Burp-UI listening. This option is ignored when using Gunicorn.
- *bind*: On which address is Burp-UI listening. This option is ignored when using Gunicorn.
- *ssl*: Whether to enable SSL or not. This option is ignored when using Gunicorn.
- *sslcert*: SSL certificate to use when SSL support is enabled.
- *sslkey*: SSL key to use when SSL support is enabled.
- *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.

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.

1.4.2 Production

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

```
[Production]
# storage backend (only used with gunicorn) for session and cache
# may be either 'default' or 'redis'
storage: default
# redis server to connect to
redis: localhost:6379
```

These settings are only used when Gunicorn is enabled and used.

1.4.3 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.

1.4.4 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.

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.

1.4.5 Authentication

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

- *LDAP*
- *Basic*

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

```
[Global]
auth: none
```

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]
# 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)

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]
admin: password
user1: otherpassword
```

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

1.4.6 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]
# Please note the double-quote around the username on the admin line are
# mandatory!
admin: ["user1","user2"]
# 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 are **mendatory**

1.5 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 gevent
pip install gunicorn
```

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

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

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

- conf: Path to the Burp-UI configuration file
- debug: Whether to run Burp-UI in debug mode or not to get some extra logging
- logfile: Path to a logfile in order to log Burp-UI internal messages

1.5.1 Daemon

If you wish to run Burp-UI as a daemon process, the recommanded 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:

```
apt-get install gunicorn
useradd -r -d /var/lib/burpui -c 'Burp-UI daemon user' burpui
mkdir /etc/burp
cp /usr/local/share/burpui/etc/burpui.sample.cfg /etc/burp/burpui.cfg
mkdir -p /var/log/gunicorn
chown -R burpui: /var/log/gunicorn
service gunicorn restart
```

1.5.2 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;
    }
}
```

1.5.3 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.

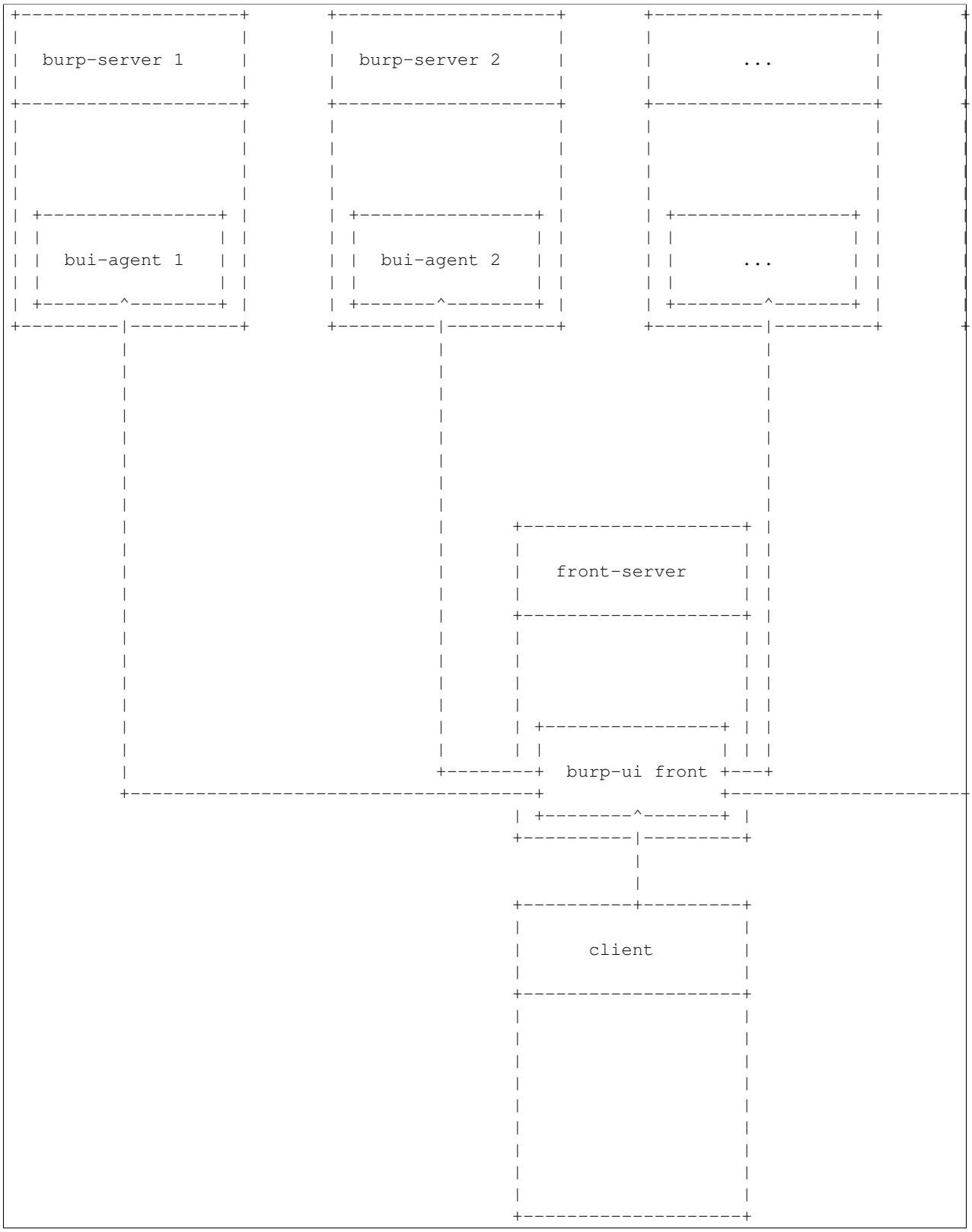
1.6 bui-agent

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.

1.6.1 Architecture

The architecture is described bellow:



1.6.2 Requirements

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

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

1.6.3 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
sslcert: /etc/burp/ssl_cert-server.pem
# ssl key
sslkey: /etc/burp/ssl_cert-server.key
# burp server version (currently only burp 1.x is implemented)
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.
- *sslcert*: 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.

1.6.4 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.

1.7 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.

1.7.1 Issues / Bugs

If you find issues using `Burp-UI` please report 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.

1.7.2 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.

1.7.3 Troubleshooting

In case you encounter troubles with `Burp-UI`, you should run it with the `-d` 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.

1.7.4 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.

1.7.5 Development

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

1.8 Changelog

1.8.1 0.1.2 (02/18/2016)

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

1.8.2 0.1.1 (02/17/2016)

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

1.8.3 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 the 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

1.8.4 0.0.7.3 (09/26/2015)

- Fix issue #77
- Doc

1.8.5 0.0.7.2 (09/01/2015)

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

1.8.6 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 Gunicorn
- 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
- Bufixes
- Full changelog

1.8.7 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

1.8.8 0.0.5 (09/22/2014)

- Add multi-server support
- Fix bugs
- Full changelog

1.8.9 0.0.4 (09/07/2014)

- Add the ability to download files directly from the web interface
- Full changelog

1.8.10 0.0.3 (09/02/2014)

- Add authentication
- Full changelog

1.8.11 0.0.2 (08/25/2014)

- Fix bugs
- Full changelog

1.8.12 0.0.1 (08/25/2014)

- Initial release

1.9 FAQ

1.9.1 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

1.9.2 How to start using Burp-UI?

You may find all the basic informations to get started with Burp-UI in the README file.

1.9.3 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 listens on port **10000** by default.

1.9.4 What are the default credentials?

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

1.9.5 How does the online restoration feature works?

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.

1.9.6 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*).

1.9.7 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.

1.9.8 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.

1.9.9 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.

1.9.10 Are there any known issues?

There is a known issue section in this documentation.

1.9.11 How can I contribute?

You can refer to the contributing section of this documentation.

1.10 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

1.10.1 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/clientconfdir/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.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.

1.10.2 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.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.

1.11 Developer Guide

1.11.1 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.

1.11.2 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
```

1.11.3 API

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

GET /api/settings/server-config
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-config

The `burpui.api.settings.ServerSettings` resource allows you to read and write the server's configuration.

This resource is part of the `burpui.api.settings` module.

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/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/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/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/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/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': '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/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

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*) **/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",
    "mode": "drwxr-xr-x",
    "name": "/",
    "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/ (*server*) **/server-config/**
path: *conf* **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* The `burpui.api.settings.ServerSettings` resource allows you to read and write the server's configuration.

This resource is part of the `burpui.api.settings` module.

PUT /api/restore/sserver-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/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/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",
    "mode": "drwxr-xr-x",
    "name": "/",
    "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.

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/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/server-config/** (**path:** *conf*)
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-config/ (**path:** *conf*)

The `burpui.api.settings.ServerSettings` resource allows you to read and write the server's configuration.

This resource is part of the `burpui.api.settings` module.

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

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**
The `burpui.api.settings.ServerSettings` resource allows you to read and write the server's configuration.

This resource is part of the `burpui.api.settings` module.

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/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

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

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

delete_client(*client=None*, *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
- **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*, *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
- **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_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_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_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*, *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
- **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": "",
```

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

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

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/clientconfdir/client1"
                },
                {
                    "name": "client2",
```

```
                    "value": "/etc/burp/clientconfdir/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

server_restore(*name=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

- **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*) – Wether 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)

set_logger(*logger*)

The *burpui.misc.backend.interface.BUIbackend.set_logger()* function is used to set the global logger of the application.

Parameters **logger** (*Logger*) – Logger object

status(*query='\n'*, *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
- **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"]]
```

1.11.5 Parser

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

class burpui.misc.parser.interface.**BUIparser**(*app=None*, *conf=None*)

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

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

path_expander(*pattern=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
- **client** (*str*) – The client name when working on client files

Returns A list of files or an empty list

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_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/clientconfdir/client1"
        },
        {
            "name": "client2",
            "value": "/etc/burp/clientconfdir/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*)

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

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*)

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*) – Wether 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*, *mode='srv'*)

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)
- **mode** (*str*) – We actually use the same method for clients and server files

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

Example:

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

1.11.6 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.

user(*name=None*)

The *burpui.misc.auth.interface.BUIhandler.user()* function returns the flask.ext.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.ext.login.UserMixin class.

login(*name=None*, *passwd=None*)

The *burpui.misc.auth.interface.BUIuser.login()* function checks if the profided username and password match.

Parameters

- **name** (*str*) – Username
- **passwd** (*str*) – Password

Returns True if the name and password match, otherwise False

1.11.7 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

S

U