In order to manage, add, and list information stored in faraday, we created *fplugin*, a simple plugin that allows you to interact directly with our Python API from the command line.

It gives Faraday powerful scripting features, and allows you to query the database without leaving your favourite workspace, be it the GTK interface, or a terminal.

Using our plugin located in `$faraday/bin/` you can do different actions from the command line

```
$ ./fplugin -h
fplugin --help
usage: fplugin [-h] [-i] [-w WORKSPACE] [-u URL] [command]

Using our plugin you can do different actions in the command line
and interact with Faraday. Faraday comes with some presets for bulk
actions such as object removal, get object information, etc.
Any parameter not recognized by fplugin, or everything after -- will be passed on
to the called script.

positional arguments:
  command               Command to execute. Example: ./fplugin getAllIps
                        (default: None)

optional arguments:
  -h, --help            show this help message and exit
  -i, --interactive     Run in interactive mode (default: False)
  -w WORKSPACE, --workspace WORKSPACE
                        Workspace to use (default: burp_pro_original)
  -u URL, --url URL     Faraday Server URL. Example: http://localhost:5985
                        (default: http://localhost:5985)

Available scripts:
    - change_vuln_status: Changes Vulns Status (to closed)
    - create_cred: Creates new credentials
    - create_host_and_interface: Creates a new host and interface in current
workspace
    - create_note: Creates a new note
    - create_service: Creates a new service in a specified interface
    - create_vuln: Creates a new vulnerability
    - create_vulnweb: Creates a new website vulnerability in a specified service
    - delAllHost: Deletes all stored hosts
    - delAllServiceClosed: Deletes all services with a non open port
    - delAllVulnsWith: Delete all vulnerabilities matched with regex
    - filter_services: Filter services by port or service name
    - getAllIpsInterfaces: Get all scanned interfaces
    - getSeverityByCwe: Get Vulns filtered by Severity and change Severity based in
CWE
    - import_csv: Import Faraday objects from CSV file
    - import_pcap: Import every host found in a PCAP file for further scanning
    - list_creds: Get all stored credentials
    - list_hosts: List hosts
    - list_ips: List all scanned IPs
    - list_os: Lists all scanned OSs
    - screenshot_server: Takes a Screenshot of the ip:ports of a given protocol
```

To view the help ofthe `fplugin`, you can use the `-h` or `--help` arguments. It is also possible to view the help of the individual commands, but as the arguments mentioned will be catched before they reach the command being called, you need to 'escape' them, like this:

```
$ ./fplugin create_host_and_interface
Creates a new host in current workspace and a new interface in the given host

positional arguments:
    host_name
    os
    interface_name
    mac

optional arguments:
-h --help
--ipv4address IPV4ADDRESS
--ipv4gateway IPV4GATEWAY
--ipv4mask IPV4MASK
--ipv4dns IPV4DNS
--ipv6address IPV6ADDRESS
--ipv6prefix IPV6PREFIX
--ipv6gateway IPV6GATEWAY
--ipv6dns IPV6DNS
--netsegment NETSEGMENT
--hostres HOSTRES
--dry-run

Everything after the `--` will be sent to the command, and will not be interpreted
by `fplugin`.
```

Usage Examples

Filter hosts by ports or services

The following command will list all running services exposed on common HTTP ports (services with ports 80, 8080, 443, 8443 open).

```
 $ ./fplugin filter_services http ssh -p 21 -a
Filtering services for ports: 21, 22, 80, 443, 8080, 8443
host            service         ports           protocol        status
host_os
---------------------------------------------------------------------------------------
-------------
192.168.20.7    ssh             22              tcp             open
Linux
192.168.20.7    http            443             tcp             open
Linux
192.168.20.15   upnp            80              tcp             open
Linux
192.168.20.22   ssh             22              tcp             open
```

```
Linux
192.168.20.48   ssh             22              tcp             open
None
192.168.20.123  ssh             22              tcp             open
Linux
192.168.20.123  http            443             tcp             open
Linux
192.168.20.11   ssh             22              tcp             open
Linux
192.168.20.11   http            80              tcp             open
Linux
192.168.20.11   http            443             tcp             open
Linux
```

Create a new host and interface

```
$ ./fplugin create_host_and_interface 192.154.33.22 Linux interface 76598709876
709381e970d2d669ee1d1b4844a6dde9d9b63c77.a47084649d94d1fb2f912872dfda906c59a623c4

$ echo $?
0

$ ./fplugin create_host_and_interface 192.154.33.22 Linux intname aa:bb:cc:dd:ee:ff
A host with ID 709381e970d2d669ee1d1b4844a6dde9d9b63c77 already exists!

$ echo $?
2
...
```

Interactive mode

This version of `fplugin` comes with an interactive mode which will help you quickly perform any of the available actions in a virtual interpreter.

```
 $ ./fplugin -i
Welcome to interactive Faraday!
Press CTRL-D to quit.
> |
```

The advantage of the interactive mode is that you can use the simple string `$last` to refer to the ID of the last added object.

For example:

```
$ ./fplugin create_host_and_interface 192.154.33.22 Linux interface 76598709876
709381e970d2d669ee1d1b4844a6dde9d9b63c77.a47084649d94d1fb2f912872dfda906c59a623c4
```

Additionaly, it has a command history of the last 1000 issued commands, for quick access. Just as with any terminal, you can cycle through it using the `UP` and `DOWN` arrow keys.

Available commands

Faraday comes with some presets for bulk actions such as object removal, etc. These are usually necessary when managing large Workspaces. The current presets are:

- `create_cred`: Creates new credentials
- `create_host`: Creates a new host in current workspace
- `create_interface`: Creates a new interface in a specified host
- `create_note`: Creates a new note
- `create_service`: Creates a new service in a specified interface
- `create_vuln`: Creates a new vulnerability
- `create_vulnweb`: Creates a new website vulnerability in a specified service
- `delAllHost`: Deletes all stored hosts
- `delAllServiceClosed`: Deletes all services with a non open port
- `delAllVulnsWith`: Delete all vulnerabilities matched with rege
- `filter_services`: Filter services by port
- `getAllIpsInterfaces`: Get all scanned interfaces
- `getExploits`: Get possible exploits from open services
- `getSeverityByCwe`: Get Vulns filtered by Severity and change Severity based in CWE
- `import_pcap`: Import every host found in a PCAP file for further scanning
- `list_creds`: Get all stored credentials
- `list_hosts`: List hosts
- `list_ips`: List all scanned IPs
- `list_os`: Lists all scanned OSs
- `change_vuln_status`: Changes all vulns status to close

Adding new commands

`fplugin` will scan the `bin` folder of the Faraday root, so adding a new command is as simple as creating a new Python2 file following this standard:

```
__description__ = 'A short command description
__prettyname__ = 'Command Name'


def main(workspace='', args=None, parser=None):
    pass
```

The `__description__` and `__prettyname__` variables will be dinamically extracted to build the available command list, and show valuable information in the help and GTK views.

The 3 parameters of the `main` function are detailed bellow:

- `workspace`: Workspace being worked on
- `args`: A Python list of arguments not parsed by `fplugin`. This corresponds to arguments passed on to the command. You will probably want to send them to the `parser` after adding the required arguments.
- `parser`: An ArgumentParser instance with pre-filled data about the command being executed. It is the task of the command to populate the parser with the optional or required arguments and call parser.parse_args to either print the help page and stop the execution, or to get a Namespace object with the parsed arguments. If no arguments are required, you can safely discard this argument. As `sys.argv` will contain additional arguments not needed by your command, you should pass the `args` list to the `parse_args` call.

The function should return a tuple with the exit code of the command (0 if execution finished without errors, ~0 otherwise), and, if an object was created, the ID of said object, or None in any other case.

Here is a simple example showing the `create_host` command:

```
def main(workspace='', args=None, parser=None):
    parser.add_argument('name', help='Host name')
    parser.add_argument('os', help='OS')

    parser.add_argument('--dry-run', action='store_true', help='Do not touch the
database. Only print the object ID')

    parsed_args = parser.parse_args(args)

    obj = factory.createModelObject(models.Host.class_signature, parsed_args.name,
                                    workspace, os=parsed_args.os, parent_id=None)

    old = models.get_host(workspace, obj.getID())

    if old is None:
        if not parsed_args.dry_run:
            models.create_host(workspace, obj)
    else:
        print "A host with ID %s already exists!" % obj.getID()
        return 2, None

    return 0, obj.getID()
```

As you can see, arguments are added to the `parser` object, and the `parser.parse_args` is called with the `args` argument passed on by `fplugin`

Additionaly, if an object (in this case a Host) is created, we return a value of 0, and the ID of the created Host. If a host with the same IP already exists, we return an error code of 2, and None.