Faraday's APIs

Faraday has 2 APIs on the Client: - An **RPC GTK API Service** by default running on 127.0.0.1:9876. - and a **RESTful GTK API Service** by default running on 127.0.0.1:9977.

and one API on the Server: - A **Server RESTful API** by default running on 127.0.0.1:5985

There are a number of examples on using this on our [[Faraday Plugin]] wiki page. There's further information on the persistance server documentation available in the *persistence/server/docs* directory.

Configuration

You can configure both ports and the IP address binded to them. As you will see, right now the only way to configure the RESTful GTK API is by hand (information below). This will be changed in the future.

Via the CLI, part of the current help shows us how to do it:

```
  -n HOST, --hostname HOST
                        The hostname where both server APIs will listen (XMLRPC and
RESTful).
                        Default = localhost
  -p PORT, --port PORT
                        Sets the port where the api XMLRPCServer will listen.
                        Default = 9876
```

So if you want to make your custom configuration, you can specify new ports and bind faraday on broadcast for example:

```
./faraday.py --hostname 0.0.0.0 --port 9999
```

There's also a shorter alias for each command flag:

```
./faraday.py -n 192.168.20.32 -p 9999
```

Manual configuration (persistent)

You can also modify the APIs configuration by hand, going to your config path of faraday, and editing the user.xml file.

Let's see an example. By default you have something like this (trimming to only the important elements):

```
<faraday>
  <api_con_info_host>127.0.0.1</api_con_info_host>
  <api_con_info_port>9884</api_con_info_port>
  <api_restful_con_info_port>9984</api_restful_con_info_port>
  <appname>Faraday - Penetration Test IDE Community</appname>
...
```

So if you want to bind the ip address to 0.0.0.0 and change the rest api to 8080, you just can edit it:

```
<faraday>
  <api_con_info_host>0.0.0.0</api_con_info_host>
  <api_con_info_port>9884</api_con_info_port>
  <api_restful_con_info_port>9984</api_restful_con_info_port>
  <appname>Faraday - Penetration Test IDE Community</appname>
...
```

Default configuration

If you want to return to the default configuration you may delete the 3 lines shown below.

```
<faraday>
  <api_con_info_host>127.0.0.1</api_con_info_host>
  <api_con_info_port>9884</api_con_info_port>
  <api_restful_con_info_port>9984</api_restful_con_info_port>
...
```

Faraday will detect that some of the configuration is missing and will use the default values specified by the launcher.

Client RPC API

The **RPC service** can be used by others tools to incorporate information directly into the database.

Let's see the following example to develop a Shodan tool with Faraday.
For this we are using Shodan's example code: https://shodan.readthedocs.org/

Shodan example:

```
import shodan

SHODAN_API_KEY = "insert your API key here"

api = shodan.Shodan(SHODAN_API_KEY)

# Wrap the request in a try/ except block to catch errors
try:
        # Search Shodan
        results = api.search('apache')
```

```
        # Show the results
        print 'Results found: %s' % results['total']
        for result in results['matches']:
                print 'IP: %s' % result['ip_str']
                print result['data']
                print ''
except shodan.APIError, e:
        print 'Error: %s' % e
```

Shodan with Faraday:

```
import shodan
SHODAN_API_KEY = "insert your API key here"
api = shodan.Shodan(SHODAN_API_KEY)
# Wrap the request in a try/ except block to catch errors
try:
# Search Shodan
        print "Search Shodan"
        results = api.search('apache')

        #Connect to faraday
        print "Connecting Farday"
        api = xmlrpclib.ServerProxy("http://127.0.0.1:9876/")

        # Show the results
        print 'Results found: %s' % results['total']
        for result in results['matches']:
                if "ip" in result:
                        print 'IP: %s' % result['ip_str']
                        print result['data']
                        print ''

                        h_id = api.createAndAddHost(result['ip_str'],result['os']
if result['os'] is not None else "")
                        i_id =
api.createAndAddInterface(h_id,result['ip_str'],"00:00:00:00:00:00",
result['ip_str'], "0.0.0.0", "0.0.0.0",[],

"0000:0000:0000:0000:0000:0000:0000:0000","00","0000:0000:0000:0000:0000:0000:0000:
0000",
                                        [],"",result['hostnames'] if result['hostnames']
is not None else [])
                        s_id = api.createAndAddServiceToInterface(h_id, i_id,
"www",

"tcp",str(result['port']),"open","Apache",result['data'])

except Exception, e:
        print 'Error: %s' % e
```

Congratulations! 5 lines of code and you have Shodan plugin working on Faraday!
You can see the finished tool in $faraday/scripts/shodan_faraday.py

Client RESTful API

When you start the CLI or GTK client it also starts a local RESTful API listening on localhost (so other devices connected to your network won't be able to use it), on port 9984 by default. You can check it's running in the output of `./faraday.py` or in the logs:

```
2017-07-07 17:41:47,541 - faraday - INFO - REST API server configured on
('localhost',9984)
```

In the future, this will allow developers to interact with the framework from external applications and not necessarily from plugins.

List of RESTful API endpoints

POST /model/edit/vulns

Edit a vulnerability

Fields

- name
- desc
- severity
- resolution
- refs

DELETE /model/del/vulns

Delete a vulnerability

Fields

- vulnid
- hostid

PUT /model/host

Create a host

Fields

- name (required)
- os

GET /model/webvulns

List web vulnerabilities

PUT /model/interface

Create an interface

Fields

- name (required)
- mac
- ipv6_address
- ipv4_mask
- ipv4_gateway
- ipv4_dns
- ipv6_address
- ipv6_prefix
- ipv6_gateway
- ipv6_dns
- network_segment
- hostname_resolution
- parent_id

PUT /model/service

Create a service

Fields

- name (required)
- protocol
- ports
- status
- version
- description
- parent_id

PUT /model/vuln

Create a vulnerability

Fields

- name (required)
- desc
- ref
- severity
- resolution
- parent_id

PUT /model/vulnweb

Create a web vulnerability

Fields

- name (required)
- desc
- ref
- severity
- resolution
- website
- path
- request
- response
- method
- pname
- params
- query
- category
- parent_id

PUT /model/note

Create a note

Fields

- name (required)
- text
- parent_id

PUT /model/cred

Create a credential

Fields

- username
- password
- parent_id

GET /status/check

Check the status of the API. Example response:

```
{
    "code": 200,
    "message": "Faraday API Status: OK"
}
```

Server RESTful API

This is used by the web and the client to connect to the server. It runs on http://localhost:5985 or wherever you configured Faraday server to listen.

The following endpoints are available:

- GET /_api/ws/**workspace_name**/hosts: Lists the hosts of a workspace
- GET /_api/ws: List all workspaces
- GET /_api/ws/**workspace_name**/summary: Get stats of a workspace (count of vulns, services, etc)
- GET /_api/ws/**workspace_name**: Get workspace details
- PUT /_api/ws/**workspace_name**: Create a workspace
- DELETE /_api/ws/**workspace_name**: Delete a workspace
- GET /_api/ws/**workspace_name**/services: List workspace's services
- GET /_api/ws/**workspace_name**/services/count: Count workspace's services
- GET /_api/ws/**workspace_name**/vulns: List workspace's vulns
- GET /_api/ws/**workspace_name**/vulns/count: Count workspace's vulns
- GET /_api/ws/**workspace_name**/notes: List workspace's notes
- GET /_api/ws/**workspace_name**/notes/count: Count workspace's notes
- GET /_api/ws/**workspace_name**/interfaces: List workspace's interfaces
- GET /_api/ws/**workspace_name**/commands: List workspace's commands
- GET /_api/ws/**workspace_name**/credentials: List workspace's credentials
- GET /_api/ws/**workspace_name**/doc/**doc_id**: Get a generic object
- PUT /_api/ws/**workspace_name**/doc/**doc_id**: Update or create a generic object
- DELETE /_api/ws/**workspace_name**/doc/**doc_id**: Delete a generic object