Plugin development is really simple.

All the plugins are in:

```
$ faraday/plugins/repo/[pluginname]
```

In the following example you can see plugin of command ping:

```
    from plugins import core
    from model import api
    import re
    class CmdPingPlugin(core.PluginBase):
        """
        This plugin handles ping command.
        Basically detects if user was able to connect to a device
        """
        def __init__(self):
            core.PluginBase.__init__(self)
            self.id              = "ping"
            self.name            = "Ping"
            self.plugin_version         = "0.0.1"
            self.version  = "1.0.0"
            self._command_regex  = re.compile(r'^(sudo ping|ping|sudo
ping6|ping6).*?')
            self._completition = {
                                "":"[-LRUbdfnqrvVaAB]  [-c  count]  [-m  mark]  [-i
interval] ...",
                                    "-a":"Audible ping.",
                                    #...
                                  }

        def parseOutputString(self, output, debug = False):
            """
            This method will be called when the command finished executing and
            the complete output will be received to work with it
            Using the output the plugin can create and add hosts, interfaces,
services, etc.
            """
            reg=re.search(r"PING ([\w\.-:]+)( |)\(([\w\.:]+)\)", output)
            if re.search("0 received|unknown host",output) is None and reg is not
None:

                ip_address = reg.group(3)
                hostname=reg.group(1)
                h_id = self.createAndAddHost(ip_address)
                i_id = self.createAndAddInterface(h_id, ip_address,
ipv4_address=ip_address, hostname_resolution=[hostname])

            return True

        def processCommandString(self, username, current_path, command_string):
            """
            With this method a plugin can add aditional arguments to the command
that
```

```
            it's going to be executed.
            """
            return None

    def createPlugin():
    return CmdPingPlugin()
```

In this plugin if the host is active we add it to the database

Key information:

```
    self._command_regex
```

This is a regex used to match the command string and determine if the plugin is suitable to handle it.

```
    self._completition
```

This have the dict used for intellisense.

```
    def parseOutputString(self, output, debug = False):
```

This method will be called when the command finished executing and the complete output will be received to work with it. Using the output, the plugin can create and add hosts, interfaces, services, vuln, webvuln, credentials, notes.

```
    def processCommandString(self, username, current_path, command_string):
```

With this method a plugin can add additional arguments to the command that it's going to be executed.

```
    createAndAddHost(self, name, os = "unknown", category = None, update = False,
old_hostname = None)
```

With this method we can create and add a host to the database.

```
    createAndAddInterface(self, host_id, name = "", mac = "00:00:00:00:00:00",
                 ipv4_address = "0.0.0.0", ipv4_mask = "0.0.0.0",
                 ipv4_gateway = "0.0.0.0", ipv4_dns = [],
                 ipv6_address = "0000:0000:0000:0000:0000:0000:0000:0000",
ipv6_prefix = "00",
                 ipv6_gateway = "0000:0000:0000:0000:0000:0000:0000:0000", ipv6_dns
= [],
                 network_segment = "", hostname_resolution = [])
```

With this method we can create and add a interface to a host.

core.PluginBase

Besides the two methods above, this is the complete list of methods in the PluginBase:

```
    def createAndAddServiceToInterface(self, host_id, interface_id, name, protocol
= "tcp?",
                ports = [], status = "running", version = "unknown", description =
""):

    def createAndAddServiceToHost(self, host_id, protocol="tcp?", status="open",
                version="unknown", description=""):

    def createAndAddVulnToHost(self, host_id, name, desc="", ref=[], severity=""):

    def createAndAddVulnToInterface(self, host_id, interface_id, name, desc="",
ref=[], severity=""):

    def createAndAddVulnToService(self, host_id, service_id, name, desc="", ref=[],
severity=""):

    def createAndAddVulnWebToService(self, host_id, service_id, name, desc="",
ref=[], severity="",
                website="", path="", request="", response="", method="", pname="",
                params="", query="", category=""):

    def createAndAddNoteToHost(self, host_id, name, text):

    def createAndAddNoteToInterface(self, host_id, interface_id, name, text):

    def createAndAddNoteToService(self, host_id, service_id, name, text):

    def createAndAddNoteToNote(self, host_id, service_id, note_id, name, text):

    def createAndAddCredToService(self, host_id, service_id, username, password):

    def log(self, msg, level='INFO'):

    def devlog(self, msg):
```