Product ▾Solutions ▾Resources ▾Open Source ▾Enterprise ▾Pricing

🔍

Sign in

Sign up

📄lanmaster53 / recon-ngPublic

🔔Notifications


 Fork637

 Star4k

<> Code🕒 Issues12🔗 Pull requests8🔄 Actions📖 Wiki🛡 Security📈 Insights

recon-ng / recon / core / base.py📄

...

lanmaster53Added the phone column to the contacts table. Closes #353718d52 · 5 years ago🕒 History

```
1  __author__      = 'Tim Tomes (@lanmaster53)'\n2\n3  from datetime import datetime\n4  from pathlib import Path\n5  from urllib.parse import urljoin\n6  import errno\n7  import imp\n8  import json\n9  import os\n10 import random\n11 import re\n12 import shutil\n13 import sys\n14 import yaml\n15 import builtins\n16\n17 # import framework libs\n18 from recon.core import framework\n19 from recon.core.constants import BANNER, BANNER_SMALL\n20\n21 # set the __version__ variable based on the VERSION file\n22 exec(open(os.path.join(Path(os.path.abspath(__file__)).parents[2], 'VERSION')).read())\n23\n24 # using stdout to spool causes tab complete issues\n25 # therefore, override print function\n26 # use a lock for thread safe console and spool output\n27 from threading import Lock\n28 _print_lock = Lock()\n29 # spooling system\n30 def spool_print(*args, **kwargs):\n31     with _print_lock:\n32         if framework.Framework._spool:\n33             framework.Framework._spool.write(f"{args[0]}{os.linesep}")\n34             framework.Framework._spool.flush()\n35         # disable terminal output for server jobs\n36         if framework.Framework._mode == Mode.JOB:\n37             return\n38         # new print function must still use the old print function via the backup\n39         builtins._print(*args, **kwargs)\n40 # make a builtin backup of the original print function\n41 builtins._print = print\n42 # override the builtin print function with the new print function\n43 builtins.print = spool_print\n44\n45 #=====\n46 # BASE CLASS\n47 #=====\n48\n49 class Recon(framework.Framework):\n50\n51     repo_url = 'https://raw.githubusercontent.com/lanmaster53/recon-ng-modules/master/'\n52\n53     def __init__(self, check=True, analytics=True, marketplace=True, accessible=False):\n54         framework.Framework.__init__(self, 'base')\n55         self._name = 'recon-ng'\n56         self._prompt_template = '{}[{}] > '\n57         self._base_prompt = self._prompt_template.format('/', self._name)
```

Files

9e907df

Go to file

recon

core

web

__init__.py

base.py

constants.py

framework.py

module.py

tasks.py

mixins

utils

__init__.py

.gitignore

Dockerfile

LICENSE

README.md

REQUIREMENTS

VERSION

docker-compose.yml

recon-cli

recon-ng

recon-web

```
57         self._base_prompt = self._prompt_template.format( , self._name)
58         # set toggle flags
59         self._check = check
60         self._analytics = analytics
61         self._marketplace = marketplace
62         self._accessible = accessible
63         # set path variables
64         self.app_path = framework.Framework.app_path = sys.path[0]
65         self.core_path = framework.Framework.core_path = os.path.join(self.app_path, 'c
66         self.home_path = framework.Framework.home_path = os.path.join(os.path.expanduse
67         self.mod_path = framework.Framework.mod_path = os.path.join(self.home_path, 'mo
68         self.data_path = framework.Framework.data_path = os.path.join(self.home_path, '
69         self.spaces_path = framework.Framework.spaces_path = os.path.join(self.home_pat
70
71     def start(self, mode, workspace='default'):
72         # initialize framework components
73         self._mode = framework.Framework._mode = mode
```

recon-ng / recon / core / base.py

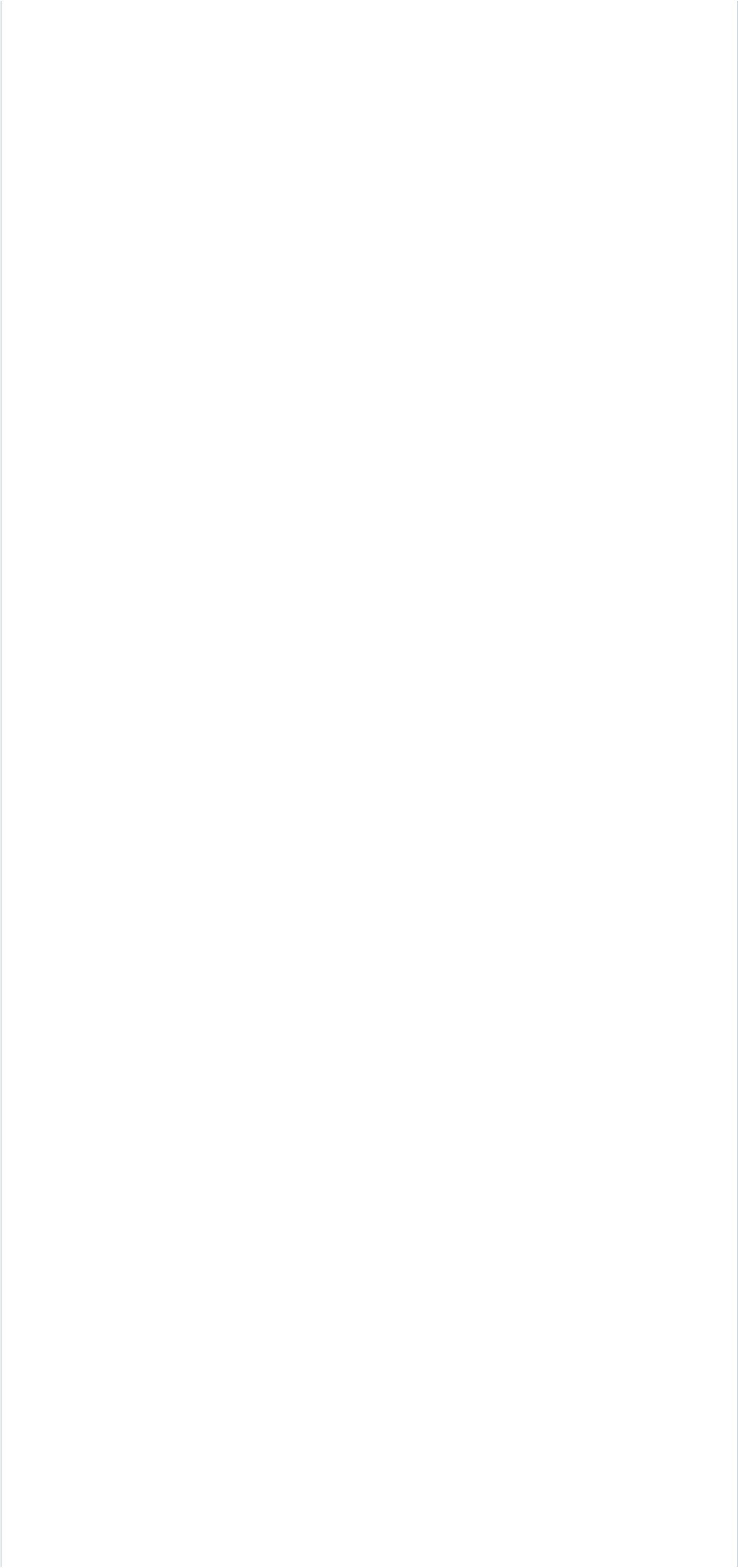
CodeBlame905 lines (819 loc) · 40.2 KB

RawCopyDownloadCode

```
71     def start(self, mode, workspace='default'):
72         self._print_banner()
73
74         self.cmdloop()
75
76
77
78
79
80
81
82         #=====
83         # SUPPORT METHODS
84         #=====
85
86     def _init_global_options(self):
87         self.options = self._global_options
88         self.register_option('nameserver', '8.8.8.8', True, 'default nameserver for the
89         self.register_option('proxy', None, False, 'proxy server (address:port)')
90         self.register_option('threads', 10, True, 'number of threads (where applicable)
91         self.register_option('timeout', 10, True, 'socket timeout (seconds)')
92         self.register_option('user-agent', f"Recon-ng/v{__version__.split('.')[0]}", Tr
93         self.register_option('verbosity', 1, True, 'verbosity level (0 = minimal, 1 = v
94
95     def _init_home(self):
96         # initialize home folder
97         if not os.path.exists(self.home_path):
98             os.makedirs(self.home_path)
99         # initialize keys database
100         self._query_keys('CREATE TABLE IF NOT EXISTS keys (name TEXT PRIMARY KEY, value
101         # initialize module index
102         self._fetch_module_index()
103
104     def _check_version(self):
105         if self._check:
106             pattern = r"(\d+\.\d+\.\d+[^']*)"
107             remote = 0
108             local = 0
109             try:
110                 remote = re.search(pattern, self.request('GET', 'https://raw.githubusercontent
111                 local = re.search(pattern, open('VERSION').read()).group(1)
112             except Exception as e:
113                 self.error(f"Version check failed ({type(e).__name__}).")
114                 #self.print_exception()
115             if remote != local:
116                 self.alert('Your version of Recon-ng does not match the latest release.
117                 self.alert('Please consider updating before further use.')
118                 self.output(f"Remote version: {remote}")
119                 self.output(f"Local version: {local}")
120             else:
121                 self.alert('Version check disabled.')
122
123     def _print_banner(self):
124         banner = BANNER
125         banner_len = len(max(banner.split(os.linesep), key=len))
126         author = '{0:^{1}}'.format(f"{framework.Colors.0}{self._name} v{__version__}",
127         if self._accessible:
128             banner = BANNER_SMALL
129             author = f"{framework.Colors.0}{self._name}, version {__version__}, by {__a
130         print(banner)
131         print(author)
```

```
132         print('')
133         counts = [(len(self._loaded_category[x]), x) for x in self._loaded_category]
134         if counts:
135             count_len = len(max([self.to_unicode_str(x[0]) for x in counts], key=len))
136             for count in sorted(counts, reverse=True):
137                 cnt = f"[{count[0]}]"
138                 print(f"{framework.Colors.B}{cnt.ljust(count_len+2)} {count[1].title()}")
139                 # create dynamic easter egg command based on counts
140                 setattr(self, f"do_{count[0]}", self._menu_egg)
141         else:
142             self.alert('No modules enabled/installed.')
143         print('')
144
145     def _send_analytics(self, cd):
146         if self._analytics:
147             try:
148                 cid_path = os.path.join(self.home_path, '.cid')
149                 if not os.path.exists(cid_path):
150                     # create the cid and file
151                     import uuid
152                     with open(cid_path, 'w') as fp:
153                         fp.write(self.to_unicode_str(uuid.uuid4()))
154                 with open(cid_path) as fp:
155                     cid = fp.read().strip()
156                 params = {
157                     'v': 1,
158                     'tid': 'UA-52269615-2',
159                     'cid': cid,
160                     't': 'screenview',
161                     'an': 'Recon-ng',
162                     'av': __version__,
163                     'cd': cd
164                 }
165                 self.request('GET', 'https://www.google-analytics.com/collect', params=
166             except Exception as e:
167                 self.debug(f"Analytics failed ({type(e).__name__}).")
168                 #self.print_exception()
169                 return
170         else:
171             self.debug('Analytics disabled.')
172
173     def _menu_egg(self, params):
174         eggs = [
175             'Really? A menu option? Try again.',
176             'You clearly need \'help\'',
177             'That makes no sense to me.',
178             '*grunt* *grunt* Nope. I got nothin\'',
179             'Wait for it...',
180             'This is not the Social Engineering Toolkit.',
181             'Don\'t you think if that worked the numbers would at least be in order?',
182             'Reserving that option for the next-NEXT generation of the framework.',
183             'You\'ve clearly got the wrong framework. Attempting to start SET...',
184             '1980 called. They want there menu driven UI back.',
185         ]
186         print(random.choice(eggs))
187         return
188
189     #=====
190     # WORKSPACE METHODS
191     #=====
192
193     def _init_workspace(self, workspace):
194         if not workspace:
195             return
196         path = os.path.join(self.spaces_path, workspace)
197         self.workspace = framework.Framework.workspace = path
198         if not os.path.exists(path):
199             os.makedirs(path)
200             self._create_db()
201         else:
202             self._migrate_db()
203         # set workspace prompt
204         self.prompt = self._prompt_template.format(self._base_prompt[:-3], self.workspa
205         # load workspace configuration
206         self.load_config()
```

```
...
207         # reload modules after config to populate options
208         self._load_modules()
209         return True
```






```
832
833 #=====
834 # COMPLETE METHODS
835 #=====
836
837 def complete_index(self, text, line, *ignored):
838     if len(line.split(' ')) == 2:
839         return [x for x in self._loaded_modules if x.startswith(text)]
840     return []
841
842 def complete_marketplace(self, text, line, *ignored):
843     arg, params = self._parse_params(line.split(' ', 1)[1])
844     subs = self._parse_subcommands('marketplace')
845     if arg in subs:
846         return getattr(self, '_complete_marketplace_'+arg)(text, params)
847     return [sub for sub in subs if sub.startswith(text)]
848
849 def _complete_marketplace_refresh(self, text, *ignored):
850     return []
851 _complete_marketplace_search = _complete_marketplace_refresh
852
853 def _complete_marketplace_info(self, text, *ignored):
854     return [x['path'] for x in self._module_index if x['path'].startswith(text)]
855 _complete_marketplace_install = _complete_marketplace_info
856
857 def _complete_marketplace_remove(self, text, *ignored):
858     return [x['path'] for x in self._module_index if x['status'] == 'installed' and
859
860 def complete_workspaces(self, text, line, *ignored):
861     arg, params = self._parse_params(line.split(' ', 1)[1])
862     subs = self._parse_subcommands('workspaces')
863     if arg in subs:
864         return getattr(self, '_complete_workspaces_'+arg)(text, params)
865     return [sub for sub in subs if sub.startswith(text)]
866
867 def _complete_workspaces_list(self, text, *ignored):
868     return []
869 _complete_workspaces_create = _complete_workspaces_list
870
871 def _complete_workspaces_load(self, text, *ignored):
872     return [x for x in self._get_workspaces() if x.startswith(text)]
873 _complete_workspaces_remove = _complete_workspaces_load
874
875 def complete_snapshots(self, text, line, *ignored):
876     arg, params = self._parse_params(line.split(' ', 1)[1])
877     subs = self._parse_subcommands('snapshots')
```

```
878         if arg in subs:
879             return getattr(self, '_complete_snapshots_'+arg)(text, params)
880         return [sub for sub in subs if sub.startswith(text)]
881
882     def _complete_snapshots_list(self, text, *ignored):
883         return []
884     _complete_snapshots_take = _complete_snapshots_list
885
886     def _complete_snapshots_load(self, text, *ignored):
887         return [x for x in self._get_snapshots() if x.startswith(text)]
888     _complete_snapshots_remove = _complete_snapshots_load
889
890     def _complete_modules_reload(self, text, *ignored):
891         return []
892
893     #=====
894     # SUPPORT CLASSES
895     #=====
896
897     class Mode(object):
898         '''Contains constants that represent the state of the interpreter.'''
899         CONSOLE = 0
900         CLI     = 1
901         WEB     = 2
902         JOB     = 3
903
904     def __init__(self):
905         raise NotImplementedError('This class should never be instantiated.')
```