



Sign in

This repository has been archived by the owner on Jan 29, 2020. It is now read-only.

EmpireProject / Empire Public archive

Notifications

Fork 2.8k

Star 7.4k

Code

Issues 64

Pull requests 37

Actions

Projects

Wiki

Security

Insights

Empire / lib / common / helpers.py



672 lines (515 loc) · 20.8 KB

Code Blame

Raw



```
1  """
2
3  Misc. helper functions used in Empire.
4
5  Includes the PowerShell functions that generate the
6  randomized stagers.
7
8  """
9
10 import re, string, commands, base64, binascii, sys, os, socket, sqlite3, iptools
11 from time import localtime, strftime
12 from Crypto.Random import random
13
14
15 #####
16 #
17 # Validation methods
18 #
19 #####
20
21 def validate_hostname(hostname):
22     """
23     Tries to validate a hostname.
24     """
```

```
25         if len(hostname) > 255: return False
26         if hostname[-1:] == ".": hostname = hostname[:-1]
27         allowed = re.compile("(?!-)[A-Z\d-]{1,63}(?!-)$", re.IGNORECASE)
28         return all(allowed.match(x) for x in hostname.split("."))
29
30
31     def validate_ip(IP):
32         """
33         Uses iptools to validate an IP.
34         """
35         return iptools.ipv4.validate_ip(IP)
36
37
38     def validate_ntlm(data):
39
40         allowed = re.compile("[0-9a-f]{32}", re.IGNORECASE)
41         if allowed.match(data):
42             return True
43         else:
44             return False
45
46
47     def generate_ip_list(s):
48         """
49         Takes a comma separated list of IP/range/CIDR addresses and
50         generates an IP range list.
51         """
52
53         # strip newlines and make everything comma separated
54         s = ",".join(s.splitlines())
55         # strip out spaces
56         s = ",".join(s.split(" "))
57
58         ranges = ""
59         if s and s != "":
60             parts = s.split(",")
61
62             for part in parts:
63                 p = part.split("-")
64                 if len(p) == 2:
65                     if iptools.ipv4.validate_ip(p[0]) and iptools.ipv4.validate_ip(p[1]):
66                         ranges += "("+str(p[0])+", "+str(p[1])+"),"
67                 else:
68                     if "/" in part and iptools.ipv4.validate_cidr(part):
69                         ranges += "'"+str(p[0])+"',"
70                     elif iptools.ipv4.validate_ip(part):
```

```
71         ranges += "'" + str(p[0]) + "'", "
72
73     if ranges != "":
74         return eval("ipertools.IpRangelist("+ranges+")")
75     else:
76         return None
77
78 else:
79     return None
80
81
82 #####
83 #
84 # Randomizers/obfuscators
85 #
86 #####
87
88 ✓ def random_string(length=-1, charset=string.ascii_letters):
89     """
90     Returns a random string of "length" characters.
91     If no length is specified, resulting string is in between 6 and 15 characters.
92     A character set can be specified, defaulting to just alpha letters.
93     """
94     if length == -1: length = random.randrange(6,16)
95     random_string = ''.join(random.choice(charset) for x in range(length))
96     return random_string
97
98
99 ✓ def obfuscate_num(N, mod):
100     """
101     Take a number and modulus and return an obsucfated form.
102
103     Returns a string of the obfuscated number N
104     """
105     d = random.randint(1, mod)
106     left = int(N/d)
107     right = d
108     remainder = N % d
109     return "%s*%s+%s" % (left, right, remainder)
110
111
112 ✓ def randomize_capitalization(data):
113     """
114     Randomize the capitalization of a string.
115     """
116     return "".join( random.choice([k.upper(), k ]) for k in data )
```

117


```
599         marker = idfun(item)
600         # in old Python versions:
601         # if seen.has_key(marker)
602         # but in new ones:
603         if marker in seen: continue
604         seen[marker] = 1
605         result.append(item)
606     return result
607
608
609     def uniquify_tuples(tuples):
610         # uniquify mimikatz tuples based on the password
611         # cred format- (credType, domain, username, password, hostname, sid)
612         seen = set()
613         return [item for item in tuples if "%s%s%s%s"%(item[0],item[1],item[2],item[3]) not in seen and
614
615
616     def decode_base64(data):
617         """
618         Try to decode a base64 string.
619         From http://stackoverflow.com/questions/2941995/python-ignore-incorrect-padding-error-when-base
620         """
```

```
621     missing_padding = 4 - len(data) % 4
622     if missing_padding:
623         data += b' ' * missing_padding
624
625     try:
626         result = base64.decodestring(data)
627         return result
628     except binascii.Error:
629         # if there's a decoding error, just return the data
630         return data
631
632
633  ✓ def encode_base64(data):
634     """
635     Decode data as a base64 string.
636     """
637     return base64.encodestring(data).strip()
638
639
640  ✓ def complete_path(text, line, arg=False):
641     """
642     Helper for tab-completion of file paths.
643     """
644
645     # stolen from dataq at
646     # http://stackoverflow.com/questions/16826172/filename-tab-completion-in-cmd-cmd-of-python
647
648     if arg:
649         # if we have "command something path"
650         argData = line.split()[1:]
651     else:
652         # if we have "command path"
653         argData = line.split()[0:]
654
655     if not argData or len(argData) == 1:
656         completions = os.listdir('.')
657     else:
658         dir, part, base = argData[-1].rpartition('/')
659         if part == '':
660             dir = './'
661         elif dir == '':
662             dir = '/'
663
664         completions = []
665         for f in os.listdir(dir):
666             # if f starts with part, then it's a match
```

```
666         if f.startswith(base):
667             if os.path.isfile(os.path.join(dir,f)):
668                 completions.append(f)
669             else:
670                 completions.append(f+'/')
671
672     return completions
```