Empire / lib / common / helpers.py

672 lines (515 loc) · 20.8 KB

```
"""

Misc. helper functions used in Empire.

Includes the PowerShell functions that generate the
randomized stagers.

"""

import re, string, commands, base64, binascii, sys, os, socket, sqlite3, iptools
from time import localtime, strftime
from Crypto.Random import random


###############################################################
#
# Validation methods
#
###############################################################

def validate_hostname(hostname):
    """
    Tries to validate a hostname.
    """
    if len(hostname) > 255: return False
    if hostname[-1:] == ".": hostname = hostname[:-1]
    allowed = re.compile("(?!-)[A-Z\d-]{1,63}(?<!-)$", re.IGNORECASE)
    return all(allowed.match(x) for x in hostname.split("."))


def validate_ip(IP):
    """
    Uses iptools to validate an IP.
    """
    return iptools.ipv4.validate_ip(IP)


def validate_ntlm(data):

    allowed = re.compile("^[0-9a-f]{32}", re.IGNORECASE)
    if allowed.match(data):
        return True
    else:
        return False


def generate_ip_list(s):
    """
    Takes a comma separated list of IP/range/CIDR addresses and
    generates an IP range list.
    """

    # strip newlines and make everything comma separated
    s = ",".join(s.splitlines())
    # strip out spaces
    s = ",".join(s.split(" "))

    ranges = ""
    if s and s != "":
        parts = s.split(",")

        for part in parts:
            p = part.split("-")
            if len(p) == 2:
                if iptools.ipv4.validate_ip(p[0]) and iptools.ipv4.validate_ip(p[1]):
                    ranges += "('"+str(p[0])+"', '"+str(p[1])+"'),"
            else:
                if "/" in part and iptools.ipv4.validate_cidr(part):
                    ranges += "'"+str(p[0])+"',"
                elif iptools.ipv4.validate_ip(part):
```

```python
                    ranges += "'"+str(p[0])+"',"

        if ranges != "":
            return eval("iptools.IpRangeList("+ranges+")")
        else:
            return None

    else:
        return None


###############################################################################
#
# Randomizers/obfuscators
#
###############################################################################

def random_string(length=-1, charset=string.ascii_letters):
    """
    Returns a random string of "length" characters.
    If no length is specified, resulting string is in between 6 and 15 characters.
    A character set can be specified, defaulting to just alpha letters.
    """
    if length == -1: length = random.randrange(6,16)
    random_string = ''.join(random.choice(charset) for x in range(length))
    return random_string


def obfuscate_num(N, mod):
    """
    Take a number and modulus and return an obsucfated form.

    Returns a string of the obfuscated number N
    """
    d = random.randint(1, mod)
    left = int(N/d)
    right = d
    remainder = N % d
    return "(%s*%s+%s)" %(left, right, remainder)


def randomize_capitalization(data):
    """
    Randomize the capitalization of a string.
    """
    return "".join( random.choice([k.upper(), k ]) for k in data )
```

```
            marker = idfun(item)
            # in old Python versions:
            # if seen.has_key(marker)
            # but in new ones:
            if marker in seen: continue
            seen[marker] = 1
            result.append(item)
        return result


    def uniquify_tuples(tuples):
        # uniquify mimikatz tuples based on the password
        # cred format- (credType, domain, username, password, hostname, sid)
        seen = set()
        return [item for item in tuples if "%s%s%s%s"%(item[0],item[1],item[2],item[3]) not in seen and


    def decode_base64(data):
        """
        Try to decode a base64 string.
        From http://stackoverflow.com/questions/2941995/python-ignore-incorrect-padding-error-when-base
        """
        missing_padding = 4 - len(data) % 4
        if missing_padding:
            data += b'='* missing_padding

        try:
            result = base64.decodestring(data)
            return result
        except binascii.Error:
            # if there's a decoding error, just return the data
            return data


    def encode_base64(data):
        """
        Decode data as a base64 string.
        """
        return base64.encodestring(data).strip()


    def complete_path(text, line, arg=False):
        """
        Helper for tab-completion of file paths.
        """

        # stolen from dataq at
        #   http://stackoverflow.com/questions/16826172/filename-tab-completion-in-cmd-cmd-of-python

        if arg:
            # if we have "command something path"
            argData = line.split()[1:]
        else:
            # if we have "command path"
            argData = line.split()[0:]

        if not argData or len(argData) == 1:
            completions = os.listdir('./')
        else:
            dir, part, base = argData[-1].rpartition('/')
            if part == '':
                dir = './'
            elif dir == '':
                dir = '/'

            completions = []
            for f in os.listdir(dir):
                if f.startswith(base):
                    if os.path.isfile(os.path.join(dir,f)):
                        completions.append(f)
                    else:
                        completions.append(f+'/')

        return completions
```