Python decryptor for newer AdWind config file - replicated from this Java version https://github.com/mhelwig/adwind-decryptor

By herrcore

Archived: 2026-04-05 16:22:20 UTC

```
#!/usr/local/bin/env python

##############################################################################################################
##
## Decrypts the AdWind configiration files!
## ** May also work for other files **
##
##
## All credit to Michael Helwig for the original Java implementation:
## https://github.com/mhelwig/adwind-decryptor
##
## See his blog here:
## https://www.codemetrix.net/decrypting-adwind-jrat-jbifrost-trojan/
##
##
## Author: @herrcore
##
##############################################################################################################

# pip install javaobj-py3 not javaobj
try:
import javaobj
except:
print "You need to install javaobj-py3... try pip install javaobj-py3"

from Crypto.Cipher import AES
from Crypto.PublicKey import RSA
```

```
import argparse
import sys

def __read_file(file_path):
with open(file_path, "rb") as fp:
data = fp.read()
if not data:
print "Error: file %s could not be read" % file_path
sys.exit(-1)
return data


def main():
parser = argparse.ArgumentParser(description="Decrypt AdWind configuration files.")
requiredNamed = parser.add_argument_group('required named arguments')
requiredNamed.add_argument('--rsa_file', dest="rsa_file", default=None, help="Specify path to the serialized RSA KeyRep file
required=True)
requiredNamed.add_argument('--aes_file', dest="aes_file", default=None, help="Specify path to the AES file (RSA encrypted)"
required=True)
requiredNamed.add_argument('--config_file', dest="config_file", default=None, help="Specify path to the encrypted config file
required=True)
args = parser.parse_args()

rsa_data = __read_file(args.rsa_file)
aes_data = __read_file(args.aes_file)
config_data = __read_file(args.config_file)

# deserialize the KeyRep RSA file
pobj = javaobj.loads(rsa_data)
# extract RSA DES key from deserilized class
rsa_priv_bytes = ''.join([chr(y&0xff) for y in pobj.encoded._data])
rsa_priv_crypt = RSA.importKey(rsa_priv_bytes)

aes_key_data = rsa_priv_crypt.decrypt(aes_data)
## Split on '\x00' and remove the first bit as it's padding
## If this fails we could fall back to hard coded: aes_key = aes_key_data[:-16]
```

```
    aes_key = aes_key_data.split('\x00')[-1]

    # default java aes is ECB with null iv
    iv = b'\x00'*16
    aes_crypt= AES.new(aes_key, AES.MODE_ECB, iv)
    ptxt_config = aes_crypt.decrypt(config_data)
    print ptxt_config


if __name__ == '__main__':
    main()
```
