github.com/muha2xmad/Python/blob/bdc7a711d5a775f8ae47b591f20fdd2e1360b77b/warzonerat/warzonerat_config_extraction.ipynb

```
{"payload":{"allShortcutsEnabled":false,"fileTree":{"warzonerat":{"items":
[{"name":"warzonerat_avemariarat_yara.yar","path":"warzonerat/warzonerat_avemariarat_yara.yar","contentType":"file"},
{"name":"warzonerat_config_extraction.ipynb","path":"warzonerat/warzonerat_config_extraction.ipynb","contentType":"file"}],"totalCount":2},"":
{"items":[{"name":"Amadey","path":"Amadey","contentType":"directory"},{"name":"Stealc","path":"Stealc","contentType":"directory"},
{"name":"warzonerat","path":"warzonerat","contentType":"directory"},
{"name":"Readme.md","path":"Readme.md","contentType":"file"}],"totalCount":4}},"fileTreeProcessingTime":2.664769,"foldersToFetch":
[],"reducedMotionEnabled":null,"repo":
{"id":672559164,"defaultBranch":"main","name":"Python","ownerLogin":"muha2xmad","currentUserCanPush":false,"isFork":false,"isEmpty":false,"c
07-30T13:52:47.000Z","ownerAvatar":"https://avatars.githubusercontent.com/u/97201148?
v=4","public":true,"private":false,"isOrgOwned":false},"symbolsExpanded":false,"treeExpanded":true,"refInfo":
{"name":"bdc7a711d5a775f8ae47b591f20fdd2e1360b77b","listCacheKey":"v0:1690725170.0","canEdit":false,"refType":"tree","currentOid":"bdc7a7
{"rawLines":["{"," \"cells\": ["," {"," \"cell_type\": \"markdown\","," \"metadata\": {},"," \"source\": ["," \"ref:\\n\","," 
\"https://kienmanowar.wordpress.com/2023/03/25/quicknote-decrypting-the-c2-configuration-of-warzone-rat/\\n\""," ]"," },"," {"," \"cell_type\":
\"code\","," \"execution_count\": 2,"," \"metadata\": {},"," \"outputs\": [],"," \"source\": ["," \"# Refs:
https://stackoverflow.com/questions/9433541/movsx-in-python\\n\","," \"\\n\","," \"def SIGNEXT(x, b):\\n\","," \" m = (1 << (b -1))\\n\","," \" x = x &
((1 << b) -1)\\n\","," \" return ((x ^ m) - m)\\n\","," \" \\n\","," \"# This routine is responsible for decrypting the stored C2.\\n\","," \"def
rc4_customized_decryptor(data, key):\\n\","," \" idx = 0\\n\","," \" counter1 = 0\\n\","," \" counter2 = 0\\n\","," \" \\n\","," \" # Initialize RC4 S-
box\\n\","," \" rc4Sbox = list(range(256))\\n\","," \" \\n\","," \" # Modify RC4 S-box\\n\","," \" for i in range(256):\\n\","," \" counter2 += (rc4Sbox[i] +
key[i%250])\\n\","," \" counter2 = counter2 & 0x000000FF\\n\","," \" rc4Sbox[i] ^= rc4Sbox[counter2]\\n\","," \" rc4Sbox[counter2 & 0xFF] ^=
rc4Sbox[counter1 & 0xFF]\\n\","," \" rc4Sbox[counter1 & 0xFF] ^= rc4Sbox[counter2 & 0xFF]\\n\","," \" counter1 = i+1 \\n\","," \" \\n\","," \" #
Decrypt data\\n\","," \" counter1 = 0\\n\","," \" counter2 = 0\\n\","," \" j = 0\\n\","," \" decrypted = []\\n\","," \" while(idx < len(data)):\\n\","," \"
counter1 = j + 1\\n\","," \" k = (j+1)\\n\","," \" rc4Sbox_value1 = rc4Sbox[k]\\n\","," \" counter2 += (SIGNEXT(rc4Sbox_value1, 8) &
0xFFFFFFFF)\\n\","," \" rc4Sbox_value1_ = (SIGNEXT(rc4Sbox_value1, 8) & 0xFFFFFFFF)\\n\","," \" rc4Sbox_value2 = rc4Sbox[counter2 &
0x000000FF]\\n\","," \" rc4Sbox[k] = rc4Sbox_value2\\n\","," \" rc4Sbox[(counter2 & 0x000000FF)] = rc4Sbox_value1\\n\","," \" tmp1 =
rc4Sbox[((0x20 * counter1) ^ (counter2 >> 3)) & 0x000000FF]\\n\","," \" tmp2 = rc4Sbox[((0x20 * counter2) ^ (counter1 >> 3)) &
0x000000FF]\\n\","," \" tmp3 = rc4Sbox[((tmp1 + tmp2) & 0x000000FF) ^ 0xAA]\\n\","," \" tmp4 = rc4Sbox[(rc4Sbox_value2 + rc4Sbox_value1_)
& 0x000000FF]\\n\","," \" tmp5 = (tmp3 + tmp4) & 0x000000FF\\n\","," \" tmp6 = rc4Sbox[(counter2 + rc4Sbox_value2) & 0x000000FF]\\n\","," \"
decrypted.append(data[idx] ^ (tmp5 ^ tmp6))\\n\","," \" \\n\","," \" counter1 += 1\\n\","," \" j = counter1\\n\","," \" idx += 1\\n\","," \" \\n\","," \" return
bytes(decrypted)\\n\","," \"\\n\","," \"import binascii\\n\","," \"data = binascii.unhexlify(b'')\\n\","," \"key = binascii.unhexlify(b'')\\n\","," \"\\n\","," \"def
unicode_strings(buf, n=4):\\n\","," \" import re\\n\","," \" ASCII_BYTE = b' !\\\\\\\"#\\\\$%&\\\\'\\\\(\\\\)\\\\*\\\\+,-\\\\./0123456789:;<=>\\\\?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ\\\\[\\\\]\\\\^_`abcdefghijklmnopqrstuvwxyz\\\\{\\\\|\\\\}\\\\\\\\\\\\~\\\\t'\\n\","," \" if type(buf) == str:\\n\","," \"
buf = buf.encode('utf-8')\\n\","," \" reg = b'((?:[%s]\\\\x00){%d,})' % (ASCII_BYTE, n)\\n\","," \" uni_re = re.compile(reg)\\n\","," \" out = []\\n\","," \"
for match in uni_re.finditer(buf):\\n\","," \" try:\\n\","," \" out.append(match.group().decode(\\\"utf-16\\\"))\\n\","," \" except
UnicodeDecodeError:\\n\","," \" continue\\n\","," \" return out\\n\""," ]"," },"," {"," \"cell_type\": \"code\","," \"execution_count\": 3,"," \"metadata\":
{},"," \"outputs\": ["," {"," \"name\": \"stdout\","," \"output_type\": \"stream\","," \"text\": ["," \"C2 host: 89.117.76.41, port: 4422\\n\""," ]"," }"," ],","
\"source\": ["," \"import pefile\\n\","," \"import struct\\n\","," \"\\n\","," \"# Load the PE file using pefile\\n\","," \"pe =
pefile.PE(r\\\"D:\\\\samples\\\\danger\\\\f65a8af1100b56f2ebe014caeaa5bb2fbbca2da76cb99f3142354e31fbba5c8c\\\") # Put your file
path\\n\","," \"\\n\","," \"# Initialize variable to store .bss section data\\n\","," \"bss_section_data = None\\n\","," \"\\n\","," \"# Iterate through
sections to find the .bss section\\n\","," \"for section in pe.sections:\\n\","," \" section_name = section.Name\\n\","," \" if
section_name.startswith(b'.bss'):\\n\","," \" bss_section_data = section.get_data()\\n\","," \"\\n\","," \"# Extract the key size and key from the .bss
section\\n\","," \"key_size = struct.unpack('
```