Malware-String-Decryptor-Scripts/Badspace/badspace.py at main · X-Junior/Malware-String-Decryptor-Scripts · GitHub

github.com/X-Junior/Malware-String-Decryptor-Scripts/blob/main/Badspace/badspace.py

X-Junior

X-Junior/**Malware-String-Decryptor-Scripts**

1 Contributor | 0 Issues | 0 Stars | 0 Forks

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

```
import arc4 , pefile , sys , os

""

Author: Mohamed Ashraf (@X__Junior)

ref: https://x.com/Gi7w0rm/status/1791970049772687797

0b26abc692b7a2877b6b6fce6aa99b29af125b063f1c41b507362def59f8dfce

0d305291091bcb0c943c6472dce450272b2291b6287a053c5c553f082654c718

124e2b15b001eb302f0a5f43604621a001d250d42afdf353dc812f41bf249a55

1dd740062b30ce02e90238d55cb6f786496e120a40e93334fef7033e75d46d79

1ea681b79f88c2f0e9344beedb8776643d735c3f8251479c9495537c40fe5ba1

283cd2138b4f1ffef36411adee02f5d684593bdf3117c760ade04e19c958028a

2a4451ef47b1f4b971539fb6916f7954f80a6735cf75333fa9d19b169c31de2e

2a5a12cc4ef2f0f527cc072243aa27d3e95e48402ef674e92c6709dc03a0836a

2cbd9f49b2dec8a36e0961b5471bdb3266a5c061ba8784e14a193e700d156a0c

2f434cc508baac8440e95e955306ee354e76680eedca4a3ec2d87f592cfdcba7

33f81ee6d9747afe1c7c5a6ed741822749ea42bb297eb642f720fd44ae35e786

34f2fc85932f6fede57846cf2a2d55172d28e4a251bb4434a88a02ce8ec030f0

3f073189506b7ca07fb352e267699688bd3a6c11cde72217ec1ffbae211b6e15

40cdac6696e84f677d7e4817fd85f32da0f9256866bb85a25da207e3d5ca7d5c

425da6a7bd4faedc97990c6458d5e6a0635839037a99611385b77b43b443d1ec

475edfbb2b03182ef7c42c1bc2cc4179b3060d882827029a6e67c045a0c1149b

48640e2fb35f073c22937784f32c157d9a0781d61a2293f73fc3566b708205bd

4b4e27824cd349192cf0913060f1481a192f2b13d44e2787edbe8d7f0c57fa06

4e731e9e0233d53c70830011690f59b0764f61aa19e49cd10bed92b6eb81762c

53db2f135883d74dcac2e620d14d7f775876bf49d3d5d4fdb131f8fed4917434

5970ba228d2afe2031b8e8c17ba284746ebb9066f0ccb8e1fe33a6e3927a6c97
```

5b360b6855e87f173b4429adcca1d5f7735112119d69a5e9268673ab5ac82394

5cd47f178fd5afc2c290c77695277183df54d886f444f5993bbbe169eb3e2b12

60cd63e288c4054f85c9ea8167e0e58c1bd9998a15e3f8ed211132b42f76bdb6

613e6a8a49a61f157a8e064b7fbc7bd5d59909d47e31f6c18cd5c5659808ee89

616b1e1127902cef942cbc8ba6b89fe2e3090e992c7ae5e08c7d54b508b0caab

63537e464742099cfaf06904676e8955c0543a621e1936297e49090587a84ac1

668e1270bdb9a3aba41389777fc1ccd8759ad1316c62ea7c3f711925b44ef0b6

676cbcaa74ee8e43abaf0a2767c7559a8f4a7c6720ecc5ae53101a16a3219b9a

6a195e6111c9a4b8c874d51937b53cd5b4b78efc32f7bb255012d05087586d8f

6ac099ab5132a17bf7a492b47442f0f6776eb76d702a5c2d947dab0ab33cfc45

6fb83280ffc0feddf3f346a4d3a8914f26c097b8aef3a276590ea44ce9d70204

770cafb3fe795c2f13eb44f0a6073b8fe4fb3ee08240b3243c747444592d85ff

7c49024676be4f90d905028675d4a714311f971c099ab01e3cd26cd13c68499c

84519a45da0535087202b576391d1952a4cc81213f0e470db65f1817b65ee9d7

90b85d2ca44186de6df202abf27e3737c52691bf5dd28841fba8860bdc4483f8

927e941acb5bc42ff2050ad04fdb6e21d33f9b02cb3fc279dfee2f814557d8e5

9a27a2ad96f7676d28f99ffc4cbc51a81b42c7739fc15a0e57295b028d6c830d

9bc4c44b24f4ba71a1c7f5dd1c8135544218235ae58efa81898e55515938da6a

a1cb61abc99eb58e30ae7a9908c260be26ce072400ad771532bfe7c039ce10ef

a5f16fa960fe0461e2009bd748bc9057ef5cd31f05f48b12cfd7790fa741a24e

a725883bd1c39e48ab60b2c26b5692f7334a3e4544927057a9ffbdabfeedf432

ad2333e1403e3d8f5d9bd89d7178e85523fa7445e0a05b57fd9bc35547ec0d98

b6ac7f6e3b03acd364123a07b2122d943c4111ac4786bb188d94eae0e5b22c02

b9278ecce14213a1920ca9cc2b23ee18641c07a2780b693f009dcac578ffef92

ba4c8be6a1eb92d79df396eea8658b778f4bc0f010da48e1d26e3fc55d83e9c7

bb74c6fc0323956dd140988372c412f8b32735fb0ed1ad416e367d29c06af9cc

bfcb215f86fc4f8b4829f6ddd5acb118e80fb5bd977453fc7e8ef10a52fc83b7

c437e5caa4f644024014d40e62a5436c59046efc76c666ea3f83ab61df615314

cbd7ba0886a3e0d60b15bed0736bfaa130d47ab247e374d79c3612ce6ce049b6

ccde1ded028948f5cd3277d2d4af6b22fa33f53abde84ea2aa01f1872fad1d13

cec5bfbbd96c9a150d740c5be7d1d86c35ade0611085de537b8d1ca4887f2780

cee576f6d4d05bfb4f0e0704a4712af10b0afcb369407f5edf3526145a53a685

cf2e04d01b3de16d9aaa90c0d95775c9a99e63b23cc42043046ba31725d80e2e

cfa312272a7e55330855325925cc449a9ca8f80626d1003b0981c4375fad69a3

d7cfd49c873810b2f3369af4f8e8d0bac57c83137b1cd173f2f79a8d5f0898b9

ddbcce9bb969bda17064796c25abcc346748e7cd5d9d0460672d8d09ea97d24f

de6dbd27a07500e11af05f0420902c4d172aa34f6681d3f1546cf5b5872b3310

e4a9105c3c44cd3f0f975f807127aae121b67c561240fefdce215c715695d5be

e79e1858fdd8cb7642f0df4b2f696126df1bd6fc5f4731af8d797e02273f307f

e8ca376afa8e85fcd0487c25fd8330455cd2a5ea17aeaed95e9fd085d81035c8

e94f9221944a764f220831eb421d4571b32e5b243aad4943b69ae2bcfb176737

ebc0ded53cd49db7ea646bd02f391dee05f6093ec26300a7389ae2ef8d769a6f

eca43317ae815a18eeaf723506c960a9b2edc39f127e5a200011e594e0ab31e2

f57dcff87305797c6488b8a45b2d48c1c119cc19a316f452c04b38e30090477d

975deab236438b6d7fa3ad1be7d9c2a3fabbd6103ff5f8b7fe536205ad715508

cd9aebcc686a8a2eb25bf5d75100b28f58aad6512222ade6630bbad59e877369

e04562fb05388e10d6d70d4cadbec059c6c0601f8232d8699ad8a6d3ee0e75d6

9d4c80ea1d6d1ce11f9bb79d7a5a4ddfcea9f20ffe039db7215e9c57fc183476

5649dcd896bf2155e790c5f05b9fa2ba6fe5befcac85a8cb0beed23945686e02

f31e28b2fd8efe63a7a2c39f7f87d895c44694d80b5fcbff91d51dc63eafa9dc

```
d20903e4f8635fc8f8a7d1ab2330a61eb1fad29e03c353ede85bc359aa019f2c
a20c9fe2888286473faea909d2f22a75a1b982387b08e2ba0bd091ae631f36fc
712738c0afe1d10f28b6aefecb44f2bc442007fdd65f8f07582120e3ec22d590
62fb7f43c677ee2fe56406e7af8876289d3751e7c001aa627dd287baf5687f06
0d2cf14d27586ff9da5832e0efaba872a1641617fdb4a47d94b645172f7d2fa6
7b340050fe9bec7024092de63d223d2a96a32d14676f6c82c9024278ae0b323e
b54b42b4dfb93502646e9e8cb0eb5b65dccf2b872ab79f67641e307a08234b94
55ace018a6c4f355511ce3f6833d4b997d4323afb890520dc815aa2f916499f3
0d59c9bef911c879011f21163a083c09b759c9757f1ade9da9f87fdce27dc5f4
30a85fa1bf6df41d841efbf986beb286eb829380ebfdf0c1ac694f3d4f24315a
d4c955b1db1e499ea47196b8f630205329f9277f3cc184d75a3b69a70d8c49da
41d9d1e0599b492fdb6fa2ce47f0094112799830dd8dc1c098690a500a8fa6b1
6db0d6eaff5279d815e66e1abbdd7e4159c58c7747b158659d875c369c153b89
1bcfed8b593a8a7c8b34e074aca3d4fc68a0ea3343b32eae89fdabf35ad40e7d
193cadbea116833efaaa0bc6fbea552a68c9694fb0177ad873d702001b4cef8d
eec7ed30a026ba5ba82c288693bb6ad16cfc5643768bb89e5a0b17109d1fc7a6
a0916d3b97c0df2ec1ed6a772dac27c24842a64d4f6e078c941fa2046cabb9ed
'''

def rc4_decryption(key, encrypted_data):
cipher = arc4.ARC4(key)
decrypted_data = cipher.decrypt(encrypted_data).decode()
return decrypted_data
def rc4key_offset(data):
counter = 0
flag_zero_byte = 0
```

```
flag_byte = 0

for i in data:

counter += 1

if i != 0 and flag_zero_byte:

flag_byte = 1

if i == 0:

flag_zero_byte = 1

if flag_zero_byte and flag_byte:

return counter - 1

def main():

# Check if the correct number of arguments are provided

if len(sys.argv) != 2:

# python3 Badspace_String_Decryptor.py Badspace.exe

print("Usage: python Badspace_String_Decryptor.py [filename]")

exit()

# Check if the file exists

if not os.path.isfile(sys.argv[1]):

print(f"The file {sys.argv[1]} does not exist.")

exit()

targeted_data = []

pe_file = pefile.PE(sys.argv[1])

for section in pe_file.sections:

if b".rdata" in section.Name:

targeted_data.append(section.get_data())

if b".data" in section.Name:
```

```
targeted_data.append(section.get_data())

for data_section in targeted_data:

for i in range(0,len(data_section)-1):

length = 0

data = b''

if i == 0:

if data_section[i] != 0 and data_section[i+1] == 0:

length = data_section[i]

data = data_section[i:i+8+length]

elif ( data_section[i-1] == 0 or data_section[i-1] == 255) and data_section[i] != 0 and
data_section[i+1] == 0:

length = data_section[i]

data = data_section[i:i+8+length]

try:

if length != 0 and data != b'':

offset = rc4key_offset(data)

rc4_key = data[offset:offset+4]

encrypted_data = data[offset+4:offset+4+length]

print(rc4_decryption(rc4_key,encrypted_data))

except Exception as e:

continue

if __name__ == '__main__':

main()
```