usualsuspect/daxin_decrypt_embedded.py

gist.github.com/usualsuspect/839fbc54e0d76bb2626329cd94274cd6

```
#!/usr/bin/env python3

#
# Algorithm used by Daxin to decrypt embedded driver
# Uses slightly modified RC4 (see comment in rc4() below)
#
# Constants fitting for sample
# b0eb4d999e4e0e7c2e33ff081e847c87b49940eb24a9e0794c6aa9516832c427
#

import hashlib
import struct

def gen_key(const1,const2):
# hardcoded into function, might also change per sample
```

```
key_data = b"\x7C\x4E\xD0\x68\x20\x4b\x42\xEB\x08\x4A\xFE\xA9\xEB\x50\x30\xa3"
d1 = struct.pack("<I",const1)
d2 = struct.pack("<I",const2)

key_data = d1 + key_data + d2

key = hashlib.md5(key_data).digest()
out = bytearray(struct.pack("<I",const1^const2) + key[4:])
h = const1 ^ const2
for i in range(16):
if (i & 1):
k = ((h << 11) & 0xFFFFFFFF) ^ ((h >> 5) & 0xFFFFFFFF)
k ^= out[i]
k ^= 0xFFFFFFFF # not
else:
k = ((h >> 3) & 0xFFFFFFFF) ^ ((h << 7) & 0xFFFFFFFF)
k ^= out[i]

h ^= k

out = struct.pack("<I",h) + out[4:]
return out

def rc4(data, key):
x = 0
box = bytearray(range(256))
for i in range(256):
x = (x + box[i] + key[i % len(key)]) % 256
box[i], box[x] = box[x], box[i]

```

```
y = x # original RC4 sets both 0
x = 0
out = bytearray()
for char in data:
x = (x + 1) % 256
y = (y + box[x]) % 256
box[x], box[y] = box[y], box[x]
out.append(char ^ box[(box[x] + box[y]) % 256])
return out


data = open("driver","rb").read()
key = gen_key(0x4373F262,0x21B33EE9)
plain = rc4(data,key)
open("out","wb").write(plain)
```