psrok1/icedid-reconstruct.py

gist.github.com/psrok1/e6bf5851d674edda03a201e7f24a5e6b

```
"""
Py3 version of IcedID custom steganographic loader (PNG parser & PE reconstructor)
Inspired by
https://github.com/hasherezade/funky_malware_formats/tree/master/iced_id_parser
Authored by @nazywam and @psrok1.

pip3 install malduck lief
"""

import sys
import malduck
from lief import PE


class IcedidSection(malduck.Structure):
_pack_ = 1
```

```
_fields_ = [
("VirtualOffset", malduck.DWORD),
("VirtualSize", malduck.DWORD),
("RawOffset", malduck.DWORD),
("RawSize", malduck.DWORD),
("Characteristics", malduck.BYTE)
]


class IcedidHeader(malduck.Structure):
_pack_ = 1
_fields_ = [
("ImageBase", malduck.QWORD),
("ImageSize", malduck.DWORD),
("EntryPoint_va", malduck.DWORD),
("ImportDir_va", malduck.DWORD),
("RelocDir_va", malduck.DWORD),
("RelocDir_size", malduck.DWORD),
("SectionCount", malduck.DWORD)
]


def decrypt_image(path: str) -> bytes:
p = malduck.procmem.from_file(path)
idat_off = next(p.findp(b'IDAT'))
idat_len = p.uint32p(idat_off - 4)
data = p.readp(idat_off + 4, idat_len)
decrypted = malduck.rc4(data[:8], data[8:])
return decrypted
```

```
def reconstruct(payload: bytes) -> bytes:
p = malduck.procmem(payload)

data_offset = offset = 0x1188

header_data = p.readp(offset, IcedidHeader.sizeof())
header = IcedidHeader.parse(header_data)

offset += IcedidHeader.sizeof()
sections_data = p.readp(offset, IcedidSection.sizeof() * header.SectionCount)

sections = [IcedidSection.parse(data) for data in malduck.chunks(sections_data,
IcedidSection.sizeof())]

pe = PE.Binary('icedid_binary', PE.PE_TYPE.PE32)
pe.optional_header.imagebase = header.ImageBase
pe.optional_header.addressof_entrypoint = header.EntryPoint_va

sections = sorted(sections, key=lambda s: s.VirtualOffset)

for idx, sec in enumerate(sections):
section = PE.Section('')
content = p.readp(data_offset + sec.RawOffset, sec.RawSize)
section.content = list(content)
section.virtual_address = sec.VirtualOffset
section.characteristics = 0xE0000000
if idx != len(sections) - 1:
section.virtual_size = sections[idx + 1].VirtualOffset - sec.VirtualOffset
pe.add_section(section)
pe.data_directory(PE.DATA_DIRECTORY.IMPORT_TABLE).rva = header.ImportDir_va
pe.data_directory(PE.DATA_DIRECTORY.BASE_RELOCATION_TABLE).rva =
header.RelocDir_va
pe.data_directory(PE.DATA_DIRECTORY.BASE_RELOCATION_TABLE).size =
header.RelocDir_size

builder = PE.Builder(pe)
builder.build()
return bytes(builder.get_build())


if __name__ == '__main__':
if len(sys.argv) < 3:
print("Usage: ./reconstruct.py [pngfile] [outfile]")
else:
inpath = sys.argv[1]
payload = decrypt_image(inpath)
data = reconstruct(payload)
with open(sys.argv[2], "wb") as f:
f.write(data)
print("[*] Stored {} bytes in {}".format(len(data), sys.argv[2]))
```