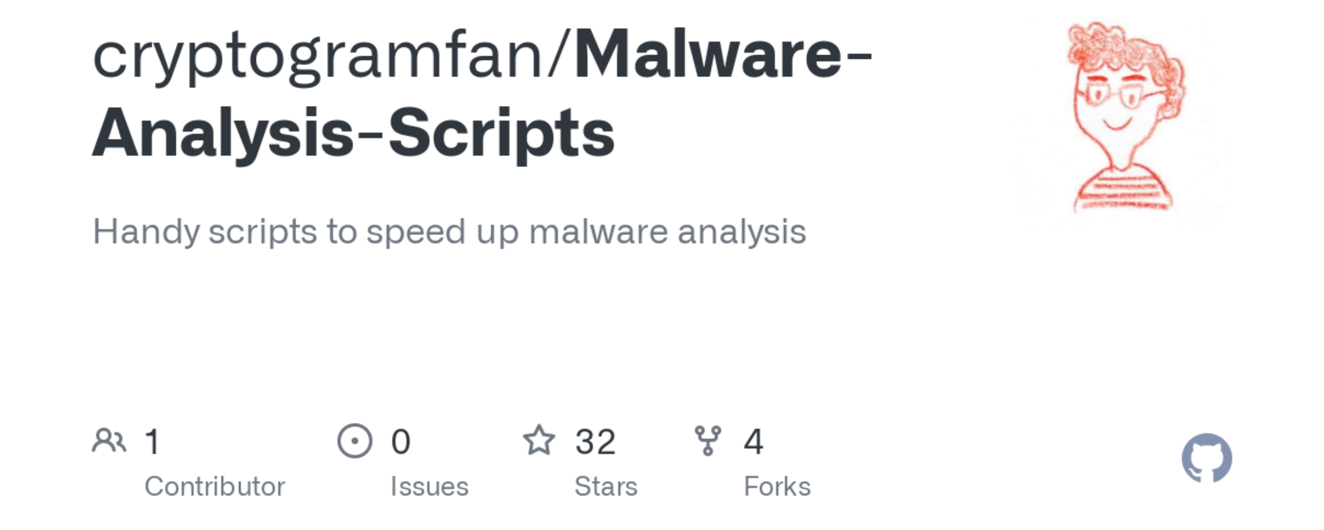
Malware-Analysis-Scripts/decrypt_l0rdix_c2.py at master · cryptogramfan/Malware-Analysis-Scripts · GitHub

github.com/cryptogramfan/Malware-Analysis-Scripts/blob/master/decrypt_l0rdix_c2.py

cryptogramfan



cryptogramfan/**Malware-Analysis-Scripts**

Handy scripts to speed up malware analysis

```
#!/usr/bin/env python
#
# A script that identifies, decrypts and extracts L0rdix RAT command and control (C2)
# traffic from a supplied PCAP file.
#
# To speed up parsing, trim your PCAP to only HTTP ports using tcpdump,
# for example:
#
# $ tcpdump -r l0rdix_c2.pcap -w l0rdix_c2_http.pcap 'tcp port 80 or 8080 or 3128'
#
# Requirements:
# pyshark-legacy
# pycryptodome
#
# Author.....: Alex Holland (@cryptogramfan)
# Date.......: 2019-07-27
# Version....: 0.1.6
# License....: CC BY 4.0
# Reference_1: https://www.bromium.com/an-analysis-of-l0rdix-rat-panel-and-builder/
# Reference_2: https://www.bromium.com/decrypting-l0rdix-rats-c2/

import sys
import argparse
import pyshark
import urllib
import re
import hashlib
import binascii
import uuid
from Crypto.Cipher import AES
from base64 import b64decode

parser = argparse.ArgumentParser(description="\nUsage: python decrypt_l0rdix_c2.py -p <l0rdix_c2.pcap> -k <OPERATOR_KEY>")
parser.add_argument("-p", dest="pcap_file", help="PCAP containing encrypted L0rdix C2 traffic.", required=True)
parser.add_argument("-k", dest="operator_key", help="UTF-8 operator key extracted from a L0rdix bot or panel. If no key is supplied, the default key \"3sc3RLrpd17\" will be used.", default="3sc3RLrpd17")
parsed_args = parser.parse_args()
operator_key = parsed_args.operator_key
```

```
aes_key = hashlib.sha256(operator_key).digest()
iv = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
parameters = []
hostnames = []
imgs = []

try:
pcap = pyshark.FileCapture(parsed_args.pcap_file, keep_packets=False,
display_filter='http.request.method == POST && http.request.uri.path == "/connect.php"
&& count(http.request.uri.query.parameter) >= 10')
print "[+] Parsing PCAP..."

except:
print(parser.description)
exit(0)

try:
print "[+] Searching for L0rdix C2 traffic..."
for packet in pcap:
# Enumerate hosts
query = packet['HTTP']
hostnames.append(query.host)

# Enumerate parameters
query = query.request_uri_query
query = urllib.unquote(query)
query = re.sub("~", "+", query)
query = re.sub("^h=", "", query)
found_parameters = re.split("&[a-z]{1,2}=", query)
```

```
parameters.extend(found_parameters)

except:
print "[!] Error, exiting."
exit(0)

if not hostnames:
print "[+] No L0rdix C2 traffic found."
exit(0)

else:
print "[+] Found references to L0rdix C2 servers (%d):\n" % (len(hostnames))
for hostname in hostnames:
print hostname

if not parameters:
print "[+] No L0rdix URI parameters found."
exit(0)

else:
print "\n[+] Found L0rdix C2 traffic (%d strings):\n" % (len(parameters))
for parameter in parameters:
print parameter

try:
print "[+] Searching for screenshots..."
for packet in pcap:
# Enumerate screenshots
img = packet['URLENCODED-FORM']
img = urllib.unquote(img.value)
img = b64decode(img)
img = bytearray(img)
img_name = str(uuid.uuid4()) + '.jpg'
imgs.append(img_name)

# Dump screenshots
f = open(img_name, 'w+b')
f.write(img)
f.close()

except:
print "[!] Error, exiting."
exit(0)

if not imgs:
print "[+] No L0rdix screenshots found."
exit(0)

else:
print "[+] Dumped L0rdix screenshots in current directory (%d):\n" % (len(imgs))
for img_name in imgs:
print img_name

print "\n[+] Decrypting strings using operator key (UTF-8): " + operator_key
print "[+] AES key (hex): " + binascii.hexlify(bytearray(aes_key))
print "[+] IV (hex): " + binascii.hexlify(bytearray(iv))
print "[+] Decrypted L0rdix C2 traffic (%d strings):\n" % (len(parameters))

for parameter in parameters:
cipher = AES.new(aes_key, AES.MODE_CBC, iv)
```

```
ciphertext = b64decode(parameter)
decrypted = cipher.decrypt(ciphertext)
decrypted = decrypted.rstrip()
print decrypted

print "[+] Finished, exiting."
```