```
import os
import sys
import csv
import hashlib
import pefile
import struct


def extract_loki_file(filepath, csv_writer=None):
        try:
                print'[+] Working on '+ filepath
                pe = pefile.PE(filepath)
                offset = 0
                size = 0
                for rsrc1 in pe.DIRECTORY_ENTRY_RESOURCE.entries:
                        for rsrc2 in rsrc1.directory.entries:
                                if rsrc2.id == 1000 and
rsrc2.directory.entries[0].id == 1033:
                                        key_entry = rsrc2.directory.entries[0]
                                        offset =
key_entry.data.struct.OffsetToData
                                        size = key_entry.data.struct.Size

                if size > 0x256:
                        print '[+] Size seems to be very large '+hex(size)
                        return

                out = ''
                count = 0
                actual_key = ''
                skip_bytes = 0
                chunk_size = 0
                start_resource_id = 0
                size_of_core_file = 0
                if size > 0x58:
                        key_data =
pe.get_memory_mapped_image()[offset:offset+size]
                        key = key_data[:0x10]
                        data = key_data[0x10:]


                        for i in range(0, len(data)):
                                i_key = (i + 1) % len(key)
                                out += chr(ord(key[i_key]) ^ ord(data[i]))
                        actual_key = out[0x38:0x48]
                        skip_bytes = ord(out[0x28])
                        chunk_size = ord(out[0x0E])
                        start_resource_id =
struct.unpack('<H',str(out[0x1E:0x1E+2]))[0]
                        size_of_core_file =
struct.unpack('<L',str(out[0x48:0x48+4]))[0]
                        count = struct.unpack('<L',str(out[0x8:0x8+4]))[0]
```

```
                        print "[+] Chunk Size : " + str(chunk_size)
                        print "[+] Skip Bytes : " + str(skip_bytes)
                        print "[+] Resource count is : " + str(count)
                        print "[+] Start of Resource id : " +
str(start_resource_id)
                        print "[+] Size of core file : "+
hex(size_of_core_file)
                        # search for encrypted data in Bitmap resource
                        sizes = []
                        offsets = []
                        for rsrc1 in pe.DIRECTORY_ENTRY_RESOURCE.entries:
                                for rsrc2 in rsrc1.directory.entries:
                                        if rsrc2.id >= start_resource_id and
rsrc2.id < start_resource_id + count:
                                                for dire in
rsrc2.directory.entries:
                                                        if dire.id is not None:

        sizes.append(dire.data.struct.Size)

        offsets.append(dire.data.struct.OffsetToData)

                        data=''
                        for i in range(0, len(offsets)):
                                data +=
pe.get_memory_mapped_image()[offsets[i]:offsets[i]+sizes[i]]

                        #print '[+] Size of encrypted data : '+hex(len(data))

                        actual_data = ''
                        index = 0
                        read_index = 0
                        while read_index < len(data):
                                #print '[+] Collecting data ',
                                actual_data += data[read_index]
                                #print len(actual_data)

                                if index % chunk_size == 0:
                                        read_index = read_index + skip_bytes

                                index = index + 1
                                read_index = read_index + 1

                        #print '[+] Size of encrypted data :
'+hex(len(actual_data))
                        out = ''
                        print '[+] Decrypting actual file ' +
str(len(actual_data))
                        for i in range(0, len(actual_data)):
                                i_key = (i + 1) % len(actual_key)
                                out += chr(ord(actual_key[i_key]) ^
ord(actual_data[i]))
                                if i == size_of_core_file-1:
```

```
                                        break
                        out_filepath = os.path.basename(filepath) + "_core"

                        try:
                                #check for output buffer is PE file
                                #print out
                                pe = pefile.PE(data=out)

                                #if PE file found then write to file
                                print('Writing output to destination ' +
out_filepath)
                                with open(out_filepath,"wb") as file:
                                        file.write(out)

                        #write csv
                                if csv_writer is not None:
                                        input_file = open(filepath,
'rb').read()
                                        parent_file_md5 =
hashlib.md5(input_file).hexdigest()
                                        extracted_file_md5 =
hashlib.md5(out).hexdigest()

        csv_writer.writerow({'parent_file_md5':parent_file_md5,"extracted_file
_md5":extracted_file_md5})

                        except pefile.PEFormatError as e:
                                print 'Exception occured'
                                print str(e)
                                return

        except pefile.PEFormatError as e:
                print "Given file is not PE file"
        except Exception as e:
                print str(e)
                pass

def main(argv):
        if len(argv) < 2:
                print"Please provide argument"
        else:
                if os.path.isfile(argv[1]):
                        extract_loki_file(argv[1])
                elif os.path.isdir(argv[1]):
                        with open('results.csv', 'wb') as csvfile:
                                fieldnames = ['parent_file_md5',
'extracted_file_md5']
                                csv_writer = csv.DictWriter(csvfile,
fieldnames=fieldnames)
                                csv_writer.writeheader()
                                for root, dirs, files in os.walk(argv[1]):
                                        for file in files:
```

```
        extract_loki_file(os.path.join(root, file),csv_writer)

if __name__ == '__main__':
        main(sys.argv)
```