LAMPIRAN

LAMPIRAN A.1

Tabel Lampiran A.1 Data hasil testing *application behavior* MsPowerPoint

NO.	Daftar *Behavior*	Data *Behavior* Per Testing			Jumlah Rata-Rata
		Test 1	*Test 2*	*Test 3*	
1.	*Total Actions*	14	13	37	21.33
2.	Open Actions	0	11	22	11.00
3.	*Created*	4	1	3	2.67
4.	*Changed*	9	1	11	7.00
5.	*Deleted*	1	2	1	1.33
6.	*Renamed*	0	0	0	0.00
7.	*Total Files*	3	3	3	3.00
8.	*Files Exist*	3	3	3	3.00
9.	*Files Missing*	0	0	0	0.00
10.	*Signature Known*	1	1	2	1.33
11.	*Signature Unknown*	2	2	1	1.67

LAMPIRAN A.2

Tabel Lampiran A.2 Data hasil testing *application behaviour* Adobe Acrobat Reader

NO.	Daftar *Behavior*	Data *Behavior* Per Testing			Jumlah Rata-Rata
		Test 1	*Test 2*	*Test 3*	
1.	*Total Actions*	2	1	2	1.67
2.	*Open*	0	0	0	0.00
3.	*Created*	1	1	1	1.00
4.	*Changed*	1	0	1	0.67
5.	*Deleted*	0	0	0	0.00
6.	*Renamed*	0	0	0	0.00
7.	*Total Files*	1	1	1	1.00
8.	*Files Exist*	1	1	1	1.00
9.	*Files Missing*	0	0	0	0.00
10.	*Signature Known*	0	0	0	0.00
11.	*Signature Unknown*	1	1	1	1.00

LAMPIRAN A.3

Tabel Lampiran A.3 Data hasil testing *application behaviour* Xn View Image

NO.	Daftar *Behavior*	Data *Behavior* Per Testing			Jumlah Rata-Rata
		Test 1	*Test 2*	*Test 3*	
1.	*Total Actions*	2	1	1	1.33
2.	*Open*	0	0	0	0.00
3.	*Created*	1	1	1	1.00
4.	*Changed*	1	0	0	0.33
5.	*Deleted*	0	0	0	0.00
6.	*Renamed*	0	0	0	0.00
7.	*Total Files*	1	1	1	1.00
8.	*Files Exist*	1	1	1	1.00
9.	*Files Missing*	0	0	0	0.00
10.	*Signature Known*	0	0	0	0.00
11.	*Signature Unknown*	1	1	1	1.00

LAMPIRAN A.4

Tabel Lampiran A.4 Data hasil testing malware Locky Ransomware behaviour

NO.	Daftar *Behavior*	Data *Behavior* Per Testing			Jumlah Rata-Rata
		Test 1	*Test 2*	*Test 3*	
1.	*Total Actions*	784	1769	643	1065.33
2.	*Open*	0	0	0	0.00
3.	*Created*	4	7	0	3.67
4.	*Changed*	437	818	278	511.00
5.	*Deleted*	0	0	0	0.00
6.	*Renamed*	235	573	180	329.33
7.	*Total Files*	185	755	153	364.33
8.	*Files Exist*	135	538	110	261.00
9.	*Files Missing*	50	217	43	103.33
10.	*Signature Known*	0	0	0	0.00
11.	*Signature Unknown*	185	755	153	364.33

LAMPIRAN B.1

FILE NAME : malde.py

```
#-----------------------------      ---
# DiEdit oleh : M. Ikhwan N.Elyas   ---
# Versi Stabil 06.02                ---
# Nama branch : malde01             ---
#-----------------------------      ---

# Copyright (c) 2009-2014, Mario Vilas
# All rights reserved.

# $Id: 09_api_hook.py 1299 2013-12-20 09:30:55Z qvasimodo $

from winappdbg import Debug, EventHandler, System
from winappdbg.win32 import *
from mymodule import *
from myvar import *
import os
import psutil
import threading
import time
#import datetime

# data3 list untuk menampung sementara data dir monitor
# data4 list untuk menampung sementara data process debug
# data5 list untuk menampung hasil komparasi data3 dan data4

data3 = []
data4 = []
data5 = []
pidtmp = set()
mylist = []
sdat = False
pid, exe, stat, string, gtag = '','','','',''
```

```
class MyEventHandler( EventHandler ):

    apiHooks = {

        # Hooks for the kernel32 library.
        'kernel32.dll' : [

            #  Function            Parameters
            ( 'CreateFileA'     , (PVOID, DWORD, DWORD, PVOID, DWORD, DWORD, HANDLE) ),
            ( 'CreateFileW'     , (PVOID, DWORD, DWORD, PVOID, DWORD, DWORD, HANDLE) ),
            ( 'CreateProcessA'  , 10 ),
            ( 'CreateProcessW'  , 10 ),

        ],

        # Hooks for the advapi32 library.
        'advapi32.dll' : [

            #  Function            Parameters
            ( 'RegCreateKeyExA' , (HKEY, PVOID, DWORD, PVOID, DWORD, REGSAM, PVOID, PVOID, PVOID) ),
            ( 'RegCreateKeyExW' , (HKEY, PVOID, DWORD, PVOID, DWORD, REGSAM, PVOID, PVOID, PVOID) ),

        ],
    }

# =======Tambahan modifikasi================
#   Fungsi --pre_CreateProcessA-- ini tidak ada indikasi apapun, jadi baiknya dinonaktifkan
    def pre_CreateProcessA( self, event, ra, lpApplicationName, lpCommandLine, lpProcessAttributes,
                lpThreadAttributes, bInheritHandles, dwCreationFlags, lpEnvironment,
                lpCurrentDirectory, lpStartupInfo, lpProcessInformation):
        self.__print_createps_ansi( event, "CREATE_PROCESS", lpApplicationName )
        # untuk win7 fungsi ini tidak bekeraja

    def pre_CreateProcessW( self, event, ra, lpApplicationName, lpCommandLine, lpProcessAttributes,
                lpThreadAttributes, bInheritHandles, dwCreationFlags, lpEnvironment, lpCurrentDirectory,
                lpStartupInfo, lpProcessInformation):
        self.__print_createps_unicode( event, "CREATE_PROCESS", lpApplicationName )
        # untuk win7 fungsi ini yang bekeraja
```

```
def post_CreateProcessA( self, event, retval ):
    self.__print_createps_success( event, retval )
def post_CreateProcessW( self, event, retval ):
    self.__print_createps_success( event, retval )


def __print_createps_ansi( self, event, tag, pointer ):
    # untuk win7 fungsi ini tidak bekeraja
    # set data to global variable supaya bisa diakses dari luar
    global pid, exe, string, gtag
    string = event.get_process().peek_string( pointer )
    pid    = event.get_pid()
    exe    = os.path.basename(event.get_process().get_filename())
    gtag   = tag


def __print_createps_unicode( self, event, tag, pointer ):
    # untuk win7 fungsi ini yang bekeraja
    # set data to global variable supaya bisa diakses dari luar
    global pid, exe, string, gtag
    string = event.get_process().peek_string( pointer, fUnicode = True )
    pid    = event.get_pid()
    exe    = os.path.basename(event.get_process().get_filename())
    gtag   = tag



def __print_createps_success( self, event, retval ):
    # set data to global variable supaya bisa diakses dari luar
    global pid, exe, stat, string, gtag, data3, mylist
    try:
        pspid  = getpidexe(os.path.basename(string))
        if retval:
            data3.append([str(pid), exe, gtag, string, str(pspid)])
            start_debug() #OK, masih ada error
            getfilesopenbyps(int(pspid),mylist)
    except:
```

```
            pass

# =======End Tambahan==================

    def pre_CreateFileA( self, event, ra, lpFileName, dwDesiredAccess,
             dwShareMode, lpSecurityAttributes, dwCreationDisposition,
                                dwFlagsAndAttributes, hTemplateFile ):
        self.__print_opening_ansi( event, "OPEN_FILE", lpFileName )


    def pre_CreateFileW( self, event, ra, lpFileName, dwDesiredAccess,
             dwShareMode, lpSecurityAttributes, dwCreationDisposition,
                                dwFlagsAndAttributes, hTemplateFile ):
        self.__print_opening_unicode( event, "OPEN_FILE", lpFileName )


    def post_CreateFileA( self, event, retval ):
        self.__print_success( event, retval )

    def post_CreateFileW( self, event, retval ):
        self.__print_success( event, retval )


    def __print_opening_ansi( self, event, tag, pointer ):
        global sdat, pid, exe, string, gtag
        try:
                    if os.path.isfile(event.get_process().peek_string(pointer)) == True:
                        for dfilter in data_filter:
                            if dfilter in event.get_process().peek_string(pointer, fUnicode = True):
                                string = event.get_process().peek_string(pointer)
                                sdat, gtag  = True, tag
                                pid    = event.get_pid()
                                exe    = os.path.basename(event.get_process().get_filename())
                    else:
                        sdat = False
        except:
            pass
```

```
    def __print_opening_unicode( self, event, tag, pointer ):
        global sdat, pid, exe, string, gtag
        try:
                    if os.path.isfile(event.get_process().peek_string(pointer, fUnicode = True)) == True:
                        for dfilter in data_filter:
                            if dfilter in event.get_process().peek_string(pointer, fUnicode = True):
                                string = event.get_process().peek_string(pointer, fUnicode = True)
                                sdat, gtag  = True, tag
                                pid    = event.get_pid()
                                exe    = os.path.basename(event.get_process().get_filename())
                    else:
                        sdat = False
        except:
            pass

    def __print_success( self, event, retval ):
        global sdat, pid, exe, string, gtag, data3
        try:
                        # tambahan ini karena selalu muncul error
                        #"--set object has no attribute 'append'--"
                    if retval :
                        if sdat:
                            data3.append([str(pid), exe, gtag, string])
                    else:
                        data3.append([str(pid), exe, gtag, string])
        except :
            pass

try:
        d1 = Debug(MyEventHandler(), bKillOnExit = False)
except Exception as E:
        print " Error Debug : " + str(E)

def start_debug(): # Hasil OK.
        try:
                d1.stop()
                for pid in getuserps_id():
                        try:
```

```
                                thexe = getexe(int(pid)).lower()
                                if thexe == "cmd.exe" or thexe == "py.exe" or thexe == "python.exe":
                                        pass
                                else:
                                        d1.attach(int(pid))
                        except: # Exception as E:
                                pass
                d1.loop()
        except: # Exception as E:
                pass
        finally:
                d1.stop()


if __name__ == "__main__":
        rpt  = 0 # untuk menghitung jumlah report
        driveC    = "C:\\"
        missfile  = 0
        act = {
          1 : "CREATED",
          2 : "DELETED",
          3 : "CHANGED",
          4 : "REN_Frm",
          5 : "REN__To"
        }


        print ' Start Monitoring... '

        def waktu_monitoring():
                try:
                        t1 = datetime.now()
                        while 1:
                                time.sleep(0.02)
                                if (datetime.now()-t1).seconds > 5:
                                        # periksa dan show data setiap 5 detik
                                        getappid()
                                        marge_comparator_data()
```

```
                                t1 = datetime.now()

                except:
                        pass

        def getappid():
                # fungsi untuk mengambil semua pid user_proses yang mengakses file
                # pid disimpan dalam var=pditmp :
                # type=set(), type set secara otomatis mengabaikan dobel item (uniq)
                global mylist, data3, pidtmp
                try:
                        time.sleep(0.4)
                        if len(data3) >0:
                                for d3 in data3:
                                        pidtmp.add(str(getpidexe(d3[1])))

                        if len(pidtmp) > 0:
                                for ppid in pidtmp:
                                        getfilesopenbyps(int(ppid),mylist)
#                                       print ' PID-TMP : ', ppid
#               except Exception as E:
#                       print ' Error getappid : ', str(E)
                except:
                        pass

        def marge_comparator_data():
                # fungsi untuk -menggabungkan data, -filter data, -modifikasi data
                # marge_comparator_data
                global data3, data4, mylist, pidtmp,rpt
                allf = set()
                fexi,fsig = 0,0
                try:

                #-----------------------------------------------
                        data5 = data3 + data4
                        if len(data5)>0:
                                dx = len(data5)
                                data5 = sort_deduplicate(data5)
```

```
        #---------------------------------------------
                        rpt = rpt+1
                        print '\n' + '='*55 + '\n'
                        for d5 in data5:
                                if len(d5) > 4:f = d5[-2]
                                else:f = os.path.join(driveC,d5[-1])
                                allf.add(f)
        #---------------------------------------------
                        i = 0
                        for sl in allf:
                                cfile = isfile(sl)
                                csig  = getfsig(sl)
                                if cfile == 'EXIST':fexi = fexi + 1
                                if csig  == 'unknown':fsig = fsig + 1
                                i = i+1
                        ufsig = len(allf) - fsig
        #---------------------------------------------
        #               TEST PRINT DATA
        #               Hasil OK, Sementera Dinonaktifkan
        #               print_event(data5)
        #---------------------------------------------
                        print_report(rpt,data3,data4,allf,fexi,ufsig)
                        data3 =[] # From Event
                        data4 =[] # From Wdir
                        data5 =[] # From Equal
                        mylist=[]
                        allf.clear()
                        fexi,fsig = 0,0
                        pidtmp.clear()
        except:
                pass


def wdir():
        global data4
        while True:
                try:
                        for ac, fl in watch_dir(14,driveC):
```

```
                                        for dfilter in data_filter:
                                                if dfilter in fl:
                                                        data4.append([act.get(ac,'---'),fl])
                                except:
                                        pass

        try:
                th1 = threading.Thread(target=start_debug)
                th1.start()

                th2 = threading.Thread(target=wdir)
                th2.start()

                th4 = threading.Thread(target=waktu_monitoring)
                th4.start()
        except:
                pass

#===KET : ========================
#       data3  = list1 : list get from debug event
#       data4  = list2 : list get wdir event
#       data5  = list3 : list get from compare between data4 to data3 live (var to var)
#       mylist = mylist: list untuk menampung data hasil psutil.processs.openfile
#       mylist = mylist: isinya <<appname : cmd.line/open.file>>
```

LAMPIRAN B.2

FILE NAME : module.py

```
#-------------------------------------------

import os
import psutil
import win32gui
import subprocess
import win32file
import win32con
import win32process
import win32api
import msvcrt
import time
#import datetime
import socket
import glob
#import winappdbg
from winappdbg import System, system, win32
from datetime import datetime
import binascii
import signal
from myvar import *
import subprocess as sp
from collections import Counter

def wtofile(fl,s):
        try:
                fl = 'LOG/'+fl
                f = open(fl,'a+')
                f.write(s+"\n")
        except:
                f = open(fl,'w')
        f.close
```

```
def wfile(fl,s):
        try:
                f = open(fl,'a+')
                f.write(s+"\n")
        except:
                f = open(fl,'w')
        f.close

def ftolist(fn):
        with open(fn,'r') as F:
                F.readlines
                for l in F:
                        l=list(l.split(' '))
#                       l = sorted(l)
#                       print type(l)
                        print '=>',l

def watch_dir(kode,path_to_watch):
#       global fileEx
        ACTIONS = {
          1 : "CREATED",
          2 : "DELETED",
          3 : "CHANGED",
          4 : "REN_Frm",
          5 : "REN__To"
        }
        FILE_LIST_DIRECTORY = 0x0001

        hDir = win32file.CreateFile (
          path_to_watch,
          FILE_LIST_DIRECTORY,
          win32con.FILE_SHARE_READ | win32con.FILE_SHARE_WRITE | win32con.FILE_SHARE_DELETE,
          None,
          win32con.OPEN_EXISTING,
          win32con.FILE_FLAG_BACKUP_SEMANTICS,
          None
        )
```

```
results = win32file.ReadDirectoryChangesW (
hDir,
1024,
True,
win32con.FILE_NOTIFY_CHANGE_FILE_NAME |
 win32con.FILE_NOTIFY_CHANGE_DIR_NAME |
 win32con.FILE_NOTIFY_CHANGE_ATTRIBUTES |
 win32con.FILE_NOTIFY_CHANGE_SIZE |
 win32con.FILE_NOTIFY_CHANGE_LAST_WRITE |
 win32con.FILE_NOTIFY_CHANGE_SECURITY,
None,
None
)
if kode == 11: # Filtered
        try:
                for action, files in results:
                        for fEx in fileEx:
                                if fEx in  files:
                                        act = ACTIONS.get(action, "Unknown")
                                        return ((act,os.path.join(path_to_watch,files)))
        except Exception as e:
                print 'Error in : ',str(e)

if kode == 12:  # NON Filter
        try:
                for action, file in results:
                        fl = str(os.path.join(path_to_watch, file))
                        pf = ACTIONS.get(action, "Unknown") + ' : ' + fl + ', [' + str(getsigf(fl)) + '] '
 # str(type(fl))   getsig(str(fl))
                        return pf
        except Exception as e:
                print 'Error in : ',str(e)

if kode == 13: # only for Test
        for action, file in results:
                full_filename = os.path.join(path_to_watch, file)
                d1 = ACTIONS.get(action, "Unknown")
                d2 = ' => %s :: %s ' % (d1,full_filename)
```

```
                        print d2

        if kode == 14:
                return results

def dumpWindow(hwnd, wantedText=None, wantedClass=None):
    windows = []
    hwndChild = None
    while True:
        hwndChild = win32gui.FindWindowEx(hwnd, hwndChild, wantedClass, wantedText)
        if hwndChild:
            textName = win32gui.GetWindowText(hwndChild)
#            textName = win32gui.GetOpenFileName(hwndChild)
            className = win32gui.GetClassName(hwndChild)
            windows.append((hwndChild, textName, className))
        else:
            return windows


def dumpWindow_ex(hwnd, wantedText=None, wantedClass=None):

        def Getps(hwnd):
                threadpid, procpid = win32process.GetWindowThreadProcessId(hwnd)
                # PROCESS_QUERY_INFORMATION (0x0400) or PROCESS_VM_READ (0x0010) or PROCESS_ALL_ACCESS (0x1F0FFF)
                mypyproc = win32api.OpenProcess(win32con.PROCESS_ALL_ACCESS, False, procpid)
                procname = win32process.GetModuleFileNameEx(mypyproc, 0)
                return os.path.basename(procname)

        windows = []
        hwndChild = None
        while True:
                hwndChild = win32gui.FindWindowEx(hwnd, hwndChild, wantedClass, wantedText)
                if hwndChild:
                    textName = win32gui.GetWindowText(hwndChild)
                    className = win32gui.GetClassName(hwndChild)
                    ps = Getps(hwndChild)
                    tid, pid = win32process.GetWindowThreadProcessId(hwndChild)
                    windows.append((hwndChild, pid, ps,  textName, className))
```

```
                else:
                    return windows


def getexe(pid):
#       get process name from pid given
        ps = psutil.Process(pid)
        return os.path.basename(ps.exe)

def getpidexe(name):
#       get pid from process contain the name string
        system = System()
        for process in system:
                if process.get_filename() != None and name != None:
                        if name.lower() in process.get_filename().lower() :
                                return process.get_pid()

def getwinuser():
#       get windows user name
        import getpass
        return getpass.getuser() #username #username = getpass.getuser()

def getuname(pid):
#       get user name from pid given
        ps = psutil.Process(pid)
        try:
                return ps.username
        except :
                pass

def getuserps():
#       return all process under user/user process
#       return pids and process names
        system = System()
        userps = []
        for process in system:
                try:
                        if getwinuser() in getuname(process.get_pid()) :
```

```
                                userps.append((process.get_pid(),  process.get_filename()))
                except:
                        pass
        return userps

def getuserps_id():
#       return all process under user/user process
#       only return pids
        system = System()
        userps = []
        for process in system:
                try:
                        if getwinuser() in getuname(process.get_pid()) :
                                userps.append((str(process.get_pid())))
                except:
                        pass
        return userps

def getuserps_idx():
#       return all process
#       only return pids
        system = System()
        userps = []
        for process in system:
                try:
                        if process.get_pid() == 0 or process.get_pid() == 4 or process.get_pid() == 8 :
                                continue
                        else:
                                userps.append((str(process.get_pid()), process.get_filename()))
                except:
                        pass
        return userps

curps = getuserps_id()
def detectnewps():
        global curps
        while True:
                time.sleep(0.2)
```

```
                try:
                        tps = getuserps_id()
                        for ps in tps:
                                if ps not in curps:
                                        curps = tps
#                                       write_modul_info(ps) # get collect the process dll
                                        print '.'*5 + 'New Process ->> ' + getexe(int(ps))
#                                       pskill(ps) # kill process after get the dll
                                        getfilesopenbyps(int(ps))
#                                       for l in getfilesopenbyps(int(ps)):
#                                               print ' => : [ %s : %s ] %s' % (l[0],l[1],l[2])
                        if len(curps) != len(tps):
                                curps = tps
                                print '.'*5 + '  ->> Mising Process  '
                        if len(curps) == len(tps):
                                for ps in tps:
                                        if ps in curps:
                                                break
#                                               print '.'*5 + 'TPS :'+ str(len(tps)) + '...CURPS :'+str(len(curps))
+'.'*5
# penggunaan CPU lumayan rendah dibanding tidak menggunakan sleep

                except Exception as E:
                        print 'Error 1 : ' +str(E)

curps = getuserps_id()
def resetnewdebug():
        global curps
        while True:
                try:
                        tps = getuserps_id()
                        if len(curps) != len(tps)  :
                                curps = tps
                                print '='*45
                                print '.'*5 + ' -> Process Changed !.. '
                                for ps in tps:
                                        print ' => ', ps, d1.is_debugee(int(ps))
                                print '='*45
```

```
        #               time.sleep(1) # penggunaan CPU lumayan rendah dibanding tidak menggunakan sleep
        #               time.sleep(0.2)
                except Exception as E:
                        print 'Error 1 : ' +str(E)

def testconn(ip,pr):
        try:
                skt = socket.socket()
                skt.settimeout(2)
                con = skt.connect((ip, pr))
                skt.close()
                if con:
                        return False
                else:
                        return True
#               skt.send(pesan)
        except socket.error, exc:
                pass

def sendata(conn,ip,pr,pesan):
        try:
                if conn == False:pass
                if conn == True:
                        skt = socket.socket()
                        skt.settimeout(2)
                        skt.connect((ip, pr))
                        skt.send(pesan)
                        skt.close()
        except socket.error, exc:
                pass
#               print " => Caught exception socket.error : %s" % exc


def listen():
        ipserver   = '192.168.56.1'
        portserver = 6677

        skt = socket.socket()
```

```
        # untuk mengatasi Error : socket.error: [Errno 98] Address already in use
        skt.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        skt.bind((ipserver, portserver))

        skt.listen(1)
        sambung, alamat = skt.accept()

        while 1:
                isi = sambung.recv(1024)
                if not isi:
                        break
                wfile('listen_6677.py',isi.lower())
        sambung.close()

def ps_info_new(pid):
        ptof ='psutil27_ps_list_len.LOG'
        g    ='----------------------------'
        gg   ='============================'

        def wf(s,ss):
                sss = s + ' \t: ' + str(ss)
                print ' --> ' + sss
#               wtofile(ptof,sss)

        def glist(s,ls):
                try:
                        for p in ls:
                                ss = s + ' \t:: '+ str(p[0])
                                print ss
#                               wtofile(ptof,ss)
                except Exception as e:
                        print s+'Errror : ' + str(e)
                        print gg

#       all_pid = psutil.get_pid_list()
#       for pid in all_pid:

        def getinfo():
```

```
#               global pid
                ps = psutil.Process(pid)
                try:
#                       wtofile(ptof,gg)
                        wf('ps.exe',ps.exe)
                        wf('ps.name',ps.name)
                        wf('ps.parent',ps.parent)
                        wf('ps.pid',ps.pid)
                        wf('ps.ppid',ps.ppid)
                        wf('ps.username',ps.username)
                        wf('ps.getcwd',ps.getcwd())
                        wf('ps.get_children',ps.get_children)
                        glist('ps.cmdline',ps.cmdline)
                        wf('ps.get_conns',ps.get_connections())
                        glist('mmaps.path',ps.get_memory_maps())
                        wf('ps.mmpercen',ps.get_memory_percent())
                        wf('ps.getnice',ps.get_nice())
                        wf('ps.getnumctx',ps.get_num_ctx_switches())
                        wf('ps.getnumhandles',ps.get_num_handles())
                        wf('ps.getnumthreads',ps.get_num_threads())
                        glist('openf.path',ps.get_open_files())
#                       wf('ps.get_threads',ps.get_threads())
                        glist('ps.get_threads',ps.get_threads())
                        print g
                except Exception as E:
                        print ' Error : ',str(E)
#                       pass

#       return getinfo()
        getinfo()

#       app = subprocess.Popen ([r"WriteTime.exe"])
#       getinfo(app.pid)
        print gg


def fhwnd(thefile):
        def Getps(hwnd):
```

```
                '''Acquire the process name from the window handle for use in the log filename.
                1. how to get process name from threadpid
                2. how to get process name from hwnd
                '''
                threadpid, procpid = win32process.GetWindowThreadProcessId(hwnd)
                # PROCESS_QUERY_INFORMATION (0x0400) or
    # PROCESS_VM_READ (0x0010) or PROCESS_ALL_ACCESS (0x1F0FFF)
                mypyproc = win32api.OpenProcess(win32con.PROCESS_ALL_ACCESS, False, procpid)
                procname = win32process.GetModuleFileNameEx(mypyproc, 0)
                return procname
        f = open(thefile,'r')
        hfile = win32file._get_osfhandle(f.fileno())
        hflms = msvcrt.get_osfhandle(f.fileno())
        print '-> ', hfile
        print '-> ', hflms
        print Getps(hfile)
        return (hflms, hfile)

def get_file_ver(fn):
#       return  winappdbg.system.System.get_file_version_info(fn)
        return  system.System.get_file_version_info(fn)

def get_ps_from_wn(wn):
#   get process name from Window manager text/name
#       wn -> exp : 'Program Manager' return explorer.exe
        try:
                hwnd = win32.user32.FindWindowW(None,wn)
                threadpid, procpid = win32process.GetWindowThreadProcessId(hwnd)
                return ((threadpid, procpid,getexe(procpid)))
        except:
                pass

def write_modul_info(pid=None):
        try:
                if pid ==None :
                        pid = os.getpid()

                dlls= []
```

```
                p = psutil.Process( int(pid) )
                for dll in  p.get_memory_maps():
                        if '.dll' in os.path.basename(dll.path):
                                dlls.append(os.path.basename(dll.path).lower())

                exe = getexe(int(pid))
#               dlls= str(sorted(set(dlls)))+ ', # '+ exe
                dlls= str(dlls)+ ', # '+ exe

                startsend('192.168.56.1',6777,dlls)
        except Exception as E:
                print ' Error : ', str(E)

def get_modul_info(pid=None):
        try:
                dlls= []
                if pid ==None :
                        pid = os.getpid()
                p = psutil.Process( int(pid) )
                for dll in  p.get_memory_maps():
                        if '.dll' in os.path.basename(dll.path):
                                dlls.append(os.path.basename(dll.path))
#               dlls=sorted(dlls)
                return dlls
        except Exception as E:
                print ' Error : ', str(E)


def diffpercen(filtr,list1,list2):
        try:
                list1,list2 = set(list1),set(list2)
                diffl = list(set(list1).intersection(list2))
                lenlist1, lendiffl = len(list1), len(diffl)
                if persen(lendiffl,lenlist1) >= int(filtr):
                        return persen(lendiffl,lenlist1)
        except Exception as E:
                print ' Error : ',str(E)
```

```
def difflist(lname,list1,list2):
        list1,list2 = set(list1),set(list2)
        diffl = list(set(list1).intersection(list2))
        lenlist1, lendiffl = len(list1), len(diffl)
        if lenlist1 == lendiffl:
                print ' %s : Sama Hasil List1:%d, List2:%d, Equal:%d, Persen:%s' %
         (lname,lenlist1, len(list2), lendiffl, persen(lendiffl,lenlist1))
        else:
                print ' %s : Tidak Sama Hasil List1:%d, List2:%d, Equal:%d, Persen:%s' %
         (lname,lenlist1, len(list2), lendiffl, persen(lendiffl,lenlist1))


def difflist_1(list1,list2):
        list1,list2 = set(list1),set(list2)
        print 'List1 %d : %s ' % (len(list1),list1)
        print 'List2 %d : %s ' % (len(list2),list2)
        diffl1  = list(set(list1) - set(list2))
        diffl2  = list(set(list2) - set(list1))
        diffl = list(set(list1).intersection(list2))
        if len(diffl1) == 0 :
                print ' All Element of List1 is in List2 [%d:%s] ' % (len(diffl1),diffl1)
        if len(diffl2) == 0 :
                print ' All Element of List2 is in List1 [%d:%s]  ' % (len(diffl2),diffl2)
        if len(diffl1) > 0 :
                print ' Element of List1 not in List2 [%d:%s] ' % (len(diffl1),diffl1)
        if len(diffl2) > 0 :
                print ' Element of List2 not in List1 [%d:%s] ' % (len(diffl2),diffl2)
        if len(diffl) > 0 :
                print ' Element in List1 and List2 [%d:%s] ' % (len(diffl),diffl)

def persen(minA,maxB):
#       hasil = (float(minA) / float(maxB)) * 100.00
#       return str(hasil)[:5]+'%'
        hasil = (float(minA) / float(maxB)) * 100
        return int(hasil)

def checkfile(pathtofile):
```

```
  # fungsi tidak bekerja (sudah di coba di windows 7 tidak berhasil)
        # pathtofile bisa seperti : "c:/windows/*.bmp"
        for fl in glob.glob(pathtofile):
                print ' => ',fl

def pspause(pid):
        try:
                p = psutil.Process(int(pid))
                p.suspend()
        except:
                pass

def psresume(pid):
        try:
                p = psutil.Process(int(pid))
                p.resume()
        except:
                pass

def pskill(pid):
        try:
                p = psutil.Process(int(pid))
                p.kill()
                p.terminate()
        except:
                pass

def kill_tree(pid, sig=signal.SIGTERM, include_parent=True,timeout=None, on_terminate=None):
        """Kill a process tree (including grandchildren) with signal
        "sig" and return a (gone, still_alive) tuple.
        "on_terminate", if specified, is a callabck function which is
        called as soon as a child terminates.
        """
        if pid == os.getpid():
                raise RuntimeError("I refuse to kill myself")
        parent = psutil.Process(pid)
        children = parent.children(recursive=True)
        if include_parent:
```

```
                children.append(parent)
        for p in children:
                p.send_signal(sig)
        gone, alive = psutil.wait_procs(children, timeout=timeout, callback=on_terminate)
        return (gone, alive)

def hitevent(sc):
        try:
                t1 = datetime.now()
                while (datetime.now()-t1).seconds <= int(sc):
                        #do something
                        print(datetime.now())
        finally:
                print 'Menghitung....'
                print 'Selesai.......'

def hitevent2(sc):
#       gunakan dibawah untuk check data count
        try:
                t1 = datetime.now()
                while 1:
                        print ' => ' +str(t1)+ ' :: ' +str(datetime.now())
                        if (datetime.now()-t1).seconds > sc:
                                break
        finally:
                print 'Selesai ....'

def getsigf(fname):
        try:
                with open(fname,'rb') as rf:
                        dat= binascii.b2a_hex(rf.read(12))
                        return dat.upper()
        except:
                pass

def getsig(fname):
        # return file_signature if match, else return None
        try:
```

```
                with open(fname,'rb') as rf:
                        dat= binascii.b2a_hex(rf.read(12))
                        for sig in fsign:
                                if sig in dat.upper():
                                        return fsign.get(sig.upper(),'UNKNOWN')
        except:
                pass
#       except Exception as E:
#               print ' getsig Error : ',str(E)

def getfsig(fname):
        try:
                hasil = getsig(fname)
                if hasil != None:
                        return str(hasil.upper())
                else:
#                       return "UNKNOWN"
                        return "unknown"
        except:
                pass
#       except Exception as E:
#               print ' getfsig Error : ',str(E)

def isfile(fname):
        try:
                if os.path.isfile(fname) == True:
                        return "EXIST"
                else:
                        return "MISSING"
        except:
                pass
#       except Exception as E:
#               print ' isfile Error : ',str(E)

def luniq(lst):
    try:
        last = object()
        for item in lst:
```

```
            if item == last:
                continue
            yield item
            last = item
    except:
        pass

def sort_deduplicate(l):
        try:
                return list(luniq(sorted(l, reverse=True)))
        except:
                pass

def print_report(rpt,r1,r2,allf,fe,fs):
        try:
#               tmp=sp.call('cls',shell=True) # bersihkan layar
                created,deleted,changed,renfom,rento =0,0,0,0,0
                opfile,crtps = 0,0
                allps = set()
        #-----------------------------------------------

                rall = r1 + r2
                if r1 > 0: # all data events in list1
                        for l1 in r1:
                                allps.add(l1[1])
                                if 'OPEN_FILE'      in l1:opfile = opfile +1
                                if 'CREATE_PROCESS' in l1:crtps  = crtps  +1

                if r2 > 0: # all data events in list2
                        for l2 in r2:
                                if 'CREATED' in l2:created = created +1
                                if 'DELETED' in l2:deleted = deleted +1
                                if 'CHANGED' in l2:changed = changed +1
                                if 'REN_Frm' in l2:renfom  = renfom  +1
                                if 'REN__To' in l2:rento   = rento   +1


                renall = int(renfom) + int(rento)
```

```
                waktu = datetime.now()
                print '*'*55
                print ' Laporan Aktifitas Mengakses Files Oleh Prosess : '
                print '-'*55
                print ' Laporan Normor        :', str(rpt)
                print ' Laporan untuk Waktu   :', waktu
                print ' Total Events          : %s [%s/%s]' % (str(len(rall)),str(len(r1)),str(len(r2)))

                print '    Open Events        : %s ' % str(opfile)
                print '    Created Events     : %s ' % str(created)
                print '    Changed Events     : %s ' % str(changed)
                print '    Deleted Events     : %s ' % str(deleted)
                print '    Renamed Events     : %s [%s/%s] ' % (str(renall),str(renfom),str(rento))
                print ' Total Files           : %s ' % str(len(allf))
                print '    Exist Files        : %s Exist / %s Missing ' % (str(fe), str(len(allf) - fe))     # From
Param
                print '    Signature Files    : %s Known / %s Unknown ' % (str(fs),str(len(allf)-fs))        # From
Param
                print ' Process               : %s ' % str(len(allps))
                print '    Process Create     : %s ' % str(crtps)
                print ' ' + '-'*55
                print ' Hasil/Status          : %s' % ceksmax(rall,changed,renall,len(allf),len(allf)-fs)
#-------------------------------------------------------------
#               TEST PRINT COUNT EXTENSION DATA
#               count_exten(rall)
                print '*'*55
        except:
                pass

def ceksmax(rall,ch,rn,tf,fg):
                # buat variable "smax" sebagai standar maximum event per report
                # tetapkan smax= 100
                # jika "rall" > "smax" maka tampilkan status = "ABNORMAL Behavior"
                # Tampilkan Nama App belum ada (plan)
                # Variabel yang digunakan adalah sesuai dengan data yang didapatkan pada malware yaitu
                # Variabel = rall:Total_Events, ch:Changed_Events, rn:Renamed_Events, tf:Total_File,
fg:File_Signature
                #--------------------------------
```

```
                # sebelumnya smax=100, diubah jadi smax=50, karena
                # untuk setiap komputer berbeda kecepatan baca datanya sehingga,
                # smax=50 dirasa sangat ideal untuk standar file yang diakses per/n-detik
                smax = 50
                smsg = ""
                #if (len(rall) >= smax or op >= smax or cr >= smax or ch >= smax or de >= smax):
                if (len(rall) >= smax and ch >= smax and rn >= smax and tf >= smax and fg >= smax ):
                        smsg = "ABNORMAL Behavior"
                else:
                        smsg = "Normal Behavior"
                return smsg

def print_event(data_list):
        #       print the list of data in data_list
        #       check data len and format the output
        for dl in data_list:
                if len(dl)   == 2:
                        sdl = " %s : %s" % (dl[0],dl[1])
#               elif len(dl) == 3:
#                       sdl = " %s : %s : %s" % (dl[0],dl[1],dl[2])
                elif len(dl) == 4:
                        sdl = " %s : %s : %s : %s" % (dl[0],dl[1],dl[2],dl[3])
                elif len(dl) == 5:
                        sdl = " %s : %s : %s : %s : %s" % (dl[0],dl[1],dl[2],dl[3],dl[4])

                print "=> %s " % sdl
#               print "=> %s : %s" % (os.path.basename(sdl),os.path.splitext(sdl)[1])


def count_exten(data_list):
        #       count extention in data_list
        allext=[]
        for dl in data_list:
                if len(dl)   == 2:
                        sdl = "%s" % dl[1]
                elif len(dl) == 4:
                        sdl = "%s" % dl[3]
                elif len(dl) == 5:
```

```
                        sdl = "%s" % dl[3]

                for ext in data_filter:
                        fex = os.path.splitext(sdl)[1]
                        if ext in fex:
                                allext.append(fex)
        ae = Counter(allext)
        for ex in ae:
                print "                                %s : %s" % (ex,ae[ex])

def getfilesopenbyps(pid,mylist):
        ps = psutil.Process(pid)
        try:
                psn = ps.name
                for psc in ps.cmdline:
                        if psc != '/n':
                                mylist.append([psn,psc])
                for psg in ps.get_open_files():
                        mylist.append([psn,psg[0]])
        except:
                pass
#       except Exception as E:
#               print ' Error : ',str(E)

def getlistitem(mylist,s):
        try:
                for l in mylist:
                        if s in str(l[1]):
                                return l[0]
                        else:
                                return " N/A "
        except:
                pass

def testps():
        time.sleep(5)
        app = subprocess.Popen ([r"calc.exe"])
#       print 'PID : ',app.pid
```

```
time.sleep(2)
pskill(app.pid)
```

LAMPIRAN B.3

FILE NAME : myvar.py

```
#-------------------------------------------

data_filter = ['.zip','.osiris','.doc','.docx','.ppt','.pptx','.xls','.xlsx','.jpg','.jpeg','.png','.pdf','.odg'] #
the Experiment

#filesign =
['504B0304','504B0506','504B0708','D0CF11E0A1B11AE1','89504E470D0A1A0A','25504446','FFD8FFDB','FFD8FFE0','FFD8FFE1',
'FFD8FFE2','FFD8FFE3','FFD8FFE8','FFD8FFE000104A4649460001']

filesign = [
'504B0304', # docx,xlsx,pptx,ods,odp,vsdx
'504B0506', # docx,xlsx,pptx,ods,odp,vsdx : Empty
'504B0708', # docx,xlsx,pptx,ods,odp,vsdx : ?
'D0CF11E0A1B11AE1', # doc,xls,ppt
'89504E470D0A1A0A', # png
'25504446', # pdf
'FFD8FFDB', # jpg,jpeg, Samsung D807 Standard
'FFD8FFE0', # jpg,jpeg Standard
'FFD8FFE1', # jpg,jpeg Standard
'FFD8FFE2', # jpg,jpeg Canon EOS Standard
'FFD8FFE3', # jpg,jpeg Samsung D500 Standard
'FFD8FFE8', # jpg,jpeg, Samsung Standard
'FFD8FFE000104A4649460001' # jpg,jpeg
]
#       act = {
#         1 : "CREATED",
#         2 : "DELETED",
#         3 : "CHANGED",
#         4 : "REN_Frm",
#         5 : "REN__To"
#       }

fsignOLD = {
```

```
                '504B0304':'docx,xlsx,pptx,ods,odp', # docx,xlsx,pptx,ods,odp,vsdx
                '504B0506':'docx,xlsx,pptx,ods,odp', # docx,xlsx,pptx,ods,odp,vsdx : Empty
                '504B0708':'docx,xlsx,pptx,ods,odp', # docx,xlsx,pptx,ods,odp,vsdx : ?
                'D0CF11E0A1B11AE1':'doc,xls,ppt', # doc,xls,ppt
                '89504E470D0A1A0A':'png', # png
                '25504446':'pdf', # pdf
                'FFD8FFDB':'jpg,jpeg', # jpg,jpeg, Samsung D807 Standard
                'FFD8FFE0':'jpg,jpeg', # jpg,jpeg Standard
                'FFD8FFE1':'jpg,jpeg', # jpg,jpeg Standard
                'FFD8FFE2':'jpg,jpeg', # jpg,jpeg Canon EOS Standard
                'FFD8FFE3':'jpg,jpeg', # jpg,jpeg Samsung D500 Standard
                'FFD8FFE8':'jpg,jpeg', # jpg,jpeg, Samsung Standard
                'FFD8FFE000104A4649460001':'jpg,jpeg' # jpg,jpeg
        }

#files_signatures
fsign = {
                '504B0304':'Documentx', # docx,xlsx,pptx,ods,odp,vsdx
                '504B0506':'DocumentX', # docx,xlsx,pptx,ods,odp,vsdx : Empty
                '504B0708':'DocumentX', # docx,xlsx,pptx,ods,odp,vsdx : ?
                'D0CF11E0A1B11AE1':'Document', # doc,xls,ppt
                '89504E470D0A1A0A':'Image', # png
                '25504446':'PdfDocment', # pdf
                'FFD8FFDB':'Image', # jpg,jpeg, Samsung D807 Standard
                'FFD8FFE0':'Image', # jpg,jpeg Standard
                'FFD8FFE1':'Image', # jpg,jpeg Standard
                'FFD8FFE2':'Image', # jpg,jpeg Canon EOS Standard
                'FFD8FFE3':'Image', # jpg,jpeg Samsung D500 Standard
                'FFD8FFE8':'Image', # jpg,jpeg, Samsung Standard
                'FFD8FFE000104A4649460001':'Image' # jpg,jpeg
        }

fts='mydb.py'
# ******************************************************************
```