security scripts

- uber basic check for kelogers.py
- start-win-defender.bat
- check for systemstats.py v1
- check for pc's pids names and usage stats and send them to a file to search later .py v2
- pid killer (ranges too)
- block all .zip .rar .mov top level domains

uber basic check for kelogers.py

```
import psutil
import os
import sys

def find_suspicious_processes():
    suspicious_processes = []
    for process in psutil.process_iter(['pid', 'name', 'exe', 'cmdline']):
        try:
            if process.info['exe'] and process.info['cmdline']:
                exe_name = os.path.basename(process.info['exe']).lower()
                cmdline = ' '.join(process.info['cmdline']).lower()

                if 'keylogger' in exe_name or 'keylogger' in cmdline:
                    suspicious_processes.append(process)
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass
    return suspicious_processes

def main():
    suspicious_processes = find_suspicious_processes()

    if not suspicious_processes:
        print("No suspicious processes found.")
    else:
        print("Suspicious processes found:")
        for process in suspicious_processes:
            print(f"PID: {process.info['pid']} - Name: {process.info['name']} - Exe: {process.info['exe']}
- Cmdline: {' '.join(process.info['cmdline'])}")

if __name__ == '__main__':
    main()
```

start-win-defender.bat

```
@echo off
echo Starting Windows Defender malware scan...
"%ProgramFiles%\Windows Defender\MpCmdRun.exe" -Scan -ScanType 1
echo Scan complete.
pause
```

check for systemstats.py v1

```
import psutil
import datetime


def check_high_memory_usage(threshold=50):
    high_memory_usage_processes = []
    total_memory = psutil.virtual_memory().total
    for proc in psutil.process_iter(['pid', 'name', 'memory_info']):
        try:
            memory_percent = (proc.info['memory_info'].rss / total_memory) * 100
            if memory_percent > threshold:
                high_memory_usage_processes.append((proc, memory_percent))
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass

    return high_memory_usage_processes


def write_processes_to_file():
    now = datetime.datetime.now()
    file_name = f"processes_{now:%Y-%m-%d_%H-%M-%S}.txt"
    with open(file_name, 'w') as f:
        try:
            f.write(f"List of highest CPU usage processes on {now}:\n\n")
            for proc in sorted(psutil.process_iter(['pid', 'name', 'memory_percent'
, 'cpu_percent']), key=lambda p: p.info['cpu_percent'], reverse=True):
                try:
                    cpu_percent = proc.info['cpu_percent']
                    if cpu_percent > 0.0:
```

```
                        f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name'
]} - CPU%: {cpu_percent:.2f} - Memory%: {proc.info['memory_percent']:.2f}\n")

                        f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f
}%\n")

                        f.write(f"\tNetwork usage: {psutil.net_io_counters().
bytes_sent/1024:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}
KB received\n")

                except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.
ZombieProcess):

                    pass

        except:

            f.write("An error occurred while writing the file.\n")



        f.write("\n\n
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n\n")

        f.write(f"List of highest memory usage processes on {now}:\n\n")

        for proc, mem_percent in sorted(check_high_memory_usage(), key=lambda p: p[
1], reverse=True):

            f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name']}
 - Memory%: {mem_percent:.2f}\n")

            f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f}%\n")

            f.write(f"\tNetwork usage: {psutil.net_io_counters().bytes_sent/1024
:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}KB received\n")

        f.write("\n\n
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n\n")

        f.write(f"List of all running processes on {now}:\n\n")

        for proc in psutil.process_iter(['pid', 'name', 'memory_percent',
'cpu_percent']):

            try:

                cpu_percent = proc.info['cpu_percent']

                mem_percent = proc.info['memory_percent']

                if cpu_percent > 0.0:

                    f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name']}
 - CPU%: {cpu_percent:.2f} - Memory%: {mem_percent:.2f}\n")

                    f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f}%\n
")

                    f.write(f"\tNetwork usage: {psutil.net_io_counters().bytes_sent
/1024:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}KB received\n")
```

```
            except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess
):
                pass


def main():
    write_processes_to_file()


if __name__ == '__main__':
    print("checking")
    main()
    print("done")
```

check for pc's pids names and usage stats and send them to a file to search later .py v2

```
import os
import psutil
import datetime

def check_high_memory_usage(threshold=50):
    high_memory_usage_processes = []
    total_memory = psutil.virtual_memory().total
    for proc in psutil.process_iter(['pid', 'name', 'memory_info']):
        try:
            memory_percent = (proc.info['memory_info'].rss / total_memory) * 100
            if memory_percent > threshold:
                high_memory_usage_processes.append((proc, memory_percent))
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            pass

    return high_memory_usage_processes

def write_processes_to_file():
    current_pid = os.getpid()
    now = datetime.datetime.now()
```

```
    file_name = f"processes_{now:%Y-%m-%d_%H-%M-%S}.txt"

    with open(file_name, 'w') as f:

        f.write(f"List of all running processes on {now}:\n\n")

        for proc in psutil.process_iter(['pid', 'name', 'memory_percent',
'cpu_percent']):

            try:

                if proc.info['pid'] != current_pid:  # Exclude the current script

                    cpu_percent = proc.info['cpu_percent']

                    mem_percent = proc.info['memory_percent']

                    f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name']}
 - CPU%: {cpu_percent:.2f} - Memory%: {mem_percent:.2f}\n")

                    f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f}%\n
")

                    f.write(f"\tNetwork usage: {psutil.net_io_counters().bytes_sent
/1024:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}KB received\n")

            except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess
):

                pass



        f.write("\n\n
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n\n")

        f.write(f"List of highest CPU usage processes on {now}:\n\n")

        for proc in sorted(psutil.process_iter(['pid', 'name', 'memory_percent',
'cpu_percent']), key=lambda p: p.info['cpu_percent'], reverse=True):

            try:

                if proc.info['pid'] != current_pid:  # Exclude the current script

                    cpu_percent = proc.info['cpu_percent']

                    if cpu_percent > 0.0:

                        f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name'
]} - CPU%: {cpu_percent:.2f} - Memory%: {proc.info['memory_percent']:.2f}\n")

                        f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f
}%\n")

                        f.write(f"\tNetwork usage: {psutil.net_io_counters().
bytes_sent/1024:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}
KB received\n")
```

```
            except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess
):

                pass



        f.write("\n\n
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=\n\n")

        f.write(f"List of highest memory usage processes on {now}:\n\n")

        for proc, mem_percent in sorted(check_high_memory_usage(), key=lambda p: p[
1], reverse=True):

            f.write(f"PID: {proc.info['pid']} - Name: {proc.info['name']}
 - Memory%: {mem_percent:.2f}\n")

            f.write(f"\tDisk usage: {psutil.disk_usage('/').percent:.2f}%\n")

            f.write(f"\tNetwork usage: {psutil.net_io_counters().bytes_sent/1024
:.2f}KB sent/{psutil.net_io_counters().bytes_recv/1024:.2f}KB received\n")



def main():

    write_processes_to_file()



if __name__ == '__main__':

    main()
```

pid killer (ranges too)

```
import psutil



while True:

    pids = input(
"Type the PID(s) you want to kill, separated by commas, or specify a range with a da
)

    if pids.lower() == "exit" or pids.lower() == "stop":

        confirm = input("Are you sure you want to stop the script? (Y/N): ")

        if confirm.lower() in ["y", "yes"]:

            break

        else:

            continue

    if '-' in pids:

        start, end = pids.split('-')

        pids_list = [str(pid) for pid in range(int(start), int(end) + 1)]

        yes_all = input(
"Do you want to kill all processes in the range without confirmation? (Y/N): ")

        if yes_all.lower() in ['y', 'yes']:

            response = 'y'

        elif yes_all.lower() in ['n', 'no']:

            response = ''

        else:

            response = ''

    else:

        pids_list = pids.split(",")

        response = ''



    for pid in pids_list:
```

```
        try:

            process = psutil.Process(int(pid))

            name = process.name()

            mem_usage = process.memory_info().rss / 1024 / 1024

            cpu_usage = process.cpu_percent()

            net_io_counters = psutil.net_io_counters(pernic=False)

            network_usage = net_io_counters.bytes_sent / 1024 / 1024 +
net_io_counters.bytes_recv / 1024 / 1024

            disk_usage = process.io_counters().write_bytes / 1024 / 1024 + process.
io_counters().read_bytes / 1024 / 1024

            if response.lower() in ['y', 'yes']:

                process.kill()

                print(f"Process with PID {pid} ({name}) terminated.")

            elif response.lower() in ['n', 'no']:

                response = input(f"Y/N are you sure that you want to kill PID {pid}
 ({name}) current: mem {mem_usage:.2f}MB, CPU {cpu_usage:.2f}%, net {network_usage
:.2f}MB, disk {disk_usage:.2f}MB? ")

                if response.lower() in ['y', 'yes', 'yeah', 'yep', 'sure', 'ok',
'okay', 'fine', 'affirmative', 'positive']:

                    process.kill()

                    print(f"Process with PID {pid} ({name}) terminated.")

                else:

                    print(f"Skipped terminating process with PID {pid} ({name}).")

            else:

                response = input(f"Y/N are you sure that you want to kill PID {pid}
 ({name}) current: mem {mem_usage:.2f}MB, CPU {cpu_usage:.2f}%, net {network_usage
:.2f}MB, disk {disk_usage:.2f}MB? ")

                if response.lower() in ['y', 'yes', 'yeah', 'yep', 'sure', 'ok',
'okay', 'fine', 'affirmative', 'positive']:

                    process.kill()

                    print(f"Process with PID {pid} ({name}) terminated.")

                else:

                    print(f"Skipped terminating process with PID {pid} ({name}).")

        except (psutil.NoSuchProcess, psutil.AccessDenied, ValueError) as e:

            print(f"Error: Cannot kill process with PID {pid}. Reason: {e}")
```

block all .zip .rar .mov top level domains

current research
impossible