

Forum:[Hack Forums](#)

Thread: [Malware & Suspicious Process Detection/Removal Script](#)

G-DillahKillah - September 7, 2025, 8:25 pm

It doesn't "Remove" everything automatically, but it can terminate suspicious PID's and some files. It will list one's to manually check as well.

STEP 1:

Download the file...

Code

```
import psutil
```

```
import hashlib
```

```
import subprocess
```

```
import socket
```

```
import os
```

```
import sys
```

```
os.system("sudo apt install rkhunter")
```

```
os.system("sudo apt install chkrootkit")
```

Optional: VirusTotal API key for hash reputation

```
VIRUSTOTAL_API_KEY = '<<<<<INSERT VIRUS TOTAL API KEY HERE!!!>>>>>'
```

```
def clear_screen():
```

```
    subprocess.run("clear" if os.name == "posix" else "cls")
```

--- Detection Functions ---

```
def rkhunt():
```

```
    subprocess.run(["rkhunter", "--check", "--rwo"])
```

```
def get_process_tree(proc):
```

```
    """Recursively get all child processes."""
```

```
    try:
```

```
        children = proc.children(recursive=True)
```

```
        return [proc] + children
```

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

```
        return []
```

```
def compute_file_hash(file_path):
```

```
    """Compute SHA256 hash of a file."""
```

```
    try:
```

```
        with open(file_path, 'rb') as f:
```

```
            return hashlib.sha256(f.read()).hexdigest()
```

```
    except:
```

```
        return None
```

```
def check_hash_reputation(hash_value):
```

```
    """Check hash against VirusTotal."""
```

```
    if not VIRUSTOTAL_API_KEY:
```

```
        return None
```

```

import requests
url = 'https://www.virustotal.com/api/vtapi/v2/file/report'
params = {'apikey': VIRUSTOTAL_API_KEY, 'resource': hash_value}
try:
    response = requests.get(url, params=params)
    if response.status_code == 200:
        return response.json()
except:
    pass
return None

def get_suspicious_processes():
    suspicious = []
    for proc in psutil.process_iter(['pid', 'name', 'exe', 'cmdline']):
        try:
            name = proc.info['name']
            # Heuristics
            if not proc.info['exe']:
                suspicious.append(('No executable path', proc))
            elif any(keyword in name.lower() for keyword in ['suspicious', 'malware', 'bad']):
                suspicious.append(('Suspicious name', proc))
            if proc.info['cmdline']:
                cmdline = ' '.join(proc.info['cmdline']).lower()
                if any(keyword in cmdline for keyword in ['powershell', 'wscript', 'cscript', 'curl', 'wget']):
                    suspicious.append(('Suspicious cmdline', proc))
            # Check executable hash against online database
            exe_path = proc.info['exe']
            if exe_path and os.path.exists(exe_path):
                hash_value = compute_file_hash(exe_path)
                report = check_hash_reputation(hash_value)
                if report and report.get('response_code') == 1:
                    if any(scan.get('malicious', False) for scan in report['scans'].values() if isinstance(scan, dict)):
                        suspicious.append(('Malicious hash', proc, exe_path))
            except (psutil.NoSuchProcess, psutil.AccessDenied):
                continue
    return suspicious

def get_network_connections():
    suspicious = []
    for conn in psutil.net_connections(kind='inet'):
        if conn.status == 'ESTABLISHED' and conn.raddr:
            ip = conn.raddr.ip
            port = conn.raddr.port
            if is_suspicious_ip(ip):
                suspicious.append((ip, port))
    return suspicious

def is_suspicious_ip(ip):
    try:
        socket.inet_aton(ip)
        octets = list(map(int, ip.split('.')))
        if (octets[0] == 10 or

```

```

        (octets[0] == 172 and 16 <= octets[1] <= 31) or
        (octets[0] == 192 and octets[1] == 168)):
            return False
    return True
except socket.error:
    return False

def get_open_files():
    suspicious_files = []
    for proc in psutil.process_iter(['pid', 'name']):
        try:
            for file in proc.open_files():
                if any(keyword in file.path.lower() for keyword in ['tmp', 'temp', 'update', 'random',
'malicious']):
                    suspicious_files.append((proc, file))
        except (psutil.NoSuchProcess, psutil.AccessDenied):
            continue
    return suspicious_files

def get_cron_jobs():
    cron_paths = ['/etc/crontab', '/etc/cron.*', '~/.cron', '~/.bashrc']
    cron_jobs = []
    for path in cron_paths:
        try:
            with open(os.path.expanduser(path), 'r') as f:
                lines = f.readlines()
                for line in lines:
                    if any(keyword in line.lower() for keyword in ['wget', 'curl', 'bash', 'sh']):
                        cron_jobs.append((path, line.strip()))
        except:
            continue
    return cron_jobs

# --- Remediation Functions ---

def terminate_process(proc):
    try:
        # Kill process tree
        for p in get_process_tree(proc.pid):
            p.terminate()
        psutil.wait_procs([proc], timeout=3)
        print(f"Terminated process {proc.pid} ({proc.name()})")
    except Exception as e:
        print(f"Error terminating process {proc.pid}: {e}")

def quarantine_file(file_path):
    try:
        quarantine_dir = "/tmp/quarantine"
        os.makedirs(quarantine_dir, exist_ok=True)
        base_name = os.path.basename(file_path)
        dest_path = os.path.join(quarantine_dir, base_name)
        os.rename(file_path, dest_path)
        print(f"Moved {file_path} to quarantine.")
    
```

```

    return dest_path
except Exception as e:
    print(f"Error quarantining {file_path}: {e}")
    return None

def delete_file(file_path):
    try:
        os.remove(file_path)
        print(f"Deleted {file_path}")
    except Exception as e:
        print(f"Error deleting {file_path}: {e}")

def run_offline_scans():
    """Run rkhunter and chkrootkit if available."""
    print("Running offline rootkit scans...")
    print(f"Running Chkrootkit...")
    print("1) Expert Mode")
    print("2) Normal")
    choice=input("Mode Number: ")
    if choice == "1":
        subprocess.run(["chkrootkit", "-q", "-x"])
    if choice == "2":
        subprocess.run(["chkrootkit", "-q"])
    print("Offline scans completed.")

def prompt_user(prompt):
    return input(prompt + ' [y/N]: ').lower() == 'y'

# --- Main Function ---

def main():
    clear_screen()
    print("=== Linux Malware Detection & Remediation ===\n")

    # 1. Detect suspicious processes
    suspicious_procs = get_suspicious_processes()
    if suspicious_procs:
        print("\n[!] Suspicious Processes Detected:")
        for desc, proc in suspicious_procs:
            print(f"PID {proc.pid} | Name: {proc.name()}")
    else:
        print("\n[+] No suspicious processes detected.")

    # 2. Detect suspicious network connections
    suspicious_conns = get_network_connections()
    if suspicious_conns:
        print("\n[!] Suspicious Network Connections:")
        for ip, port in suspicious_conns:
            print(f"{ip}:{port}")
    else:
        print("\n[+] No suspicious network activity detected.")

    # 3. Open files

```

```

open_files = get_open_files()
if open_files:
    print("\n[!] Suspicious open files:")
    for proc, file in open_files:
        print(f"PID {proc.pid} ({proc.name()}) - {file.path}")
else:
    print("\n[+] No suspicious open files.")

# 4. Cron jobs
cron_jobs = get_cron_jobs()
if cron_jobs:
    print("\n[!] Suspicious cron jobs:")
    for path, line in cron_jobs:
        print(f"{path}: {line}")
else:
    print("\n[+] No suspicious cron jobs detected.")

# --- Automated Remediation ---
# Terminate suspicious processes
for desc, proc in suspicious_procs:
    if prompt_user(f"Terminate process {proc.pid} ({proc.name()})?"):
        terminate_process(proc)

# Quarantine/delete suspicious files
for proc, file in open_files:
    if prompt_user(f"Quarantine file {file.path}?"):
        quarantine_file(file.path)
    elif prompt_user(f>Delete file {file.path}?"):
        delete_file(file.path)

# Run offline scans
if prompt_user("Run offline rootkit scans?"):
    rkhunt()
    run_offline_scans()
print("\n[!] Remediation actions completed.")
print("Review logs and verify system integrity.")

```

```

if __name__ == "__main__":

```

```

    import shutil
    main()

```

STEP 2:

- Insert your virus total API key in the place holder
- Save as abclinux4.py

STEP 3:

```

Code sudo chmod +x abclinux4.py
sudo python3 abclinux4.py

```

(For Linux)