Microproject of CCL 202 : Scripting Languages for Security

Network Twin

Batch: 2023-2027

Branch & Semester: S4 CSE-CY

Group Members

4, ABRAHAM JUDE BAIJU

8, AIDAN JAISON

14, AMALEKH BIJU

24, BERNARD MATHEW JOHN

30, GOURI NANDA M S

31, GOURI SREENATH

Introduction

A Network Twin is a virtual, dynamic replica of a physical network infrastructure. It mirrors components like routers, switches, servers, and endpoints, along with their configurations, connections, and potentially even traffic patterns. This concept is increasingly vital in cybersecurity. By providing a high-fidelity sandbox, network twins allow security teams to safely simulate attacks, test defenses (like firewall rules or Intrusion Detection Systems), analyze vulnerabilities, and understand network behavior without risking the live operational environment. This proactive approach enhances security posture and incident response capabilities.

This microproject develops a foundational component for building or monitoring such a twin: a script that profiles a single network node. We use Python, a powerful scripting language widely used in security, along with the psutil library, to automatically collect critical security-relevant information from the local machine. This includes system identification, active network listeners (open ports), established network connections, and currently running processes. This data snapshot represents the state of a node, which is essential information for constructing or verifying its representation within a network twin.

Outline of Microproject

- **Objective:** To develop a Python script that automatically gathers key security-relevant system and network state information from the local machine it's run on.
- **Scope:** The script will collect and display the following information:
  - Basic System Information: Hostname, Operating System type and version, system architecture.
  - Listening Network Ports: TCP and UDP ports awaiting incoming connections, including the process associated with each port (requires appropriate permissions).
  - Active Network Connections: Established TCP connections showing local and remote endpoints, and the associated process (requires appropriate permissions).
  - Running Processes: A list of currently executing processes, including their Process ID (PID), name, and the user running them.
- **Tools/Languages:**
  - Language: **Python 3**
  - Libraries: platform (for system info), socket (implicitly used by psutil for network types), psutil (for network connections and processes), datetime (for timestamping).
- **Methodology:**
  1. The script utilizes the platform module to fetch OS and hostname details.
  2. The psutil library's net_connections() function is used to retrieve all network sockets. These are then filtered to identify 'LISTEN' status sockets (listening ports) and 'ESTABLISHED' status sockets (active connections). Process information for each socket is obtained using the PID provided by psutil, handling potential access errors.
  3. The psutil library's process_iter() function is used to iterate through all running processes, extracting PID, name, and username for each. Error handling is included for processes that might terminate during iteration or require special permissions.
  4. All collected information is formatted and printed clearly to the standard output, section by section.
  5. A timestamp is included to indicate when the profile snapshot was taken.

Code/Program

Python

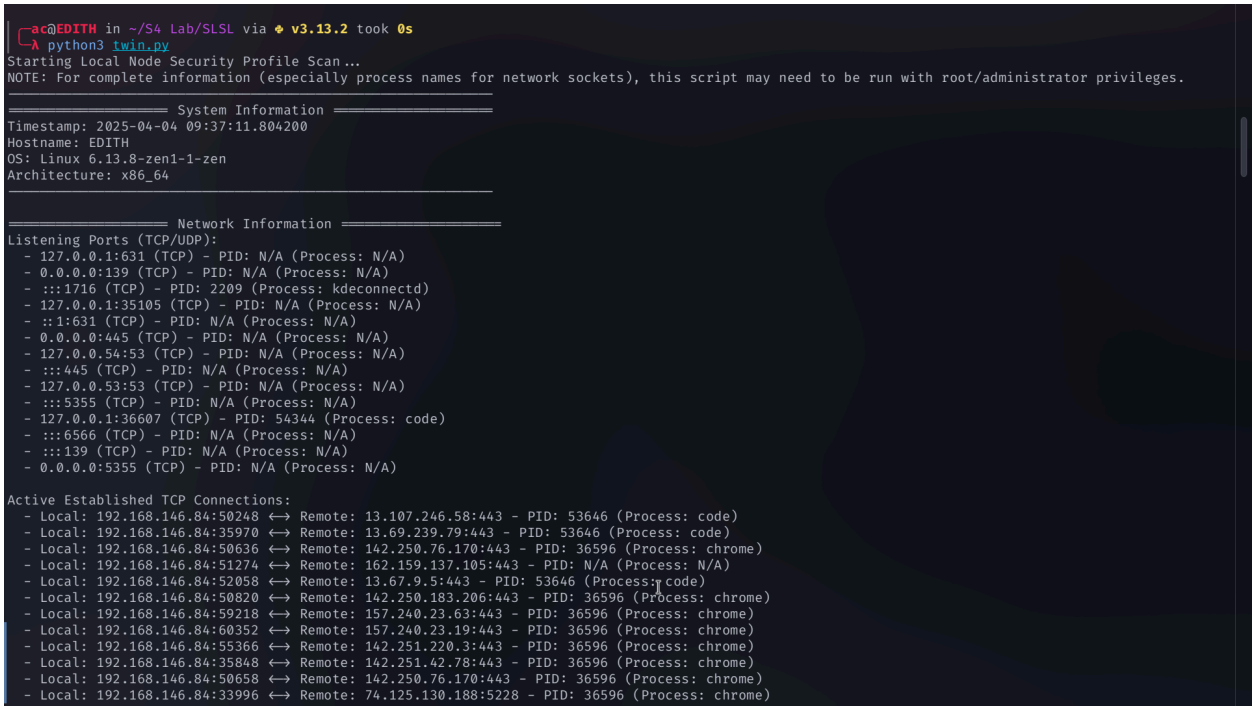
```
# Filename: node_profiler.py

#!/usr/bin/env python3

import socket
import psutil
import platform
from datetime import datetime

def get_system_info():
    """Gathers and prints basic system information."""
    print("="*20 + " System Information " + "="*20)
    try:
        print(f"Timestamp: {datetime.now()}")
        print(f"Hostname: {platform.node()}")
        print(f"OS: {platform.system()} {platform.release()}")
        print(f"Architecture: {platform.machine()}")
    except Exception as e:
        print(f"  [Error] Could not retrieve system info: {e}")
    print("-" * 60)

def get_network_info():
    """Gathers and prints network port and connection information."""
    print("\n" + "="*20 + " Network Information " + "="*20)
    listening_ports = []
    active_connections = []

    print("Listening Ports (TCP/UDP):")
    try:
        connections = psutil.net_connections(kind='inet') # 'inet' covers IPv4 and IPv6
        for conn in connections:
            # conn.laddr = local address (ip, port), conn.raddr = remote address
            # conn.status = e.g., 'LISTEN', 'ESTABLISHED'
```

```
        # conn.pid = process ID using the connection

        # --- Listening Ports ---
        if conn.status == psutil.CONN_LISTEN and conn.laddr:
            proc_name = 'N/A'
            if conn.pid:
                try:
                    proc_name = psutil.Process(conn.pid).name()
                except (psutil.NoSuchProcess, psutil.AccessDenied):
                    proc_name = '[Access Denied/Gone]'
                except Exception as e:
                    proc_name = f'[Error: {e}]'

            # Determine protocol type based on conn.type
            proto = 'TCP' if conn.type == socket.SOCK_STREAM else 'UDP' if conn.type ==
socket.SOCK_DGRAM else 'Other'

            listening_ports.append(f"  - {conn.laddr.ip}:{conn.laddr.port} ({proto}) - PID: {conn.pid
or 'N/A'} (Process: {proc_name})")

        # --- Active Connections (mainly TCP) ---
        elif conn.status == psutil.CONN_ESTABLISHED and conn.laddr and conn.raddr:
            proc_name = 'N/A'
            if conn.pid:
                try:
                    proc_name = psutil.Process(conn.pid).name()
                except (psutil.NoSuchProcess, psutil.AccessDenied):
                    proc_name = '[Access Denied/Gone]'
                except Exception as e:
                    proc_name = f'[Error: {e}]'

            # Determine protocol type
            proto = 'TCP' if conn.type == socket.SOCK_STREAM else 'UDP' if conn.type ==
socket.SOCK_DGRAM else 'Other'
            # Typically established connections are TCP
            if proto == 'TCP':
                active_connections.append(f"  - Local: {conn.laddr.ip}:{conn.laddr.port} <->
Remote: {conn.raddr.ip}:{conn.raddr.port} - PID: {conn.pid or 'N/A'} (Process: {proc_name})")
```

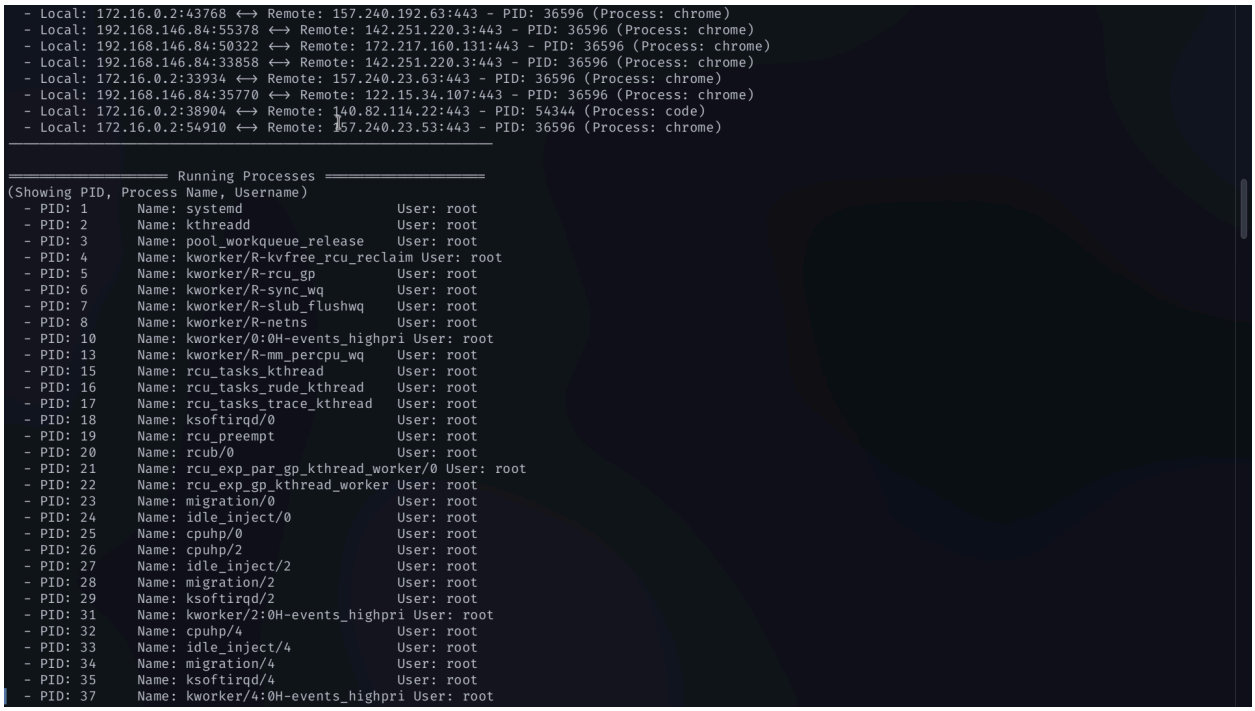
```
    except psutil.AccessDenied:
        print("  [Error] Insufficient permissions to retrieve full network info. Try running with sudo/administrator.")
    except Exception as e:
        print(f"  [Error] An unexpected error occurred gathering network info: {e}")

    # Print collected ports and connections
    if listening_ports:
        for port_info in listening_ports:
            print(port_info)
    else:
        print("  No listening ports found (or requires higher privileges).")

    print("\nActive Established TCP Connections:")
    if active_connections:
        for conn_info in active_connections:
            print(conn_info)
    else:
        print("  No active TCP connections found.")

    print("-" * 60)


def get_running_processes():
    """Gathers and prints information about running processes."""
    print("\n" + "="*20 + " Running Processes " + "="*20)
    print("(Showing PID, Process Name, Username)")
    process_count = 0
    try:
        # Iterate over all running processes fetching specified attributes
        for proc in psutil.process_iter(['pid', 'name', 'username']):
            try:
                # Access process info using .info dictionary
                pid = proc.info['pid']
                name = proc.info.get('name', 'N/A') # Use .get for robustness
                username = proc.info.get('username', 'N/A')
                print(f"  - PID: {pid:<6} Name: {name:<25} User: {username}")
                process_count += 1
            except (psutil.NoSuchProcess, psutil.AccessDenied):
```

```
            # Handle cases where process ended or access is denied for specific details
            print(f"  - PID: {proc.info.get('pid', 'N/A'):<6} Name: [Access Denied/Gone]       User: [Access Denied/Gone]")
        except Exception as e:
            print(f"  - PID: {proc.info.get('pid', 'N/A'):<6} Error accessing details: {e}")

    print(f"\nTotal processes listed: {process_count}")

  except psutil.AccessDenied:
      print("\n  [Error] Insufficient permissions to list all processes. Try running with sudo/administrator.")
  except Exception as e:
      print(f"\n  [Error] An unexpected error occurred while listing processes: {e}")

  print("-" * 60)

if __name__ == "__main__":
  print("Starting Local Node Security Profile Scan...")
  print("NOTE: For complete information (especially process names for network sockets), "
        "this script may need to be run with root/administrator privileges.")
  print("-" * 60)

  get_system_info()
  get_network_info()
  get_running_processes()

  print("\nScan Complete.")
```

Output (Screenshot)



```
┌──(EDITH in ~/S4 Lab/SLSL via  v3.13.2 took 0s
└─$ python3 twin.py
Starting Local Node Security Profile Scan...
NOTE: For complete information (especially process names for network sockets), this script may need to be run with root/administrator privileges.
------------------------------------------------------------

================== System Information ==================
Timestamp: 2025-04-04 09:37:11.804200
Hostname: EDITH
OS: Linux 6.13.8-zen1-1-zen
Architecture: x86_64
------------------------------------------------------------

================== Network Information ==================
Listening Ports (TCP/UDP):
  - 127.0.0.1:631 (TCP) - PID: N/A (Process: N/A)
  - 0.0.0.0:139 (TCP) - PID: N/A (Process: N/A)
  - :::1716 (TCP) - PID: 2209 (Process: kdeconnectd)
  - 127.0.0.1:35105 (TCP) - PID: N/A (Process: N/A)
  - ::1:631 (TCP) - PID: N/A (Process: N/A)
  - 0.0.0.0:445 (TCP) - PID: N/A (Process: N/A)
  - 127.0.0.54:53 (TCP) - PID: N/A (Process: N/A)
  - :::445 (TCP) - PID: N/A (Process: N/A)
  - 127.0.0.53:53 (TCP) - PID: N/A (Process: N/A)
  - :::5355 (TCP) - PID: N/A (Process: N/A)
  - 127.0.0.1:36607 (TCP) - PID: 54344 (Process: code)
  - :::6566 (TCP) - PID: N/A (Process: N/A)
  - :::139 (TCP) - PID: N/A (Process: N/A)
  - 0.0.0.0:5355 (TCP) - PID: N/A (Process: N/A)

Active Established TCP Connections:
  - Local: 192.168.146.84:50248 ⟷ Remote: 13.107.246.58:443 - PID: 53646 (Process: code)
  - Local: 192.168.146.84:35970 ⟷ Remote: 13.69.239.79:443 - PID: 53646 (Process: code)
  - Local: 192.168.146.84:50636 ⟷ Remote: 142.250.76.170:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:51274 ⟷ Remote: 162.159.137.105:443 - PID: N/A (Process: N/A)
  - Local: 192.168.146.84:52058 ⟷ Remote: 13.67.9.5:443 - PID: 53646 (Process: code)
  - Local: 192.168.146.84:50820 ⟷ Remote: 142.250.183.206:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:59218 ⟷ Remote: 157.240.23.63:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:60352 ⟷ Remote: 157.240.23.19:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:55366 ⟷ Remote: 142.251.220.3:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:35848 ⟷ Remote: 142.251.42.78:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:50658 ⟷ Remote: 142.250.76.170:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:33996 ⟷ Remote: 74.125.130.188:5228 - PID: 36596 (Process: chrome)
```



```
  - Local: 172.16.0.2:43768 ⟷ Remote: 157.240.192.63:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:55378 ⟷ Remote: 142.251.220.3:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:50322 ⟷ Remote: 172.217.160.131:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:33858 ⟷ Remote: 142.251.220.3:443 - PID: 36596 (Process: chrome)
  - Local: 172.16.0.2:33934 ⟷ Remote: 157.240.23.63:443 - PID: 36596 (Process: chrome)
  - Local: 192.168.146.84:35770 ⟷ Remote: 122.15.34.107:443 - PID: 36596 (Process: chrome)
  - Local: 172.16.0.2:38904 ⟷ Remote: 140.82.114.22:443 - PID: 54344 (Process: code)
  - Local: 172.16.0.2:54910 ⟷ Remote: 157.240.23.53:443 - PID: 36596 (Process: chrome)
------------------------------------------------------------

================== Running Processes ==================
(Showing PID, Process Name, Username)
  - PID: 1      Name: systemd                  User: root
  - PID: 2      Name: kthreadd                 User: root
  - PID: 3      Name: pool_workqueue_release   User: root
  - PID: 4      Name: kworker/R-kvfree_rcu_reclaim User: root
  - PID: 5      Name: kworker/R-rcu_gp         User: root
  - PID: 6      Name: kworker/R-sync_wq        User: root
  - PID: 7      Name: kworker/R-slub_flushwq   User: root
  - PID: 8      Name: kworker/R-netns          User: root
  - PID: 10     Name: kworker/0:0H-events_highpri User: root
  - PID: 13     Name: kworker/R-mm_percpu_wq   User: root
  - PID: 15     Name: rcu_tasks_kthread        User: root
  - PID: 16     Name: rcu_tasks_rude_kthread   User: root
  - PID: 17     Name: rcu_tasks_trace_kthread  User: root
  - PID: 18     Name: ksoftirqd/0              User: root
  - PID: 19     Name: rcu_preempt              User: root
  - PID: 20     Name: rcub/0                   User: root
  - PID: 21     Name: rcu_exp_par_gp_kthread_worker/0 User: root
  - PID: 22     Name: rcu_exp_gp_kthread_worker User: root
  - PID: 23     Name: migration/0              User: root
  - PID: 24     Name: idle_inject/0            User: root
  - PID: 25     Name: cpuhp/0                  User: root
  - PID: 26     Name: cpuhp/2                  User: root
  - PID: 27     Name: idle_inject/2            User: root
  - PID: 28     Name: migration/2              User: root
  - PID: 29     Name: ksoftirqd/2              User: root
  - PID: 31     Name: kworker/2:0H-events_highpri User: root
  - PID: 32     Name: cpuhp/4                  User: root
  - PID: 33     Name: idle_inject/4            User: root
  - PID: 34     Name: migration/4              User: root
  - PID: 35     Name: ksoftirqd/4              User: root
  - PID: 37     Name: kworker/4:0H-events_highpri User: root
```

```
  - PID: 38     Name: cpuhp/5                  User: root
  - PID: 39     Name: idle_inject/5            User: root
  - PID: 40     Name: migration/5              User: root
  - PID: 41     Name: ksoftirqd/5              User: root
  - PID: 43     Name: kworker/5:0H-events_highpri User: root
  - PID: 44     Name: cpuhp/6                  User: root
  - PID: 45     Name: idle_inject/6            User: root
  - PID: 46     Name: migration/6              User: root
  - PID: 47     Name: ksoftirqd/6              User: root
  - PID: 49     Name: kworker/6:0H-events_highpri User: root
  - PID: 50     Name: cpuhp/7                  User: root
  - PID: 51     Name: idle_inject/7            User: root
  - PID: 52     Name: migration/7              User: root
  - PID: 53     Name: ksoftirqd/7              User: root
  - PID: 55     Name: kworker/7:0H-events_highpri User: root
  - PID: 56     Name: cpuhp/8                  User: root
  - PID: 57     Name: idle_inject/8            User: root
  - PID: 58     Name: migration/8              User: root
  - PID: 59     Name: ksoftirqd/8              User: root
  - PID: 61     Name: kworker/8:0H-events_highpri User: root
  - PID: 62     Name: cpuhp/9                  User: root
  - PID: 63     Name: idle_inject/9            User: root
  - PID: 64     Name: migration/9              User: root
  - PID: 65     Name: ksoftirqd/9              User: root
  - PID: 67     Name: kworker/9:0H-events_highpri User: root
  - PID: 68     Name: cpuhp/10                 User: root
  - PID: 69     Name: idle_inject/10           User: root
  - PID: 70     Name: migration/10             User: root
  - PID: 71     Name: ksoftirqd/10             User: root
  - PID: 74     Name: cpuhp/11                 User: root
  - PID: 75     Name: idle_inject/11           User: root
  - PID: 76     Name: migration/11             User: root
  - PID: 77     Name: ksoftirqd/11             User: root
  - PID: 79     Name: kworker/11:0H-events_highpri User: root
  - PID: 80     Name: cpuhp/1                  User: root
  - PID: 81     Name: idle_inject/1            User: root
  - PID: 82     Name: migration/1              User: root
  - PID: 83     Name: ksoftirqd/1              User: root
  - PID: 85     Name: kworker/1:0H-events_highpri User: root
  - PID: 86     Name: cpuhp/3                  User: root
  - PID: 87     Name: idle_inject/3            User: root
  - PID: 88     Name: migration/3              User: root
  - PID: 89     Name: ksoftirqd/3              User: root
```

```
  - PID: 53850  Name: kworker/7:0-cgroup_destroy User: root
  - PID: 54086  Name: kworker/u48:10-btrfs-endio-meta User: root
  - PID: 54087  Name: kworker/u48:12-btrfs-delalloc User: root
  - PID: 54343  Name: code                     User: ac
  - PID: 54344  Name: code                     User: ac
  - PID: 54659  Name: code                     User: ac
  - PID: 54691  Name: kworker/u49:3-rtw_tx_wq   User: root
  - PID: 54802  Name: pet                      User: ac
  - PID: 54871  Name: code                     User: ac
  - PID: 54901  Name: bash                     User: ac
  - PID: 55046  Name: code                     User: ac
  - PID: 55057  Name: bash                     User: ac
  - PID: 55216  Name: bash                     User: ac
  - PID: 55381  Name: cloudcode_cli            User: ac
  - PID: 55435  Name: kworker/u48:13+i915      User: root
  - PID: 55436  Name: kworker/u48:14-btrfs-endio-write User: root
  - PID: 55437  Name: kworker/u48:15-btrfs-delalloc User: root
  - PID: 55438  Name: kworker/u48:16-events_unbound User: root
  - PID: 55439  Name: kworker/u48:17-btrfs-endio-write User: root
  - PID: 55440  Name: kworker/u48:18-btrfs-endio-write User: root
  - PID: 55441  Name: kworker/u48:19-btrfs-compressed-write User: root
  - PID: 56875  Name: kworker/8:2              User: root
  - PID: 56876  Name: kworker/8:3-events       User: root
  - PID: 56921  Name: bash                     User: ac
  - PID: 57147  Name: konsole                  User: ac
  - PID: 57162  Name: fish                     User: ac
  - PID: 57356  Name: kworker/9:0-cgroup_destroy User: root
  - PID: 57724  Name: kworker/6:0              User: root
  - PID: 57842  Name: systemd-userwork:        User: root
  - PID: 57962  Name: systemd-userwork:        User: root
  - PID: 57963  Name: systemd-userwork:        User: root
  - PID: 57974  Name: chrome                   User: ac
  - PID: 57984  Name: chrome                   User: ac
  - PID: 57994  Name: chrome                   User: ac
  - PID: 58026  Name: python3                  User: ac

Total processes listed: 351
──────────────────────────────────────────────────────────

Scan Complete.

┌─ac@EDITH in ~/S4 Lab/SLSL via ♣ v3.13.2 took 0s
└─λ
```

Description of Output:

The script execution produces text output divided into distinct sections:

1. **System Information:** Displays the timestamp of the scan, the machine's hostname, the operating system name and version, and the system architecture (e.g., x86_64).
2. **Network Information:** This section is further divided:
    - **Listening Ports:** Lists each TCP and UDP port the system is listening on. For each port, it shows the local IP address and port number, the protocol (TCP/UDP), the Process ID (PID) of the application listening (if permissions allow, otherwise 'N/A'), and the name of that process (if permissions allow, otherwise 'N/A' or an error indicator).
    - **Active Established TCP Connections:** Lists ongoing TCP communications. For each connection, it shows the local IP/port, the remote IP/port it's connected to, the PID of the local process involved (if available), and the name of that process (if available).
3. **Running Processes:** Provides a list of processes currently active on the system. Each entry typically shows the Process ID (PID), the executable name of the process, and the username under which the process is running. A total count of listed processes is often shown at the end of this section.

```
Starting Local Node Security Profile Scan...
NOTE: For complete information (especially process names for network sockets), this script may need to be run with root/administrator privileges.
------------------------------------------------------------
==================== System Information ====================
Timestamp: 2025-04-04 09:27:29.123456
Hostname: student-dev-vm
OS: Linux 5.19.0-generic
Architecture: x86_64
------------------------------------------------------------

==================== Network Information ====================
Listening Ports (TCP/UDP):
  - 127.0.0.53:53 (UDP) - PID: 678 (Process: systemd-resolve)
  - 0.0.0.0:22 (TCP) - PID: 901 (Process: sshd)
```

```
  - 127.0.0.1:631 (TCP) - PID: 789 (Process: cupsd)
  - [::]:22 (TCP) - PID: 901 (Process: sshd)
  # ... more ports ...

Active Established TCP Connections:
  - Local: 192.168.1.15:22 <-> Remote: 192.168.1.10:51234 - PID: 12345 (Process: sshd)
  - Local: 192.168.1.15:34567 <-> Remote: 142.250.196.142:443 - PID: 11223 (Process:
firefox)
  # ... more connections ...
------------------------------------------------------------

==================== Running Processes ====================
(Showing PID, Process Name, Username)
  - PID: 1      Name: systemd                User: root
  - PID: 2      Name: kthreadd               User: root
  - PID: 678    Name: systemd-resolve        User: systemd-resolve
  - PID: 789    Name: cupsd                  User: root
  - PID: 901    Name: sshd                   User: root
  - PID: 1050   Name: gnome-session-b        User: studentuser
  - PID: 11223  Name: firefox                User: studentuser
  - PID: 12345  Name: sshd                   User: studentuser
  # ... many more processes ...

Total processes listed: 215
------------------------------------------------------------

Scan Complete.

_____
```

Conclusion

This microproject successfully demonstrated the creation of a Python script capable of profiling a local machine's security-relevant state. By utilizing the platform and psutil libraries, the script effectively gathers system identification, network listeners, active connections, and running processes, presenting them in a structured format.

This script serves as a practical example of using scripting for security monitoring, a core skill in the field. The collected data forms a baseline snapshot of a node's status, directly applicable to the concept of a Network Twin. In a Network Twin scenario, such data could be used to instantiate or validate the state of a virtual node, compare current state against a known-good baseline to detect anomalies (potential security incidents), or provide context for simulated security events.

While effective for its scope, the script has limitations. It captures only a single point-in-time snapshot of one machine and doesn't delve into deeper details like specific service versions, detailed firewall configurations, or file integrity. Future enhancements could involve scheduling the script for periodic execution to track changes over time, integrating it with configuration management databases, adding vulnerability checking based on software found, or sending the output to a centralized logging or SIEM (Security Information and Event Management) system for broader network visibility, thereby moving closer to a functional component of a dynamic network twin monitoring system.
