**Reverse Shell Simulation Report**

**Project:** Basic Stealth Reverse Shell Simulation  
**Author:** Aryan Jain  
**Date:** 2025-07-23

1. **Introduction**

This project demonstrates a Python-based reverse shell with basic stealth features, designed for educational purposes. It simulates a stealthy Remote Code Execution (RCE) attack, where a payload is executed on a Windows target. The resulting network traffic is captured and analyzed using Wireshark to understand reverse shell behavior and detection techniques used by SOC analysts.

**2. Objective**

* Create a basic reverse shell script with stealth features.
* Convert the script into a Windows executable using PyInstaller.
* Simulate an attacker-client connection over a chosen IP and port.
* Capture and analyze network traffic during the reverse shell session using Wireshark.
* Identify indicators of compromise (IOC) and network behaviors relevant for SOC analysts.

**3. Environment Setup**

* **Attacker Machine:** Kali Linux
* **Target Machine:** Windows 11
* **SOC Machine:** Parrot OS
* **Tools Used:** Python 3.11, PyInstaller, Wireshark, Netcat (for listener)
* **Communication:** TCP reverse shell over port 8088
* **Payload Delivery:** Manual execution of the compiled .exe payload on the target machine

**4. Reverse Shell Implementation**

* The reverse shell script establishes a TCP connection from the victim to the attacker.
* Commands received from the attacker are executed using Python's subprocess module.
* Output is sent back over the established connection.
* Stealth elements include silent execution (--noconsole), minimal error output.

**5. Findings from Network Capture Analysis**

**Detection Method**  
To detect the reverse shell activity, **Wireshark** was used to monitor network traffic. By applying a **destination IP filter** for the victim machine and following the **TCP stream**, I was able to reconstruct the full communication between attacker and victim.

**Observed Behaviors:**

* **Connection Patterns**:
* Outbound TCP connections from the victim to a suspicious IP and port (e.g., 192.168.x.x:8088)
* No DNS resolution, indicating a direct IP connection
* **Traffic Characteristics**:
* Cleartext C2 traffic—commands like whoami, dir, and output data
* Lack of encryption or tunneling made it observable on network level
* **Anomalies Noted**:
* Use of **non-standard port** (e.g., 8088 instead of 80/443)
* **Short bursts** of TCP data packets indicating command/response
* **No valid application-layer protocol signature** (i.e., no HTTP/HTTPS)
* **Indicators of Compromise (IOCs)**:
* Persistent TCP session from internal host to unknown IP
* Execution of remote commands visible in packet payload
* Activity occurred outside normal business hours

**6. SOC Analyst Perspective**

* Monitor for outbound connections on uncommon ports (e.g., TCP 8088), especially if initiated by internal hosts.
* Analyze packet payloads using **TCP stream reassembly** in Wireshark to detect suspicious command-line activity.
* Correlate observed **command strings** (e.g., whoami, dir) with known Indicators of Compromise (IOCs).
* Watch for **unusual persistence** in TCP sessions without typical application-layer protocols (HTTP/HTTPS).
* Use **destination-based filtering** in Wireshark to isolate suspicious traffic from a specific internal machine.
* Investigate outbound connections initiated immediately after new processes are spawned (like python.exe).

**7. Limitations and Recommendations**

* The reverse shell payload is basic and sends commands in clear text; encrypting traffic could improve stealth.
* Payload delivery method in real environments often requires phishing or exploitation vectors.
* Integration with automated detection systems and endpoint monitoring is recommended for comprehensive defense.

**8. Conclusion**

This simulation successfully demonstrates the lifecycle of a reverse shell attack and its detection opportunities from a SOC analyst’s viewpoint. It highlights the importance of network traffic monitoring and endpoint security in identifying and mitigating such threats.