Project Documentation

Project name: CQr (Secure), Developers: Mohamed Ali, Rami Qaraqra

Tool Overview & Goal Fulfillment:

CQr is a real-time file server protection system designed to secure Windows environments. It continuously monitors designated directories for file system events. When such events occur, CQr instantly scans it using the local ClamAV engine. And if it finds it infected it efficently naturalizes the threat, by an automated response: locks the file and move it to a restricted quarantine folder

Does it answer the goal? Yes. The primary goal was to prevent the malware on a server, CQr achives this by:

1. Detection: Catching viruses immeditely upon creation
2. Isolation: Removing the file from the public protected directory before other users can “touch” it
3. Notification: Replaces the infected file with a safe text note alerting the users

Challenges we faced:

Quarantine Permission Lockout: When implementing the secure quarantine folder using Windows ACLs (`icacls`), we initially set the permissions too strictly (`/inheritance:r`), causing the script itself (running as a user) to lose write access. We resolved this by modifying the ACL command to explicitly grant the `SYSTEM` and `Administrators` group full control while removing standard `Users`.

Strengths and Weaknesses:

Strengths:

- **Zero-Latency Monitoring:** Uses kernel-level file system events (`watchdog`) rather than slow polling intervals.
- **Secure Quarantine:** The quarantine folder is not just a hidden folder; it is protected at the OS level (ACLs), preventing unauthorized users from accessing or restoring the malware.
- **User Experience:** Features a polished CLI (`CQr.py`) with color-coded logs and an automated setup guide for ClamAV isntegration.

Weaknesses:

- **Signature Reliance:** Since it relies on ClamAV, it detects known threats (Signature-based) but may miss brand-new "Zero-Day" attacks that lack signatures.

Description of Tool Operation (Workflow)

The system follows a linear pipeline architecture:

1. **Configuration:** The user adds paths to `Paths.txt` via the CLI command `python CQr.py add <path>`.
2. **Observation:** The `observer.py` module starts the watchdog observer, listening for `FILE_CREATED` or `FILE_MODIFIED` events in the target folders.
3. **Trigger:** When an event occurs, the file path is passed to `scanner.py`.
4. **Scanning:** The scanner communicates with the local ClamAV Daemon via TCP (Port 3310) to request a scan.
5. **Response:**
    - If **Clean**: The event is logged as safe.
    - If **Infected**: `utils.py` is triggered. It calculates a unique ID, moves the file to `C:\CQr_Quarantine`, changes the extension to `.infected`, and writes a warning note in the original location.