Project Report: Upload Secure Sample

Abeer Agarwal

November 29, 2025

Abstract

The "Upload Secure Sample" project is a C-based command-line utility designed to enhance cybersecurity workflows. It interfaces directly with the VirusTotal API to scan files for malware, providing a lightweight alternative to browser-based scanning. The system automates file uploading, analysis polling, and result retrieval, ensuring that users can verify file integrity efficiently without relying on heavy graphical interfaces.

Contents

1 Problem Definition

1.1 Objective

The primary objective was to develop a functional file scanner that interacts with a third-party REST API using low-level C programming. The tool aims to detect malicious content in files by automating the "Upload" and "Scan" lifecycle, providing immediate feedback to the user via a terminal interface.

1.2 Scope

The project operates under the following constraints and specifications:

- **Language:** Implemented strictly in C.
- **Dependencies:** Uses `libcurl` for HTTP requests and `cjson` for parsing API responses.
- **Environment:** Supports `.env` configuration for secure API key management.
- **Platform:** Cross-compatible with Windows (MSYS2) and Linux environments.

2 System Design

2.1 Architecture Overview

The system is modularized to separate the User Interface (UI) from the API logic:

- **Core:** `src/main.c` handles user input, file validation, and the main execution loop.
- **Utilities:** `src/virustotal_utils.c` (and its header) encapsulates the specific API calls to VirusTotal.
- **Configuration:** A `.env` loader ensures sensitive API keys are not hardcoded.

2.2 Flowcharts & Algorithms

The core logic follows a sequential workflow to ensure data integrity before network operations begin.

2.2.1 Algorithm 1: Main Scanning Workflow

```
START
    Load API Key from Environment
    Prompt User for Filename
    IF Input is Empty THEN
        Set Filename = "sample_input.txt"
    ENDIF

    IF File Does Not Exist THEN
        Print Error
        EXIT
```

```
    ENDIF

    AnalysisID = UploadToVirusTotal(Filename)
    IF AnalysisID is NULL THEN
        Print Upload Error
        EXIT
    ENDIF

    Result = GetAnalysis(AnalysisID)
    Display Result
    Free Memory
END
```

Listing 1: Pseudocode for Scan Logic

3 Implementation Details

3.1 Data Structures Libraries

Instead of custom structs for data holding, the project relies on robust external libraries to handle complex data types:

- **cJSON:** Used to parse the complex nested JSON objects returned by the VirusTotal API.
- **libcurl:** Manages the internal structures required for HTTP POST/GET requests and buffer management.

3.2 Core Logic Snippets

3.2.1 User Input and Validation

The following snippet demonstrates how the system handles user input and validates file existence before attempting network operations.

```c
int main(void) {
    char filename[MAX_FILENAME_LEN];
    char filepath[MAX_PATH_LEN];
    char *vt_file_id = NULL;

    // Prompt user for filename
    printf("Enter filename (or press Enter for sample_input.txt): ");
    if (fgets(filename, sizeof(filename), stdin) == NULL) {
        return 1;
    }

    // Default fallback logic
    if (len == 0) strcpy(filename, "sample_input.txt");

    // Validate file exists
    if (!file_exists(filepath)) {
        fprintf(stderr, "Error: File '%s' not found.\n", filename);
        return 1;
    }

```

```
    // Initiate Upload
    vt_file_id = upload_to_virustotal(filepath);
}
```

Listing 2: Main Execution Loop (src/main.c)

4 Testing & Results

4.1 Test Cases

We performed manual testing to ensure the CLI handles various edge cases gracefully.

ID	Scenario	Result
TC-01	Upload valid existing file	PASS (Analysis ID received)
TC-02	User inputs empty string	PASS (Defaults to sample_input.txt)
TC-03	Upload non-existent file	PASS (Error caught locally)
TC-04	API Key missing	PASS (Connection rejected gracefully)

4.2 Performance

- **Memory:** Efficient memory usage due to manual memory management in C.
- **Latency:** Dependent on VirusTotal API limits, but the polling mechanism ensures results are displayed as soon as they are available.

5 Conclusion & Future Work

5.1 Conclusion

The "Upload Secure Sample" project successfully meets its objective of providing a C-based interface for malware scanning. By integrating `libcurl` and `cJSON`, we demonstrated how low-level languages can effectively interact with modern RESTful web services.

5.2 Future Work

- **Hashing:** Calculate file hash (MD5/SHA256) locally to check if the file was already scanned, saving API quota.
- **Batch Processing:** Allow the user to input a directory to scan multiple files sequentially.
- **GUI:** Implement a simple graphical interface using GTK or Raylib.

6 References

1. VirusTotal API v3 Documentation. `https://developers.virustotal.com/`
2. CURL Library Documentation. `https://curl.se/libcurl/`

3. Dave Gamble. cJSON Library. https://github.com/DaveGamble/cJSON

4. Project Repository. https://github.com/YourUsername/Upload-Secure-Sample-with-C