Implementing Data Security in Starcraft 2 Analytic Tools

Author: Cody Strange

Professor: Sayeed Sajal

Institution: Utah Valley University

Course: CS-3100-001 2024 Spring

# Abstract

This project encompasses the development and refinement of a StarCraft II (SC2) data analyzer program, designed to analyze large sets of replay data from a database, to identify player strategies and statistics. Central to this project is not only the analyzer's capability to process and interpret extensive datasets efficiently but also its adherence to stringent security standards.

Initially, the program will exhibit critical vulnerabilities in two key security areas. Firstly, inadequate error handling, where the system fails to manage exceptions and errors, leading to potential information leakage and system instability. Such vulnerabilities could expose sensitive data or provide clues to attackers about the system's inner workings, thereby compromising data confidentiality and integrity. Secondly, the program will lack logging and monitoring capabilities, a shortfall that hinders the timely detection of unauthorized access attempts or abnormal activities.

The project's subsequent phases focus on addressing these vulnerabilities. By redesigning the error handling logic, the program will encapsulate (isolating from unauthorized access or exposure) sensitive information, prevent data leaks, and ensure continuous operation even under erroneous conditions. The addition of logging and monitoring will be implemented to ensure comprehensive tracking of user activities and system events, enabling the early detection of potential security breaches and operational anomalies.

This paper details the journey from identifying these critical security flaws to implementing and verifying the efficacy of the remedial measures. It serves as a case study in balancing functionality and security in software development, particularly in data-intensive applications like the SC2 data analyzer, where data integrity and system reliability are paramount.

# Research and Design

## Cryptographic algorithms

### Modern algorithms used in data analytics

* **Advanced Encryption Standard (AES):** A symmetric key encryption that is used by the government to protect classified information. Used to secure the data stored in database files.
* **Rivest-Shamir-Adleman (RSA):** A asymmetric key encryption that is used to secure data transmission for insecure networks. It is also used to securely exchange symmetric keys for symmetric key encryptions like AES.
* **Elliptic Cure Cryptograph (ECC):** A better version of RSA, same level of security with smaller key sizes. Though newer than RSA so not as widely used, especially in legacy technology.
* **Homomorphic Encryption:** An asymmetric encryption that allows computations to be performed on ciphertext so that the data does not have to be decrypted before it is analyzed. Is computationally expensive compared to other methods such as AES.

### Algorithm requirements

* **Security:** The algorithm must be impervious to brute force attacks, cryptanalysis, and side-channel attacks.

**Performance:** The algorithm can’t slow down the performance of the overall application, can’t affect user experience.

**Scalability:** The algorithm should be able to keep its performance levels as greater amounts of data are added to the database.

## Error handling standards

* **Explicit Error Handling:** Check for and handle potential errors without exception handling where possible
  + Invalid inputs and outputs
* **Exception Handling:** Handle errors that cannot be done without exception handling methods.
  + File accessing
  + Failed operations
* **Error Reporting:** Provide clear and concise descriptions of the errors.
  + Possible causes of error
  + Where the error was located
  + How to possibly fix the error
* **Error Recover:** Recover from errors with minimal lost or data corruption
  + Rolling back transactions
  + Halting program if data could be corrupted

## Logging and monitoring software

### Software

* **Python Logging Module:** Python comes with logging capabilities included in the standard library. It can be used to log different types of messages along with logging exceptions.
* **Prometheus:** Is an open-source data monitoring software, not sure how compatible it will be with my analysis program.

## Design principles

### Exception Handling

* **Consistency:** Exception Handling should be applied consistently throughout the codebase
* **Fail Fast:** Detect errors as soon as possible and as close to their source as possible
* **Use Specific Exception Types:** Use specific exceptions rather than general exceptions
* **Avoid Ignoring Exceptions:** Once you catch and exception always do something with it
* **Use Exceptions for Exceptional Situations:** Exceptions should only be used to detect exceptional conditions. If conditions can be detected through normal means such as if-else statements use those first

### Logging

* **Structured Logging:** Use logging formats such as JSON that are easily parsed
* **Log Levels Appropriately:** Log the severity of the message such as debug, info, warning, error, fatal
* **Sensitivity Information:** Make sure all sensitive information is properly encrypted
* **Clean Logs:** Clean out irrelevant logs on a regular basis

### Monitoring

* **Comprehensive Coverage:** Monitor a variety of indicators, such as performance and errors
* **Alerting Thresholds:** Set up alerting thresholds that are meaningful and actionable
* **Dashboards:** Use dashboards to allow for visualizations of metrics

# Implementation

## Cryptographic algorithms

### Raw algorithm

### Implemented into SC2 analysis program

## Testing

# Code Security Analysis

## Review

### Static analysis tools

### Manual reviews

## Vulnerabilities

## Report

# Remediation

## Remediation plan

## Fixes

## Testing

# Metric Reports

## Comprehensive reports

## Future recommendations