CSE 5382-001: Secure Programming
Spring 2019, © TJ, UTA 2019

Project
Part 1 Due: March 15, 2019, 23:59 UTA time
Part 2 Due: May 3, 2019, 23:59 UTA time

Overview

This project will give you the opportunity to apply everything you've learned throughout the semester in the development of an application.

You must demonstrate evidence that you are applying security principles throughout the entire system development lifecycle (SDLC) in an effort to build security in from the beginning and not bolt it on at the end. The focus of this project is not on WHAT you produce but HOW you produce it.

This is a team assignment. Teams will be randomly assigned and will be 5-6 students each. You are strongly encouraged to pick a team lead to coordinate all activities and delegate the workload among your team members. All team members are expected to become familiar with all of the artifacts described below by participating in reviews, design discussions, etc.

Project Management

As you execute the project, to help give you real world experience as well as make your effort more efficient, you should incorporate the following practices:

- Attempt to follow a secure development lifecycle (SDL) in the context of a software development lifecycle.
- Collaborate using both face-to-face as well as online tools (e.g. Discord, Skype, Blackboard, etc.). A face-to-face meeting at least once a week or (worst case) once every two weeks would be desirable.
- Utilize a source code versioning system (e.g. Git) to store artifacts under development and to manage releases (using Git tagging or other baselining mechanisms appropriate to chosen tool). Along with this, follow a branching/merging workflow (such as Gitflow Workflow as documented at https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow). You may choose to use Github or Bitbucket to manage your project and permissions.
- Decide what software lifecycle methodology you will use (e.g. waterfall, agile, spiral, etc.). You might choose to approach the artifacts in an iterative manner. However, consider that the due date for part 1 implies that all the artifacts required for part 1 must be in at least a final draft state (consider the part 1 due date like a Critical Design Review milestone on a real project).
- Consider using a lifecycle tracking tool (excel spreadsheet, Jira, etc.) to help track tasks to be worked, version release dates, issues being fixed for each release, etc.

Being able to manage the project execution using practices such as those described above will have an affect on whether you succeed or fail.

Part 1 (Due March 15, 2019)

For the first part of the assignment, you should put together a plan for how you will execute the project and create an early design, architecture, and requirements. Deliverables should include the following:

- **(75 Points)** Artifacts Submission (Group Assessment)
  - **(10 points)** System Overview with Design Details, Rationale, Ground Rules and Assumptions, and Concept of Operations (ConOps) for how the system will be used.
  - **(10 points)** Security Architecture – This should include diagrams of the system's components, data flows, etc. with the inclusion of security controls allocated to components of the architecture.
  - **(10 points)** Threat Modeling and Attack Surface Analysis – Identify the attack surface for each component of the system and show what the threats are (potentially using STRIDE per element) and how you will mitigate them. Identify, communicate, and understand threats and mitigations within the context of protecting something of value. Understand the potential motivations for why an attacker might target the system and which components would be of high value. Review the slides covered in class for approaches to these. Also look at CAPEC for some potential attack patterns (https://capec.mitre.org/).
  - **(5 points)** Security Requirements – This should include "shall" statements for what security features/attributes the system must implement. Also include traceability to how you will verify the requirements. These should reflect the "voice of the customer".
  - **(5 points)** Coding Standards:
    - For each language you use, develop rules that your team must follow (you are expected to demonstrate that you are following these rules based on the code you implement in Part 2).
    - Cite any online sources used to contribute to it and document any tailoring.
    - The following are good starting points for various coding standards (not an exhaustive list):
      - https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards
      - http://google.github.io/styleguide/
  - **(5 points)** Supply Chain Risk Management – Show how third-party software dependencies you use and how they are vetted. Should include assessment of known vulnerabilities in these dependencies and how you will mitigate them. You should be using one or more online vulnerability databases. Remember: Your dependencies may also have dependencies (and those may have dependencies…and so on). Also, if you are using third-party hosting services or cloud providers, document how you are addressing security concerns with those providers and how you are vetting them.
  - **(10 points)** Information/Functional/Behavioral Modeling – UML diagrams including Class Diagrams, Activity Diagrams, Sequence Diagrams, Use Cases, Abuse Cases, State Transition Diagrams.

    - **(5 points)** Testing Strategy – What testing methodologies and tools will you use. Should include plans for dynamic application security testing (DAST) tools (a.k.a. dynamic code analysis tools).
    - **(5 points)** Database Table Design (if applicable) – Possibly use Entity/Relationship diagram. Talk about any access controls placed on the various database tables/views (how do you control access to the data based on user roles).
    - **(10 points)** UI mockups – Show wireframes and prototypes of the user interface (GUI or web pages). You might decide to storyboard certain use cases.
- **(25 points)** Peer Evaluation (the form is attached) (Individual Assessments).
    - **Note**: if any member gets average of 5 out of 25, he/she will get **zero in this project phase out of 100**. It indicates that this member did not sufficiently contribute/participate in the group project.
    - These forms will be turned in by all team members individually through a separate submission on Blackboard.

I expect that the team as a whole will participate in the review of the above artifacts and provide feedback to the responsible authors. Just like in the real world, keep a findings journal of the review comments and turn those in as well.

By March 1, 2019, the team lead should send the professor an e-mail with status (estimated percent complete) for all of the above artifacts. This will allow an assessment of whether or not the due date is at risk.

It is expected that if you finish part 1 prior to the due date, you are encouraged to begin part 2.

Part 2 (Due May 3, 2019)

For the second part of the assignment, you focus on finalizing the design and architecture, code and test the system, and assess the effectiveness of your security control implementations. It is important to collect and provide evidence artifacts during these phases that will provide assurance of the security and pedigree of the system. The expected submissions are as follows:

- **(75 Points)** Artifacts Submission (Group Assessment)
    - **(30 Points)** Project Report containing the following:
        - **(20 points)** Show vignettes of the system for the following practices (including configuration file excerpts, source code excerpts, etc. as necessary):
            - Input/output validation
            - Use of centralized logging and logging of various application events
            - List all events that your application specifically audits
            - Exception/error handling
            - Sample log file entries
            - Session management
            - Authentication and authorization
            - Cryptography
            - Privilege management
            - Mitigations for buffer overflows

        - Mitigations for Cross Site Request Forgeries (CSRF)
        - Mitigations for injection attacks
        - Handling of sensitive information
        - And any other types of mitigations/techniques that were used.

      If any of the above weren't used, state so.

    - **(10 points)** Show execution of scenarios using the application (follow use cases documented in design) and include actual screen shots.
- **(5 points)** Software Version Description (SVD) Document (also known as a Version Description Document or VDD) containing the following:
    - Include the version identifier for the final release of your application.
    - Include list of issues resolved since prior release (if any).
    - Include list of known issues that weren't resolved (if any).
    - Include list of all dependencies and their version numbers that are required for the application to operate (should match versions used during testing as part of your configuration baseline). This should include all of the following:
        - Operating System
        - APIs/Frameworks/Libraries
        - DBMS
        - Application Server
        - Web Server
        - Any other middleware required.
    - A sample template for this document is attached to the assignment.
- **(5 points)** All deliverables from Part 1 with updates as appropriate based on changes to your architecture, design, plans, etc.
- **(5 points)** Zip file containing complete set of source code, configuration files, graphical assets (icons and images), etc. for your application.
- **(20 points)** Static Code Analysis Evidence including the following:
    - Manual Code Reviews
        - Findings log from manual code reviews.
            - For each entry, include the following:
                - File name.
                - Line number(s).
                - Defect description.
                - Disposition (not a problem, fixed, not fixed/deferred)
                    - If deferred, expectation is a ticket is generated for the finding (ticket number included in entry) and listed in known issues in SVD.
                    - If fixed, describe how corrected.
                - Fix verified by whom and when (other reviewer name and date).
        - In a report, take a sampling of defects and show code before and after fixing.

    - Automated
        - Indicate which tools are used (both static code analysis tools and vulnerability scanning tools).
        - Include information about what rulesets were used for the SCA tools.
        - Include findings report produced by tools.
        - Indicate how findings are dispositioned (not a problem, fixed, not fixed/deferred).
- **(5 points)** Project Management data including the following (these can be reports generated from project management tools, Excel spreadsheets, or screenshots as a last resort):
    - Include metrics/graphs/dashboards used to track project execution.
    - Include list of tickets generated and their disposition.
    - Include backlog of tasks and assignments.
    - Include project schedules.
    - Include CCB minutes.
- **(20 points)** Testing Report
    - Include security testing results.
        - There should be evidence of testing of various exploits discussed in class which should all fail due to appropriate coding techniques (e.g. input validation, use of safe functions, etc.).
    - Include penetration testing results.
    - Include unit testing results.
    - Include integration testing results.
- **(10 points)** Security Assessment/Compliance:
    - STIG checklists.
    - Operating system, web server, application server security hardening configuration snippets (show any changes you make from the defaults).
    - Vulnerability scan reports from tools such as Nessus, OpenVAS, etc. against the system (usually targets third-party dependencies with known vulnerabilities) and dispositioning of those findings.
- **(25 points)** Peer Evaluation (the form is attached) (Individual Assessments).
    - **Note**: if any member gets average of 5 out of 25, he/she will get **zero in this project phase out of 100**. It indicates that this member did not sufficiently contribute/participate in the group project.
    - These forms will be turned in by all team members individually through a separate submission on Blackboard.

Proposed Systems

Pick one of the following systems to implement:

- Money exchange web site (e.g. PayPal) - Allows users to send money to each other as well as pay merchants for goods or services (either one time or subscription).
- Battleship Game
- Blackjack Game
- Casino Slot Machine Game

- Course gradebook (captures students' grades in a class).

Or suggest your own (but describe in your report what you are implementing and its capabilities). The system should have data of value to an adversary that he/she would either want to obtain or modify.