IoT Security Validation

Design Document

sdmay21-41
Megan Ryan
Kristin Rozier
Team Members/Roles
Team Email
Team Website

Revised: 9-21-2020/0.01

Executive Summary

Development Standards & Practices Used

Software: SV-COMP, Ubuntu 2004, ABET

Summary of Requirements

Ubuntu 2004 VM machine/x86_64-linux, Ubuntu 18.04, a memory limit of 15 GB (14.6 GiB) of RAM, a runtime limit of 15 min of CPU time, and a limit to 8 processing units of a CPU.

Applicable Courses from Iowa State University Curriculum

SE/CPrE 185, COMS227, COMS228, COMS311, SE339, SE329

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 Acknowledgement

If a client, an organization, or an individual has contributed or will contribute significant assistance in the form of technical advice, equipment, financial aid, etc, an acknowledgement of this contribution shall be included in a separate section of the project plan.

1.2 Problem and Project Statement

The Internet of Things (IoT) is becoming more and more a part of people's everyday lives. Devices such as locks, cameras, and smart-speakers are just a very small view of all the ways our lives are going online. With all of these devices having important roles, being located in private places, and gathering loads of information, the security of them is much more prevalent as it would be problematic if it got into the wrong hands.

There are already some ways that the security of the code behind these IoT devices is being tested. However, there are a lot of security properties that aren't being as thoroughly checked. One of the ways is through a program called SV-COMP. SV-COMP helps compare different software verification tools to help find which tools will suitably satisfy your needs. Our project is developing another version of SV-COMP to focus on IoT device code and test different IoT libraries.

The final goal of this project is to have a working version of SV-COMP that is able to test many IoT libraries, and from that be able to confidently verify the security of different IoT libraries. In doing that, we will have Iot code benchmarks for others to use to secure code with their own validation tasks. A further goal, if time and resources persist, is to combine the secure code we find into an IoT library that is trustworthy and reliable.

1.3 Operational Environment

Ubuntu 2004 VM machine/x86_64-linux, Ubuntu 18.04, a memory limit of 15 GB (14.6 GiB) of RAM, a runtime limit of 15 min of CPU time, and a limit to 8 processing units of a CPU.

1.4 Requirements

The software can be downloaded, it can be replicated and evaluated.

The software can be archived in a ZIP file, with a directory within.

The software should not require any special software on the competition machines; all necessary libraries and external tools should be contained in the archive.

The software can report its version.

Remains free of unnecessary data, with only the core code and descriptions within the code, free of things such as test files.

1.5 Intended Users and Uses

The intended users of this software will be developers and academia of IoT.

End users will be able to use our developed benchmarks to test their own IoT code through validation tasks in the SV-COMP environment.

1.6 Assumptions and Limitations

Assumptions:

- Only using Java and C IoT code
- BenchExec will run smoothly on a different Ubuntu version
- Verification tasks will run correctly on BenchExec

Limitations:

- The monetary cost to produce the end result shall be zero
- VMs are limited to one core on 8 GB of RAM running Ubuntu
- We are limited to the end of spring semester to finish the project

1.7 Expected End Product and Deliverables

List of several well rounded IoT libraries

> These IoT libraries will all come from open source code on the internet. A well rounded library will be one that does not have a profusion of dependencies in it. Some of these libraries may be slightly modified to make them into well rounded libraries. This list will then be run through IoT verification tasks to see how secure they are.

A running instance of SV-COMP

> This instance of SV-COMP will be set up on VMs provided by the university. The instance of SV-COMP will be able to run the IoT verification tasks on the open source IoT libraries.

Verification tasks to run against C and Java IoT code

> This set of verification tasks will be focused around the security aspect of IoT devices. Ideally, this will include both the C and Java languages. Part of these may come from the SV-COMP benchmarks repository, whereas others will have to be written on our own. These tasks will cover a slew of security issues with IoT devices, and will not focus on any one particular aspect/weakness.

C and Java library consisting of code that has been verified using IoT verification tasks

This deliverable is a bonus one that we would like to do if time allows it. After testing all of the open source IoT code with the verification tasks by using SV-COMP, the good code will then be compiled into a library. This will result in separate C and Java libraries of secure IoT code.

2 Project Plan

2.1 Task Decomposition

Below is a list of our planned tasks, and some of the steps required within each task to complete it. Some tasks require specific steps, whereas others require more open-ended research and data collection. Most tasks build upon the knowledge found in the first task of general research.

- General Research
  - Team introductions
  - Vulnerabilities
  - SV-COMP
  - IoT code
  - Code libraries versus frameworks
  - Code verification tools
- Identify Milestones
  - Discuss with advisor and client
  - Have a clear end-goal decided on
  - Develop a Gantt chart
  - Verify milestones with advisor and client
- Identify IoT Libraries for use - the team will be testing many libraries throughout the project
  - Create library benchmarks - based on data found in general research
  - Create IoT code benchmarks - based on data found in general research
  - Research available IoT libraries based on determined benchmarks
  - Choose a select amount of libraries for use
- Identify verification properties to test
  - Research which properties are tested often - knowledge of vulnerabilities from general research used
  - Choose properties not tested as often - partially based on IoT libraries decided upon
- Set up SV-COMP
  - Get access to an ISU virtual machine (each team member)
  - Decide on SV-COMP tools to use for Java - based on knowledge from general research
  - Set up SV-COMP on virtual machines - uses knowledge from general research
- Design Java Verification Tasks
  - Use example verification tasks for guidance
  - Use knowledge of decided upon SV-COMP tools
  - Use knowledge of SV-COMP
  - Create based upon decided security properties to test

- Design C Verification Tasks
    - Use knowledge from created Java verification tasks
- Run Verification Tasks
    - Use knowledge of SV-COMP
    - Plug in IoT libraries previously decided upon
    - Verify libraries based on previously chosen security properties
- Build SV-COMP
    - Use the virtual machines each member has access to
    - Compile and run previously created verification tasks in the already set up SV-COMP environment

2.2 Risks And Risk Management/Mitigation

For our project we have identified the following risks and risk mitigation plans:
Licensing Issues - a few IoT platforms require capital to use. Our mitigation strategy for this is to use open source IoT code and libraries.

Inadequate Design - a risk associated with a misunderstanding of the project's goals. Our mitigation strategy is to define our project problem and our statement and use those definitions to expand on our design of our project.

Team Dynamics - a risk with any project that has a team. Our mitigation strategy is to have weekly team meetings, address issues as the arise and be proactive.

Developing wrong software functions - this risk can come from miscommunication or misunderstanding of the project requirements. A mitigation strategy for this risk is to have code peer reviewed and have weekly team meetings to go over functions that are required for our task. Gold Plating - adding more features to a product that the client did not ask for. Our mitigation strategy for this is to keep within our project plan and use our weekly client meetings to stay within the scope.

Incompatible Libraries - we may run into a library that requires too much time to ‘round’ off. One way to mitigate this is to look for standard libraries in addition to checking libraries versus library definition.

Incompatible IoT Code - code that is heavily library dependent requires too much time to properly ‘round’ off to run as a validation task. Our mitigation strategy is to look for good IoT code commonly used in IoT and use that as a comparison tool with other IoT code to validate.

Time Constraints -as we run these verification runs our system will have to do model checking. This takes real time and since we have limited ram it may take a full day or more to run a verification run. Our mitigation strategy for this is to set soft limits for verification runs. For

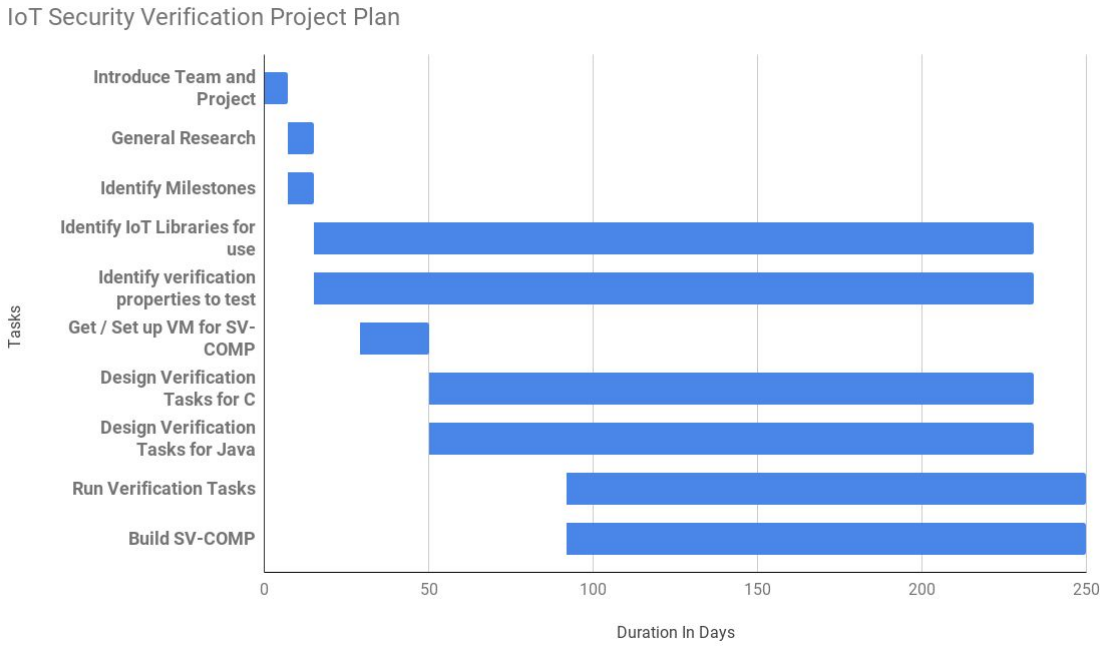
example, a normal run would be considered <24hrs, but anything >24hrs we will consider as an unknown failure.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

There are currently seven milestones for this project, and each of them fall under a specific deliverable. The first milestone falls within the deliverable, “Project Research”, and is described as being met once we have sufficiently researched key materials related to our project abstract. The second milestone is met once the team has identified both the milestones and timelines of the project. This falls under the “Prepare Project Plan” deliverable. The third milestone, within the “Set-up SV-COMP” deliverable, is met once the team builds the SV-COMP environment, and is able to run verification tasks against IoT libraries. The fourth milestone is achieved once the team has successfully completed designs for verification tasks that run against C Iot code. The fifth milestone is the same as the fourth except that it will be running against Java IoT code. The sixth milestone combines both the C and Java verification tasks completed in the previous two milestones and is used against common IoT libraries the team finds. Finally, the seventh milestone will be completed once we compile all of the benchmarks made from the previous milestone and construct a verifiably secure IoT library. The first two milestones are evaluated by all of the team members, and are considered completed once we are satisfied with the work put into them. The remaining five milestones will also be evaluated by all team members, but will rely more heavily on the functionality of our designs.

2.4 Project Timeline/Schedule

Our Gantt chart is still a work in progress as we continue to discover new requirements and difficulties. This being said, our schedule plans long periods of time for the tasks that we think may uncover new difficulties and will take the longest. These long periods of time will allow us to break them down into smaller tasks. This way the project end date will not be delayed.



Introduce Team and Project (8/24/2020 - 8/31-2020) - In this section we introduced ourselves to our team, client, and advisor. We also were given a summary of what our project would consist of. As a team, we decided when meetings would be and set internal roles and responsibilities.

General Research (8/31/2020 - 9/8/2020) - We were assigned research topics by our advisor to understand our project in a further manner. We created slides containing our research and additional questions we had. These questions turned into further research.

Identify Milestones (8/31/2020 - 9/8/2020) - After being given an overview of the project we brainstormed general milestones that our team could use to guide our project. Though we came up with a set of milestones, we all agreed that this would continue when new goals arose.

Identify IoT Libraries for use (9/8/2020 - 4/15/2021) - One major portion of our project is to identify IoT libraries that we may use for verification tasks. This milestone started by identifying criteria that we may consider before picking a library. It continues to 4/15/2020 because we will continue to pick and choose new libraries as we continually design verification tasks.

Identify verification properties to test (9/8/2020 - 4/15/2021) - Verification properties are the properties that we test in a library to determine if it meets certain criteria (in our case, security). Verification properties will continually be edited as we consider new libraries. Because we are also considering new libraries until 4/15/2021, we must also identify new verification properties until this same time.

Get / Set up SV-COMP (9/22/2020 - 10/13/2020) - Our team must request and set up virtual machines such that SV-COMP and other testing tools may run. This time period is 3 weeks because we must wait on the ISU IT department to give us access to these VMs.

Design Verification Tasks for Java (10/13/2020 - 4/15/2021) - Our team must use the chosen libraries to design verification tasks to run in SV-COMP. These verification tasks will be written on libraries specifically in Java. As we pick new libraries we will also need to write new verification tasks for these libraries. Since we will be considering libraries until 4/15/2021, we must also write new verification tasks until this point.

Design Verification Tasks for C(10/13/2020 - 4/15/2021) - Our team must use the chosen libraries to design verification tasks to run in SV-COMP. These verification tasks will be written on libraries specifically in C. As we pick new libraries we will also need to write new verification tasks for these libraries. Since we will be considering libraries until 4/15/2021, we must also write new verification tasks until this point.

Run Verification Tasks (11/24/2020 - 5/1/2021) - As our team writes verification tasks we will continually be running and testing them to make sure of completion. This process has a potential to take a long time and thus takes up most of our allotted schedule. After 4/15/2021 (When no new libraries will be chosen by our team), we will continually develop with the libraries that we have in our hands at that time. By 5/1/2021 we wish to have all verification tasks for the libraries in a complete state.

Build SV-COMP (11/24/2020 - 5/1/2021) - Verification tasks will be compiled and run in the SV-COMP environment. As verification tasks are completed - they will also be ran in the SV-COMP environment.

2.5 Project Tracking Procedures

The group has decided to utilize a waterfall method and a gantt chart to track the progress of the project in its various stages. The Waterfall model will be used to make sure that all members are aware of what parts of the project are being worked on at what time. Should additional information force the group to focus on one specific project within the project, additional time will be allocated within the waterfall model so that everything is kept on track for our deliverable due date.

The Gantt chart will give a detailed breakdown of what parts of the project have been completed, and how much longer the group must work on that particular area of the project. The chart will be updated during every meeting, and will be an accurate depiction of the progress that has been made.

2.6 Personnel Effort Requirements

Task	Person Hours
General Research (8/31/2020 - 9/8/2020)	200-300
Identify IoT Libraries for use (9/8/2020 - 4/15/2021)	60-100
Identify verification properties to test (9/8/2020 - 4/15/2021)	60-100
Get / Set up SV-COMP (9/22/2020 - 10/13/2020)	250-400
Design Verification Tasks for Java (10/13/2020 - 4/15/2021)	500+
Design Verification Tasks for C(10/13/2020 - 4/15/2021)	500+
Run Verification Tasks (11/24/2020 - 5/1/2021)	400+

2.7 Other Resource Requirements

We will be using virtual machines provided by the university to run our SV-COMP and do our model checking.

2.8 Financial Requirements

All code and virtual machines are provided via the university, and all code is personally made or publically available. As such, currently, the project requires no financial support in order to proceed.

3 Design

3.1 Previous Work And Literature

Include relevant background/literature review for the project

– If similar products exist in the market, describe what has already been done

– If you are following previous work, cite that and discuss the **advantages/shortcomings**

– Note that while you are not expected to “compete” with other existing products / research groups, you should be able to differentiate your project from what is available

Detail any similar products or research done on this topic previously. Please cite your sources and include them in your references. All figures must be captioned and referenced in your text.

3.2 Design Thinking

Detail any design thinking driven design "define" aspects that shape your design. Enumerate some of the other design choices that came up in your design thinking "ideate" phase.

3.3 Proposed Design

Include any/all possible methods of approach to solving the problem:

- Discuss what you have done so far – what have you tried/implemented/tested?
- Some discussion of how this design satisfies the **functional and non-functional requirements** of the project.
- If any **standards** are relevant to your project (e.g. IEEE standards, NIST standards) discuss the applicability of those standards here
- This design description should be in **sufficient detail** that another team of engineers can look through it and implement it.

3.4 Technology Considerations

Highlight the strengths, weakness, and trade-offs made in technology available.

Discuss possible solutions and design alternatives

3.5 Design Analysis

- Did your proposed design from 3.3 work? Why or why not?
- What are your observations, thoughts, and ideas to modify or iterate over the design?

3.6 Development Process

Discuss what development process you are following with a rationale for it – Waterfall, TDD, Agile. Note that this is not necessarily only for software projects. Development processes are applicable for all design projects.

3.7 Design Plan

Describe a design plan with respect to use-cases within the context of requirements, modules in your design (dependency/concurrency of modules through a module diagram, interfaces, architectural overview), module constraints tied to requirements.

4 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, or software.

1. Define the needed types of tests (unit testing for modules, integrity testing for interfaces, user-study or acceptance testing for functional and non-functional requirements).
2. Define/identify the individual items/units and interfaces to be tested.
3. Define, design, and develop the actual test cases.
4. Determine the anticipated test results for each test case
5. Perform the actual tests.
6. Evaluate the actual test results.
7. Make the necessary changes to the product being tested
8. Perform any necessary retesting
9. Document the entire testing process and its results

Include Functional and Non-Functional Testing, Modeling and Simulations, challenges you have determined.

4.1 Unit Testing

– Discuss any hardware/software units being tested in isolation

4.2 Interface Testing

– Discuss how the composition of two or more units (interfaces) are to be tested. Enumerate all the relevant interfaces in your design.

4.3 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

4.4 Results

– List and explain any and all results obtained so far during the testing phase

- Include failures and successes
- Explain what you learned and how you are planning to change the design iteratively as you progress with your project
- If you are including figures, please include captions and cite it in the text

5 Implementation

Describe any (preliminary) implementation plan for the next semester for your proposed design in 3.3.

6 Closing Material

6.1 Conclusion

Summarize the work you have done so far. Briefly re-iterate your goals. Then, re-iterate the best plan of action (or solution) to achieving your goals and indicate why this surpasses all other possible solutions tested.

6.2 References

List technical references and related work / market survey references. Do professional citation style (ex. IEEE).

6.3 Appendices

Any additional information that would be helpful to the evaluation of your design document.

If you have any large graphs, tables, or similar data that does not directly pertain to the problem but helps support it, include it here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc,. PCB testing issues etc., Software bugs etc.