Project Title: AI Generated Formal Software Security Vulnerability Descriptions

Primary Author: Irena Bojanova

Primary Author Division: 775

Submission Type: Research, $250k

Subject Area: Reliable Computing

Context of Proposal Research:

Defects in code may lead to security vulnerabilities and failures that are commonly used to attack cyberspace and the critical infrastructure. The first essential step towards preventing such attacks, logically, is to identify bugs and weaknesses, detect vulnerabilities and develop defenses against new exploits. For example, a wrong data type could cause a casting or calculation error, resulting in a value that is inconsistent with the size of a buffer, leading to a pointer repositioning overbound, which would allow reads of data that should not be read – aka buffer overflow, which if exploited can expose sensitive information, such as passwords and private keys.

The creation of modern security tools needs structured taxonomies of bugs/weaknesses and detailed, unambiguous descriptions of observed vulnerabilities initiated by software security weaknesses. The focus is on both code and specification defects. The resulting errors from such defects may propagate through software security faults, including in embedded systems, and possibly resurface and propagate through security faults in high-level OSs, leading to severe final errors and security failures. A repository of formally described software security vulnerabilities will allow in-depth understanding and formal analysis of the software security weaknesses underlying software security vulnerabilities to improve bug fixing and mitigation.

Technical Plan:

We will create a dataset of formally described software security vulnerabilities. For that, we will use the NIST Bugs Framework (BF) approach and methodology and will create tools for automated document curation and vulnerability descriptions generation.

To find and fix bugs that trigger software security vulnerabilities or to be able to reason over mitigation techniques, we need to first clearly understand all the chained underlying weaknesses that lead to an observed security failure. BF's approach, methodology, and formal vulnerability model assure this and allow the creation of appropriate tools (including use of AI based methods) that would also allow the creation of a dataset of vulnerability descriptions for use in AI/ML research. We anticipate the following tasks:

Stage I: We will first create bugs taxonomies. In more details, we will:

- Analyze bugs, weaknesses, and vulnerabilities related repositories and literature.
- Perform brainstorming sessions to create bug models.
- Perform brainstorming sessions to create taxonomies, formal language, and ontology for software security bugs/weaknesses/vulnerabilities.

- Create a public searchable online repository (and API) with bugs/weaknesses taxonomies.

Stage II: We will then create AI-related tools for retrieving bugs, weaknesses, and vulnerability related terms and structured information from software security reports, non-formal repositories, GitHub source codes, and other related sources. These tasks are impossible to do manually. Their automation would be achievable only via the developed in Stage I taxonomies and ontology. In more details, we will:

- Create tools for mapping identified entries to operations and other entities of our ontology.
- Create a tool to generate lists of easy to describe software security vulnerabilities with the taxonomies based on multiple techniques including mining of unstructured information and possibly the actual buggy code.
- Create a tool for generation of matrices of all meaningful cause-consequence (fault-operation-error) transitions for our taxonomies. The eventual combined matrix would comprise the transition paths of all theoretically possible security vulnerabilities allowing awareness about future vulnerabilities, as well as backtracking from an observable security failure to its possible root cause bugs that need to be fixed.
- Create a tool for generation of formal software security vulnerability descriptions.
- Generate and curate formal software security vulnerability descriptions for use in AI and ML.
- Create a public searchable online repository (and API) with formal vulnerability descriptions.

Potential Impact:

We anticipate the following deliverables:

- Data and operations flow models for identified phases of execution in which software security bugs and weaknesses happen.
- A formal language for clearly describing software security vulnerabilities, triggered by code or specification defects. Such a vulnerability description will be a “defect – error/fault–…–error/fault–final error” chain, leading to a security failure.
- A formal LL1 grammar in BNF form and an ontology corresponding to the taxonomies and the BF vulnerability model.
- Several database tools supporting the previously listed deliverables.
- A dataset of labeled vulnerabilities descriptions for use by software security tools and other AI-based projects related to chips security vulnerabilities and failures.
- Several AI-based tools supporting the creation of the dataset.

The formal classification system of software security bugs and weaknesses and the database of formally described software security vulnerabilities will benefit wide areas of government, industry, and academia projects related to security. The developed taxonomies, formal language, and ontology will help improve the software testing tools and their bug reports and facilitate the implementation of automatic bugs finding and fixing. The taxonomies will enable detailed, unambiguous reporting from software code analyzers. The large datasets of correct, meaningful

formally described known vulnerabilities will enable the creation/training of ML and AI models for bug finding and vulnerability detection. Clear descriptions of software security vulnerabilities will help also improve and extend the public vulnerability repositories). Finally, the created bug models and vulnerabilities dataset would allow academics to teach better about software security bugs and weaknesses and conduct formal research about software security vulnerabilities and security failures.