
# OptiSan: Using Multiple Spatial Error Defenses to Optimize Stack Memory Protection within a Budget

Rahul George, *University of California, Riverside;* Mingming Chen and Kaiming Huang, *The Pennsylvania State University;* Zhiyun Qian, *University of California, Riverside;* Thomas La Porta, *The Pennsylvania State University;* Trent Jaeger, *University of California, Riverside*

https://www.usenix.org/conference/usenixsecurity24/presentation/george

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

# USENIX Security '24 Artifact Appendix: OPTISAN: Using Multiple Spatial Error Defenses to Optimize Stack Memory Protection within a Budget

Rahul George[2], Mingming Chen[1], Kaiming Huang[1],
Zhiyun Qian[2], Thomas La Porta[1], Trent Jaeger[2]

[1]The Pennsylvania State University
[2]University of California, Riverside

## A Artifact Appendix

### A.1 Abstract

This artifact provides a comprehensive guide on installing and utilizing our proposed tool, OPTISAN, for generating placements of spatial memory defenses for stack objects that utilizes multiple defenses (Address Sanitizer and Baggy Bounds) where the goal is to maximize the protection of stack memory from spatial memory errors within a cost budget. The OPTISAN system includes: (1) a static analysis to compute which stack objects may be exploited by unsafe memory operations, which we call *usable targets*, to estimate the security impact ; (2) static analyses to to save the execution profile and model the performance of each defense, using this execution profile, - the costs for the placement of metadata management and bounds checks; and (3) a *mixed-integer non-linear program* (MINLP) formulation to generate an optimal solution for protecting usable targets within a cost budget. Within this artifact, we outline the necessary prerequisites, requirements, and software dependencies for OPTISAN, along with detailed instructions on accessing, setting up, and installing the tool.

### A.2 Description & Requirements

The source code of OPTISAN consists of multiple LLVM passes for security analysis, cost estimation and instrumentation. It also includes the MINLP Formulation (Matlab + Gurobi), an implementation of Baggy Bounds (ported from an open-source project) and several utility scripts to help with various tasks such as generating inputs for the solver. OPTISAN also relies on code from prior works to identify the spatially unsafe operations and to generate the program dependency graph along with the SVF tool (available at https://github.com/SVF-tools/SVF). We also include the necessary patches, inside the patches folder, to modify the AddressSanitizer's (ASan) and Baggy Bounds instrumentation mechanism to disable classes of operations as needed. It is important to note that compiling and building the LLVM source code requires CMake or the Ninja build system.

#### A.2.1 Security, privacy, and ethical concerns

Our evaluation includes building and running multiple programs with known vulnerabilities, many of which are taken from the MAGMA dataset. Hence, caution must be taken when reproducing this.

#### A.2.2 How to access

The source code of OPTISAN is available publicly on GitHub at https://github.com/rahultgeorge/OptiSan with commit *e6c8a2c81d8d6a24fd0620226de10a7c05125609*. The full URL is https://github.com/rahultgeorge/OptiSan/tree/e6c8a2c81d8d6a24fd0620226de10a7c05125609

#### A.2.3 Hardware dependencies

To compute the unsafe memory operations, using static value range analysis from prior work, requires 48 GB of memory.

#### A.2.4 Software dependencies

To compile and build OPTISAN's LLVM passes we use LLVM 10 on Ubuntu 22.04. You will need CMake version 3.20 or higher to successfully compile and build the LLVM source code. Further, we require Python3 for the scripts, Neo4j 3.5 (graph database) to save the results, and MATLAB along with Gurobi to compute the defense placement. However, setting up MATLAB and Gurobi cannot be automated as they require licenses. We include the relevant links to obtaining both and activating Gurobi for MATLAB. Ideally, on a system with all the necessary prerequisites, all components should build successfully.

### A.2.5 Benchmarks

All the datasets and source codes that have been used in our evaluation are publicly available and listed in the paper.

## A.3 Set-up

To replicate the evaluation environment for OPTISAN, we recommend setting up an Ubuntu 22.04 distribution. We provide a convenient script called *install_deps.sh* which installs all the necessary prerequisites and required components such as LLVM, Neo4j. It also builds LLVM 10.0 and SVF from source. If you have a fresh installation of the Ubuntu 22.04 distribution, please follow these steps from the root directory of our repository:

```
./install_deps.sh
```

By following these steps, you can quickly set up the required environment for OPTISAN and ensure all dependencies are properly installed. As noted in the previous section, MATLAB and Gurobi need to set up manually.

### A.3.1 Installation

To compile and build OPTISAN along with the LLVM source, one needs to issue the build script (*build_optisan.sh*) provided in our repository. This script will build all the OPTISAN LLVM passes.

### A.3.2 Basic Test

We include a simple example which setups a program from the MAGMA dataset and tests the core components of our OPTISAN.

```
./setup_example.sh
python3 run_example
```

This script will do four major tests - static analyses, cost estimation, defense placement computation and the instrumentation.

## A.4 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.