

# SAINT: SIMPLE TAINT ANALYSIS TOOL

## USER'S MANUAL

by

*Dipl.-Inf. Xavier Noumbissi Noundou*  
*xavier.noumbis@gmail.com*

June 9, 2015

## Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Installation Instructions</b>	<b>2</b>
2.1	Required Software . . . . .	2
2.2	Environment Variables . . . . .	2
2.3	How to Configure "clang+llvm" for use with SAINT . . . . .	3
2.4	How to Configure "LLVM" for use with SAINT . . . . .	3
2.5	How to Configure "poolalloc" for use with SAINT . . . . .	4
<b>3</b>	<b>Compiling and Running SAINT</b>	<b>4</b>
3.1	Folder Structure . . . . .	4
3.2	Configuration Files . . . . .	4
3.3	Running SAINT . . . . .	4

## 1 Abstract

Businesses increasingly use software. This is even more relevant for companies relying on e-commerce. However, software is error-prone and contain several bugs. Security bugs are one of the major problems faced by companies today. In the worst case, security bugs enable unauthorized users to gain full control of an application.

My PhD thesis introduces the concept of *tainted paths* and describes techniques and algorithms to compute them in any imperative programming language that uses pointers (C, C++, Java, etc.). I implemented these algorithms in SAINT.

SAINT computes tainted paths in a C program without running it. SAINT does not require the developer to annotate the program under analysis. SAINT implements a flow-sensitive, interprocedural and context-sensitive analysis that computes tainted paths in C programs at compile-time.

## 2 Installation Instructions

This section of the manual explains how to install `SAINT` on a Linux machine. We have not tested `SAINT` on a Windows machine, but the installation should follow similar steps.

### 2.1 Required Software

This section enumerates all software that you need to run `SAINT`.

- a) `SAINT`: <https://github.com/xnoumbis/saint.git>
- b) The compiler infrastructure **LLVM**, version 3.3 (<http://llvm.org>)
- c) The precompiled LLVM's tool chain **clang+llvm**, version 3.3 which include binaries like `clang`, `llvm-link`, etc.
- d) The DSA pointer analysis **poolalloc** (<https://github.com/llvm-mirror/poolalloc.git>).

### 2.2 Environment Variables

Table 1 that shows all environment variables that you have to define and export in order to successfully run `SAINT`.

Environment variables	Description
<b>SAINT_HOME</b>	<code>SAINT</code> home folder (e.g.: <code>/home/user/saint</code> )
<b>LLVM_HOME</b>	<code>llvm</code> home folder (e.g.: <code>/home/user/llvm</code> )
<b>LLVM_LIB</b>	<code>llvm</code> compiled libraries folder (e.g.: <code>\$LLVM_HOME/build/Release+Asserts/lib</code> )
<b>LLVM_BIN</b>	<code>llvm</code> compiled binaries (e.g.: <code>\$LLVM_HOME/build/Release+Asserts/bin</code> )
<b>POOLALLOC</b>	<code>poolalloc</code> home folder (e.g.: <code>/home/user/poolalloc</code> )
<b>CLANGLLVMBIN</b>	<code>clang+llvm</code> binaries' folder (e.g.: <code>/home/user/clang+llvm/bin</code> )

Table 1: Table with all environment variables required to install and use `SAINT`

✓ You define and export an environment variable **ENV\_VAR** by writing the following commands in your `".bashrc"` file:

```
ENV_VAR=path_to_folder
export ENV_VAR
```

## 2.3 How to Configure "clang+llvm" for use with SAINT

- a) Download and unpack **clang+llvm**, version 3.3.
- b) Add the `bin` folder to your environment variable **PATH**.
  - ✓ For instance by adding the following line in your file `".bashrc"`

```
PATH=$PATH:$CLANGLLVN_BIN
export PATH
```

## 2.4 How to Configure "LLVM" for use with SAINT

- a) Create a folder "build" in **\$LLVM\_HOME**.
- b) Copy and customized the script `configure-llvm.sh` from SAINT's "script" folder into the newly created "build" folder.
- c) Create a symbolic link to SAINT sources in the folder "**\$LLVM\_HOME/lib/Analysis**". You can achieve this by running: `ln -s $SAINT_HOME/src $LLVM_HOME/lib/Analysis/saint`.
- d) Open the file "**\$LLVM\_HOME/lib/Analysis/Makefile**" and append the string "saint" to the 'DIRS' variable. Following is an excerpt of the file.

```
##===- lib/Analysis/Makefile -----*- Makefile
#
#                               The LLVM Compiler Infrastructure
#
# This file is distributed under the University of Illinois Open Source
# License. See LICENSE.TXT for details.
#
##===-----

LEVEL = ../..
LIBRARYNAME = LLVMAnalysis
DIRS = IPA saint
BUILD_ARCHIVE = 1

include $(LEVEL)/Makefile.common
```

- e) Run the script `configure-llvm.sh`.
- f) Run the command `make` within the 'build' folder: `cd $LLVM_HOME/build; make`.
- g) After the previous step, you can run `make` from the SAINT folder during the development: `cd $SAINT_HOME; make -f Makefile.saint`.

## 2.5 How to Configure "poolalloc" for use with SAINT

- a) The sources of the DSA pointer analysis `poolalloc` can be gathered using the command `"git clone https://github.com/llvm-mirror/poolalloc.git"`.
- b) After getting the sources of `poolalloc`, the user has to checkout the `git` version under commit `'181c62f1d29ae9de660bad0a6593130d15803abc'` using the command `"git checkout 181c62f1d29ae9de660bad0a6593130d15803abc"`.
- c) Copy and customized the script `configure-poolalloc.sh` from `SAINT`'s "script" folder into `$POOLALLOC`, and run it.
- d) Then run `"make"`, and `"make install"`. Make sure to run `"make install"` as root (or administrator on a Windows system).
- e) Create a symbolic link to `poolalloc`'s `dsa` include folder in the folder `"$LLVM_HOME/include"`. You can achieve this by running: `"ln -s $POOLALLOC/include/dsa $LLVM_HOME/include/dsa"`.

## 3 Compiling and Running SAINT

✓ You need to execute the command `"make -f Makefile.saint"` within the folder `"$LLVM_HOME/lib/Analysis/saint"` to compile `SAINT`.

Also, the `SAINT` gets compiled when you run it using the Bash script `runOpt.sh`.

### 3.1 Folder Structure

The following folders constitute `SAINT` directory structure:

- "benchmarks": folder with sample scripts to run `SAINT`.
- "doc": folder with the manual.
- "scripts": folder with installation configuration files for `poolalloc` and `LLVM`.
- "src": folder with all C++ source files, and Bash scripts to compile and run `SAINT`.
- "src/cfg": folder with all configuration files

### 3.2 Configuration Files

### 3.3 Running SAINT

Among others, `SAINT` source folder contains the following two important Bash scripts:

- a) `ctainthelp.sh`: this script is used to generate and merge `llvm` intermediate representation (IR) files.
- b) `runOpt.sh`: this script is used to run the analysis of `SAINT` on the program under analysis. We encourage users to look at the sample scripts in the folder "benchmarks" to learn how to use `runOpt.sh`.