ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX'23 Artifact Appendix: Trojan Source: Invisible Vulnerabilities

Nicholas Boucher & Ross Anderson

A Artifact Appendix

A.1 Abstract

We provide a repository with proofs-of-concept implementing Trojan Source attacks in C, C++, C#, JavaScript, Java, Rust, Go, Python, SQL, Bash, Assembly, and Solidity. When viewed and compiled using vulnerable tools, these short proof-of-concept programs will output different values when executed than would be expected from reading the rendered source code.

A.2 Description & Requirements

Our paper describes a manner of encoding source code that can hide malicious logic in a manner that is not rendered to the user. Our artifact is a collection of proofs-of-concept, for which validation will take the form of verifying that the proofs-of-concept visually match the claimed rendering described in the paper and output the same adversarial logic when executed as described in the paper.

A.2.1 Security, privacy, and ethical concerns

The code provided does not take destructive action. While execution of the provided programs will output different values than expected from reading the rendered source code, viewing, compiling, and executing the provided code is designed to have no negative consequences for the reviewer.

A.2.2 How to access

The artifact is provided as a GitHub repository: `https://github.com/nickboucher/trojan-source/tree/e3dc153fcf465f4a84424ea874ff39be29adb1f7`.

A.2.3 Hardware dependencies

None

A.2.4 Software dependencies

Validating this artifact will required opening the proofs-of-concept programs any vulnerable language (Table 2 of the paper) in a vulnerable code viewer (Table 3 of the paper). We recommend C viewed with Visual Studio Code (v1.61) and compiled with `clang` (v12.0.*), and these tools are reasonably cross-platform.

Our experiments were repeated across Window 10 build 19043, MacOS Big Sur, and Ubuntu 20.04, although we anticipate that any modern version of Windows, MacOS, or Ubuntu will work.

A.2.5 Benchmarks

None

A.3 Set-up

Clone the artifact repository.

A.3.1 Installation

Install at least one vulnerable compiler and code viewer as listed Section A.2.4.

A.3.2 Basic Test

Validate that C/commenting-out.c is rendered as shown in Figure 4 of the paper when opened in a vulnerable editor such as Visual Studio Code (v1.61)

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): Trojan Source attacks produce different outputs when executing compiled programs written with Trojan Source techniques than would be expected from the rendered source code (absent any defenses).

A.4.2 Experiments

(E1): *[10 human-minutes + .01 compute-hours + <1GB disk]*:
How to: Open each of the proofs-of-concept in the C sub directory of the artifact repository using a vulnerable language and compiler as described in Section A.2.4. Confirm that the code is visualized as shown in Figure 4

of the paper. Compile the code, and confirm that the output matches the output claimed in the paper and differs from what would be expected from reading the rendered source code.

Preparation: Complete the software dependency installations described above.

Execution: Compile and execute the proofs-of concept, with e.g. `clang commenting-out.c && ./a.out`.

Results: The output should match Section 4.2 of the paper.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.