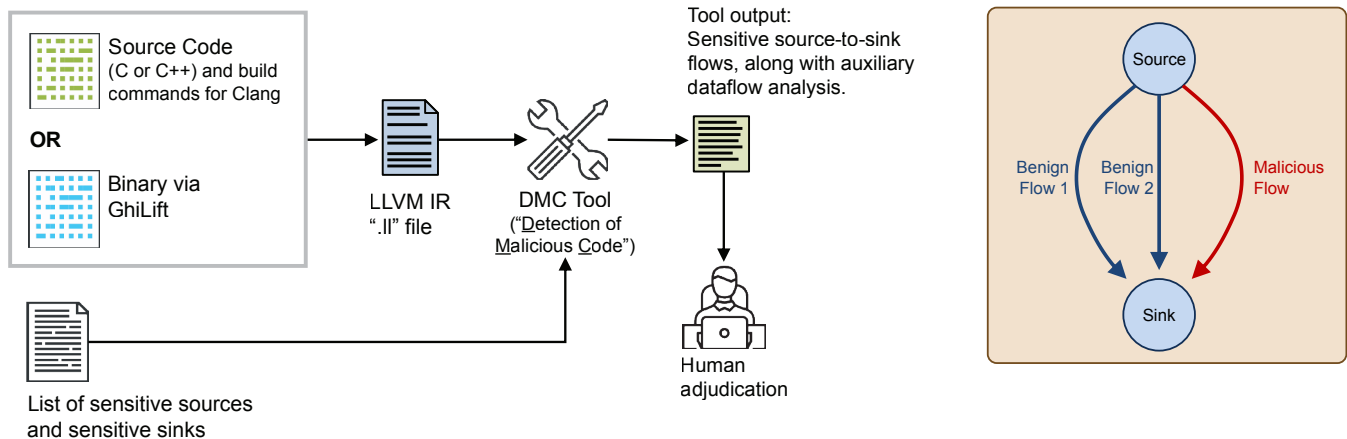


Detection of Malicious Code: Taint Flow Analysis for Weapons Systems Software

for the 2024 DoD Weapon Systems Software Summit



Our tool detects two types of malicious code: (1) exfiltration of sensitive data and (2) timebombs, logic bombs, remote-access Trojans (RATs), and similar malicious code. We use only static analysis, not dynamic analysis, so we do not need to execute potentially malicious code. While our focus is on C/C++ codebases, we perform all analysis at the level of LLVM intermediate representation (IR), so we also have some support for binaries via lifting to LLVM IR. Our tool uses taint analysis to track the flow of sensitive and auxiliary information. A limitation of traditional taint analysis is that it conflates all flow paths from a given source to a given sink, so a malicious flow path can be "hidden" by a benign flow path. In contrast, in this project we use auxiliary information (e.g., whether a filename was specified by a local user or whether it came from network data) to help separate benign flows from malicious flows.

Our tool flags code as potentially malicious, but further human analysis is required. Whether behavior is malicious depends on the what the program is supposed to do. The goal for our tool is to produce output that concisely and precisely characterizes the potentially malicious behaviors of the codebase, so that a human analyst can quickly and accurately determine whether the behavior is benign or malicious.

Tool on GitHub



Project page:
<https://github.com/cmu-sei/dmc>

Contact us if you're interested in using our tool or working with us to extend it !



Will Klieber
weklieber@sei.cmu.edu

Software Security Engineer
Principal Investigator



Lori Flynn, PhD
lflynn@sei.cmu.edu

Senior Software
Security Researcher

Copyright 2024 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific entity, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute nor of Carnegie Mellon University - Software Engineering Institute by any such named or represented entity.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License. Requests for permission for non-licensed uses should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM24-1567