Abstract—This hands-on tutorial teaches participants how to audit static analysis alerts, using an auditing lexicon and rules. There is no widely-accepted lexicon or standard set of rules for auditing static analysis alerts in the software engineering community. Auditing rules and a lexicon should guide different auditors to make the same determination for an alert. Standard terms and processes are necessary so that initial determinations are correctly interpreted, which helps organizations reduce code flaws. They are also needed to improve the quality of audit data to benefit research on alert prioritization. This tutorial teaches a suggested set of auditing rules and a lexicon, briefly detailing rationales based on modern software engineering practices for each rule and each lexicon term. The majority of time in the tutorial will be spent by participants working with provided small programs and associated static analysis alerts, examining them using the lexicon and rules to make a determination, separately and as a group. These hands-on activities will be interspersed with presenting the auditing rules, so participants immediately put what was taught into practice on relevant code and alerts. We hope that the auditing rules and lexicon taught will be immediately useful for participants to adopt (partially or in full) in their workplace, and that learning about them will motivate community discussion leading to agreed-upon standards.

I. TOPIC

In the software engineering community, there is no widely-accepted lexicon or standard set of rules for auditing static analysis (SA) tool alerts. This is in spite of static analysis being an integral part of the software development lifecycle [1][2]. A standard auditing rule set and lexicon should guide different auditors to reach the same determination for an alert, achieving consistent results by reducing ambiguity. Well-defined auditing terms and processes can help organizations to reduce flaws during code repair, code development, and future audits. A standard lexicon and rules can also improve audit data quality, benefitting research on alert prioritization and classification [3][4][5].

SA tools attempt to automatically identify defects in software products. At a high level, these tools define a set of conditions describing a well-behaved program (e.g., a null pointer shall not be dereferenced). SA tools then analyze a program to find violations of those conditions. The analysis may inspect the source code of the program, or some other representation thereof, such as the compiled binary.

However, SA tools often wrongly accuse programs of violating a condition. That is, static analysis is prone to false positives. Formally determining statically whether a program violates arbitrary conditions is undecidable. Hence, tools apply heuristics to find potential bugs, and these heuristics tend to over-approximate the occurrence of violations. False positives may also result from bugs in the tool.

Consequently, SA reports must be validated. This task often falls to human auditors.

This tutorial teaches participants a suggested lexicon and rules for auditing, and guides them in hands-on practise auditing real code and alerts.

II. TUTORIAL FORMAT AND RELEVANT MATERIALS

This tutorial teaches a suggested set of 12 auditing rules and a lexicon (both introduced in [6]), briefly detailing rationales based on modern software engineering practices for each rule and each lexicon term. The majority of time in the tutorial will be spent by participants working with provided small programs and associated SA alerts, examining them using the lexicon and rules to make a determination, separately and as a group. These hands-on activities will be interspersed with presenting the auditing rules, so participants immediately put what was taught into practice on relevant code and alerts.

Participants will receive small test programs and files with SA alerts for those programs. We will provide these on ALL of the following formats, to accommodate varying computing equipment (or none) brought by attendees:

- on a website for attendees to download in advance (along with SHA256 hash): [https://www.cert.org/audit_tutorial/](https://www.cert.org/audit_tutorial/)
- on a virtual machine (VM) distributed on a DVD
- as files they can copy from a thumbdrive
- on printouts

Although SA alerts will be provided in documents, participants with computers will also be encouraged to run SA tools on the test programs. The VMs will include SA tools, and participants can also run their own (non-VM) SA tools on the test programs. Participants can also use the printouts and pens we provide to make notes.

We plan to use (and in some cases, slightly modify) existing materials we previously developed to explain and teach about the alert auditing rules and lexicon, including:

- Slide sets developed for training courses about alert auditing rules (multi-day classes for DoD organizations and a 2-hour class at Carnegie Mellon University)
• Slides from our paper (“Static Analysis Alert Audits: Lexicon and Rules”) presentation at SecDev 2016.
• Example small test C programs we developed for the DoD training courses, as examples for various determinations.
• Example open-source C programs we audited some alerts for, for the DoD training courses, as examples for various determinations.

III. EXPECTED AUDIENCE AND LEARNING OUTCOMES

We expect two levels of competency for attendees:
• Those who plan to do the auditing exercises are expected to feel comfortable reading C code.
• Those who won’t necessarily do the auditing exercises, but are interested in learning about SA alert auditing in general and being exposed to related issues.

Participants will learn a lexicon and 12 rules for auditing SA alerts, which together guide different auditors to reach the same determination for an alert, achieving consistent results by reducing ambiguity. Participants will learn about corner cases that should be considered.

Participants who don’t start out comfortable reading C code can expect to learn some things about C. Those who are experienced C coders may still learn about CERT Coding Standards [7] and Common Weakness Enumeration (CWE) [8].

All who try (or observe) the auditing exercises will be exposed to SA alerts, and will learn techniques to examine code (particularly control flow, data flow, and data types) to make a determination about alerts.

Participants with computers will have the opportunity to use SA tools on programs, learning how to do so and gaining experience doing it. Those without computers still get opportunities to view SA tool use, seeing instructor demonstration and looking at other participants’ screens, so they should learn something about how to run the SA tools. (Those without computers can participate by using printouts with SA alerts to audit, plus printouts of the test programs.)

We hope that the auditing rules and lexicon taught will be useful for participants to adopt (partially or in full) in their workplace, and further that learning about them will motivate community discussion leading to agreed-upon standards.

IV. PRIOR SIMILAR- TOPIC PRESENTATIONS BY AUTHORS

This tutorial will be taught by the three researchers who developed the auditing rules and lexicon and co-authored a paper about it in SecDev 2016.

Lori Flynn has presented about her software security research at conferences and workshops (audiences ranging from 7-100 people), DoD organizations (including two 2-day trainings that covered alert auditing rules, with 31 trainee total), groups of technical researchers and others at her workplace, and to other technical organizations (including presenting to about 50 people at the Steel City Information Security meetup group). She and David delivered a 2-hour class on alert auditing rules for Carnegie Mellon University students in a Secure Coding course. Lori is leading the research project that developed this auditing lexicon and rules.

David Svoboda has given many presentations on software security in C and Java, including [9] with approximately 100 attendees and [10] with approximately 150 attendees.

William Snavely teaches a Secure Coding course for C/C++, with an average of 25 attendees per class. He presented “Static Analysis Alert Audits: Lexicon and Rules” in the main track of SecDev 2016, with approximately 100 people attending.

ACKNOWLEDGMENT

Copyright 2017 IEEE. All Rights Reserved. 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. References herein to any specific commercial 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. Carnegie Mellon and CERT are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. DM17-0319

REFERENCES