Carnegie Mellon University
Software Engineering Institute

Summer 2018 Edition

We published our last newsletter more than six months ago, which is longer than usual and longer than we generally like. We've been busy in these last six months growing our team. You can read about our new team members below. We are still looking for more team members, so if you have the interest and skills, apply using the information we provide in the Open Positions in the SEI CERT Secure Coding Team section of this newsletter.

This past month, we released our Source Code Analysis Lab (SCALe) tool to GitHub. This is SCALe's first open source release to the public. We're very excited to provide this tool to the community, and we hope to hear how you're using it.
As usual, we provide references to events and publications that may interest you, and we refer to the upcoming events where we'll be making presentations. We also provide a list of updates to our Secure Coding Standards.

guide and support the community and help improve the security of software through secure coding practices. We are interested, as always, in collaborating with others on research and developing new techniques to improve coding practices. Please reach out to us if you are interested in collaboration projects, and let us know how our contributions are helping you.

—Bob Schiela

News

SCALe Release on GitHub

We are pleased to announce the first-ever public, open source release of our SCALe (Source Code Analysis Lab) tool, which provides a framework for auditing static analysis alerts. You can download SCALe from GitHub at https://github.com/cmu-sei/SCALe.

SCALe provides a GUI front end that auditors can use to examine alerts from one or more static analysis tools and the associated code to make audit determinations (e.g., identify true violations and false positives) and export the project audit information to a database or CSV file(s). This open source version of SCALe provides categories of alerts for tools based on two code flaw taxonomies: SEI CERT Coding Standards and MITRE's Common Weakness Enumeration (CWE). The SEI CERT Coding Standards and SCALe provide analysts with detailed guidance for secure development in C, C++, Java, and Perl.

For the last three years, most of the development of the SCALe tool has been for classification and prioritization research projects, led by Lori Flynn, to add features and functionality needed by the projects. This initial GitHub release is based on a version of the research project's code from around February 2018. This version also includes some bug fixes and performance improvements made by the wider Secure Coding team to prepare the code for this release. Please contact us if you are interested in collaborating with us or using more recent (non-public) versions of SCALe and related software.

We have been developing SCALe since 2010. In 2015, we first shared SCALe with some non-SEI organizations. Some of the new features developed in the last three years that are part of the GitHub SCALe release include the following:

- A CWE taxonomy was added.
- Alerts can now map to multiple code flaws.
- Alerts with the same filepath, line, and condition are now fused for auditor efficiency.
- Hyperlinked fused-alert IDs now provide checks for related audit determinations and notes.
- Code metrics are now allowed to be used.
- A new 'Notes' field was added.

We hope you will try this version of SCALe, and we welcome your feedback and code contributions!

New Team Members

We are happy to introduce our newest Secure Coding team members: Ebonie McNeil, Ryan Steele, and Derek Leung. Jiyeon Lee and Theodor Johansson, student interns who worked through the summer, will also continue working with us in the fall.

Open Positions in the SEI CERT Secure Coding Team

Join our Secure Coding team. If you have the right qualifications and are interested in researching and developing improvements to the state of the art and practice in secure coding, secure development, and software assurance, please consider applying for any of the following positions:

- Senior Software Security Engineer
- Senior Software Assurance Engineer (Pittsburgh, PA | Arlington, VA | Bedford, MA)
- Associate Compiler Researcher
- Software Security Engineer
- Associate Software Security Engineer

Inter-Taxonomy Precise Mapping

CERT researchers worked with MITRE to test a new method we developed to do more precise mappings between taxonomies. We modified fields of our CERT coding standard to record precise mapping information. In December 2017 and January 2018, we updated rules in the CERT C standard to enable more precise mapping. We added new fields to the Related Guidelines table, and we added a new Mapping Notes field for rules that have a precise mapping or mapping notes.

We are collaborating with MITRE to refresh the mappings between CERT guidelines and MITRE CWEs. Both MITRE and the CERT Division will publish the revised mappings once they are complete.

Recent Events and Publications

- On August 29, the SEI hosted the CERT Data Science in Cybersecurity Symposium 2018 in Arlington, VA. Details about this event are available at https://www.eventbrite.com/e/cert-data-science-in-cybersecurity-symposium-2018-registration-46365336971.
- In August, Will Klieber presented at the International Workshop on Security of Mobile Applications (IWSMA 2018). The title of the paper he presented is “Practical and Precise Taint Flow Static Analysis for Android App Sets,” which he co-authored with Lori Flynn, Will Snavely, and Michael Zheng.
- On August 8, Secure Coding team members published SCALe to the general public for the first time. Download SCALe from GitHub at https://github.com/cmu-sei/SCALe.
- In June, Lori Flynn presented “Automating Static Analysis Alert Handling with Machine Learning” at the Cyber Security, Exploitation and Operations Workshop at MIT Lincoln Labs in Lexington, MA.
- In May, Lori Flynn, William Snavely, David Svoboda, Nathan Vanhoudnos, Richard Qin, Jennifer Burns, David Zubrow, Robert Stoddard, and Guillermo Marce-Santurio" presented "Prioritizing Alerts from Multiple Static Analysis Tools, Using Classification Models" at the SQUADE workshop at ICSE 2018.
- In April, Lori Flynn, Zach Kurtz, and Will Snavely published an SEI blog post titled, "Static Analysis Alert Test Suites as a Source of Training Data for Alert Classifiers."
- In April, Lori Flynn presented "Challenges and Progress: Automating Static Analysis Alert Handling with Machine Learning" to the Foundations and Applications of Program Analysis (COMS/CPRE 513x) class at Iowa State University, at the invitation of Dr. Wei Le. Slides are available from the SEI's Digital Library: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=518021.
- In February, David Svoboda presented an SEI Cyber Minute about SCALe.

- The presentation titled "Flight Software Programming Language Selection: A Security Perspective," written by Will Snavely and Craig Meyers, was accepted and will be presented at the 2018 AIAA Space conference.
- The paper titled "Detecting leaks of sensitive data due to stale reads," written by Will Snavely, Will Klieber, Ryan Steele, and David Svoboda, was accepted and will be presented at the IEEE SecDev 2018 conference in September.
- On October 9-10, 2018, the Secure Coding team, along with many other SEI researchers, will present their research at the SEI Research Review 2018. Details about this event are available on the SEI website: https://resources.sei.cmu.edu/news-events/events/research-review/index.cfm.

SEI CERT Secure Coding Standard Updates

We updated the page Top 10 Secure Coding Practices to include new information about the Konsequenz photograph depicted on the page.

CERT C Coding Standard

Editor: David Svoboda, CERT Division of the Software Engineering Institute

Download the latest stable version.

No C rules were removed.

Added

- MSC41-C. Never hard code sensitive information, which is analogous to Java's rule MSC03-J. Never hard code sensitive information

Changed

- Several improvements were made to DCL39-C. Avoid information leakage when passing a structure across a trust boundary:
  - The second noncompliant code example elaborates more clearly why the example can still leak sensitive information.
  - The first compliant solution zeroes out any uninitialized chars in the array before copying them, as copying uninitialized data

- Several improvements were made in EXP34-C. Do not dereference null pointers:
  - The first compliant solution has a new check to prevent a null pointer being passed to `memcpy()`, even if the call was trivial, because the behavior might still be undefined.
  - The final compliant solution now asserts that a pointer must not be null before it gets dereferenced.
- For cases of assignments in unusual contexts, EXP45-C. Do not perform assignments in selection statements now references EXP30-C. Do not depend on the order of evaluation for side effects. These errors typically arise from conflating = with ==. Furthermore, the code samples in the final noncompliant code example and compliant solution were changed to use || rather than &&, which is more practical.
- The arguments in the call to `func()` in the last compliant solution in EXP47-C. Do not call va_arg with an argument of the incorrect type have been reordered correctly.
- We replaced != with == in the POSIX non-compliant code example FIO30 -C. Exclude user input from format strings.
- FIO32-C. Do not perform operations on devices that are only appropriate for files now explicitly cites *Writing Secure Code* (by Michael Howard and David C. LeBlanc) for the claim that a web browser could lock the mouse by specifying <IMG src="file:///dev/mouse"> in HTML.
- The standard C function name was corrected to `c32rtomb()` in ERR30-C. Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure.
- In MSC30-C. Do not use the rand() function for generating pseudorandom numbers, the Windows compliant solution now uses `BCryptGenRandom()`instead of `CryptGenRandom()`, which is deprecated.
- The two recommendations EXP15-C. Do not place a semicolon on the same line as an if, for, or while statement and API01-C. Avoid laying out strings in memory directly before sensitive data now have Risk Assessment metrics.

wrapping and the possibility of calling malloc() with an argument of 0. These code examples are better addressed by rule INT30-C. Ensure that unsigned integer operations do not wrap and recommendation MEM04-C. Beware of zero-length allocations.

- Several improvements were made to MEM12-C. Consider using a goto chain when leaving a function on error when using and releasing resources.
  - All code examples assume that `errno` is set if `fopen()` or `malloc()` fail, but this is not guaranteed by C11; it is only guaranteed by POSIX. Consequently, all code examples are now limited in scope to POSIX.
  - The guideline has a new compliant solution that showcases using nested if clauses to handle errors and cleanup; it also discusses the pros and cons of using nested ifs versus goto chains.
  - The guideline cites Dijkstra's famous paper, "Go To Statement Considered Harmful."
- A rogue space character was eliminated in the "POSIX" compliant solution in FIO02-C. Canonicalize path names originating from tainted sources.
- MSC12-C. Detect and remove code that has no effect or is never executed now has an exception for unused library functions, as well as code that is #ifdef'ed out.
- MSC13-C. Detect and remove unused values now has an exception for initializing variables with a sentinel value, such as 0, to prevent reading uninitialized memory, as explained by EXP33-C. Do not read uninitialized memory.
- In CC. Undefined Behavior, the CERT rule CON37-C. Do not call signal() in a multithreaded program as incorrectly associated with UB 137. It is now correctly associated with UB 135.

CERT C++ Secure Coding Standard

Editor: David Svoboda, CERT Division of the Software Engineering Institute

Download the latest stable version.

No C++ rules were added or removed.

Changed

with properly aligned pointers to sufficient storage capacity. Incorrectly had a <=where it now correctly has a >.

- In the first noncompliant code example and compliant solution in EXP63-CPP. Do not rely on the value of a moved-from object, the function g() was changed to take an argument of type string, rather than string&&.
- OOP53-CPP. Write constructor member initializers in the canonical order now has an exception to excuse constructors that lack member initialization.
- The first sentence in OOP50-CPP. Do not invoke virtual functions from constructors or destructors better explains about when virtual functions are resolved.

CERT Oracle Secure Coding Standard for Java

Editor: David Svoboda, CERT Division of the Software Engineering Institute

Order the latest version of *The CERT Oracle Secure Coding Standard for Java* book.

No C rules were added or removed.

Changed

- MET09-J. Classes that define an equals() method must also define a hashCode() method now explains the OOD paradox between respecting the `Object.equals()` contract and allowing extendable subclasses.
- TSM03-J. Do not publish partially initialized objects. In the *Final Field* compliant solution, the introductory text now indicates that a final field's initialized value is available after (rather than before) construction has completed.
- MET12-J. Do not use finalizers now mentions that finalizers are officially deprecated as of Java 9.
- IDS04-J. Safely extract files from ZipInputStream now includes information about the ZipSlip vulnerability.
- MSC12-C. Detect and remove code that has no effect or is never executed now has an exception for unused library classes and methods.

CERT Secure Coding Standard for Android

Editor: Lori Flynn, CERT Division of the Software Engineering Institute

CERT Perl Secure Coding Standard

Editor: David Svoboda, CERT Division of the Software Engineering Institute

No Perl rules were added, changed, or removed.

Our People

In our newsletters, we highlight various staff members behind our Secure Coding research. In this issue, we feature Ryan Steele.

In January 2018, Ryan Steele joined the Secure Coding team as an Associate Software Security Analyst. His background is in C/C++ software development for FPGA-based embedded systems for aerospace, robotics, military, and industrial applications. His latest work focuses on programming language compilers and automated code repair.