SEI WEBINAR SERIES | Keeping you informed of the latest solutions

Software Engineering Institute Carnegie Mellon University



#SEIwebinar

[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.

Carnegie Mellon University

This video and all related information and materials ("materials") are owned by Carnegie Mellon University. These materials are provided on an "as-is" "as available" basis without any warranties and solely for your personal viewing and use.

You agree that Carnegie Mellon is not liable with respect to any materials received by you as a result of viewing the video, or using referenced websites, and/or for any consequences or the use by you of such materials.

By viewing, downloading, and/or using this video and related materials, you agree that you have read and agree to our terms of use (<u>www.sei.cmu.edu/legal/</u>).

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

© 2016 Carnegie Mellon University.



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University [Distribution Statement A] This material has been approved for public release and unimited distribution. Please see Copyright notice for non-US Government use and distribution.

Distribution Statements

Copyright 2016 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

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 material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon[®] and CERT[®] are registered marks of Carnegie Mellon University.

DM-0003519



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University 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.

Secure Coding Best Practices

Robert Schiela

Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213



© 2016 Carnegie Mellon University

[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.



Software Engineering Institute | Carnegie Mellon University

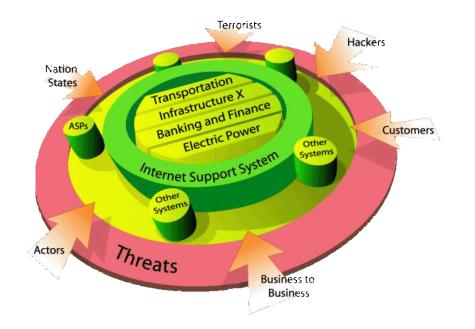
Why Software Security?

Developed nations' economies and defense depend, in large part, on the reliable execution of software

Software is ubiquitous, affecting all aspects of our personal and professional lives.

Software vulnerabilities are equally ubiquitous, jeopardizing:

- personal identities
- intellectual property
- consumer trust
- business services, operations, and continuity
- critical infrastructures & government



#SElwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University Distribution Statement AI This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Most Vulnerabilities Are Caused by Programming Errors

64% of the vulnerabilities in the NIST National Vulnerability Database due to programming errors

•51% of those were due to classic errors like buffer overflows, cross-site scripting, injection flaws

Top vulnerabilities include

- Integer overflow
- Buffer overflow
- Uncontrolled Format String
- Missing authentication
- Missing or incorrect authorization
- Reliance on untrusted inputs (aka tainted inputs)

Sources: Heffley/Meunier: Can Source Code Auditing Software Identify Common Vulnerabilities and Be Used to Evaluate Software Security?; cwe.mitre.org/top25 Jan 6, 2015



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University 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.

Secure Software Development

Secure software development starts with understanding insecure coding practices, and how these may be exploited.

Insecure designs can lead to "intentional errors", that is, the code is correctly implemented but the resulting software contains a vulnerability.

Secure designs require an understanding of functional and non-functional software requirements.

Secure coding requires an understanding of implementation specifics.



#SElwebinar

Assons Learned from the Jeep Hack April 11, 2016 2016 Carnegie Mellon University Distribution Statement A) This material has been approved for public release and unimitted distribution. Please see Copyright notice for non-US Governmen se and distribution.

Sources of Software Insecurity

Absent or minimal consideration of security during all life cycle phases Complexity, inadequacy, and change Incorrect or changing assumptions Not thinking like an attacker Flawed specifications & designs Poor implementation of software interfaces Unintended, unexpected interactions • with other components with the software's execution environment

Inadequate knowledge of secure coding practices

#SElwebinar

Unspecified and Undefined Behaviors

implementation-defined behavior - Unspecified behavior whereby each implementation documents how the choice is made.

unspecified behavior - Behavior for which the standard provides two or more possibilities and imposes no further requirements on which is chosen in any instance.

undefined behavior - Behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which the standard imposes no requirements. An example of undefined behavior is the behavior on integer overflow.

From ISO/IEC 9899-1999.



#SElwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University Distribution Statement AI This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Polling Question

Does your organization use a coding standard for security?

- Yes
- No
- Maybe?



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University Distribution Statement AJ This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Polling Question

What programming language do you work with most in your organization?

- Assembly
- C
- C++
- C#
- Java
- Java-Script
- Objective-C
- Perl
- PHP
- Python
- PL/SQL or SQL
- Ruby
- Swift
- Visual Basic
- Other
- Little to none developed in-house



#SElwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University [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.

Adopting Secure Coding Practices

Secure Coding Infrastructure

- Defining Secure Coding Practices
- Influencing Language Standards
- Influencing Tool Vendors

People

• Training

Processes

• Coding Standards and Security Standards, Testing

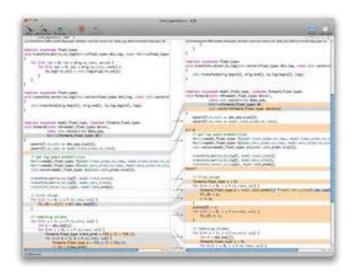
Technology

- Tools: IDE's and Analyzers
- Automated transformation and remediation

#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University [Distribution Statement A] This material has been approved for public release and unimited distribution. Please see Copyright notice for non-US Government use and distribution.

Tools encourage application of secure coding



	The desired & he Batta II Adver
<pre>A performance is in the second particular to the second particular</pre>	Control of the second split Control of the second split
Contracting Lines, 7 Security of Security Street, 2 Security Street, 1	
Parates Rent Add Lotter Tax	
E fans (f) fang S Fang E fang	

Moving rules into IDEs improves application of secure coding:

- Early feedback corrects errors on introduction.
- Exceptions are understood in context.

Adoption of secure coding IDEs

- help deploy tools
- training on tools
- extend tools to meet targeted needs



#SEIwebinar

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright **3** notice for non-US Government use and distribution.

Conformance Testing

The use of secure coding standards defines a proscriptive set of rules and recommendations to which the source code can be evaluated for compliance.

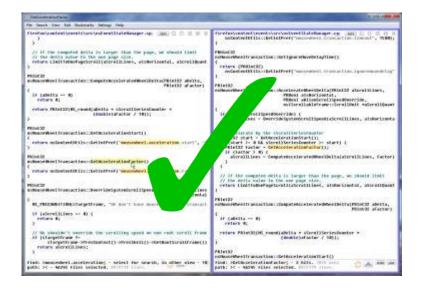
For each secure coding standard, the source code is certified as provably nonconforming, conforming, or provably conforming against each guideline in the standard:

Provably nonconforming	The code is provably nonconforming if one or more violations of a rule are discovered for which no deviation has been allowed.
Conforming	The code is conforming if no violations of a rule can be identified.
Provably conforming	Finally, the code is provably conforming if the code has been verified to adhere to the rule in all possible cases.

Evaluation violations of a particular rule ends when a "provably nonconforming" violation is discovered.



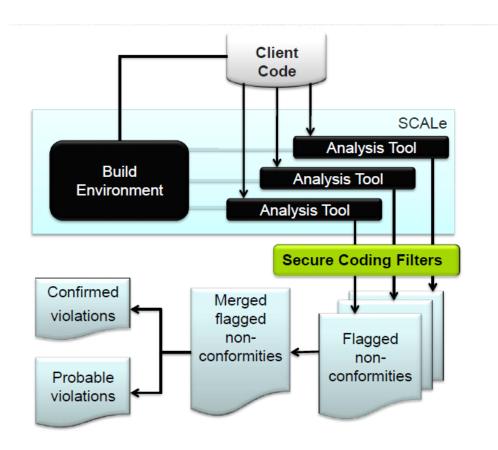
Static Testing – Source code analysis tools



Secure Code Analysis Laboratory (SCALe)

- C, C++, Java, PERL, Python, Android rule conformance checking
- Thread safety analysis
- Information flows across Android applications
- Operating system call flows

SCALe Multitool evaluation



Improve expert review productivity by focusing on high priority violations Filter select secure coding rule violations

- Eliminate irrelevant diagnostics
- Convert to common CERT Secure Coding rule labeling

Single view into code and all diagnostics

Maintain record of decisions

Polling Question

What testing does your organization perform on your software?

- Static Analysis
- Dynamic Analysis
- Both
- None



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University 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.

Supply Chain Software

Install Security Updates.

Test Source Code.

Review vendors' security and software assurance practices and results.

Request reports from their own testing or request independent security reviews and testing.

Test Binaries.

Apply Defense in Depth – only enable features that are required, and protect them.

Isolate critical and non-critical services and data.

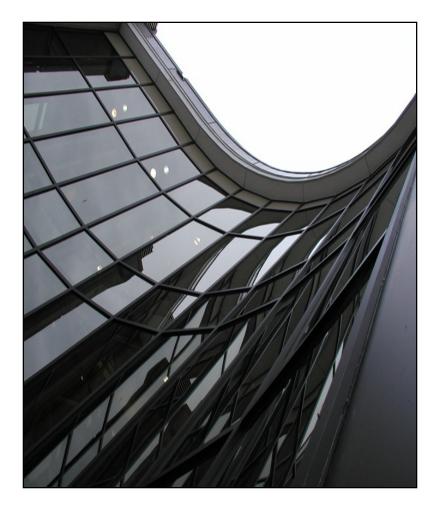
Perform penetration testing.

Install an independent monitoring system.

Contact Information

Robert Schiela Technical Manager, Secure Coding Telephone: +1 412.268.3637 Email: <u>rschiela@cert.org</u>

Web Resources http://www.sei.cmu.edu/ http://www.cert.org/ http://www.cert.org/secure-coding/ http://securecoding.cert.org/ (SEI CERT Secure Coding Standards Wiki)



#SEIwebinar

Lessons Learned from the Jeep Hack April 11, 2016 © 2016 Carnegie Mellon University 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.







#SElwebinar

[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.