WEBINAR

Poor Software Quality Costs the United States *How Much?*

January 27, 2021

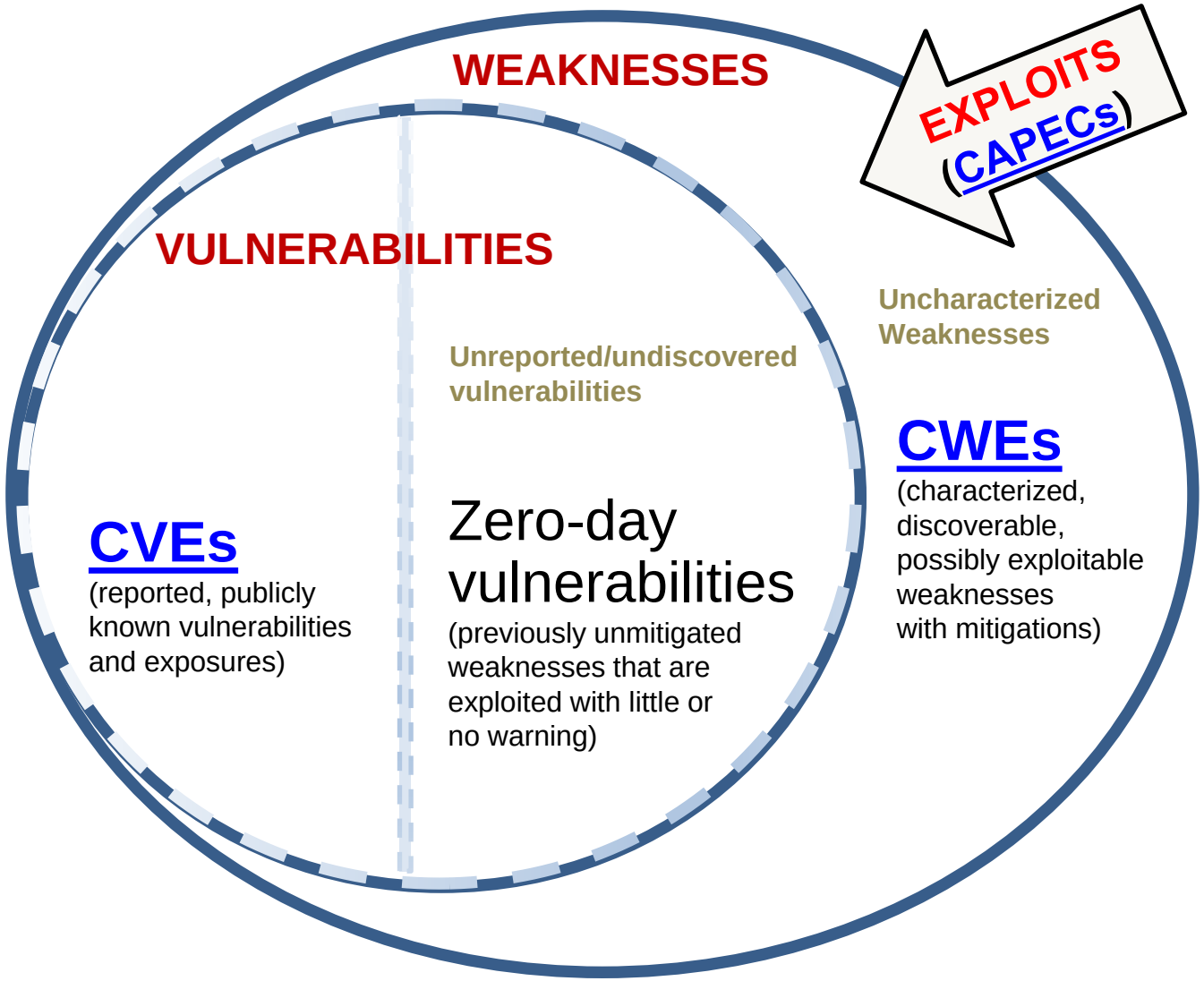
CISQ Consortium for Information and Software Quality

Recommendations for Addressing CPSQ

Joe Jarzombek, CSSLP
Director, Government & Critical Infrastructure Programs
Synopsys, Inc

Recommendations for Addressing CPSQ

- Minimize software failures by focusing on mitigation of weaknesses (CWEs) & vulnerabilities/exposures (CVEs):
  - Address security as part of quality defect prevention
  - Automate quality in workflows by using tools with 'security domain checkers under the hood'
  - Understand relationship of CWEs and CVEs, and existence of exploits designed to take advantage of a weakness (or multiple weaknesses) and achieve a negative technical impact is what makes a weakness a vulnerability
  - Leverage use of CISQ Automated Source Code Quality Measures based on CWEs, including Data Protection Measure.
- Shift left to address quality early in software lifecycle



Part of the ITU-T CYBEX 1500 series
(CVE ITU-T X.1520, CWE ITU-T X.1524, CAPEC ITU-T X.1544) & USG SCAP

Recommendations for Addressing CPSQ

- Minimize risks attributable to open source software (modules and libraries) by mitigating weaknesses (CWEs), vulnerabilities (CVEs) and license conflicts:
  - Automate the detection and mitigation of open source license issues
  - Automate quality in workflows using tools with ‘quality domain checkers under the hood’

- Leverage Synopsys report* on “DevSecOps Practices and Open Source Software Management 2020” to help answer a vital question ‘How does your organization compare to your peers?’ in terms of:
  - DevOps and the secure software development life cycle (SDLC)
  - Adoption of DevSecOps tools
  - Open source selection and governance
  - Open source security and patching
  - Open source project sustainability

*Referenced in “The Cost of Poor Software Quality in the US: A 2020 Report”, this Synopsys report is based on a survey of 1,500 software security and development professionals around the world, and it provides in-depth analysis of who, how, and why organizations of all sizes and in a variety of industries are tackling open source management and DevSecOps.

Recommendations for Addressing CPSQ

- Overcome ‘lack of understanding of internal functionality’ in legacy systems and identify weaknesses, vulnerabilities, failure symptoms, defects and improvement targets:
  - Encapsulate - to hide/isolate details (create APIs and containerization)
  - Rehost – to move systems off the mainframe and migrate them to new hardware or the cloud
  - Replatform - to speed up with new hardware
  - Repair – to fix the bugs, maintain
  - Refactor - to reduce technical debt and future failures
  - Rearchitect - to adapt to a new platform or technology
  - Rebuild – to fine tune it
  - Replace – with new or SaaS solution – the most difficult choice
- Create Software Bill of Materials for all software in network-connected assets to enable resilient response to changing threat environment

Recommendations for Addressing CPSQ

- Use relevant standards such as ISO/IEC 25010 ‘Software Product Quality’
- Use OMG Automated Source Code Quality Measures from CISQ to provide a model to guide process improvements and automate quality in workflows using tools with ‘quality domain checkers under the hood’

Security	Measures 74 CWEs in source code representing the most exploited security weaknesses in software including the CWE/Sans Institute Top 25 Most Dangerous Security Errors and OWASP Top 10
Reliability	Measures 74 CWEs in source code impacting the availability, fault tolerance, and recoverability of software
Performance Efficiency	Measures 18 CWEs in source code impacting response time and utilization of processor, memory, and other resources
Maintainability	Measures 29 CWEs in source code impacting the comprehensibility, changeability, testability, and scalability of software
Data Protection	Measures 89 CWEs in source code impacting data leakage or data corruption (potential vectors that could enable unauthorized reading or modification of data)

Recommendations for Addressing CPSQ

- Employ such practices as:
  - prioritizing needs and requirements
  - controlling scope changes
  - minimizing complexity
  - using systems engineering discipline
  - assigning quality champions on the team
  - planning for defect fixing and refactoring
  - establishing rigorous quality gates
  - testing early and often
  - rigorous analysis of included components
  - investing in quality engineering tools
  - focusing on improving good teamwork and effective communication

- Test all software prior to release for use, including components available as open source and external dependencies, such as libraries