Expecting Secure, High-Quality Software: Mitigating Risks throughout the Lifecycle by Reducing Attack Vectors

Joe Jarzombek, CSSLP, PMP
Director for Government, Aerospace & Defense Programs

Previously
Global Manager, Software Supply Chain Solutions
Synopsys Software Integrity Group
Director, Software and Supply Chain Assurance
U.S. Department of Homeland Security
Deputy Director, Information Assurance (Software Assurance)
        OCIO, U.S. Department of Defense
        USAF LtCol (Retired)

Joe.Jarzombek@synopsys.com
+1 (703) 627-4644


10 Sep 2018
What is the Consortium for IT Software Quality?

CISQ is chartered to define automatable measures of software size and quality that can be measured in the source code, and promote them to become Approved Specifications of the OMG®.
Synopsys Today: From Silicon to Software

FY17 Revenue: ~$2.7B

Employees: >11,600

Years: 30+

- #1 EAD electronic design automation tools & services
- Brodest IP portfolio and #1 interface, analog, embedded memories & physical IP
- ‘Leader’ in Gartner’s Magic Quadrant for application security testing
- ‘Engineer Driven Culture’ Over half our employees have advance degrees
Security and quality are in our DNA

Software Integrity Group (SIG)

Acquired 2015
Team and technology that found Heartbleed

codenomiicon

Acquired 2017
The authority on open source security and risk management

BLA**KDUCK**

FORCHECK

Acquired 2016
400+ security experts
Full portfolio of managed and professional services

codiscope
Cigital

Coverity™

Acquired 2014
The leading static analysis solution for security and quality

coverity

kali stick V

Seeker
Gaining confidence in ICT/IoT software-based technologies

- Dependencies on software-reliant Information Communications Technology (ICT) and IoT devices are greater than ever.
- Possibility of disruption is likely because software is vulnerable and exploitable.
- Loss of confidence alone can lead to stakeholder actions that disrupt critical business activities.
An ever-more connected world . . .

Organizations expanding their IoT efforts need comprehensive software security initiatives to address weaknesses resulting from technological vulnerabilities, a lack of ‘cyber hygiene’ and caution among those who develop and use IoT devices.
Cyber Risks and Consequences in IoT Solutions
Creating More Attack Vectors

- **Edge Devices** (including Applications, Sensors, Actuators, Gateways & Aggregation)
  - Device Impersonation, Counterfeiting & Hacking
  - Snooping, Tampering, Disruption, Damage

- **IoT Platform** (Data Ingestion/Analytics, Policy/Orchestration, Device/Platform Mgmt)
  - Platform Hacking
  - Data Snooping & Tampering
  - Sabotaging Automation & Devices

- **Enterprise** (Business/Mission Applications, Business Processes, etc)
  - Business/Mission Disruption
  - Espionage & Fraud / Financial Waste

If you cannot afford to protect IoT; then you cannot afford to connect it -
Cost of recovering from exploitation far exceeds costs to protect IoT
Growing Concern with Internet of Things (IoT)

• Lax security without liability for the growing number of IoT embedded devices in appliances, industrial applications, vehicles, smart homes, smart cities, healthcare, medical devices, etc.
  – Sloppy manufacturing ‘hygiene’ is compromising privacy, safety and security – incurring risks for faster time to market
  – IoT risks provide more source vectors for financial exploitation
  – IoT risks include virtual harm to physical harm
    – Cyber exploitation with physical consequences;
    – Increased risk of bodily harm from hacked devices

• Growing dependence on external third-party supplier put users and enterprises more at risk due to exploitable software
Safety/Security Risks with IOT Embedded Systems

Engineering Community concerns:

- Poorly designed embedded devices can kill;
- Security is not taken seriously enough;
- Proactive techniques for increasing safety and security are used less often than they should be.

Barr Group: “Industry is not taking safety & security seriously enough”

Based on results of survey of more than 2400 engineers worldwide to better understand the state of safety- and security-aware embedded systems design around the world (Feb 2016).
Shifting Business Concerns: Increased Software Liability

1980’s: Standalone Software Apps - Quality
1990’s: Internet & WWW - Quality / Security
2000’s: Software Controlled Devices - Quality / Security / Safety & Privacy
2010’s: Financial Liability

© 2017 Synopsys, Inc.
Software Supply Chain Management

Enabling Enterprise Control of Risks Attributable to Exploitable Software
Risk Management (Enterprise ↔ Project): Shared Processes & Practices ↔ Different Focuses

• Enterprise-Level:
  – Regulatory compliance
  – Changing threat environment
  – Business Case

• Program/Project-Level:
  – Cost
  – Schedule
  – Performance

Who makes risk decisions?
Who determines ‘fitness for use’ for ‘technically acceptable’ criteria?
Who “owns” residual risk from tainted/counterfeit products?

* “Tainted” products are those that are corrupted with malware, or exploitable weaknesses & vulnerabilities
IoT supply chain risk management
Mitigating 3rd-party risks attributable to exploitable software in IoT devices

**Increased risk from supply chain due to:**

- **Increasing dependence on globally sourced devices**
  - Varying levels of development/outsourcing controls
  - Lack of transparency in process chain of custody
  - Varying levels of acquisition ‘due-diligence’

- **Residual risk from tainted components**
  - Tainted products with malware, exploitable weaknesses (CWEs) and vulnerabilities (CVEs)
  - Defective and unauthentic/counterfeit products

- **Growing technological sophistication among adversaries**
  - Internet enables adversaries to probe, penetrate, & attack remotely
  - Supply chain attacks can exploit products and processes

Software in the supply chain is often the vector of attack.
Majority of Breaches Attributable to Exploitable Software

Data Breaches make headlines – the cause of them rarely do

✓ 84% of breaches originate at the application layer (CMU SEI, 2018)
✓ 84% of all cyber-attacks happen on the application layer (SAP)
✓ Over 70% of security breaches happen at the Application (Gartner)
✓ 92% of vulnerabilities are in application layer (NIST)
✓ Up to 80% of Data Breaches originate in the Supply Chain (SANs Institute)
✓ More than 80% of Enterprises depend on third-party code (Gartner)
✓ 90% of a typical application is comprised of third-party / OSS components (SANS)
✓ Most developers lack sufficient security training (Gartner)
✓ Web Application Attacks are the #1 source of data breaches (Verizon DBIR)

Data breaches exploit vulnerabilities and weaknesses in applications -- root causes in unsecure software -- this is a supply chain issue
Unmitigated Software Vulnerabilities and Weaknesses: Example of root causes/attack vectors for exploitation

1. Social lure typically referring to a recent event in the news
2. Exploit link in the e-mail
3. Fingerprinting script by exploit kit
4. Flash exploit (CVE-2016-7855)
5. Windows privilege escalation exploit (CVE-2016-7255)
6. Payload
Trustworthiness of an Industrial IoT System

Trustworthiness (in any IIoT device) is degree of confidence one has that the system performs as expected in respect:

- to all the key system characteristics (associated with safety, resilience, reliability, security, and privacy)
- in the face of environmental disruptions, human errors, system faults and attacks.

Source Definition: Industrial Internet Consortium (IIC) Industrial Internet of Things (IIoT) Security Framework
Enterprises Have Used Reactive Technologies to Defend…

They are good; designed for known threats. What about broader risks to enterprises and users?

Enterprises cannot stop the threats; yet can control their attack vectors/surfaces.
Exploitable Software Weaknesses (CWEs) are exploit targets/vectors for future Zero-Day Attacks catalogued as Vulnerabilities (CVEs).
Exploits, Weaknesses, Vulnerabilities & Exposures

- **The existence of an exploit designed to take advantage of a weakness (or multiple weaknesses) and achieve a negative technical impact is what makes a weakness a vulnerability.**

- **Weakness:** mistake or flaw condition in ICT/IoT architecture, design, code, or process that, if left unaddressed, could under the proper conditions contribute to a cyber-enabled capability being vulnerable to exploitation; represents potential source vectors for zero-day exploits -- Common Weakness Enumeration (CWE) [https://cwe.mitre.org/](https://cwe.mitre.org/)

- **Vulnerability:** mistake in software that can be directly used by a hacker to gain access to a system or network; **Exposure:** configuration issue of a mistake in logic that allows unauthorized access or exploitation -- Common Vulnerability and Exposure (CVE) [https://cve.mitre.org/](https://cve.mitre.org/)

- **Exploit:** action that takes advantage of weakness(es) to achieve a negative technical impact -- attack approaches from the set of known exploits are used in the Common Attack Pattern Enumeration and Classification (CAPEC) [https://capec.mitre.org/](https://capec.mitre.org/)

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
CVE & CWE Can Be Used to Assess Software Maturity

- Are the commercial and open source applications being used as part of the system, the development environment, the test environment, and the maintenance environment to detect CWEs/CVEs and patched for known CVEs?
- Are any components/libraries incorporated in the system that have CVEs?
- Have pen testing tools/teams found any CVEs?
- Does the project team monitor for Advisories?
- Do projects utilize CVSS/CWSS scores to prioritize remediation efforts?
- Is the use of CWE and CVE Identifiers and public advisories a consideration when selecting commercial and open source applications?

CVE & CWE are some of the means for sharing information about risk exposures in software supply chain management
Products on “Whitelisted” Approved Products List or “Assessed & Cleared” Products List should be Tested for:

- **Exploitable Weaknesses** (CWEs, ITU-T X.1524)
  - If suppliers do not mitigate exploitable weaknesses or flaws in products (which are difficult for users to mitigate), then those weaknesses represent vectors of future of exploitation and ‘zero day’ vulnerabilities.

- **Known Vulnerabilities** (CVEs, ITU-T X.1520)
  - If suppliers cannot mitigate known vulnerabilities prior to delivery and use, then what level of confidence can anyone have that patching and reconfiguring will be sufficient or timely to mitigate exploitation?

- **Malware** (MAEC, ITU-T X.1546)
  - If suppliers do not check that the software they deliver does not have malware (typically signature-based), then users and using enterprises are at risk of whitelisting the malware.
Software development is more challenging every day

**New Attack Vectors**
- Embedded/IoT
- Cloud
- Mobile
- Open Source

**Shorter Product Cycles**
- Agile
- CI/CD
- DevOps
- Containers

**Increasing Complexity**
- Languages
- Tools
- Dependencies
- Supply Chain
Software Today Is Assembled

Software supply chain includes development and third-party components
Today, Up to 90% of an Application Consists of Third-Party Code
Today, up to 90% of an application consists of third-party code (Free Open Source Software or FOSS).
Do you trust what’s in your Third-Party Code?
Software Testing

Enabling Insight into Risks Attributable to Exploitable Software
Software Supply Chain Risk Management:
Testing Software & Enabling Cybersecurity Assurance for Network-Connectable Devices

Software is buggy

How many exploitable weaknesses and vulnerabilities are in your systems and devices?

Input processing can be exploited

Any software processing input can be attacked:
- network interfaces,
- device drivers,
- user interfaces, etc..

Hackers use binary analysis & fuzzing techniques to find vulnerabilities

These are used to exploit or launch attacks
These can also be discovered & mitigated by suppliers;
These should be used in test criteria for acceptance testing

Software is buggy

Input processing can be exploited

Hackers use binary analysis & fuzzing techniques to find vulnerabilities
Different techniques address different risks

<table>
<thead>
<tr>
<th>Static Analysis</th>
<th>Software Composition Analysis</th>
<th>Dynamic Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Analyzes source code</td>
<td>• Scans for open source</td>
<td>• Tests running apps</td>
</tr>
<tr>
<td>• Finds common security weaknesses (CWEs):</td>
<td>• Finds open source vulnerabilities (CVEs):</td>
<td>• Finds vulnerable app behavior:</td>
</tr>
<tr>
<td>– SQL injection</td>
<td>– Detects known vulns</td>
<td>– Misconfigurations</td>
</tr>
<tr>
<td>– Cross-site scripting</td>
<td>– Works through full SDLC</td>
<td>– Authentication issues</td>
</tr>
<tr>
<td>– Buffer overflows, etc.</td>
<td>– Monitors for new vulns</td>
<td>– Business logic flaws</td>
</tr>
</tbody>
</table>

Best for proprietary code

Best for open source

Best for running apps

Coverity

Black Duck

Defensics & Seeker
Some Prioritized Lists To Consider

- SANS CWE Top 25 – A list of top 25 most commonly encountered Cyber Weakness Enumerators (CWEs), found in (https://www.sans.org/top25-software-errors/)
Which static analysis tools and Pen Testing services find the CWEs I care about?

Utilizing a Priority List of Weaknesses

- Code Review
- Static Analysis Tool A
- Static Analysis Tool B
- Pen Testing Services

CWEs a capability
claims to cover

Most Important Weaknesses (CWEs)

Slide provided by Bob Martin, MITRE
Software Composition Analysis (SCA) is needed:

Components of Software Composition Analysis (SCA) solution:

- Vulnerability assessment and tracking
- [FOSS] license management and export compliance
- Software Bill of Materials (BOM) identification and management
SCA Provides Ingredients List (Software Bill of Materials): Resource for determining risk

Your application = Proprietary Code + Open Source Components + Application Behavior & Config

Application Facts

<table>
<thead>
<tr>
<th>Code Label</th>
<th>% Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protex</td>
<td></td>
</tr>
<tr>
<td>Code Base: 20.452GB</td>
<td></td>
</tr>
<tr>
<td>Total Open Source: 4.947GB</td>
<td></td>
</tr>
<tr>
<td>Reciprocal as Components: 3.915GB</td>
<td>24.19%</td>
</tr>
<tr>
<td>Reciprocal as Files: 0.252GB</td>
<td>19.14%</td>
</tr>
<tr>
<td>Permissive: 0.003GB</td>
<td>1.23%</td>
</tr>
<tr>
<td>Owned: 0MB</td>
<td>0%</td>
</tr>
<tr>
<td>Total Proprietary: 11.335GB</td>
<td></td>
</tr>
<tr>
<td>Licensed 3rd Party: 0.354GB</td>
<td>1.73%</td>
</tr>
<tr>
<td>Owned: 10.981GB</td>
<td>53.69%</td>
</tr>
<tr>
<td>Total Unknown: 0MB</td>
<td>0%</td>
</tr>
</tbody>
</table>

Code Genetics
Comprehensive Software Composition Analysis (SCA)

Scan and Report Components with Known Security Vulnerabilities

Detect and manage 3rd party and open source components or portions thereof

The versatility and breadth of this solution makes it viable for many use cases and appealing to many personas

Ensure Licensing, IP, and Export Control Compliance
Total Economic Impact of Synopsys Software Testing Tools
Forrester Case Study – Useful Framework

Using Coverity and Defensics in the development lifecycle…

• Improved product quality and security
  – Avoided remediation expenses in 8 code bases of 1.5M LoC each; saving $3.86M (NPV)
  – Lowered defect density within its code base… prevented future costs of allowing error-prone code to be reused.

• Reduced time to market
  – Using fuzz testing and static analysis, reduced product release cycle from 12 to 8 months; enabling company to redirect resources toward other productive activities.
  – Decreased time to detect and remediate defects/vulnerabilities;

• Prevented high-profile breaches
  – Lowered future risk exposure attributable to exploitable software

• Mitigated costly post-deployment malfunctions
  – Required 2 times fewer labor hours than in post-release phase

Numerical Data
ROI: 136% // Total NPV: $5.46m
Cost to find & fix bugs: ↓2x-10x
Time to release new products: ↓4mo

The Synopsys Software Integrity Toolbelt
Everything you need to build security and quality into your SDLC

**STAGE 1: PLAN**

**TRAINING**

**RECOMMENDED TOOLS:**
- BSIMM: What is the current state of your software security?
- MAP: How can you improve your software security?
- Test Optimization: Have tests been prioritized and created?

**STAGE 2: DESIGN**

**TRAINING**

**RECOMMENDED TOOLS:**
- Architecture Risk Analysis: Have secure design best practices been implemented?
- Threat Modeling: How can an attacker exploit your design?
- Security Control Design Analysis: Are there any missing or weak security controls?

**STAGE 3: IMPLEMENT**

**TRAINING**

**RECOMMENDED TOOLS:**
- SAST: Are you finding and fixing bugs while you code?
- Software Composition Analysis (SCA): Are there any bugs in the OSS or third-party software?
- IAST: Are you able to identify real business threats?
- Fuzz Test: Can each code component properly handle malformed inputs and unexpected loads?

**STAGE 4: VERIFY**

**TRAINING**

**RECOMMENDED TOOLS:**
- SAST / SCA: Are there any remaining bugs in custom code or OSS/third-party software?
- IAST: Can you automate IAST testing into your CI/CD processes?
- DAST: Are you able to dynamically test your app before release?
- Fuzz Test: Is the integrated code able to properly handle malformed inputs?
- Pen Test: Can you manually hack your app before it goes live?

**STAGE 5: RELEASE**

**TRAINING**

**RECOMMENDED TOOLS:**
- DAST: Are new bugs uncovered when the app is running?
- Network Pen Test: Is the network configuration secure?
- Pen Test: What impact could an attacker have?

**STAGE 6: RESPOND**

**TRAINING**

**RECOMMENDED TOOLS:**
- Insider Threat Detection: How easily can you be breached?
- Red Team: Are you set to prevent and respond to incidents?
- Threat Intelligence: Is your organization monitoring for known breaches?
Software supply chain risk management

Procurement requirements, independent testing and certification
Deliver Uncompromised: A Strategy for Supply Chain Security and Resilience in Response to the Changing Character of War

US Department of Defense and Intelligence Community’s focus to “Deliver Uncompromised” adds Security of products and services to existing contract evaluation criteria of Cost, Schedule and Performance.

Its roots in the MITRE report “Deliver Uncompromised: A Strategy for Supply Chain Security and Resilience in Response to the Changing Character of War” are driving changes in the law, as well as acquisition and procurement policies and practices.
Deliver Uncompromised: A Strategy for Supply Chain Security and Resilience in Response to the Changing Character of War

• “Software assurance needs to be made a priority for all phases of system acquisition and sustainment. DoD needs to work closely with technical community industrial partners to demonstrate and deploy new methods and measures to identify and respond to software vulnerabilities. Such initiatives acquire new urgency as more and more systems become interdependent and reliant upon the growing instrumentalities of the Internet of Things (IoT).”

• “Address the full span of software vulnerability through measures in acquisition and operations through full life cycle continuous security and risk reduction practices from concept through retirement. Determine where and for what programs or missions it is recommended or necessary to require submission of a Software Bill of Materials (SBOM) and require a documented Secure Software Design Life Cycle (SSDL).”

• Each Service component in both acquisition and sustainment should look for and coordinate information sharing among themselves and with designated software vulnerability information sharing mechanisms such as Common Vulnerabilities and Exposures (CVE), Information Sharing and Analysis Organizations (ISAOs), United States Computer Emergency Readiness Team (US-CERT), National Telecommunications and Information Administration (NTIA), and Department of Justice (DOJ).
Software Supply Chain Risk Management:
Proactive Control with Procurement Language for Supply Chain Cyber Assurance

Exemplar (freely available for download; used by other organizations)

https://www.synopsys.com/software-integrity/resources/white-papers/procurement-language-risk.html
Supply Chain Cyber Assurance –

Procurement Requirements

- Product Development Specification and Policy
- Security Program
- System Protection and Access Control
- Product Testing and Verification
  - Communication Robustness Testing
  - Software Composition Analysis
  - Static Source Code Analysis
  - Dynamic Runtime Analysis
  - Known Malware Analysis
  - Bill of Materials
  - Validation of Security Measures
- Deployment and Maintenance

Source: Financial Services Sector Coordinating Council for Critical Infrastructure Protection and Homeland Security
Software Security Initiatives are a Journey

- Programmatically managing risk across your software release cycles
- Driving efficiencies through SDLC integration
- Purposeful blend of automated and manual testing processes

- Augmenting internal teams with external resources for scalability
- Identify and prioritize vulnerabilities for remediation
- Integrating with DevOps

- Pen testing to find vulnerabilities
- Compliance driven
- Low level testing

© 2018 Synopsys, Inc.
Expecting Secure, High-Quality Software: Mitigating Risks throughout the Lifecycle by Reducing Attack Vectors

Joe Jarzombek, CSSLP, PMP
Director for Government, Aerospace & Defense Programs

Previously
Global Manager, Software Supply Chain Solutions
Synopsys Software Integrity Group

Director, Software and Supply Chain Assurance
U.S. Department of Homeland Security

Deputy Director, Information Assurance (Software Assurance)
OCIO, U.S. Department of Defense

USAF LtCol (Retired)

Joe.Jarzombek@synopsys.com
+1 (703) 627-4644

10 Sep 2018