Cybersecurity Technical Risk Indicators: A Measure of Technical Debt

Joe Jarzombek, CSSLP, PMP
Global Manager, Software Supply Chain Solutions
Synopsys Software Integrity Group

Previously
Director, Software and Supply Chain Assurance
U.S. Department of Homeland Security
& Deputy Director, Information Assurance
OCIO, U.S. Department of Defense

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

March 2, 2017
Cybersecurity Technical Risk Indicators: A Measure of Technical Debt

Learning Points:

1. Organizations rely on third-party software that should be tested prior to use or integration within new software.

2. Application weaknesses and vulnerabilities can be detected and mitigated in development and testing prior to use or integration in corporate assets.

3. Technical Risk Indicators, derived from ITU-T CYBEX 1500-series standards, can be used by professionals seeking to improve software quality and security.
An ever-more connected world . . .

- **People**
  - Wellness monitoring
  - Medical case management
  - Social needs

- **Goods & Services**
  - Track materials
  - Speed distribution
  - Product feedback

- **Communities**
  - Traffic status
  - Pollution alerts
  - Infrastructure checks

- **Environment**
  - Pollution checks
  - Resource status
  - Water monitoring

- **Homes**
  - Utilities control
  - Security monitoring
  - Structure integrity

- **Goods & Services**
  - Track materials
  - Speed distribution
  - Product feedback
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 evolving from virtual harm to physical harm
    - Cyber exploitation with physical consequences;
    - Increased risk of bodily harm from hacked devices
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

1990’s

2000’s

2010’s

Standalone Software Apps

Internet & WWW

Software Controlled Devices

Quality

Quality / Security

Quality / Security / Safety & Privacy

Financial Liability
Software Security Enumerations and Definitions

Enabling Standards-based Security Automation & Information Sharing
**Software Supply Chain Assurance Focus on Components**

*Mitigating risks attributable to tainted, exploitable non-conforming constructs in ICT/IoT software*

“Tainted” products are corrupted with malware, and/or exploitable weaknesses & vulnerabilities that put enterprises and users at risk

- Enable ‘scalable’ detection, reporting and mitigation of tainted software components in ICT/IoT
- Leverage related existing standardization efforts
- Leverage taxonomies, schema & structured representations with defined observables & indicators for conveying information:
  - Tainted constructs:
    - Malicious logic/malware (MAEC) ITU-T X.1546
    - Exploitable Weaknesses (CWE) ITU-T X.1524
    - Vulnerabilities (CVE) ITU-T X.1520
    - Attack Patterns (CAPEC) ITU-T X.1544
- Leverage catalogued diagnostic methods, controls, countermeasures, & mitigation practices
- Use publicly reported weaknesses and vulnerabilities with patches accessible via National Vulnerability Database (NVD) hosted by NIST

International uptake in security automation standards via ITU-T CYBEX 1500 series

© 2017 Synopsys, Inc.
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/

- **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/

- **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/
Technical Risk Indicators –
Linking Software Vulnerabilities and Weaknesses with Business Risks

• ITU-T X.1521 Common Vulnerability Scoring System (CVSS) https://www.first.org/cvss is an open framework for communicating the characteristics and severity of software vulnerabilities; providing a standardized method for rating IT vulnerabilities and determining the urgency of response.

• ITU-T X.1525 Common Weakness Scoring System (CWSS) https://cwe.mitre.org/cwss/cwss_v1.0.1.html provides a mechanism for prioritizing software weaknesses in a consistent, flexible, open manner.

• **Technical Impact:** derived from CWSS Base Finding metric group that captures the inherent risk of the weakness, a technical impact represents the potential result that can be produced by the weakness, assuming that the weakness can be successfully reached and exploited.

• **Software Security Risk:** using software with known vulnerabilities and weaknesses.

• **Technical Risk Indicator:** derived from vulnerability rating and/or weakness technical impact, an indicator of technical security risk that, if unpatched or unmitigated, could represent a source vector for attack, and if exploited could result in negative business/mission consequences.
Indicators of Software Risk – A Form of Technical Debt

Code Analysis & Bill of Materials Results with Policy Element Count
- OWASP Top 10 Issues (CWEs & CVEs)
- CWE/SANS Top 25 Issues (CWEs)
- CWE/SANS On the Cusp (26-41) Issues
- Issues with CWE IDs
- Other issues (weaknesses without IDs)
- Known Vulnerabilities (from CVEs)
- Critical Vulnerabilities (7+ on CVSS)
- Types of Licenses
- Components with unconfirmed pedigree

Technical Risk Indicators (Count of Elements,) that if left unmitigated, represent or could contribute to:
- Denial of Service
- Unauthorized Bypass of Protection Mechanism
- Unauthorized Gain of Privileges /Assumption of Identity
- Execution of Unauthorized Code or Command
- Unauthorized Alteration of Execution Logic
- Unauthorized Modification of data, files, directories or memory
- Information leakage or unauthorized reading of data, files, directories or memory
- Hiding of Activities
- Degradation of Quality
- Unexpected State or other Technical Risk

Links urgency of CVE patch and/or CWE Negative Technical Impacts with Business Risks
Software Supply Chain Management

Mitigating Risks Attributable to Exploitable Software
Increased risk from supply chain due to:

- Increasing dependence on commercial ICT/IoT for enterprise business/mission critical systems
- Increasing reliance on globally-sourced software for ICT/IoT
  - Varying levels of development/outsourcing controls
  - Lack of transparency in process chain of custody
  - Varying levels of acquisition ‘due-diligence’
- Residual risk passed to end-user enterprise
  - Defective and Unauthentic/Counterfeit products
  - Tainted products with malware, exploitable weaknesses and vulnerabilities
- Growing technological sophistication among adversaries
  - Internet enables adversaries to probe, penetrate, and attack remotely
  - Supply chain attacks can exploit products and processes throughout the lifecycle

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

Data Breaches make headlines – the cause of them rarely do

- 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 2016)

Data breaches exploit vulnerabilities and weaknesses in applications -- root causes in unsecure software

This is a Software Supply Chain Issue
Unmitigated Software Vulnerabilities and Weaknesses: Example of root causes/attack vectors for exploitation
Assurance Required for Gaining Confidence and Trust

Managing Effects of Unintentional Defects in Component or System Integrity

Managing Consequences of Unintentional Defects

Quality

Safety

Security

Managing Effects and Consequences of Attempted/Intentional Actions Targeting Exploitable Constructs, Processes & Behaviors
Have Products on “Whitelisted” Approved Products List or “Assessed & Cleared” Products List been Tested for…

• Exploitable Weaknesses (CWEs)?
  • 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)?
  • 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?
  • 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 Today Is Assembled

Software Development

Supply Chain

SW development process

Part Original

Part Third Party

SW 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) as opposed to first-party custom code.
Do you trust what’s in your Third-Party Code?
Why test your software?

• Software is buggy, only question is how many bad and exploitable known or unknown bugs are out there?

• Hackers use binary analysis and fuzzing techniques to find vulnerabilities
  • Found vulnerabilities are used to develop exploits or launch DOS attacks

• Any software processing input can be attacked: network interfaces, device drivers, user interface, etc.....
Who Should Be Testing and Why?

**Who:** All stakeholders in the supply chain

**Why:** Because all stakeholders are affected by failures in cyber security (but in different ways).

At some point, someone (usually the end user) has to validate and verify.

However, not all links in the chain are as well-suited to perform testing.
Types of Automated Tools/Testing

What They Find; How They Support Origin Analysis & Risk Management

• **Dynamic Runtime Analysis** – Finds security issues during runtime, which can be categorized as CWE’s
  – **Malformed input testing** (fuzz testing, DoS testing) – Finds zero-days and robustness issues through negative testing.
  – **Behavioral analysis** – Finds exploitable weaknesses by analyzing how the code behaves during “normal” runtime.

• **Software Composition Analysis** – Finds known vulnerabilities and categorizes them as CVE’s and other issues.

• **Static Code Analysis** – Finds defects in source code and categorizes them as CWE’s.

• **Known Malware Testing** – Finds known malware (e.g. viruses and other rogue code).

These tests can be used to enumerate CVE’s, CWE’s, and malware which can be further categorized into prioritized lists.
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

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

UL Cybersecurity Assurance Program

• UL Cybersecurity Assurance Program (UL CAP) will be **Product Oriented & Industry Specific** with these goals:
  
  ➢ Reduce software vulnerabilities
  ➢ Reduce weaknesses, minimize exploitation
  ➢ Address known malware
  ➢ Increase security awareness

• Product service offerings apply to:
  
  ➢ Connectable Products
  ➢ Products Eco-Systems
  ➢ Products System Integration
  ➢ Critical IT Infrastructure Integration
Underwriters Labs Cybersecurity Assurance Program

• Leverages use of tools in the Synopsys Software Integrity Platform
• Cybersecurity Assurance Program (CAP) in includes:
  – **Malformed Input Testing** (Fuzz Testing and DoS Testing)
    – For all externally accessible protocols
    – Also addresses application level protocols
  – **Software Composition Analysis**
    – Compiled code
    – Up to 90% of all code is third-party
  – **Malware Analysis**
    – Know malware
  – **Static Code Analysis** – Source Code
  – **Runtime Analysis** – Running code
  – **Penetration Testing** – Hands on testing
Software Risk Assessment (SwRA)
Software Composition Analysis, Static Code Analysis, and Fuzzing
Tying Software Risks to Business Risks

Conveying software weaknesses and vulnerabilities as Business Risks – and further into risk tolerance

Translating technical impacts into Technical Risk Indicators (TRIs)
### How it works – Two Parties Involved

<table>
<thead>
<tr>
<th>The Supply Chain Manager</th>
<th>The Supplier(s)</th>
</tr>
</thead>
</table>
| **Who is the Supply Chain Manager?**  
• The person responsible for managing software acceptance for their company  
• Must ensure that not only are the feature requirements met, but also the quality and security requirements are met as well | **Who is the Supplier?**  
• Either 3rd party OR internal suppliers of software to Supply Chain Managers |
| **Why should the Supply Chain Manager care?**  
• The Supply Chain Manager’s company will ship a product that contains contributions from software suppliers | **Why should the Supplier Care?**  
• They are required to meet software quality and security requirements prior to acceptance of their software by the Supply Chain Manager |
Both Supply Chain Manager and Supplier(s) work takes place on the Software Integrity Platform (SWIP).

Supply Chain Manager creates an account on SWIP

Supply Chain Manager invites its Suppliers to SWIP to have their software assessed

Supplier creates an account on SWIP – both Supply Chain Manager & Supplier are now on SWIP

Supplier securely uploads software for assessment – software accessible to supplier only

A report is generated providing insight into supplier’s software quality & security
Key Synopsys SIG Technologies combine to enable Software Risk Assessment

- **Software Composition Analysis**
  - Finds known vulnerabilities and categorizes them as CVE’s and other issues

- **Static Code Analysis**
  - Finds defects in source code and categorizes them as CWE’s

- **Fuzz Testing**
  - Finds zero-days and robustness issues through negative testing
Supplier uploads software for Assessment (securely)

Assessment generated via three key SIG technologies

Summary Audit Report provides Risk Assessment of Software – allows for informed decision making

Secure transmission and protection of IP is ensured by uploading binary representation and abstract syntax tree (AST) of the source code to the cloud via https for BOM analysis.
Software Risk Assessment Flow

* Supplier’s contributions and subsequent Software Integrity Report delivered to Supplier. Supply Chain Manager and Supplier must decide how report will be shared.
Result – Assessment of Risk Achieved

Software Risk Assessment Report
The combined results of SAST, SCA, and FUZZING provide the Enterprise with key insight into their Supplier’s Software in a report that provides an Assessment of Risk contained in Software – enabling Risk-based decisions making.
Software Risk Assessment – Output Report

Code Analysis & Bill of Materials Results with Policy Element Count

- OWASP Top 10 Issues
- CWE/SANS Top 25 Issues
- CWE/SANS On the Cusp (26-41) Issues
- Issues with CWE IDs
- Other Issues (weaknesses without IDs)
- Known Vulnerabilities
- Critical Vulnerabilities
- Types of Licenses
- Components with unconfirmed pedigree

Technical Risk Indicators (Count of Elements,) that if left unmitigated, represent or could contribute to:

- Denial of Service
- Unauthorized Bypass of Protection Mechanism
- Unauthorized Gain of Privileges /Assumption of Identity
- Execution of Unauthorized Code or Command
- Unauthorized Alteration of Execution Logic
- Unauthorized Modification of data, files, directories or memory
- Information leakage or unauthorized reading of data, files, directories or memory
- Hiding of Activities
- Degradation of Quality
- Unexpected State or other Technical Risk

Links urgency of CVE patch and/or CWE Negative Technical Impacts with Business Risks
Software Risk Assessments Aid QA & Testing: Inform Supply Chain Decisions on Quality & Security

| “Software is primarily being assembled.” | “Software Composition Analysis (SCA) provides a high level impact in security, liability and risk mitigation for its adopters; it reduces the risk introduced by inclusion of third-party components.” | “Software Supply Chain Managers can make more informed risk decisions by conducting software risk assessments prior to acceptance of software.” |

Technical Risk Indicators Enable Better Informed Risk Management throughout the Software Lifecycle
Follow-up Discussion…
Joe Jarzombek – Global Manager, Software Supply Chain Solutions
Joe.Jarzombek@synopsys.com | 703.627.4644
Synopsys Software Integrity Group | www.synopsys.com/software
Software Risk Assessment at http://software.synopsys.com/ssca

Thank You