A Hierarchical Software Quality Assurance Approach to Leveraging Existing Technologies and Building Quality Gates

November 29, 2021
Dr. Clemente Izurieta
Associate Professor of Computer Science
Software Engineering Laboratory
Montana State University
- Bozeman: Pop ~50,000
- Widely accessible outdoor recreation
- Significant industry presence
- ~17,000 students
- ~93% U.S. Citizens
- R1 Carnegie - Very High Research Activity
- 18:1 Student/Faculty Ratio
- 8 Ph.D.
- 3MS
- 1 PostDoc
- 3 undergraduates
• Quality Assurance
• Total Quality Index (TQI)
• Quality Gates
• Example models
• SEL Current Work
  • PLC and FPGAs
  • ML
  • Cloud based QA
  • Composition, stylometry and origination of software
  • Security zones and sensitive sections of source code
• ARL
ISO 25K

SOFTWARE PRODUCT QUALITY

Functional Suitability
- Functional Completeness
- Functional Correctness
- Functional Appropriateness

Performance Efficiency
- Time Behaviour
- Resource Utilization
- Capacity

Compatibility
- Co-existence
- Interoperability

Usability
- Appropriateness
- Recognizability
- Learnability
- Operability
- User Error Protection
- User Interface Aesthetics
- Accessibility

Reliability
- Maturity
- Availability
- Fault Tolerance
- Recoverability

Security
- Confidentiality
- Integrity
- Non-repudiation
- Authenticity
- Accountability

Maintainability
- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

Portability
- Adaptability
- Installability
- Replaceability
Hierarchical Software QA

Theoretical

Quality

Characteristic 1  Characteristic 2  Characteristic 3 ....... Characteristic n

Subcharacteristic 1  Subcharacteristic 2  Subcharacteristic n

Quality property 1  Quality property 2  Quality property n

Quality property 1  Quality property 2  Quality property 3 ....... Quality property n

Operational

Total Software Quality value: 0.8610

Portability value: 0.922

Maintainability value: 0.866

Security value: 0.0

Modularity value: 1.0

Redundancy value: 0.9

Documentation value: 0.83

Encryption value: 0.9

Factors

utility function F(0.67, 0.4, 0.9)
utility function F(0.67, 0.3, 0.6)
utility function F(0.67, 0.2, 0.6)
utility function F(0.67, 0.1, 0.0)

ISO/IEC 9126:2001
ISO/IEC 25010:2011
NIST 800-53/82
RMF (Risk Management Framework)

Quamoco (2012 Wagner et al.)
Qatch (2017 Miltiades et al.)
PIQUE (2020 SEL MSU)
Objective of this work:

• Improve the quality of software systems as they move through the supply chain and their corresponding development and production environments
• Deploy an innovative, operational, and affordable solution that can leverage existing vetted technologies
PIQUE is an operational platform that facilitates software quality-assurance (SQA) efforts.

PIQUE features the breakdown of quality-related characteristics and properties into a hierarchical tree-based data structure.

It is a collection of library functions, and assessment mechanisms.

Allows for model derivation and quality assessment.

Allows for integration of existing or new tools. PIQUE is tool agnostic.

Initial models already in prototype stages for Air Force/Army research.

---

Samoladas et al., 2008
Generating a Quality Gate

The PIQUE Framework is Generic and provides the basic scaffolding.

The PIQUE Model is populated with weights (from user preferences, benchmarks or ML algorithms). Each QA characteristic is also chosen to fit the context.

The instantiation of the QA model uses chosen tools that collect measurements in the context of use.
PIQUE-Bin Design

- Model structures
  - Microsoft STRIDE
  - CWE-699 View
- Tools in use:
  - CWE_Checker
  - CVE-Bin-Tool
  - Yara Rules

<table>
<thead>
<tr>
<th>Threat</th>
<th>Security Aspect</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Spoofing Identity</td>
</tr>
<tr>
<td>T</td>
<td>Tampering with data</td>
</tr>
<tr>
<td>R</td>
<td>Repudiation</td>
</tr>
<tr>
<td>I</td>
<td>Information Disclosure</td>
</tr>
<tr>
<td>D</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>E</td>
<td>Elevation of Privilege</td>
</tr>
</tbody>
</table>

STRIDE Model

Subsection of CWE-699 View

Andrew Johnson, MS
CWE-699 View Structure

- Categories in CWE-699 become Product Factors
- Unknown-Other category to capture non-mapped CVEs/Weaknesses
- Partial Example below

Andrew Johnson, MS
Microsoft STRIDE

- Microsoft’s cyber threat model
- Informs quality aspects
Tools

- Yara-rules
- CWE_Checker
- CVE-bin-tool
Model Validation

• For every binary in the benchmark, we want to know if other attributes associated with a binary influence the score produced by any tools
  • Size of binary
  • Static vs dynamic linking
  • Compiler type
  • Domain (system vs. network)
Busybox Case Study Results
Swift
Swift
PIQUE Models

- Pique-Bin (INL, DHS)
- Pique-C# (CERL Army)
- Pique-C#-Sec (CERL Army, DHS)
- Pique-Azure (DHS)
Pique-C#-Sec

Confidentiality
Integrity
Non-repudiation
Authenticity
Accountability
Availability
Authorization
Injection
Cryptography
Cookies
ViewState
Request validation
Password management
Other
Pique-C#-Sec

PIQUE-C#-Sec Model

Harrison
Quality assurance model enhancement for binary executables

PLC Aim 1: Monitor binary execution in real-time from a PLC.
- Can we feed real-time binaries from PLC into QA model?
- Can we detect execution patterns in real time under attack?

PLC Aim 2: Replace PLC processor w/ MSU’s heterogenous cores.
- Can we insert binaries into the MSU malware core using PLC development environment.
- Can we detect/defeat injected malware.

We have acquired an AutomationDirect PLC starter kit.

We discovered the AutomationDirect PLC processor is implemented on an Altera Cyclone FPGA.
Classification, clustering, and anomaly detection using graph representations of code


Clustering Results: embedding method comparison

Preliminary results suggest Graph2Vec embedding preserves similarity of various classes of code
Summary of Other Preliminary Results

**Classification:**
Promising results using 50% malware and 50% benign code in both training and test set, and a few standard classification methods with Graph2Vec embedding

**Anomaly detection:**
Promising results using standard anomaly detection methods combined with Graph2Vec representation

**Graph embedding methodology:**
Although many graph embedding methods are available, none so far have been competitive with Graph2Vec in preserving similarities and differences in control flow graphs extracted from binaries
The Cloud

• The emergence of DevOps has fueled the growth of microservices
• Benefits include: scalability, improved vulnerability isolation, decentralized data management, faster deployment, costs

However;
• Weaknesses include: consolidation of microservices into cloud instances, shared resource consumption, performance hits

Elements to investigate:
• *How do engineers know if performance degradation may be due to malicious software running concurrently on a cloud instance?*
• *Can we measure possible interference with microservices?*
• *What coding principles are microservices violating? (Elements to check: API keys, authentication, single sign on, logging, network segregation, OS)*
Improving the Software Assurance Ecosystem

1. **Measure source code quality and maturity of ICS and cloud-based software**

   Specifically:
   
   • Instantiate a series of quality gates customized to fit in various phases of the SDLC, supply chain, and the build paths suitable for continuous integration (CI) environments.
   
   • Instantiate gates to estimate quality indices for ICS, and supply chain environments. We will focus on Programmable Logic Controllers (PLCs) and Azure microservices in the cloud.
   
   • Instantiate a PIQUE-binary gate that is customized to assess compiled binaries of ICS systems in critical infrastructure.
2. **Assess the composition, stylometry and origination of software to verify that they are truthful, complete and accurate**

Specifically:

- Leverage existing tools to separate out third party libraries and utilities to understand software components
- Research BOMs and comparisons against found items
- Assess SQA of each separate component in software packages for potential vulnerabilities
## Decomposition Tools

<table>
<thead>
<tr>
<th>Tool Provider</th>
<th>Open Source</th>
<th>Intermediate Language</th>
<th>Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manticore</td>
<td>Yes</td>
<td>Unknown</td>
<td>x86, ARM</td>
</tr>
<tr>
<td>@Disco (INL)</td>
<td>Yes (with clearance)</td>
<td>Vex</td>
<td>Many</td>
</tr>
<tr>
<td>BAP (Binary Analysis Platform)</td>
<td>Yes</td>
<td>Lisp</td>
<td>x86, ARM, MIPS, PowerPC</td>
</tr>
<tr>
<td>BARF (Binary Analysis and Reverse Engineering Framework)</td>
<td>Yes</td>
<td>REIL</td>
<td>x86, ARM</td>
</tr>
<tr>
<td>FACT (Firmware Analysis and Comparison Tool)</td>
<td>Yes</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Nano Spark (Hoplite Industries)</td>
<td>No</td>
<td>N/A</td>
<td>Many</td>
</tr>
<tr>
<td>Other proprietary (if needed)</td>
<td>No</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
</tbody>
</table>
Improving the Software Assurance Ecosystem

3. Identify security zones and sensitive sections of source code

Specifically:

- Investigate operationalization of the CWE-200, “Exposure of Sensitive Information to an Unauthorized Actor” class level abstraction using PIQUE
- A targeted PIQUE model instantiation for sensitive sections will be developed (i.e., detect tokens, passwords, ids)
- Focus on Azure microservice source code sensitive sections (directory client ids, access tokens, secrets)

1: https://cwe.mitre.org/data/definitions/200.html
Decomposition of CWE-200
Potential Collaborations

Hoplite is a leading-edge cybersecurity company specializing in the mitigation of cyber risks. Founded in 2013, Hoplite Industries has developed a set of automated cyber defense capabilities and specialized AI solutions driven by cyber research at a global scale.

Cybercore brings together experts in critical infrastructure security assessments, cyber forensic analysis, threat detection and consequence-based targeting to provide real-world technical solutions and innovations that protect operational environments from an ever-evolving threat landscape.
MSU’s Applied Research Laboratory

EXPANDING MSU’S ABILITY TO PARTNER WITH INDUSTRY AND GOVERNMENT TO PROVIDE STUDENTS WITH OPPORTUNITIES IN CLASSIFIED RESEARCH
ARL FEATURES

Secure Research Facility

- 8 DOD accredited laboratory spaces (closed areas) at 1,000 square feet each
- 5 laboratory spaces built to ICD-705 (SCIF) standards - various sizes available
- Cutting edge security system
- Natural gas, clean compressed air, heating & cooling water
- Secret Internet Protocol Router (SIPR) Network access in progress
- Private loading bay with building access
- State of the art conference room
- Backup generator
- Access to MSU Engineering and Science undergraduate and graduate students
- DoE accreditation in progress
MSU’S “Big 8” RESEARCH CAPABILITIES

- Optics and Photonics
- Quantum Advanced Applied Materials
- System Engineering & Prototyping
- Information Assurance
- Cube-Satellite Platforms
- Cybersecurity
- Materials engineering and Characterization
- Experimental Mechanics and Diagnostics
MSU Research Expertise

• Reconfigurable computing, embedded systems, optics, lasers, MEMS/MOEMS, acoustics and audio, communications, power electronics
• Software engineering, software evolution, robotics, computer vision, computational geometry, scientific computing, parallel computing, artificial intelligence, machine learning, data mining, large-scale data analysis
• Design and manufacturing, systems engineering, measurement systems and experimental mechanics, composite structures
• Coherent Lidar/LADAR, Digital Holographic Imaging, Quantum Information Processing, Spatial-Spectral Holographic Microwave Photonics,
• Small satellite programs with executed flight hardware
• Materials Science and Engineering, Optical and Quantum systems, condensed matter