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

unclassified

- 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

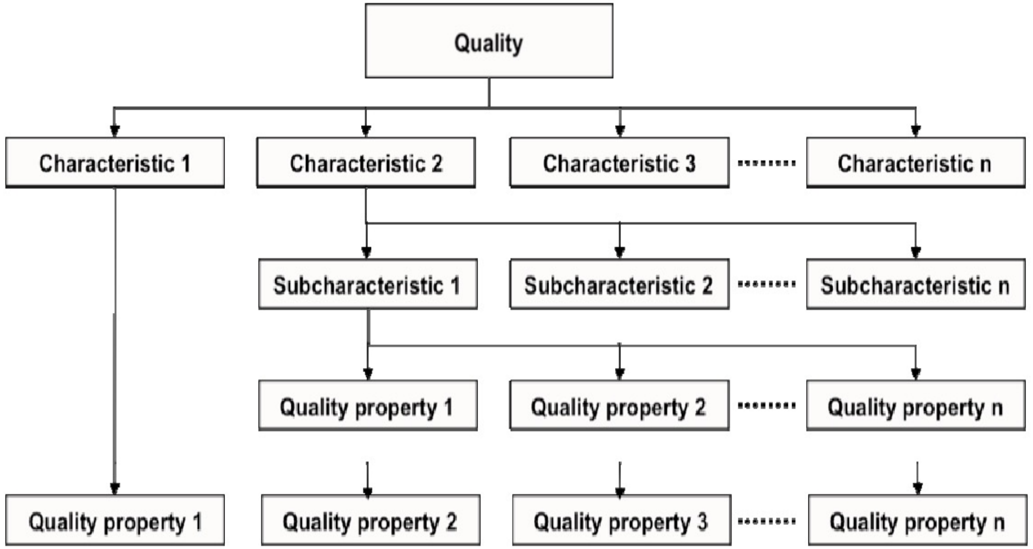
- Quality Assurance
- Total Quality Index (TQI)
- Quality Gates
- Example models
- SEL Current Work
  - PLC and FPGAs
  - ML
  - Cloud based QA
  - Composition, stylometry and origination of software
  - Security zones and sensitive sections of source code
- ARL

ISO 25K

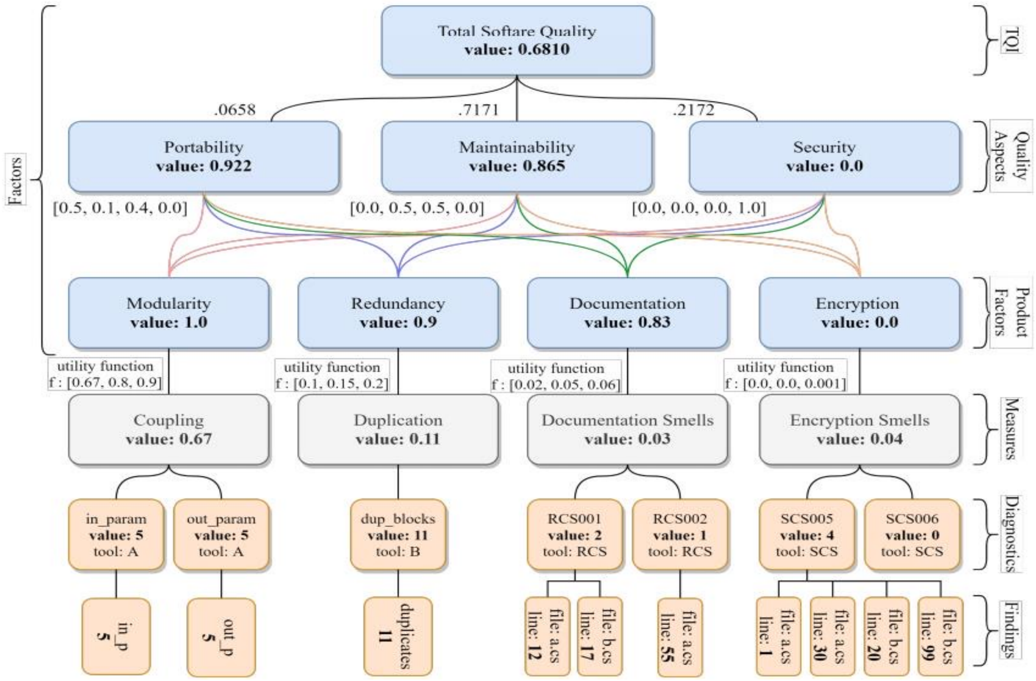

SOFTWARE PRODUCT QUALITY
Functional Suitability
• Functional Completeness
• Functional Correctness
• Functional Appropriateness
iso25000.com
Performance Efficiency
• Time Behaviour
• Resource Utilization
• Capacity
Compatibility
• Co-existence
• Interoperability
Usability
• Appropriateness Recognizability
• 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

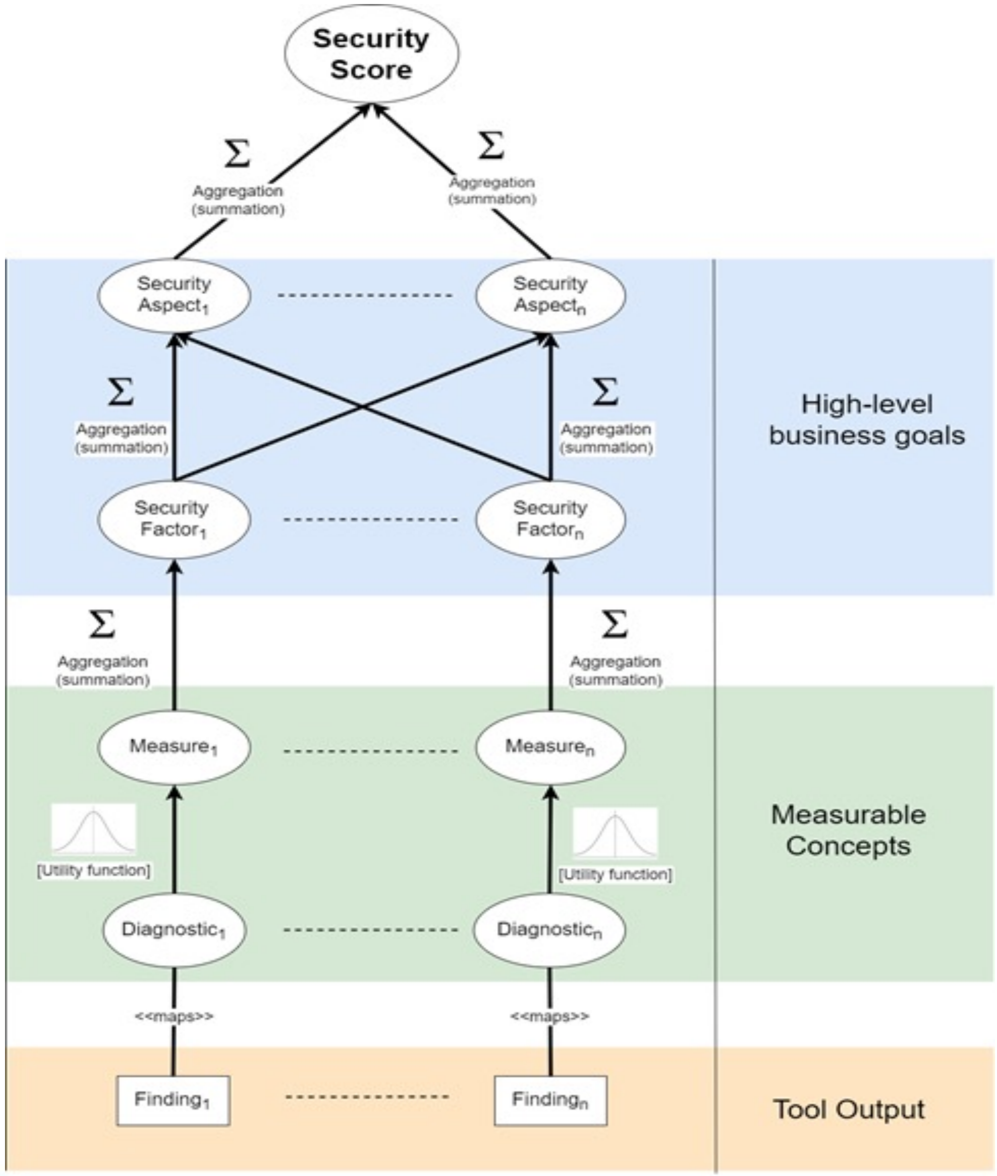


Operational

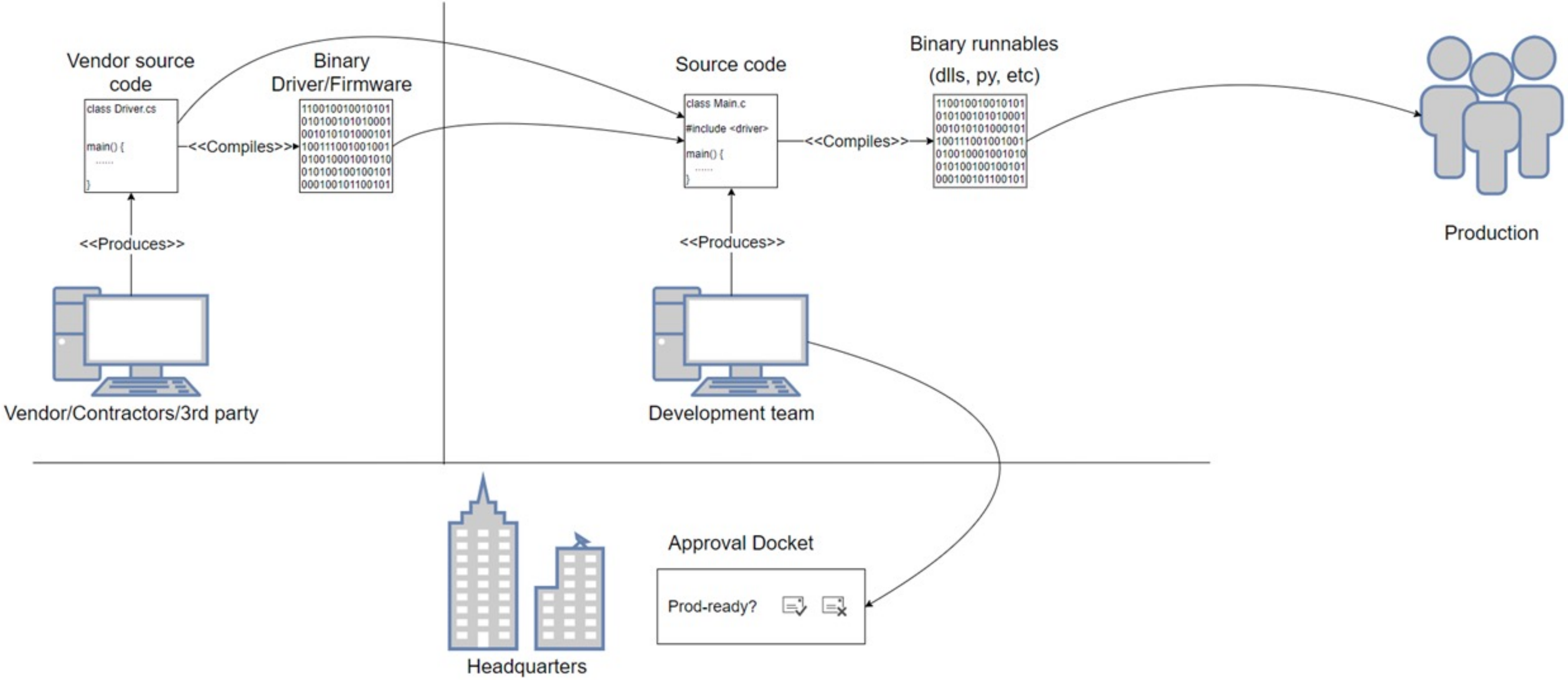


Standards
ISO/IEC 9126:2001
ISO/IEC 25010:2011
NIST 800-53/82
RMF (Risk Management Framework)

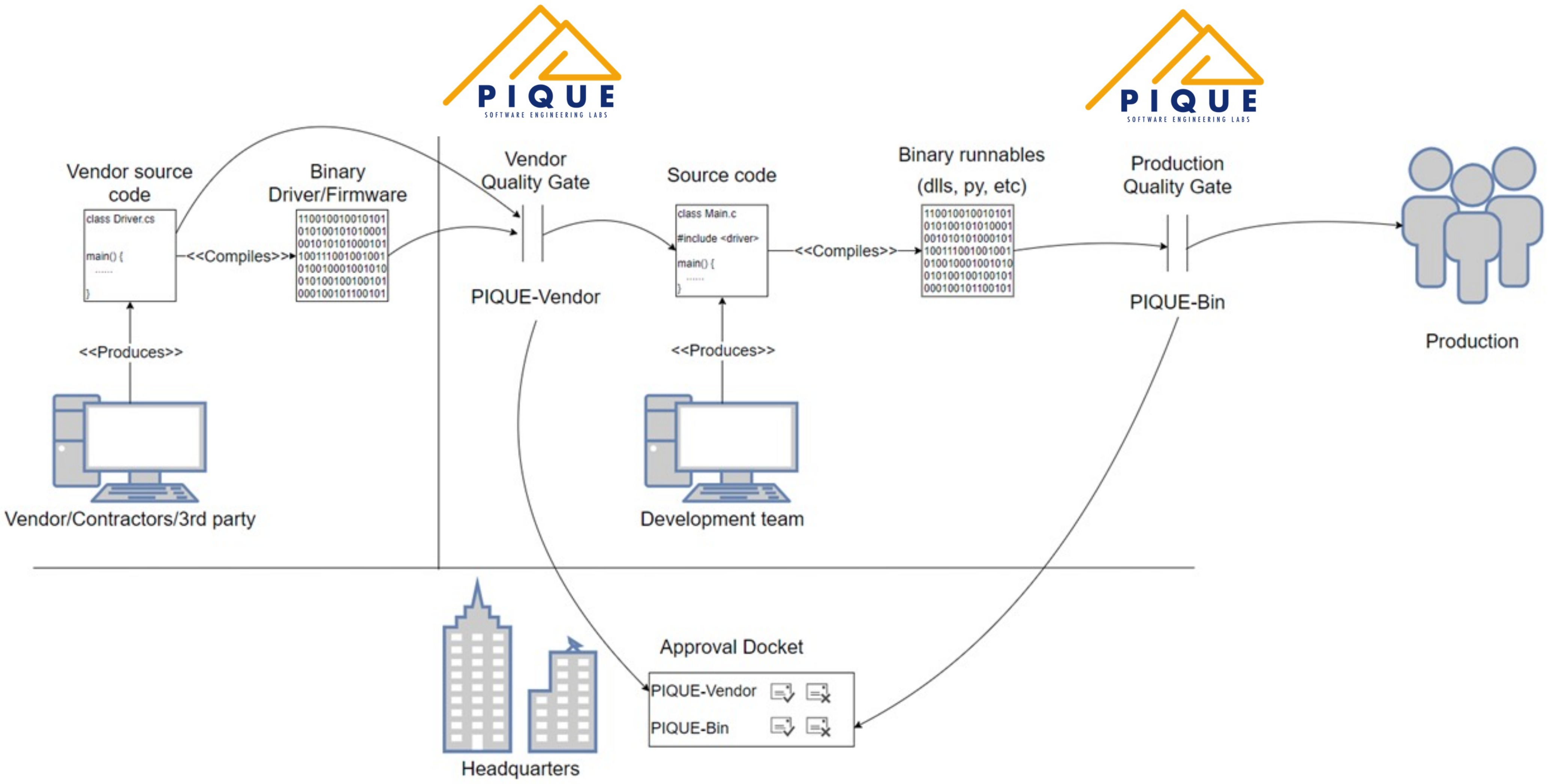
Quamoco (2012 Wagner et al.)
Qatch (2017 Miltiades et al.)
PIQUE (2020 SEL MSU)



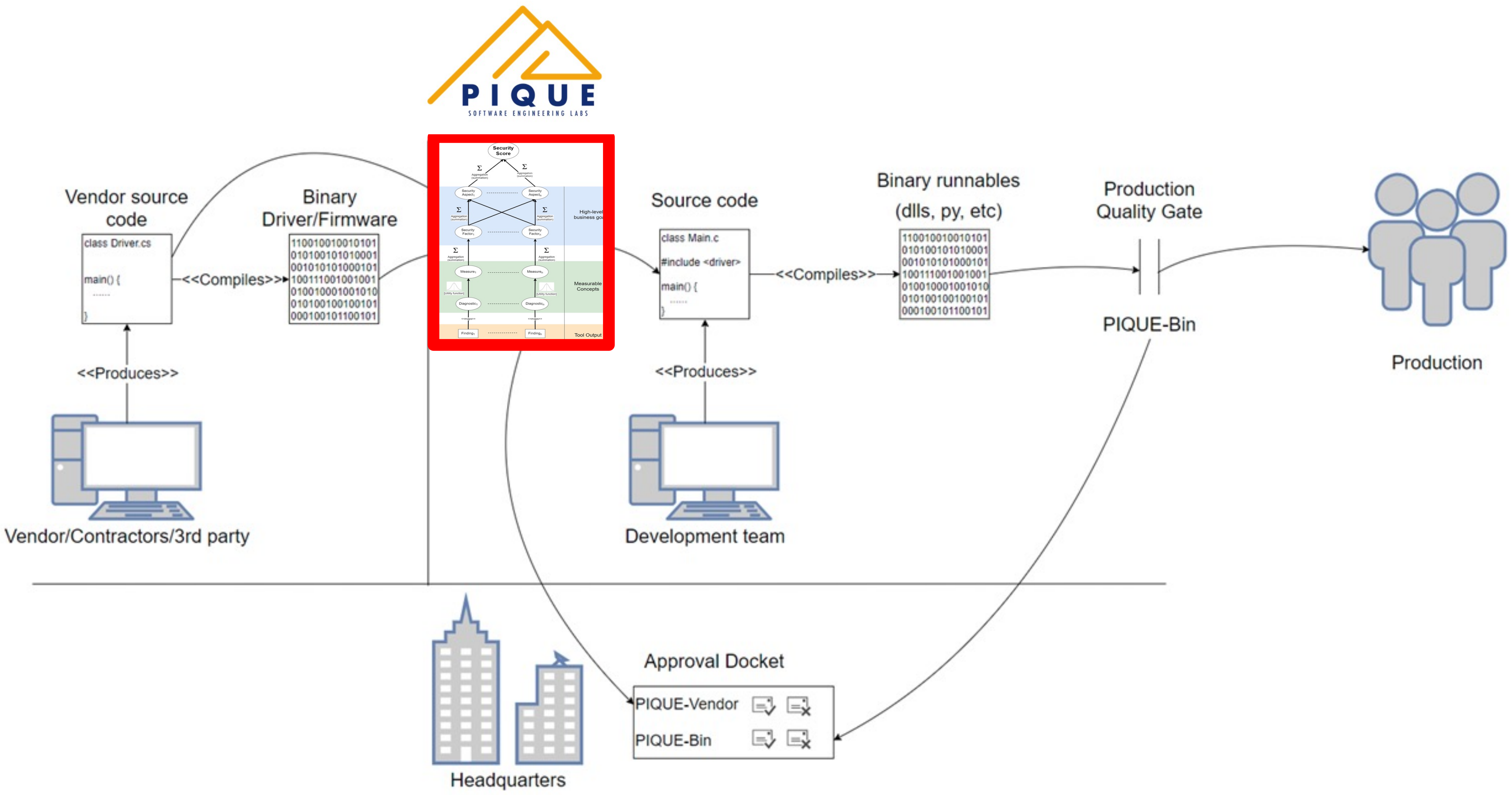
Security Score
Σ
Aggregation (summation)
Security Aspect1
Security Aspectn
Security Factor1
Security Factorn
High-level business goals
Measure1
Measuren
[Utility function]
Diagnostic1
Diagnosticn
Measurable Concepts
<<maps>>
Finding1
Findingn
Tool Output

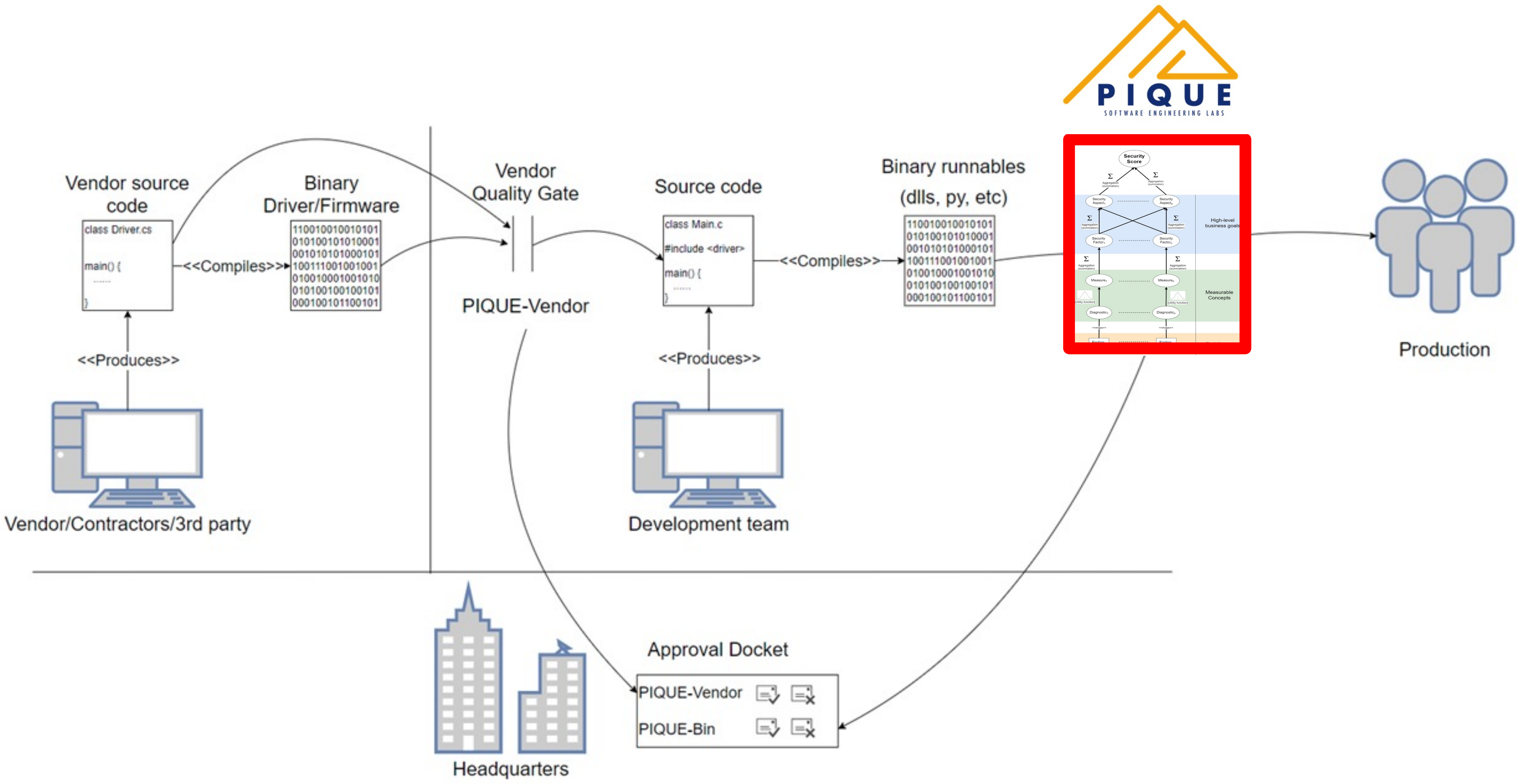


Vendor source code
class Driver.cs
main() {
......
}
Binary Driver/Firmware
110010010010101
010100101010001
001010101000101
100111001001001
010010001001010
010100100100101
000100101100101
<<Compiles>>
<<Produces>>
Vendor/Contractors/3rd party
Source code
class Main.c
#include <driver>
main() {
......
}
<<Compiles>>
Binary runnables
(dlls, py, etc)
110010010010101
010100101010001
001010101000101
100111001001001
010010001001010
010100100100101
000100101100101
<<Produces>>
Development team
Production
Headquarters
Approval Docket
Prod-ready?



PIQUE
SOFTWARE ENGINEERING LABS
PIQUE
SOFTWARE ENGINEERING LABS
Vendor source code
class Driver.cs
main() {
Binary Driver/Firmware
<<Compiles>>
<<Produces>>
Vendor/Contractors/3rd party
Vendor Quality Gate
PIQUE-Vendor
Source code
class Main.c
#include <driver>
main() {
<<Produces>>
Development team
Binary runnables (dlls, py, etc)
<<Compiles>>
Production Quality Gate
PIQUE-Bin
Production
Headquarters
Approval Docket
PIQUE-Vendor
PIQUE-Bin


PIQUE
SOFTWARE ENGINEERING LABS
Vendor source code
class Driver.cs
main() {
}
<<Compiles>>
Binary Driver/Firmware
<<Produces>>
Vendor/Contractors/3rd party
Security Score
Tool Output
Source code
class Main.c
#include <driver>
main() {
}
<<Compiles>>
Binary runnables (dlls, py, etc)
Production Quality Gate
PIQUE-Bin
Production
<<Produces>>
Development team
Headquarters
Approval Docket
PIQUE-Vendor
PIQUE-Bin



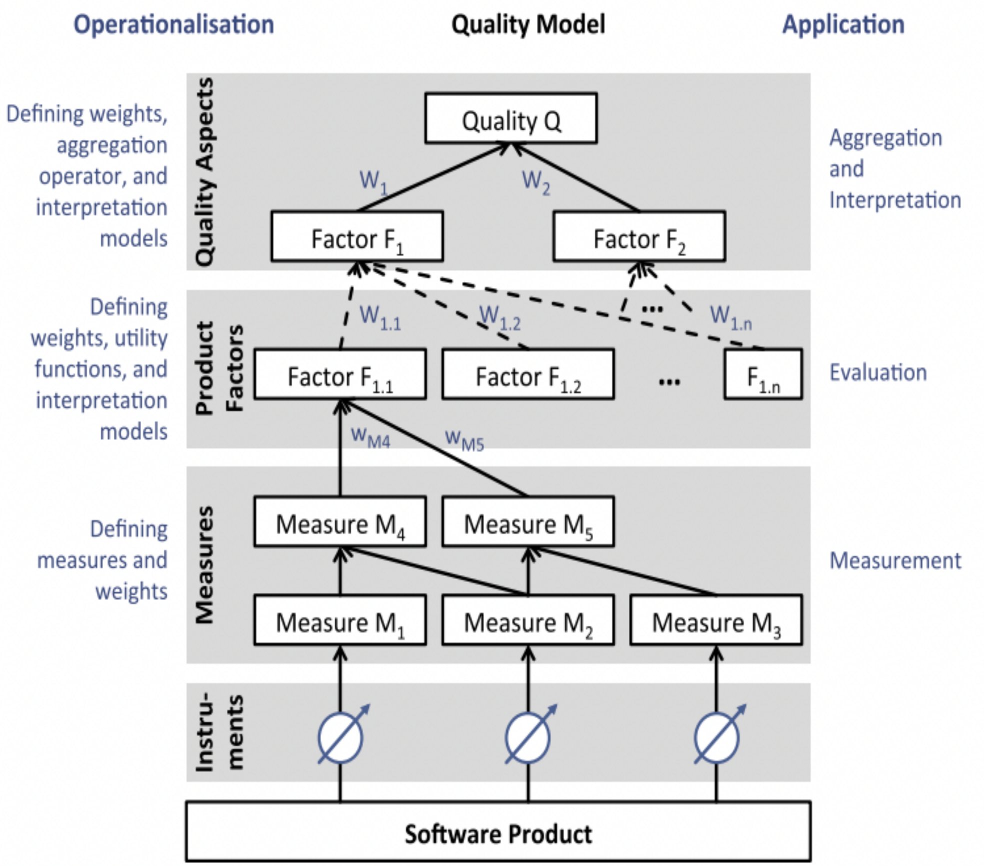
PIQUE
SOFTWARE ENGINEERING LABS
Vendor source code
class Driver.cs
main() {
Binary Driver/Firmware
<<Compiles>>
<<Produces>>
Vendor/Contractors/3rd party
Vendor Quality Gate
PIQUE-Vendor
Source code
class Main.c
#include <driver>
main() {
<<Compiles>>
<<Produces>>
Development team
Binary runnables (dlls, py, etc)
Security Score
Production
Headquarters
Approval Docket
PIQUE-Vendor
PIQUE-Bin

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

Platform for Investigative software Quality Understanding and Evaluation (PIQUE)

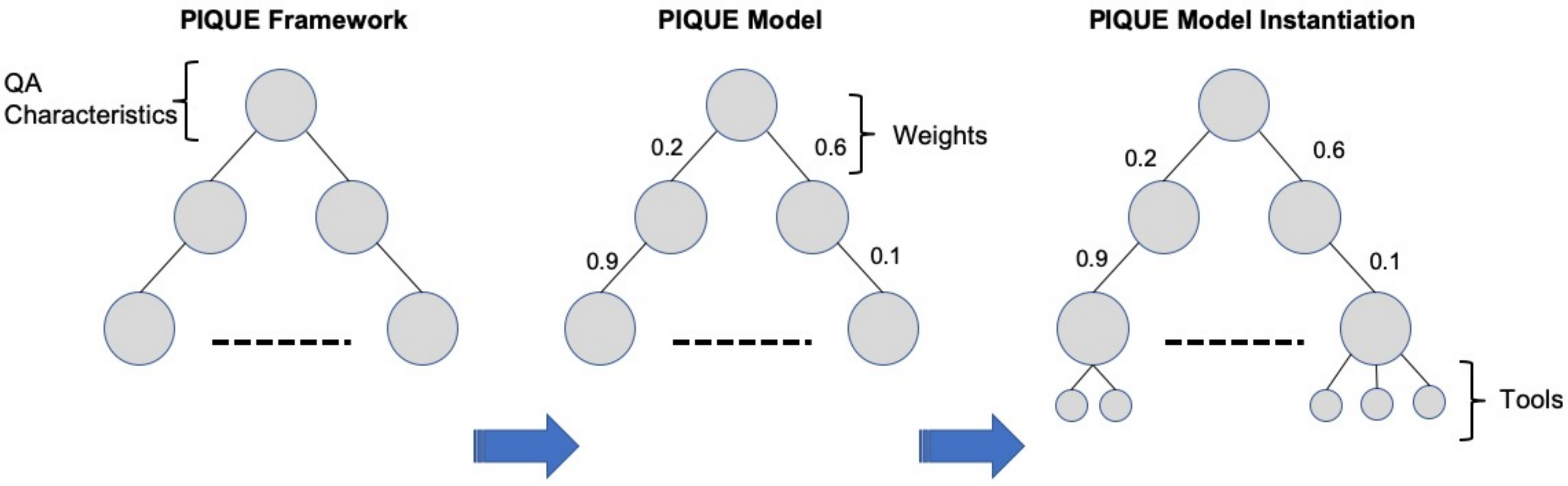
- PIQUE is an operational platform that facilitates software quality-assurance (SQA) efforts
- PIQUE features the breakdown of quality-related characteristics and properties into a hierarchical tree-based data structure
- It is a collection of library functions, and assessment mechanisms
- Allows for 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

Platform for Investigative software Quality Understanding and Evaluation (PIQUE)

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

Andrew Johnson, MS

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

STRIDE Model

	Threat	Security Aspect
S	Spoofing Identity	Authentication
T	Tampering with data	Integrity
R	Repudation	Non-repudation
I	Information Disclosure	Confidentiality
D	Denial of Service	Availability
E	Elevation of Privilege	Authorization

Subsection of CWE-699 View

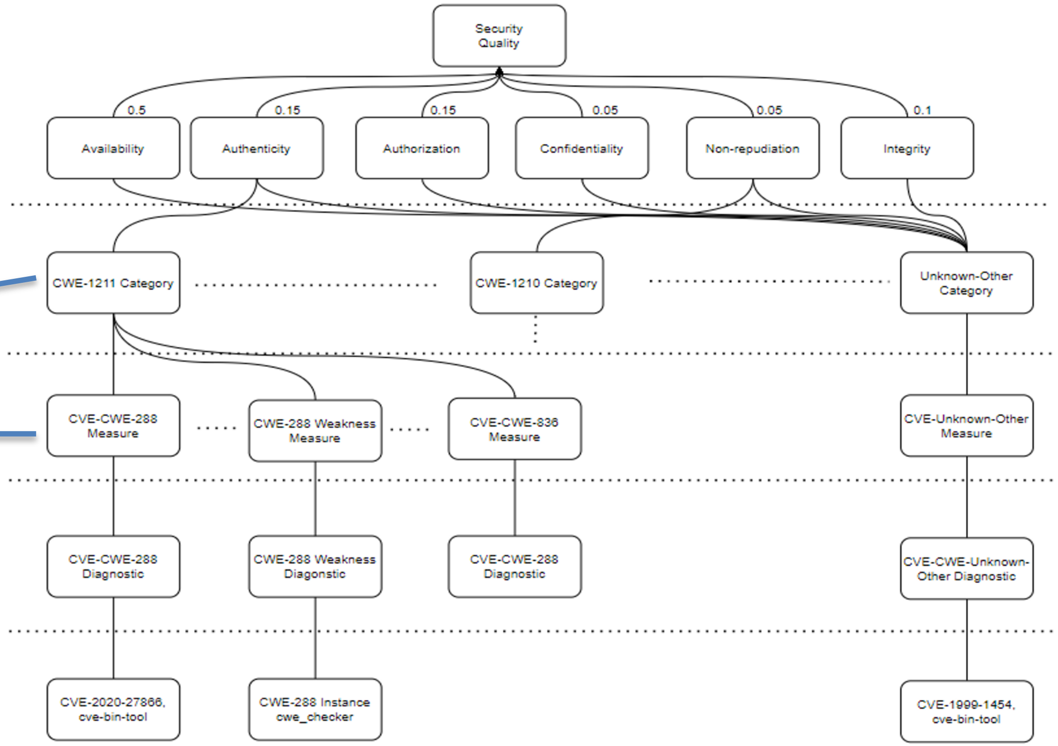
- C Audit / Logging Errors - *(1210)*
  - B Improper Output Neutralization for Logs - *(117)*
  - B Truncation of Security-relevant Information - *(222)*
  - B Omission of Security-relevant Information - *(223)*
  - B Obscured Security-relevant Information by Alternate Name - *(224)*
  - B Insertion of Sensitive Information into Log File - *(532)*
  - B Insufficient Logging - *(778)*
  - B Logging of Excessive Data - *(779)*
- C Authentication Errors - *(1211)*
  - B Authentication Bypass Using an Alternate Path or Channel - *(288)*
  - B Authentication Bypass by Spoofing - *(290)*
  - B Authentication Bypass by Capture-replay - *(294)*
  - B Improper Certificate Validation - *(295)*
  - B Improper Following of a Certificate's Chain of Trust - *(296)*
  - B Improper Check for Certificate Revocation - *(299)*
  - B Incorrect Implementation of Authentication Algorithm - *(303)*
  - B Missing Critical Step in Authentication - *(304)*
  - B Authentication Bypass by Primary Weakness - *(305)*
  - B Missing Authentication for Critical Function - *(306)*
  - B Improper Restriction of Excessive Authentication Attempts - *(307)*
  - B Use of Single-factor Authentication - *(308)*
  - B Use of Password System for Primary Authentication - *(309)*
  - B Key Exchange without Entity Authentication - *(322)*
  - B Use of Client-Side Authentication - *(603)*
  - B Overly Restrictive Account Lockout Mechanism - *(645)*
  - B Guessable CAPTCHA - *(804)*
  - B Use of Password Hash Instead of Password for Authentication - *(836)*
- C Authorization Errors - *(1212)*
  - B Direct Request ('Forced Browsing') - *(425)*
  - B Incorrect Behavior Order: Authorization Before Parsing and Canonicalization - *(551)*
  - B Improper Authorization of Index Containing Sensitive Information - *(612)*

CWE-699 View Structure

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

699 - Software Development

- C API / Function Errors - *(1228)*
  - B Use of Inherently Dangerous Function - *(242)*
  - B Use of Function with Inconsistent Implementations - *(474)*
  - B Undefined Behavior for Input to API - *(475)*
  - B Use of Obsolete Function - *(477)*
  - B Use of Potentially Dangerous Function - *(676)*
  - B Use of Low-Level Functionality - *(695)*
  - B Exposed Dangerous Method or Function - *(749)*
- C Audit / Logging Errors - *(1210)*
  - B Improper Output Neutralization for Logs - *(117)*
  - B Truncation of Security-relevant Information - *(222)*
  - B Omission of Security-relevant Information - *(223)*
  - B Obscured Security-relevant Information by Alternate Name - *(224)*
  - B Insertion of Sensitive Information into Log File - *(532)*
  - B Insufficient Logging - *(778)*
  - B Logging of Excessive Data - *(779)*
- C Authentication Errors - *(1211)*
  - B Authentication Bypass Using an Alternate Path or Channel - *(288)*
  - B Authentication Bypass by Spoofing - *(290)*
  - B Authentication Bypass by Capture-replay - *(294)*
  - B Improper Certificate Validation - *(295)*
  - B Improper Following of a Certificate's Chain of Trust - *(296)*
  - B Improper Check for Certificate Revocation - *(299)*
  - B Incorrect Implementation of Authentication Algorithm - *(303)*
  - B Missing Critical Step in Authentication - *(304)*
  - B Authentication Bypass by Primary Weakness - *(305)*
  - B Missing Authentication for Critical Function - *(306)*
  - B Improper Restriction of Excessive Authentication Attempts - *(307)*
  - B Use of Single-factor Authentication - *(308)*
  - B Use of Password System for Primary Authentication - *(309)*
  - B Key Exchange without Entity Authentication - *(322)*
  - B Use of Client-Side Authentication - *(603)*
  - B Overly Restrictive Account Lockout Mechanism - *(645)*
  - B Guessable CAPTCHA - *(804)*
  - B Use of Password Hash Instead of Password for Authentication - *(836)*



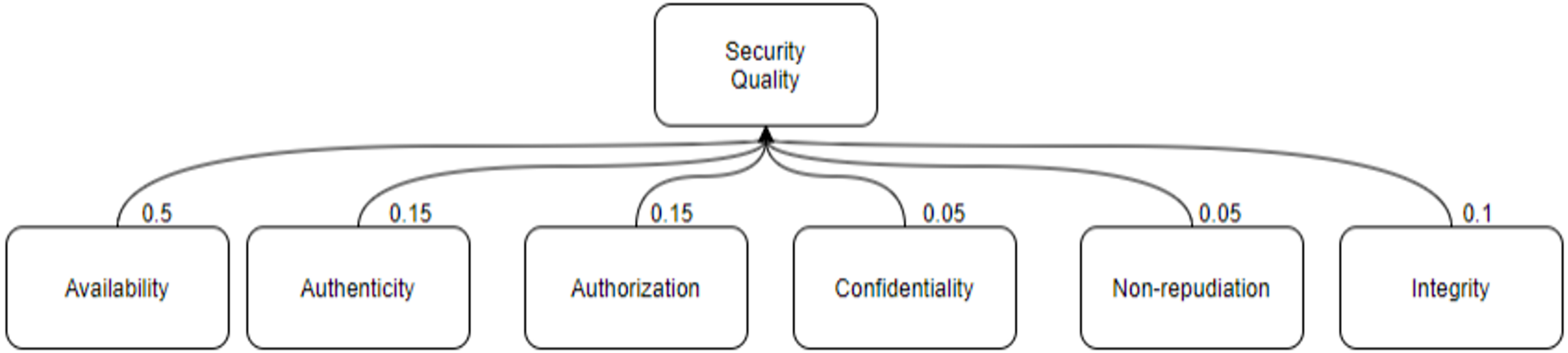
Andrew Johnson, MS

Microsoft STRIDE

Andrew Johnson, MS

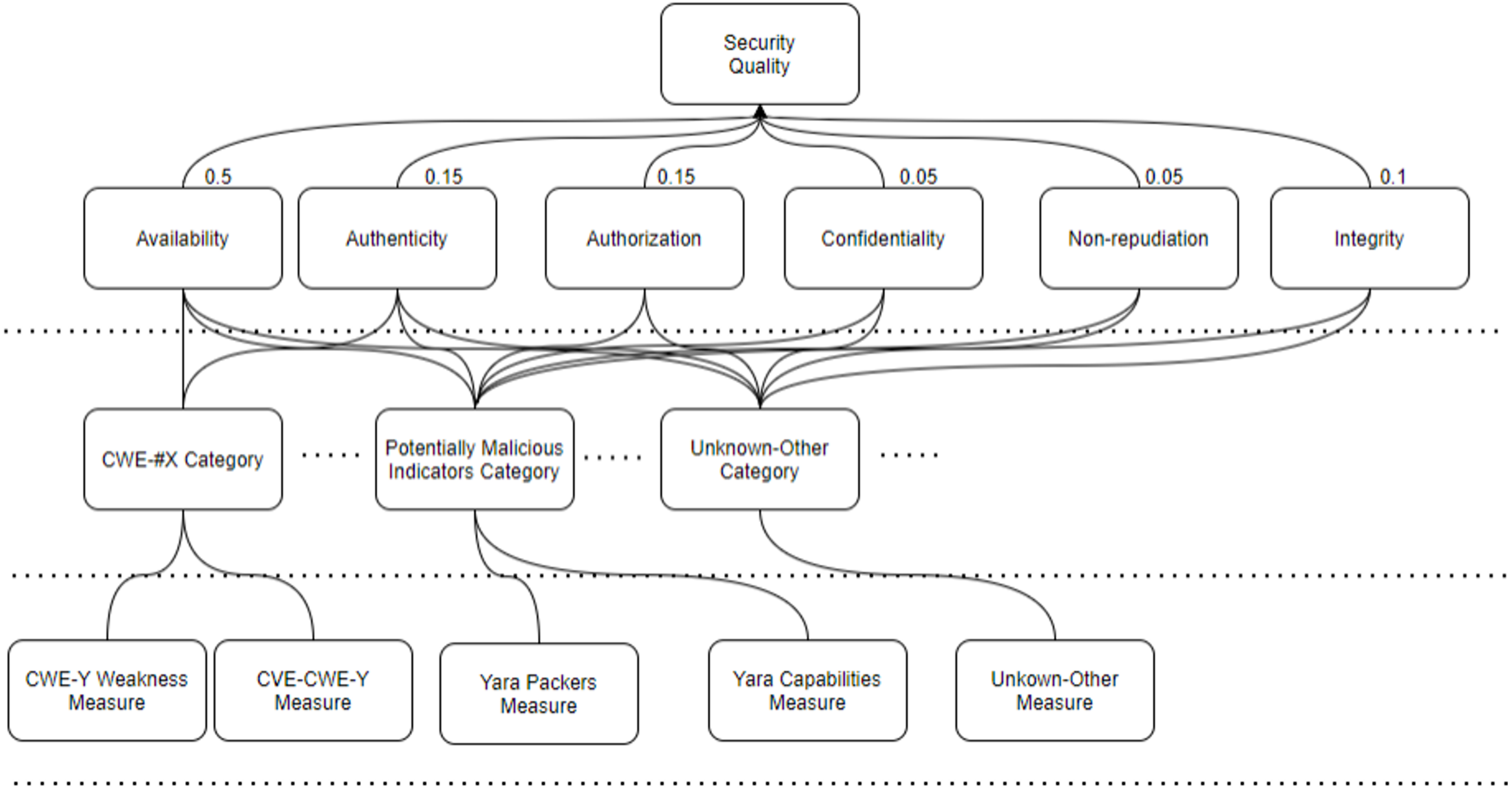
- Microsoft's cyber threat model
- Informs quality aspects

	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do



Andrew Johnson, MS

Tools



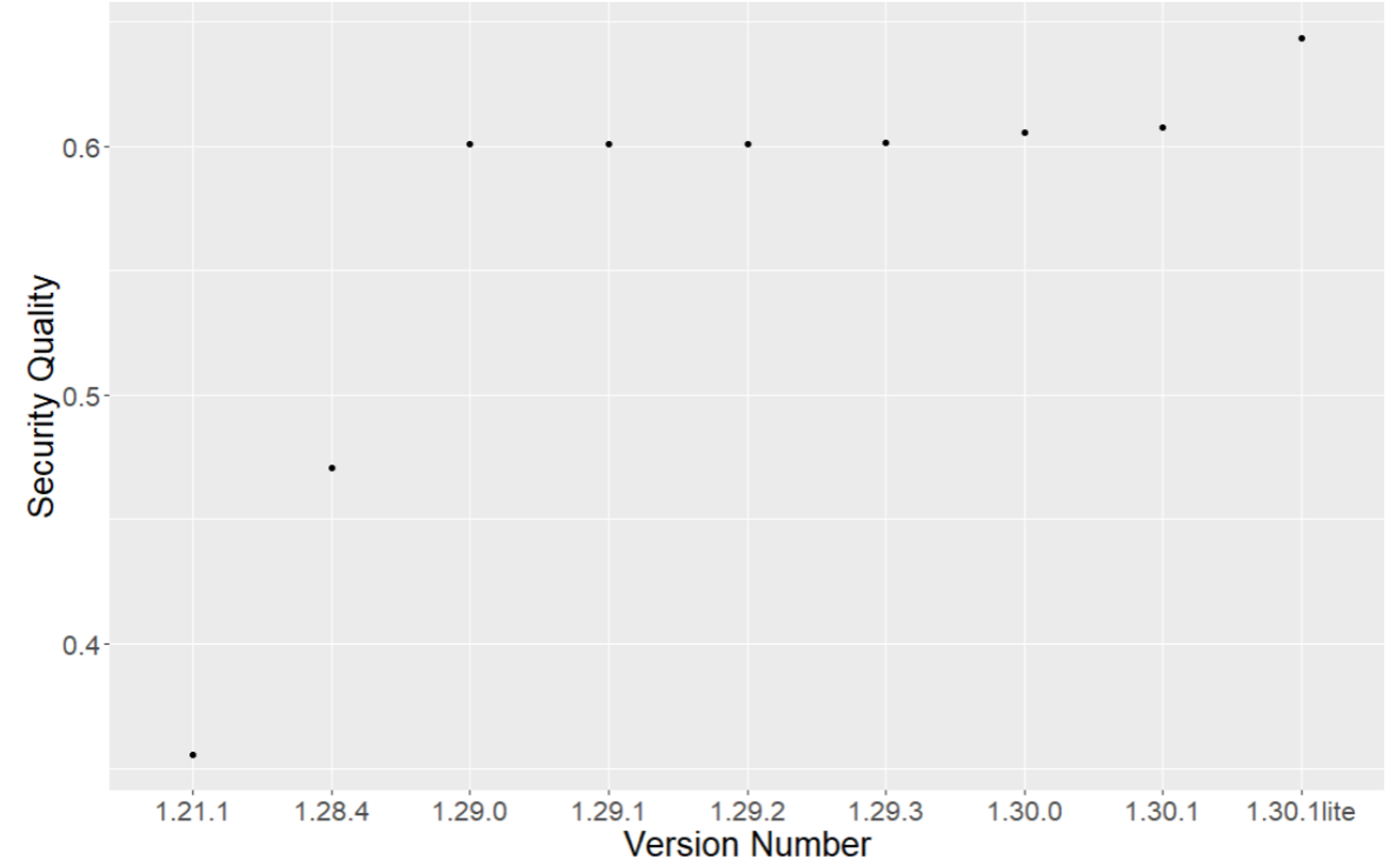
- Yara-rules
- CWE_Checker
- CVE-bin-tool

Model Validation

Andrew Johnson, MS

- 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



PIQUE

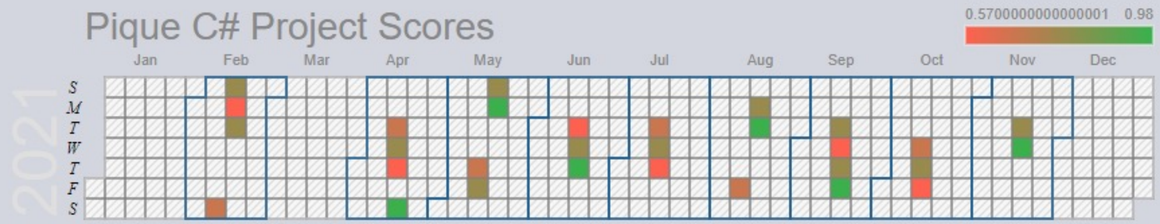
David S

Search...

sign out

- Dashboard
- Pique Tree
- Visualize
- Projects
- Settings

Pique C# Project Scores

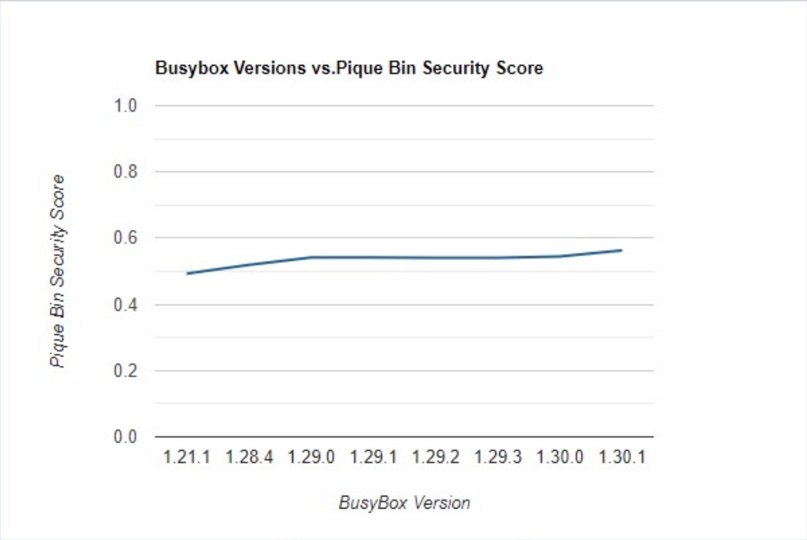


Tqi 0.3

Security 0.4

securty 0.6

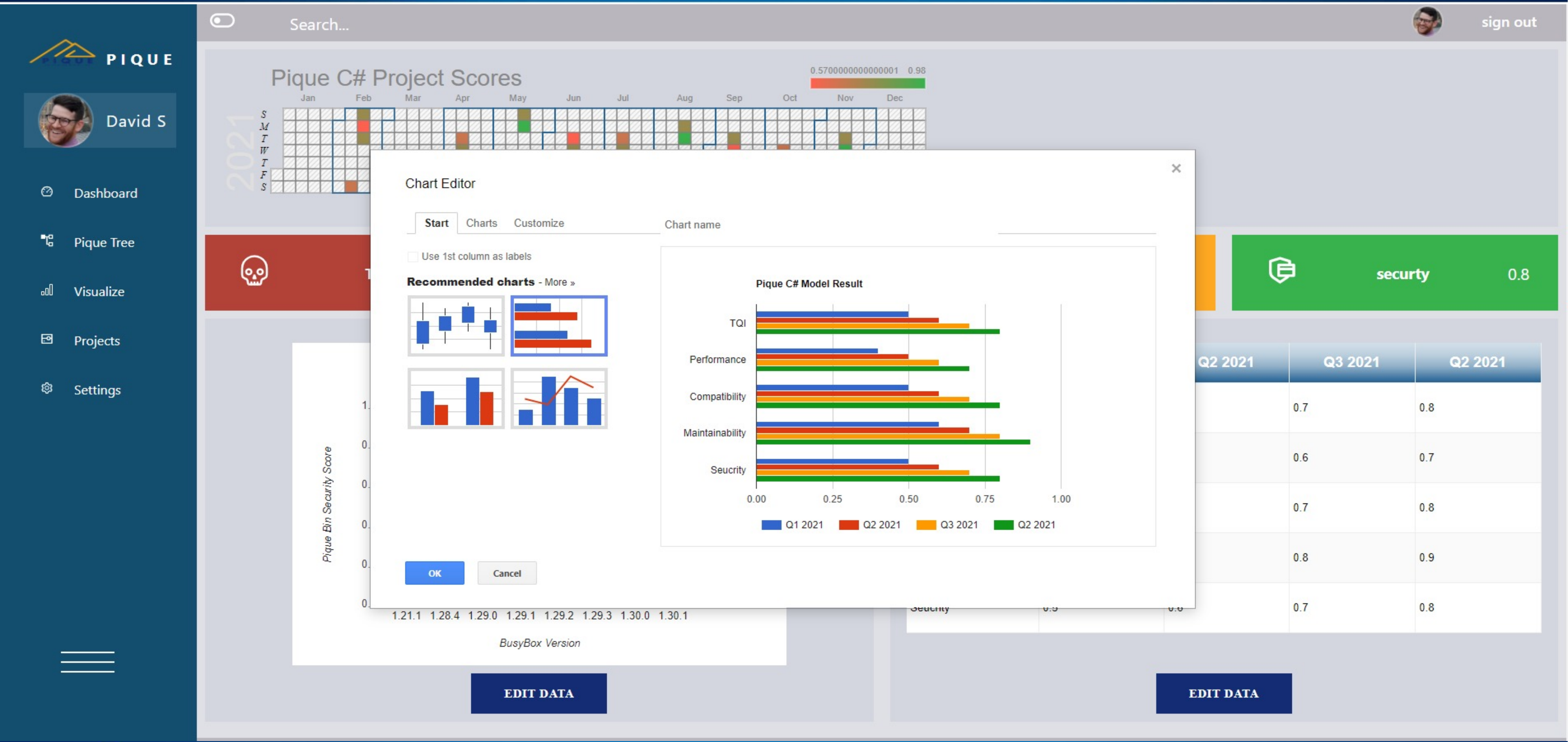
securty 0.8



EDIT DATA

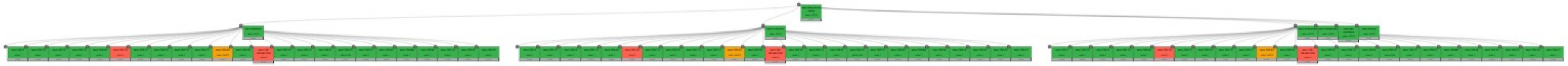
Score	Q1 2021	Q2 2021	Q3 2021	Q2 2021
TQI	0.5	0.6	0.7	0.8
Performance	0.4	0.5	0.6	0.7
Compatibility	0.5	0.6	0.7	0.8
Maintainability	0.6	0.7	0.8	0.9
Seucrity	0.5	0.6	0.7	0.8

EDIT DATA

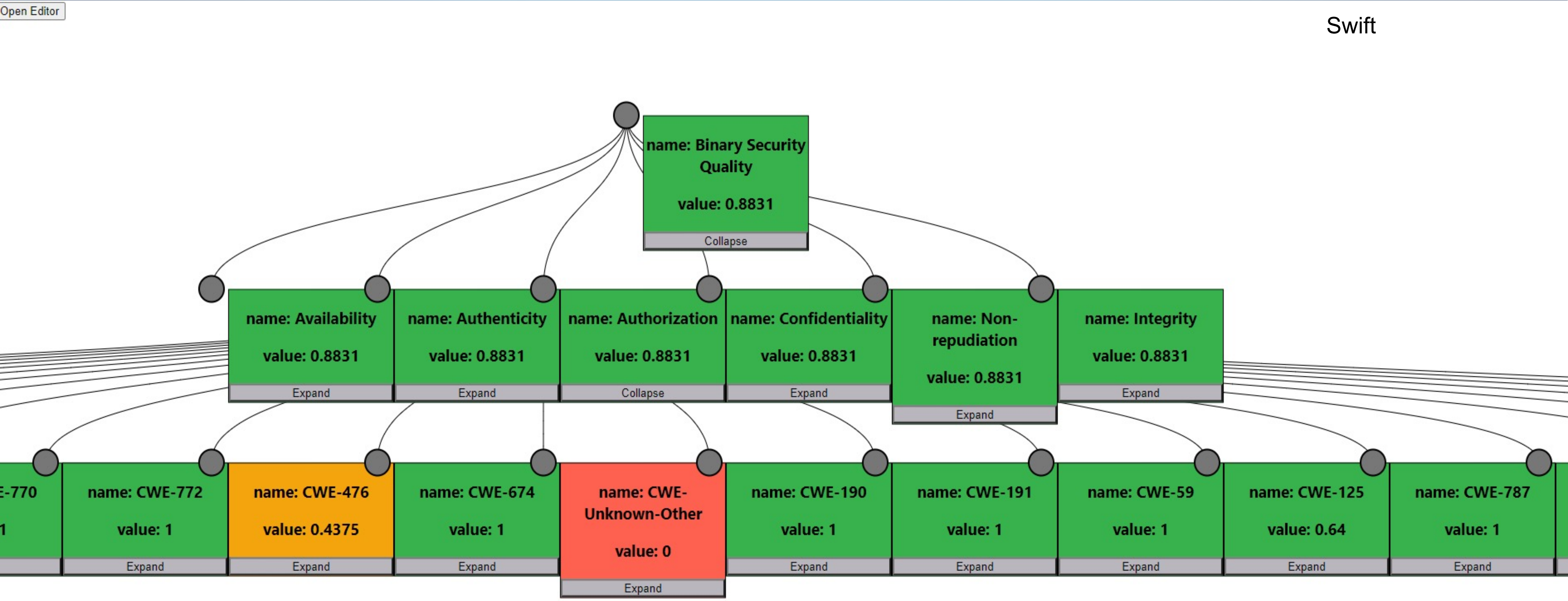

Search...
sign out
PIQUE
David S
Dashboard
Pique Tree
Visualize
Projects
Settings
Pique C# Project Scores
0.5700000000000001 0.98
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2021
securty 0.8
Chart Editor
Start Charts Customize
Chart name
Use 1st column as labels
Recommended charts - More »
Pique C# Model Result
TQI
Performance
Compatibility
Maintainability
Seucrity
0.00 0.25 0.50 0.75 1.00
Q1 2021 Q2 2021 Q3 2021 Q2 2021
OK
Cancel
Pique Bin Security Score
1.21.1 1.28.4 1.29.0 1.29.1 1.29.2 1.29.3 1.30.0 1.30.1
BusyBox Version
Q2 2021 Q3 2021 Q2 2021
0.7 0.8
0.6 0.7
0.7 0.8
0.8 0.9
0.7 0.8
EDIT DATA
EDIT DATA

Open Editor

Swift



Swift



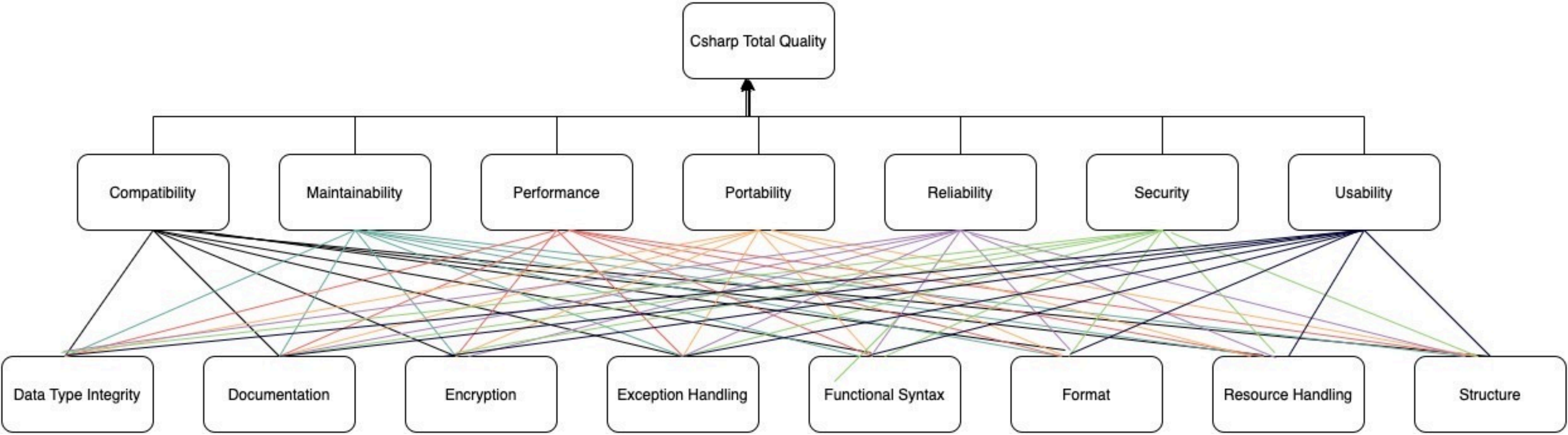
Open Editor
name: Binary Security Quality
value: 0.8831
Collapse
name: Availability
value: 0.8831
Expand
name: Authenticity
value: 0.8831
Expand
name: Authorization
value: 0.8831
Collapse
name: Confidentiality
value: 0.8831
Expand
name: Non-repudiation
value: 0.8831
Expand
name: Integrity
value: 0.8831
Expand
name: CWE-772
value: 1
Expand
name: CWE-476
value: 0.4375
Expand
name: CWE-674
value: 1
Expand
name: CWE-Unknown-Other
value: 0
Expand
name: CWE-190
value: 1
Expand
name: CWE-191
value: 1
Expand
name: CWE-59
value: 1
Expand
name: CWE-125
value: 0.64
Expand
name: CWE-787
value: 1
Expand

PIQUE Models

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

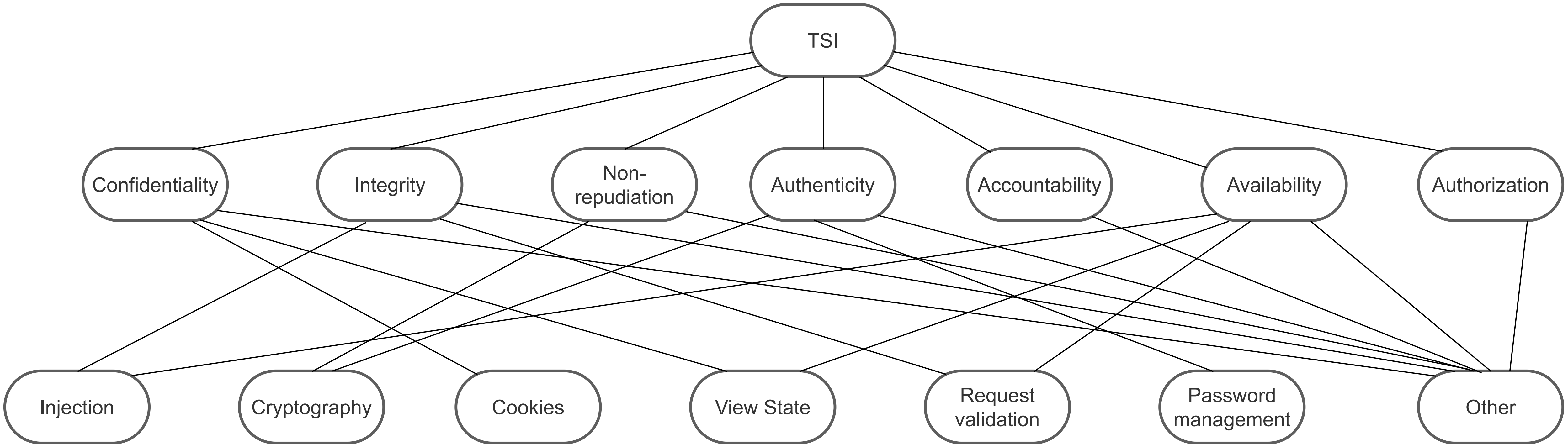
Pique-C#

Rice and Swift



Pique-C#-Sec

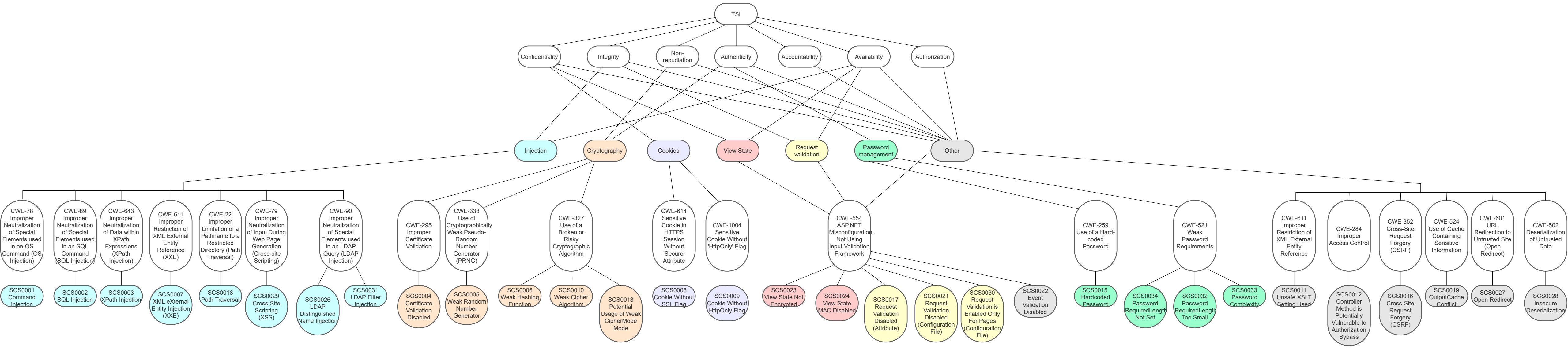
Harrison



Pique-C#-Sec

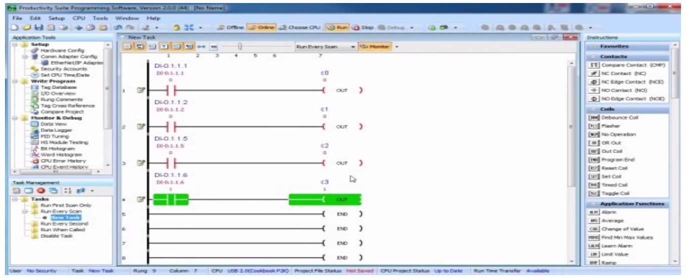
Harrison

PIQUE-C#-Sec Model



Quality assurance model enhancement for binary executables

LaMeres et al.

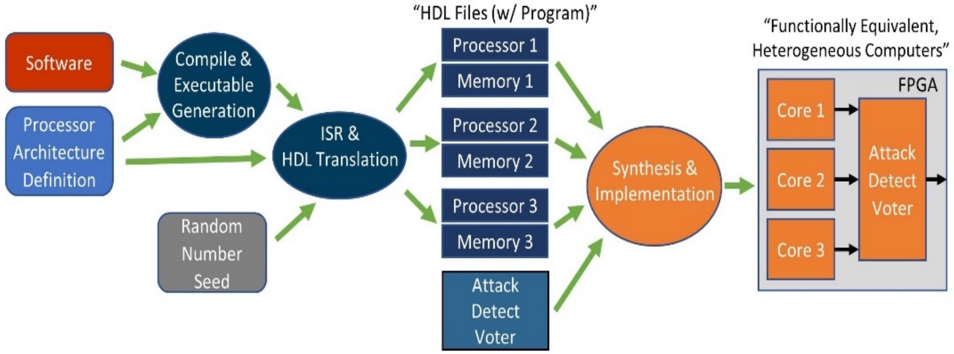


PLC Aim 1: Monitor binary execution in real-time from a PLC.

- Can we feed real-time binaries from PLC into QA model?
- Can we detect execution patterns in real time under attack?

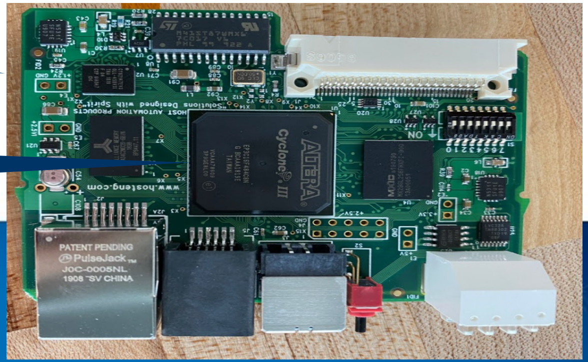
PLC Aim 2: Replace PLC processor w/ MSU's heterogenous cores.

- Can we insert binaries into the MSU malware core using PLC development environment.
- Can we detect/defeat injected malware.

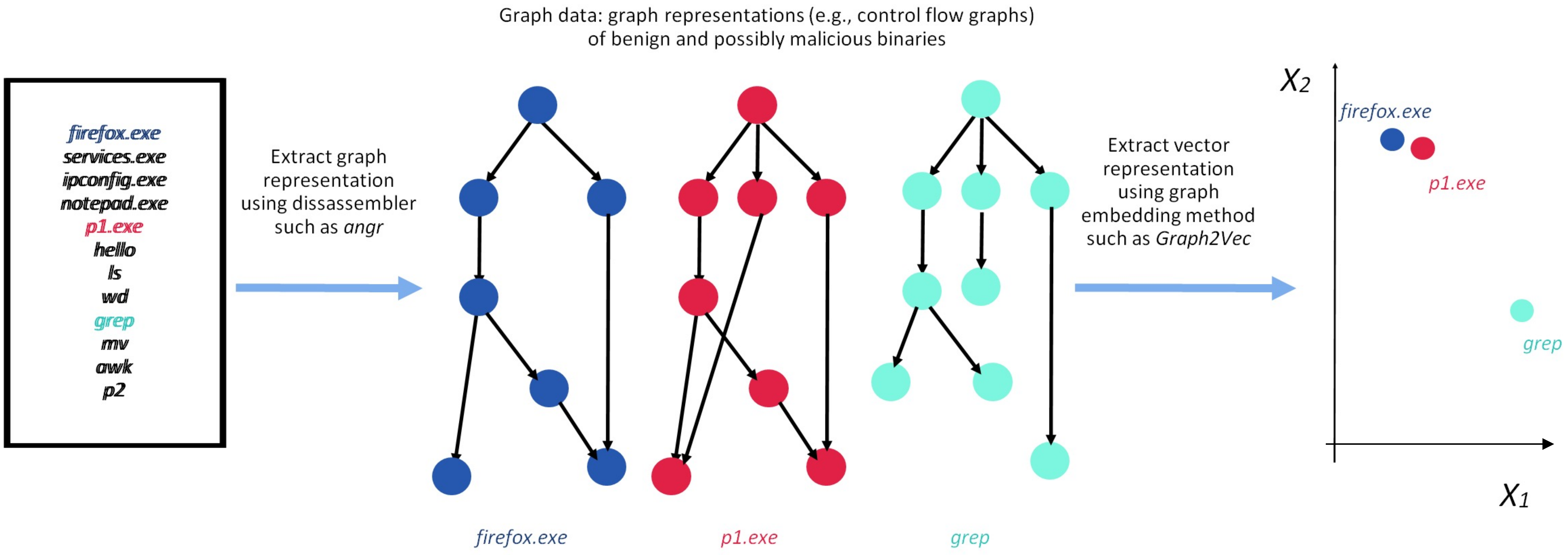


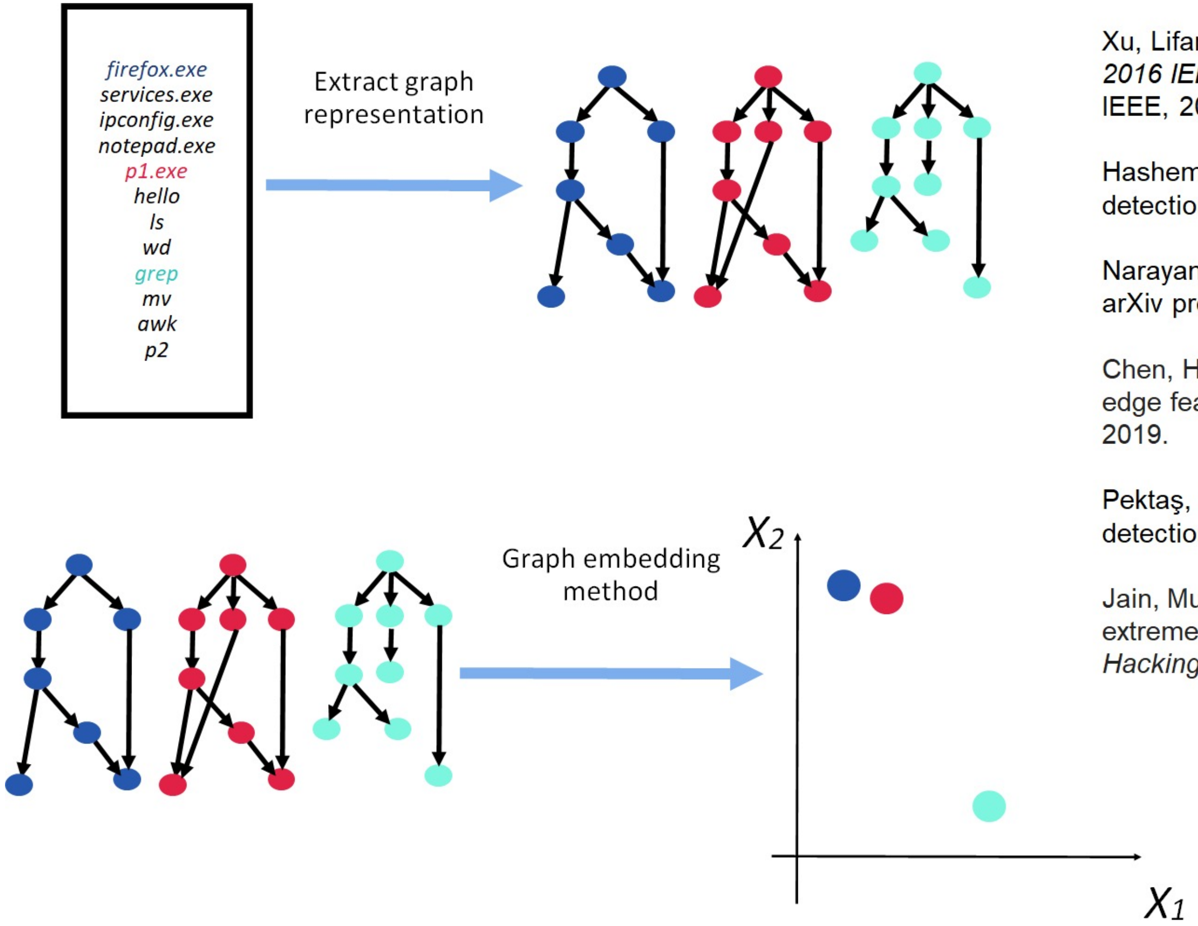
We have acquired an *AutomationDirect* PLC starter kit.

We discovered the *AutomationDirect* PLC processor is implemented on an Altera Cyclone FPGA.



Classification, clustering, and anomaly detection using graph representations of code





Xu, Lifan, et al. "Dynamic android malware classification using graph-based representations." *2016 IEEE 3rd international conference on cyber security and cloud computing (CSCloud)*. IEEE, 2016.

Hashemi, Hashem, et al. "Graph embedding as a new approach for unknown malware detection." *Journal of Computer Virology and Hacking Techniques.* 2017

Narayanan, Annamalai, et al. "graph2vec: Learning Distributed Representations of Graphs." arXiv preprint arXiv:1707.05005. 2017

Chen, Hong, and Hisashi Koga. "Gl2vec: Graph embedding enriched by line graphs with edge features." *International Conference on Neural Information Processing*. Springer, Cham, 2019.

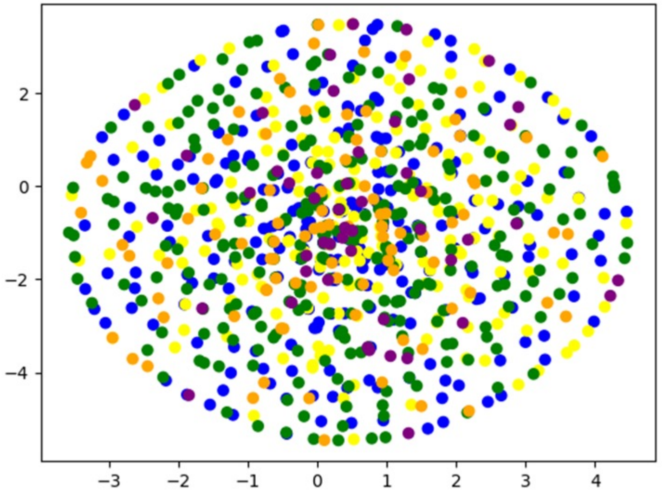
Pektaş, Abdurrahman, and Tankut Acarman. "Deep learning for effective Android malware detection using API call graph embeddings." Soft Computing 24.2 (2020): 1027-1043.

Jain, Mugdha, William Andreopoulos, and Mark Stamp. "Convolutional neural networks and extreme learning machines for malware classification." *Journal of Computer Virology and Hacking Techniques* 16.3 (2020): 229-244.

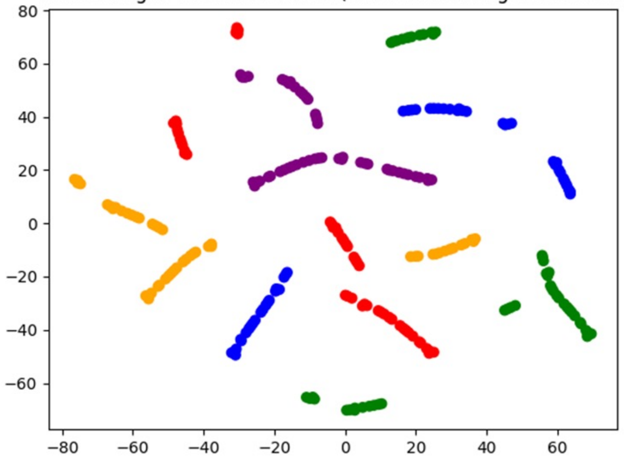
Clustering Results: embedding method comparison

Preliminary results suggest Graph2Vec embedding preserves similarity of various classes of code

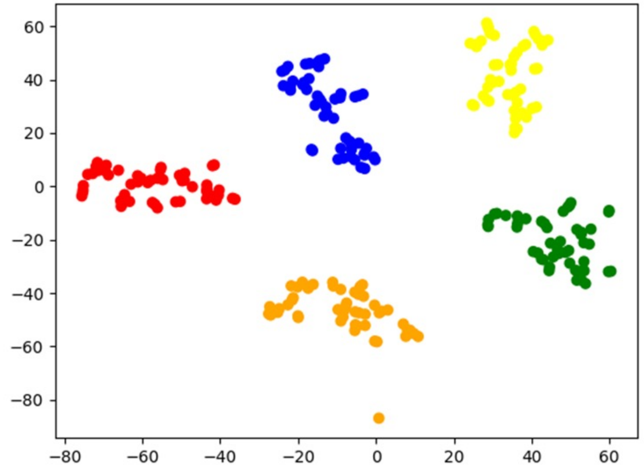
t-SNE visualization of the Agglomerative clustering of the LDP embedding of the DBC dataset (4 histogram bins)



t-SNE visualization of the Agglomerative clustering of the GL2Vec embedding of the DBC dataset (1024 embedding dimensions)



t-SNE visualization of the DBSCAN clustering of the Graph2Vec embedding of the DBC dataset (64 embedding dimensions)



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

<u>Elements to investigate:</u>

- *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

Improving the Software Assurance Ecosystem

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

Tool Provider	Open Source	Intermediate Language	Architectures
Manticore	Yes	Unknown	x86, ARM
@Disco (INL)	Yes (with clearance)	Vex	Many
BAP (Binary Analysis Platform)	Yes	Lisp	x86, ARM, MIPS, PowerPC
BARF (Binary Analysis and Reverse Engineering Framework)	Yes	REIL	x86, ARM
FACT (Firmware Analysis and Comparison Tool)	Yes	Unknown	Unknown
Nano Spark (Hoplite Industries)	No	N/A	Many
Other proprietary (if needed)	No	Unknown	Unknown

Improving the Software Assurance Ecosystem

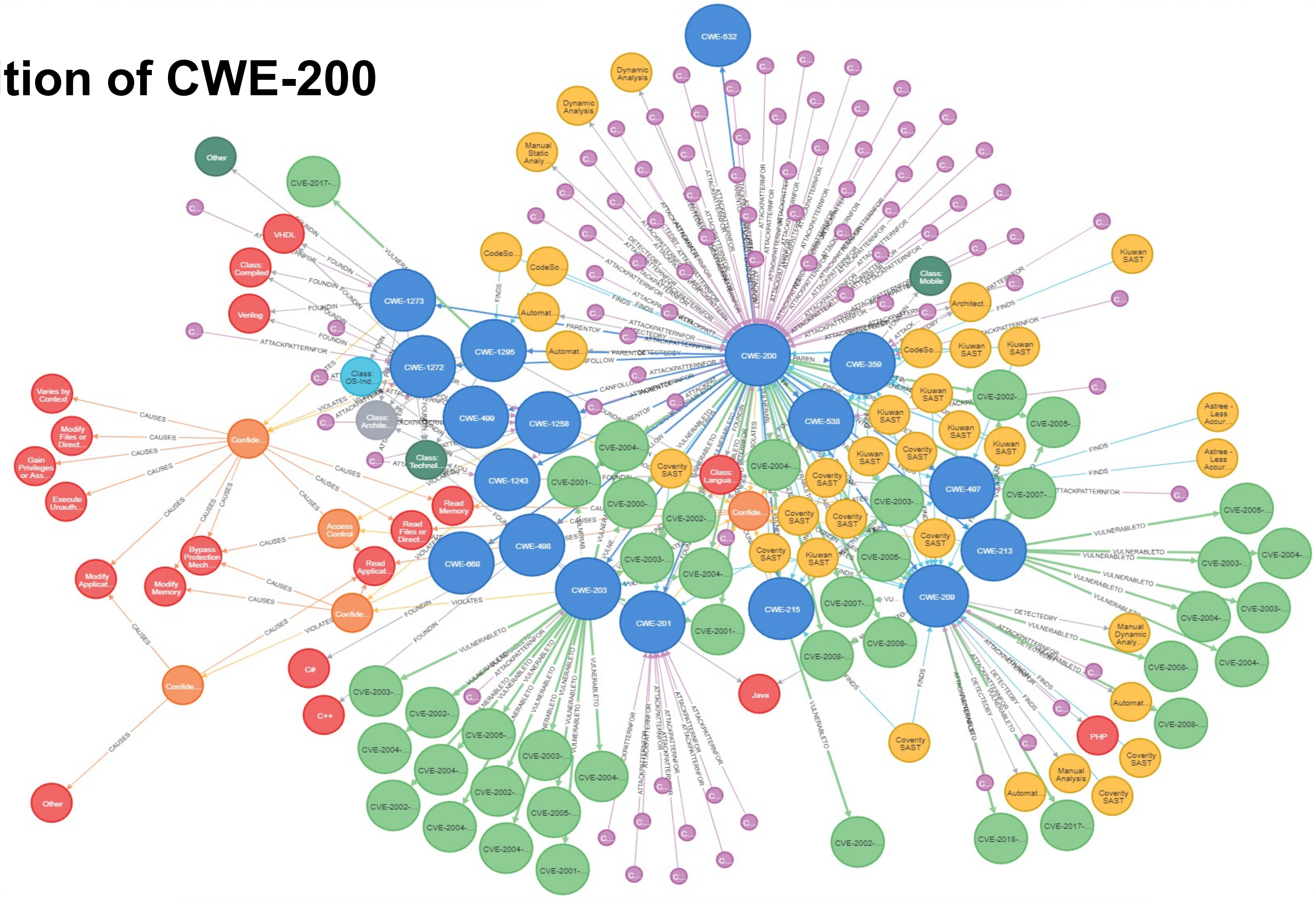
3. *Identify security zones and sensitive sections of source code*

Specifically:

- Investigate operationalization of the CWE-200[1] *"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 INDUSTRIES

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 integration center | INL Idaho National Laboratory

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.

MONTANA STATE UNIVERSITY APPLIED RESEARCH LABORATORY

Carnegie Mellon University Software Engineering Institute

WASHINGTON STATE UNIVERSITY

ROCHESTER INSTITUTE OF TECHNOLOGY R·I·T 1829

MONTANA
STATE UNIVERSITY
APPLIED RESEARCH LABORATORY

MSU's Applied Research Laboratory

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

MONTANA STATE UNIVERSITY

Applied Research Laboratory

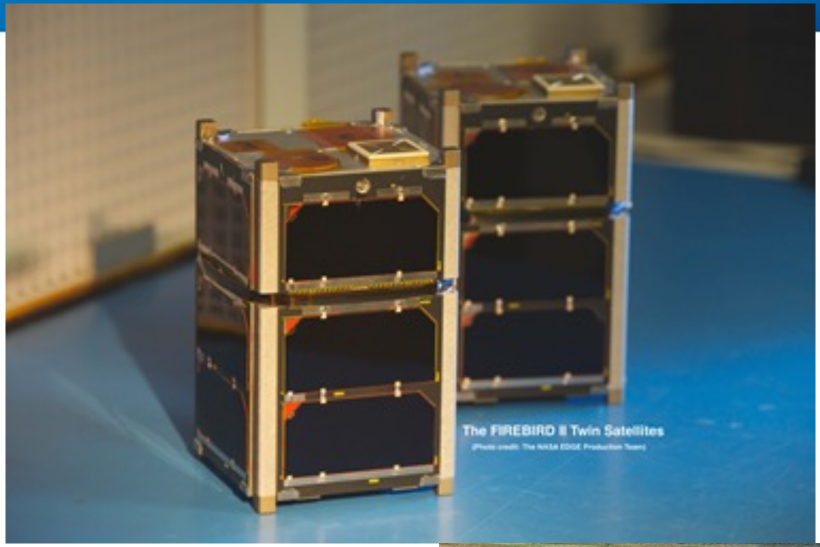
ARL FEATURES

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

Questions & Open Discussion